

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
“ ____ ” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

**“ Програмне забезпечення системи інтелектуального аналізу даних для
управління звичками людини”**

Виконав здобувач вищої освіти
IV курсу, групи _____
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні
науки»
_____ Рудь І.В.
« ____ » _____ 2026р.

Керівник проекту
кандидат технічних наук, доцент
_____ Доренський О.П.
« ____ » _____ 2026р.
Рецензент _____

Центральноукраїнський національний технічний університет

Факультет Механіко-технологічний

Кафедра Кібербезпеки та програмного забезпечення

Рівень вищої освіти бакалавр

Галузь знань ._12 _____ “Інформаційні технології”

Спеціальність 122 “Комп’ютерні науки”

Освітньо-професійна (освітньо-наукова) програма “ Комп’ютерні науки ”

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

_____ Олексій СМІРНОВ

“ _____ ” _____ 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Рудю Ігорю Вадимовичу

(прізвище, ім’я, по батькові)

1. Тема роботи Програмне забезпечення системи інтелектуального аналізу даних для управління звичками людини

2. Керівник роботи Доренський Олександр Павлович, канд. техн. наук, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 06 ” 02 ” 20 26 року № 116-02

3. Строк подання роботи до захисту 01.06.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи Метою роботи є розробка програмного забезпечення для покращення системи управління звичками людини за допомогою інтелектуального аналізу даних

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. Реалізація роботи.

5. Впровадження системи в експлуатацію

6. Висновки.

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

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 3 аркуша

7. Дата видачі завдання «06» 02 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	30.04.2026 р.	
8.	Попередній захист роботи	16.05.2026 р.	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

(прізвище та ініціали)

АНОТАЦІЯ

Рудь І. В. Програмне забезпечення системи інтелектуального аналізу даних для управління звичками людини. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В цій кваліфікаційній бакалаврській роботі розроблено програмне забезпечення системи, яке призначення для покращення системи управління звичками людини за допомогою інтелектуального аналізу даних, побудованого на основі генеративного штучного інтелекту.

Метою розробки є ПЗ інтелектуального аналізу даних у системах управління звичками людини, який використовує можливості генеративного штучного інтелекту.

В процесі роботи проаналізовано існуючі системами управління звичками людини, технологіями, архітектурами та програмними рішеннями. Після аналізу прийнято рішення використовувати сервіс OpenAI – Responses API, задля інтелектуального аналізу даних, в ролі генеративного ШІ, веб-фреймворк Laravel, як бекенд фреймворк на мові програмування PHP та хмарне середовище AWS, задля розгортання проєкту.

ПЗ використовує протокол HTTP для передачі даних через мережу Інтернет та JWT-токени для аутентифікації користувача в системі. Для реалізації ПЗ використано мову програмування PHP.

Для реалізації інтерфейсу розроблено гнучкий програмний інтерфейс - API. Для зберігання даних обрано реляційну базу даних. Вся інформація через API передається у JSON форматі.

Результат роботи – створено програмне забезпечення інтелектуального аналізу даних у системах управління звичками людини на основі Responses API сервісу генеративного штучного інтелекту OpenAI.

Ключові слова: ПЗ, система, аналіз, API, ШІ, БД, OpenAI.

ABSTRACT

Rud I.V. Intelligent data analysis system software for managing human habits. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This bachelor's thesis develops software designed to improve human habit management systems through intelligent data analysis based on generative artificial intelligence.

The goal of the development is to create software for intelligent data analysis in human habit management systems that utilizes the capabilities of generative artificial intelligence.

In the course of the work, existing human habit management systems, technologies, architectures, and software solutions were analyzed. After analysis, it was decided to use the OpenAI – Responses API service for intelligent data analysis as generative AI, the Laravel web framework as a backend framework in the PHP programming language, and the AWS cloud environment for project deployment.

The software uses the HTTP protocol to transfer data over the Internet and JWT tokens for user authentication in the system. The PHP programming language was used to implement the software.

A flexible software interface (API) was developed to implement the interface. A relational database was chosen for data storage. All information is transmitted via the API in JSON format.

The result of the work is the creation of software for intelligent data analysis in human habit management systems based on the Responses API service of the OpenAI generative artificial intelligence.

