Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Шевчук В.О.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____Мелешко Є.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
« _6_ » _вересня_ 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Шевчука Владислава Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності*

2. Керівник роботи *Мелешко Єлизавета Владиславівна, доктор техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти № 19-13 від 07.08.2024 року

3. Строк подання роботи до захисту _*2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи *Дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності з використанням мови програмування Dart*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Дані про економічну ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Міжсторінкова структурна схема додатку	*1 аркуш*
Структурна схема таблиць бази даних	*1 аркуш*
Функціональна схема роботи архітектури представлення BLoC з використанням альтернативного пакету cubit.	*1 аркуш*
Діаграма процесів додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.	*1 аркуш*
Частина блок-схеми основної програми – авторизації, реєстрації користувачів та переліку днів.	*1 аркуш*

Частина блок-схеми основної програми – алгоритми створення та управління задачами. *1 аркуш*

Маркетингове та економічне обґрунтування ІТ-проєкту *1 аркуш*

7. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	25.10.2024	10.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	03.11.2024	21.11.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	16.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	10.12.2024 р.	

Дата видачі завдання
«_10_»__вересня___2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«_16_»_вересня__2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Шевчук В.О. Дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній магістерській роботі розроблено мобільний додаток для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.

Метою роботи є дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності з використанням мови програмування Dart.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

1. Огляд існуючих мобільних додатків, в основі яких лежить використання методик продуктивності та тайм менеджменту, аналіз їх інтерфейсу, функціональності та ефективності.

2. Визначення основних вимог до розробки мобільного додатка для управління власними задачами та вибір оптимальних інструментів для втілення поставленої мети.

3. Програмна реалізація додатка з методиками продуктивності, що ґрунтується на матриці Ейзенхауера з використанням мови програмування Dart та розробка серверної частини на мові програмування Python.

4. Імплементація клієнт-серверної архітектури REST-API.

5. Оцінка результатів дослідження та розробленого мобільного додатка, визначення його переваг та обмежень.

Об'єктом дослідження є процес автоматизації методик продуктивності за допомогою мобільних додатків.

Предметом дослідження є методи та алгоритми інтеграції сучасних методик продуктивності у мобільний додаток.

Методи дослідження: аналіз наукових і технічних статей, книг та інших матеріалів, що стосуються теми; систематизація літератури та наукових джерел; методи автоматизованого проєктування (моделювання системи, проєктування інтерфейсу користувача, тестування створеного мобільного додатка для оцінки його відповідності поставленим цілям); методи розробки програмного забезпечення (спіральна модель); методи розробки мобільних додатків (кросплатформна розробка).

Результат роботи – програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на Емуляторах та телефонах з операційними системами iOS та Android.

Програму розроблено на мові програмування Dart з використанням фреймворку Flutter.

Ключові слова: комп’ютерна інженерія, мобільний додаток, методики продуктивності, матриця Ейзенхауера

ABSTRACT

Shevchuk V.O. Research and software implementation of a mobile application for managing tasks based on the Eisenhower matrix and modern productivity techniques. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi 2024.

In this master's thesis, a mobile application for task management based on the Eisenhower Matrix and modern productivity techniques was developed.

The purpose of the work is the research and software implementation of a task management application based on the Eisenhower Matrix and modern productivity techniques using the Dart programming language.

To achieve this purpose, a research program was defined, consisting of the following tasks:

1. Overview of existing mobile applications based on the use of productivity and time management techniques, analysis of their interface, functionality and efficiency.

2. Defining the basic requirements for the development of a mobile application for managing own tasks and choosing the optimal tools for implementing the goal.

3. Software implementation of an application using productivity techniques based on the Eisenhower Matrix using the Dart programming language and development of the server-side component in the Python programming language.

4. Implementation of a client-server REST-API architecture.

5. Evaluation of the results of the study and the developed mobile application, determination of its advantages and limitations.

The object of the research is the process of automating productivity techniques through mobile applications.

The subject of the research is methods and algorithms for integrating modern productivity techniques into a mobile application.

Research methods: analysis of scientific and technical articles, books and other materials related to the topic; systematization of literature and scientific

sources; automated design methods (system modeling, user interface projection, testing of the created mobile application to assess its compliance with the set goals); software development methods (spiral model); mobile application development methods (cross-platform development).

The result of the work is the software implementation of a mobile task management application based on the Eisenhower Matrix and modern productivity techniques.

During the development of the software model, an analysis of existing hardware and software tools was performed.

All components of the software developed are fully described. User-friendly user interface is developed. These are instructions for working with software.

The application can be used on emulators and phones with iOS and Android operating systems.

The program was developed using the Dart programming language and the Flutter.

Keywords: computer engineering, mobile application, productivity techniques, Eisenhower Matrix

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

Flutter – фреймворк для розробки мобільних додатків.

ООП – об'єктно-орієнтоване програмування.

BLoC – архітектурний патерн.

Dart – однопоточна мова програмування.

REST API – архітектурний стиль взаємодії між різними системами через інтернет або мережу.

ВСТУП

Актуальність теми. У сучасному світі, де темпи життя постійно зростають, ефективне управління часом стає ключовим фактором успіху як для окремих людей, так і для організацій. Мобільні додатки, що допомагають оптимізувати робочий процес, відіграють важливу роль у підвищенні продуктивності, дозволяючи користувачам ефективно розподіляти завдання та керувати своїми справами. Однією з найбільш відомих та перевірених методик управління завданнями є матриця Ейзенхауера, яка базується на принципах пріоритизації справ за критеріями важливості та терміновості. Поєднання цієї методики з сучасними підходами до підвищення продуктивності відкриває нові можливості для розробки інструментів, що покращують управління часом та ресурсами.

У міру того, як зростає роль технологій у повсякденному житті людини, з'являється потреба у вирішені широкого спектру завдань у найменш енергозатратний спосіб. Одним з універсальних інструментів, призначених для цього, стають мобільні додатки спрямовані на оптимізацію планування особистих справ. Попри велику кількість додатків для управління задачами, більшість із них не враховує важливість індивідуальних підходів до планування та продуктивності, зокрема таких, які спираються на психологічні та поведінкові особливості користувачів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності з використанням мови програмування Dart.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

1. Огляд існуючих мобільних додатків, в основі яких лежить використання методик продуктивності та тайм менеджменту, аналіз їх інтерфейсу, функціональності та ефективності.

2. Визначення основних вимог до розробки мобільного додатка для управління власними задачами та вибір оптимальних інструментів для втілення поставленої мети.

3. Програмна реалізація додатка з методиками продуктивності, що ґрунтується на матриці Ейзенхауера з використання мови програмування Dart та розробка серверної частини на мові програмування Python.

4. Імплементація клієнт-серверної архітектури REST-API.

5. Оцінка результатів дослідження та розробленого мобільного додатка, визначення його переваг та обмежень.

Об'єктом дослідження є процес автоматизації методик продуктивності за допомогою мобільних додатків.

Предметом дослідження є процес автоматизації методик продуктивності за допомогою мобільних додатків.

Методи дослідження. Для досягнення мети магістерської роботи та вирішення поставлених завдань обрано такі методи: аналіз наукових і технічних статей, книг та інших матеріалів, що стосуються теми; систематизація літератури та наукових джерел; методи автоматизованого проєктування (моделювання системи, проєктування інтерфейсу користувача, тестування створеного мобільного додатка для оцінки його відповідності поставленим цілям); методи розробки програмного забезпечення (спіральна модель); методи розробки мобільних додатків (кросплатформна розробка).

Наукова новизна отриманих результатів. У процесі вирішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод автоматизації методик продуктивності 90/30, Pomodoro та матриці Ейзенхауера.

2. Розроблено вітчизняний мобільний додаток для менеджменту задач за допомогою методик продуктивності, який має більш ширший функціонал та більш адаптивний інтерфейс у порівнянні з існуючими аналогами та спрямований на потреби сучасного користувача у вирішені широкого спектру завдань у найбільш продуктивний та ефективний спосіб.

Практична цінність отриманих результатів полягає в тому, що **розроблено** додаток**, який** дозволяє автоматизувати методики продуктивності.

Достовірність наукових результатів підтверджена теоретичними обґрунтуваннями, результатами тестування розробленого програмного забезпечення, а також відповідністю отриманих результатів окремим висновкам, представленим у науковій літературі.

Таким чином, виходячи з вищеперерахованого, робота спрямована на вирішення актуальної проблеми підвищення продуктивності через інноваційне поєднання традиційних підходів (матриця Ейзенхауера) з новітніми технологічними рішеннями, що відповідають потребам сучасного користувача в швидкому та зручному управлінні своїм часом і завданнями.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення мобільного додатку

Призначенням мобільного додатка є надання користувачам інструменту для ефективного управління своїми завданнями та часом, заснованого на матриці Ейзенхауера та сучасних методиках підвищення продуктивності. Додаток допомагає пріоритизувати завдання за критеріями важливості та терміновості, забезпечуючи структуроване планування, яке сприяє виконанню насамперед найбільш важливих справ. Крім того, у додатку інтегровано різні методики продуктивності, такі як техніка Pomodoro або GTD, що дає можливість індивідуально налаштовувати робочий процес, враховуючи особливості користувача, підвищуючи ефективність і контроль над часом.

Функціональні вимоги для такого додатка включають реєстрацію та авторизацію за електронною поштою, створення, редагування, сортування та процеси виконання задач користувача. Створення графіків для відслідковування продуктивності, збереження задач в реляційній базі даних та редагування даних користувача.

Також мобільний додаток має функцію локалізації, яка дозволяє обирати мову інтерфейсу. Додаток підтримує як англійську, так і українську мови, що робить його більш зручним і доступним для користувачів з різних країн та регіонів. Завдяки вибору мови інтерфейсу користувачі зможуть легко розуміти всі функції та можливості додатка, що значно покращить їхній досвід взаємодії з програмою.

Крім того, додаток буде інтегровано з календарем та іншими системами керування часом, що дозволить синхронізувати свої завдання з іншими важливими подіями чи проєктами. Користувач також зможе відслідковувати статус виконання задач через загальний статус дня, переглядати прогрес за

допомогою візуальних індикаторів, що допоможе краще контролювати власну продуктивність та відслідковувати динаміку досягнень.

1.2 Область застосування

Мобільні додатки для продуктивності та управління часом широко використовуються в сучасному світі як основний інструмент для організації особистих і професійних завдань. Вони стали важливим елементом у сфері ефективного управління часом, планування, а також підвищення продуктивності як для окремих осіб, так і для компаній.

У сфері бізнесу, мобільні додатки для продуктивності та тайм-менеджменту допомагають користувачам ефективно організовувати свій робочий день, розподіляти завдання за пріоритетом та управляти проєктами. Вони дозволяють керівникам і працівникам стежити за дедлайнами, розподіляти ресурси та аналізувати прогрес. Завдяки цим додаткам можна легко контролювати виконання завдань, призначати їх колегам і ефективно керувати часом, що суттєво підвищує продуктивність робочих процесів.

У сфері особистого розвитку, мобільні додатки для продуктивності дозволяють користувачам планувати свій день, встановлювати цілі та відстежувати їх виконання. Вони також можуть включати різні методики тайм-менеджменту, такі як техніка Pomodoro, метод GTD (Getting Things Done) або матриця Ейзенхауера, що допомагає користувачам уникати прокрастинації. Завдяки цим додаткам користувачі мають можливість ставити особисті цілі та досягати їх у зручний для них спосіб.

Крім того, мобільні додатки для продуктивності часто використовуються для сімейного чи групового планування, де кілька людей можуть синхронізувати свої графіки та завдання. Такий функціонал може бути корисним для організації спільних подій, проєктів або навіть побутових справ, що сприяє більш злагодженій командній роботі та гармонії в особистих відносинах.

Мобільні додатки для управління завданнями стали необхідним інструментом для багатьох компаній, організацій та індивідуальних користувачів. Вони допомагають відстежувати виконання проєктів, організовувати віддалену роботу, а також інтегруються з іншими інструментами для спільної роботи, такими як календарі або системи управління проєктами. Крім того, додатки дозволяють ефективно відстежувати витрати часу, що може впливати на вдосконалення робочих процесів та підвищення продуктивності компаній.

Усі ці функції роблять мобільні додатки для продуктивності та тайм-менеджменту важливим інструментом в організації часу та завдань у сучасному світі, допомагаючи користувачам досягати своїх цілей та підвищувати ефективність роботи та життя в цілому.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми магістерської роботи

На основі фреймворка Flutter було розроблено кілька популярних додатків для менеджменту. Розглянемо їх детальніше:

Reflectly – це додаток для ведення щоденника, який допомагає користувачам відстежувати свої емоції та щоденні події. Він дозволяє вести записи і аналізувати свій стан та є корисним для особистого менеджменту.

Super Productivity – це додаток для управління завданнями, який дозволяє планувати, відстежувати виконання задач і управляти своїм часом. Він має функції тайм-трекінгу та інтеграцію з іншими сервісами.

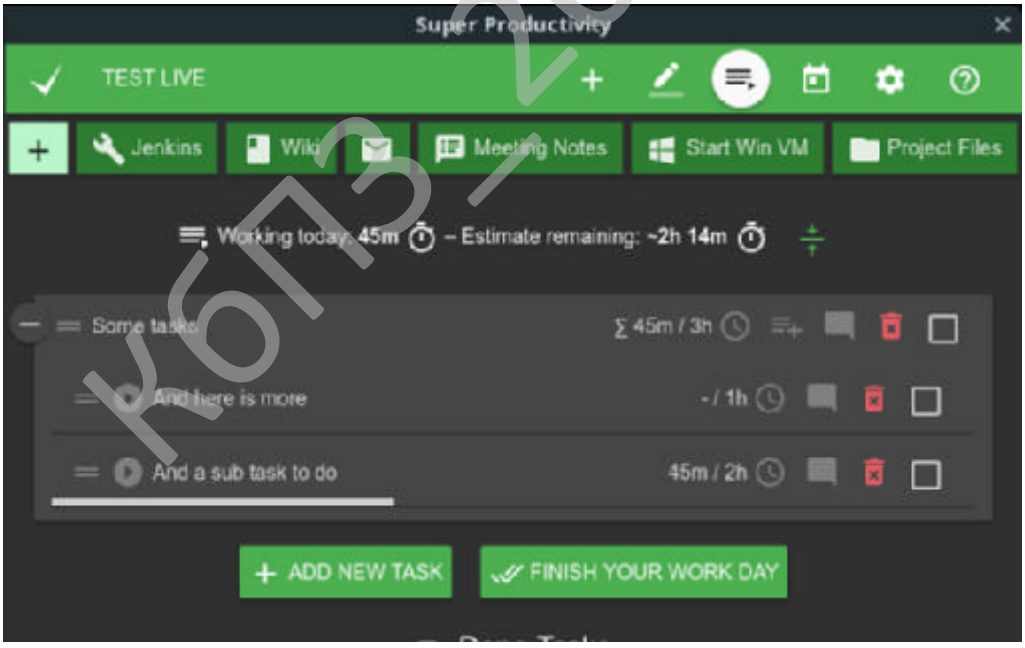

Рисунок 2.1 – Інтерфейс програми Super Productivity

Todoist, хоча основна версія цього додатка не була створена на Flutter, надихнула багатьох розробників на створення проєктів для управління задачами на основі Flutter. Завдяки широким можливостям Flutter, такі додатки дозволяють

користувачам не лише легко управляти своїми списками справ, але й пропонують адаптивний інтерфейс, підтримку на різних платформах (Android, iOS, Web), та інтеграцію з іншими сервісами. Крім того, використання Flutter сприяє швидшому розвитку додатків, зменшенню витрат на підтримку коду та розширенню функціоналу, такого як нагадування, пріоритизація задач і аналітика продуктивності.

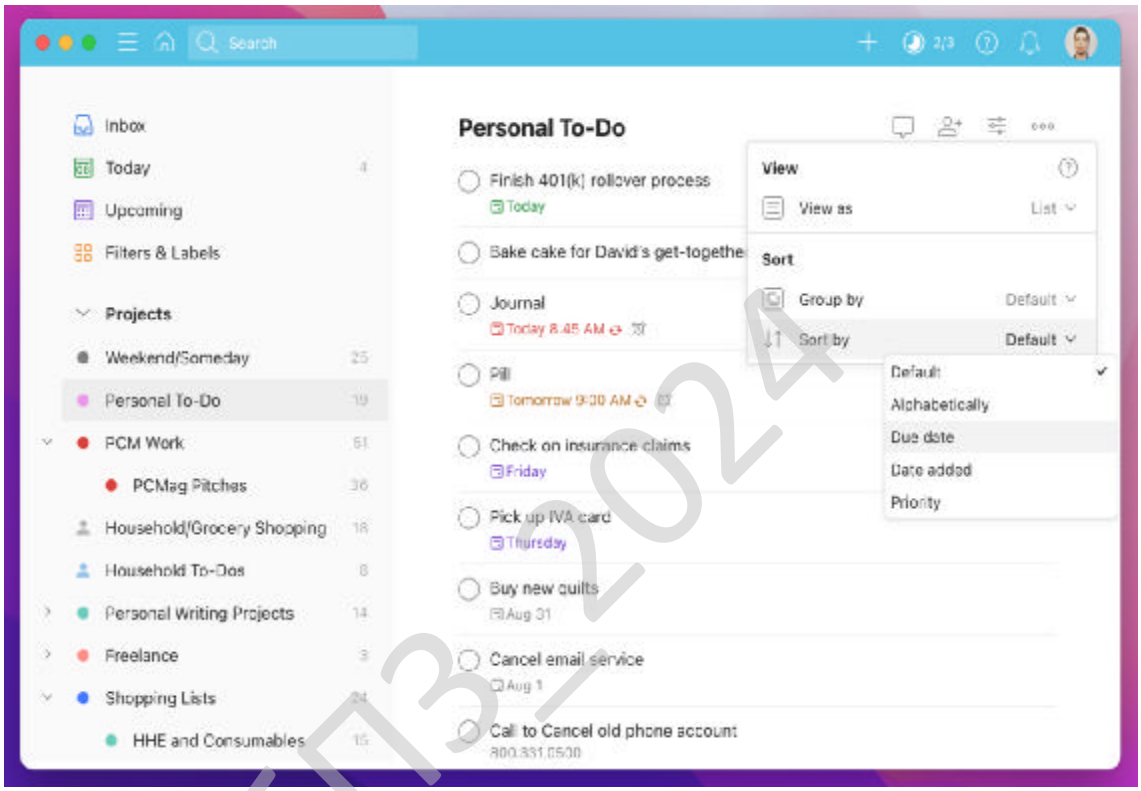

Рисунок 2.2 – Інтерфейс програми Todoist

Habit Tracker – це додаток, який дозволяє користувачам відстежувати свої звички і досягнення, що є важливим елементом особистого менеджменту та саморозвитку. Він допомагає перетворювати рутинні дії на систематичні звички, що є ключем до особистісного та професійного зростання. Використовуючи додаток, люди можуть покращувати своє здоров'я, продуктивність і загальну якість життя.

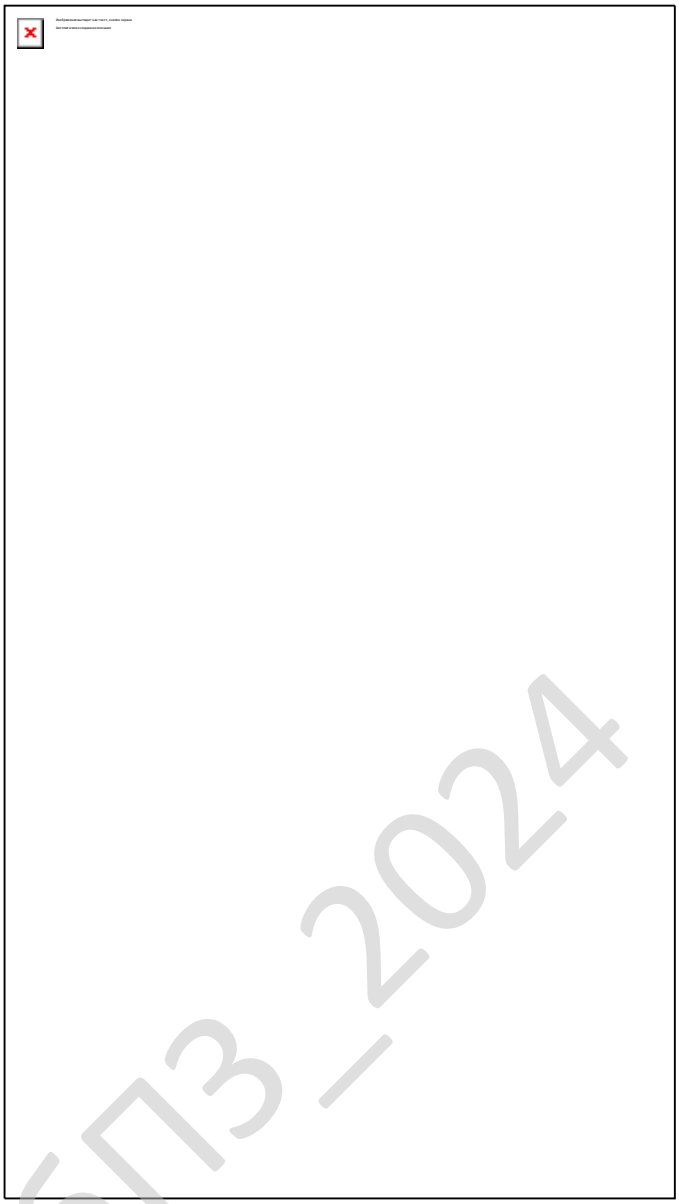

Рисунок 2.3 – Інтерфейс програми Habit Tracker

KTimer – це додаток для управління часом, який використовує техніку Pomodoro для покращення продуктивності. Користувачі можуть встановлювати таймери для роботи та відпочинку.

Any.do – це популярний додаток для управління завданнями та списками справ, який допомагає користувачам легко створювати нагадування, планувати проєкти та організовувати свій день. Хоча цей додаток не створений виключно на Flutter, він надихає багатьох розробників своїм інтуїтивним інтерфейсом та гнучкими функціями.

Taskist – це додаток для керування завданнями, створений на основі

Flutter, який дозволяє користувачам планувати свої справи та дотримуватися пріоритетів. Простий і зрозумілий інтерфейс робить його популярним серед користувачів, для яких важливі комфорт та легкість у використанні.

Focus Keeper – додаток, створений для підтримки високої концентрації під час виконання завдань, заснований на техніці Pomodoro. Він дозволяє користувачам працювати в циклах, відслідковувати продуктивність і регулювати час відпочинку.

Nozbe – це інструмент для командної співпраці і управління проєктами, який пропонує можливість організувати робочий процес та ділитися завданнями між членами команди. Він забезпечує інтеграцію з календарями та іншими сервісами, що спрощує процес управління задачами.

Moo.do – це додаток для планування та управління завданнями, який комбінує можливості списків справ, календарів і записів, що робить його ідеальним для багатозадачності.

Ці додатки демонструють, як Flutter може бути використаний для створення інструментів управління, що допомагають користувачам організувати своє життя та роботу більш ефективно.

Оскільки додаток реалізований на об'єктно-орієнтованій мові програмування, варто розглянути абревіатуру принципів ООП – SOLID. Загальна мета всіх принципів SOLID полягає в поліпшенні масштабованості, підтримки та зрозумілості коду.

Принцип єдиного обов'язку (Single Responsibility Principle, SRP) вимагає, щоб кожен клас або модуль мав лише одну причину для зміни, тобто був відповідальним за одну конкретну функцію або аспект системи. Це спрощує підтримку, тестування, та розширення коду, оскільки зміни в одній частині системи не впливають на інші частини.

Принцип відкритості/закритості (Open/Closed Principle, OCP) є одним із ключових принципів SOLID і стверджує, що класи, модулі чи функції повинні бути відкритими для розширення, але закритими для змін. Це означає, що

поведінку системи можна змінити або розширити без модифікації існуючого коду.

Дотримання цього принципу знижує ризик внесення помилок у вже протестований код і сприяє гнучкості системи.

Принцип підстановки Лісков передбачає, що підкласи можуть бути використані замість своїх батьківських класів без зміни коректності роботи програми, що зменшує залежності між класами.

Принцип розділення інтерфейсу та реалізації, також відомий як принцип залежності від абстракцій (Dependency Inversion Principle, DIP), є одним із фундаментальних принципів SOLID. Він стверджує, що класи повинні залежати від абстракцій (інтерфейсів), а не від конкретних реалізацій. Це забезпечує більшу гнучкість і масштабованість коду, оскільки зміни в реалізаціях не впливають на споживачів (клієнтів) інтерфейсу.

Принцип інверсії залежностей (Dependency Inversion Principle, DIP) стверджує, що високорівневі модулі (ті, які визначають логіку або бізнес-правила) не повинні залежати від низькорівневих модулів (ті, які реалізують деталі). Обидві групи мають залежати від абстракцій (інтерфейсів або абстрактних класів).

Це сприяє створенню більш модульного, гнучкого та перевикористовуваного коду, а також полегшує тестування та впровадження нових функцій.

Застосування принципів SOLID допомагає зменшити залежності між класами, зробити код більш зрозумілим та гнучким, а також полегшити його тестування і підтримку.

Загалом Об’єктно-орієнтоване програмування базується на чотирьох ключових принципах: інкапсуляція, абстракція, успадкування та поліморфізм. Кожен з них відіграє важливу роль у створенні структурованого, масштабованого та підтримуваного коду.

Інкапсуляція - це один із фундаментальних принципів об’єктно-

орієнтованого програмування (ООП), який передбачає приховування внутрішньої реалізації об’єкта та надання доступу до нього лише через визначений інтерфейс.

Простіше кажучи, інкапсуляція дозволяє "заховати" деталі роботи об'єкта (дані та методи) і обмежити до них доступ, надаючи лише ті можливості, які потрібні для взаємодії з ним.

Поліморфізм - це один із основних принципів об’єктно-орієнтованого програмування (ООП), який дозволяє об’єктам різних класів реагувати на однакові методи по-різному. Це означає, що один інтерфейс може бути використаний для різних типів об’єктів, при цьому кожен тип може мати свою реалізацію цього інтерфейсу. В результаті, одна і та ж операція може виконуватись по-різному залежно від типу об’єкта.

Абстракція – це принцип об’єктно-орієнтованого програмування, який полягає в приховуванні деталей реалізації і наданні лише необхідної інформації для користувачів об’єкта. Абстракція дозволяє зосередитися на важливих аспектах об’єкта, а не на його внутрішніх деталях. Вона дає змогу працювати з об’єктами через їх загальний інтерфейс, не турбуючись про їх точну реалізацію.

Спадкування - це один із основних принципів об’єктно-орієнтованого програмування (ООП), який дозволяє створювати нові класи на основі існуючих. Це означає, що новий клас (підклас або дочірній клас) може успадковувати властивості і методи іншого класу (батьківського класу). Спадкування дозволяє повторно використовувати код, розширювати функціональність та підтримувати ієрархії класів.

Також варто врахувати принцип KISS, що рекомендує тримати код простим і зрозумілим. Основна ідея полягає в тому, що прості рішення зазвичай кращі за складні, оскільки складний код може містити більше можливостей для помилок. Застосування принципу KISS передбачає, що код має бути зрозумілим і легким для читання. Це підвищує продуктивність розробників та зменшує ймовірність помилок.

Принцип DRY акцентує увагу на тому, що кожен елемент у коді має бути

представлений лише в одному місці, що допомагає уникати дублювання коду та спрощує його підтримку.

YAGNI (You Aren't Gonna Need It) - це принцип розробки програмного забезпечення, який наголошує, що не слід реалізовувати функціонал, який не потрібен прямо зараз. Він є важливим елементом методологій Agile та Extreme Programming (XP), спрямованих на створення простого, функціонального і легко підтримуваного коду.

Підхід APO (Agile Product Ownership) вимагає регулярних змін та оновлень функціональності відповідно до вимог користувачів і ринку. Він орієнтований на швидке створення цінності через короткі ітерації та постійну інтеграцію зворотного зв'язку, що дозволяє продукту залишатися актуальним.

BDUF (Big Design Up Front) - це підхід до розробки програмного забезпечення, який передбачає детальне планування та проєктування всієї системи на початковому етапі розробки, до того як розпочнеться реалізація коду. Цей підхід був широко поширений у традиційних (водоспадних) методологіях розробки.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації додатка було обрано мову програмування Dart - це сучасна, об'єктно-орієнтована мова програмування, створена компанією Google, яка призначена для вирішення завдань багатоплатформної розробки. Основна мета цієї мови - забезпечити високу продуктивність і ефективність, особливо у кросплатформних додатках. Dart широко використовується у фреймворку Flutter для розробки мобільних, веб- та десктопних додатків.

Однією з основних переваг Dart є його синтаксис, що нагадує мови, такі як Java, JavaScript або C++, завдяки чому він є зрозумілим для багатьох розробників і легко засвоюється. Dart дозволяє писати код, який компілюється як у машинний

код, так і в JavaScript для веб-додатків, що робить його універсальним інструментом для розробки додатків під різні платформи. Мова має вбудовану підтримку асинхронних операцій за допомогою ключових слів async і await, що полегшує роботу з асинхронними завданнями, такими як обробка запитів до серверів або взаємодія з базами даних.

Dart забезпечує високу продуктивність завдяки компіляції в машинний код, що робить додатки швидкими та ефективними на мобільних пристроях. Як об'єктно-орієнтована мова, Dart підтримує класи, об'єкти, наслідування та інші основні принципи ООП, що дозволяє будувати структуровані та масштабовані додатки. Особливо цінною є функція "гарячого перезапуску" (Hot Reload), яка дозволяє миттєво вносити зміни в код і бачити їхні результати без перезапуску програми, що значно прискорює процес розробки.

Dart також підтримує строгий контроль типів, що забезпечує стабільність коду на етапі компіляції. Завдяки цьому можна уникнути багатьох помилок ще до виконання програми. Мова дозволяє легко створювати додатки, які працюють на різних платформах, включаючи Android, iOS, веб, а також настільні операційні системи, такі як Windows, macOS та Linux.

Серед головних переваг Dart – швидкість і продуктивність, можливість створювати кросплатформні додатки, а також простота використання. Мова відзначається гнучкістю й адаптивністю, що дозволяє масштабувати додатки від невеликих проєктів до великих корпоративних рішень. Функція гарячого перезапуску значно підвищує зручність розробки, особливо для роботи з інтерфейсом користувача.

Завдяки інтеграції з Flutter, Dart став основним вибором для розробників, які створюють кросплатформні додатки. Це робить його особливо привабливим для мобільної розробки, але також дозволяє ефективно використовувати його у веб-додатках та навіть десктопних рішеннях. Dart – це потужний інструмент, що поєднує в собі високу продуктивність, кросплатформеність і простоту у використанні, що робить його важливим для сучасної розробки програмного

забезпечення.

Фреймворк для реалізації інтерфейсу додатка був обраний Flutter – це фреймворк з відкритим кодом, розроблений компанією Google для створення кросплатформних додатків. Flutter надає все необхідне для побудови візуально привабливих, продуктивних додатків з відмінною продуктивністю на всіх підтримуваних платформах.

Однією з головних переваг Flutter є його здатність забезпечувати нативну продуктивність додатків на Android та iOS завдяки використанню мови програмування Dart, яка компілюється в машинний код. Крім того, Flutter використовує власний графічний рушій, що дозволяє розробникам створювати чудові, високоякісні інтерфейси користувача з плавною анімацією та швидким відгуком. Це робить додатки, створені за допомогою Flutter, не лише красивими, але й швидкими.

Однією з найбільш цінних функцій Flutter є "гарячий перезапуск" (Hot Reload), що дозволяє розробникам миттєво бачити результати змін у коді без необхідності перезапуску додатка. Це значно прискорює процес розробки, особливо при роботі з інтерфейсом користувача, де візуальні зміни можуть бути відразу відображені на екрані. Ця функція є одним із ключових аспектів, які роблять Flutter привабливим для розробників, особливо у порівнянні з іншими кросплатформними інструментами.

Ще однією перевагою Flutter є можливість повного контролю над виглядом та поведінкою додатка, оскільки він не використовує нативні компоненти інтерфейсу, а замість цього рендерить інтерфейс самостійно. Це означає, що незалежно від платформи - будь то Android, iOS чи веб - додаток виглядатиме однаково, що забезпечує єдиний користувацький досвід на всіх пристроях.

Flutter також підтримує різноманітні бібліотеки та плагіни, що дозволяє інтегрувати різні функції – від геолокації до камер або доступу до баз даних. Спільнота розробників Flutter постійно зростає, і разом з нею зростає кількість

доступних інструментів, які полегшують розробку та підтримку додатків.

Ще одним важливим аспектом є кросплатформність Flutter. Завдяки цьому фреймворку можна створювати додатки для різних платформ - Android, iOS, веб та десктопні системи - використовуючи одну кодову базу. Це значно скорочує витрати на розробку та підтримку проєктів, оскільки немає необхідності писати окремий код для кожної платформи.

Крім того, Flutter забезпечує високу продуктивність додатків, оскільки Dart-компіляція в нативний код дозволяє додаткам працювати швидко і без затримок, що особливо важливо для мобільних і десктопних платформ. Додатки на Flutter також легко тестуються і масштабуються, що робить цей фреймворк оптимальним рішенням як для невеликих, так і для великих проєктів.

Завдяки своїм можливостям та простоті використання, Flutter стає одним з найпопулярніших інструментів для розробки кросплатформних додатків. Він пропонує не лише продуктивність, але й елегантність візуальних рішень, а також швидкість розробки, що робить його ідеальним вибором для створення сучасних додатків, що працюють на різних пристроях та платформах.

Для серверної частини був обраний Python – це мова програмування, яка завдяки своїй простоті, гнучкості та потужності, широко використовується для серверної розробки. Вона має багато інструментів та фреймворків, які дозволяють ефективно створювати серверні додатки різного рівня складності, від простих веб-сайтів до складних високонавантажених систем.

Однією з головних переваг Python для серверної розробки є його читабельний синтаксис і низький поріг входу. Це дозволяє швидко створювати код і полегшує його підтримку. Python також підтримує великий вибір бібліотек та фреймворків, які допомагають розробникам швидко реалізувати серверні рішення без необхідності писати все з нуля.

Також була обрана клієнт-серверна архітектура REST-API вона є популярним вибором для розробки з кількох причин:

1. Простота використання: REST використовує стандартні HTTP методи

(GET, POST, PUT, DELETE), що робить його зрозумілим та зручним для розробників.

2. Масштабованість: REST-API підтримує архітектуру клієнт-сервера, що дозволяє легко масштабувати систему. Клієнти та сервери можуть розвиватися незалежно один від одного.

3. Статусність: Кожен запит до REST-API є автономним, тобто сервер не зберігає стан між запитами. Це спрощує управління сесіями і підвищує надійність.

4. Універсальність: REST-API може бути використано з різними форматами даних (JSON, XML тощо), що робить його гнучким у різних сценаріях.

5. Кешування: REST-API підтримує кешування, що може покращити продуктивність та зменшити навантаження на сервер.

6. Легкість інтеграції: REST-API легко інтегрується з іншими сервісами та платформами, що робить його популярним вибором для мікросервісної архітектури.

7. Підтримка різних клієнтів: Завдяки універсальності протоколів, REST-API можна використовувати з різними типами клієнтів (веб-додатки, мобільні додатки тощо).

2.3 Розгорнута постановка завдання

У роботі було поставлено задачу розробити додаток з методиками продуктивності, що ґрунтується на матриці Ейзенхауера. Завдання буде розділено на декілька основних пунктів:

1. Розробка інтерфейсу (UI) додатка. Цей пункт виконано за допомогою віджетів, що містить Фреймворк Flutter. Для розробки сторінок використані основні віджети такі, як Stateless та Statefull віджети. Stateless віджети не зберігають свій стан. Вони є незмінними, тобто їхній вигляд не змінюється після

створення. Використовуються для статичних інтерфейсів, де віджет відображає дані, які не змінюються, або змінюються тільки ззовні.

Stateful віджети зберігають свій стан. Вони можуть змінювати свій вигляд у відповідь на зміни стану. Використовуються, коли потрібна взаємодія з користувачем або коли дані змінюються в процесі роботи програми. Також було застосовано InheritedWidget - це спеціальний тип віджету, який дозволяє ефективно передавати дані через дерево віджетів. Він використовується для управління станом і надає можливість підписки на зміни даних в дочірніх віджетах. InheritedWidget є потужним інструментом для управління станом і передачі даних у Flutter.

2. Розробка бізнес-логіки додатка за допомогою BLoC за пакетом cubit. BLoC – це архітектурний патерн, який часто використовується в Flutter для відокремлення бізнес-логіки від UI. Основна ідея полягає в тому, щоб створити чистий, тестований і підтримуваний код, що покращує організацію додатка.

3. Розробка серверу та реаляційної (SQL) бази даних для збереження даних про користувачів та їх задачі за допомогою мови програмування Python та архітектури REST-API.

SQL (Structured Query Language) - це стандартна мова для роботи з реляційними базами даних (РСУБД). Вона використовується для створення, читання, оновлення та видалення даних у базі даних, а також для управління самою структурою бази даних.

Реляційна база даних складається з таблиць, які містять дані у вигляді рядків і стовпців. Кожна таблиця має свою схему і є зв'язана з іншими таблицями через **зовнішні ключі**.

4. Підготовка Емуляторів iOS та Android для запуску додатка. Емулятори iOS та Android використовуються для тестування, відлагодження та запуску мобільних додатків без потреби у фізичних пристроях. Вони є важливим елементом процесу розробки, оскільки забезпечують швидкий і зручний спосіб перевірки додатка у різних умовах.

Крім реалізації зазначеної функціональності, важливим аспектом розробки програмного забезпечення є тестування. Для забезпечення якості продукту рекомендується виконувати різні види тестування, такі як модульне тестування та тестування з реальними користувачами. Це дозволяє виявити та виправити помилки і недоліки, що можуть виникнути під час розробки, і забезпечує високий рівень якості продукту.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування мобільного додатку

Функціонування мобільного додатка ґрунтується на кількох ключових аспектах. Основним із них є архітектура BLoC, яка базується на потоках.

BLoC (Business Logic Component) – це архітектурний патерн, що використовується у Flutter для створення масштабованих і повторно використовуваних мобільних додатків. Його основна мета полягає у відокремленні бізнес-логіки від відображення даних та взаємодії з користувачем.

Архітектура BLoC складається з трьох основних компонентів:

1. Потік подій (Events) – відображає дії користувача або зовнішні події, що впливають на стан додатка.

2. BLoC – клас, який обробляє вхідний потік подій і відображає стан додатка, що залежить від цих подій. Він виконує бізнес-логіку та взаємодіє з різними джерелами даних (наприклад, базами даних або мережами) для отримання та збереження інформації.

3. Потік стану (State) – показує поточний стан додатка, який залежить від вхідних подій та бізнес-логіки BLoC. Цей потік використовується для відображення даних в елементах інтерфейсу користувача.

Така архітектура дозволяє розділити бізнес-логіку і представлення даних, що сприяє більшій масштабованості та повторному використанню коду. Вона також полегшує тестування окремих компонентів додатка та їх використання в різних проєктах. Крім того, ця архітектура підтримує єдину структуру програми, незалежно від її розміру та складності, що забезпечує зручне та ефективне управління додатком у майбутньому.

BLoC Cubit – це простіший варіант архітектури BLoC, який забезпечує легший спосіб управління станом у Flutter-додатках. Cubit є частиною бібліотеки

flutter_bloc і служить для спрощення взаємодії з бізнес-логікою без необхідності визначення подій. Cubit надає зручний інтерфейс для управління станом без потреби в створенні окремих класів для подій. Необхідно просто визначити методи, які змінюють стан. Cubit управляє станом через прості методи, які викликають emit для оновлення стану. Це спрощує читання і написання коду. Cubit використовує потоки (streams) для передачі оновлень стану в UI. Ви можете підписатися на зміни стану, як у звичайному BLoC.

У даному додатку архітектура представлення BLoC Cubit розділяється на:

1. RegisterPageCubit – що відповідає за логіку реєстрації користувача та відправки даних на сервер для валідації та запису даних в базу даних, містить RegisterPageState що в свою чергу має наступні статуси: idle – початковий стан, loading – процесс відправки та збереження даних на сервер, error – помилка при реєстрації, success – успішна реєстрація користувача.

2. LoginPageCubit – що відповідає за логіку авторизації користувача та перевірку пароля та наявності даного користувача на сервері. Містить LoginPageState що в свою чергу має наступні статуси: idle – початковий стан, loading – процесс перевірки даних на сервері, error – помилка при авторизації, success – успішна авторизація користувача.