Keywords: software, system, analysis, API, AI, database, OpenAI.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ	2
ВСТУП	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 ПРИЗНАЧЕННЯ СИСТЕМИ	5
1.2 ОБЛАСТЬ ЗАСТОСУВАННЯ	6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень ...	9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	32
2.3 Розгорнута постановка завдання	34
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ	36
3.1 Опис функціонування системи	39
3.2 Розробка структурної схеми	41
3.3 Розробка функціональної схеми	44
3.4 Розробка діаграми процесів	47
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЄКТНИХ І ПРОГРАМНИХ РІШЕНЬ	51
4.1 Розробка блок-схем та опис алгоритмів функціонування системи	51
4.2 Захист розробленого програмного забезпечення	61
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ЕКСПЛУАТАЦІЮ	64
ОСНОВНІ ВИСНОВКИ	67
СПИСОК ДЖЕРЕЛ	71

					ВКРБ-122.26.0026.00.00.ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Рудь І.В.			Програмне забезпечення системи інтелектуального аналізу даних для управління звичками людини	Літ.	Арк.	Акрушів
Перевір.		Доренський О.П.				Б	1	74
						ЦНТУ КН-22		
Н. Контр.		Коваленко А.С.						
Затверд.		Смірнов О.А.						

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IT - Information Technology
VPS - Virtual Private Server
WWW - World Wide Web
LLM - Large Language Model
TMA - Telegram Mini App
MIT - Massachusetts Institute of Technology
API - Application Programming Interface
JSON - JavaScript Object Notation
XML - Extensible Markup Language
CI/CD - Continuous Integration and Continuous Development
HTTP - Hyper Text Transfer Protocol
SQL - Structured Query Language
AES - Advanced Encryption Standard
TDD - Test Driven Development
UI/UX - User Interface and User Experience
TCP - Transmission Control Protocol
HMAC - Hash-based Message Authentication Code
RSA - Rivest-Shamir-Adleman
EDCSA - Elliptic Curve Digital Signature Algorithm
UUID - Universally Unique Identifier
CSRF - Cross-Site Request Forgery-
MFA - Multi-Factor Authentication
AMI - Amazon Machine Image
RDS - Relational Database Service
AWS - Amazon Web Services
GUI - Graphical User Interface
ML - Machine Learning

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Інформаційні технології (ІТ) стали невід’ємною частиною життя кожної людини на планеті та вплинули на всі сфери життя. Поява всесвітньої павутини під назвою WWW або просто Web ознаменувала нову епоху, революцію у тому, як люди взаємодіють між собою та новим світом під назвою Інтернет. Інтернет дав нам багато нових можливостей від спілкування в соціальних мережах до придбання товарів в інтернет-магазинах, що дає простір для багатьох можливостей цифровізації для яких ми можемо тільки-но уявити.

“Звичка – це автоматизована дія або патерн поведінки, який виникає в результаті повторення. Вона може бути позитивною або негативною, і вона формується через повторення певної дії в однаковому контексті або під впливом певного сприйняття, наприклад, потреби, емоційного стану або зовнішнього спонукача.”[1].

Інтелектуальний аналіз даних – це процес формування звички, який є індивідуальним для кожної людини, але в загальному випадку, вчені виділяють декілька етапів формування звичок:

1. Стимул – це сигнал, який змушує нас зробити дію або ж інакше кажучи, це мотивація.
2. Відповідь – це певний стан, який ми отримуємо в результаті виконання дії під впливом стимулу.
3. Нагорода – відчуття позитивне або негативне, яке отримується після виконання дії.
4. Зв’язок – асоціація між стимулом та нагородою, яку отримано в результаті певної дії.
5. Автоматизація – останній етап формування звички, коли звичка виконується майже безсвідомо. Про це, іноді кажуть: “зробив на автоматі”

Інтелектуальним аналізом даних в системах управління звичками людини є процес виявлення невідомих патернів, або цікавих ідей, як можна ефективніше позбутися або ж набутися ту чи іншу автоматизовану дію[2].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

Маючи програмне забезпечення, яке буде імплементувати інтелектуальний аналіз даних за допомогою можливостей штучного інтелекту можна аналізувати відомі патерні, а також, що є не менш важливим виявляти невідомі патерни.

У сучасному світі швидкого допаміну, спостерігається зниження уваги до звичок, і як результат ріст попиту на застосунки, які націлені на вирішення нагальних проблем, а саме формування звичок. Відсутність саморефлексії під час набуття нових звичок стала неабиякою проблемою сьогодення.

Підсумувавши вищесказане, можемо зробити висновок, що потреба у програмному забезпеченні, яке перш за все націлене на інтелектуальний аналіз даних у системах управління звичками людини може стати не тільки-но революційним, а й монетизованим згодом. Інтелектуальний аналіз даних згодом можна розширити не тільки для однієї людини, а для групи людей.

Мета й завдання дослідження. Мета роботи полягає у створенні ПЗ для покращення системи управління звичками людини за допомогою інтелектуального аналізу даних, побудованого на основі генеративного штучного інтелекту.

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

- Ретельний огляд існуючих систем управління звичками людини.
- Програмна реалізація ПЗ системи інтелектуального аналізу даних для управління звичками людини.