3. AppControlCubit – що відповідає за локальні данні користувача здебільшого за локалізацію додатка.

4. DaysPageCubit – що відповідає за створення нових днів, відображення днів зі статусом виконаних задач за день, а також видалення днів. Містить DaysPageState що в свою чергу має наступні статуси: idle – початковий стан, loading – процесс отримання даних на сервері, error – помилка при отриманні або видалені днів, success – успішне видалення або отримання днів.

5. ManagementPageCubit – що відповідає за сортування актуальних задач за матрицею Ейзенхауера та за статусом виконання. Містить ManagePageState, що в свою чергу має статуси та параметри схожі з вище наведеними.

6. ThingManagePageCubit – що відповідає за відображення інформації про

задачу (статус виконання, статус терміновості, статус важливості, витрачений час на задачу, заголовок та опис). Виконання задачі шляхом змінення статусу або використання однієї з наведених методик. Видалення задачі або видалення запису часу. Редагування задачі. Має статуси схожі на вище наведені окрім removed, що свідчить про видалення задачі.

7. ThingPageCubit – що відповідає за створення задачі користувачем, містить ThingPageState що в свою чергу має наступні статуси: idle – початковий стан, loading – процесс відправки створеної задачі на сервер для збереження в базі даних, error – помилка при створенні задачі, success – успішне створення нової задачі.

8. SettingsPageCubit – що відповідає за налаштування додатка такі, як зміну мови або даних користувача, вихід з додатка або видалення облікового запису користувача. Містить SettingsPageState що в свою чергу має статуси та параметри схожі з вище наведеними окрім logout – що свідчить за вихід з додатка.

Міжсторінкова навігація є одним з основних функціоналів проєкту. Яка реалізована через пакет app auto route – це пакет для Flutter, який спрощує управління навігацією в додатках, надаючи можливість використовувати автоматизовану маршрутизацію. Він дозволяє створювати маршрути на основі анотацій, що значно полегшує процес налаштування навігації в додатках. У даному випадку додаток має наступні шляхи:

1. LoginRoute – що є початковим шляхом на сторінці авторизації (LoginPage), з якої є навігація через метод pushRegisterPage на сторінку реєстрації (RegisterPage), та pushMainPage на головну сторінку (MainPage).

2. RegisterRoute – що є шляхом на сторінці реєстрації (RegisterPage), з якої є навігація через методи pop (повернення назад) до сторінки навігації, pushMainPage до головної сторінки.

3. MainRoute – що є шляхом на головній сторінці (MainPage), з якої є навігація до сторінки з переліком днів через метод pushDaysPage

4. DaysRoute – що є шляхом на сторінці з переліком днів (DaysPage) з якої іде навігація до сторінки налаштувань через метод pushSettingsPage, до сторінки з матрицею Ейзенхауера (ManagementPage) через метод pushManagementPage та до сторінки для створення нової задачі (ThingsPage) через метод pushThingsPage.

5. ThingRoute – що є шляхом на сторінці для створення нової задачі (ThingsPage) з якої іде навігація через метод pop (повернення назад).

6. ManagmentRoute – що є шляхом на сторінці з матрицею Ейзенхауера (ManagementPage), яка містить в собі три вкладки: Виконати, в Прогресі та Виконано. А також наявна навігація до сторінки управлінню задачею.

7. ThingManageRoute – що є шляхом на сторінці управління задачею (ThingsManagePage), з якої є навігація до сторінок з методиками 90/30 та 25/5, сторінки редагування задачі та навігація через метод pop (повернення назад).

8. SettingsRoute – що є шляхом на сторінці (SettingsPage) , з якої є навігація до сторінки авторизації через метод resetToLoginPage.

Також важливим аспектом є список основних віджетів (контент-модулів), які використовуються в проєкті для створення інтерфейсу користувача:

initState() – це метод, який викликається один раз під час створення Stateful віджета і призначений для ініціалізації початкового стану. У цьому методі можна задавати змінні, встановлювати початкові значення, завантажувати дані з мережі тощо.

setState() – це метод, що викликається після зміни стану Stateful віджета. З його допомогою можна оновити стан віджета та переробити його, щоб відобразити зміни на екрані. Важливо зазначити, що setState() не змінює стан безпосередньо, а лише помічає віджет для перероблення на наступному кадрі.

Далі наведено список віджетів, які становлять основу UI/UX частини проєкту:

1. Container – використовується для обгортання інших віджетів, дозволяючи налаштовувати їхній відступ, фон, форму та інші параметри. Наприклад, можна вставити кнопку, використовуючи Container, та налаштувати

його параметри для досягнення потрібного розміру та відстані.

2. Row – організовує віджети в один ряд, де вони займають рівномірну частину доступного простору. Це дозволяє розташовувати віджети горизонтально та змінювати їхній порядок. Наприклад, для розміщення двох кнопок поруч можна використовувати Row.

3. Column – використовується для організації віджетів у стовпчик, де віджети також займають рівномірну частину простору. Це дозволяє розташовувати віджети вертикально та змінювати їхній порядок. Наприклад, для вставлення двох текстових блоків один під одним можна використовувати Column.

4. Text – призначений для відображення тексту. Він дозволяє використовувати різні шрифти, розміри та стилі. Це віджет, який може приймати текст змінної довжини та форматувати його на потребу.

5. TextField – використовується для введення тексту користувачем. Він дозволяє вводити текст у поле, яке можна налаштувати для різних типів даних, таких як числа, електронні адреси, паролі тощо. TextField також має вбудовану можливість валідації введеного тексту та автозаповнення. За допомогою цього віджета можна збирати дані від користувачів під час реєстрації, входу до облікового запису, пошуку та в інших сценаріях.

6. ListView – призначений для відображення списку елементів, який можна прокручувати. ListView може бути як горизонтальним, так і вертикальним, залежно від налаштувань. Є кілька способів створення ListView у Flutter, один з яких полягає у використанні конструктора ListView та передачі списку даних для відображення. ListView автоматично генерує віджети для кожного елемента у списку.

7. BlocProvider – це віджет у Flutter, що дозволяє надавати доступ до об'єкта BLoC для дочірніх віджетів у дереві без явної передачі через конструктор. BlocProvider працює з пакетом `flutter_bloc` і слугує посередником між Bloc і дочірніми віджетами, зберігаючи об'єкт Bloc і надаючи до нього доступ через

дочірні елементи за допомогою `BlocProvider.of(BuildContext context)`.

8. Scaffold – віджет, який створює базовий макет сторінки з заголовком, тулбаром (AppBar), бічним меню (Drawer), навігаційними кнопками та іншими важливими компонентами, які часто використовуються в мобільних додатках.

9. Expanded – віджет, що розширює дочірній віджет на весь доступний простір, залишений батьківським віджетом. Він зазвичай використовується в комбінації з Row або Column, щоб забезпечити розміщення дочірніх віджетів горизонтально або вертикально. Таким чином, якщо потрібно розмістити кілька віджетів в одному рядку або стовпці, Expanded допомагає заповнити весь вільний простір.

10. Button – віджет для створення кнопок у Flutter, що дозволяють користувачам виконувати певні дії при натисканні. Основні типи кнопок включають: ElevatedButton – піднята кнопка, зазвичай використовується для основних дій; має піднятий стиль і тінь. TextButton – кнопка без обрамлення, зазвичай застосовується для вторинних дій або посилань. OutlinedButton – кнопка з контуром, що зазвичай використовується для додаткових дій і не привертає стільки уваги, як ElevatedButton. Кнопки можуть містити текст або іконки, а також мати різні стилі та налаштування. Наприклад, можна створити кнопку "Відправити" для надсилання повідомлень або "Додати" для створення нового елемента в додатку. Кнопки використовуються для взаємодії користувачів з додатком і виконання різних дій, таких як збереження даних чи переходи на інші сторінки.

3.2 Розробка структурної схеми

На рисунку 3.1 представлено структурну міжсторінкову схему додатка. Структура сторінок складається з 14 сторінок, що показують послідовність їх появи в результаті певних дій у додатку.

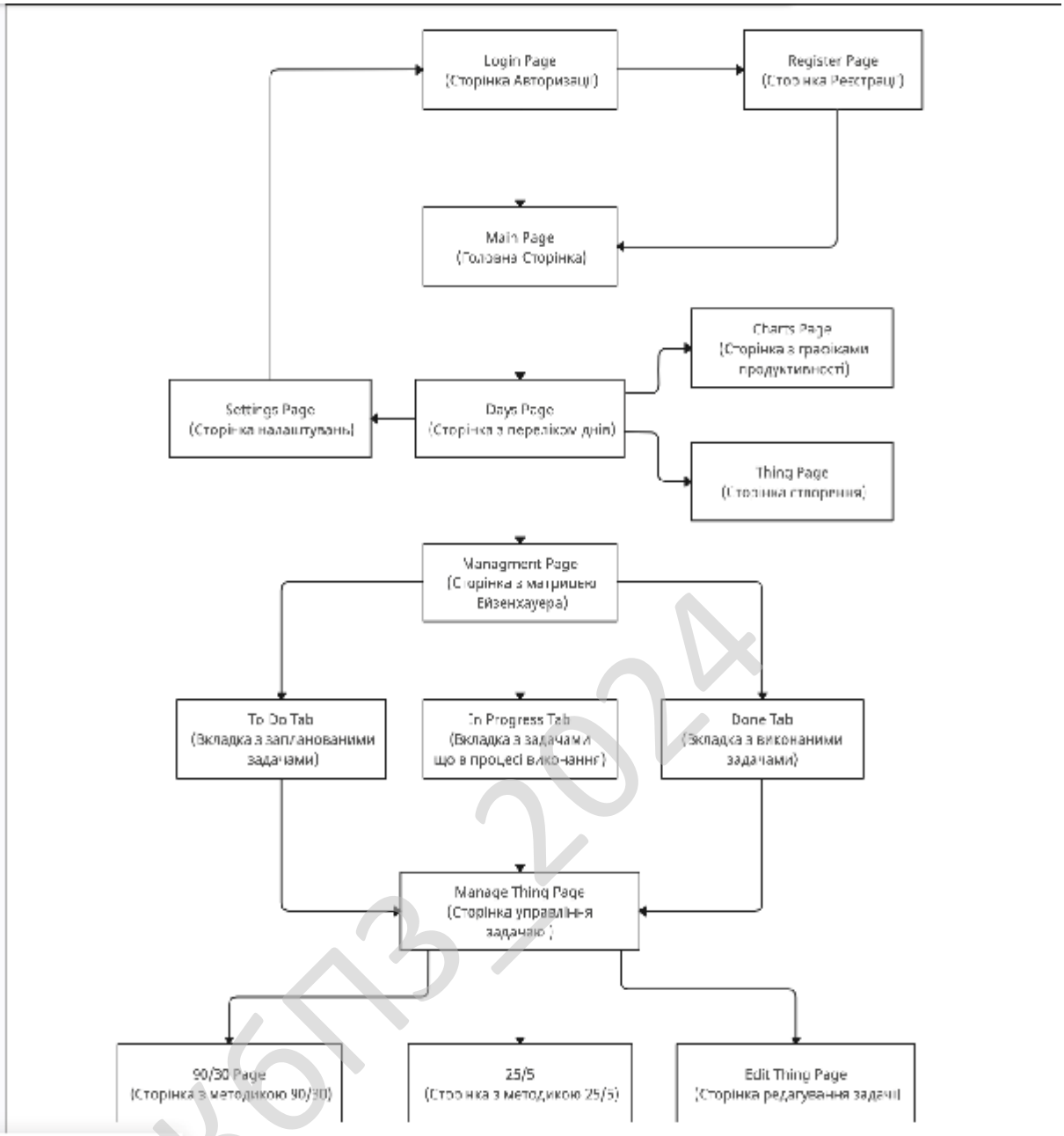

Рисунок 3.1 – Міжсторінкова структурна схема додатка

Кожна сторінка включає в себе окремий cubit та згенерований клас.

Типу Route за допомогою пакета autoroute, за яким відбувається міжсторінкова навігація, де початковою сторінкою є Login Page.

На рисунку 3.2 представлено структурну схему таблиць в базі даних, що містить в собі наступні класи: AccessToken, User, Day, Area, Thing, та дани типу Enum.

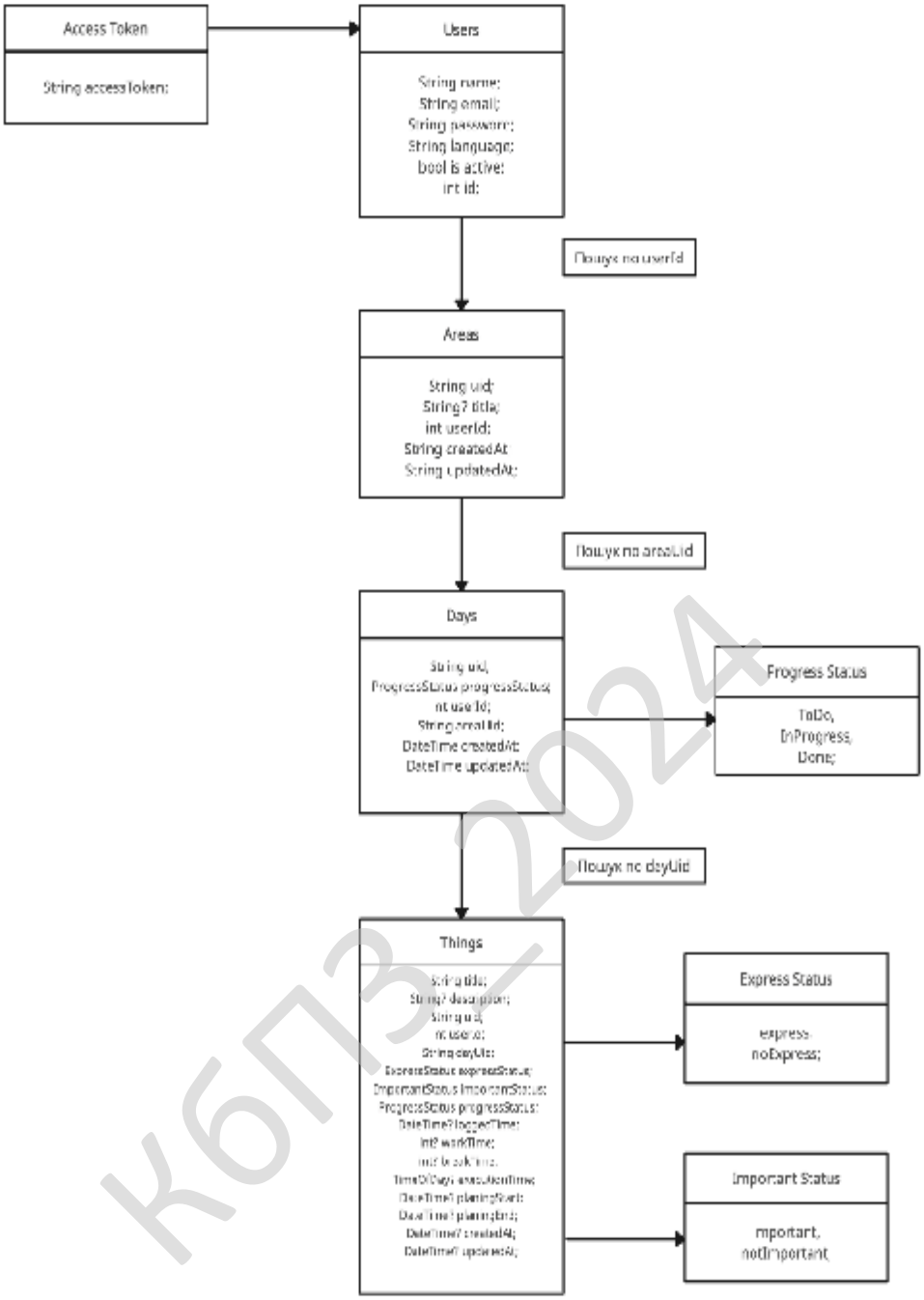

Рисунок 3.2 – Структурна схема таблиць бази даних

Кожен елемент має свій унікальний ідентифікатор – id або uid, за яким відбувається основний пошук нижчих за ієрархією елементів, що є в базі даних. Пошук елементів також може відбуватись за датою створення (createdAt).

Деякі елементи містять в собі Enum – що є методом моделювання даних,

який дозволяє зберігати різні атрибути для об'єктів у базах даних. У контексті Dart це включає реалізацію таких моделей для роботи з даними в додатках.

Також більшість моделей містять Datetime – клас у Flutter та Dart, який використовується для роботи з датою та часом. Він дозволяє створювати, форматувати, змінювати і порівнювати дати й часи, а також витягувати інформацію про рік, місяць, день, годину тощо.

3.3 Розробка функціональної схеми

На рисунку 3.3 зображено функціональну схему роботи архітектури представлення BLoC з використанням альтернативного пакету cubit, який є спрощеною версією даної архітектури представлення та процеси конвертації, взаємодії з сервером та ін.

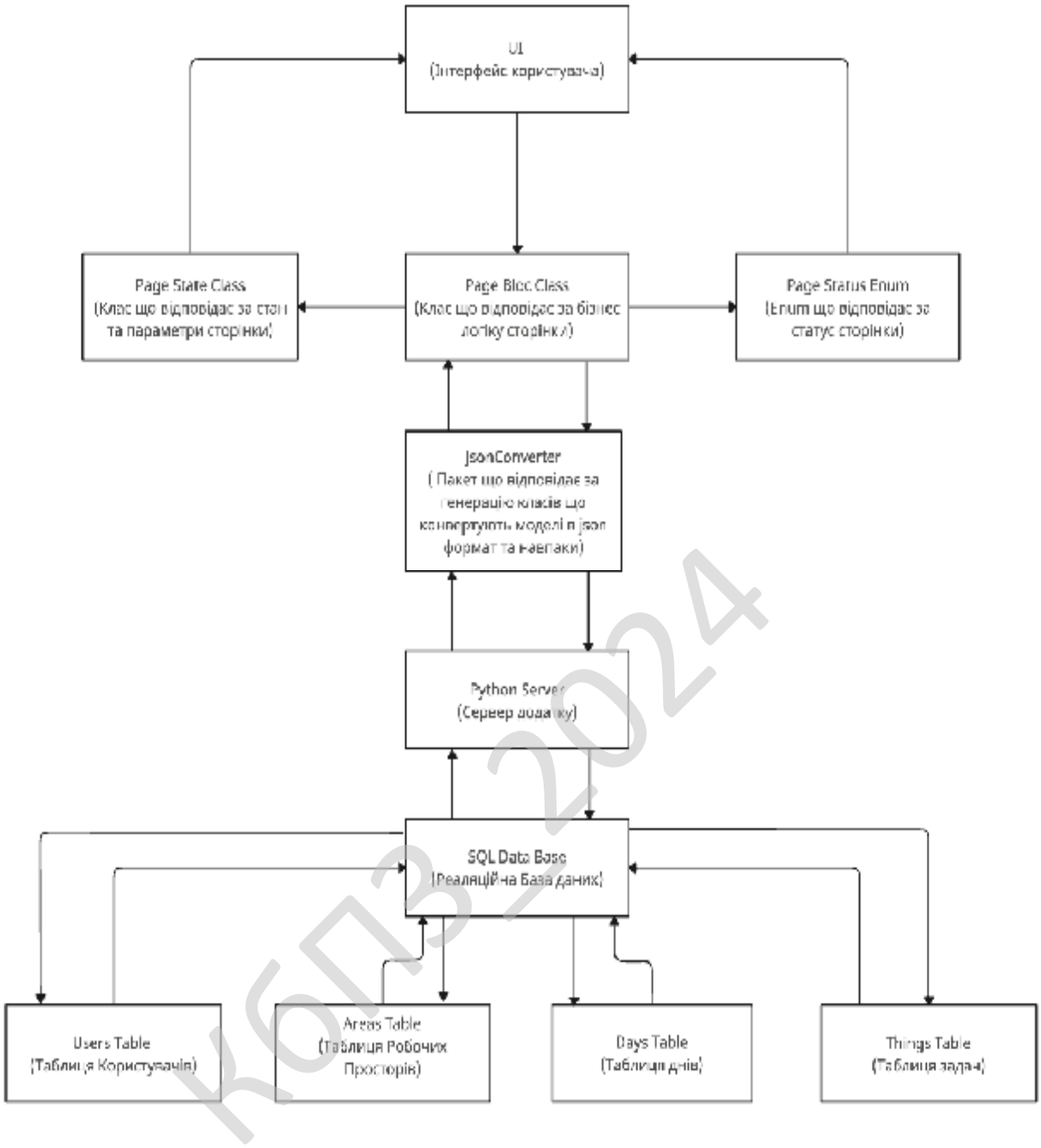

Рисунок 3.3 – Функціональна схема роботи архітектури представлення BLoC з використанням альтернативного пакету cubit

З інтерфейсу користувача викликається метод Bloc класу сторінки, що в свою чергу оброблює інформацію, створює модель та конвертує її за допомогою JsonConverter в json формат, далі іде відправка запиту (request) на сервер, що обробляє та дістає потрібну модель або моделі з таблиць реляційної бази даних, потім надсилає її як відповідь (response) в додаток, де вона конвертується з

фомату json в модель класу та за допомогою методу emit змінює параметри state, метод сторінки onStateChanged за допомогою потоку (Stream) слухає зміни State та оновлює сторінку.

3.4 Розробка діаграми процесів

На рисунку 3.4 зображено діаграму процесів мобільного додатка для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності. Дана діаграма включає в себе усі процеси в додатку та їх взаємодію, де основні взаємодії відбуваються з авторизацією користувача, головною сторінкою, сторінка управління задачею, сторінка з матрицею Ейзенхауера, сторінками з методиками та сторінкою з графіками.

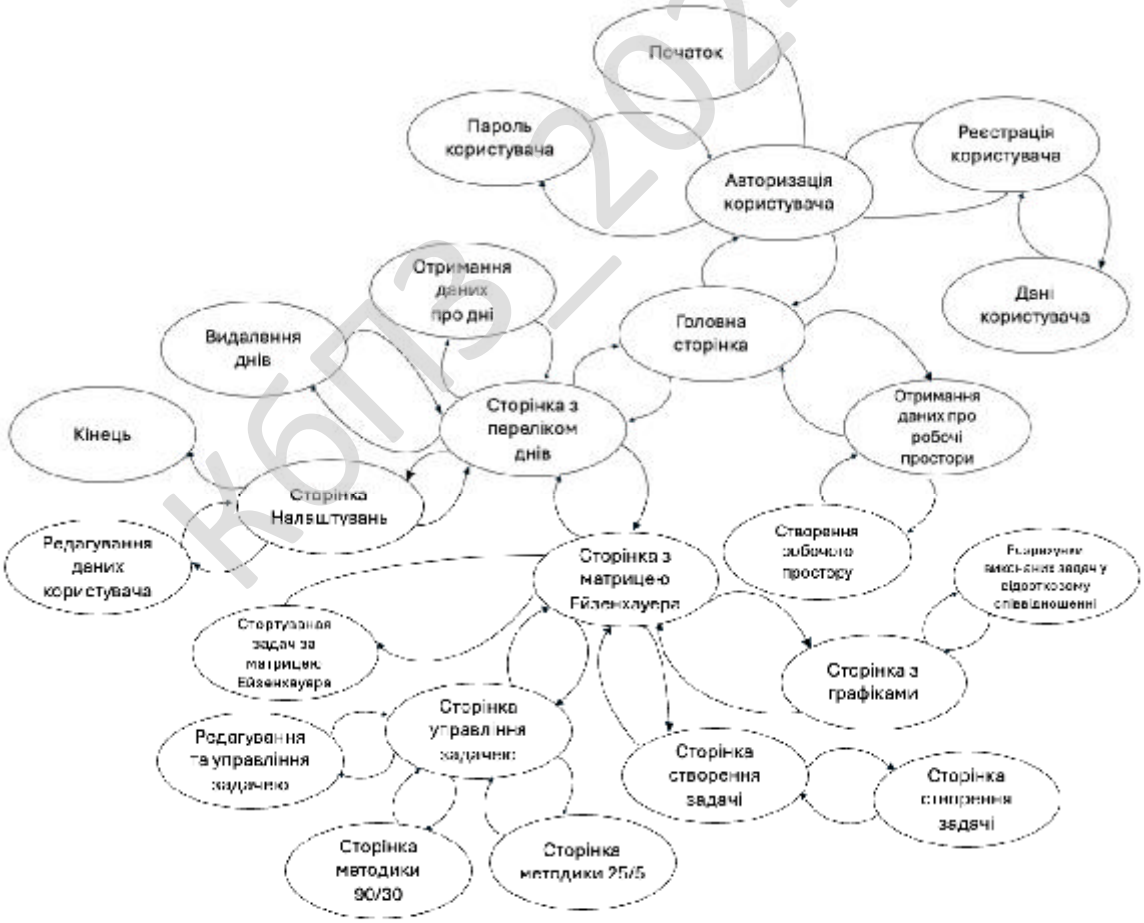

Рисунок 3.4 – Діаграма процесів додатка для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності

Більшість процесів можна поділити на http запити, де GET – це отримання даних про данні користувача, отримання днів та задач користувача, та POST – реєстрація користувача, авторизація користувача, створення задач, створення днів, та робочих середовищ, PUT – редагування задач. DELETE – видалення задач, днів або облікового запису користувача.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 представлено фрагмент блок-схеми головної програми, що ілюструє процес авторизації та реєстрації користувачів у додатку.

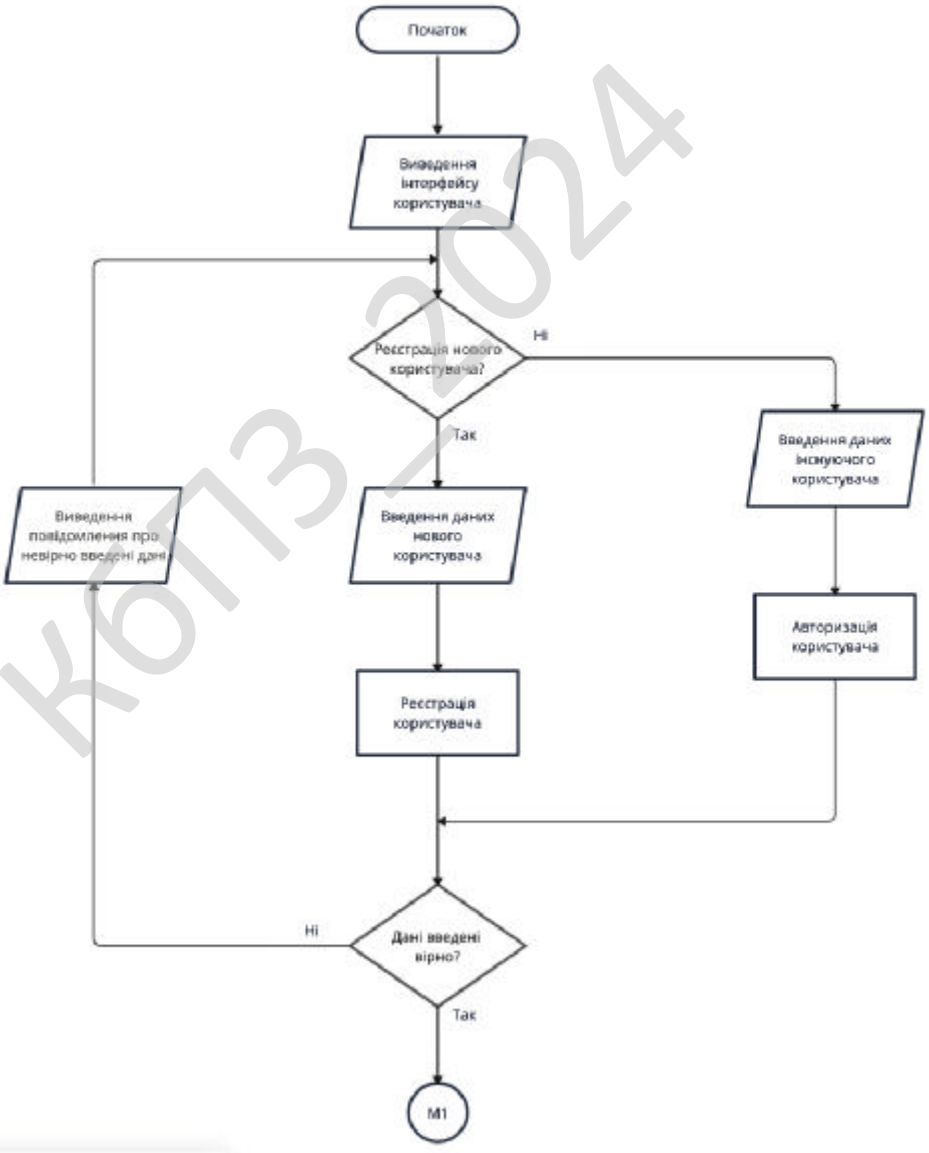

Рисунок 4.1 – Частина блок-схеми основної програми – авторизації та реєстрації користувачів

```
Future<void> login(LoginRequest loginRequest) async {
  return makeErrorParsedCall(() async {
       final accessToken = await _apiClient.login(
       LoginRequest(
       email: loginRequest.email,
       password: loginRequest.password,
       ),
       );
       await _preferencesService.setAuthDetails(
       AuthDetails(
       userId: 0,
       accessJwtToken: accessToken.accessToken,
       ),
       );
       final user = await _apiClient.getUser();
       await _preferencesService.setAuthDetails(
       AuthDetails(
       userId: user.id,
       accessJwtToken: accessToken.accessToken,
       ),
       );
  });
}

Future<RegisterResponse>register(RegisterRequest registerRequest) async {
       return makeErrorParsedCall(() async {
       final response = await _apiClient.register(registerRequest);
       final accessToken = await _apiClient.login(LoginRequest(
       email: response.email,
       password: registerRequest.password,
       ));
       await _preferencesService.setAuthDetails(
            AuthDetails(
            userId: response.id,
            accessJwtToken: accessToken.accessToken,
```

```
            ),
          );
          return response;
        });
    }
```

Вище зазначено роботу декількох методів та сервісів таких як, register method, login method, apiClient service, preference service.

apiClient login – відповідає за відправку request на сервер з метою авторизації користувача, після успішної авторизації, відбувається запит на отримання даних користувача методом getUser, збереження даних про ідентифікатор та accessToken користувача за допомогою PreferenceService.

Моделі, які були використані для реєстрації та авторизації: AuthDetails, RegisterRequest, LoginRequest.

```
@JsonSerializable()
class RegisterRequest extends Equatable {
     final String password;
     final String language;
     final String name;
     final String email;

     const RegisterRequest({
     required this.password,
     required this.language,
     required this.name,
     required this.email,
     });
}

@HiveType(typeId: HiveTypeConstants.authDetailsTypeId)
@JsonSerializable()
class AuthDetails extends Equatable {
  @HiveField(0)
  final int userId;
  @HiveField(1)
  final String accessJwtToken;

  const AuthDetails({
```

```
    required this.userId,
    required this.accessJwtToken,
  });
}
```

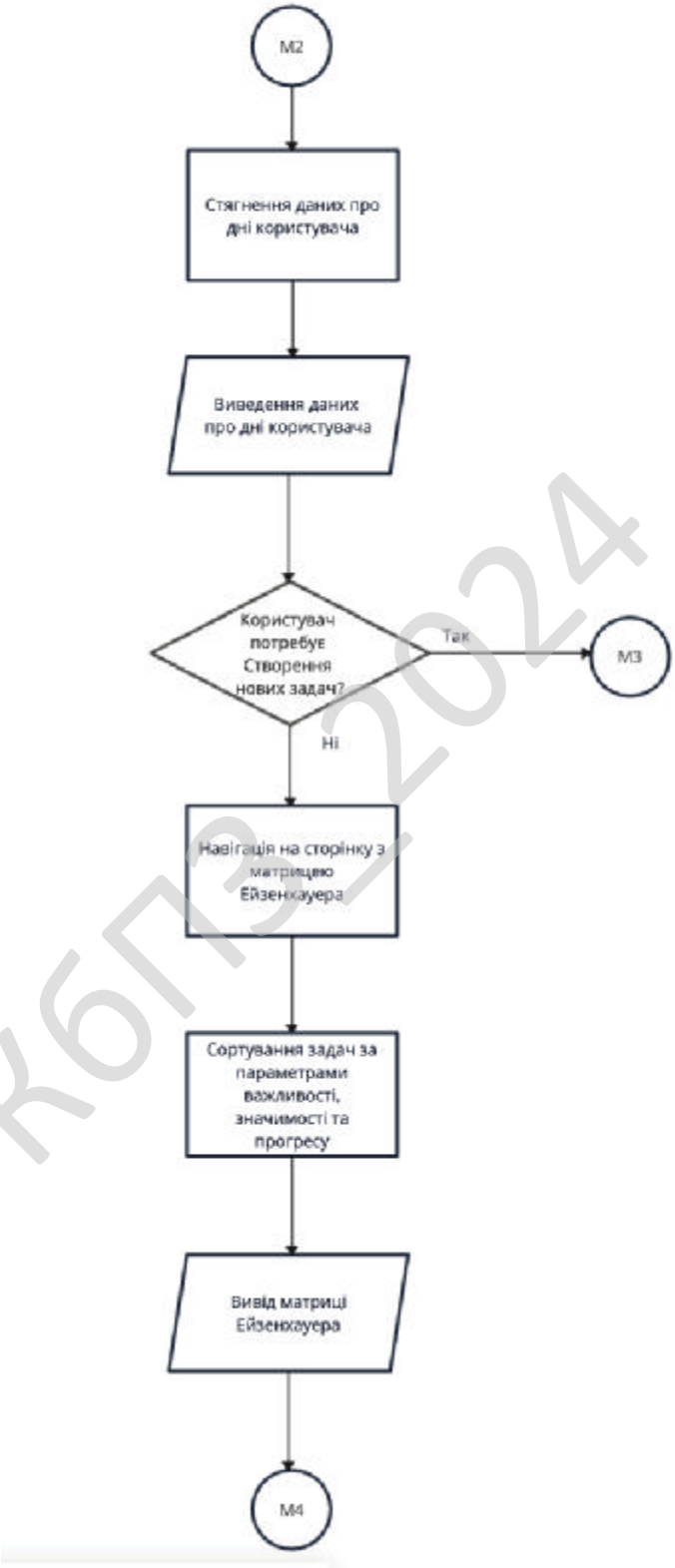

Рисунок 4.2 – Частина блок-схеми основної програми – сторінка переліку днів та матриця Ейзенхауера

Опишемо роботу окремих процесів сортування задач:

```
Future<void> getTasksAndSort(String dayUid) async {
    await makeErrorHandledCall(() async {
      emitIfNotClosed(state.copyWith(
        status: ManagementPageStatus.loading,
      ));
      final tasks = await taskService.getTask(dayUid);
      final toDo = tasks
          .where((thing) => thing.progressStatus == ProgressStatus.toDo)
          .toList();
      final inProgress = tasks.where((thing) => thing.progressStatus ==
ProgressStatus.inProgress)
          .toList();
      final done = tasks
          .where((thing) => thing.progressStatus == ProgressStatus.done)
          .toList();
      emitIfNotClosed(state.copyWith(
        done: done,
        inProgress: inProgress,
        toDo: toDo,
        status: ManagementPageStatus.success,
      ));
    });
  }
```

Вище вказаний алгоритм сортування задач за параметром прогресу на три окремих масиви, таких як toDo, inProgress, done.

```
List<Thing> _getExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
```

```
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getExpressAndNoImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.notImportant)
        .toList();

    return things;
  }
```

Вище вказаний алгоритм сортування задач за параметрами важливості та терміновості для створення матриці Ейзенхауера.

Метод where працює, використовуючи ітерацію та умовні перевірки з використанням генераторів для створення нового Iterable, що містить елементи, які відповідають заданій умові.

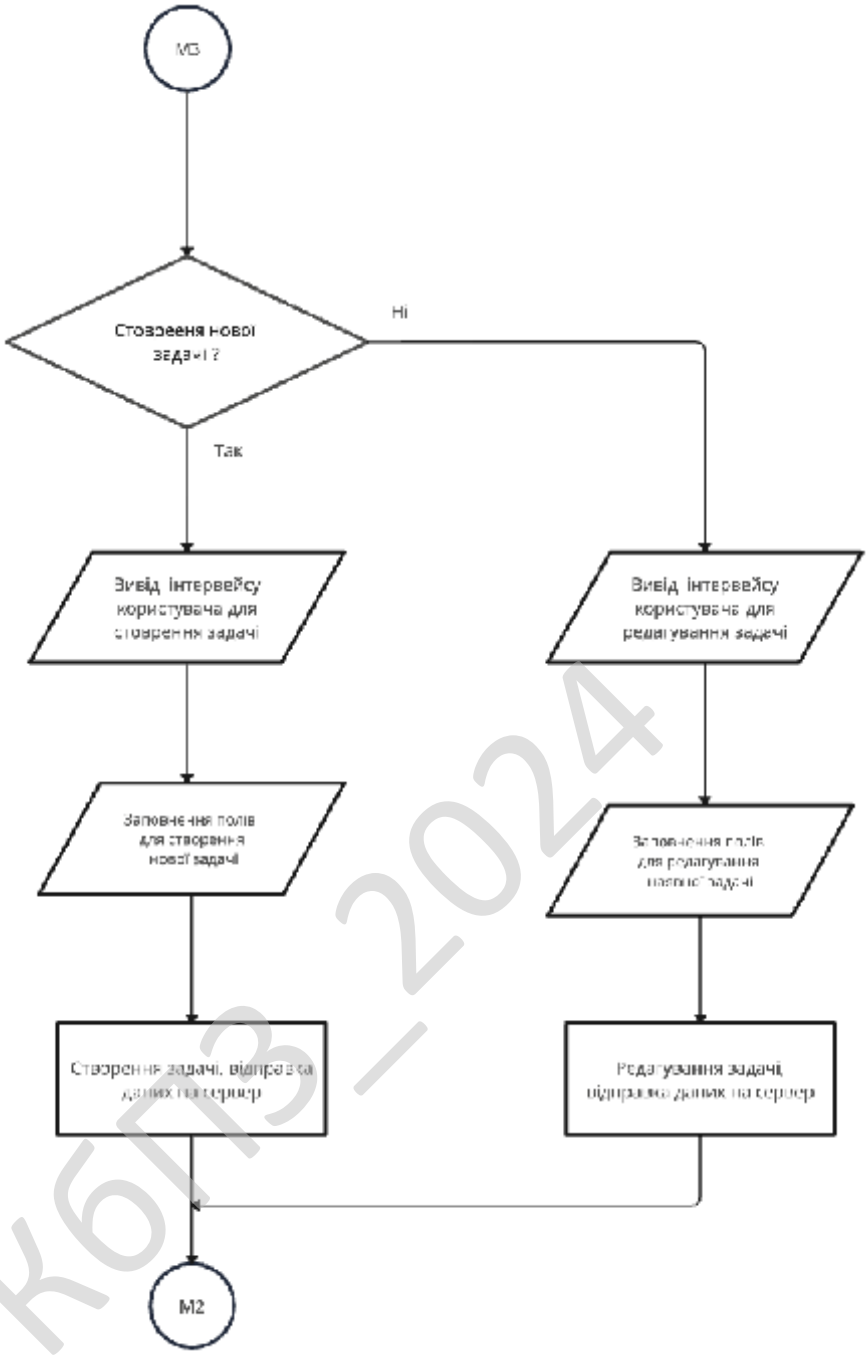

Рисунок 4.3 – Частина блок-схеми основної програми – Сторінка створення та редагування задачі

Сторінка відповідає за дві основні задачі, редагування наявних задач та створення нових.

Після створення або редагування задачі генерується наступна модель для відправки на сервер

```
class Thing {
  final String title;
  final String? description;
  final String uid;
  final int userId;
  final String dayUid;
  final ExpressStatus expressStatus;
  final ImportantStatus importantStatus;
  final ProgressStatus progressStatus;
  @DateTimeJsonConverter()
  final DateTime? loggedTime;
  final int? workTime;
  @TimeOfDayJsonConverter()
  final TimeOfDay? executionTime;
  @DateTimeJsonConverter()
  final DateTime? planingStart;
  @DateTimeJsonConverter()
  final DateTime? planingEnd;
  @DateTimeJsonConverter()
  final DateTime? createdAt;
  @DateTimeJsonConverter()
  final DateTime? updatedAt;
  Thing({
    required this.title,
    this.description,
    required this.uid,
    required this.expressStatus,
    required this.importantStatus,
    required this.progressStatus,
    required this.dayUid,
    required this.userId,
    this.loggedTime,
    this.workTime,
    this.executionTime,
    this.planingStart,
    this.planingEnd,
    this.updatedAt,
    this.createdAt,
  });
```

TimeOfDayJsonConverter() відповідає за конвертування моделі TimeOfDay в timestamp.