Практична цінність отриманих результатів полягає в тому, щоб надати змогу кінцевому користувачеві використовувати можливості ШІ для інтелектуального аналізу звичок.

Частина результатів цієї бакалаврської роботи апробована і схвалена на VII Міжнародній конференції молодих учених «Актуальні питання розвитку інформаційних технологій» (м. Дніпро, 19 листопада 2025 р.), а також опублікована[39].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Розроблена система управління звичками людини підкріплена інтелектуальним аналізом даних за допомогою ШІ, дозволить значно ефективніше аналізувати звички та бачити неочевидні патерни. Основна проблема, що дозволить вирішити ШІ, це виявлення скритих патернів, які заважають подоланню поганої звички.

Запропоноване ПЗ забезпечить користувача доступом до аналізу своїх звичок на основі створених нотаток, таким чином, можна віддати нотатки ШІ для аналізу та отримати висновок щодо покращення конкретної звички.

Цільова аудиторія – це люди, які користуються програмним забезпеченням або яким потрібен продукт чи послуга. Незнання цільової аудиторії це поганий початок будь-якого продукту. Автор ПЗ повинен знати портрет свого користувача для більш якіснішого таргетування аудиторії згодом.

1.1 Призначення системи

Цільова аудиторія – це користувачі розробленого ПЗ. Користувачів потрібно визначити не тільки для того, щоб скоротити витрати на рекламу в майбутньому, а й для покращення розробленого продукту. У кожного товару є свій покупець, так можна й сказати про програмне забезпечення: у кожного програмного забезпечення є свої користувачі. Цільова аудиторія для ПЗ системи інтелектуального аналізу даних для управління звичками може бути різною, для цього потрібно звернути увагу на існуючі аналоги та продукти, тобто інші програмні забезпечення і скласти портрет цільової аудиторії. Варто зазначити, що цей метод не є стовідсотково вірним, він є лише наближеним. Отже, люди, які мають проблеми зі звичками та шукають нові методи набуття хороших звичок, і позбуття поганих, вже є першим кроком для створення портрету цільової аудиторії. Також це люди свідомі, перш за все вони усвідомлюють, що

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

мають проблеми зі звичками і хочуть владнати цю сферу свого життя. Молоді та соціальні люди, які хочуть знайомитися з іншими однодумцями та набувати звичок разом – і є соціальний чинник ПЗ.

1.2 Область застосування

Платформа або місце, де буде розгортатись програмне забезпечення є однією з найважливіших частин запуску ПЗ. Але для того, щоб вибрати платформу, потрібно з'ясувати, де розгортаються наявні подібні ПЗ до ПЗ системи інтелектуального аналізу даних для управління звичками людини. З найпопулярніших відзначимо:

1. App Store.
2. Google Play.
3. Telegram Mini App.

Погляньмо на кожну з цих платформ детальніше, і ми зможемо помітити, як вони відрізняються одна від одної, тому що це є один з найважливіших кроків, від якого буде будуватись процес розробки ПЗ. Помилка на ранньому етапі може завдати нищівних збитків як у витраченому часі на розробку так і у витрачених грошах на етапі розгортання проєкту на платформі. Варто зазначити, що кожна платформа має свої особливості, які також важливо взяти до уваги, перш ніж обирати ту чи іншу платформу.

App Store – магазин мобільних та десктопних додатків розроблений компанією Apple для операційних систем iOS та iPadOS. Як і будь-яка диджитал платформа для мобільних додатків App Store надає змогу своїм користувачам шукати додатки, вибирати різні категорії, дивитись за рейтингами та цінами. Деякі з додатків в App Store є платними. Варто зазначити, що для того, щоб розмістити додаток в App Store, потрібно мати Apple Developer Account. Ставши частиною Apple Developer Program, можна досягти потенційних клієнтів через App Store для всіх платформ iOS, iPadOS, macOS. Членство забезпечує всіма

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

необхідними інструментами, ресурсами та підтримкою для того, щоб розповсюджувати додатки та ігри з легкістю[4].

Google Play – платформа для публікації переважно мобільних додатків під операційною системою Android та в деяких випадках ChromeOS розроблена в компанії Google та запущена у вересні 2008 року. Як і більшість інших схожих платформ, Google Play містить як платні додатки, так і безкоштовні. Google Play є однією з найбільших платформ налічуючи 3.5 мільйона додатків станом на 2017-й рік та 82 мільярди установок додатків станом на 2016-й рік[5].

Telegram Mini App – перш ніж дати визначення що таке Telegram Mini App, дамо визначення Telegram. Telegram – це месенджер, який має мобільний, десктопні застосунки та націлений на швидкість і захищеність. Telegram є досить простим у використанні, безкоштовним та як зазначають, автори – надшвидким. Також Telegram, має синхронізацію між девайсами та номерами телефонів, що робить його дуже зручним у використанні. Цільова аудиторія Telegram є люди, які бажають надійно обмінюватись повідомленнями без можливості дешифрувати повідомлення зловмисниками. Telegram є одним з найпопулярніших месенджерів на планеті, зайнявши перше місце за завантаженнями у січні 2021-го року.