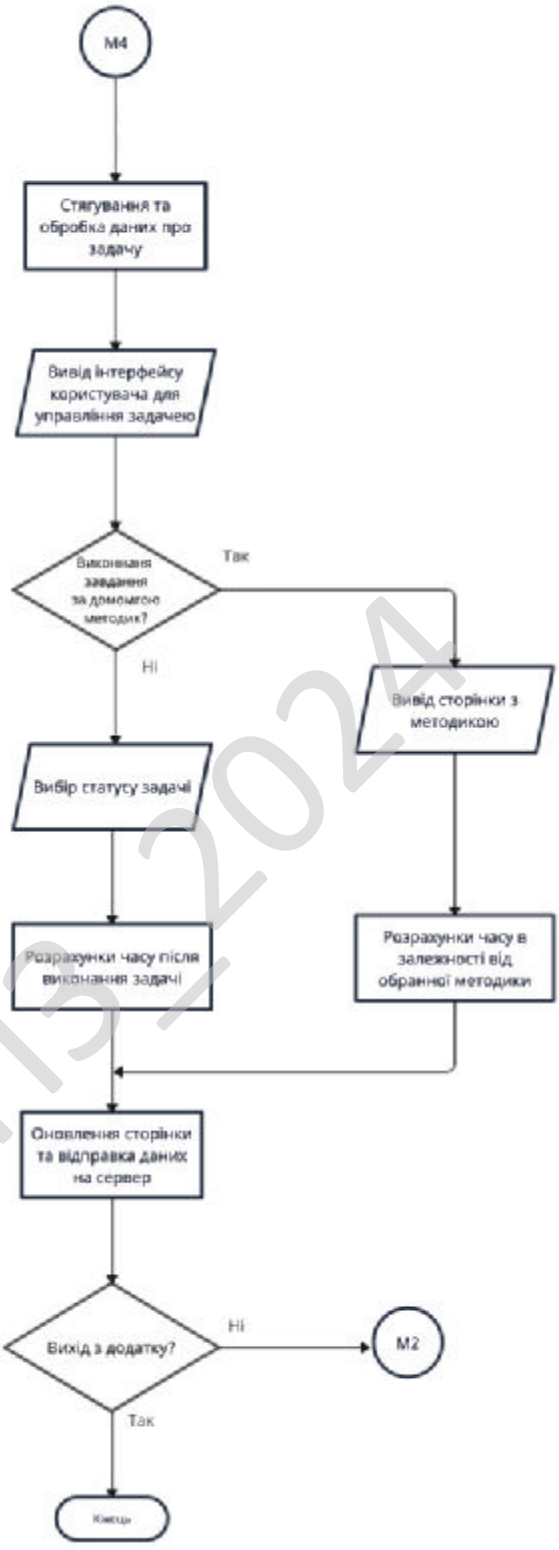

Рисунок 4.4 – Частина блок-схеми основної програми – Сторінка створення управління задачею

На сторінці управління задачею користувачу представлено декілька функцій таких, як оновлення статусу задачі, виконання задачі за допомогою методик продуктивності, запис часу на виконання задачі, можливість видалення записів часу відведеного на задачу, можливість видалення задачі та вихід з додатку.

4.2 Захист розробленого програмного забезпечення

Додаток поширюється за ліцензією GNU General Public License (GPL), яка дозволяє вільно використовувати, змінювати та поширювати програмне забезпечення з відкритим кодом, зберігаючи при цьому авторські права та вказуючи на джерело походження.

GNU General Public License (GPL) – це вільна ліцензія на програмне забезпечення, створена Фондом вільного програмного забезпечення (FSF), яка забезпечує доступність програм для користувачів і захищає їхні права. Згідно з умовами GPL, додатки, поширювані під цією ліцензією, повинні залишатися відкритими для користувачів і доступними для модифікацій та подальшого розповсюдження.

Основні принципи GPL полягають у тому, що програмне забезпечення з відкритим вихідним кодом може вільно використовуватися, змінюватися та поширюватися користувачами. Крім того, ця ліцензія гарантує безкоштовне використання програмного забезпечення та зобов’язує розробників зберігати авторські права і вказувати джерело походження.

Однією з ключових характеристик GPL є її принцип ліцензійної взаємності. Це означає, що якщо програмне забезпечення, яке поширюється за умовами GPL, використовується в проєкті з відкритим кодом, то цей проєкт також повинен поширюватися під ліцензією GPL. Таким чином, GPL гарантує збереження відкритості та свободи відкритого програмного забезпечення, навіть якщо його використовують у комерційних проєктах.

GPL також захищає права користувачів, запобігаючи зміні ліцензії після поширення програмного забезпечення під GPL. Це означає, що якщо додаток був випущений під GPL, користувачі мають право використовувати, змінювати, розповсюджувати його в оригінальному вигляді чи у зміненій версії, а також застосовувати його в комерційних проєктах за умови дотримання ліцензійних вимог.

Однак, ліцензія GPL має і свої обмеження. Наприклад, згідно з її умовами, весь код, який інтегрується з додатком, також повинен бути відкритим і розповсюджуватися під GPL. Це може викликати труднощі, якщо використовується код з іншими типами ліцензій, зокрема, пропрієтарними.

Незважаючи на це, GPL залишається однією з найпопулярніших ліцензій для вільного програмного забезпечення, яка гарантує відкритість, свободу використання та захист прав як користувачів, так і розробників. Якщо ви плануєте поширювати свій додаток за ліцензією GPL, важливо детально ознайомитися з її умовами. Використання GPL передбачає додавання файлу з текстом ліцензії до програмного забезпечення та забезпечення вільного доступу до його вихідного коду. Також потрібно дотримуватись вимог ліцензії, зокрема, надавати вільний доступ до вихідного коду, дозволяти його зміну і поширення під GPL.

Один із способів застосування GPL полягає у включенні файлу ліцензії до кореневої папки програмного забезпечення та додаванні відповідних вказівок у кожен файл із вихідним кодом проєкту. У цьому файлі повинні бути зазначені умови ліцензії та права користувачів.

Крім того, необхідно забезпечити доступ до вихідного коду програми через Інтернет або на носії, який поставляється разом із програмним забезпеченням. Вихідний код має бути доступний у тому самому форматі, в якому розроблялася програма. Наприклад, якщо програма була написана на мові Dart, то вихідний код також має бути наданий у цьому форматі.

Застосування GPL також передбачає, що весь код, пов’язаний із

програмним забезпеченням, повинен поширюватися під GPL або іншою сумісною вільною ліцензією. Це стосується бібліотек, фреймворків та іншого коду, який використовується в проєкті.

У цьому випадку всі вихідні дані зберігаються на платформі GitHub, що дозволяє переглядати та експортувати файли. Всі бібліотеки, які використовуються в додатку, є відкритими для використання.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблений додаток має широкий функціонал, додаток сумісний з операційними системами iOS та Android, забезпечуючи підтримку для користувачів пристроїв обох платформ. Задля безпеки паролі користувачів паролі хешуються, а користувачі мають змогу змінити свої облікові дані.

Для впровадження додатка в промислову експлуатацію потрібно налаштувати Apple Developer Account, для його розміщення на iOS, налаштування Google Developer Account для розміщення додатка на Android. Задля виконання вимог розміщення додатка на різних платформах було впроваджено видалення облікових даних користувача с додатка, а також інформування користувача про умови використання, довготривале збереження задач та інших даних користувача.

Додаток має візуально приємний інтерфейс що задовольняє потреби користувачів.

На рисунку 5.1 зображена матриця Ейзенхауера в додатку, на ній можна побачити розподіл на 4 секції, та кількість задач у кожній з них, якщо задач в секції немає, додаток повідомляє користувачу про їх відсутність. У верхній панелі додатка зображена назва сторінки та іконка у виді плюса, що дозволяє перейти до створення додаткових задач.

На нижній панелі є можливість переглянути які задачі заплановані на виконання, які в прогресі виконання та які вже виконано, вони поділяються на окремі Tab.

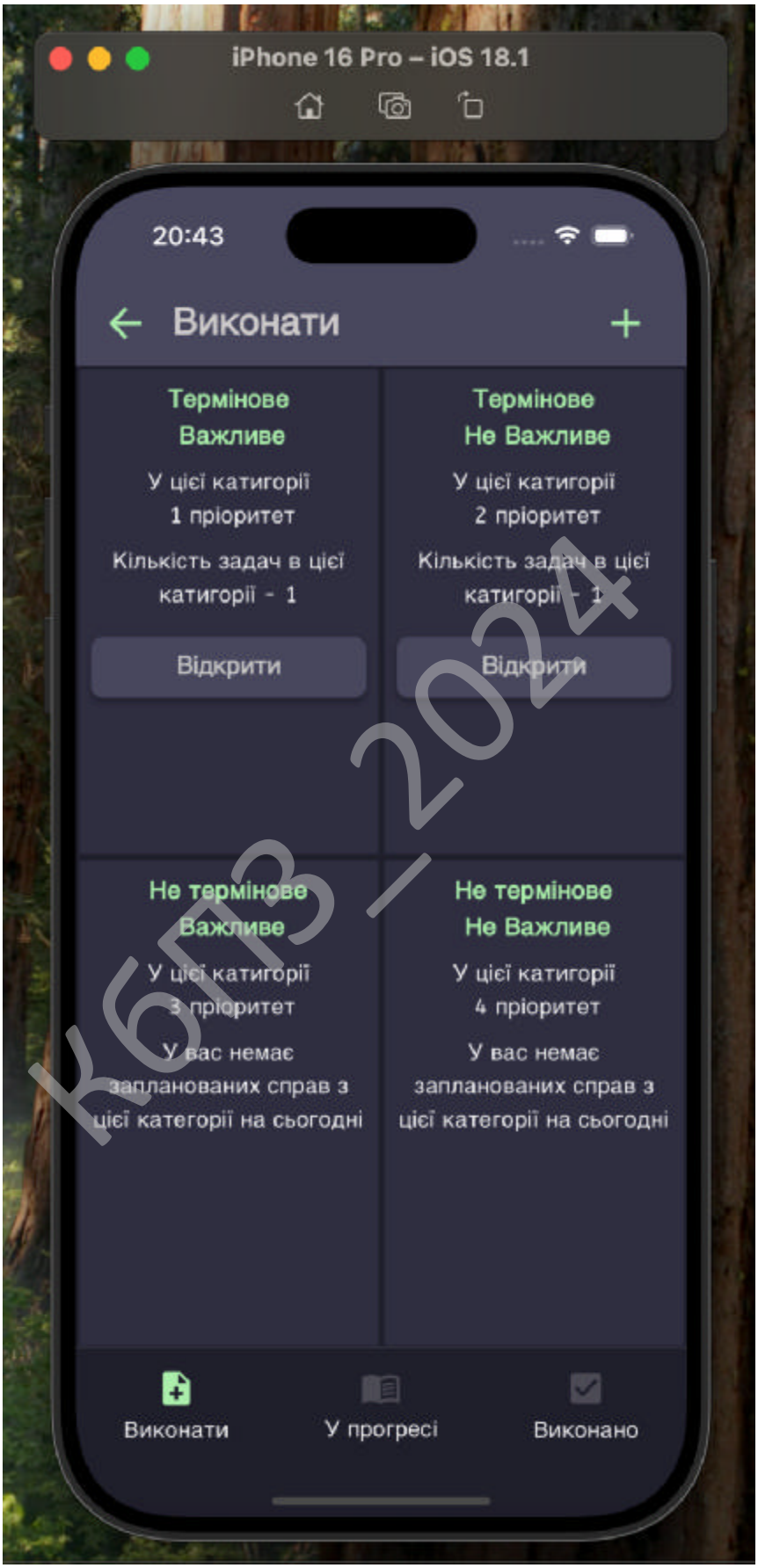

Рисунок 5.1 – Матриця Ейзенхауера в додатку

На рисунку 5.2 зображено перелік задач окремої секції матриці Ейзенхауера, кожну задачу можна розгорнути та переглянути її опис, дату створення, та назву. Існує можливість видалення задачі або переходу до сторінки управління задачею. На верхній панелі є можливість повернутись назад.

Рисунок 5.2 – Перелік задач окремої секції матриці Ейзенхауера

На рисунку 5.3 зображена сторінка управління задачею. Вона містить інформацію про задачу, її статуси, функції виконання (з методиками та без), функції запису та видалення часу. Та можливість переходу до сторінки навігації.

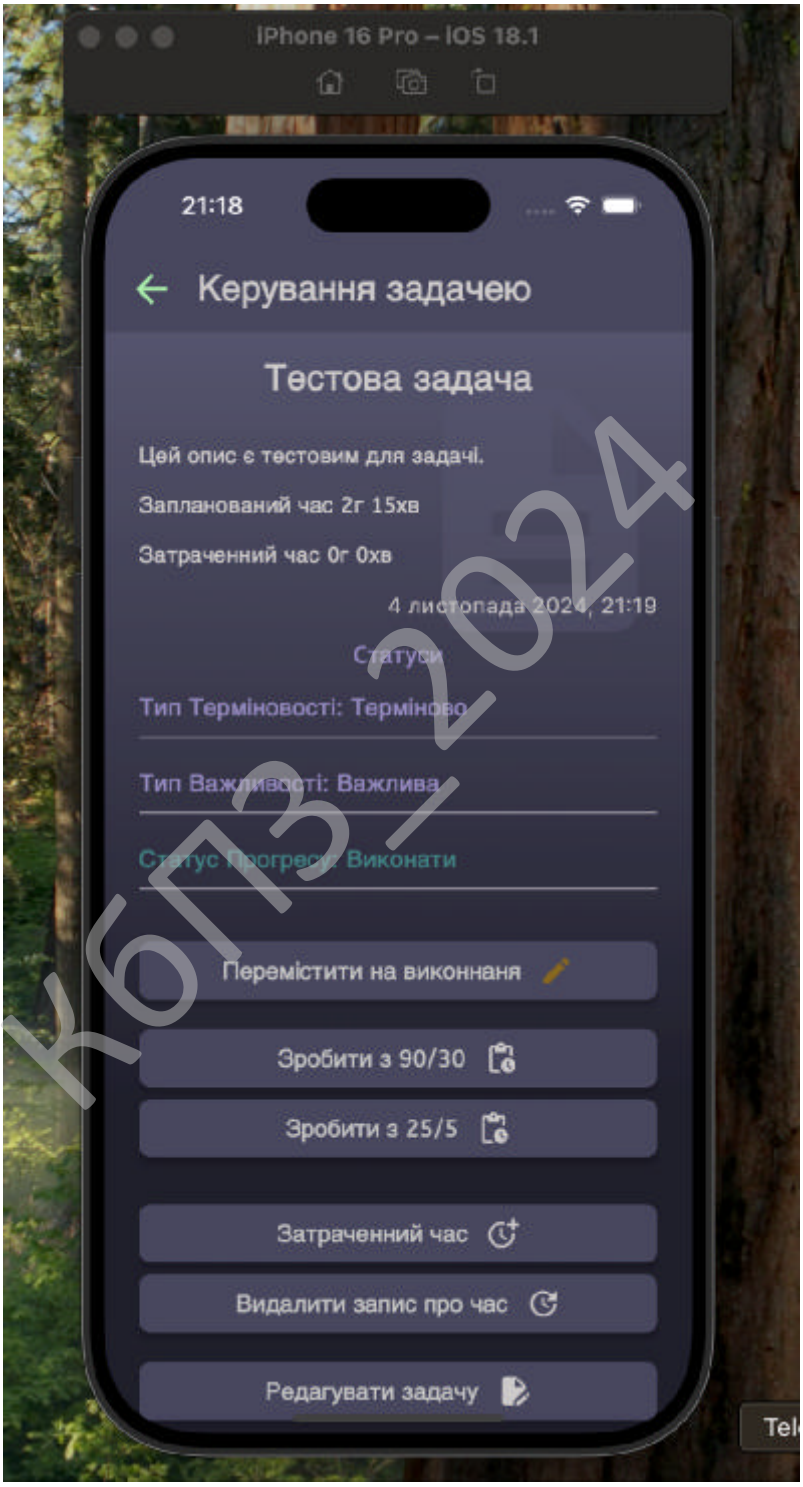

Рисунок 5.3 – Сторінка управління задачею

На рисунку 5.4 зображено роботу методики 25/5 в додатку. Запускається умовний таймер на 25 хвилин, в цей час користувач розпочинає виконувати завдання, потім після дзвінка запускається таймер перерви, в цей час користувач відпочиває.

Є функції паузи, тоді таймер призупиняється, та функції “Виконано”, що поверне користувача до сторінки управління задачею та оновить статус прогресу.

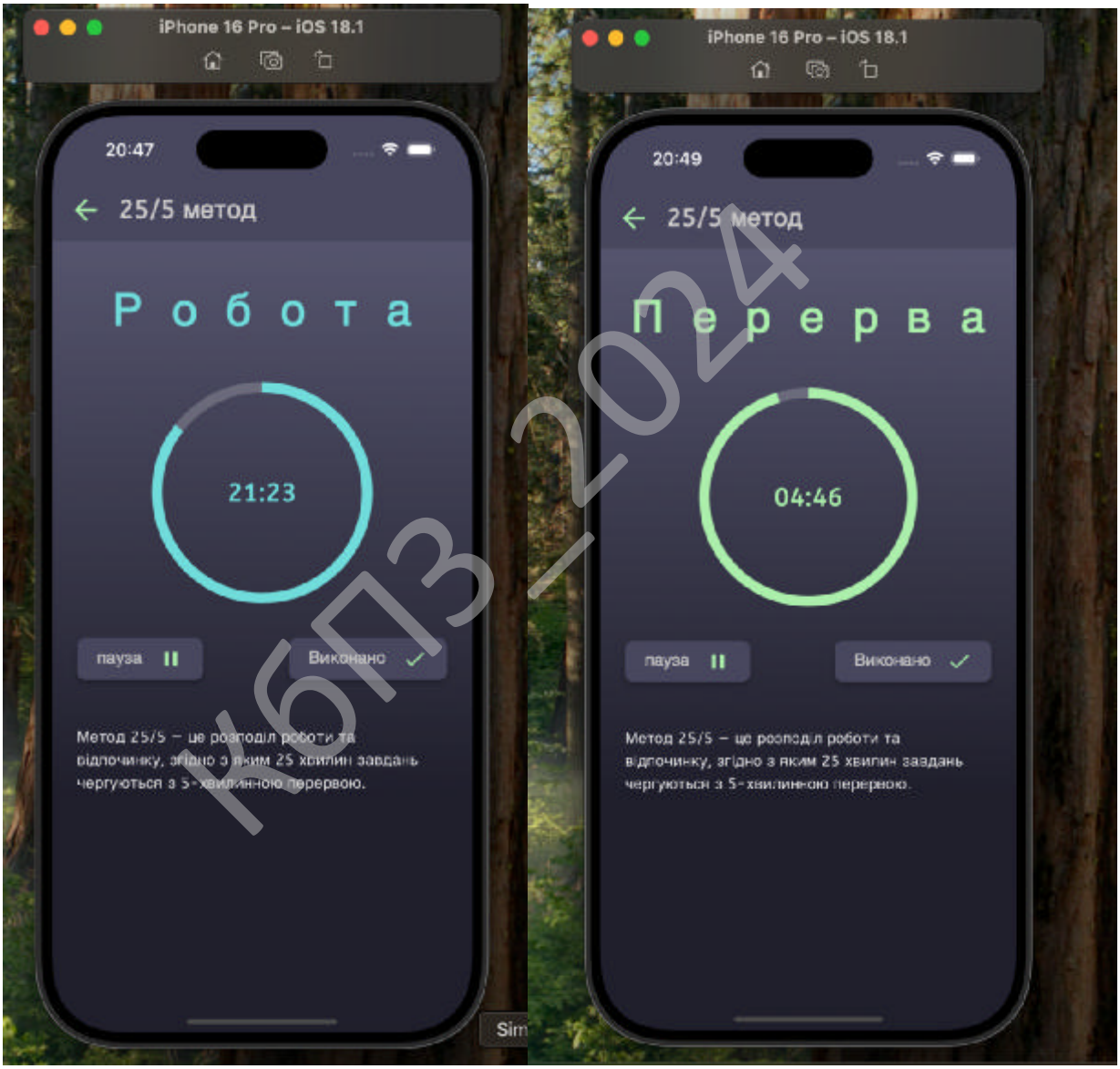

Рисунок 5.4 – Сторінка методики 25/5

На рисунку 5.5 зображено сторінку з графіками, що відображають прогрес виконаних задач у відсотковому співвідношенні, інтервал осі x – задачі за місяць.

Інтервал осі у – відсоткове співвідношення виконаних задач.

Рисунок 5.5 – Сторінка з Графіками

На рисунку 5.6 зображено сторінку налаштувань користувача, де можна змінити ім’я, мову, пароль, вийти з додатку або видалити аккаунт.

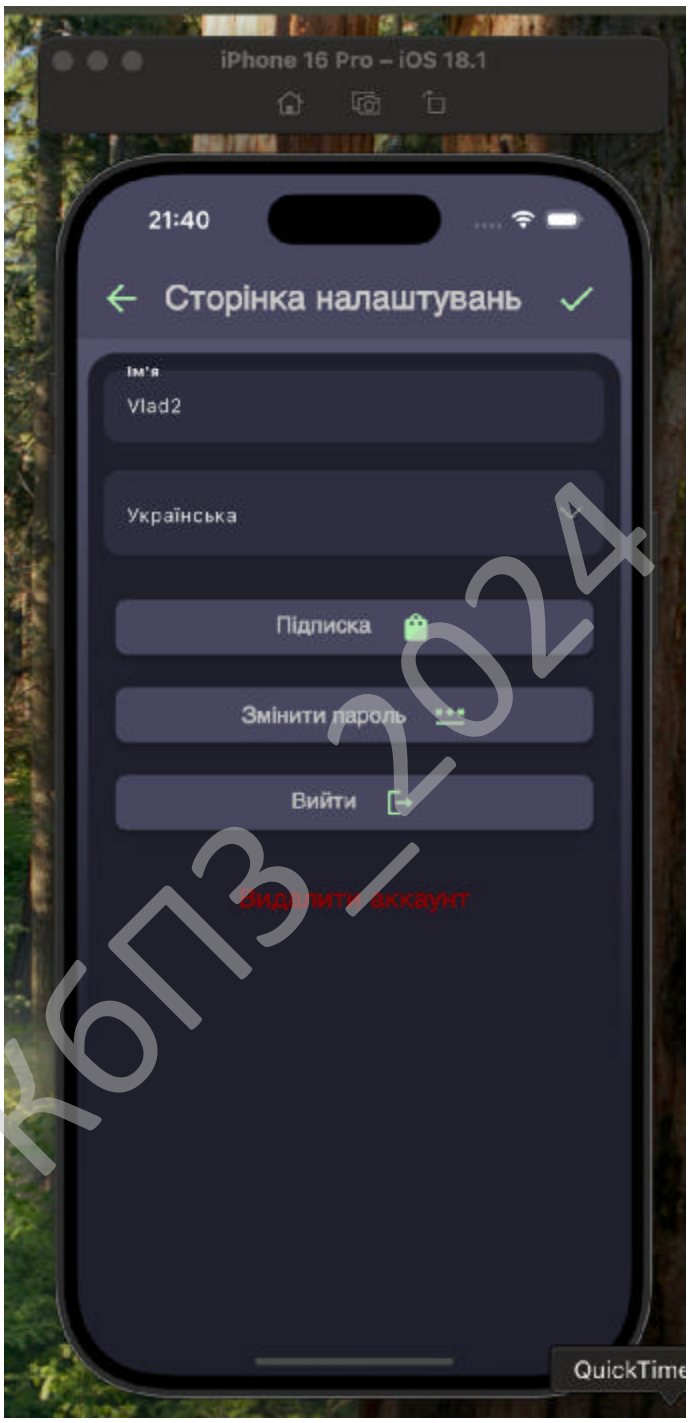

Рисунок 5.6 – Сторінка налаштування

На рисунку 5.7 зображена сторінка з переліком днів та загальним статусом задач. Загальний статус задач визначається наступним чином, якщо усі задачі

мають статус toDo (тільки заплановані), тоді загальний статус буде to Do, якщо в прогресі знаходиться хоча б одна задача, тоді загальний статус буде in Progress, якщо усі задачі виконані, тоді загальний статус буде done, новий день створюється автоматично або вручну також є функція видалення дня.

Рисунок 5.7 – Сторінка з переліком днів

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення мобільного додатка для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.

Об'єктом дослідження є процес автоматизації методик продуктивності за допомогою мобільних додатків.

Предметом дослідження є методи та алгоритми інтеграції сучасних методик продуктивності у мобільний додаток.

Методи дослідження: аналіз наукових і технічних статей, книг та інших матеріалів, що стосуються теми; систематизація літератури та наукових джерел; методи автоматизованого проєктування (моделювання системи, проєктування інтерфейсу користувача, тестування створеного мобільного додатка для оцінки його відповідності поставленим цілям); методи розробки програмного забезпечення (спіральна модель); методи розробки мобільних додатків (кросплатформна розробка).

Наукова новизна отриманих результатів. У процесі вирішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод автоматизації методик продуктивності 90/30, Pomodoro та матриці Ейзенхауера.

2. Розроблено вітчизняний мобільний додаток для менеджменту задач за допомогою методик продуктивності, який має більш ширший функціонал та більш адаптивний інтерфейс у порівнянні з існуючими аналогами та спрямований на потреби сучасного користувача у вирішені широкого спектру завдань у найбільш продуктивний та ефективний спосіб.

Практична цінність отриманих результатів полягає в тому, що **розроблено** додаток**, який** дозволяє автоматизувати методики продуктивності.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Для мобільного додатка, що допомагає в управлінні задачами на основі матриці Ейзенхауера та сучасних методик продуктивності, цільова аудиторія визначена за наступними характеристиками:

Вік: 20–45 років. Це переважно люди активного робочого віку, які часто зіштовхуються з великою кількістю задач та потребують інструментів для ефективного управління часом.

Стать: як чоловіки, так і жінки.

Доходи: середній та вище середнього. Люди, які готові інвестувати в інструменти для підвищення особистої та професійної ефективності.

Освіта: вища або незакінчена вища освіта.

Розглянемо також поведінкові характеристики:

Інтереси: управління часом, особистий розвиток, ефективність праці, баланс між роботою та особистим життям.

Потреби: покращення продуктивності, зменшення стресу через хаотичне управління задачами, упорядкування робочих процесів.

Стиль життя: активний, зайнятий; часто це професіонали, які працюють в офісах, фрілансери, підприємці, менеджери середньої та вищої ланки.

Інші параметри:

Цінності: ефективне використання часу, орієнтація на результат, особистий розвиток, здоровий робочий підхід.

Технологічна грамотність: високий рівень. Це люди, які активно користуються сучасними додатками та сервісами для організації свого часу та роботи.

Важливо зазначити наступні потенційні сегменти (рисунок 7.1):

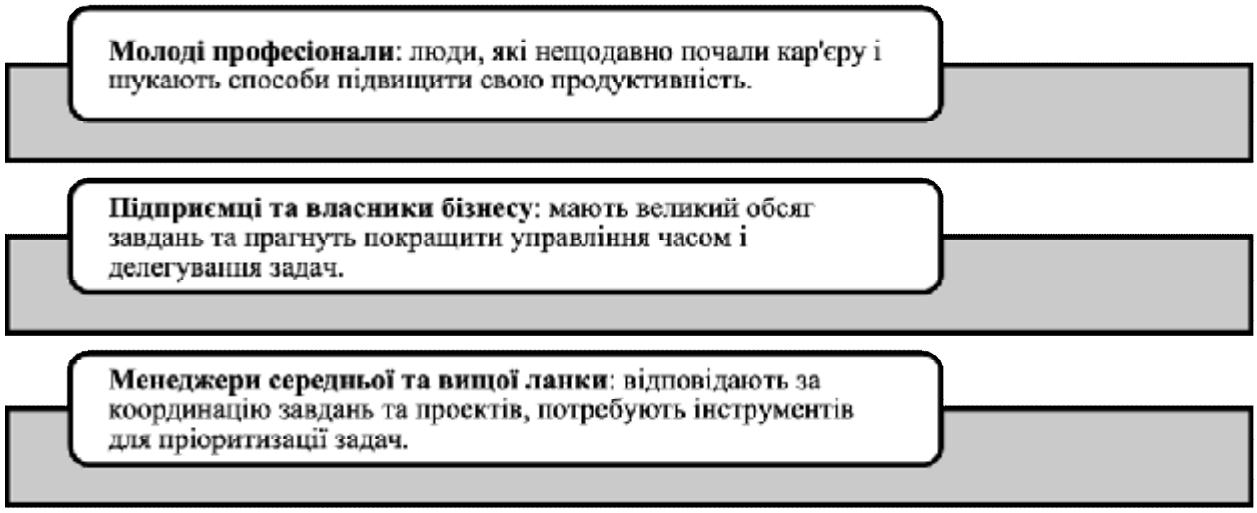

Рисунок 7.1 – Цільова аудиторія

Ця група має спільну мету – покращити управління часом і продуктивність, використовуючи інноваційні методи та підходи, такі як матриця Ейзенхауера та сучасні техніки ефективності.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості продукту за допомогою методів експертних оцінок передбачає аналіз та врахування різних факторів, що визначають його успішність. Першочергово обираються критерії для оцінки (рисунок 7.2):

Рисунок 7.2 – Критерії оцінювання

Більш детально розглянемо запропоновані критерії.

Функціональність: мобільний додаток має декілька методик продуктивності в основі якої лежить матриця Ейзенхауера, користувач може відслідковувати прогрес своїх задач по графіку, користувач має свій обліковий запис, усі його задачі зберігаються на сервері, що дає змогу зайти в додаток з іншого пристрою та продовжити роботу.

Додаток підтримує 2 мови: українська, англійська. Додаток розповсюджується на платформи iOS та Android. Користувач також має змогу менеджменту задачі власноруч. Також є можливість запису часу.

Зручність використання (юзабіліті): додаток має приємний та інтуїтивно зрозумілий інтерфейс (UI). Перевагу було надано темному оформленню, оскільки темний режим мобільного інтерфейсу, зменшує навантаження на очі користувача, позитивно впливає на зорове сприйняття та робить використання додатка більш комфортним.

Також є валідація полів з повідомленнями користувачу. Можливість редагувати задачу, обліковий запис, змінити мову. Кожна сторінка не навантажена інформацією та розділяється на окремі табки, діалогові вікна тощо.

У додатку присутні скролл пікери, що забезпечують зручний вибір часу.

Продуктивність: додаток написаний на сучасній мові програмування Dart з фреймворком Flutter з використанням додаткових пакетів та алгоритмів, що забезпечує високу продуктивність.

Надійність: додаток був ретельно протестований з усіма можливими випадками на різних емуляторах, пристроях та платформах.

Безпека: усі дані надійно захищенні на сервері, паролі користувачів хешуються, а доступ до задач користувача та іншої інформації надається виключно за accessToken. Також доступна функція зміни паролю.

Інноваційність: автоматизація методик продуктивності та використання передових технологій для розборки мобільних додатків, роблять дану роботу унікальною та інноваційною.

Сумісність: як вже було сказано вище додаток працює на двох платформах iOS та Android, та на телефонах з різним розширенням, завдяки фреймворку Flutter є також можливість подальшої інтеграції програми на комп'ютери та веб-сайти.

Вартість: вартість місячної підписки становить 0.99$ (~40 грн), вона надає доступ до створення більше ніж 7 задач на день, створення нових робочих просторів, перегляду графіків, і повного доступу до методик.

Підтримка та оновлення: наразі планується стабільна підтримка проєкту, можливість надання фідбеку користувачами та постійне розширення функціоналу додатка з автоматизацією нових методик.

Відгуки та репутація: з 10 тестувальників альфа версії продукту 8 залишились задоволенні функціоналом та додатком в цілому, з відгуків були прохання імплементації нотаток до задач.

Методи експертних оцінок надають можливість виявити важливі інсайти, які складно отримати іншими методами, і допомагають приймати більш обґрунтовані рішення в процесі розробки програмного забезпечення.

Група експертів складається з 6 експертів: двох тестувальників (QA) з IT компанії, психолога з університету ЦДУ імені Володимира Винниченка, Бекенд Розробників, та людей з Sales Department (відділу продаж).

Розробка методології оцінки: метод експертних оцінок реалізовано шляхом проведення інтерв'ю кожному із учасників експертної групи. Також була обрана 10-бальна шкала.

Проведення оцінки: після тижневого використання додатком кожен із експертів під час інтерв'ю оцінив додаток за вище наведеними критеріями, в залежності від спеціальності кожен експерт оцінював певний критерій мобільного додатка. Тестувальники (QA) та Психолог – функціональність, зручність використання (юзабіліті). Бекенд розробники – продуктивність, надійність. Sales Department (відділ продажу) – вартість.

Аналіз зібраних даних: функціональність – 8/10, зручність використання

(юзабіліті) – 9/10, продуктивність – 7/10, надійність - 7/10, вартість – 7/10, технічна підтримка – 6/10.

Визначення привабливості програмного забезпечення: виходячи з результатів інтрев'ю, сильною стороною додатка є зручність використання (юзабіліті) зокрема експерти вказують на приємний та інтуїтивно зрозумілий інтерфейс. Слабкою стороною додатка є технічна підтримка, у зв'язку з прогресом технологій, зокрема штучного інтелекту, експерти мають певні сумніви щодо розвитку продукту у подальшій перспективі. Середній бал становить - 7,3.

Розробка рекомендацій: здебільшого експерти радили додати більше методик, обдумати питання імплементації штучного інтелекту та нових функцій для спільного користування.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності можна застосувати кілька підходів (рисунок 7.3):

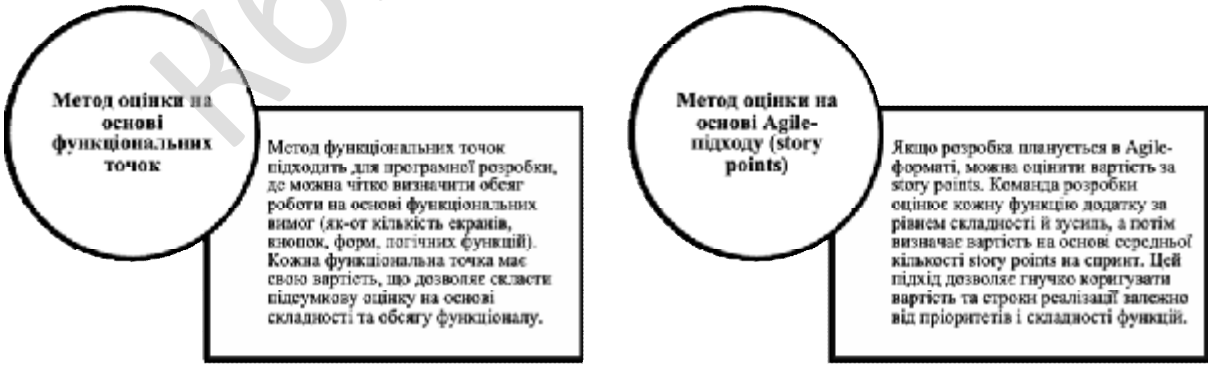

Рисунок 7.3 – Методи оцінки вартості

Вважаючи на специфіку задачі (додаток на основі матриці Ейзенхауера та методик продуктивності), метод функціональних точок або оцінка на основі story

points буде оптимальним варіантом. Ці методи дозволяють деталізувати складність різних функцій (наприклад, функції визначення пріоритетів задач, інтеграція з іншими інструментами продуктивності) і краще оцінити витрати з урахуванням особливостей додатка.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність від впровадження мобільного додатка для управління задачами, заснованого на матриці Ейзенхауера та сучасних методиках продуктивності, може проявлятися у кількох напрямках (рисунок 7.4).

Рисунок 7.4 – Економічна ефективність від реалізації проєкту для клієнта

Додатково, компанія може знизити витрати на інші інструменти та підвищити загальну ефективність команди, що в сумі може забезпечити значний економічний ефект.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

1. Підготовчий етап: аналіз ринку (1 тиждень). Після дослідження основних конкурентів було виділено 3 основних конкуренти. Todoist – це популярний додаток для управління задачами та підвищення продуктивності, який дозволяє користувачам організовувати свої справи, планувати завдання та відстежувати їх виконання. Він використовується для створення списків справ, управління проєктами та організації особистих і робочих процесів.

Trello – це інструмент для управління проєктами та завданнями, який базується на системі канбан-дощок. Він допомагає користувачам організовувати робочий процес, відстежувати прогрес виконання завдань і співпрацювати з іншими учасниками команди. Trello відомий своєю простотою використання та гнучкістю, що дозволяє адаптувати його під різні види проєктів.

JIRA (тепер відома як Jira Software) – це популярний інструмент для управління проєктами та відстеження завдань, створений компанією Atlassian. Спочатку Jira була розроблена для відстеження помилок у програмному забезпеченні, але з часом стала багатофункціональним інструментом для управління проєктами, особливо для команд, які використовують методології Agile (наприклад, Scrum і Kanban).

2. Визначення унікальних торговельних пропозицій. Унікальною торговою пропозицією буде можливість застосування методик продуктивності та відображення задач в матриці Ейзенхауера.

3. Планування стратегій просування. Був обраний маркетинговий канал SEO та SMM. Буде створений блог з інформацією про додаток та новими релізами, описання нового функціоналу додатку та інших новин. Ключові

показники ефективності - Відстеження загальної кількості завантажень через магазини додатків (App Store, Google Play) та Показник реєстрацій у додатку після встановлення.

4. Створення онлайн присутності (2-3 тиждень). Блог буде систематично оновлюватись актуальною інформацією про додаток.

5. Розробка контенту (4 тиждень). Можливий Огляд функціоналу в соціальній мережі TikTok.

6. Контекстна реклама та платні оголошення (4-6 тиждень). Налаштування рекламних кампаній через Google Ads та соціальні мережі.

7. Публікації та партнерства (6-8 тиждень). Наразі ведеться обговорення зі спеціалістами у сфері SMM про співпрацю та можливі колаборації з іншими можливими проєктами та компаніями пов'язаними з продуктивністю.

8. E-mail маркетинг (9 тиждень). Буде розроблено модель email оголошень та невеликих опитувань в самому додатку.

9. Залучення зворотного зв'язку та підтримка (10 тиждень). В додатку буде розроблено окрему сторінку з двома полями: email, та зворотній зв'язок.

10. Оцінка результатів та оптимізація (11-14 тиждень). За першим тестуванням додатка експертною групою, в розробку взято ідею за імплементацію спільного режиму менеджменту задач.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту мобільного додатка для управління задачами можна запропонувати наступні шляхи оптимізації збуту (рисунок 7.5).

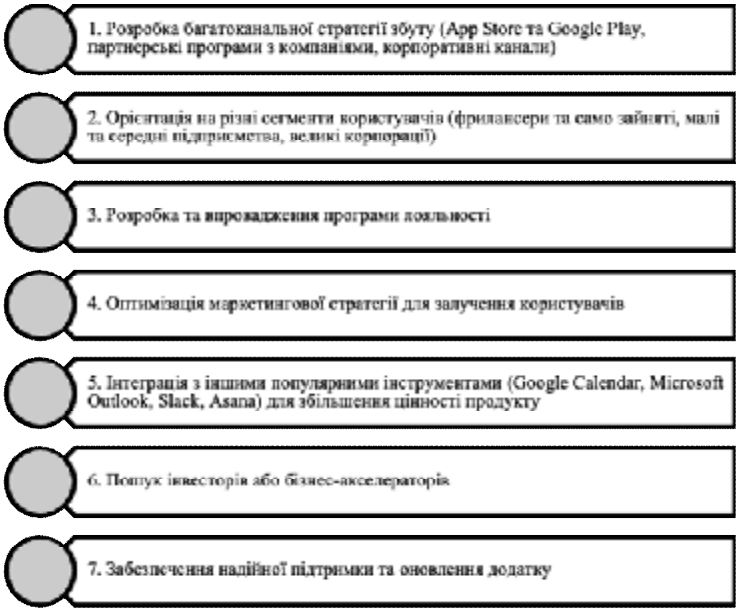

Рисунок 7.5 – Оптимізація каналів збуту

Кожен з цих підходів допоможе збільшити охоплення аудиторії та сприятиме ефективному виведенню продукту на ринок.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проєкту програмної реалізації мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності подано на рисунку 7.6.