Однією з неймовірних частин функціоналу Telegram є можливість створювати та користуватись Telegram Bot всередині застосунку. Telegram Bot – не що інше як маленький додаток, який запускається виключно в самому Telegram, і має гнучкий інтерфейс, що дозволяє виконувати будь-які задачі, починаючи від отримання прогнозу погоди на сьогодні до більш складних додатків, як автоматизовані модератори чатів. Але мінус Telegram Bot складається у тому що, інтерфейс для складних додатків, може бути надто складним для пересічного користувача, тому команда Telegram, розробила Telegram Mini App (ТМА). ТМА – дозволяє вставити майже будь-який вебсайт в додаток Telegram, що складає враження справжнього сайту, але всередині додатку Telegram.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Тому Telegram Mini App, чудово підходить для реалізації та апробації ПЗ системи інтелектуального аналізу даних для управління звичками людини. Потрібно лише розробити мобільну версію вебсайту та інтегрувати з Telegram[6].

Підсумовуючи, цей розділ вдалося дати визначення, що таке цільова аудиторія, зробити мінімальний аналіз цільової аудиторії для ПЗ інтелектуального аналізу даних для управління звичками людини, а також розібрати доцільні та схожі платформи, для розгортання схожих додатків.

КБПЗ_2026

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

Категорії ментального здоров'я та самовдосконалення стали невід'ємною частиною майже кожної людини на планеті, якщо подивитись на існуючі системи-аналоги, що надають подібні можливості та кількість їхніх рейтингів в App Store можна дійти до висновку, що ця сфера має значний потенціал. Productive – 3.3 тисяч рейтингів, Habit – 2.8 тисяч рейтингів, Break Bad Habit – 1.7 тисяч рейтингів, Way of Life – 306 рейтингів та Habitica – 71 рейтинг.

App Store – це платформа, яка надає можливість безпечно встановлювати додатки користувачам на iPhone, а розробникам цих додатків на цьому заробляти. Платформа була відкрита для користувачів відразу після виходу першого iPhone у 2008 році. Де-факто, Apple була першою компанією, запровадивши подібну платформу, інші стали копіювати цю ідею. Варто зазначити, що окрім розміщення додатків, розроблених іншими розробниками, Apple віддає лише 70% доходу розробникам, а 30% Apple забирає як їхню комісію за дану можливість. Раніше основною мовою написання додатків для iPhone та iPad була мова програмування Objective-C, пізніше Apple розробила свою мову програмування для написання цих додатків, яку згодом назвали Swift, що використовується донині і залишається основною мовою програмування для написання додатків під iOS, iPadOS та macOS.

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень

Перш за все, потрібно визначитись з тим, де потрібно шукати подібні застосунки. Вибір насправді очевидний – це одні з найпопулярніших платформ у світі, Google Play, а також App Store, що були розглянуті в розділі 1.2. ТМА все ще відносно молода платформа, на якій мало схожих додатків.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Habituca

Habituca – це безкоштовний додаток, який надає змогу будувати звички та покращувати своє життя, уявляючи, що гра це і є реальне життя, в якому потрібно прокачувати свого персонажа. Маючи нагороди та покарання всередині гри, Habituca створена для того, щоб мотивувати та надихати. Habituca може допомогти досягати поставлених цілей, ставати здоровішим, старанним в роботі та просто щасливішим[7].

Habituca має два мобільних додатки під операційні системи Android та iOS. Також вебсайт, який написаний мовою JavaScript використовуючи HTML та CSS. Habituca має відкритий вихідний код своїх мобільних додатків та вебсайту[8]. Окрім базових технологій що зазначені вище, доцільно розглянути й інші фреймворки, які розробники Habituca використали при створенні вебсайту. Але перш за все потрібно зрозуміти із чого складається вебсайт.

Вебсайт складається з клієнтської частини та серверної, тобто модель client-server model. Client-server model – це розподілена структура ПЗ, яке розділяє задачі або навантаження між сервером та клієнтом. Дуже часто клієнти та сервери комунікують через комп'ютерну мережу, яка знаходиться на іншому ПЗ, але і клієнт, і сервер повинні знаходитися на одному пристрої. Сервер запускає серверні програми, які діляться своїми ресурсами з клієнтами. Зазвичай, клієнт не ділиться ніякими ресурсами із сервером, але натомість він запитує контент від сервера, яким може з ним поділитись. Прикладами програмних засобів, що використовують модель клієнт-сервер є надсилання електронних листів або ж WWW.