Рисунок 7.6 – Ключові фактори успіху проєкту

Фокусування на цих ключових факторах дозволить створити конкурентний продукт, який стане корисним для широкої аудиторії та матиме високі шанси на успіх у користувачів.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1. Вступ

Охорона праці є невід'ємною частиною управління будь-яким виробничим процесом, у тому числі й роботою програмістів. Рівень захисту здоров'я працівників безпосередньо впливає на ефективність та безпеку виконання ними своїх завдань. Основні принципи охорони праці передбачають створення таких умов, що дозволяють працівникам уникнути травмування та захворювань, а також забезпечення їхнього комфортного перебування на робочому місці.

У сфері охорони праці в ІТ-галузі діє низка нормативних актів, які регламентують безпечні умови праці. Ось деякі з основних нормативних актів:

Закон України «Про охорону праці» (№ 2694-XII від 14.10.1992 р.) встановлює загальні положення щодо охорони праці в усіх галузях, зокрема й у сфері ІТ. Він регулює відносини між роботодавцем і працівниками у сфері безпеки праці, обов’язки щодо створення безпечних умов праці, гарантії прав працівників на охорону здоров’я та безпеку на робочому місці. Відповідно до цього закону, роботодавець зобов’язаний забезпечити належний рівень безпеки на робочому місці та створити умови, що сприятимуть збереженню здоров’я працівників.

Державні санітарні норми і правила (СанПіН). Визначають санітарні вимоги до умов праці, включаючи норми освітлення, рівні шуму, вентиляцію, температуру, які є важливими для офісів і робочих місць, де використовуються комп’ютери.

Державні будівельні норми України (ДБН). Містять норми щодо проєктування та облаштування приміщень, де виконуються роботи з використанням комп’ютерної техніки. Наприклад, ДБН В.2.5-28:2006 – норми щодо природного та штучного освітлення, важливі для створення комфортного

робочого середовища.

Накази та постанови Державної служби України з питань праці. Наприклад, Наказ № 15 "Про затвердження Правил охорони праці під час роботи з комп’ютерними системами". Цей документ встановлює вимоги до безпеки під час роботи з комп'ютерною технікою, включаючи ергономічні параметри робочого місця та організацію робочого часу.

ДСТУ ISO 45001:2019 – Система управління охороною здоров’я та безпекою праці. Стандарт, що визначає вимоги до систем управління охороною здоров'я і безпекою праці, які підприємства можуть застосовувати для запобігання нещасним випадкам та забезпечення здорових умов роботи.

Також стаття 43 Конституції України яка встановлює право людини на належні, безпечні і здорові умови праці.

До основних законодавчих актів про охорону праці належать і “Основи законодавства України про охорону здоров’я”, які регулюють суспільні відносини в цій сфері. Їхня мета – сприяти гармонійному розвитку фізичних і духовних сил, забезпечувати високу працездатність та активне довголіття громадян, усувати чинники, що негативно впливають на здоров’я, а також запобігати та знижувати рівень захворюваності, інвалідності та смертності, забезпечуючи спадковість покращення.

Кодекс законів про працю (КЗпП, № 322-VIII від 10.12.71), що регулює трудові відносини всіх працівників, встановлює високий рівень умов праці, всебічну охорону трудових прав працівників, (розділ XI “Охорона праці”).

Ці нормативні акти допомагають забезпечити безпечне середовище для працівників ІТ-галузі та знизити ризики, пов’язані з професійною діяльністю.

У цьому розділі розглянуто сучасні підходи до покращення умов праці програмістів, розрахунки захисного занулення електроустановки та заходи протипожежної профілактики.

8.2. Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Робота програміста характеризується значним навантаженням на зоровий апарат, опорно-руховий апарат, а також можливістю впливу на організм електромагнітного випромінювання. Дотримання санітарно-гігієнічних умов є критичним для забезпечення продуктивної роботи і запобігання професійним захворюванням.

Згідно з ДБН В.2.5-28:2018, рівень освітленості в офісі має бути не нижче 300-500 lx, що досягається за допомогою комбінації природного та штучного освітлення.

Рекомендується використовувати світлодіодні лампи з тепло-білим спектром світла (колірна температура 4000-5000 K).

Мікроклімат робочого приміщення визначається температурою, вологістю та рухом повітря. Для комфортної роботи програмістів рекомендована температура становить 22-24° С, а вологість — 40-60 %.

Робоче місце програміста має відповідати нормативним вимогам, що встановлені законодавством України та міжнародними стандартами. Згідно з ДСанПІН 3.3.2-007-98, основними вимогами до робочих місць з використанням комп'ютерної техніки є забезпечення належного рівня освітлення, оптимального мікроклімату та ергономічності робочого простору. Площа приміщення на одного працівника не повинна бути меншою за 6 м², тому для шести програмістів загальна площа приміщення площею 45 м² відповідає нормам (7.5 м² на людину).

Нормативи мікроклімату регулюються санітарними правилами ДСН 3.3.6.042-99. Приміщення площею 45 м², де працюють 6 програмістів, повинно бути обладнане системою кондиціонування повітря з функцією автоматичного підтримання параметрів температури та вологості.

Шум у робочому приміщенні не повинен перевищувати 50 дБ, що забезпечується звукоізоляцією стін та стелі. Ергономіка робочих місць має

особливе значення, оскільки від правильного налаштування висоти столу, положення монітора та крісла залежить збереження здоров'я хребта і запобігання синдромам зап'ястного каналу.

Забезпечення належних умов праці в ІТ-відділі потребує комплексного підходу, що включає організаційні заходи та використання технічних засобів. Ось основні аспекти такого забезпечення:

Розробка та впровадження політики охорони праці: включає правила використання обладнання, порядок роботи з комп’ютерними системами, регулярні інструктажі з охорони праці.

Графік роботи та перерви: затвердження графіка, який включає регулярні перерви для зменшення напруги зору та втоми.

Проведення інструктажів та навчання: регулярні інструктажі з техніки безпеки та правильного використання обладнання.

Технічні засоби:

Монітори: перевагу слід надавати моніторам із діагоналлю 24–27 дюймів, роздільною здатністю не менше 1920x1080 пікселів (Full HD), а краще – 2560x1440 пікселів (2K) або вище, щоб зменшити навантаження на очі. Моделі повинні мати функцію захисту від синього світла та регулювання яскравості. Наприклад, Dell UltraSharp U2719D.

Системні блоки або ноутбуки: обладнання з процесорами Intel Core i5 або i7, 16+ ГБ оперативної пам’яті, SSD на 512 ГБ або більше для швидкої роботи програм. Наприклад Lenovo ThinkPad X1 Carbon Gen 9 – ноутбук із процессором Intel Core i7.

Клавіатури та миші: ергономічні моделі з регульованими параметрами для зниження напруги м’язів рук. Наприклад Logitech MX Keys Advanced Wireless Illuminated Keyboard - ергономічна клавіатура з підсвіткою, оптимальною для довготривалого набору тексту.

Стільці: офісні крісла з регульованою висотою, підтримкою поперекового відділу хребта, підлокітниками та можливістю нахилу спинки.

Робочі столи: регульовані за висотою столи, що дозволяють працювати як сидячи, так і стоячи, з достатньою площею для розміщення обладнання. Наприклад, IKEA BEKANT – стіл із регулюванням висоти, що дозволяє перемикатися між сидячим і стоячим режимом роботи. Виготовлений з міцних матеріалів, має достатньо місця для розміщення комп'ютера та аксесуарів.

Акустичні панелі: використання панелей, які поглинають звук і знижують рівень шуму до комфортних значень (35–45 дБ). Наприклад, Interface Carpet Tiles - килимові плитки, що мають властивість поглинати звук. Легко встановлюються на підлогу та значно знижують рівень шуму.

Шумопоглинаюче покриття підлоги: килимове покриття або спеціальні панелі для зменшення відбиття звуку. Наприклад, Mitsubishi Electric Lossnay LGH-RVX Series – система вентиляції з рекуперацією тепла, що забезпечує свіжий приплив повітря та відведення вуглекислого газу з робочих приміщень, зберігаючи оптимальний температурний режим. Daikin Ururu Sarara - кондиціонер із функціями зволоження, очищення та регулювання температури повітря. Чудово підходить для підтримки комфортного клімату в офісному приміщенні.

8.3 Розробка заходів з поліпшення охорони праці

Освітлення: для підвищення рівня освітлення необхідно встановити додаткові світильники з регульованою яскравістю, що дозволить співробітникам налаштовувати освітлення під свої потреби. Згідно з ДСанПІН 3.3.2-007-98, використання моніторів з антивідблисковим покриттям сприятиме зменшенню навантаження на очі. Крім того, неякісне або неправильне освітлення може викликати загальну втому і знижувати продуктивність.

Світильники мають мати коефіцієнт кольоропередачі (CRI) понад 80 для більш природного освітлення. Гарно варіантом буде BenQ ScreenBar – настільна лампа, яка кріпиться до монітора, має регульовану яскравість і колірну

температуру, не створює відблисків. Philips Hue Go – мобільна лампа з можливістю налаштування колірної температури від теплого до холодного світла. Luxo L-1 LED Task Light – класична настільна лампа з можливістю регулювання напрямку та яскравості, що підходить для тривалої роботи за комп'ютером.

Для загального освітлення при рівні 500 люксів: 45 м^2×500 люкс = 22500 люменів. 45м^2×300люкс = 13500 люменів.

Таким чином, для загального освітлення приміщення площею 45 м² необхідно забезпечити світильники, що сумарно дають від 13 500 до 22 500 люменів.

Вентиляція і кондиціювання: впровадження системи автоматичного контролю мікроклімату забезпечить підтримку оптимальних умов роботи. Рекомендується встановлення датчиків якості повітря для виявлення підвищеного рівня вуглекислого газу, що буде сигналом для активації системи вентиляції.

Датчики якості повітря: вони дозволяють автоматично виявляти підвищений рівень CO_2 та інших забруднювачів, сигналізуючи про необхідність провітрювання або активуючи вентиляційну систему. Гарним варіантом буде Netatmo Smart Indoor Air Quality Monitor - вимірює температуру, рівень CO_2, вологість та рівень шуму. Awair Element - вимірює температуру, вологість, концентрацію CO_2, леткі органічні сполуки (VOC) та дрібні частинки PM2.5.

Модернізація робочих місць включає надання програмістам крісел з регулюванням висоти і нахилу, що сприяє підтримці правильної постави. Також важливо використовувати столи з можливістю регулювання висоти для роботи стоячи.

Проведення профілактичних оглядів: періодичні медичні огляди для оцінки стану здоров'я працівників і запобігання професійним захворюванням.

Захист від електромагнітного випромінювання: розміщення обладнання з урахуванням безпечних відстаней і застосування екранування моніторів.

8.4 Техніка безпеки та протипожежна профілактика

Електробезпека: електроустановки повинні бути захищені заземленням та зануленням відповідно до вимог ДБН В.2.5-28:2018. Встановлення ПЗВ дозволить уникнути ураження електричним струмом у випадку замикань.

Забезпечення протипожежної безпеки включає застосування сучасних засобів раннього виявлення пожежі, таких як системи димової сигналізації та автоматичні вогнегасники. Приміщення, в якому працюють 6 програмістів, повинно бути обладнане щонайменше одним порошковим вогнегасником ємністю не менше 5 кг. Рекомендовано використовувати вогнегасники типу ВП-5 або аналогічні моделі, які є ефективними для гасіння пожеж класу А (тверді речовини) та класу Е (електрообладнання під напругою).

Елементи приміщення з підвищеною пожежонебезпекою:

Електричні установки та обладнання: проводка, розетки, розподільчі щити та інші електричні прилади, особливо якщо вони знаходяться в поганому стані або перевантажені, є потенційними джерелами пожежі.

Кабелі та проводка: неправильне прокладання кабелів, їх пошкодження або недостатній захист можуть призвести до короткого замикання та пожежі.

Меблі та оздоблення: старі або низькоякісні меблі, особливо виготовлені з легкозаймистих матеріалів, також можуть бути джерелом загрози.

Матеріали оздоблення приміщення з низькою здатністю до займання:

Керамічна плитка: має високу термостійкість і низьку здатність до займання.

Ламінат з вогнестійким покриттям: спеціальні моделі ламінату з вогнестійкими властивостями, які відповідають стандартам протипожежної безпеки.

Металеві панелі або декоративні покриття: не підтримують горіння, чудово підходять для офісів з підвищеними вимогами до безпеки.

Фарби з вогнестійкими властивостями: спеціальні фарби та покриття, які

зменшують горючість поверхонь і не сприяють поширенню вогню.

Меблі з металу та скла: часто використовуються в офісах для забезпечення не тільки естетичного вигляду, але й безпеки.

Слід регулярно проводити навчання з протипожежної безпеки, включаючи інструктаж працівників щодо дій у разі пожежі та використання первинних засобів пожежогасіння. У робочих приміщеннях також має бути чітко позначений план евакуації.

У приміщенні необхідно встановити світлові вказівники для позначення виходів та шляхів евакуації. Згідно з нормативними вимогами, двері евакуаційних виходів повинні відкриватися у напрямку виходу. Підлога та інші поверхні в приміщенні не повинні бути слизькими, щоб уникнути травмувань під час евакуації.

План евакуації має бути розміщений на видному місці та продубльований в електронному форматі для зручного доступу працівників. Він повинен містити вказівки щодо порядку дій у разі виникнення пожежі, шляхів виходу та місць збору.

8.5 Розрахункова частина

Проведемо розрахунок занулення електроустановки потужністю
4 кВт у приміщенні площею 45 м².

Початкові данні :

1. довжина магістрального кабеля − LM , 50м.
2. Коефіцієнт потужності, cosφ=0.85.
3. довжина розгалуження − L , 10м.
4. Напруга мережі, UL =380 В.

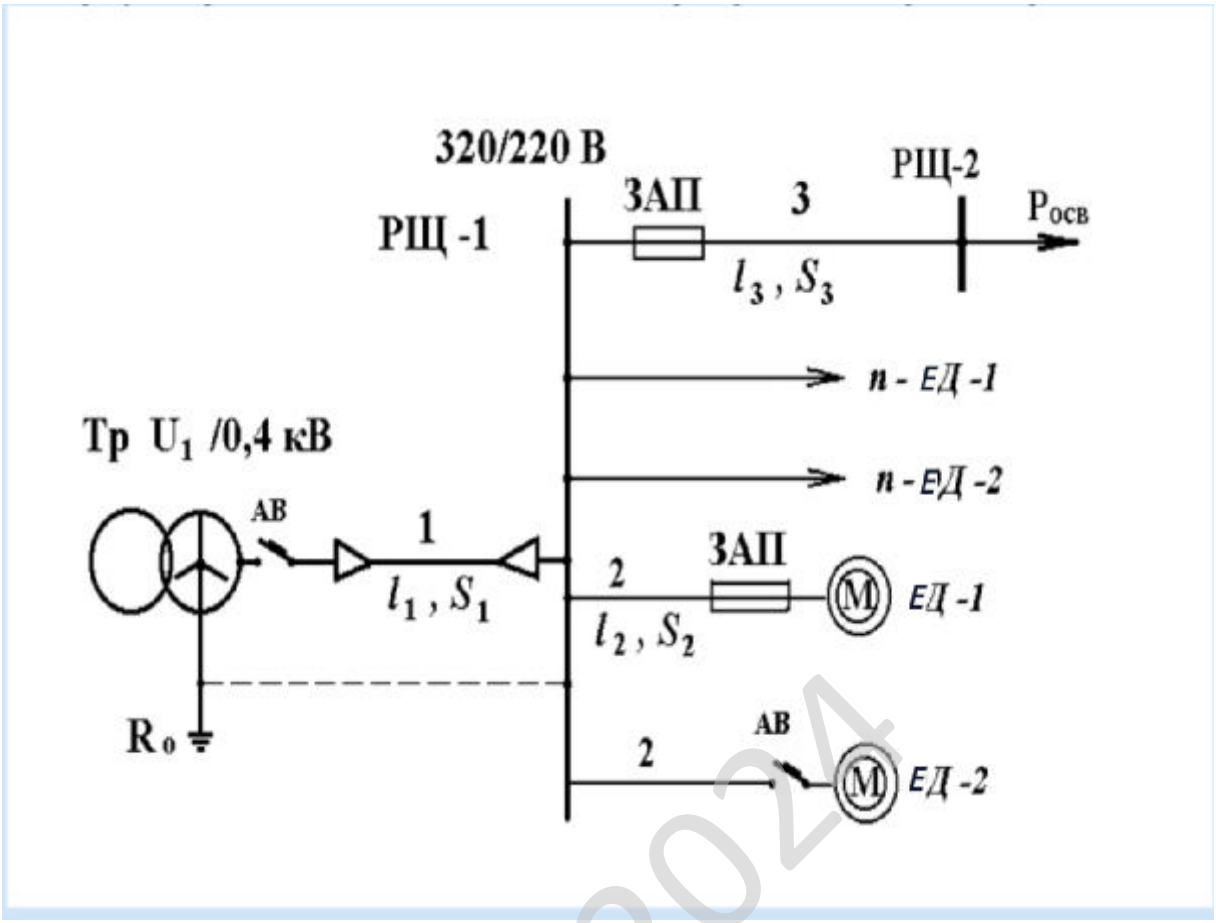

Рисунок 8.1 – Схема електромережі

Розрахунок номінального струму електроустановки

I= P/3·UL·cosφ

де: P=4 кВт — потужність електроустановки, UL=380 В, cosφ=0.85.

Розрахуємо: I=4000/3·380·0.85=4000/558.97≈7.16А.

Розрахунок пускового струму електродвигуна

Пусковий струм визначається як: Іпуск= I * 5.

Розрахуємо: Іпуск= 7.16·5 = 35.8А.

Номінальна сила струму захисного пристрою: Іном=Іпуск/β де β = 2.5β для легких умов пуску.

Розрахуємо: Іном=35.8/2.5=14.32А.

Для цього значення вибираємо автоматичний вимикач із номінальним струмом захисту 16 А, що є стандартним значенням у доступних таблицях.

Мінімальний струм короткого замикання: Ік мін = Іном * К де К=3 для

автоматичного вимикача з інверсно-залежною характеристикою струму.

Ік мін=16·3=48А.

Вибираємо площу перерізу для фазного провідника, щоб він витримував номінальний струм I≈7.16А, і щоб значення тривалого допустимого струму не було меншим за Іном=14.32 А.

Вибираємо переріз 2.5 мм² з таблиці для допустимого струму у алюмінієвих проводах, що забезпечує необхідну міцність для даного навантаження.

Максимальний робочий струм: Іроб=I · Ko · Kз де Ko=0.8 — коефіцієнт одночасності роботи, Kз=0.85 — коефіцієнт завантаження.

Розрахуємо: Іроб=7.16·0.8·0.85≈4.87А.

Переріз нульового захисного провідника обираємо, щоб його площа була не меншою за половину площі фазного провідника: Sн=0.5·Sф де Sф=.5 мм², отже, Sн=0.5·2.5=1.25мм² Округлюючи значення до стандартного, приймаємо Sн=1.5 мм².

Перевірка значення струму короткого замикання

Обчислимо дійсне значення струму однофазного короткого замикання за формулою:

Ікр=Uф/Zпетлі

Де Uф=220В (фазна напруга), а Zпетлі розраховується на основі опорів фазного та нульового провідників. підставимо значення опорів для алюмінію (питомий опір ρ=0.028 Ом·мм²/м), довжина фазного та нульового провідників сумарно складає L=60м: Rпетлі = ρ· L/Sф+Sн = 0.028·60/2.5+1.5=0.672Ω.

Отже: Оскільки Ікр> Імін, обраний переріз проводів є правильним.

Розрахунок проводився відповідно до схеми електромережі.

8.6 Висновки до розділу

Розглянуті заходи щодо охорони праці і техніки безпеки вказують на важливість забезпечення комфортних та безпечних умов праці для програмістів. Застосування рекомендацій з освітлення, мікроклімату, ергономіки, а також організація протипожежного захисту та електробезпеки дозволять значно знизити ризики виробничого травматизму та захворювань.

Проведений розрахунок захисного занулення підтверджує необхідність забезпечення електробезпеки у приміщенні за допомогою сучасних засобів захисту.

9 ОСНОВНІ ВИСНОВКИ

У даній магістерській роботі було досліджено та розроблено мобільний додаток для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.

Додаток включає в себе наступні сторінки та функціонал: Авторизація та реєстрація користувача. Редагування даних користувача. Створення задач. Редагування задач. Управління задачами власноруч та за допомогою методик продуктивності (90/30, Pomodoro). Збереження даних на сервер та в локальну базу даних, синхронізація даних. Локалізація додатка на двох мовах – українська та англійська. Сторінка з днями та статусом кожного дня, з можливістю видалити день. Була також реалізована можливість змінити пароль. Створення графіків виконаних задач за місяць у відсотковому співвідношенні.

Також був реалізований інтерфейс користувача (UI/UX). Бізнес-логіка додатка за архітектурою представлення BLoC у її спрощеній версії cubit, що зменшує складність і дозволяє працювати без необхідності створення подій (Events) і станів (States) за допомогою спеціального StreamController, який є у звичайному BLoC. Cubit використовує лише один тип потоку стану, що дозволяє легше реалізовувати управління станами.

За результатами тестування було підтверджено, що програмне забезпечення працює швидко і стабільно, забезпечуючи надійну передачу повідомлень між користувачами. Також слід зазначити, що розроблене програмне забезпечення для обміну миттєвими повідомленнями на Flutter є відкритим і поширюється за ліцензією GNU General Public License (GPL), що дає можливість користувачам доступити вихідний код і брати участь у подальшому розвитку додатку.

Розроблений додаток дотримується основних принципів об’єктно-орієнтованого програмування, а також SOLID, YAGNI, KISS, DRY.

Це сприятиме підвищенню масштабованості та підтримуваності коду, зменшенню залежності між компонентами застосунку, а також забезпечить його більшу стабільність та зручність у розробці.

У майбутньому планується впровадження нових методик продуктивності, інтеграція пуш-повідомлень, можливість спільного управління задачами та розширення функціоналу для підтримки колективної роботи, включаючи спільні календарі та систему коментарів. Будуть додані аналітичні інструменти для відстеження прогресу та підвищення ефективності використання додатку.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 18144,37 грн (99$ - Розміщення на платформі AppStore: Apple developer account, 25$ - Google Play: Google developer account, 300$ - послуги IT юристів у складані документів: Умов використання, політики конфіденційності та інше, розходи по серверній частині – 14$/міс. У сумі 439$ = ~ 18144,37 грн). З урахуванням вартості розробки програми та обладнання, строк окупності становить 18144.37/4000 ≈ 0.22 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура MVVM у Flutter: підхід для масштабних проектів. URL. https://flutter.dev/docs/development/data-and-backend/state-mgmt/mvvm (date of access: 01.06.2023).

2. Волошин І. П. Ефективні практики управління станом у Flutter. Львів : Видавництво «Технології», 2023. 190 с.

3. Вступ до роботи з анімаціями у Flutter. Офіційна документація Flutter. URL. https://docs.flutter.dev/development/ui/animations/tutorial (date of access: 03.04.2023).

4. Горбань М. В. Поглиблене програмування на мові Dart для мобільних додатків. Одеса : «Техніка і програмування», 2022. 180 с.

5. Дмитрук С. Г. Робота з базами даних у Flutter з використанням Hive та Drift. Київ : Видавництво «Програміст», 2023. 145 с.

6. Жеребецька І. О. Основи анімації в Flutter. Львів : Видавництво «Інновації», 2023. 220 с.

7. Загальні принципи роботи з потоками даних у Dart. URL. https://medium.com/dartlang/understanding-dart-streams-2b123c49424a (date of access: 16.05.2023).

8. Захарченко Д. С. Розробка кросплатформних додатків з використанням Flutter. Львів : «Сучасні технології», 2022. 160 с.

9. Калініченко І. П. Оптимізація додатків у Flutter для роботи з великими даними. Кропивницький : ЦНТУ, 2023. 145 с.

10. Коваленко О. С. Адаптивні інтерфейси користувача у Flutter. Дніпро : «Сучасні технології», 2023. 165 с.

11. Кулик В. А. Застосування Flutter для створення мультиплатформних додатків: методологічні аспекти. Кропивницький : ЦНТУ, 2023. 175 с.

12. Купрій І. В. Розробка інтерактивних додатків на основі Flutter: практичні рекомендації. Київ : Видавництво «Освіта», 2022. 200 с.

13. Методичні вказівки по виконанню розрахунків з курсу "Охорона праці" (Частина 2. Занулення) з використанням персональних ЕОМ IBM−сумісного типу. 2-ге видання, перероблене та доповнене / Кропивницький : ЦНТУ, 2019. 27 с.

14. Моделювання в оцінці та аналізі діяльності підприємства [Електронний ресурс] : монографія / Л. М. Малярець, О. В. Мінєнкова, Л. О. Сабадаш. Харків : ХНЕУ ім. С. Кузнеця, 2018. 202 с.

15. Петров В. Ю. Кращі практики дизайну UI у Flutter. Одеса : «Інноваційні рішення», 2023. 180 с.

16. Пономаренко Ю. В. Flutter для початківців: основи створення інтерфейсів. Харків : Видавництво «Академія», 2022. 130 с.

17. Робота з потоками в Dart: паралельне програмування. Видавництво «Програміст», 2023. 175 с.

18. Сидоренко А. В. Робота зі станом додатку: використання BLoC та Provider у Flutter. Одеса : Видавництво «Технології», 2022. 155 с.

19. Соколова О. М. Архітектура програм на основі Flutter: порівняння методик. Київ : Видавництво «Інновації», 2023. 140 с.

20. Страпчук С. І. Менеджмент: навчальний посібник. Л. : Видавництво «Новий Світ – 2000», 2024. 356 с.

21. Сучасні підходи до архітектури додатків у Flutter. URL. https://flutter.dev/docs/development/data-and-backend/state-mgmt (date of access: 20.06.2023).

22. Шевченко О. В. Основи розробки мобільних додатків на Flutter. Харків : Видавництво «Інжиніринг», 2023. 150 с.

23. Шевчук В. О. Програмне забезпечення мобільного додатку для обміну миттєвими повідомленнями : кваліфікаційна бакалаврська робота : спец. 123 "Комп'ютерна інженерія" / наук. кер. Є. В. Мелешко ; Центральноукраїн. нац.

тех. ун-т. - Кропивницький : ЦНТУ, 2023. 103 с.

24. AboutDialog class – material library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/material/AboutDialog-class.html (date of access: 17.04.2023).

25. Advanced Dart programming concepts: Generators, async-await, and more. URL: https://dart.dev/guides/language/language-tour#generators (date of access: 19.03.2023).

26. AlertDialog class – material library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/material/AlertDialog-class.html (date of access: 29.04.2021).

27. AlignmentGeometry class – painting library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html (date of access: 20.04.2023).

28. A solid guide to SOLID principles | baeldung. Baeldung. URL: https://www.baeldung.com/solid-principles (date of access: 29.04.2021).

29. Basic principles of object-oriented programming. Sitecore | Kentico | Umbraco | nopCommerce & .NET ontwikkeling en interim consultancy, support en training – ParTech. URL: https://www.partech.nl/en/publications/2020/10/basic-principles-of-object-oriented-programming (date of access: 29.04.2022).

30. Codemy.com. Maps in dart – learn dart programming 5, 2022. YouTube. URL: https://www.youtube.com/watch?v=pEhpWxQxLrw (date of access: 29.04.2021).

31. Create a bloc provider. Home | Flutter by Example. URL: https://flutterbyexample.com/lesson/create-a-bloc-provider (date of access: 29.04.2021).

32. Dart Futures: How they work and practical examples. URL. https://dart.dev/codelabs/async-await (date of access: 29.03.2023).

33. Dart programming best practices for large-scale projects. URL. https://medium.com/dartlang/dart-best-practices-for-large-scale-applications-

7e3ec923e97d (date of access: 21.05.2023).

34. Dart programming language. Dart programming language | Dart. URL: https://dart.dev/ (date of access: 26.04.2022).

35. Dart Streams and their practical usage in Flutter apps. URL. https://medium.com/flutter-community/dart-streams-and-their-practical-usage-in-flutter-apps-4f4e3e25dc72 (date of access: 04.05.2023).

36. Dart type system and null safety: an overview. URL: https://dart.dev/null-safety (date of access: 09.04.2023).

37. Dart:ui library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/dart-ui/dart-ui-library.html (date of access: 24.04.2021).

38. Digital Tiger: the Power of Ukrainian IT – 2023. URL. https://mind.ua/publications/20270953-it-industriya-v-cifrah-najcikavishi-daniz-doslidzhennya-digital-tiger

39. Efficient state management in Flutter using BLoC pattern. DevTo. URL https://dev.to/flutter/efficient-state-management-in-flutter-using-bloc-pattern-4hfj (date of access: 12.02.2023).

40. Erinç Y. K. The SOLID principles of object-oriented programming explained in plain english. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/ (date of access: 29.04.2021).

41. Expanded class – widgets library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/widgets/Expanded-class.html (date of access: 20.04.2021).

42. Firebase authentication. Firebase. URL: https://firebase.google.com/docs/auth?hl=ru (date of access: 01.10.2022).

43. Firebase. Firebase. URL. https://firebase.google.com/ (date of access: 27.04.2022).

44. Flutterly. #3 – Flutter BLoC Concepts – BlocProvider, BlocBuilder,

BlocListener | BLoC – from Zero to Hero, 2020. YouTube. URL: https://www.youtube.com/watch?v=NqUx-NfTts4 (date of access: 29.04.2021).

45. Flutter Navigator 2.0 – Advanced navigation patterns. URL. https://medium.com/flutter/navigator-2-0-in-flutter-62d5d019562d (date of access: 22.06.2023).

46. Flutter performance optimization techniques. Flutter Performance Blog. URL. https://flutter.dev/docs/perf (date of access: 15.07.2023).

47. Flutter R. Uncomfortable australia: looking back on xavier herbert's masterwork poor fellow my country. Independently Published, 2019. 161 p.

48. freeCodeCamp.org. How to explain object-oriented programming concepts to a 6-year-old. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/ (date of access: 28.04.2021).

49. Handling user input in Flutter: Form widgets and validation. URL. https://docs.flutter.dev/development/ui/widgets/forms (date of access: 25.03.2023).

50. Injectable | dart package. Dart packages. URL: https://pub.dev/packages/injectable (date of access: 29.04.2022).

51. Integrating RESTful APIs in Flutter applications. URL. https://flutter.dev/docs/cookbook/networking/fetch-data (date of access: 06.07.2023).

52. Introduction to Flutter plugins and how to use them. URL. https://docs.flutter.dev/development/packages-and-plugins/using-packages (date of access: 11.05.2023).

53. Introduction to testing in Flutter using widget tests. Flutter.dev. URL. https://flutter.dev/docs/testing (date of access: 14.03.2023).

54. JAGAN DIGITECH. How to read data from local JSON file and open the next page from ListView | Json Parse in Flutter, 2021. YouTube. URL: https://www.youtube.com/watch?v=Vll8ARb05qY (date of access: 13.04.2022).

55. Json_serializable | dart package. Dart packages. URL: https://pub.dev/packages/json_serializable (date of access: 03.04.2022).

56. ListTile class – material library – Dart API. Flutter – Dart API docs. URL: https://api.flutter.dev/flutter/material/ListTile-class.html (date of access: 12.02.2023).

57. Managing dependencies in Flutter projects using pubspec.yaml. Flutter Community Blog. URL. https://flutter.dev/docs/development/packages-and-plugins/using-packages (date of access: 20.05.2023).

58. Practical use of Flutter's Navigator 2.0 for routing. Flutter Documentation. URL. https://docs.flutter.dev/development/ui/navigation (date of access: 30.03.2023).

59. Proto Coders Point. How to read local json file in flutter & show json data in listview builder, 2021. YouTube. URL: https://www.youtube.com/watch?v=vpFxmFl5cUk (date of access: 21.04.2022).

60. Singleton – Dart API docs. Dart packages. URL. https://pub.dev/documentation/singleton/latest/ (date of access: 29.04.2021).

61. Snyder D. C. Hybrid Project Management. New Jersey : John Wiley & Sons, 2022. 320 p.

62. State management approaches in Flutter – Riverpod vs. Provider. Medium. URL. https://medium.com/flutterdevs/state-management-approaches-in-flutter-riverpod-vs-provider-dc543f9b7323 (date of access: 15.05.2023).

63. The complete guide to widgets in Flutter. URL. https://www.fluttertutorials.com/complete-guide-to-flutter-widgets (date of access: 18.06.2023).

64. The dart programming language. Boston, Massachusetts, USA : Addison-Wesley, 2016. 201 p.

65. The importance of SOLID design principles. BMC Blogs. URL: https://www.bmc.com/blogs/solid-design-principles/#:~:text=SOLID%20is%20an%20acronym%20that,some%20important%20b

enefits%20for%20developers. (date of access: 29.04.2021).

66. The Net Ninja. Firebase authentication tutorial #1 – introduction, 2019. YouTube. URL: https://www.youtube.com/watch?v=aN1LnNq4z54 (date of access: 05.04.2021).

67. Tutorial on advanced custom widgets in Flutter. Flutter University. URL. https://flutter-university.com/tutorials/custom-widgets/ (date of access: 01.04.2023).

68. Understanding the Flutter rendering pipeline. URL. https://medium.com/flutter-community/flutter-rendering-pipeline-overview-3edcf19317e6 (date of access: 29.04.2023).

69. Using Firebase Firestore with Flutter for data storage. URL. https://firebase.flutter.dev/docs/firestore/usage (date of access: 05.03.2023).

70. Utilizing Flutter's built-in accessibility features. Flutter Docs. URL. https://docs.flutter.dev/accessibility (date of access: 02.05.2023).

71. Yazeed AlKhalaf. How to use easy localization package? | flutter, 2021. YouTube. URL: https://www.youtube.com/watch?v=cposNqIsyAY (date of access: 08.01.2023).

72. What is the dart programming language? A guide to its applications. Emeritus Online Courses. URL: https://emeritus.org/blog/coding-dart-programming-language/ (date of access: 28.04.2021).

73. Working with animations in Flutter using the AnimationController class. Flutter Tutorials. URL. https://flutter.dev/docs/development/ui/animations (date of access: 11.04.2023).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності з використанням мови програмування Dart.

4 Джерела розробки

Джерелом цієї магістерської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Вміст проекту

Складовими розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частини системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці розробників програмного забезпечення;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Додаток повинен забезпечувати:

– створення та менеджмент задач за допомогою матриці Ейзенхауера та методик продуктивності;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно містити обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Застосунок, що розробляється повинен використовувати системні та апаратні засоби, які на даному етапі розвитку обчислювальної техніки найбільш поширені.

5.5 Вимоги до надійності

Програмні модулі написані згідно з усіма правилами, які стосуються

стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80 %;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати на Емуляторах Android та пристроях з цією ОС.

5.8 Вимоги до інформаційної та програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Android.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування Dart.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена в вигляді опису структури даних, схем та опису алгоритму, а також текстів вхідних модулів програмного забезпечення у відповідності з ЄСПД.

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 1 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті умови праці програмістів під час розробки програмного забезпечення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 85 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок-схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії структур даних.

10.5 Створення прототипу ПЗ.

10.6 Відлагодження ПЗ, аналіз отриманих результатів.

10.7 Робота над питаннями охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист 07.12.2024 р.

11.2 Подання магістерської роботи на захист 16.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Є.В. Мелешко

Дослідження та програмна реалізація мобільного додатку для управління задачами на основі матриці Ейзенхауера та сучасних методик продуктивності

Лістинг програми

Код документу 12

Носій: USB-флеш-накопи́чувач

Загальна кількість аркушів: 114

Літера: РП

Кропивницький – 2024 року

// main.dart – Базовий файл що містить підключення бази даних, локалізації та віджету що є основою - MatirialApp який знаходиться в StatelessWidget App

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

import 'config/di/di.dart';
import 'config/hive_init.dart';
import 'vinchesta_app.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await initHive();
  await configureDependencies();

  await SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
  ]);

  runApp(
    VinchestaApp.create(),
  );
}
```

// app.dart – файл що містить підключення міжсторінкову маршрутизацію та віджету що є основою - MatirialApp

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:sizer/sizer.dart';
import 'package:vinchesta_story/config/router/app_auto_router.gr.dart';

import 'config/di/locator.dart';
import 'config/router/app_auto_router.dart';
import 'config/themes/light/light_theme.dart';
import 'config/themes/main_theme.dart';
import 'cubit/app_control_cubit/app_control_cubit.dart';
import 'l10n/vinchesta_localizations.dart';

class VinchestaApp extends StatelessWidget {
  static Widget create() {
    return BlocProvider<AppControlCubit>(
      create: (_) => locator<AppControlCubit>(),
      child: const VinchestaApp._(),
    );
  }

  const VinchestaApp._();

  @override
  Widget build(BuildContext context) {
    return BlocBuilder<AppControlCubit, AppControlState>(
```

```
        builder: (context, state) {
      return MainTheme(
        theme: LightThemeData(),
        child: Sizer(
          builder: (context, orientation, deviceType) {
            final appRouter = locator<AppAutoRouter>();
            return MaterialApp.router(
              debugShowCheckedModeBanner: false,
              theme: MainTheme.of(context).themeData,
              locale: state.locale,
              localizationsDelegates:
                  VinchestaLocalizations.localizationsDelegates,
              supportedLocales: VinchestaLocalizations.supportedLocales,
              routeInformationParser: appRouter.defaultRouteParser(),
              routeInformationProvider: appRouter.routeInfoProvider(),
              routerDelegate: appRouter.delegate(
                deepLinkBuilder: ((deepLink) => const DeepLink(
                      kDebugMode ? [LoginRoute()] : [],
                    )),
                navigatorObservers: () => [],
              ),
            );
          },
        ),
      );
    });
  }
}
```

// hive_init.dart – файл що містить ініціалізацію локального середовища

```
import 'package:hive_flutter/hive_flutter.dart';

import '../models/auth_details.dart';

Future<void> initHive() async {
  await Hive.initFlutter();
  Hive.registerAdapter(AuthDetailsAdapter());
}
```

// header_api_interceptor.dart – файл що містить клас для обробки запитів

```
import 'package:dio/dio.dart';
import 'package:injectable/injectable.dart';

import '../../services/preference_service.dart';

@injectable
class HeaderApiInterceptor implements Interceptor {
  final PreferencesService _preferencesService;

  HeaderApiInterceptor(
    this._preferencesService,
  );

  @override
  // ignore: deprecated_member_use
  void onError(DioError err, ErrorInterceptorHandler handler) {
    if (_shouldLogOut(err.response?.statusCode)) {
      // _logoutService.logout();
    } else {
      handler.next(err);
    }
  }

  @override
  void onRequest(
    RequestOptions options,
```

```
    RequestInterceptorHandler handler,
  ) async {
    final accessToken = _preferencesService.getAccessToken();
    if (_preferencesService.isLoggedIn) {
      options.headers['Authorization'] = 'Bearer $accessToken';
    }

    if (options.contentType == Headers.multipartFormDataContentType) {
      options.sendTimeout = const Duration(minutes: 10);
      options.receiveTimeout = const Duration(minutes: 10);
    }

    handler.next(options);
  }

  @override
  void onResponse(Response response, ResponseInterceptorHandler handler) async {
    handler.next(response);
  }

  bool _shouldLogOut(int? statusCode) {
    if (statusCode == null) return false;

    const logOutStatuses = [401];

    return logOutStatuses.contains(statusCode);
  }
}
```

// api_error.dart – файл що містить клас для обробки помилок

```
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';

part 'api_error.g.dart';

@JsonSerializable()
class ApiError extends Equatable {
  final int status;
  final String? detail;
  final String message;

  const ApiError({
    required this.status,
    required this.detail,
    required this.message,
  });

  factory ApiError.fromJson(Map<String, dynamic> json) {
    return _$ApiErrorFromJson(json);
  }

  Map<String, dynamic> toJson() => _$ApiErrorToJson(this);

  @override
  List<Object?> get props {
    return [
      status,
      detail,
      message,
    ];
  }
}
```

// di.dart – файл що містить клас для генерації файлів таких як route, model, service

```
import 'dart:core';

import 'package:injectable/injectable.dart';

//import 'di.config.dart';
import 'di.config.dart';
import 'locator.dart';

@InjectableInit(
  initializerName: r'$configureDependencies',
  preferRelativeImports: true,
  asExtension: false,
)
Future<void> configureDependencies() async {
  await $configureDependencies(locator);
}
```

// locator.dart – файл що містить клас для обробки сінгелтонів

```
import 'package:get_it/get_it.dart';
import
'package:vinchesta_story/cubit/management_page_cubit/management_page_cubit.dart'
;

final locator = GetIt.instance;

extension GetItExt on GetIt {
  Future<void> resetAllLazySingletons() async {
    Future.wait<void>([
      // TODO: reset all LazySingletons
      _resetLazySingleton<ManagementPageCubit>(),
    ]);
  }

  Future<void> _resetLazySingleton<T extends Object>() async {
    if (!isRegistered<T>()) {
      return;
    }
    return await resetLazySingleton<T>();
  }
}
```

// app_auto_router.dart – файл що містить клас з переліком шляхів для генерації сторінок та табок

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:swipeable_page_route/swipeable_page_route.dart';

import 'app_auto_router.gr.dart';

const _durationInMilliseconds = 250;
const _fadeIn = TransitionsBuilders.fadeIn;
const _slideBottom = TransitionsBuilders.slideBottom;
const _slideLeft = TransitionsBuilders.slideLeft;

Route<T> swipeablePageRoute<T>(
  BuildContext _,
  Widget child,
  AutoRoutePage<T> page,
) {
  return SwipeablePageRoute<T>(
    builder: (_) => child,
    settings: page,
    canOnlySwipeFromEdge: true,
    fullscreenDialog: page.fullscreenDialog,
    maintainState: page.maintainState,
```

```
    transitionDuration: const Duration(
      milliseconds: _durationInMilliseconds,
    ),
  );
}

@AutoRouterConfig(
  replaceInRouteName: 'Page|Tab,Route',
)
class AppAutoRouter extends $AppAutoRouter {
  @override
  RouteType get defaultRouteType => const RouteType.custom(
        transitionsBuilder: _fadeIn,
        durationInMilliseconds: _durationInMilliseconds,
      );
  @override
  final List<AutoRoute> routes = [
    AutoRoute(
      page: LoginRoute.page,
    ),
    AutoRoute(
      page: MainRoute.page,
    ),
    AutoRoute(
      page: DaysRoute.page,
    ),
    AutoRoute(
      page: ThingsRoute.page,
    ),
    AutoRoute(
      page: TimePickerRoute.page,
    ),
    AutoRoute(
      page: ManagementRoute.page,
      path: '/management',
      children: [
        AutoRoute(
          page: ToDoRoute.page,
          path: 'toDo',
        ),
        AutoRoute(
          page: InProgressRoute.page,
          path: 'inProgress',
        ),
        AutoRoute(
          page: DoneRoute.page,
          path: 'done',
        ),
      ],
    ),
    AutoRoute(
      page: ThingManageRoute.page,
    ),
    AutoRoute(
      page: RegisterRoute.page,
    ),
    AutoRoute(
      page: ConfirmEmailRoute.page,
    ),
    AutoRoute(
      page: SettingsRoute.page,
    ),
    AutoRoute(
      page: TwentyFiverTimerRoute.page,
    ),
    AutoRoute(
      page: NinetyThirtyTimerRoute.page,
    ),
    AutoRoute(
```

```
      page: ScheduleRoute.page,
    ),
    AutoRoute(
      page: TasksRoute.page,
    ),
    AutoRoute(
      page: ChangePasswordRoute.page,
    ),
  ];
}
```

// base_router.dart – файл що містить клас з загальними методами навігації

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';

abstract class BaseRouter {
  final StackRouter stackRouter;

  const BaseRouter(this.stackRouter);

  // TabsRouter? getTabsRouter(BuildContext context);

  // Future<void> pushRoot();

  // Future<void> replaceWithRoot();

  T? getObserver<T extends AutoRouteObserver>(BuildContext context) {
    return RouterScope.of(context).firstObserverOfType<T>();
  }

  @visibleForTesting
  void popToRoot() => stackRouter.popUntilRoot();

  Future<void> pushByPath(String path) => stackRouter.pushNamed(path);

  Future<void> pop<T>([T? result]) => stackRouter.pop<T>(result);

  @protected
  Future<T?> push<T>(PageRouteInfo<dynamic> route) =>
      stackRouter.push<T>(route);

  // Future<void> popOrRoot() =>
  //     stackRouter.canPopSelfOrChildren ? pop() : pushRoot();

  void popUntilPath(String path) => stackRouter.popUntilRouteWithName(path);

  Future<void> replace(PageRouteInfo<dynamic> route) =>
      stackRouter.replace(route);

  Future<void> replacePath(String path) => stackRouter.replaceNamed(path);

  Future<void> pushAndClearStack(PageRouteInfo<dynamic> route) {
    return stackRouter.pushAndPopUntil(route, predicate: (_) => false);
  }
}
```

```
import 'package:flutter/material.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/api/response/area_response.dart';
import 'package:vinchesta_story/api/response/day_response.dart';

import 'package:vinchesta_story/pages/tasks_page/task_page_dto.dart';

import '../../models/thing.dart';
```

```dart
import '../di/locator.dart';
import 'app_auto_router.dart';
import 'app_auto_router.gr.dart';
import 'base_router.dart';

/// Getter (NOT a final variable) to allow Hot Reloading
/// with new routes without restart
VinchestaRouter get router => locator<VinchestaRouter>();

@singleton
class VinchestaRouter extends BaseRouter {
  VinchestaRouter(AppAutoRouter super.router);

  void pushMainPage() async {
    router.push(const MainRoute());
  }

  void pushDaysPage(AreaResponse areaModel) async {
    router.push(
      DaysRoute(
        areaModel: areaModel,
      ),
    );
  }

  Future<T?> pushTimePickerPage<T extends Object?>({DateTime? time}) {
    return push<T>(
      const TimePickerRoute(),
    );
  }

  void pushManagementPage(DayResponse dayThings) async {
    return router.push(
      ManagementRoute(dayThings: dayThings),
    );
  }

  void resetToLoginPage() async {
    return await pushAndClearStack(
      const LoginRoute(),
    );
  }

  void pushChangePassword() async {
    return router.push(
      const ChangePasswordRoute(),
    );
  }

  Future<T?> pushThingsPage<T extends Object?>({
    Thing? thing,
    String? dayUid,
  }) async {
    return await router.push<T>(
      ThingsRoute(
        thing: thing,
        dayUid: dayUid,
      ),
    );
  }

  void pushRegisterPage() async {
    return router.push(
      const RegisterRoute(),
    );
  }

  void pushConfirmEmailPage() async {
    return router.push(
```

```
      const ConfirmEmailRoute(),
    );
  }

  void pushSettingsPage() async {
    return router.push(
      const SettingsRoute(),
    );
  }

  void pushTwentyFiveTimerPage(VoidCallback onDoneTap) async {
    return router.push(
      TwentyFiverTimerRoute(
        onDoneTap: onDoneTap,
      ),
    );
  }

  void pushNinetyThirtyTimerPage(VoidCallback onDoneTap) async {
    return router.push(
      NinetyThirtyTimerRoute(
        onDoneTap: onDoneTap,
      ),
    );
  }

  void pushChartsPage() async {
    return router.push(
      const ScheduleRoute(),
    );
  }

  Future<void> pushTasksPage({
    required List<Thing> tasks,
    required TaskPageDto taskPageDto,
  }) async {
    print(taskPageDto);
    return router.push(
      TasksRoute(
        tasks: tasks,
        taskPageDto: taskPageDto,
      ),
    );
  }

  Future<void> pushManageThingPage(Thing thing) async {
    return router.push(
      ThingManageRoute(
        thing: thing,
      ),
    );
  }
}
```

// constants.dart – файл що містить клас з константами основних кольорів додатку

```
part of '../light_theme.dart';

const _textButtonLetterSpacing = 0.01;

const _displayLarge = _TextStyle(48, FontWeight.w800, _AppColors.black);
const _displayMedium = _TextStyle(
  42,
  FontWeight.w700,
  _AppColors.black,
  height: 1,
);
const _displaySmall = _TextStyle(30, FontWeight.w500, _AppColors.black);
const _headlineMedium = _TextStyle(22, FontWeight.w500, _AppColors.black);
```

```
const _headlineSmall = _TextStyle(20, FontWeight.w500, _AppColors.black);
const _titleLarge = _TextStyle(16, FontWeight.w400, _AppColors.black);
const _titleMedium = _TextStyle(18, FontWeight.w400, _AppColors.blueDark);
const _titleSmall = _TextStyle(14, FontWeight.w400, _AppColors.blueDark);
const _bodyLarge = _TextStyle(14, FontWeight.w400, _AppColors.black);
const _bodyMedium = _TextStyle(16, FontWeight.w400, _AppColors.blueDark);

const _labelLarge = _TextStyle(
  20,
  FontWeight.w600,
  _AppColors.white,
  letterSpacing: _textButtonLetterSpacing,
);

const _narrowButton = _TextStyle(
  14,
  FontWeight.w400,
  _AppColors.white,
  height: 1.2,
  letterSpacing: 0.52,
);

class _TextStyle extends TextStyle {
  const _TextStyle(
    double size,
    FontWeight weight,
    Color color, {
    super.letterSpacing,
    super.height,
  }) : super(
          fontSize: size,
          fontWeight: weight,
          color: color,
          fontFamily: WybFontFamily.sFProRounded,
        );
}

abstract class _AppColors {
  ///
  /// Red
  ///

  static const redDark = Color(0xFF8B0000);
  static const redRegular = Color(0xFFD66359);
  static const purple = Color(0xFF968bc0);
  static const purpleLight = Color.fromARGB(255, 130, 55, 141);
  // TODO: implement color redLight = Color(0xFFE6958D);

  ///
  /// Green
  ///

  static const greenDark = Color(0xFF497353);
  static const greenRegular = Color(0xFF78bd8e);
  static const greenLight = Color(0xFFb4daa7);
  static const greenPale = Color(0xFFf3f8f0);
  static const greenAccent = Color(0xFFabefac);

  ///
  /// Blue
  ///
  static const blueBlack = Color.fromARGB(255, 86, 85, 110);
  static const blueDark = Color(0xFF2f2e40);
  static const blueLight = Color.fromARGB(255, 72, 71, 94);
  static const blueExtraDark = Color.fromARGB(255, 32, 31, 44);
  static const blueRegular = Color(0xFF007ED5);
  static const blueTurquoiseDark = Color(0xFF3A8F8A);
  static const blueTurquoise = Color(0xFF70DBDB);
  static const blueAccent = Color(0xFF4ba49e);
```

```
  static const blueAccent3 = Color.fromARGB(255, 18, 102, 96);
  static const blueAccent4 = Color.fromARGB(255, 96, 75, 164);
  static const blueAccent5 = Color(0xFF4ba49e);

  ///
  /// Greyscale
  ///

  static const black = Colors.black;
  static const greyDark = Color(0xFFA8A6A6);
  static const grey = Color.fromARGB(255, 205, 202, 202);
  static const greyRegular = Color(0xFFc3c3c3);
  static const white = Colors.white;

  ///
  /// Yellow
  ///
  static const yellow = Color.fromARGB(255, 165, 116, 1);
  static const yellowLight = Color.fromARGB(255, 224, 207, 22);
}

part of '../light_theme.dart';

class _LightColorThemeData implements MainColorThemeData {
  const _LightColorThemeData();

  @override
  Color get textFieldFillColor => _AppColors.blueDark;

  @override
  Color get textFieldDisabledBorder => _AppColors.greyRegular;

  @override
  Color get textFieldEnabledBorder => _AppColors.blueDark;

  @override
  Color get textFieldFocusedBorder => _AppColors.purple;

  @override
  Color get textFieldErrorBorder => _AppColors.redRegular;

  @override
  Color get textFieldFocusedErrorBorder => _AppColors.redRegular;

  @override
  Color get textNoBorderFieldDisabledBorder => _AppColors.greenPale;

  @override
  Color get textNoBorderFieldEnabledBorder => _AppColors.greenPale;

  @override
  Color get textNoBorderFieldFocusedBorder => _AppColors.purple;

  @override
  Color get textNoBorderFieldErrorBorder => _AppColors.redRegular;

  @override
  Color get textNoBorderFieldFocusedErrorBorder => _AppColors.redRegular;

  @override
  Color get authBanner => _AppColors.blueExtraDark;

  @override
  Color get primaryColor => _AppColors.purple;

  @override
  Color get areaIconColor => _AppColors.grey;

  @override
```

```
  Color get dayTileColor => _AppColors.white.withOpacity(0.2);

  @override
  Color get doneProgressColor => _AppColors.greenDark.withOpacity(0.7);

  @override
  Color get inProgressColor => _AppColors.yellow.withOpacity(0.7);

  @override
  Color get toDoColor => _AppColors.blueAccent5.withOpacity(0.7);

  @override
  Color get daysPageDelete => _AppColors.redDark.withOpacity(0.7);

  @override
  Color get daysPageEdit => _AppColors.blueAccent4.withOpacity(0.9);

  @override
  Color get daysPageFore => _AppColors.greyRegular;

  @override
  Color get appBackIcon => _AppColors.greenAccent;

  @override
  Color get appBarBackground => _AppColors.blueLight;

  @override
  Color get appBarDivider => _AppColors.black;

  @override
  Color get appBarShadow => _AppColors.greyDark;

  @override
  Color get vinchestaDropdownSelectedValue =>
      _AppColors.blueAccent5.withOpacity(0.2);

  @override
  Color get errorColor => _AppColors.redDark;

  @override
  Color get backgroundIconColor => _AppColors.blueBlack.withOpacity(0.5);

  @override
  Color get workTimeColor => _AppColors.blueTurquoise;
}
```

// text_theme.dart - файл що містить клас з стилями тексту додатку

```
part of '../light_theme.dart';

class _LightTextThemeData implements MainTextThemeData {
  const _LightTextThemeData();
  @override
  TextStyle get floatingLabelTextStyle => const TextStyle(
        color: _AppColors.white,
        fontWeight: FontWeight.w700,
      );

  @override
  TextStyle get floatingInactiveLabelText => const TextStyle(
        color: _AppColors.greyRegular,
        fontSize: 20,
        fontWeight: FontWeight.w600,
      );

  @override
  TextStyle get primaryFieldText => const TextStyle(
        color: _AppColors.blueDark,
        fontWeight: FontWeight.w700,
```

```
    );

@override
TextStyle get vinchestaTextField => const TextStyle(
      color: _AppColors.white,
    );

@override
TextStyle get labelUnselected => const TextStyle(
      color: _AppColors.greyRegular,
      fontWeight: FontWeight.w600,
    );

@override
TextStyle get loginPageTitle => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.lato().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 32.sp,
      letterSpacing: 28,
    );

@override
TextStyle get loginPageBanner => TextStyle(
      color: _AppColors.grey.withOpacity(0.2),
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w800,
      fontSize: 100.sp,
    );

@override
TextStyle get loginPageBannerStar => TextStyle(
      color: _AppColors.grey.withOpacity(0.2),
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w500,
      fontSize: 64.sp,
    );

@override
TextStyle get mainPageBackground => TextStyle(
      color: _AppColors.blueDark.withOpacity(0.2),
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w500,
      fontSize: 340.sp,
    );

@override
TextStyle get loginPageDescription => TextStyle(
      color: _AppColors.purple,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 12.sp,
    );

@override
TextStyle get loginPageButton => TextStyle(
      color: _AppColors.greyRegular,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 12.sp,
    );

@override
TextStyle get mainPageTitle => TextStyle(
      color: _AppColors.greyDark,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 32.sp,
      letterSpacing: 12,
```

```
    );

@override
TextStyle get loginPageDescription2 => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 12.sp,
    );

@override
TextStyle get mainPageDescription => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 11.sp,
    );

@override
TextStyle get mainPageTileLabel => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 11.sp,
    );

@override
TextStyle get daysPageLabel => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 14.sp,
    );

@override
TextStyle get appBarTitle => TextStyle(
      color: _AppColors.grey,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 18.sp,
    );

@override
TextStyle get vinchestaTextButton => TextStyle(
      color: _AppColors.greenAccent,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 14.sp,
    );

@override
TextStyle get vinchestaTimePickerSelectValue => TextStyle(
      color: _AppColors.greenAccent,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w600,
      fontSize: 15.sp,
    );

@override
TextStyle get vinchestaTimePickerValue => TextStyle(
      color: _AppColors.white,
      fontFamily: GoogleFonts.aBeeZee().fontFamily,
      fontWeight: FontWeight.w500,
      fontSize: 15.sp,
    );

@override
TextStyle get thingTileLabel => TextStyle(
      color: _AppColors.white,
```

```
        fontFamily: GoogleFonts.aBeeZee().fontFamily,
        fontWeight: FontWeight.w500,
        fontSize: 11.sp,
      );

  @override
  TextStyle get thingTitle => TextStyle(
        color: _AppColors.greenAccent,
        fontFamily: GoogleFonts.aBeeZee().fontFamily,
        fontWeight: FontWeight.w600,
        fontSize: 12.sp,
      );

  @override
  TextStyle get thingDate => TextStyle(
        color: _AppColors.grey,
        fontWeight: FontWeight.w500,
        fontSize: 11.sp,
      );

  @override
  TextStyle get deleteThing => TextStyle(
        color: _AppColors.redDark,
        fontFamily: GoogleFonts.aBeeZee().fontFamily,
        fontWeight: FontWeight.w600,
        fontSize: 14.sp,
      );

  @override
  TextStyle get dataChart => TextStyle(
        color: _AppColors.white,
        fontFamily: GoogleFonts.aBeeZee().fontFamily,
        fontWeight: FontWeight.w500,
        fontSize: 9.sp,
      );

  @override
  TextStyle get chartTitle => TextStyle(
        color: _AppColors.greenPale,
        fontFamily: GoogleFonts.aBeeZee().fontFamily,
        fontWeight: FontWeight.w500,
        fontSize: 12.sp,
      );
}
```

// vinchesta_uk.dart – файл що містить Українську локалізацію проекту

```
{
    "@locale": "en",
    "appName": "VinchestaStory",
    "signIn": "Увійти",
    "signUp": "Зареєструватись",
    "title": "Mbook",
    "loginPageDescription": "Ефективно керуйте своїми справами",
    "email": "Електронна Пошта",
    "password": "Пароль",
    "forgotPassword": "Забули пароль?",
    "repeatPassword": "Повторіть пароль",
    "currentPassword": "Поточний пароль",
    "confirmNewPassword": "Підтвердіть новий пароль",
    "newPassword": "Новий пароль",
    "changePasswordCapitalized": "Змінити пароль",
    "forgotPasswordWithSmallLetter": "Забули пароль?",
    "enterCorrectEmailError": "Будь ласка, введіть правильний електронний
адресу",
    "confirmEmail": "Підтвердіть ваш Email",
    "fieldIsRequiredError": "Це поле обов'язкове",
    "forgotPasswordPageError": "Зазначеної електронної адреси не пов'язано з
обліковим записом.",
```

```
    "forgotPasswordPageButtonSubmit": "Надіслати інструкції",
    "passwordRestoredPageTitle": "Пароль успішно скинуто!",
    "bothPasswordsMustMatch": "Обидва паролі повинні співпадати",
    "passwordValidationError": "8 символів, одна велика літера, цифра та
спеціальний символ",
    "passwordValidationNoUppercase": "Пароль повинен містити принаймні 1 велику
літеру",
    "passwordValidationNoDigits": "Пароль повинен містити принаймні 1 цифру",
    "passwordValidationNoSpecials": "Пароль повинен містити принаймні 1
спеціальний символ",
    "passwordValidationNotLongEnough": "Пароль повинен бути принаймні довжиною 8
символів",
    "mainPageAsk": "Плануй",
    "mainPageYour": "Керуй",
    "mainPageQuestion": "Записуй",
    "loginPageStatus": "Статус: ",
    "statusInProgress": "У прогресі",
    "statusToDo": "Виконати",
    "statusDone": "Виконано",
    "express": "Терміново",
    "noExpress": "Не терміново",
    "important": "Важлива",
    "notImportant": "Не важлива",
    "name": "Ім'я",
    "continueButton": "Продовжити",
    "ukraine": "Українська",
    "english": "English",
    "pause": "пауза",
    "play": "відтворити",
    "break1": "Перерва",
    "work": "Робота",
    "ninetyDescription": "Метод 90/30 — це розподіл роботи та відпочинку, згідно
з яким 90 хвилин завдань чергуються з 30-хвилинною перервою.",
    "thirtyDescription": "Метод 25/5 — це розподіл роботи та відпочинку, згідно
з яким 25 хвилин завдань чергуються з 5-хвилинною перервою.",
    "notExpressImportant": "Не термінове \n Важливе",
    "expressNotImportant": "Термінове \n Не Важливе",
    "expressImportant": "Термінове \n Важливе",
    "notExpressNotImportant": "Не термінове \n Не Важливе",
    "ninetyMethod": "90/30 метод",
    "twentyFiveMethod": "25/5 метод",
    "doWith25": "Зробити з 25/5",
    "doWith90": "Зробити з 90/30",
    "moveToDo": "Перемістити в зробити",
    "moveInProgress": "Перемістити на виконнаня",
    "moveToDone": "Перемістити в зробленні",
    "manageTask": "Керування задачею",
    "executionTime": "Запланований час",
    "logTime": "Затраченний час",
    "hMFormat": "г:хв",
    "statuses": "Статуси",
    "editThing": "Редагувати задачу",
    "deleteThing": "Видалити задачу",
    "removeLog": "Видалити запис про час",
    "expressType": "Тип Терміновості: ",
    "importantType": "Тип Важливості: ",
    "progressStatus": "Статус Прогресу: ",
    "titleThing": "Назва *",
    "chooseExpressStatus": "Оберіть експерс тип *",
    "chooseImportantStatus": "Оберіть тип важливості *",
    "workTime": "Робочий час",
    "description": "Опис (Це поле не обов'язкове)",
    "fillInAllRequiredFields": "Заповніть усі обов'язкові поля",
    "createThing": "Створити задачу",
    "selectTime": "Оберіть час",
    "settingsPage": "Сторінка налаштувань",
    "chooseLanguage": "Оберіть мову",
    "subscription": "Підписка",
    "changePassword": "Змінити пароль",
```

```
    "logout": "Вийти",
    "deleteAccount": "Видалити аккаунт",
    "diary": "Щоденник",
    "schedule": "Графік",
    "progressPercentage": "Прогрес у відсотках",
    "chartDescription": "Графік виконаних задач у відсотковому співвідношені",
    "emptyTasksToDoDescription": "У вас немає запланованих справ з цієї
категорії на сьогодні",
    "emptyTasksInProgressDescription": "У вас немає справ в прогресі з цієї
категорії на сьогодні",
    "emptyTasksDoneDescription": "У вас немає виконаних справ з цієї категорії
на сьогодні",
    "remove": "Видалити",
    "open": "Відкрити",
    "tasksAmount": "Кількість задач в цієї катигорії - {tasksAmount}",
    "@tasksAmount": {
        "placeholders": {
            "tasksAmount": {
                "type": "String"
            }
        }
    },
    "priority": "У цієї катигорії \n {priority} пріоритет",
    "@priority": {
        "placeholders": {
            "priority": {
                "type": "String"
            }
        }
    },
    "tasks": "Задачі",
    "withoutDescription": "Без опису",
    "createdAt": "Дата створення: ",
    "functionalityIsNotAvailable": "На жаль, цей функціонал наразі не
доступний",
    "agree": "Ок",
    "logTimeDescription": "Впишіть час який ви щойно витратили на задачу ",
    "logTimeBottonDescription": "Загальний час витрачений на задачу вирахується
автоматично",
    "logTimeDescription2": "Видалення запису про час",
    "logTimeBottonDescription2": "Час витрачений на задачу вирахується
автоматично",
    "hour": "Години",
    "minutes": "Хвилини",
    "hourShort": "г",
    "minutesShort": "хв",
    "changePasswordPage": "Сторінка зміни пароля"
}
```

// vinchesta_en.dart – файл що містить Англійську локалізацію проекту

```
{
  "@locale": "en",
  "appName": "VinchestaStory",
  "signIn": "Sign In",
  "signUp": "Sign Up",
  "title": "Mbook",
  "loginPageDescription": "Manage your affairs effectively",
  "email": "Email",
  "password": "Password",
  "forgotPassword": "Forgot password?",
  "currentPassword": "Current password",
  "repeatPassword": "Repeat password",
  "confirmNewPassword": "Confirm new password",
  "newPassword": "New password",
  "changePasswordCapitalized": "Change Password",
  "forgotPasswordWithSmallLetter": "Forgot password?",
  "enterCorrectEmailError": "Please enter correct email",
  "confirmEmail": "Confirm Email",
```

```
  "fieldIsRequiredError": "Field is required",
  "forgotPasswordPageError": "There isn’t an account associated with that
email.",
  "forgotPasswordPageButtonSubmit": "Send Instructions",
  "passwordRestoredPageTitle": "Password reset successfully!",
  "bothPasswordsMustMatch": "Both passwords must match",
  "passwordValidationError": "8 characters, one capital letter, number, &
special character",
  "passwordValidationNoUppercase": "Password must have at least 1 capital
letter",
  "passwordValidationNoDigits": "Password must have at least 1 digit",
  "passwordValidationNoSpecials": "Password must have at least 1 special
symbol",
  "passwordValidationNotLongEnough": "Password must be at least 8 symbols
length",
  "mainPageAsk": "Plan",
  "mainPageYour": "Manage",
  "mainPageQuestion": "Write",
  "loginPageStatus": "Status: ",
  "statusInProgress": "In progress",
  "statusToDo": "To Do",
  "statusDone": "Done",
  "express": "Express",
  "noExpress": "No express",
  "important": "Important",
  "notImportant": "Not important",
  "name": "Name",
  "continueButton": "Continue",
  "ukraine": "Українська",
  "english": "English",
  "pause": "pause",
  "play": "play",
  "break1": "Break",
  "work": "Work",
  "ninetyDescription": "The 90/30 method is a division of work and rest,
according to which 90 minutes of tasks alternate with a 30-minute break.",
  "thirtyDescription": "The 25/5 method is a division of work and rest,
according to which 25 minutes of tasks alternate with a 5-minute break.",
  "notExpressImportant": "Not Express \n Important",
  "expressImportant": "Express \n Important",
  "expressNotImportant": "Express \n Not Important",
  "notExpressNotImportant": "Not Express \n Not Important",
  "ninetyMethod": "90/30 method",
  "twentyFiveMethod": "25/5 method",
  "doWith25": "Do with 25/5",
  "doWith90": "Do with 90/30",
  "moveToDo": "Move To Do",
  "moveInProgress": "Move in progress",
  "moveToDone": "Move to done",
  "manageTask": "Manage task",
  "executionTime": "Execution time",
  "logTime": "Log time",
  "hMFormat": "h:m",
  "statuses": "Statuses",
  "editThing": "Edit Thing",
  "deleteThing": "Delete Thing",
  "removeLog": "Remove Log",
  "expressType": "Express Type: ",
  "importantType": "Important Type: ",
  "progressStatus": "Progress Status: ",
  "titleThing": "Title *",
  "chooseExpressStatus": "Choose Express Type *",
  "chooseImportantStatus": "Choose Important Status *",
  "workTime": "Work time",
  "description": "Description (This field is not required)",
  "fillInAllRequiredFields": "Fill in all required fields",
  "createThing": "Create Thing",
  "selectTime": "Select Time",
  "settingsPage": "Settings page",
```

```
  "chooseLanguage": "Choose Language",
  "subscription": "Subscription",
  "changePassword": "Change password",
  "logout": "Logout",
  "deleteAccount": "Delete account",
  "diary": "Diary",
  "schedule": "Schedule",
  "progressPercentage": "Progress in percentage",
  "chartDescription": "Schedule of completed tasks in percentage ratio",
  "emptyTasksToDoDescription": "You have no scheduled cases in this category for
today",
  "emptyTasksInProgressDescription": "You have no cases in progress in this
category for today",
  "emptyTasksDoneDescription": "You have no completed tasks in this category for
today",
  "remove": "Remove",
  "open": "Open",
  "tasksAmount": "The number of tasks in this category - {tasksAmount}",
  "@tasksAmount": {
    "placeholders": {
      "tasksAmount": {
        "type": "String"
      }
    }
  },
  "priority": "This category \n has {priority} priority",
  "@priority": {
    "placeholders": {
      "priority": {
        "type": "String"
      }
    }
  },
  "tasks": "Tasks",
  "withoutDescription": "Without description",
  "createdAt": "Creation data:",
  "functionalityIsNotAvailable": "Unfortunately, this functionality is currently
not available",
  "agree": "Ok",
  "logTimeDescription": "Enter the time you just spent on the task",
  "logTimeBottonDescription": "The total time spent on the task is calculated
automatically",
  "logTimeDescription2": "Deleting a time entry",
  "logTimeBottonDescription2": "The time spent on the task is calculated
automatically",
  "hour": "Hours",
  "minutes": "Minutes",
  "hourShort": "h",
  "minutesShort": "m",
  "changePasswordPage": "Change Password Page"
}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:flutter/material.dart';
import 'package:json_annotation/json_annotation.dart';

import 'package:vinchesta_story/models/express_status.dart';
import 'package:vinchesta_story/models/important_status.dart';
import 'package:vinchesta_story/models/progress_status.dart';

import '../utils/json_converters/datetime_converter.dart';
import '../utils/json_converters/time_of_day_converter.dart';

// thing.dart – файл що містить модель задачі

part 'thing.g.dart';

@JsonSerializable(
  fieldRename: FieldRename.snake,
```

```
)
class Thing {
  final String title;
  final String? description;
  final String uid;
  final int userId;
  final String dayUid;
  final ExpressStatus expressStatus;
  final ImportantStatus importantStatus;
  final ProgressStatus progressStatus;
  @DateTimeJsonConverter()
  final DateTime? loggedTime;
  final int? workTime;
  final int? breakTime;
  @TimeOfDayJsonConverter()
  final TimeOfDay? executionTime;
  @DateTimeJsonConverter()
  final DateTime? planingStart;
  @DateTimeJsonConverter()
  final DateTime? planingEnd;
  @DateTimeJsonConverter()
  final DateTime? createdAt;
  @DateTimeJsonConverter()
  final DateTime? updatedAt;

  Thing({
    required this.title,
    this.description,
    required this.uid,
    required this.expressStatus,
    required this.importantStatus,
    required this.progressStatus,
    required this.dayUid,
    required this.userId,
    this.loggedTime,
    this.workTime,
    this.breakTime,
    this.executionTime,
    this.planingStart,
    this.planingEnd,
    this.updatedAt,
    this.createdAt,
  });

  const Thing.empty({
    this.title = '',
    this.uid = '',
    this.expressStatus = ExpressStatus.notExpress,
    this.importantStatus = ImportantStatus.notImportant,
    this.progressStatus = ProgressStatus.toDo,
    this.description,
    this.dayUid = '',
    this.userId = 0,
    this.loggedTime,
    this.workTime,
    this.breakTime,
    this.executionTime,
    this.planingStart,
    this.planingEnd,
    this.updatedAt,
    this.createdAt,
  });

  factory Thing.fromJson(Map<String, dynamic> json) {
    return _$ThingFromJson(json);
  }

  Map<String, dynamic> toJson() => _$ThingToJson(this);
```

```
  TimeOfDay getTimeOfDay() {
    if (breakTime == 0 || breakTime == null) {
      return const TimeOfDay(hour: 0, minute: 0);
    }
    int hours = breakTime! ~/ 60;
    int minutes = breakTime! % 60;
    return TimeOfDay(hour: hours, minute: minutes);
  }

  TimeOfDay getWorkedTimeOfDay() {
    if (workTime == 0 || workTime == null) {
      return const TimeOfDay(hour: 0, minute: 0);
    }
    int hours = workTime! ~/ 60;
    int minutes = workTime! % 60;
    return TimeOfDay(hour: hours, minute: minutes);
  }

  Thing copyWith({
    String? title,
    String? description,
    String? uid,
    int? userId,
    String? dayUid,
    ExpressStatus? expressStatus,
    ImportantStatus? importantStatus,
    ProgressStatus? progressStatus,
    DateTime? loggedTime,
    int? workTime,
    int? breakTime,
    TimeOfDay? executionTime,
    DateTime? planingStart,
    DateTime? planingEnd,
    DateTime? createdAt,
    DateTime? updatedAt,
  }) {
    return Thing(
      title: title ?? this.title,
      description: description ?? this.description,
      uid: uid ?? this.uid,
      userId: userId ?? this.userId,
      dayUid: dayUid ?? this.dayUid,
      expressStatus: expressStatus ?? this.expressStatus,
      importantStatus: importantStatus ?? this.importantStatus,
      progressStatus: progressStatus ?? this.progressStatus,
      loggedTime: loggedTime ?? this.loggedTime,
      workTime: workTime ?? this.workTime,
      breakTime: breakTime ?? this.breakTime,
      executionTime: executionTime ?? this.executionTime,
      planingStart: planingStart ?? this.planingStart,
      planingEnd: planingEnd ?? this.planingEnd,
      createdAt: createdAt ?? this.createdAt,
      updatedAt: updatedAt ?? this.updatedAt,
    );
  }
}
```

// access_response.dart – файл що містить модель токену доступу до даних користувача

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';

part 'access_response.g.dart';

@JsonSerializable(
  fieldRename: FieldRename.snake,
```

```
)
class AccessResponse extends Equatable {
  final String accessToken;

  const AccessResponse({
    required this.accessToken,
  });

  factory AccessResponse.fromJson(Map<String, dynamic> json) {
    return _$AccessResponseFromJson(json);
  }

  Map<String, dynamic> toJson() => _$AccessResponseToJson(this);

  @override
  List<Object?> get props => [accessToken];
}
```

// area_response.dart – файл що містить модель робочого простору

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';

part 'area_response.g.dart';

@JsonSerializable(
  fieldRename: FieldRename.snake,
)
class AreaResponse extends Equatable {
  final String uid;
  final String? title;
  final int userId;
  final String createdAt;
  final String updatedAt;

  const AreaResponse({
    required this.uid,
    this.title,
    required this.userId,
    required this.createdAt,
    required this.updatedAt,
  });

  factory AreaResponse.fromJson(Map<String, dynamic> json) {
    return _$AreaResponseFromJson(json);
  }

  Map<String, dynamic> toJson() => _$AreaResponseToJson(this);

  @override
  List<Object?> get props => [uid, title, userId, createdAt, updatedAt];
}
```

// day_response.dart – файл що містить модель дня

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';
import 'package:vinchesta_story/models/progress_status.dart';

import '../../utils/json_converters/datetime_converter.dart';
import '../../utils/jsonable.dart';

part 'day_response.g.dart';

@JsonSerializable(
  fieldRename: FieldRename.snake,
)
```

```
class DayResponse extends Equatable implements Jsonable {
  final String uid;
  @JsonEnum()
  final ProgressStatus progressStatus;
  final int userId;
  final String areaUid;
  @DateTimeJsonConverter()
  final DateTime createdAt;
  @DateTimeJsonConverter()
  final DateTime updatedAt;

  const DayResponse({
    required this.uid,
    required this.progressStatus,
    required this.userId,
    required this.createdAt,
    required this.updatedAt,
    required this.areaUid,
  });

  factory DayResponse.fromJson(Map<String, dynamic> json) {
    return _$DayResponseFromJson(json);
  }

  @override
  Map<String, dynamic> toJson() => _$DayResponseToJson(this);

  @override
  List<Object?> get props => [
        uid,
        progressStatus,
        userId,
        areaUid,
        createdAt,
        updatedAt,
      ];
}
```

// edit_user_request.dart – файл що містить модель для редагування даних користувача

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';

part 'register_response.g.dart';

@JsonSerializable(
  fieldRename: FieldRename.snake,
)
class RegisterResponse extends Equatable {
  final int id;
  final String name;
  final String password;
  final String email;
  final String language;
  final bool isActive;

  const RegisterResponse({
    required this.id,
    required this.name,
    required this.password,
    required this.email,
    required this.language,
    required this.isActive,
  });

  factory RegisterResponse.fromJson(Map<String, dynamic> json) {
    return _$RegisterResponseFromJson(json);
```

```
  }

  Map<String, dynamic> toJson() => _$RegisterResponseToJson(this);

  @override
  List<Object?> get props => [
        id,
        name,
        password,
        email,
        language,
        isActive,
      ];

  RegisterResponse copyWith({
    int? id,
    String? name,
    String? password,
    String? email,
    String? language,
    bool? isActive,
  }) {
    return RegisterResponse(
      id: id ?? this.id,
      name: name ?? this.name,
      password: password ?? this.password,
      email: email ?? this.email,
      language: language ?? this.language,
      isActive: isActive ?? this.isActive,
    );
  }
}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:equatable/equatable.dart';
import 'package:json_annotation/json_annotation.dart';

part 'edit_user_request.g.dart';

@JsonSerializable()
class EditUserRequest extends Equatable {
  final String language;
  final String name;
  final String email;

  const EditUserRequest({
    required this.language,
    required this.name,
    required this.email,
  });

  EditUserRequest copyWith({
    String? language,
    String? name,
    String? email,
  }) {
    return EditUserRequest(
      language: language ?? this.language,
      name: name ?? this.name,
      email: email ?? this.email,
    );
  }

  factory EditUserRequest.fromJson(Map<String, dynamic> json) {
    return _$EditUserRequestFromJson(json);
  }

  Map<String, dynamic> toJson() => _$EditUserRequestToJson(this);
```

```
  @override
  List<Object?> get props {
    return [
      email,
      language,
      name,
    ];
  }
}
```

// vinchesta_api_client.dart – файл що містить запити на сервер

```
import 'package:dio/dio.dart';
import 'package:injectable/injectable.dart';
import 'package:retrofit/retrofit.dart';
import 'package:vinchesta_story/api/request/edit_user_request.dart';
import 'package:vinchesta_story/api/request/login_request.dart';
import 'package:vinchesta_story/api/request/register_request.dart';

import '../models/thing.dart';
import 'api_constants.dart';
import 'request/area_request.dart';
import 'request/day_request.dart';
import 'response/access_response.dart';
import 'response/area_response.dart';
import 'response/day_response.dart';
import 'response/register_response.dart';

part 'vinchesta_api_client.g.dart';

@RestApi(baseUrl: ApiConstants.baseUrl)
abstract class VinchestaApiClient {
  factory VinchestaApiClient(
    @Named(ApiConstants.baseApiKey) Dio dio, {
    String baseUrl,
  }) = _VinchestaApiClient;

  @POST('/areas/')
  Future<AreaResponse> postArea(
    @Body() AreaRequest areaRequest,
  );

  @GET('/areas/{userId}')
  Future<List<AreaResponse>> getAreas({
    @Path() required int userId,
  });

  @GET('/day_things/{userId}')
  Future<List<DayResponse>> getDays({
    @Path() required int userId,
  });

  @DELETE('/day_things/{dayThingUid}')
  Future<void> removeDays({
    @Path() required String dayThingUid,
  });

  @GET('/things/{dayThingUid}')
  Future<List<Thing>> getTask({
    @Path() required String dayThingUid,
  });

  @POST('/things/')
  Future<Thing> postTask({
    @Body() required Thing thing,
  });

  @PUT('/things/{thingUid}')
  Future<Thing> editTask({
```

```
    @Path() required String thingUid,
    @Body() required Thing thing,
  });

  @DELETE('/things/{thingUid}')
  Future<void> deleteTask({
    @Path() required String thingUid,
  });

  @POST('/day_things/')
  Future<DayResponse> postDay(
    @Body() DayRequest dayRequest,
  );

  @POST('/users/register')
  Future<RegisterResponse> register(
    @Body() RegisterRequest userCreate,
  );

  @POST('/users/login')
  Future<AccessResponse> login(
    @Body() LoginRequest login,
  );

  @GET('/users/me')
  Future<RegisterResponse> getUser();

  @DELETE('/users/{userId}')
  Future<void> removeUser({
    @Path() required int userId,
  });

  @PUT('/users/{userId}')
  Future<void> editUser({
    @Path() required int userId,
    @Body() required EditUserRequest editUserRequest,
  });
}
```