Habituca складається з наступних технологій, що використовуються на сервері:

1. [Express.js](#)

Express.js – це вебфреймворк для розробки вебдодатків та API, які побудовані за допомогою Node.js. Express має відкритий вихідний код та розповсюджується під ліцензією типу MIT License[9]. MIT License – ліцензія для програмного забезпечення розроблена Масачусетським технологічним

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

інститутом, що дозволяє перевикористовувати код проєктів, що знаходяться під цією ліцензією в інших проєктах. Деякі з яскравих прикладів проєктів що використовують MIT License включають в себе: React, Node.js, Ruby on Rails, Lua та багато інших[10].

Application programming interface (API) – це набір певних правил, які визначають як різні програмні забезпечення можуть обмінюватись даними комунікувати між собою. Протокол визначає формат та структуру повідомлень. Найпоширенішим для передачі даних через протокол є формат JSON, але також існує підтримка XML[11].

JavaScript Object Notation (JSON) – є одним з найпопулярніших форматів використовуваних під час передачі даних через API. JSON легко читається не тільки машинами, але й людьми. Побудований на основі мови програмування JavaScript, є зрозумілим для програмістів також із C-подібних мов програмування. Можна привести такий приклад JSON: {"id": 1, "name": "Central National Technical University"}, де id та name є ключами об'єкту, а 1 та "Central National Technical University" є значеннями цих ключів[12].

Extensible Markup Language – є мовою розмітки досить схожою на мову розмітки HTML, але зі своєю принциповою відмінністю, яка полягає у тому що в XML майже всі теги користувач може створювати, якими вважає за потрібне, на відміну від HTML, де є набір визначених тегів[13].

2. [Node.js](#)

Node.js – є асинхронним середовищем виконання JavaScript кероване подіями. Node.js проєктувався з думкою про масштабованість, тому він ідеально підходить для високо навантажених серверних додатків[14].

3. MongoDB

MongoDB – це документно орієнтована операційна база даних, побудована як альтернатива реляційним базам даних, таким як MySQL, PostgreSQL та іншим. На відміну від реляційних баз даних, MongoDB дозволяє розробникам додатків перетворювати збережені JSON подібні об'єкти в об'єкти які

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

знаходяться в додатку, що робить її дуже привабливою та популярною серед розробників в усьому світі[15].

4. [gulp.js](#)

gulp.js – є проєкт з відкритим вихідним кодом, написаний на мові програмування JavaScript та побудований за принципом потоку вузлів. gulp.js створений для того, щоб автоматизувувати рутинні та довгі за часом задачі. Має вбудовану підтримку більшості сучасних інтегрованих середовищ розробки, а також має мінімальне API, яке легко освоїти та працювати з ним[16].

5. **Git**

Git – розподілена система контролю версій, яка має відкритий вихідний код та є безкоштовною. Git ідеально підходить від малих до великих проєктів, щоб пришвидшити розробку та зробити її ефективною. Git легко вивчити, а також він мало займає місця, що робить його дуже швидким та ефективним при розробці малих та великих проєктів. Git є де-факто найкращою системою контролю версій на даний момент[17].

6. **GitHub**

GitHub – сервіс що пропонує розробникам зберігати, управляти та поширювати код. GitHub використовує Git, для того, щоб забезпечити розподілену систему контролю версій, а також GitHub надає деякі з функцій як-от управління проєктами, завданнями, створення обговорень, неперервну інтеграцію та багато чого іншого, що робить його одним з найпопулярніших сервісів, для розміщення коду за допомогою розподіленої системи контролю версій як Git[18].

7. **ESLint**

ESLint – це інструмент, який надає статичний аналіз JavaScript коду та пропонує можливі вирішення проблем. ESLint є неймовірним інструментом, тому що дозволяє розробнику швидко виявляти та виправляти знайдені помилки. Також має чудову підтримку багатьох сучасних редакторів, що робить його дуже зручним у використанні. Врешті-решт, надає змогу для інтеграції в існуючий неперервний процес розгортання проєкту CI/CD[19].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

8. [Vue.js](#)

Vue.js – це прогресивний JavaScript фреймворк, який націлений на швидкодію, доступність та різнобічність, для створення користувацьких вебінтерфейсів. Справді, Vue.js має документацію світового рівня, що робить його дуже доступним. Екосистема, навколо якої побудований фреймворк, є різною, що робить його різнобічним.

Way of Life

Інтуїтивний додаток для ведення звичок мотивує побудувати хороші звички, ставати сильнішим, краще і до того ж дійсно працює. Так себе позиціонують творці додатку Way of Life[28].

Після проходження етапу онбордингу у додатку, можна спостерігати головний екран, зображений на рисунку 2.1.