// app_control_cubit.dart – файл що містить загальні методи бізнес логіки додатку

```
import 'package:equatable/equatable.dart';
import 'package:flutter/material.dart';
import 'package:injectable/injectable.dart';
import 'package:jiffy/jiffy.dart';
import 'package:vinchesta_story/cubit/base_cubit.dart';

import '../../services/preference_service.dart';

part 'app_control_state.dart';

@lazySingleton
class AppControlCubit extends BaseCubit<AppControlState> {
  final PreferencesService _preferencesService;

  AppControlCubit(
    this._preferencesService,
  ) : super(AppControlState(locale: _preferencesService.getLocale())) {
    Jiffy.setLocale(_preferencesService.getLocale().languageCode);
  }

  @override
  void handleError(String errorMessage) {
    emit(state.copyWith(errorMessage: errorMessage));
  }

  Future<void> setLocale(Locale? locale) async {
    if (locale == null) return;
```

```
    await makeErrorHandledCall(() async {
      await _preferencesService.setLocale(locale);
      Jiffy.setLocale(locale.languageCode);
      emit(state.copyWith(locale: locale));
    });
  }
}

part of 'app_control_cubit.dart';

@immutable
class AppControlState extends Equatable {
  final Locale locale;
  final String errorMessage;

  const AppControlState({
    required this.locale,
    this.errorMessage = '',
  });

  AppControlState copyWith({
    Locale? locale,
    String? errorMessage,
  }) {
    return AppControlState(
      locale: locale ?? this.locale,
      errorMessage: errorMessage ?? this.errorMessage,
    );
  }

  @override
  List<Object?> get props => [locale, errorMessage];
}
```

// days_page_cubit.dart – файл що містить бізнес логіку сторінки з переліком днів

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:uuid/uuid.dart';
import 'package:vinchesta_story/api/request/day_request.dart';
import 'package:vinchesta_story/models/progress_status.dart';
import 'package:vinchesta_story/services/days_service.dart';

import '../../api/response/day_response.dart';
import '../../services/preference_service.dart';
import '../base_cubit.dart';

part 'days_page_state.dart';

@injectable
class DaysPageCubit extends BaseCubit<DaysPageState> {
  final PreferencesService _preferencesService;
  final DaysService _daysService;
  final String areaUid;

  DaysPageCubit(
    this._daysService,
    this._preferencesService,
    @factoryParam this.areaUid,
  ) : super(const DaysPageState(
          status: DaysPageStatus.idle,
        )) {
    getDays();
  }

  @override
```

```
  void handleError(String errorMessage) {
    emit(DaysPageState(
      status: DaysPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> removeDay(DayResponse day) async {
    return await makeErrorHandledCall(() async {
      emit(state.copyWith(status: DaysPageStatus.loading));
      final days = List.of(state.daysResponse);
      days.remove(day);
      await _daysService.removeDay(day.uid);
      emit(state.copyWith(
        status: DaysPageStatus.success,
        daysResponse: days,
      ));
    });
  }

  Future<void> getDays() async {
    return await makeErrorHandledCall(
      () async {
        emit(state.copyWith(status: DaysPageStatus.loading));
        final uid = const Uuid().v4();
        final userId = _preferencesService.getUserId();
        List<DayResponse> days = await _daysService.getDays();
        if (days.isNotEmpty) {
          days.sort((a, b) => b.createdAt.compareTo(a.createdAt));

          if (days.first.createdAt.toLocal().day != DateTime.now().day ||
              days.first.createdAt.toLocal().month != DateTime.now().month ||
              days.first.createdAt.toLocal().year != DateTime.now().year) {
            final day = await _daysService.postDay(
              DayRequest(
                areaUid: areaUid,
                uid: uid,
                progressStatus: ProgressStatus.toDo,
                userId: userId!,
              ),
            );
            days.insert(0, day);
          }
        } else {
          final day = await _daysService.postDay(
            DayRequest(
              areaUid: areaUid,
              uid: uid,
              progressStatus: ProgressStatus.toDo,
              userId: userId!,
            ),
          );
          days.add(day);
        }
        emit(state.copyWith(
          status: DaysPageStatus.success,
          daysResponse: days,
        ));
      },
    );
  }
}

part of 'days_page_cubit.dart';

enum DaysPageStatus {
  idle,
  loading,
  error,
```

```
  success,
}

class DaysPageState extends Equatable {
  final DaysPageStatus status;
  final String errorMessage;
  final List<DayResponse> daysResponse;

  const DaysPageState({
    required this.status,
    this.errorMessage = '',
    this.daysResponse = const [],
  });

  DaysPageState copyWith({
    required DaysPageStatus status,
    String? errorMessage,
    List<DayResponse>? daysResponse,
  }) {
    return DaysPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      daysResponse: daysResponse ?? this.daysResponse,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      daysResponse,
    ];
  }
}
```

// login_page_cubit.dart – файл що містить бізнес логіку сторінки з авторизацією

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';

import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../api/request/login_request.dart';
import '../base_cubit.dart';

part 'login_page_state.dart';

@injectable
class LoginPageCubit extends BaseCubit<LoginPageState> {
  final UserService userService;

  LoginPageCubit(
    this.userService,
  ) : super(const LoginPageState(
          status: LoginPageStatus.idle,
        ));

  @override
  void handleError(String errorMessage) {
    emit(LoginPageState(
      status: LoginPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> login(LoginRequest loginRequest) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(status: LoginPageStatus.loading));
```

```
      await userService.login(loginRequest);

      emit(state.copyWith(
        status: LoginPageStatus.success,
      ));
    });
  }
}

part of 'login_page_cubit.dart';

enum LoginPageStatus {
  idle,
  loading,
  error,
  success,
}

class LoginPageState extends Equatable {
  final LoginPageStatus status;
  final String errorMessage;
  final bool buttonEnabled;
  final bool buttonPressed;
  final bool accountExist;

  const LoginPageState({
    required this.status,
    this.errorMessage = '',
    this.buttonEnabled = false,
    this.buttonPressed = false,
    this.accountExist = false,
  });

  LoginPageState copyWith({
    required LoginPageStatus status,
    String? errorMessage,
    bool? buttonEnabled,
    bool? buttonPressed,
    bool? accountExist,
  }) {
    return LoginPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      buttonPressed: buttonPressed ?? this.buttonPressed,
      buttonEnabled: buttonEnabled ?? this.buttonEnabled,
      accountExist: accountExist ?? this.accountExist,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      buttonEnabled,
      buttonPressed,
      accountExist,
    ];
  }
}

import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:uuid/uuid.dart';
import 'package:vinchesta_story/api/request/area_request.dart';

import '../../api/response/area_response.dart';
import '../../services/area_service.dart';
import '../../services/preference_service.dart';
```

```
import '../base_cubit.dart';

// main_page_cubit.dart – файл що містить бізнес логіку головної сторінки

part 'main_page_state.dart';

@injectable
class MainPageCubit extends BaseCubit<MainPageState> {
  final AreaService areaService;
  final PreferencesService _preferencesService;

  MainPageCubit(
    this.areaService,
    this._preferencesService,
  ) : super(const MainPageState(
          status: MainPageStatus.loading,
        )) {
    getAreas();
  }

  @override
  void handleError(String errorMessage) {
    emit(MainPageState(
      status: MainPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> getAreas() async {
    return await makeErrorHandledCall(() async {
      emit(state.copyWith(status: MainPageStatus.loading));
      final uid = const Uuid().v4();
      final userId = _preferencesService.getUserId();
      final areas = await areaService.getAreas();
      if (areas.isEmpty) {
        final area = await areaService.postArea(
          AreaRequest(
            uid: uid,
            userId: userId!,
            title: "Your work area",
          ),
        );
        areas.add(area);
      }

      emit(state.copyWith(
        status: MainPageStatus.success,
        areas: areas,
      ));
    });
  }
}

part of 'main_page_cubit.dart';

enum MainPageStatus {
  idle,
  loading,
  error,
  success,
}

class MainPageState extends Equatable {
  final MainPageStatus status;
  final String errorMessage;
  final List<AreaResponse> areas;

  const MainPageState({
    required this.status,
```

```
    this.errorMessage = '',
    this.areas = const [],
  });

  MainPageState copyWith({
    required MainPageStatus status,
    String? errorMessage,
    List<AreaResponse>? areas,
  }) {
    return MainPageState(
      status: status,
      areas: areas ?? this.areas,
      errorMessage: errorMessage ?? this.errorMessage,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      areas,
    ];
  }
}
```

// management_page_cubit.dart – файл що містить бізнес логіку сторінки з матрицею Ейзенхаура

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/models/progress_status.dart';
import 'package:vinchesta_story/services/task_service.dart';

import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../config/di/locator.dart';
import '../../models/thing.dart';
import '../base_cubit.dart';

part 'management_page_state.dart';

@lazySingleton
class ManagementPageCubit extends BaseCubit<ManagementPageState> {
  final UserService userService;
  final TaskService taskService;

  ManagementPageCubit(
    this.userService,
    this.taskService,
  ) : super(const ManagementPageState(
          status: ManagementPageStatus.loading,
        ));

  @override
  void handleError(String errorMessage) {
    emit(ManagementPageState(
      status: ManagementPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  @override
  Future<void> close() async {
    locator.resetLazySingleton<ManagementPageCubit>();
    super.close();
  }

  Future<void> getTasksAndSort(String dayUid) async {
```

```
    await makeErrorHandledCall(() async {
      emitIfNotClosed(state.copyWith(
        status: ManagementPageStatus.loading,
      ));
      final tasks = await taskService.getTask(dayUid);
      final toDo = tasks
          .where((thing) => thing.progressStatus == ProgressStatus.toDo)
          .toList();

      final inProgress = tasks
          .where((thing) => thing.progressStatus == ProgressStatus.inProgress)
          .toList();

      final done = tasks
          .where((thing) => thing.progressStatus == ProgressStatus.done)
          .toList();

      emitIfNotClosed(state.copyWith(
        done: done,
        inProgress: inProgress,
        toDo: toDo,
        status: ManagementPageStatus.success,
      ));
    });
  }
}

part of 'management_page_cubit.dart';

enum ManagementPageStatus {
  idle,
  loading,
  error,
  success,
}

class ManagementPageState extends Equatable {
  final ManagementPageStatus status;
  final String errorMessage;
  final List<Thing> toDo;
  final List<Thing> inProgress;
  final List<Thing> done;

  const ManagementPageState({
    required this.status,
    this.errorMessage = '',
    this.toDo = const [],
    this.inProgress = const [],
    this.done = const [],
  });

  ManagementPageState copyWith({
    required ManagementPageStatus status,
    String? errorMessage,
    List<Thing>? toDo,
    List<Thing>? inProgress,
    List<Thing>? done,
  }) {
    return ManagementPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      toDo: toDo ?? this.toDo,
      inProgress: inProgress ?? this.inProgress,
      done: done ?? this.done,
    );
  }

  @override
  List<Object> get props {
```

```
    return [
      status,
      errorMessage,
      toDo,
      inProgress,
      done,
    ];
  }
}
```

// register_page_cubit.dart – файл що містить бізнес логіку сторінки з реєстрацією користувача

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/api/request/register_request.dart';
import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../services/area_service.dart';
import '../../services/preference_service.dart';
import '../base_cubit.dart';

part 'register_page_state.dart';

@injectable
class RegisterPageCubit extends BaseCubit<RegisterPageState> {
  final UserService userService;
  final PreferencesService _preferencesService;
  final AreaService areaService;

  RegisterPageCubit(
    this.userService,
    this.areaService,
    this._preferencesService,
  ) : super(const RegisterPageState(
          status: RegisterPageStatus.idle,
        ));

  @override
  void handleError(String errorMessage) {
    emit(RegisterPageState(
      status: RegisterPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> register(RegisterRequest registerRequest) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(status: RegisterPageStatus.loading));
      await userService.register(registerRequest);

      emit(state.copyWith(
        status: RegisterPageStatus.success,
      ));
    });
  }
}


part of 'register_page_cubit.dart';

enum RegisterPageStatus {
  idle,
  loading,
  error,
  success,
}

class RegisterPageState extends Equatable {
```

```
  final RegisterPageStatus status;
  final String errorMessage;
  final bool buttonEnabled;
  final bool buttonPressed;
  final bool accountExist;

  const RegisterPageState({
    required this.status,
    this.errorMessage = '',
    this.buttonEnabled = false,
    this.buttonPressed = false,
    this.accountExist = false,
  });

  RegisterPageState copyWith({
    required RegisterPageStatus status,
    String? errorMessage,
    bool? buttonEnabled,
    bool? buttonPressed,
    bool? accountExist,
  }) {
    return RegisterPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      buttonPressed: buttonPressed ?? this.buttonPressed,
      buttonEnabled: buttonEnabled ?? this.buttonEnabled,
      accountExist: accountExist ?? this.accountExist,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      buttonEnabled,
      buttonPressed,
      accountExist,
    ];
  }
}
```

// settings_page_cubit.dart – файл що містить бізнес логіку сторінки з налаштуваннями користувача

```
import 'package:equatable/equatable.dart';
import 'package:flutter/material.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/api/request/edit_user_request.dart';
import 'package:vinchesta_story/cubit/app_control_cubit/app_control_cubit.dart';
import 'package:vinchesta_story/services/preference_service.dart';

import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../api/response/register_response.dart';
import '../../config/di/locator.dart';
import '../base_cubit.dart';

part 'settings_page_state.dart';

@injectable
class SettingsPageCubit extends BaseCubit<SettingsPageState> {
  final UserService userService;
  final PreferencesService _preferencesService;
  final AppControlCubit _appControlCubit;

  SettingsPageCubit(
    this.userService,
    this._appControlCubit,
```

```
    this._preferencesService,
  ) : super(const SettingsPageState(
          status: SettingsPageStatus.idle,
        )) {
    init();
  }

  @override
  void handleError(String errorMessage) {
    emit(SettingsPageState(
      status: SettingsPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> init() async {
    emit(state.copyWith(
      status: SettingsPageStatus.loading,
    ));
    await makeErrorHandledCall(() async {
      final user = await userService.getUser();
      emit(state.copyWith(
        status: SettingsPageStatus.idle,
        user: user.copyWith(
          language: _preferencesService.getNullbleLocale()?.languageCode ??
              user.language,
        ),
      ));
    });
  }

  Future<void> changeLanguage(Locale locale) async {
    emit(state.copyWith(
      status: SettingsPageStatus.loading,
    ));
    await _appControlCubit.setLocale(locale);
    emit(state.copyWith(
      status: SettingsPageStatus.idle,
      user: state.user?.copyWith(
        language: locale.languageCode,
      ),
    ));
  }

  Future<void> delete() async {
    emit(state.copyWith(
      status: SettingsPageStatus.loading,
    ));
    final userId = state.user?.id;
    if (userId != null) {
      await userService.deleteUser(userId);
      await _preferencesService.logout();
      await locator.resetAllLazySingletons();
      emit(state.copyWith(
        status: SettingsPageStatus.logout,
      ));
    }
  }

  Future<void> logout() async {
    emit(state.copyWith(
      status: SettingsPageStatus.loading,
    ));
    await _preferencesService.logout();
    await locator.resetAllLazySingletons();
    emit(state.copyWith(
      status: SettingsPageStatus.logout,
    ));
  }
```

```
  Future<void> editUser(EditUserRequest editUserRequest, int userId) async {
    emit(state.copyWith(
      status: SettingsPageStatus.loading,
    ));
    await makeErrorHandledCall(() async {
      await userService.editUser(editUserRequest, userId);
    });
    emit(state.copyWith(
      status: SettingsPageStatus.success,
    ));
  }
}

part of 'settings_page_cubit.dart';

enum SettingsPageStatus {
  idle,
  loading,
  error,
  success,
  logout,
}

// TODO: user.empty
class SettingsPageState extends Equatable {
  final SettingsPageStatus status;
  final String errorMessage;
  final RegisterResponse? user;

  const SettingsPageState({
    required this.status,
    this.errorMessage = '',
    this.user,
  });

  SettingsPageState copyWith({
    required SettingsPageStatus status,
    String? errorMessage,
    RegisterResponse? user,
  }) {
    return SettingsPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      user: user ?? this.user,
    );
  }

  @override
  List<Object?> get props {
    return [
      status,
      errorMessage,
      user,
    ];
  }
}
```

// task_page_cubit.dart – файл що містить бізнес логіку сторінки переліку задач певної категорії

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/pages/tasks_page/task_page_dto.dart';

import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../models/thing.dart';
import '../../services/task_service.dart';
```

```
import '../base_cubit.dart';

part 'task_page_state.dart';

@injectable
class TaskPageCubit extends BaseCubit<TaskPageState> {
  final UserService userService;
  final TaskService taskService;
  final TaskPageDto _taskPageDto;
  final List<Thing> _tasks;

  TaskPageCubit(
    this.userService,
    this.taskService,
    @factoryParam this._taskPageDto,
    @factoryParam this._tasks,
  ) : super(TaskPageState(
          status: TaskPageStatus.idle,
          tasks: _tasks,
        ));

  @override
  void handleError(String errorMessage) {
    emit(TaskPageState(
      status: TaskPageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> deleteTask(Thing thing) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(
        status: TaskPageStatus.loading,
      ));
      await taskService.deleteTask(thing.uid);
      final tasks = List.of(state.tasks);
      tasks.remove(thing);
      emit(state.copyWith(
        status: TaskPageStatus.idle,
        tasks: tasks,
      ));
    });
  }

  Future<void> getCurrentTasks() async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(
        status: TaskPageStatus.loading,
      ));
      final newTasks = await taskService.getTask(_taskPageDto.dayUid);
      final sortedTask = newTasks
          .where((thing) =>
              thing.expressStatus == _taskPageDto.expressStatus &&
              thing.importantStatus == _taskPageDto.importantStatus &&
              thing.progressStatus == state.tasks.first.progressStatus)
          .toList();
      emit(state.copyWith(
        status: TaskPageStatus.idle,
        tasks: sortedTask,
      ));
    });
  }
}

part of 'task_page_cubit.dart';

enum TaskPageStatus {
  idle,
  loading,
```

```
  error,
  success,
}

class TaskPageState extends Equatable {
  final TaskPageStatus status;
  final String errorMessage;
  final List<Thing> tasks;

  const TaskPageState({
    required this.status,
    this.errorMessage = '',
    this.tasks = const [],
  });

  TaskPageState copyWith({
    required TaskPageStatus status,
    String? errorMessage,
    List<Thing>? tasks,
  }) {
    return TaskPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      tasks: tasks ?? this.tasks,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      tasks,
    ];
  }
}
```

// thing_page_cubit.dart – файл що містить бізнес логіку сторінки управління задачею

```
import 'package:equatable/equatable.dart';
import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/services/task_service.dart';

import 'package:vinchesta_story/services/user_service.dart/user_service.dart';

import '../../models/thing.dart';
import '../base_cubit.dart';

part 'thing_manage_page_state.dart';

@injectable
class ThingManagePageCubit extends BaseCubit<ThingManagePageState> {
  final UserService userService;
  final TaskService taskService;
  final Thing thing;

  ThingManagePageCubit(
    this.userService,
    this.taskService,
    @factoryParam this.thing,
  ) : super(const ThingManagePageState(
          status: ThingManagePageStatus.idle,
        )) {
    emit(ThingManagePageState(
      status: ThingManagePageStatus.idle,
      task: thing,
    ));
  }
```

```
  @override
  void handleError(String errorMessage) {
    emit(ThingManagePageState(
      status: ThingManagePageStatus.error,
      errorMessage: errorMessage,
    ));
  }

  Future<void> removeTask(Thing task) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(
        status: ThingManagePageStatus.loading,
      ));
      await taskService.deleteTask(task.uid);
      emit(state.copyWith(
        status: ThingManagePageStatus.removed,
      ));
    });
  }

  Future<void> updateTask(Thing thing) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(
        status: ThingManagePageStatus.loading,
      ));
      emit(state.copyWith(
        status: ThingManagePageStatus.idle,
        task: thing,
      ));
    });
  }

  Future<void> editThing(Thing task) async {
    await makeErrorHandledCall(() async {
      emit(state.copyWith(
        status: ThingManagePageStatus.loading,
      ));
      final newTask = await taskService.editTask(task);
      emit(state.copyWith(
        status: ThingManagePageStatus.idle,
        task: newTask,
      ));
    });
  }
}

part of 'thing_manage_page_cubit.dart';

enum ThingManagePageStatus {
  idle,
  loading,
  error,
  success,
  removed,
}

class ThingManagePageState extends Equatable {
  final ThingManagePageStatus status;
  final String errorMessage;
  final Thing task;

  const ThingManagePageState({
    required this.status,
    this.errorMessage = '',
    this.task = const Thing.empty(),
  });

  ThingManagePageState copyWith({
```

```
    required ThingManagePageStatus status,
    String? errorMessage,
    Thing? task,
  }) {
    return ThingManagePageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      task: task ?? this.task,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      task,
    ];
  }
}
```

// thing_page_cubit.dart – файл що містить бізнес логіку сторінки створення задачі

```
part of 'thing_manage_page_cubit.dart';

enum ThingManagePageStatus {
  idle,
  loading,
  error,
  success,
  removed,
}

class ThingManagePageState extends Equatable {
  final ThingManagePageStatus status;
  final String errorMessage;
  final Thing task;

  const ThingManagePageState({
    required this.status,
    this.errorMessage = '',
    this.task = const Thing.empty(),
  });

  ThingManagePageState copyWith({
    required ThingManagePageStatus status,
    String? errorMessage,
    Thing? task,
  }) {
    return ThingManagePageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      task: task ?? this.task,
    );
  }

  @override
  List<Object> get props {
    return [
      status,
      errorMessage,
      task,
    ];
  }
}

part of 'thing_page_cubit.dart';
```

```
enum ThingPageStatus {
  idle,
  loading,
  error,
  success,
}

class ThingPageState extends Equatable {
  final ThingPageStatus status;
  final String errorMessage;
  final int? userId;

  const ThingPageState({
    required this.status,
    this.errorMessage = '',
    this.userId,
  });

  ThingPageState copyWith({
    required ThingPageStatus status,
    String? errorMessage,
    int? userId,
  }) {
    return ThingPageState(
      status: status,
      errorMessage: errorMessage ?? this.errorMessage,
      userId: userId ?? this.userId,
    );
  }

  @override
  List<Object?> get props {
    return [
      status,
      errorMessage,
      userId,
    ];
  }
}
```

// base_cubit.dart - файл що містить абстрактний класс бізнес логіки для спадкування з базовими методам

```
import 'package:flutter/foundation.dart';
import 'package:flutter/services.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:logger/logger.dart';

import '../../config/di/locator.dart';
import '../../exceptions/api_exception.dart';

import '../config/lost_connection_exeption.dart';

abstract class BaseCubit<TState> extends Cubit<TState> {
  BaseCubit(super.initialState);

  void handleError(String errorMessage);

  Logger get logger => locator<Logger>();

  void emitIfNotClosed(TState state) {
    if (!isClosed) {
      emit(state);
    }
  }

  Future<void> makeErrorHandledCall(AsyncCallback callback) async {
    try {
      await callback();
```

```
    } on PlatformException catch (exception, stackTrace) {
      logger.e(
        'PlatformException in makeErrorHandledCall()',
        error: exception,
        stackTrace: stackTrace,
      );
      handleError(exception.toString());
    } on ApiException catch (exception, stackTrace) {
      logger.e(
        'ApiException in makeErrorHandledCall()',
        error: exception,
        stackTrace: stackTrace,
      );
      handleError(exception.toString());
    } on LostConnectionException catch (exception, stackTrace) {
      logger.e(
        'Lost connection error in makeErrorHandledCall()',
        error: exception,
        stackTrace: stackTrace,
      );
      handleError(exception.toString());
    } catch (exception, stackTrace) {
      logger.e(
        'Error in makeErrorHandledCall()',
        error: exception,
        stackTrace: stackTrace,
      );
      handleError(exception.toString());
    }
  }
}
```

// change_password_page.dart – файл що містить сторінку заміни пароля

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/utils/input_validators.dart';
import 'package:vinchesta_story/views/fileds/vinchesta_text_filed.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';

import '../../l10n/vinchesta_localizations.dart';

enum _ChangePasswordPageFields {
  password,
  newPassword,
  repeatPassword,
}

@RoutePage()
class ChangePasswordPage extends StatefulWidget {
  const ChangePasswordPage({super.key});

  @override
  State<ChangePasswordPage> createState() => _ChangePasswordPageState();
}

class _ChangePasswordPageState extends State<ChangePasswordPage> {
  final _fbKey = GlobalKey<FormBuilderState>();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};
  bool _obcurseText = true;

  String? _confirmedPasswordValidator(
      String? value, VinchestaLocalizations localizations) {
    final validator = confirmedPasswordValidator(
```

```
    localizations,
    getPassword: () => _fbValues[_ChangePasswordPageFields.newPassword.name],
  );
  return validator.call(value);
}

void _onButtonTap() {
  if (_fbState?.saveAndValidate() ?? false) {
    // TODO implement password changes
    context.toasts.showError(
      message: context.strings.functionalityIsNotAvailable,
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: VinchestaAppBar(
      title: context.strings.changePasswordPage,
    ),
    body: FormBuilder(
      key: _fbKey,
      child: Container(
        constraints: BoxConstraints(
          minHeight: MediaQuery.of(context).size.height,
        ),
        decoration: BoxDecoration(
          gradient: context.gradient.primaryGradient,
        ),
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(
              horizontal: 24,
              vertical: 12,
            ),
            child: Column(
              children: [
                const SizedBox(height: 24),
                VinchestaTextField(
                  name: _ChangePasswordPageFields.password.name,
                  labelText: context.strings.password,
                  labelStyle: context.text.vinchestaTextField,
                  validator: passwordValidator(context.strings),
                  suffix: IconButton(
                    onPressed: () {
                      setState(() {
                        _obcurseText = !_obcurseText;
                      });
                    },
                    icon: Icon(
                      _obcurseText ? Icons.visibility_off : Icons.visibility,
                    ),
                  ),
                  obscureText: _obcurseText,
                ),
                const SizedBox(height: 24),
                VinchestaTextField(
                  labelText: context.strings.newPassword,
                  labelStyle: context.text.vinchestaTextField,
                  name: _ChangePasswordPageFields.newPassword.name,
                  validator: passwordValidator(context.strings),
                  suffix: IconButton(
                    onPressed: () {
                      setState(() {
                        _obcurseText = !_obcurseText;
                      });
                    },
                    icon: Icon(
```

```
                        _obcurseText ? Icons.visibility_off : Icons.visibility,
                      ),
                    ),
                    obscureText: _obcurseText,
                  ),
                  const SizedBox(height: 24),
                  VinchestaTextField(
                    labelText: context.strings.repeatPassword,
                    labelStyle: context.text.vinchestaTextField,
                    name: _ChangePasswordPageFields.repeatPassword.name,
                    validator: (value) => _confirmedPasswordValidator(
                      value,
                      context.strings,
                    ),
                    suffix: IconButton(
                      onPressed: () {
                        setState(() {
                          _obcurseText = !_obcurseText;
                        });
                      },
                      icon: Icon(
                        _obcurseText ? Icons.visibility_off : Icons.visibility,
                      ),
                    ),
                    obscureText: _obcurseText,
                  ),
                  const SizedBox(height: 44),
                  VinchestaButton(
                    content: Text(
                      context.strings.changePassword,
                      style: context.text.loginPageButton,
                    ),
                    onPressed: () {
                      _onButtonTap();
                    },
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

// change_password_page.dart – файл що містить сторінку заміни пароля

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/utils/input_validators.dart';
import 'package:vinchesta_story/views/fileds/vinchesta_text_filed.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';

import '../../l10n/vinchesta_localizations.dart';

enum _ChangePasswordPageFields {
  password,
  newPassword,
  repeatPassword,
}

@RoutePage()
class ChangePasswordPage extends StatefulWidget {
  const ChangePasswordPage({super.key});
```

```
  @override
  State<ChangePasswordPage> createState() => _ChangePasswordPageState();
}

class _ChangePasswordPageState extends State<ChangePasswordPage> {
  final _fbKey = GlobalKey<FormBuilderState>();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};
  bool _obcurseText = true;

  String? _confirmedPasswordValidator(
      String? value, VinchestaLocalizations localizations) {
    final validator = confirmedPasswordValidator(
      localizations,
      getPassword: () => _fbValues[_ChangePasswordPageFields.newPassword.name],
    );
    return validator.call(value);
  }

  void _onButtonTap() {
    if (_fbState?.saveAndValidate() ?? false) {
      // TODO implement password changes
      context.toasts.showError(
        message: context.strings.functionalityIsNotAvailable,
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: VinchestaAppBar(
        title: context.strings.changePasswordPage,
      ),
      body: FormBuilder(
        key: _fbKey,
        child: Container(
          constraints: BoxConstraints(
            minHeight: MediaQuery.of(context).size.height,
          ),
          decoration: BoxDecoration(
            gradient: context.gradient.primaryGradient,
          ),
          child: SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.symmetric(
                horizontal: 24,
                vertical: 12,
              ),
              child: Column(
                children: [
                  const SizedBox(height: 24),
                  VinchestaTextField(
                    name: _ChangePasswordPageFields.password.name,
                    labelText: context.strings.password,
                    labelStyle: context.text.vinchestaTextField,
                    validator: passwordValidator(context.strings),
                    suffix: IconButton(
                      onPressed: () {
                        setState(() {
                          _obcurseText = !_obcurseText;
                        });
                      },
                      icon: Icon(
                        _obcurseText ? Icons.visibility_off : Icons.visibility,
                      ),
                    ),
                    obscureText: _obcurseText,
                  ),
```

```
                    const SizedBox(height: 24),
                    VinchestaTextField(
                      labelText: context.strings.newPassword,
                      labelStyle: context.text.vinchestaTextField,
                      name: _ChangePasswordPageFields.newPassword.name,
                      validator: passwordValidator(context.strings),
                      suffix: IconButton(
                        onPressed: () {
                          setState(() {
                            _obcurseText = !_obcurseText;
                          });
                        },
                        icon: Icon(
                          _obcurseText ? Icons.visibility_off : Icons.visibility,
                        ),
                      ),
                      obscureText: _obcurseText,
                    ),
                    const SizedBox(height: 24),
                    VinchestaTextField(
                      labelText: context.strings.repeatPassword,
                      labelStyle: context.text.vinchestaTextField,
                      name: _ChangePasswordPageFields.repeatPassword.name,
                      validator: (value) => _confirmedPasswordValidator(
                        value,
                        context.strings,
                      ),
                      suffix: IconButton(
                        onPressed: () {
                          setState(() {
                            _obcurseText = !_obcurseText;
                          });
                        },
                        icon: Icon(
                          _obcurseText ? Icons.visibility_off : Icons.visibility,
                        ),
                      ),
                      obscureText: _obcurseText,
                    ),
                    const SizedBox(height: 44),
                    VinchestaButton(
                      content: Text(
                        context.strings.changePassword,
                        style: context.text.loginPageButton,
                      ),
                      onPressed: () {
                        _onButtonTap();
                      },
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      );
  }
}
```

// days_page.dart – файл що містить сторінку з перелком днів

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_slidable/flutter_slidable.dart';
import 'package:vinchesta_story/api/response/area_response.dart';
import 'package:vinchesta_story/api/response/day_response.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
```

```
import 'package:vinchesta_story/utils/extensions/date_time_ext.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';
import '../../cubit/days_page_cubit/days_page_cubit.dart';
import '../../views/vinchesta_appbar/vinchesta_appbar.dart';

@RoutePage()
class DaysPage extends StatefulWidget implements AutoRouteWrapper {
  final AreaResponse areaModel;

  const DaysPage({
    required this.areaModel,
    super.key,
  });

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<DaysPageCubit>(param1: areaModel.uid),
      child: this,
    );
  }

  @override
  State<DaysPage> createState() => _DaysPageState();
}

class _DaysPageState extends State<DaysPage> {
  DaysPageCubit get _cubit => context.read<DaysPageCubit>();
  void _onStateChanged(
    BuildContext context,
    DaysPageState state,
  ) {
    switch (state.status) {
      case DaysPageStatus.success:
        break;

      default:
        break;
    }
  }

  @override
  Widget build(BuildContext context) {
    return BlocConsumer<DaysPageCubit, DaysPageState>(
      listener: _onStateChanged,
      builder: (context, state) {
        return Scaffold(
          appBar: VinchestaAppBar(
            title: context.strings.diary,
            actions: [
              GestureDetector(
                onTap: () {
                  router.pushSettingsPage();
                },
                child: Icon(
                  Icons.settings,
                  color: context.color.appBackIcon,
                  size: 28,
                ),
              ),
              const SizedBox(width: 20),
              Padding(
                padding: const EdgeInsets.only(right: 24.0),
                child: GestureDetector(
                  onTap: () {
                    router.pushChartsPage();
                  },
```

```
                child: Icon(
                  Icons.add_chart,
                  color: context.color.appBackIcon,
                  size: 28,
                ),
              ),
            ),
          ],
        ),
        body: Container(
          width: double.maxFinite,
          decoration: BoxDecoration(
            gradient: context.gradient.primaryGradient,
          ),
          child: Padding(
            padding: const EdgeInsets.symmetric(
              vertical: 12,
            ),
            child: Stack(
              children: [
                Positioned(
                  right: -2,
                  top: -140,
                  child: Text(
                    '☽',
                    style: context.text.mainPageBackground,
                  ),
                ),
                ListView.builder(
                  itemCount: state.daysResponse.length,
                  itemBuilder: (BuildContext context, int index) {
                    final dayThing = state.daysResponse[index];
                    return Padding(
                      padding: const EdgeInsets.all(8.0)
                      child: _buildTile(dayThing),
                    );
                  },
                ),
              ],
            ),
          ),
        ),
      );
    },
  );
}

Widget _buildTile(DayResponse dayThing) {
  return GestureDetector(
    onTap: () {
      // if (AreaMocks.areaMocks.first.dayThings.isEmpty) {
      //   router.pushThingsPage();
      // } else {

      // }
      router.pushManagementPage(dayThing);
    },
    child: Slidable(
      endActionPane: ActionPane(
        motion: const ScrollMotion(),
        children: [
          SlidableAction(
            borderRadius: const BorderRadius.only(
              bottomRight: Radius.circular(15),
              topRight: Radius.circular(15),
            ),
            onPressed: (_) async {
              await _cubit.removeDay(dayThing);
            },
```

```
                backgroundColor: context.color.daysPageDelete,
                foregroundColor: context.color.daysPageFore,
                icon: Icons.delete,
              ),
            ],
          ),
          child: Container(
            decoration: BoxDecoration(
              color: context.color.dayTileColor,
              borderRadius: const BorderRadius.only(
                topLeft: Radius.circular(15),
              ),
            ),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Text(
                    style: context.text.daysPageLabel,
                    dayThing.createdAt.toLocal().formattedDate,
                  ),
                ),
                Container(
                  width: double.maxFinite,
                  decoration: BoxDecoration(
                    color: dayThing.progressStatus.getColor(context),
                  ),
                  child: Padding(
                    padding: const EdgeInsets.only(left: 8.0, top: 2, bottom: 2),
                    child: Text(
                      style: context.text.loginPageDescription2,
                      context.strings.loginPageStatus +
                          dayThing.progressStatus.getLabel(context),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      );
  }
}
```

// login_page.dart – файл що містить сторінку авторизації

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:flutter_svg/svg.dart';
import 'package:vinchesta_story/config/di/locator.dart';
import 'package:vinchesta_story/config/router/vinchesta_router.dart';
import 'package:vinchesta_story/gen/assets.gen.dart';
import 'package:vinchesta_story/services/o_auth2.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/utils/input_validators.dart';
import 'package:vinchesta_story/views/redrawble_scroll_view.dart';

import '../../api/request/login_request.dart';
import '../../cubit/login_page_cubit/login_page_cubit.dart';
import '../../views/fileds/vinchesta_text_filed.dart';
import '../../views/vinchesta_buttons.dart';

enum _LoginPageFields {
  email,
  password;
```

```
}

@RoutePage()
class LoginPage extends StatefulWidget implements AutoRouteWrapper {
  const LoginPage({super.key});

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<LoginPageCubit>(),
      child: this,
    );
  }

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final GoogleSingIn _oauth2Helper = GoogleSingIn();
  final _fbKey = GlobalKey<FormBuilderState>();
  bool _obcurseText = true;
  LoginPageCubit get _cubit => context.read<LoginPageCubit>();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};

  void _onSignInPressed() {
    if (_fbState?.saveAndValidate() ?? false) {
      final password = _fbValues[_LoginPageFields.password.name];
      final email = _fbValues[_LoginPageFields.email.name];

      _cubit.login(
        LoginRequest(
          password: password,
          email: email,
        ),
      );
    }
  }

  void _onStateChanged(
    BuildContext context,
    LoginPageState state,
  ) {
    switch (state.status) {
      case LoginPageStatus.success:
        router.pushMainPage();
        break;

      default:
        break;
    }
  }

  @override
  Widget build(BuildContext context) {
    return BlocConsumer<LoginPageCubit, LoginPageState>(
      listener: _onStateChanged,
      builder: (context, state) {
        return Scaffold(
          body: state.status == LoginPageStatus.loading
              ? const Center(
                  child: CircularProgressIndicator(),
                )
              : Container(
                  decoration: BoxDecoration(
                    gradient: context.gradient.primaryGradient,
                  ),
                  child: Padding(
```

```
                    padding: const EdgeInsets.fromLTRB(12, 12, 12, 0),
                    child: FormBuilder(
                      key: _fbKey,
                      child: RedrawableScrollView(
                        margin: EdgeInsets.zero,
                        children: [
                          const SizedBox(height: 44),
                          _buildTitlePart(),
                          const SizedBox(height: 44),
                          Container(
                            decoration: BoxDecoration(
                              color: context.color.authBanner,
                              borderRadius: BorderRadius.circular(20),
                            ),
                            padding: const EdgeInsets.all(16),
                            child: _buildActionsPart(),
                          ),
                          Padding(
                            padding: const EdgeInsets.symmetric(
                              horizontal: 8,
                              vertical: 12,
                            ),
                            child: _buildBottomActions(),
                          ),
                          const Spacer(),
                        ],
                      ),
                    ),
                  ),
                ),
      );
    },
  );
}

Widget _buildBottomActions() {
  return Column(
    children: [
      VinchestaButton(
        onPressed: () {
          router.pushRegisterPage();
        },
        content: Text(
          context.strings.signUp,
          style: context.text.loginPageButton,
        ),
      ),
      const SizedBox(height: 18),
      // Align(
      //   alignment: Alignment.centerRight,
      //   child: Text(
      //     context.strings.forgotPassword,
      //     style: context.text.loginPageButton,
      //   ),
      // ),
    ],
  );
}

Widget _buildActionsPart() {
  return Column(
    children: [
      VinchestaTextField(
        name: _LoginPageFields.email.name,
        hintText: context.strings.email,
        validator: createEmailValidator(context.strings),
      ),
      const SizedBox(height: 8),
      VinchestaTextField(
```

```
        name: _LoginPageFields.password.name,
        hintText: context.strings.password,
        validator: passwordValidator(context.strings),
        suffix: IconButton(
          onPressed: () {
            setState(() {
              _obcurseText = !_obcurseText;
            });
          },
          icon: Icon(
            _obcurseText ? Icons.visibility_off : Icons.visibility,
          ),
        ),
        obscureText: _obcurseText,
      ),
      const SizedBox(height: 18),
      VinchestaButton(
        onPressed: _onSignInPressed,
        content: Text(
          context.strings.signIn,
          style: context.text.loginPageButton,
        ),
      ),
      VinchestaButton(
        onPressed: () async {
          // var client = await _oauth2Helper.signInWithGoogle();
          context.toasts.showError(
            message: context.strings.functionalityIsNotAvailable,
          );
        },
        content: Row(
          children: [
            SvgPicture.asset(
              height: 20,
              width: 20,
              Assets.icons.google,
            ),
            const SizedBox(width: 16),
            Text(
              'Auth with Google',
              style: context.text.loginPageButton,
            ),
          ],
        ),
      ),
    ],
  );
}

Widget _buildTitlePart() {
  return Container(
    decoration: BoxDecoration(
      color: context.color.primaryColor,
      borderRadius: BorderRadius.circular(20),
      boxShadow: [
        BoxShadow(
          offset: const Offset(0, 2),
          blurRadius: 40,
          color: context.color.authBanner,
        )
      ],
    ),
    padding: const EdgeInsets.all(4),
    child: Column(
      children: [
        Container(
          decoration: BoxDecoration(
            color: context.color.textFieldFillColor,
            borderRadius: BorderRadius.circular(20),
```

```
            ),
            child: Stack(
              children: [
                Positioned(
                  left: 60,
                  top: -20,
                  child: Text(
                    '◕',
                    style: context.text.loginPageBanner,
                  ),
                ),
                Positioned(
                  right: 38,
                  top: -30,
                  child: Text(
                    '  ',
                    style: context.text.loginPageBannerStar,
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.symmetric(
                    vertical: 4,
                    horizontal: 12,
                  ),
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                      Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          Padding(
                            padding: const EdgeInsets.only(top: 12.0),
                            child: Column(
                              children: [
                                Text(
                                  maxLines: 2,
                                  context.strings.title,
                                  style: context.text.loginPageTitle,
                                ),
                              ],
                            ),
                          ),
                          Padding(
                            padding: const EdgeInsets.only(bottom: 12.0),
                            child: Text(
                              context.strings.loginPageDescription,
                              style: context.text.loginPageDescription2,
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

// main_page.dart – файл що містить сторінку авторизації

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
```

```dart
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';

import '../../cubit/main_page_cubit/main_page_cubit.dart';
import '../../views/animations/faded_slide_animated_widget.dart';

@RoutePage()
class MainPage extends StatefulWidget implements AutoRouteWrapper {
  const MainPage({super.key});

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<MainPageCubit>(),
      child: this,
    );
  }

  @override
  State<MainPage> createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> with TickerProviderStateMixin {
  bool _animatedTitle = false;
  MainPageCubit get _cubit => context.read<MainPageCubit>();
  late AnimationController _animationController;
  late Animation<double> _opacityAnimation;

  @override
  void initState() {
    super.initState();
    _animatedTitle = true;

    _animationController = AnimationController(
      duration: const Duration(milliseconds: 500),
      vsync: this, // Adjust the duration as needed
    );

    // Initialize the opacity animation
    _opacityAnimation =
        Tween<double>(begin: 0.0, end: 1.0).animate(_animationController);

    // Start the animation when the page is opened
    WidgetsBinding.instance.addPostFrameCallback((_) {
      _animationController.forward();
    });
  }

  @override
  void dispose() {
    _animationController.dispose();
    super.dispose();
  }

  void _onStateChanged(
    BuildContext context,
    MainPageState state,
  ) {
    switch (state.status) {
      case MainPageStatus.success:
        break;

      default:
        break;
    }
  }
```

```
@override
Widget build(BuildContext context) {
  return BlocConsumer<MainPageCubit, MainPageState>(
    listener: _onStateChanged,
    builder: (context, state) {
      return Scaffold(
        floatingActionButton: FloatingActionButton(
          elevation: 0,
          backgroundColor: context.color.textNoBorderFieldFocusedBorder,
          onPressed: () {
            context.dialogs.showInfoDialog(
              context.strings.functionalityIsNotAvailable,
            );
          },
          shape: const CircleBorder(),
          child: const Icon(Icons.add),
        ),
        body: state.status == MainPageStatus.loading
            ? const Center(
                child: CircularProgressIndicator(),
              )
            : Container(
                decoration: BoxDecoration(
                  gradient: context.gradient.primaryGradient,
                ),
                child: SingleChildScrollView(
                  child: Stack(
                    children: [
                      Positioned(
                        right: -100,
                        top: -140,
                        child: Text(
                          '✯',
                          style: context.text.mainPageBackground,
                        ),
                      ),
                      Padding(
                        padding: const EdgeInsets.symmetric(
                          horizontal: 12,
                        ),
                        child: Column(
                          mainAxisSize: MainAxisSize.max,
                          crossAxisAlignment: CrossAxisAlignment.center,
                          mainAxisAlignment: MainAxisAlignment.start,
                          children: [
                            const SizedBox(height: 64),
                            Align(
                              alignment: Alignment.centerLeft,
                              child: FadedSlideAnimatedWidget.slightlyFromTop(
                                delay: const Duration(milliseconds: 500),
                                child: Text(
                                  context.strings.mainPageAsk,
                                  style: context.text.mainPageTitle,
                                ),
                              ),
                            ),
                            FadedSlideAnimatedWidget.slightlyFromTop(
                              delay: const Duration(milliseconds: 1300),
                              child: Align(
                                alignment: Alignment.centerLeft,
                                child: Text(
                                  context.strings.mainPageQuestion,
                                  style: context.text.mainPageTitle,
                                ),
                              ),
                            ),
                            FadedSlideAnimatedWidget.slightlyFromTop(
                              delay: const Duration(milliseconds: 2100),
```

```
                                child: Align(
                                  alignment: Alignment.centerLeft,
                                  child: Text(
                                    context.strings.mainPageYour,
                                    style: context.text.mainPageTitle,
                                  ),
                                ),
                              ),
                              const SizedBox(height: 40),
                              Container(
                                constraints: const BoxConstraints(
                                  minWidth: double.maxFinite,
                                  minHeight: 400,
                                ),
                                decoration: BoxDecoration(
                                  color: context.color.authBanner
                                      .withOpacity(0.75),
                                  borderRadius: BorderRadius.circular(20),
                                ),
                                padding: const EdgeInsets.all(16),
                                child: Wrap(
                                  spacing: 40,
                                  children: state.areas.map((element) {
                                    return GestureDetector(
                                      onTap: () {
                                        router.pushDaysPage(element);
                                      },
                                      child:
                                          _buildTile(element.title ?? 'title'),
                                    );
                                  }).toList(),
                                ),
                              ),
                            ],
                          ),
                        ),
                      ),
                    ),
                  ),
                ),
              );
            },
          );
        }

        Widget _buildTile(String name) {
          return SizedBox(
            width: 100,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Icon(
                  Icons.note,
                  color: context.color.areaIconColor,
                  size: 54,
                ),
                Text(
                  name,
                  maxLines: 3,
                  overflow: TextOverflow.ellipsis,
                  textAlign: TextAlign.center,
                  style: context.text.mainPageTileLabel,
                ),
              ],
            ),
          );
        }
      }
```

// mangament_page.dart – файл що містить сторінку з матрицею Ейзенхауера

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:vinchesta_story/api/response/day_response.dart';
import
'package:vinchesta_story/cubit/management_page_cubit/management_page_cubit.dart'
;

import 'package:vinchesta_story/models/management_page_tab_type.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';

import '../../models/thing.dart';
import '../../views/vinchesta_bottom_navigation_bar.dart';

@RoutePage()
class ManagementPage extends StatefulWidget implements AutoRouteWrapper {
  final DayResponse dayThings;

  const ManagementPage({
    super.key,
    required this.dayThings,
  });

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<ManagementPageCubit>(),
      child: this,
    );
  }

  @override
  State<ManagementPage> createState() => _ManagementPageState();
}

class _ManagementPageState extends State<ManagementPage> {
  ManagementPageCubit get _cubit => context.read<ManagementPageCubit>();
  void onActiveIndexChanged({
    required TabsRouter tabsRouter,
    required int index,
  }) {
    tabsRouter.setActiveIndex(index);
  }

  @override
  void initState() {
    super.initState();
    _cubit.getTasksAndSort(widget.dayThings.uid);
  }

  void _onStateChanged(
    BuildContext context,
    ManagementPageState state,
  ) {
    switch (state.status) {
      case ManagementPageStatus.success:
        break;

      default:
        break;
```

```
  }
}

List<Thing> _getThings(
  ManagementPageState state,
  ManagementPageTabType item,
) {
  switch (item) {
    case ManagementPageTabType.done:
      return state.done;
    case ManagementPageTabType.inProgress:
      return state.inProgress;
    case ManagementPageTabType.toDo:
      return state.toDo;
  }
}

@override
Widget build(BuildContext context) {
  const tabs = ManagementPageTabType.values;
  return BlocConsumer<ManagementPageCubit, ManagementPageState>(
    listener: _onStateChanged,
    builder: (context, state) {
      if (state.status == ManagementPageStatus.loading) {
        return const Scaffold(
          body: Center(
            child: CircularProgressIndicator(),
          ),
        );
      }
      return AutoTabsRouter(
        routes: tabs
            .map((item) => item.route(
                  _getThings(state, item),
                  widget.dayThings.uid,
                ))
            .toList(),
        homeIndex: 0,
        builder: (context, child) {
          final tabsRouter = AutoTabsRouter.of(context);

          final mainPageTabType = tabs[tabsRouter.activeIndex];
          final label = mainPageTabType.getLabel(context);

          return Scaffold(
            appBar: VinchestaAppBar(
              title: label,
              trailingIconType: TrailingIconType.add,
              onTrailingTap: () async {
                await router.pushThingsPage(
                  dayUid: widget.dayThings.uid,
                );
                _cubit.getTasksAndSort(widget.dayThings.uid);
              },
              onLeadingTap: () async {
                await router.pop();
              },
            ),
            body: Container(
              decoration: BoxDecoration(
                gradient: context.gradient.primaryGradient,
              ),
              child: Column(children: [
                Expanded(child: child),
                _buildBottomNavigationBar(
                  tabsRouter: tabsRouter,
                  items: tabs,
                ),
              ]),
```

```
              ),
            );
          },
        );
      },
    );
  }

  Widget _buildBottomNavigationBar({
    required TabsRouter tabsRouter,
    required List<ManagementPageTabType> items,
  }) {
    return VinchestaBottomNavigationBar<ManagementPageTabType>(
      items: items,
      activeIndex: tabsRouter.activeIndex,
      onActiveIndexChanged: (int index) => onActiveIndexChanged(
        tabsRouter: tabsRouter,
        index: index,
      ),
    );
  }
}
```

// done_tab.dart – файл що містить вкладку з виконаними задачами

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:vinchesta_story/pages/tasks_page/task_page_dto.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';

import '../../../config/di/locator.dart';
import '../../../config/router/vinchesta_router.dart';
import '../../../cubit/management_page_cubit/management_page_cubit.dart';
import '../../../models/express_status.dart';
import '../../../models/important_status.dart';
import '../../../models/thing.dart';
import '../../../views/vinchesta_buttons.dart';

@RoutePage()
class DoneTab extends StatefulWidget {
  final List<Thing> dayThings;
  final String dayUid;

  const DoneTab({
    super.key,
    required this.dayThings,
    required this.dayUid,
  });

  @override
  State<DoneTab> createState() => _DoneTabState();
}

class _DoneTabState extends State<DoneTab> {
  List<Thing> _getExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
            thing.importantStatus == ImportantStatus.important)
```

```
      .toList();

  return things;
}

List<Thing> _getExpressAndNoImportant() {
  final things = widget.dayThings
      .where((thing) =>
          thing.expressStatus == ExpressStatus.express &&
          thing.importantStatus == ImportantStatus.notImportant)
      .toList();

  return things;
}

List<Thing> _getNoExpressAndNoImportant() {
  final things = widget.dayThings
      .where((thing) =>
          thing.expressStatus == ExpressStatus.notExpress &&
          thing.importantStatus == ImportantStatus.notImportant)
      .toList();

  return things;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: BoxDecoration(
        gradient: context.gradient.primaryGradient,
      ),
      child: LayoutBuilder(
        builder: (BuildContext context, BoxConstraints constraints) {
          return Column(
            children: [
              Expanded(
                child: _buildTopContainers(),
              ),
              Expanded(
                child: _buildBottomContainers(),
              ),
            ],
          );
        },
      ),
    ),
  );
}

Widget _buildTopContainers() {
  return Row(
    children: [
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(
                color: context.color.authBanner,
                width: 2,
              )),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.expressImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
```

```
            ),
            const SizedBox(height: 8),
            Text(
              context.strings.priority('1'),
              textAlign: TextAlign.center,
              style: context.text.thingTileLabel,
            ),
            const SizedBox(height: 8),
            _getExpressAndImportant().isEmpty
                ? Text(
                    context.strings.emptyTasksDoneDescription,
                    textAlign: TextAlign.center,
                    style: context.text.thingTileLabel,
                  )
                : Column(
                    children: [
                      Text(
                        context.strings.tasksAmount(
                          _getExpressAndImportant().length.toString(),
                        ),
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      ),
                      const SizedBox(height: 12),
                      VinchestaButton(
                        content: Text(
                          context.strings.open,
                          style: context.text.loginPageButton,
                        ),
                        onPressed: () async {
                          await router.pushTasksPage(
                            tasks: _getExpressAndImportant(),
                            taskPageDto: TaskPageDto(
                              dayUid: widget.dayUid,
                              importantStatus: ImportantStatus.important,
                              expressStatus: ExpressStatus.express,
                            ),
                          );
                          await locator
                              .get<ManagementPageCubit>()
                              .getTasksAndSort(widget.dayUid);
                        },
                      )
                    ],
                  ),
          ],
        ),
      ),
    ),
  ),
  Expanded(
    child: Container(
      decoration: BoxDecoration(
          color: context.color.textFieldFillColor,
          border: Border.all(color: context.color.authBanner, width: 2)),
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Column(
          children: [
            Text(
              context.strings.expressNotImportant,
              textAlign: TextAlign.center,
              style: context.text.thingTitle,
            ),
            const SizedBox(height: 8),
            Text(
              context.strings.priority('2'),
              textAlign: TextAlign.center,
              style: context.text.thingTileLabel,
```

```
                ),
                const SizedBox(height: 8),
                _getExpressAndNoImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksDoneDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getExpressAndNoImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
                              await router.pushTasksPage(
                                tasks: _getExpressAndNoImportant(),
                                taskPageDto: TaskPageDto(
                                  dayUid: widget.dayUid,
                                  importantStatus:
                                      ImportantStatus.notImportant,
                                  expressStatus: ExpressStatus.express,
                                ),
                              );
                              await locator
                                  .get<ManagementPageCubit>()
                                  .getTasksAndSort(widget.dayUid);
                            },
                          )
                        ],
                      ),
              ],
            ),
          ),
        ),
      )
    ],
  );
}

Widget _buildBottomContainers() {
  return Row(
    children: [
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(color: context.color.authBanner, width: 2)),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.notExpressImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('3'),
```

```
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getNoExpressAndImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksDoneDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getNoExpressAndImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
                              await router.pushTasksPage(
                                tasks: _getNoExpressAndImportant(),
                                taskPageDto: TaskPageDto(
                                  dayUid: widget.dayUid,
                                  importantStatus: ImportantStatus.important,
                                  expressStatus: ExpressStatus.notExpress,
                                ),
                              );
                              await locator
                                  .get<ManagementPageCubit>()
                                  .getTasksAndSort(widget.dayUid);
                            },
                          )
                        ],
                      ),
              ],
            ),
          ),
        ),
      ),
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(color: context.color.authBanner, width: 2)),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.notExpressNotImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('4'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getNoExpressAndNoImportant().isEmpty
                    ? Text(
```

```
                            context.strings.emptyTasksDoneDescription,
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          )
                        : Column(
                            children: [
                              Text(
                                context.strings.tasksAmount(
                                  _getNoExpressAndNoImportant().length.toString(),
                                ),
                                textAlign: TextAlign.center,
                                style: context.text.thingTileLabel,
                              ),
                              const SizedBox(height: 12),
                              VinchestaButton(
                                content: Text(
                                  context.strings.open,
                                  style: context.text.loginPageButton,
                                ),
                                onPressed: () async {
                                  await router.pushTasksPage(
                                    tasks: _getNoExpressAndNoImportant(),
                                    taskPageDto: TaskPageDto(
                                      dayUid: widget.dayUid,
                                      importantStatus:
                                          ImportantStatus.notImportant,
                                      expressStatus: ExpressStatus.notExpress,
                                    ),
                                  );
                                  await locator
                                      .get<ManagementPageCubit>()
                                      .getTasksAndSort(widget.dayUid);
                                },
                              )
                            ],
                          ),
                    ],
                  ),
                ),
              ),
            ),
          ),
        ],
      );
  }
}
```

// in_progess_tab.dart – файл що містить вкладку з задачами в прогресі

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';

import '../../../config/di/locator.dart';
import '../../../config/router/vinchesta_router.dart';
import '../../../cubit/management_page_cubit/management_page_cubit.dart';
import '../../../models/express_status.dart';
import '../../../models/important_status.dart';
import '../../../models/thing.dart';
import '../../../views/vinchesta_buttons.dart';
import '../../tasks_page/task_page_dto.dart';

@RoutePage()
class InProgressTab extends StatefulWidget {
  final List<Thing> dayThings;
  final String dayUid;

  const InProgressTab({
    super.key,
    required this.dayThings,
```

```
    required this.dayUid,
  });

  @override
  State<InProgressTab> createState() => _InProgressTabState();
}

class _InProgressTabState extends State<InProgressTab> {
  List<Thing> _getExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getExpressAndNoImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.notImportant)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndNoImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
            thing.importantStatus == ImportantStatus.notImportant)
        .toList();

    return things;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: BoxDecoration(
          gradient: context.gradient.primaryGradient,
        ),
        child: LayoutBuilder(
          builder: (BuildContext context, BoxConstraints constraints) {
            return Column(
              children: [
                Expanded(
                  child: _buildTopContainers(),
                ),
                Expanded(
                  child: _buildBottomContainers(),
                ),
              ],
            );
          },
        ),
```

```
    ),
  );
}

Widget _buildTopContainers() {
  return Row(
    children: [
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(
                color: context.color.authBanner,
                width: 2,
              )),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.expressImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('1'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getExpressAndImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksInProgressDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getExpressAndImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
                              await router.pushTasksPage(
                                tasks: _getExpressAndImportant(),
                                taskPageDto: TaskPageDto(
                                  dayUid: widget.dayUid,
                                  importantStatus: ImportantStatus.important,
                                  expressStatus: ExpressStatus.express,
                                ),
                              );
                              await locator
                                  .get<ManagementPageCubit>()
                                  .getTasksAndSort(widget.dayUid);
                            },
                          )
                        ],
                      ),
              ],
```

```
                ),
              ),
            ),
          ),
          Expanded(
            child: Container(
              decoration: BoxDecoration(
                  color: context.color.textFieldFillColor,
                  border: Border.all(color: context.color.authBanner, width: 2)),
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Column(
                  children: [
                    Text(
                      context.strings.expressNotImportant,
                      textAlign: TextAlign.center,
                      style: context.text.thingTitle,
                    ),
                    const SizedBox(height: 8),
                    Text(
                      context.strings.priority('2'),
                      textAlign: TextAlign.center,
                      style: context.text.thingTileLabel,
                    ),
                    const SizedBox(height: 8),
                    _getExpressAndNoImportant().isEmpty
                        ? Text(
                            context.strings.emptyTasksInProgressDescription,
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          )
                        : Column(
                            children: [
                              Text(
                                context.strings.tasksAmount(
                                  _getExpressAndNoImportant().length.toString(),
                                ),
                                textAlign: TextAlign.center,
                                style: context.text.thingTileLabel,
                              ),
                              const SizedBox(height: 12),
                              VinchestaButton(
                                content: Text(
                                  context.strings.open,
                                  style: context.text.loginPageButton,
                                ),
                                onPressed: () {
                                  router.pushTasksPage(
                                    tasks: _getExpressAndNoImportant(),
                                    taskPageDto: TaskPageDto(
                                      dayUid: widget.dayUid,
                                      importantStatus:
                                          ImportantStatus.notImportant,
                                      expressStatus: ExpressStatus.express,
                                    ),
                                  );
                                },
                              )
                            ],
                          ),
                  ],
                ),
              ),
            ),
          )
        ],
      );
    }
```

```
Widget _buildBottomContainers() {
  return Row(
    children: [
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(color: context.color.authBanner, width: 2)),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.notExpressImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('3'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getNoExpressAndImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksInProgressDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getNoExpressAndImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
                              await router.pushTasksPage(
                                tasks: _getNoExpressAndImportant(),
                                taskPageDto: TaskPageDto(
                                  dayUid: widget.dayUid,
                                  importantStatus: ImportantStatus.important,
                                  expressStatus: ExpressStatus.notExpress,
                                ),
                              );
                              await locator
                                  .get<ManagementPageCubit>()
                                  .getTasksAndSort(widget.dayUid);
                            },
                          )
                        ],
                      ),
              ],
            ),
          ),
        ),
      ),
      Expanded(
        child: Container(
          decoration: BoxDecoration(
```

```
                color: context.color.textFieldFillColor,
                border: Border.all(color: context.color.authBanner, width: 2)),
            child: Padding(
              padding: const EdgeInsets.all(8.0),
              child: Column(
                children: [
                  Text(
                    context.strings.notExpressNotImportant,
                    textAlign: TextAlign.center,
                    style: context.text.thingTitle,
                  ),
                  const SizedBox(height: 8),
                  Text(
                    context.strings.priority('4'),
                    textAlign: TextAlign.center,
                    style: context.text.thingTileLabel,
                  ),
                  const SizedBox(height: 8),
                  _getNoExpressAndNoImportant().isEmpty
                      ? Text(
                          context.strings.emptyTasksInProgressDescription,
                          textAlign: TextAlign.center,
                          style: context.text.thingTileLabel,
                        )
                      : Column(
                          children: [
                            Text(
                              context.strings.tasksAmount(
                                _getNoExpressAndNoImportant().length.toString(),
                              ),
                              textAlign: TextAlign.center,
                              style: context.text.thingTileLabel,
                            ),
                            const SizedBox(height: 12),
                            VinchestaButton(
                              content: Text(
                                context.strings.open,
                                style: context.text.loginPageButton,
                              ),
                              onPressed: () async {
                                await router.pushTasksPage(
                                  tasks: _getNoExpressAndNoImportant(),
                                  taskPageDto: TaskPageDto(
                                    dayUid: widget.dayUid,
                                    importantStatus:
                                        ImportantStatus.notImportant,
                                    expressStatus: ExpressStatus.notExpress,
                                  ),
                                );
                                await locator
                                    .get<ManagementPageCubit>()
                                    .getTasksAndSort(widget.dayUid);
                              },
                            )
                          ],
                        ),
                ],
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

// to_do_tab.dart – файл що містить вкладку з запланованими задачами

```
import 'package:auto_route/auto_route.dart';
```

```dart
import 'package:flutter/material.dart';
import 'package:vinchesta_story/config/di/locator.dart';
import
'package:vinchesta_story/cubit/management_page_cubit/management_page_cubit.dart'
;
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';

import '../../../config/router/vinchesta_router.dart';
import '../../../models/express_status.dart';
import '../../../models/important_status.dart';
import '../../../models/thing.dart';
import '../../tasks_page/task_page_dto.dart';

@RoutePage()
class ToDoTab extends StatefulWidget {
  final List<Thing> dayThings;
  final String dayUid;

  const ToDoTab({
    super.key,
    required this.dayThings,
    required this.dayUid,
  });

  @override
  State<ToDoTab> createState() => _ToDoTabState();
}

class _ToDoTabState extends State<ToDoTab> {
  List<Thing> _getExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
            thing.importantStatus == ImportantStatus.important)
        .toList();

    return things;
  }

  List<Thing> _getExpressAndNoImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.express &&
            thing.importantStatus == ImportantStatus.notImportant)
        .toList();

    return things;
  }

  List<Thing> _getNoExpressAndNoImportant() {
    final things = widget.dayThings
        .where((thing) =>
            thing.expressStatus == ExpressStatus.notExpress &&
            thing.importantStatus == ImportantStatus.notImportant)
        .toList();

    return things;
  }
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: BoxDecoration(
        gradient: context.gradient.primaryGradient,
      ),
      child: LayoutBuilder(
        builder: (BuildContext context, BoxConstraints constraints) {
          return Column(
            children: [
              Expanded(
                child: _buildTopContainers(),
              ),
              Expanded(
                child: _buildBottomContainers(),
              ),
            ],
          );
        },
      ),
    ),
  );
}

Widget _buildTopContainers() {
  return Row(
    children: [
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(
                color: context.color.authBanner,
                width: 2,
              )),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.expressImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('1'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getExpressAndImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksToDoDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getExpressAndImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
```

```
                        VinchestaButton(
                          content: Text(
                            context.strings.open,
                            style: context.text.loginPageButton,
                          ),
                          onPressed: () async {
                            await router.pushTasksPage(
                              tasks: _getExpressAndImportant(),
                              taskPageDto: TaskPageDto(
                                dayUid: widget.dayUid,
                                importantStatus: ImportantStatus.important,
                                expressStatus: ExpressStatus.express,
                              ),
                            );
                            await locator
                                .get<ManagementPageCubit>()
                                .getTasksAndSort(widget.dayUid);
                          },
                        )
                      ],
                    ),
              ],
            ),
          ),
        ),
      ),
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(color: context.color.authBanner, width: 2)),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.expressNotImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('2'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getExpressAndNoImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksToDoDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getExpressAndNoImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
```

```
                                await router.pushTasksPage(
                                  tasks: _getExpressAndNoImportant(),
                                  taskPageDto: TaskPageDto(
                                    dayUid: widget.dayUid,
                                    importantStatus:
                                        ImportantStatus.notImportant,
                                    expressStatus: ExpressStatus.express,
                                  ),
                                );
                                await locator
                                    .get<ManagementPageCubit>()
                                    .getTasksAndSort(widget.dayUid);
                              },
                            )
                          ],
                        ),
                    ],
                  ),
                ),
              ),
            )
          ],
        );
      }

      Widget _buildBottomContainers() {
        return Row(
          children: [
            Expanded(
              child: Container(
                decoration: BoxDecoration(
                    color: context.color.textFieldFillColor,
                    border: Border.all(color: context.color.authBanner, width: 2)),
                child: Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    children: [
                      Text(
                        context.strings.notExpressImportant,
                        textAlign: TextAlign.center,
                        style: context.text.thingTitle,
                      ),
                      const SizedBox(height: 8),
                      Text(
                        context.strings.priority('3'),
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      ),
                      const SizedBox(height: 8),
                      _getNoExpressAndImportant().isEmpty
                          ? Text(
                              context.strings.emptyTasksToDoDescription,
                              textAlign: TextAlign.center,
                              style: context.text.thingTileLabel,
                            )
                          : Column(
                              children: [
                                Text(
                                  context.strings.tasksAmount(
                                    _getNoExpressAndImportant().length.toString(),
                                  ),
                                  textAlign: TextAlign.center,
                                  style: context.text.thingTileLabel,
                                ),
                                const SizedBox(height: 12),
                                VinchestaButton(
                                  content: Text(
                                    context.strings.open,
                                    style: context.text.loginPageButton,
```

```
                        ),
                        onPressed: () async {
                          await router.pushTasksPage(
                            tasks: _getNoExpressAndImportant(),
                            taskPageDto: TaskPageDto(
                              dayUid: widget.dayUid,
                              importantStatus: ImportantStatus.important,
                              expressStatus: ExpressStatus.notExpress,
                            ),
                          );
                          await locator
                              .get<ManagementPageCubit>()
                              .getTasksAndSort(widget.dayUid);
                        },
                      )
                    ],
                  ),
            ],
          ),
        ),
      ),
      Expanded(
        child: Container(
          decoration: BoxDecoration(
              color: context.color.textFieldFillColor,
              border: Border.all(color: context.color.authBanner, width: 2)),
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              children: [
                Text(
                  context.strings.notExpressNotImportant,
                  textAlign: TextAlign.center,
                  style: context.text.thingTitle,
                ),
                const SizedBox(height: 8),
                Text(
                  context.strings.priority('4'),
                  textAlign: TextAlign.center,
                  style: context.text.thingTileLabel,
                ),
                const SizedBox(height: 8),
                _getNoExpressAndNoImportant().isEmpty
                    ? Text(
                        context.strings.emptyTasksToDoDescription,
                        textAlign: TextAlign.center,
                        style: context.text.thingTileLabel,
                      )
                    : Column(
                        children: [
                          Text(
                            context.strings.tasksAmount(
                              _getNoExpressAndNoImportant().length.toString(),
                            ),
                            textAlign: TextAlign.center,
                            style: context.text.thingTileLabel,
                          ),
                          const SizedBox(height: 12),
                          VinchestaButton(
                            content: Text(
                              context.strings.open,
                              style: context.text.loginPageButton,
                            ),
                            onPressed: () async {
                              await router.pushTasksPage(
                                tasks: _getNoExpressAndNoImportant(),
                                taskPageDto: TaskPageDto(
                                  dayUid: widget.dayUid,
```

```
                                        importantStatus:
                                            ImportantStatus.notImportant,
                                        expressStatus: ExpressStatus.notExpress,
                                      ),
                                    );
                                    await locator
                                        .get<ManagementPageCubit>()
                                        .getTasksAndSort(widget.dayUid);
                                  },
                                )
                              ],
                            ),
                          ],
                        ),
                      ),
                    ),
                  ),
                ],
              );
  }
}
```

// ninety_thirty_timer_page.dart – файл що містить сторінку з методикою 90/30

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';

import '../../views/vinchesta_timer.dart';

@RoutePage()
class NinetyThirtyTimerPage extends StatelessWidget {
  final VoidCallback onDoneTap;

  const NinetyThirtyTimerPage({
    super.key,
    required this.onDoneTap,
  });

  @override
  Widget build(BuildContext context) {
    // TODO set REAL TIMER 1500 : 300
    return Scaffold(
      appBar: VinchestaAppBar(
        title: context.strings.ninetyMethod,
      ),
      body: Container(
        width: double.maxFinite,
        decoration: BoxDecoration(
          gradient: context.gradient.primaryGradient,
        ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
            const SizedBox(height: 40),
            VinchestaTimer(
              isLongMethod: true,
              workTime: 5400,
              onDoneTap: onDoneTap,
              breakTime: 1800,
            ),
          ],
        ),
      ),
    );
  }
}
```

// regster_page.dart – файл що містить сторінку реєстрації

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:form_builder_validators/form_builder_validators.dart';
import 'package:vinchesta_story/api/request/register_request.dart';
import
'package:vinchesta_story/cubit/register_page_cubit/register_page_cubit.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';
import '../../utils/input_validators.dart';
import '../../views/fileds/vinchesta_text_filed.dart';
import '../../views/vinchesta_buttons.dart';
import '../../views/vinchesta_text_button.dart';

enum _RegisterPageFields {
  name,
  email,
  password,
  repeatPassword;
}

@RoutePage()
class RegisterPage extends StatefulWidget implements AutoRouteWrapper {
  const RegisterPage({super.key});

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<RegisterPageCubit>(),
      child: this,
    );
  }

  @override
  State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final _fbKey = GlobalKey<FormBuilderState>();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};
  bool _obcurseText = true;
  RegisterPageCubit get _cubit => context.read<RegisterPageCubit>();

  void _onStateChanged(
    BuildContext context,
    RegisterPageState state,
  ) {
    switch (state.status) {
      case RegisterPageStatus.success:
        router.pushMainPage();
        break;

      default:
        break;
    }
  }

  @override
  Widget build(BuildContext context) {
    return BlocConsumer<RegisterPageCubit, RegisterPageState>(
      listener: _onStateChanged,
```

```
builder: (context, state) {
  return Scaffold(
    appBar: VinchestaAppBar(
      title: 'Register Page',
    ),
    body: Container(
      constraints: BoxConstraints(
        minHeight: MediaQuery.of(context).size.height,
      ),
      decoration: BoxDecoration(
        gradient: context.gradient.primaryGradient,
      ),
      child: RegisterPageStatus.loading == state.status
          ? const Center(
              child: CircularProgressIndicator(),
            )
          : SingleChildScrollView(
              child: FormBuilder(
                key: _fbKey,
                child: Padding(
                  padding: const EdgeInsets.only(
                    top: 60,
                    left: 12,
                    right: 12,
                  ),
                  child: Column(
                    children: [
                      _buildTitlePart(),
                      const SizedBox(height: 40),
                      Container(
                        decoration: BoxDecoration(
                          color: context.color.authBanner,
                          borderRadius: BorderRadius.circular(20),
                        ),
                        padding: const EdgeInsets.all(16),
                        child: Column(
                          children: [
                            _buildFields(),
                            const SizedBox(height: 24),
                            VinchestaButton(
                              onPressed: () {
                                if (_fbState?.saveAndValidate() ??
                                    false) {
                                  final name = _fbValues[
                                      _RegisterPageFields.name.name];
                                  final password = _fbValues[
                                      _RegisterPageFields.password.name];
                                  final email = _fbValues[
                                      _RegisterPageFields.email.name];
                                  const language = 'en';
                                  _cubit.register(
                                    RegisterRequest(
                                      password: password,
                                      language: language,
                                      name: name,
                                      email: email,
                                    ),
                                  );
                                }

                                // router.pushConfirmEmailPage();
                              },
                              content: Text(
                                context.strings.continueButton,
                                style: context.text.loginPageButton,
                              ),
                            ),
                            const SizedBox(height: 12),
                            VinchestaTextButton(
```

```
                                    onButtonTap: () {
                                      router.pop();
                                    },
                                    title: context.strings.signIn,
                                    style: context.text.loginPageButton,
                                    enabled: true,
                                  ),
                                ],
                              ),
                            ),
                          ],
                        ),
                      ),
                    ],
                  ),
                ),
              ),
            ),
          ),
        ),
      );
    },
  );
}

Widget _buildFields() {
  return Column(
    children: [
      VinchestaTextField(
        name: _RegisterPageFields.name.name,
        hintText: context.strings.name,
        validator: FormBuilderValidators.required(),
      ),
      const SizedBox(height: 16),
      VinchestaTextField(
        name: _RegisterPageFields.email.name,
        hintText: context.strings.email,
        validator: createEmailValidator(context.strings),
      ),
      const SizedBox(height: 16),
      VinchestaTextField(
        name: _RegisterPageFields.password.name,
        hintText: context.strings.password,
        validator: passwordValidator(context.strings),
        suffix: IconButton(
          onPressed: () {
            setState(() {
              _obcurseText = !_obcurseText;
            });
          },
          icon: Icon(
            _obcurseText ? Icons.visibility_off : Icons.visibility,
          ),
        ),
        obscureText: _obcurseText,
      ),
      const SizedBox(height: 16),
      VinchestaTextField(
        name: _RegisterPageFields.repeatPassword.name,
        hintText: context.strings.repeatPassword,
        validator: passwordValidator(context.strings),
        suffix: IconButton(
          onPressed: () {
            setState(() {
              _obcurseText = !_obcurseText;
            });
          },
          icon: Icon(
            _obcurseText ? Icons.visibility_off : Icons.visibility,
          ),
        ),
        obscureText: _obcurseText,
      ),
```

```
    ],
  );
}

Widget _buildTitlePart() {
  return Container(
    decoration: BoxDecoration(
      color: context.color.primaryColor,
      borderRadius: BorderRadius.circular(20),
      boxShadow: [
        BoxShadow(
          offset: const Offset(0, 2),
          blurRadius: 40,
          color: context.color.authBanner,
        )
      ],
    ),
    padding: const EdgeInsets.all(4),
    child: Column(
      children: [
        Container(
          decoration: BoxDecoration(
            color: context.color.textFieldFillColor,
            borderRadius: BorderRadius.circular(20),
          ),
          child: Stack(
            children: [
              Positioned(
                left: 60,
                top: -20,
                child: Text(
                  '◕',
                  style: context.text.loginPageBanner,
                ),
              ),
              Positioned(
                right: 38,
                top: -30,
                child: Text(
                  ' ',
                  style: context.text.loginPageBannerStar,
                ),
              ),
              Padding(
                padding: const EdgeInsets.symmetric(
                  vertical: 4,
                  horizontal: 12,
                ),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.center,
                  children: [
                    Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        Padding(
                          padding: const EdgeInsets.only(top: 12.0),
                          child: Column(
                            children: [
                              Text(
                                maxLines: 2,
                                context.strings.title,
                                style: context.text.loginPageTitle,
                              ),
                            ],
                          ),
                        ),
                        Padding(
```

```
                                padding: const EdgeInsets.only(bottom: 12.0),
                                child: Text(
                                  context.strings.loginPageDescription,
                                  style: context.text.loginPageDescription2,
                                ),
                              ),
                            ],
                          ),
                        ],
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      );
    }
}
```

// charts_page.dart – файл що містить сторінку з графіками

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';

@RoutePage()
class SchedulePage extends StatefulWidget {
  const SchedulePage({super.key});

  @override
  State<SchedulePage> createState() => _SchedulePageState();
}

class _SchedulePageState extends State<SchedulePage> {
  late List<_ChartData> data;
  late TooltipBehavior _tooltip;
  late ZoomPanBehavior _zoomPanBehavior;
  late ScrollController _scrollController;

  @override
  void initState() {
    super.initState();
    _tooltip = TooltipBehavior(enable: true);
    _zoomPanBehavior = ZoomPanBehavior(
      enablePinching: true,
      enableSelectionZooming: true,
      enableMouseWheelZooming: true,
      enablePanning: true, // Enable panning
      zoomMode: ZoomMode.x, // Allow zooming and panning along the x-axis
    );
    _scrollController = ScrollController();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: VinchestaAppBar(
        title: context.strings.schedule,
      ),
      body: Container(
        constraints: BoxConstraints(
          minHeight: MediaQuery.of(context).size.height,
          minWidth: MediaQuery.of(context).size.width,
        ),
        decoration: BoxDecoration(
```

```
      gradient: context.gradient.primaryGradient,
    ),
    child: Padding(
      padding: const EdgeInsets.only(
        top: 24,
        bottom: 24,
        left: 0,
      ),
      child: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(4.0),
            child: Text(
              context.strings.chartDescription,
              style: context.text.chartTitle,
              textAlign: TextAlign.center,
            ),
          ),
          Scrollbar(
            radius: const Radius.circular(20),
            thumbVisibility: true,
            trackVisibility: true,
            controller: _scrollController,
            scrollbarOrientation: ScrollbarOrientation.bottom,
            child: Padding(
              padding: const EdgeInsets.only(bottom: 40),
              child: SingleChildScrollView(
                controller: _scrollController,
                scrollDirection: Axis.horizontal,
                child: SizedBox(
                  width: data.length * 50,
                  child: SfCartesianChart(
                      enableMultiSelection: true,
                      enableSideBySideSeriesPlacement: true,
                      zoomPanBehavior: _zoomPanBehavior,
                      enableAxisAnimation: true,
                      primaryXAxis: CategoryAxis(
                        interactiveTooltip:
                            const InteractiveTooltip(enable: true),
                        maximum: data.length.toDouble() - 1,
                        minimum: 0,
                        labelStyle: context.text.dataChart,
                      ),
                      primaryYAxis: NumericAxis(
                        axisLabelFormatter:
                            (AxisLabelRenderDetails details) {
                          return ChartAxisLabel(
                              '${details.text}%', context.text.dataChart);
                        },
                        labelStyle: context.text.mainPageTileLabel,
                        minimum: 0,
                        maximum: 100,
                        interval: 20,
                      ),
                      tooltipBehavior: _tooltip,
                      series: <CartesianSeries<_ChartData, String>>[
                        ColumnSeries<_ChartData, String>(
                          dataSource: data,
                          xValueMapper: (_ChartData data, _) => data.x,
                          yValueMapper: (_ChartData data, _) => data.y,
                          name: context.strings.progressPercentage,
                          color: context.color.primaryColor,
                        )
                      ]),
                ),
              ),
            ),
          ),
        ],
```