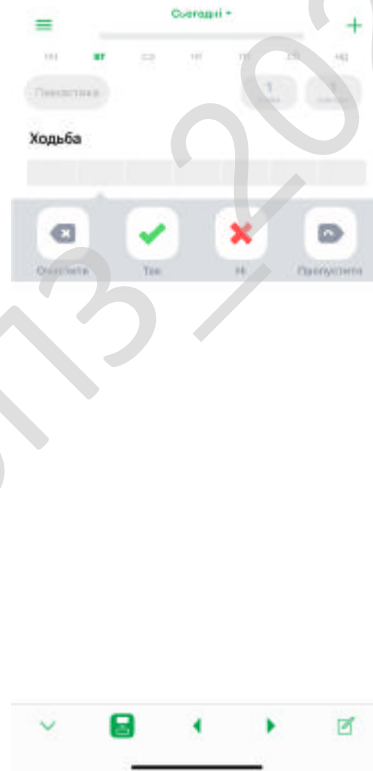


Рисунок 2.1 – Головний екран після проходження етапу онбордингу в додатку Way of Life

Відразу ж була обрана звичка “Ходьба”, тому що інакше, не створивши першу звичку, додаток не пропускав далі на головний екран. Що можна спостерігати з самого початку так – це поп-ап з діями, які можна сьогодні

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

застосувати до цієї звички, а саме: “очистити”, “так”, “ні” та “перенести”. Якщо натиснути “так”, сьогоднішній день зафарбується зеленим кольором. Досить інтуїтивний головний екран.

Перейдемо до наступних екранів, а саме: графіки, примітки, нагадування та налаштування. Розпочнемо з екрану з графіками, що зображений на рисунку 2.2.

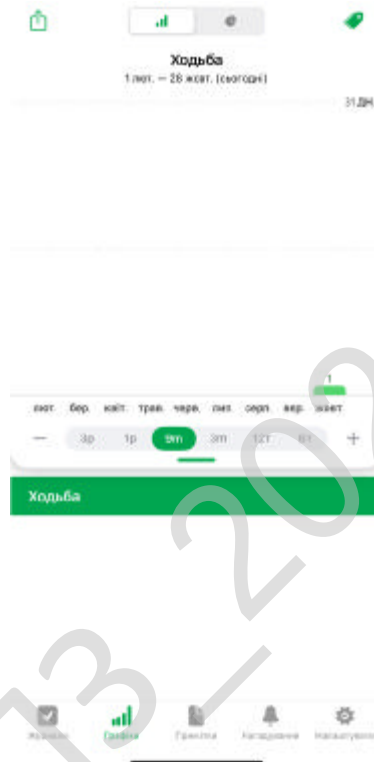


Рисунок 2.2 – Екран із графіками у застосунку Way of Life

Як можемо спостерігати на цьому екрані, відображено графіки або інакше кажучи, статистика. Є можливість переключити тип статистики, за замовчуванням це графік, на якому по осі абсцис розташовані місяць року, а по осі ординат кількість виконаних днів, і неважливо чи успішно виконаних тобто зелених або червоних. Інший тип статистики – кругова діаграма.

Наступний екран це “Примітки”, що зображений на рисунку 2.3.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

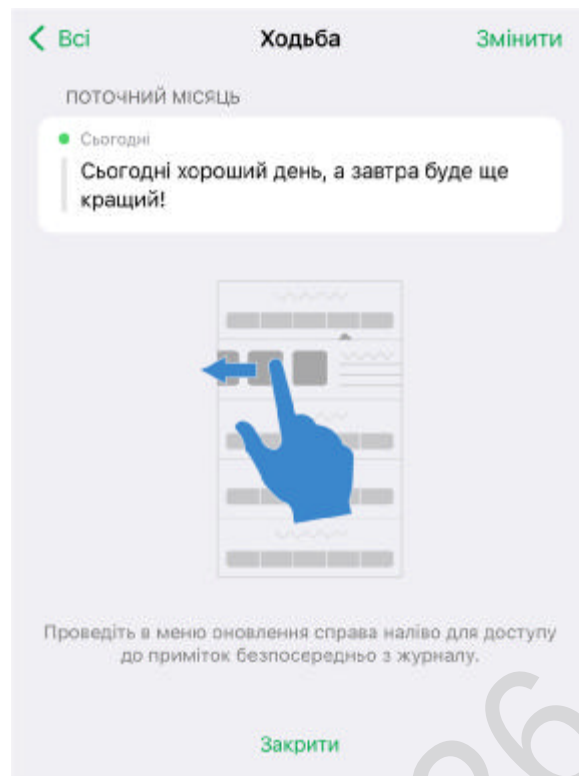


Рисунок 2.3 – Екран “Примітки” у застосунку Way of Life

Як можна зрозуміти з назви, екран, де можна створювати, переглядати, редагувати та видаляти примітки для вибраної звички. Виглядає досить просто, але водночас цікава ідея, додавати примітки до щоденних звичок.

Екран з нагадуваннями, зображений на рисунку 2.4, теж досить інтуїтивний і зрозумілий, нічого зайвого, лише нагадування.

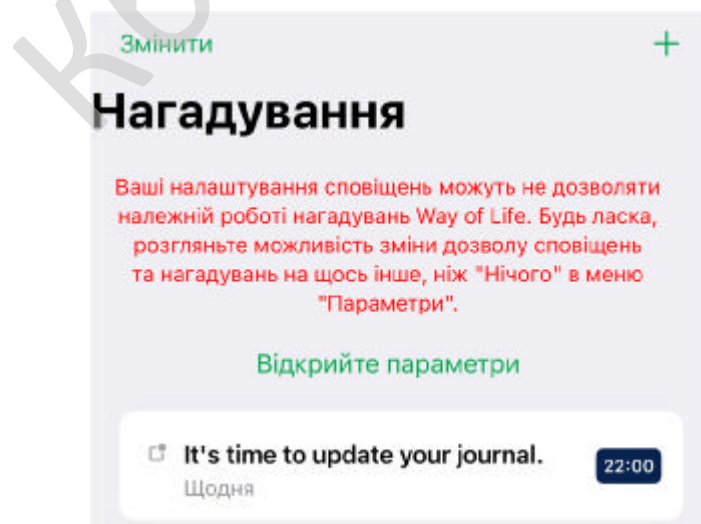


Рисунок 2.4 – Екран з нагадування застосунку Way of Life

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

Way of Life має багато різних налаштувань, від зміни іконки для додатка до автокоригування нотаток. Також є секції з новинами про додаток, секція з допомогою у використанні додатку. Є можливість допомогти розробникам з перекладом застосунку на різні мови та багато іншого.

Підсумовуючи цей додаток, можна лише сказати, що він є досить зрозумілим, простим, швидким та приємним у використанні. Але водночас, відсутність гейміфікації та соціалізації на простому рівні, робить його досить нудним. Хоча, додаток є дуже популярним в своєму сегменті, що говорить про правильність дій виконуваних з боку творців цього застосунку.

Bad Habit Break

При першому вході у застосунок, нам пропонують обрати версію: версія, яка не має реклами, але є платною або версію, яка включає в себе рекламу та є безкоштовною, скористаймося останньою.

Екран створення “залежності”, зображений на рисунку 2.5.

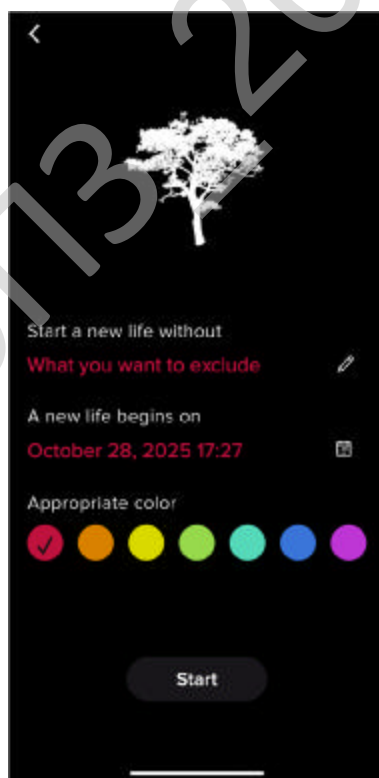


Рисунок 2.5 – Екран створення залежності у застосунку Bad Habit Break

Як можемо бачити, можна вибрати колір звички, дату початку та назву поганої звички. Досить прості три поля для заповнення, єдине що потрібно

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

відзначити це ліміт на кольори для звичок, тому що це незручно якщо користувач матиме понад 7 поганих звичок. Після створення “поганої звички” користувача перенаправляє на сторінку перегляду цієї звички, на якій можна побачити наступні параметри: назву, ціль, вимірювану в днях, історію, кількість спроб, рекорд (найбільшу кількість днів без цієї звички), таймер і кнопка для скидання прогресу, тобто якщо користувач не зміг втриматися перед “залежністю”, то все починається з самого початку. Також з цього екрану можна видалити звичку та редагувати її. Інтерфейс звички, зображений на рисунку 2.6.