```
          ),
        ),
      ),
    );
  }
}

class _ChartData {
  _ChartData(this.x, this.y);

  final String x;
  final double y;
}
```

// settings_page.dart – файл що містить сторінку з налаштуваннями користувача

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:form_builder_validators/form_builder_validators.dart';
import 'package:vinchesta_story/api/request/edit_user_request.dart';
import 'package:vinchesta_story/config/router/vinchesta_router.dart';
import 'package:vinchesta_story/models/languages_type.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/fileds/vinchesta_text_filed.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';

import '../../config/di/locator.dart';
import '../../cubit/settings_page_cubit/settings_page_cubit.dart';
import '../../views/text_field_dropdown.dart';
import '../../views/vinchesta_text_button.dart';

enum _SettingsPageFields {
  name,
  language;
}

@RoutePage()
class SettingsPage extends StatefulWidget implements AutoRouteWrapper {
  const SettingsPage({super.key});

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<SettingsPageCubit>(),
      child: this,
    );
  }

  @override
  State<SettingsPage> createState() => _SettingsPageState();
}

void _onStateChanged(
  BuildContext context,
  SettingsPageState state,
) {
  switch (state.status) {
    case SettingsPageStatus.success:
      router.pop();
      break;
    case SettingsPageStatus.logout:
      router.resetToLoginPage();
      break;

    default:
      break;
```

```
  }
}

class _SettingsPageState extends State<SettingsPage> {
  final _fbKey = GlobalKey<FormBuilderState>();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};
  SettingsPageCubit get _cubit => context.read<SettingsPageCubit>();
  LanguagesType? _getInitialValue(String? languageCode) {
    if (languageCode == null) {
      return null;
    }
    if (languageCode == 'en') {
      return LanguagesType.en;
    }
    return LanguagesType.uk;
  }

  void _onEditUserPressed(SettingsPageState state) {
    final user = state.user;
    if ((_fbState?.saveAndValidate() ?? false) && user != null) {
      final language =
          _fbValues[_SettingsPageFields.language.name] as LanguagesType;
      final name = _fbValues[_SettingsPageFields.name.name];
      _cubit.editUser(
        EditUserRequest(
          language: language.name,
          name: name,
          email: user.email,
        ),
        user.id,
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return BlocConsumer<SettingsPageCubit, SettingsPageState>(
        listener: _onStateChanged,
        builder: (context, state) {
          return Scaffold(
            appBar: VinchestaAppBar(
              title: context.strings.settingsPage,
              trailingIconType: TrailingIconType.check,
              onTrailingTap: () {
                _onEditUserPressed(state);
              },
            ),
            body: state.status == SettingsPageStatus.loading
                ? const Center(
                    child: CircularProgressIndicator(),
                  )
                : Container(
                    constraints: BoxConstraints(
                      minHeight: MediaQuery.of(context).size.height,
                    ),
                    decoration: BoxDecoration(
                      gradient: context.gradient.primaryGradient,
                    ),
                    child: Padding(
                      padding: const EdgeInsets.symmetric(
                        vertical: 12,
                        horizontal: 8,
                      ),
                      child: FormBuilder(
                        key: _fbKey,
                        child: SingleChildScrollView(
                          child: Container(
                            constraints: BoxConstraints(
```

```
      minHeight: MediaQuery.of(context).size.height,
    ),
    decoration: BoxDecoration(
      color: context.color.authBanner,
      borderRadius: BorderRadius.circular(20),
    ),
    child: Padding(
      padding: const EdgeInsets.all(12),
      child: Column(
        children: [
          VinchestaTextField(
            initialValue: state.user?.name,
            labelText: context.strings.name,
            name: _SettingsPageFields.name.name,
          ),
          const SizedBox(height: 20),
          TextFieldDropdown<LanguagesType>.filled(
            name: _SettingsPageFields.language.name,
            items: LanguagesType.values,
            initialValue:
                _getInitialValue(state.user?.language),
            hintText: context.strings.chooseLanguage,
            getLabel: (item) {
              return item.getTitle(context);
            },
            onChanged: (value) {
              if (value != null) {
                _cubit.changeLanguage(
                  value.getLocale(context),
                );
              }
            },
            validator: FormBuilderValidators.required(),
          ),
          const SizedBox(height: 20),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: VinchestaButton(
              content: Row(
                children: [
                  Text(
                    context.strings.subscription,
                    style: context.text.loginPageButton,
                  ),
                  const SizedBox(width: 20),
                  Icon(
                    Icons.shopping_bag,
                    color: context.color.appBackIcon,
                  ),
                ],
              ),
              onPressed: () {},
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: VinchestaButton(
              content: Row(
                children: [
                  Text(
                    context.strings.changePassword,
                    style: context.text.loginPageButton,
                  ),
                  const SizedBox(width: 20),
                  Icon(
                    Icons.password,
                    color: context.color.appBackIcon,
                  ),
                ],
```

```
                                    ),
                                    onPressed: () {
                                      router.pushChangePassword();
                                    },
                                  ),
                                ),
                                Padding(
                                  padding: const EdgeInsets.all(8.0),
                                  child: VinchestaButton(
                                    content: Row(
                                      children: [
                                        Text(
                                          context.strings.logout,
                                          style: context.text.loginPageButton,
                                        ),
                                        const SizedBox(width: 20),
                                        Icon(
                                          Icons.logout_outlined,
                                          color: context.color.appBackIcon,
                                        ),
                                      ],
                                    ),
                                    onPressed: () {
                                      _cubit.logout();
                                    },
                                  ),
                                ),
                                const SizedBox(height: 24),
                                VinchestaTextButton(
                                  enabled: false,
                                  style: context.text.deleteThing,
                                  title: context.strings.deleteAccount,
                                  onButtonTap: () {
                                    _cubit.delete();
                                  },
                                ),
                              ],
                            ),
                          ),
                        ),
                      ),
                    ),
                  ),
                ),
              );
            });
      }
}
```

// thing_manage_page.dart – файл що містить сторінку управління задачею

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:vinchesta_story/models/progress_status.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/utils/extensions/date_time_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';
import 'package:vinchesta_story/views/vinchesta_text_button.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';
import '../../cubit/thing_manage_page_cubit/thing_manage_page_cubit.dart';
import '../../models/thing.dart';

@RoutePage()
class ThingManagePage extends StatefulWidget implements AutoRouteWrapper {
  final Thing thing;
```

```
  const ThingManagePage({
    super.key,
    required this.thing,
  });

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<ThingManagePageCubit>(param1: thing),
      child: this,
    );
  }

  @override
  State<ThingManagePage> createState() => _ThingManagePageState();
}

class _ThingManagePageState extends State<ThingManagePage> {
  ThingManagePageCubit get _cubit => context.read<ThingManagePageCubit>();
  void _onStateChanged(
    BuildContext context,
    ThingManagePageState state,
  ) {
    switch (state.status) {
      case ThingManagePageStatus.removed:
        router.pop();
        break;
      case ThingManagePageStatus.success:
        break;

      default:
        break;
    }
  }

  @override
  Widget build(BuildContext context) {
    return BlocConsumer<ThingManagePageCubit, ThingManagePageState>(
        listener: _onStateChanged,
        builder: (context, state) {
          return Scaffold(
            appBar: VinchestaAppBar(
              title: context.strings.manageTask,
            ),
            body: Container(
              constraints: BoxConstraints(
                minHeight: MediaQuery.of(context).size.height,
              ),
              decoration: BoxDecoration(
                gradient: context.gradient.primaryGradient,
              ),
              child: state.status == ThingManagePageStatus.loading
                  ? const Center(
                      child: CircularProgressIndicator(),
                    )
                  : SingleChildScrollView(
                      child: Padding(
                        padding: const EdgeInsets.symmetric(
                          vertical: 16,
                          horizontal: 24,
                        ),
                        child: Stack(
                          children: [
                            Positioned(
                              right: -20,
                              top: 8,
                              child: Icon(
                                Icons.description,
```

```
                                size: 200,
                                color: context.color.backgroundIconColor,
                              ),
                            ),
                            Column(
                              mainAxisAlignment: MainAxisAlignment.center,
                              crossAxisAlignment: CrossAxisAlignment.start,
                              children: [
                                Align(
                                  alignment: Alignment.center,
                                  child: Text(
                                    state.task.title,
                                    style: context.text.appBarTitle,
                                  ),
                                ),
                                const SizedBox(height: 24),
                                if (state.task.description != null)
                                  Text(
                                    state.task.description ?? '',
                                    style: context.text.mainPageTileLabel,
                                  ),
                                const SizedBox(height: 12),
                                if (state.task.workTime != null)
                                  Text(
                                    "${context.strings.executionTime}
${state.task.getWorkedTimeOfDay().hour}${context.strings.hourShort}
${state.task.getWorkedTimeOfDay().minute}${context.strings.minutesShort}",
                                    style: context.text.mainPageTileLabel,
                                  ),
                                const SizedBox(height: 12),
                                Text(
                                  "${context.strings.logTime}
${state.task.getTimeOfDay().hour}${context.strings.hourShort}
${state.task.getTimeOfDay().minute}${context.strings.minutesShort}",
                                  style: context.text.mainPageTileLabel,
                                ),
                                const SizedBox(height: 12),
                                Align(
                                  alignment: Alignment.bottomRight,
                                  child: Text(
                                    state.task.createdAt
                                            ?.toLocal()
                                            .formattedDateFullWithTime ??
                                        '',
                                    style: context.text.thingDate,
                                  ),
                                ),
                                const SizedBox(height: 12),
                                Align(
                                  alignment: Alignment.center,
                                  child: Text(
                                    context.strings.statuses,
                                    style: context.text.loginPageDescription,
                                  ),
                                ),
                                const SizedBox(height: 12),
                                _buildStatuses(state),
                                const SizedBox(height: 24),
                                _buildActions(state),
                                if (state.task.progressStatus ==
                                        ProgressStatus.toDo ||
                                    state.task.progressStatus ==
                                        ProgressStatus.inProgress)
                                  Padding(
                                    padding: const EdgeInsets.symmetric(
                                        vertical: 12.0),
                                    child: _buildSpecialActions(state),
                                  ),
                                Padding(
```

```
                                padding: const EdgeInsets.symmetric(
                                    vertical: 12.0),
                                child: _buildLogButtons(state),
                              ),
                              VinchestaButton(
                                content: Row(
                                  children: [
                                    Text(
                                      context.strings.editThing,
                                      style: context.text.loginPageButton,
                                    ),
                                    const SizedBox(width: 12),
                                    Icon(
                                      Icons.edit_document,
                                      color: context.color.areaIconColor,
                                    ),
                                  ],
                                ),
                                onPressed: () async {
                                  final thing = await router.pushThingsPage(
                                    thing: state.task,
                                    dayUid: state.task.uid,
                                  );
                                  if (thing is Thing) {
                                    await _cubit.updateTask(thing);
                                  }
                                },
                              ),
                              const SizedBox(height: 12),
                              Align(
                                alignment: Alignment.center,
                                child: VinchestaTextButton(
                                  title: context.strings.deleteThing,
                                  style: context.text.deleteThing,
                                  enabled: true,
                                  onButtonTap: () {
                                    _cubit.removeTask(widget.thing);
                                  },
                                ),
                              ),
                            ],
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              ),
            );
          });
}

Widget _buildLogButtons(ThingManagePageState state) {
  return Column(
    children: [
      VinchestaButton(
        content: Row(
          children: [
            Text(
              context.strings.logTime,
              style: context.text.loginPageButton,
            ),
            const SizedBox(width: 12),
            Icon(
              Icons.more_time,
              color: context.color.areaIconColor,
            ),
          ],
        ),
        onPressed: () {
```

```
            context.dialogs.showTimeDialog((time) {
              final hoursInMinutes = time.hour * 60;
              final minutes = time.minute;
              _cubit.editThing(state.task.copyWith(
                breakTime:
                    (state.task.breakTime ?? 0) + hoursInMinutes + minutes,
              ));
            });
          },
        ),
        VinchestaButton(
          content: Row(
            children: [
              Text(
                context.strings.removeLog,
                style: context.text.loginPageButton,
              ),
              const SizedBox(width: 12),
              Icon(
                Icons.update,
                color: context.color.areaIconColor,
              ),
            ],
          ),
          onPressed: () {
            context.dialogs.showDeletingTimeDialog((time) {
              final hoursInMinutes = time.hour * 60;
              final minutes = time.minute;
              if ((hoursInMinutes + minutes) > (state.task.breakTime ?? 0)) {
                _cubit.editThing(state.task.copyWith(
                  breakTime: 0,
                ));
              }
              _cubit.editThing(state.task.copyWith(
                breakTime:
                    (state.task.breakTime ?? 0) - (hoursInMinutes + minutes),
              ));
            });
          },
        ),
      ],
    );
  }

  Widget _buildSpecialActions(ThingManagePageState state) {
    return Column(
      children: [
        VinchestaButton(
          content: Row(
            children: [
              Text(
                context.strings.doWith90,
                style: context.text.loginPageButton,
              ),
              const SizedBox(width: 12),
              Icon(
                Icons.pending_actions,
                color: context.color.areaIconColor,
              )
            ],
          ),
          onPressed: () {
            router.pushNinetyThirtyTimerPage(() {
              _cubit.editThing(state.task.copyWith(
                progressStatus: ProgressStatus.done,
              ));
            });
          },
        ),
```

```
      VinchestaButton(
        content: Row(
          children: [
            Text(
              context.strings.doWith25,
              style: context.text.loginPageButton,
            ),
            const SizedBox(width: 12),
            Icon(
              Icons.pending_actions,
              color: context.color.areaIconColor,
            )
          ],
        ),
        onPressed: () {
          router.pushTwentyFiveTimerPage(() {
            _cubit.editThing(state.task.copyWith(
              progressStatus: ProgressStatus.done,
            ));
          });
        },
      ),
    ],
  );
}

Widget _buildActions(ThingManagePageState state) {
  switch (state.task.progressStatus) {
    case ProgressStatus.toDo:
      return _buildInProgressButton(state);
    case ProgressStatus.inProgress:
      return Column(
        children: [
          VinchestaButton(
            content: Row(
              children: [
                Text(
                  context.strings.moveToDo,
                  style: context.text.loginPageButton,
                ),
                const SizedBox(width: 12),
                Icon(
                  Icons.event_note,
                  color: context.color.areaIconColor,
                ),
              ],
            ),
            onPressed: () {
              _cubit.editThing(state.task.copyWith(
                progressStatus: ProgressStatus.toDo,
              ));
            },
          ),
          VinchestaButton(
            content: Row(
              children: [
                Text(
                  context.strings.moveToDone,
                  style: context.text.loginPageButton,
                ),
                const SizedBox(width: 12),
                Icon(
                  Icons.check,
                  color: context.color.appBackIcon,
                ),
              ],
            ),
            onPressed: () {
              _cubit.editThing(state.task.copyWith(
```

```
                progressStatus: ProgressStatus.done,
              ));
            },
          ),
        ],
      );
    case ProgressStatus.done:
      return _buildInProgressButton(state);
  }
}

Widget _buildInProgressButton(ThingManagePageState state) {
  return VinchestaButton(
    content: Row(
      children: [
        Text(
          context.strings.moveInProgress,
          style: context.text.loginPageButton,
        ),
        const SizedBox(width: 12),
        Icon(
          Icons.edit,
          color: context.color.inProgressColor,
        ),
      ],
    ),
    onPressed: () {
      _cubit.editThing(state.task.copyWith(
        progressStatus: ProgressStatus.inProgress,
      ));
    },
  );
}

Widget _buildStatuses(ThingManagePageState state) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(

'${context.strings.expressType}${widget.thing.expressStatus.getTitle(context)}',
        style: context.text.loginPageDescription,
      ),
      const Divider(),
      const SizedBox(height: 12),
      Text(

'${context.strings.importantType}${widget.thing.importantStatus.getTitle(context
)}',
        style: context.text.loginPageDescription,
      ),
      const Divider(),
      const SizedBox(height: 12),
      Text(

'${context.strings.progressStatus}${state.task.progressStatus.getLabel(context)}
',
        style: context.text.loginPageDescription.copyWith(
          color: state.task.progressStatus.getColor(context),
        ),
      ),
      const Divider(),
    ],
  );
}
}
```

// things_page.dart – файл що містить сторінку створення задач

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_form_builder/flutter_form_builder.dart';
import 'package:form_builder_validators/form_builder_validators.dart';
import 'package:uuid/uuid.dart';
import 'package:vinchesta_story/models/express_status.dart';
import 'package:vinchesta_story/models/important_status.dart';
import 'package:vinchesta_story/models/thing.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/utils/extensions/date_time_ext.dart';
import 'package:vinchesta_story/views/fileds/vinchesta_text_filed.dart';
import 'package:vinchesta_story/views/text_field_dropdown.dart';

import '../../config/di/locator.dart';
import '../../config/router/vinchesta_router.dart';
import '../../cubit/thing_page_cubit/thing_page_cubit.dart';
import '../../models/progress_status.dart';
import '../../utils/extensions/time_of_day_ext.dart';
import '../../views/vinchesta_appbar/vinchesta_appbar.dart';

enum _ThingsPageFields {
  title,
  express,
  importantStatus,
  workTime,
  description;
}

@RoutePage<Thing?>()
class ThingsPage extends StatefulWidget implements AutoRouteWrapper {
  final Thing? thing;
  final String? dayUid;

  const ThingsPage({
    super.key,
    this.thing,
    this.dayUid,
  });

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (_) => locator.get<ThingPageCubit>(),
      child: this,
    );
  }

  @override
  State<ThingsPage> createState() => _ThingsPageState();
}

class _ThingsPageState extends State<ThingsPage> {
  final _fbKey = GlobalKey<FormBuilderState>();

  DateTime? _dateTime;
  Thing? _thing;
  UniqueKey _timeKey = UniqueKey();
  FormBuilderState? get _fbState => _fbKey.currentState;
  Map<String, dynamic> get _fbValues => _fbState?.value ?? {};
  ThingPageCubit get _cubit => context.read<ThingPageCubit>();

  @override
  initState() {
    super.initState();
    _thing = widget.thing;
  }
```

```
String? _getInitialTime(BuildContext context) {
  if (_dateTime?.formattedTime is String) {
    _fbState?.patchValue({
      _ThingsPageFields.workTime.name: _dateTime?.formattedTime,
    });
    return _dateTime?.formattedTime;
  }
  if (widget.thing?.workTime != null) {
    final worked = widget.thing?.getWorkedTimeOfDay();
    return TimeOfDayExt.formmatedTime(
      worked ?? const TimeOfDay(hour: 0, minute: 0),
      context,
    );
  }
  return null;
}

void _onStateChanged(
  BuildContext context,
  ThingPageState state,
) {
  switch (state.status) {
    case ThingPageStatus.success:
      router.pop(_thing);
      break;

    default:
      break;
  }
}

int _parseWorkTime(String workTime) {
  List<String> parts = workTime.split(':');
  int hours = int.parse(parts[0]) * 60;
  int minutes = int.parse(parts[1]);
  return hours + minutes;
}

@override
Widget build(BuildContext context) {
  return BlocConsumer<ThingPageCubit, ThingPageState>(
    listener: _onStateChanged,
    builder: (context, state) {
      return Scaffold(
        appBar: VinchestaAppBar(
          title: context.strings.createThing,
          trailingIconType: TrailingIconType.check,
          onTrailingTap: () {
            _fbState?.save();

            final express = _fbValues[_ThingsPageFields.express.name];
            final title = _fbValues[_ThingsPageFields.title.name];
            final workTime = _fbValues[_ThingsPageFields.workTime.name];
            final description = _fbValues[_ThingsPageFields.description.name];
            final important =
                _fbValues[_ThingsPageFields.importantStatus.name];
            final areDropdownsValid = express != null && important != null;

            const Uuid uid = Uuid();
            final dayUid = widget.dayUid;
            final thing = widget.thing;
            if ((_fbState?.saveAndValidate() ?? false) &&
                areDropdownsValid &&
                thing != null) {
              _thing = Thing(
                uid: thing.uid,
                title: title,
                dayUid: thing.dayUid,
                userId: thing.userId,
```

```
            description: description,
            expressStatus: express,
            importantStatus: important,
            workTime: _parseWorkTime(workTime),
            loggedTime: thing.loggedTime,
            progressStatus: ProgressStatus.inProgress,
          );
          _cubit.editTask(_thing!);
        }

        if ((_fbState?.saveAndValidate() ?? false) &&
            areDropdownsValid &&
            dayUid != null) {
          _cubit.postTask(
            Thing(
              uid: uid.v4(),
              title: title,
              dayUid: dayUid,
              userId: state.userId!,
              description: description,
              expressStatus: express,
              workTime: _parseWorkTime(workTime),
              importantStatus: important,
              progressStatus: ProgressStatus.toDo,
            ),
          );
        } else {
          context.toasts.showError(
            message: context.strings.fillInAllRequiredFields,
          );
        }
      },
    ),
    body: Container(
      decoration: BoxDecoration(
        gradient: context.gradient.primaryGradient,
      ),
      child: FormBuilder(
        key: _fbKey,
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(
              horizontal: 20,
              vertical: 20,
            ),
            child: Column(
              children: [
                VinchestaTextField(
                  name: _ThingsPageFields.title.name,
                  labelStyle: context.text.vinchestaTextField,
                  labelText: context.strings.titleThing,
                  initialValue: widget.thing?.title,
                  validator: FormBuilderValidators.compose([
                    FormBuilderValidators.required(),
                  ]),
                ),
                const SizedBox(height: 20),
                TextFieldDropdown<ExpressStatus>.filled(
                  name: _ThingsPageFields.express.name,
                  items: ExpressStatus.values,
                  initialValue: widget.thing?.expressStatus,
                  hintText: context.strings.chooseExpressStatus,
                  getLabel: (item) {
                    return item.getTitle(context);
                  },
                  onChanged: (_) {},
                ),
                const SizedBox(height: 20),
                TextFieldDropdown<ImportantStatus>.filled(
```

```
                        name: _ThingsPageFields.importantStatus.name,
                        items: ImportantStatus.values,
                        initialValue: widget.thing?.importantStatus,
                        hintText: context.strings.chooseImportantStatus,
                        getLabel: (item) {
                          return item.getTitle(context);
                        },
                        onChanged: (_) {},
                      ),
                      const SizedBox(height: 20),
                      GestureDetector(
                        onTap: () async {
                          final dateTime = await router.pushTimePickerPage();
                          if (dateTime is DateTime) {
                            setState(() {
                              _dateTime = dateTime;
                              _timeKey = UniqueKey();
                            });
                          }
                        },
                        child: VinchestaTextField(
                          key: _timeKey,
                          name: _ThingsPageFields.workTime.name,
                          enabled: false,
                          onTap: () {},
                          suffix: Padding(
                            padding: const EdgeInsets.only(top: 12.0),
                            child: Text(
                              context.strings.hMFormat,
                              style: context.text.mainPageTileLabel,
                            ),
                          ),
                          initialValue: _getInitialTime(context),
                          labelStyle: context.text.vinchestaTextField,
                          labelText: context.strings.workTime,
                        ),
                      ),
                      const SizedBox(height: 20),
                      VinchestaTextField(
                        helperText: context.strings.description,
                        labelStyle: context.text.vinchestaTextField,
                        initialValue: widget.thing?.description,
                        name: _ThingsPageFields.description.name,
                        maxLines: 10,
                      ),
                    ],
                  ),
                ),
              ),
            ),
          ),
        ),
      );
    },
  );
  }
}
```

// twenty_fiver_timer_page.dart – файл що містить сторінку з методикою Pomodoro

```
import 'package:auto_route/auto_route.dart';
import 'package:flutter/material.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_appbar/vinchesta_appbar.dart';

import '../../views/vinchesta_timer.dart';
```

```
@RoutePage()
class TwentyFiverTimerPage extends StatelessWidget {
  final VoidCallback onDoneTap;

  const TwentyFiverTimerPage({
    super.key,
    required this.onDoneTap,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: VinchestaAppBar(
        title: context.strings.twentyFiveMethod,
      ),
      body: Container(
        width: double.maxFinite,
        decoration: BoxDecoration(
          gradient: context.gradient.primaryGradient,
        ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
            const SizedBox(height: 40),
            VinchestaTimer(
              workTime: 1500,
              onDoneTap: onDoneTap,
              breakTime: 300,
            ),
          ],
        ),
      ),
    );
  }
}
```

// preference_service.dart – файл що містить сервіс з локальною базою даних

```
import 'dart:ui';

import 'package:hive/hive.dart';
import 'package:injectable/injectable.dart';
import 'package:jwt_decoder/jwt_decoder.dart';
```

```dart
import '../models/auth_details.dart';

@injectable
@preResolve
class PreferencesService {
  static const _preferencesBox = '_preferencesBox';

  static const _localeKey = '_localeKey';
  static const _authDetailsKey = '_authDetailsKey';

  final Box<dynamic> _box;

  PreferencesService._(this._box);

  @factoryMethod
  static Future<PreferencesService> getInstance() async {
    final box = await Hive.openBox<dynamic>(_preferencesBox);
    return PreferencesService._(box);
  }

  /// Localization

  String getDefaultLanguageCode() {
    return 'en';
  }

  Locale getLocale() {
    final languageCode = _box.get(_localeKey) ?? getDefaultLanguageCode();
    return Locale(languageCode);
  }

  Locale? getNullbleLocale() {
    final languageCode = _box.get(_localeKey);
    if (languageCode is String) {
      return Locale(languageCode);
    }
    return null;
  }

  Future<void> setLocale(Locale locale) async {
    await _box.put(_localeKey, locale.languageCode);
  }

  /// Auth Details

  bool get hasToken => getAccessToken().isNotEmpty;
```

```
  bool get isTokenExpired {
    final token = getAccessToken();

    if (token.isEmpty) return true;

    return JwtDecoder.isExpired(token);
  }

  bool get isLoggedIn => hasToken && !isTokenExpired;

  String getAccessToken() => getAuthDetails()?.accessJwtToken ?? '';

  int? getUserId() => getAuthDetails()?.userId;

  AuthDetails? getAuthDetails() => _box.get(_authDetailsKey);

  Future<void> setAuthDetails(AuthDetails authDetails) async {
    await _box.put(_authDetailsKey, authDetails);
  }

  Future<void> logout() async {
    // TODO: reset singletons and clear hive

    await _box.delete(_authDetailsKey);
  }
}

import 'package:injectable/injectable.dart';
import 'package:vinchesta_story/api/request/login_request.dart';
import 'package:vinchesta_story/api/request/register_request.dart';
import 'package:vinchesta_story/models/auth_details.dart';
import 'package:vinchesta_story/services/preference_service.dart';

import '../../api/request/edit_user_request.dart';
import '../../api/response/register_response.dart';
import '../../api/vinchesta_api_client.dart';
import '../base_service.dart';
```

// user_service.dart – файл що містить сервіс для користувача

```
@injectable
class UserService extends BaseService {
  final VinchestaApiClient _apiClient;
  final PreferencesService _preferencesService;
```

```
UserService(
  this._apiClient,
  this._preferencesService,
);

Future<void> login(LoginRequest loginRequest) async {
  return makeErrorParsedCall(() async {
    final accessToken = await _apiClient.login(
      LoginRequest(
        email: loginRequest.email,
        password: loginRequest.password,
      ),
    );

    await _preferencesService.setAuthDetails(
      AuthDetails(
        userId: 0,
        accessJwtToken: accessToken.accessToken,
      ),
    );
    final user = await _apiClient.getUser();

    await _preferencesService.setAuthDetails(
      AuthDetails(
        userId: user.id,
        accessJwtToken: accessToken.accessToken,
      ),
    );
  });
}

Future<void> deleteUser(int userId) async {
  return makeErrorParsedCall(() async {
    await _apiClient.removeUser(userId: userId);
  });
}

Future<void> editUser(EditUserRequest editUserRequest, int userId) async {
  return makeErrorParsedCall(() async {
    await _apiClient.editUser(
        userId: userId, editUserRequest: editUserRequest);
  });
}

Future<RegisterResponse> getUser() async {
  return makeErrorParsedCall(() async {
```

```
      final user = await _apiClient.getUser();

      return user;
    });
  }

  Future<RegisterResponse> register(RegisterRequest registerRequest) async {
    return makeErrorParsedCall(() async {
      final response = await _apiClient.register(registerRequest);

      final accessToken = await _apiClient.login(LoginRequest(
        email: response.email,
        password: registerRequest.password,
      ));

      await _preferencesService.setAuthDetails(
        AuthDetails(
          userId: response.id,
          accessJwtToken: accessToken.accessToken,
        ),
      );
      return response;
    });
  }
}
```

// task_service.dart – файл що містить сервіс для задач

```
import 'package:injectable/injectable.dart';

import '../api/vinchesta_api_client.dart';
import '../models/thing.dart';
import 'base_service.dart';

@injectable
class TaskService extends BaseService {
  final VinchestaApiClient _apiClient;

  TaskService(
    this._apiClient,
  );

  Future<List<Thing>> getTask(String dayThingUid) async {
    return makeErrorParsedCall(() async {
      final task = await _apiClient.getTask(dayThingUid: dayThingUid);
      return task;
```

```
    });
  }

  Future<Thing> editTask(Thing thing) async {
    return makeErrorParsedCall(() async {
      return await _apiClient.editTask(
        thing: thing,
        thingUid: thing.uid,
      );
    });
  }

  Future<void> deleteTask(String thingUid) async {
    return makeErrorParsedCall(() async {
      return await _apiClient.deleteTask(thingUid: thingUid);
    });
  }

  Future<Thing> postTask(Thing thing) async {
    return makeErrorParsedCall(() async {
      return await _apiClient.postTask(thing: thing);
    });
  }
}
```

// views – директорія що містить створені(custom) віджети

```
import 'package:audioplayers/audioplayers.dart';
import 'package:flutter/material.dart';
import 'package:sizer/sizer.dart';
import 'package:timer_count_down/timer_controller.dart';
import 'package:timer_count_down/timer_count_down.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';
import 'package:vinchesta_story/views/vinchesta_buttons.dart';

import '../config/router/vinchesta_router.dart';

class VinchestaTimer extends StatefulWidget {
  final int workTime;
  final int breakTime;
  final bool isLongMethod;
  final VoidCallback onDoneTap;

  const VinchestaTimer({
    super.key,
    required this.workTime,
```

```
    required this.breakTime,
    required this.onDoneTap,
    this.isLongMethod = false,
  });

  @override
  _VinchestaTimerState createState() => _VinchestaTimerState();
}

class _VinchestaTimerState extends State<VinchestaTimer> {
  // TODO: move to init
  final CountdownController _controller = CountdownController(autoStart: true);
  AudioPlayer audioPlayer = AudioPlayer();
  int _duration = 1500;
  int _break = 300;
  bool _pause = false;
  UniqueKey uniqueKey = UniqueKey();

  @override
  void initState() {
    super.initState();
    _duration = widget.workTime;
    _break = widget.breakTime;
  }

  Future<void> _toggleDuration() async {
    await audioPlayer.play(AssetSource('sounds/break_sound.mp3'));
    setState(() {
      _duration = _duration == widget.workTime ? _break : widget.workTime;
      _controller.restart();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text(
          _duration != widget.workTime
              ? context.strings.break1
              : context.strings.work,
          style: context.text.loginPageTitle.copyWith(
              color: _duration != widget.workTime
                  ? context.color.appBackIcon
                  : context.color.workTimeColor),
        ),
```

```
const SizedBox(height: 40),
Countdown(
    key: uniqueKey,
    controller: _controller,
    seconds: _duration,
    build: (_, double time) {
      final minutes = (time ~/ 60).toString().padLeft(2, '0');
      final seconds = (time % 60).toStringAsFixed(0).padLeft(2, '0');

      return Stack(
        alignment: Alignment.center,
        children: [
          SizedBox(
            width: MediaQuery.of(context).size.width / 2,
            height: MediaQuery.of(context).size.width / 2,
            child: CircularProgressIndicator(
              backgroundColor: context.color.dayTileColor,
              value: time / _duration,
              strokeWidth: 10,
              color: _duration != widget.workTime
                  ? context.color.appBackIcon
                  : context.color.workTimeColor,
            ),
          ),
          Text(
            '$minutes:$seconds',
            style: context.text.thingTileLabel.copyWith(
              fontSize: 18.sp,
              fontWeight: FontWeight.w600,
              color: _duration != widget.workTime
                  ? context.color.appBackIcon
                  : context.color.workTimeColor,
            ),
          ),
        ],
      );
    },
    onFinished: () async {
      await _toggleDuration();
    }),
const SizedBox(height: 34),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 24.0),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
```

```
        VinchestaButton(
          key: uniqueKey,
          content: Row(
            children: [
              Text(
                !_pause ? context.strings.pause : context.strings.play,
                style: context.text.loginPageButton,
              ),
              const SizedBox(width: 16),
              Icon(
                !_pause ? Icons.pause : Icons.chevron_right,
                color: context.color.appBackIcon,
              ),
            ],
          ),
          onPressed: () {
            _pause = !_pause;
            !_pause ? _controller.resume() : _controller.pause();
            setState(() {});
          },
        ),
        VinchestaButton(
          content: Row(
            children: [
              Text(
                context.strings.statusDone,
                style: context.text.loginPageButton,
              ),
              const SizedBox(width: 16),
              Icon(
                Icons.done,
                color: context.color.appBackIcon,
              ),
            ],
          ),
          onPressed: () {
            widget.onDoneTap.call();
            router.pop();
          },
        ),
      ],
    ),
  ),
  const SizedBox(height: 40),
  Padding(
    padding: const EdgeInsets.symmetric(horizontal: 24),
```

```
          child: Text(
            widget.isLongMethod
                ? context.strings.ninetyDescription
                : context.strings.thirtyDescription,
            style: context.text.thingTileLabel,
          ),
        ),
      ],
    );
  }
}

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:sizer/sizer.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';

class SelectTimeWidget extends StatefulWidget {
  final DateTime initialTime;
  final ValueSetter<DateTime> onChanged;

  const SelectTimeWidget({
    super.key,
    required this.initialTime,
    required this.onChanged,
  });

  @override
  State<SelectTimeWidget> createState() => _SelectTimeWidgetState();
}

class _SelectTimeWidgetState extends State<SelectTimeWidget> {
  static const double _scrollItemExtent = 84;
  static const int _maxHours = 23;
  static const int _maxMinutes = 59;

  final List<int> _hourScrollValues = [];
  final List<int> _minuteScrollValues = [];

  late DateTime _selectTime;

  late FixedExtentScrollController _firstFixedExtentScrollController;
  late FixedExtentScrollController _secondFixedExtentScrollController;

  @override
  void initState() {
```

```
  super.initState();
  _selectTime = widget.initialTime;
  _setHoursScrollValues();
  _setMinutesScrollValues();
  _setScrollsInitialValues();
}

@override
void dispose() {
  _firstFixedExtentScrollController.dispose();
  _secondFixedExtentScrollController.dispose();
  super.dispose();
}

/// Calculating size of TextWidget
Size _calculateTextSize(
  BuildContext context,
  String text,
  TextStyle textStyle,
) {
  return (TextPainter(
    text: TextSpan(text: text, style: textStyle),
    maxLines: 1,
    textScaler: MediaQuery.of(context).textScaler,
    textDirection: TextDirection.ltr,
  )..layout())
      .size;
}

void _setScrollsInitialValues() {
  _firstFixedExtentScrollController = FixedExtentScrollController(
    initialItem: _hourScrollValues.indexOf(
      widget.initialTime.hour,
    ),
  );
  _secondFixedExtentScrollController = FixedExtentScrollController(
    initialItem: _minuteScrollValues.indexOf(widget.initialTime.minute),
  );
}

void _setHoursScrollValues() {
  for (int i = 0; i <= _maxHours; i++) {
    _hourScrollValues.add(i);
  }
}
```

```
void _setMinutesScrollValues() {
  for (int i = 0; i <= _maxMinutes; i++) {
    _minuteScrollValues.add(i);
  }
}

void _onTimeChanged(int index, bool isHour) {
  setState(() {
    if (isHour) {
      _firstFixedExtentScrollController =
          FixedExtentScrollController(initialItem: index);
      _selectTime = _selectTime.copyWith(hour: _hourScrollValues[index]);
    } else {
      _secondFixedExtentScrollController =
          FixedExtentScrollController(initialItem: index);
      _selectTime = _selectTime.copyWith(minute: _minuteScrollValues[index]);
    }
  });

  widget.onChanged(_selectTime);
}

String _getTimePointWithZero(int value) {
  return value.toString().padLeft(2, '0');
}

@override
Widget build(BuildContext context) {
  return LayoutBuilder(
    builder: (context, constraints) {
      return _buildBody(constraints.maxHeight);
    },
  );
}

Widget _buildBody(double scrollPickerHeight) {
  final size = _calculateTextSize(
    context,
    _getTimePointWithZero(_selectTime.minute),
    context.text.vinchestaTimePickerValue,
  );
  final double topSeparateLinePosition =
      scrollPickerHeight / 2 - size.height * 2.4;
  return Stack(
    alignment: Alignment.center,
    children: [
```

```
      _buildScrollPickers(),
      Positioned(
        top: topSeparateLinePosition,
        right: 0,
        left: 0,
        child: _buildSeparateLines(
          size,
        ),
      ),
    ],
  );
}

Widget _buildScrollPickers() {
  return Column(
    children: [
      Expanded(
        flex: 2,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            SizedBox(
              width: 20.w,
              child: _buildScrollPicker(true),
            ),
            Padding(
              padding: const EdgeInsets.only(bottom: 48),
              child: Text(
                ':',
                  style: context.text.vinchestaTimePickerSelectValue,
              ),
            ),
            SizedBox(
              width: 20.w,
              child: _buildScrollPicker(false),
            ),
          ],
        ),
      ),
    ],
  );
}

Widget _buildSeparateLines(Size fontSize) {
  return Column(
```

```
    children: [
      Divider(
        height: 1,
        color: context.color.dayTileColor,
      ),
      SizedBox(height: fontSize.height * 2.5),
      Divider(
        height: 1,
        color: context.color.dayTileColor,
      ),
    ],
  );
}

Widget _buildScrollPicker(bool isHourScroll) {
  return CupertinoPicker(
    looping: true,
    diameterRatio: 44,
    itemExtent: _scrollItemExtent,
    squeeze: 1,
    magnification: 1,
    useMagnifier: true,
    selectionOverlay: const CupertinoPickerDefaultSelectionOverlay(
      background: Colors.transparent,
    ),
    scrollController: isHourScroll
        ? _firstFixedExtentScrollController
        : _secondFixedExtentScrollController,
    onSelectedItemChanged: (int index) {
      _onTimeChanged(index, isHourScroll);
    },
    children: _buildScrollLabels(isHourScroll),
  );
}

List<Widget> _buildScrollLabels(bool isHourScroll) {
  if (isHourScroll) {
    return _hourScrollValues.map(
      (value) {
        return Text(
          _getTimePointWithZero(value),
          style: _selectTime.hour == value
              ? context.text.vinchestaTimePickerSelectValue
              : context.text.vinchestaTimePickerValue,
        );
      },
```

```
      ).toList();
    }
    return _minuteScrollValues
        .map(
          (value) => Text(
            _getTimePointWithZero(value),
            style: _selectTime.minute == value
                ? context.text.vinchestaTimePickerSelectValue
                : context.text.vinchestaTimePickerValue,
          ),
        )
        .toList();
  }
}

import 'package:flutter/material.dart';
import 'package:sizer/sizer.dart';
import 'package:vinchesta_story/utils/extensions/build_context_ext.dart';

import '../models/base_tab_type.dart';

class VinchestaBottomNavigationBar<TEnum extends ComplicatedBaseTabType>
    extends StatelessWidget {
  final List<TEnum> items;
  final int activeIndex;
  final ValueSetter<int> onActiveIndexChanged;

  VinchestaBottomNavigationBar({
    super.key,
    required this.items,
    required this.activeIndex,
    required this.onActiveIndexChanged,
  });

  final double iconSize = 20.sp;

  Color _getColor({
    required BuildContext context,
    required int index,
  }) {
    if (index == activeIndex) {
      return context.color.appBackIcon;
    }

    return context.color.dayTileColor;
  }
```

```
@override
Widget build(BuildContext context) {
  return SafeArea(
    top: false,
    maintainBottomViewPadding: false,
    child: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        const Divider(
          height: 0,
          color: Colors.white12,
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: List.generate(
            items.length,
            (index) {
              return _buildTab(
                context,
                index,
              );
            },
          ),
        ),
      ],
    ),
  );
}

Widget _buildTab(BuildContext context, int index) {
  final item = items[index];
  final color = _getColor(
    context: context,
    index: index,
  );

  return GestureDetector(
    behavior: HitTestBehavior.opaque,
    onTap: () => onActiveIndexChanged(index),
    child: Padding(
      padding: const EdgeInsets.symmetric(
        horizontal: 12,
        vertical: 12,
      ),
      child: Stack(
```

```
          children: [
            Column(
              children: [
                Icon(
                  item.iconData,
                  size: iconSize,
                  color: color,
                ),
                const SizedBox(height: 4),
                Text(
                  style: context.text.thingTileLabel,
                  item.getLabel(context) ?? '',
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}

import 'package:flutter/material.dart';

class RedrawableScrollView extends StatefulWidget {
  final List<Widget> children;
  final bool reverse;
  final EdgeInsets margin;
  final ScrollController? scrollController;

  const RedrawableScrollView({
    super.key,
    required this.children,
    this.reverse = false,
    this.margin = const EdgeInsets.symmetric(horizontal: 16),
    this.scrollController,
  });

  @override
  State<RedrawableScrollView> createState() => _RedrawableScrollViewState();
}

class _RedrawableScrollViewState extends State<RedrawableScrollView> {
  late ScrollController _scrollController;

  @override
```

```
  void initState() {
    super.initState();
    _scrollController = widget.scrollController ?? ScrollController();
  }

  @override
  void dispose() {
    _scrollController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(
      builder: (context, cons) {
        return SingleChildScrollView(
          controller: _scrollController,
          reverse: widget.reverse,
          child: ConstrainedBox(
            constraints: BoxConstraints(
              minWidth: cons.maxWidth,
              minHeight: cons.maxHeight,
            ),
            child: IntrinsicHeight(
              child: Padding(
                padding: widget.margin,
                child: Column(
                  mainAxisSize: MainAxisSize.max,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  children: widget.children,
                ),
              ),
            ),
          ),
        );
      },
    );
  }
}
```