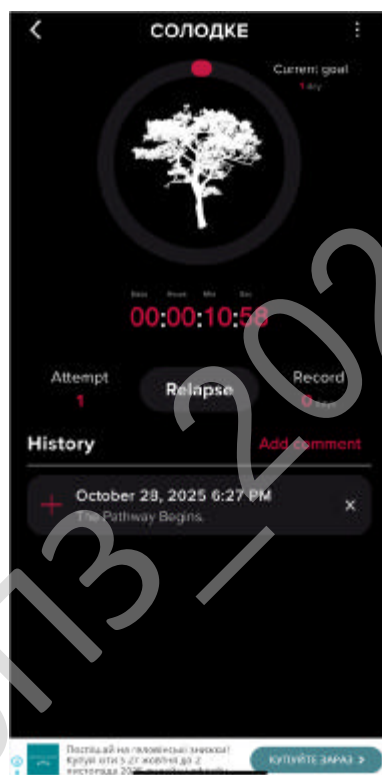


Рисунок 2.6 – Інтерфейс звички у застосунку Bad Habit Break

Також є ще екран з усіма звичками з пошуком, але його не варто брати до уваги, тому що функціонал екрану дуже малий. Лише варто додати, що є бічна панель з налаштуваннями застосунку, статтями про продуктивність та інші налаштування такі, як вибір теми та мови.

Підсумовуючи, огляд цього аналогу, можна зробити висновок, що додаток є дуже простим, окрім лише створення поганих звичок, більше функціоналу в ньому немає. З мінусів це те, що надмірна кількість реклами, яка показується на кожному кроці, що суттєво впливає на користувацький досвід[29].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Habit

Що ж після встановлення додатка, користувача зустрічають декілька слайдів про те навіщо цей застосунок, та серія з питань, на які потрібно відповісти перш ніж користуватись додатком. Список цих питань:

1. Чи часто ви прокрастинуете?
2. Коли ви дієте після встановлення цілі?
3. Якщо ви не дієте відразу ж, то чому?
4. Що вас мотивує будувати хороші звички?

Після проходження всіх питань, користувача зустрічає екран, де він може обрати метод check-in. Як можемо спостерігати, методи лише два: свайпом у правий бік та методом натискання на звичку. Що ж на даному етапі припустімо, що користувач обрав останній із запропонованих, тобто check-in буде відбуватися методом натискання на звичку. Екран з вибором методу відмічування звички зображений на рисунку 2.7.

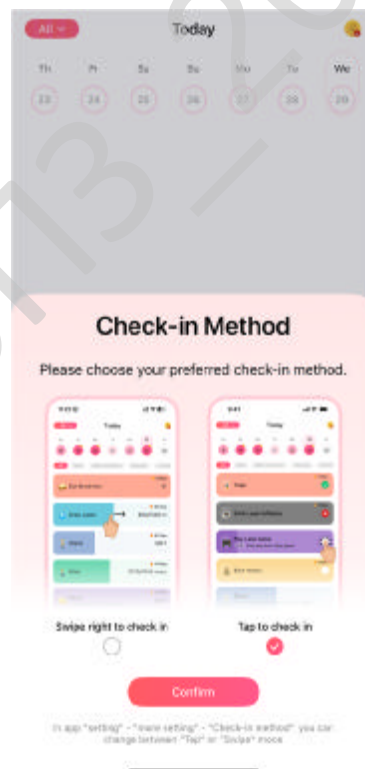


Рисунок 2.7 – Екран з вибором методу відмічування звички у застосунку Habit

Натиснемо на іконку “+”, щоб почати створювати звичку, також варто зазначити, що хоча меню і складається лише з іконок та без назв, іконка “+” є

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

досить інтуїтивно зрозумілою, що складно сказати за інші іконки. Спершу ми побачимо список усіх уже створених звичок, що поділені на категорії: популярні, здоров'я, спорт, спосіб життя, час та звички, яких потрібно позбутись. Екран створення звички зображений на рисунку 2.8.

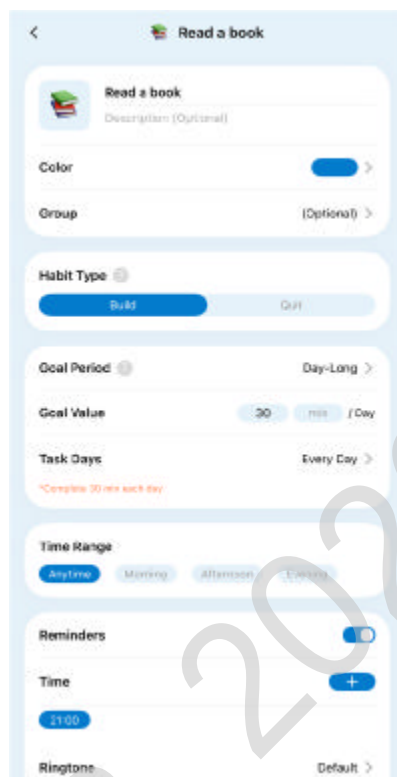


Рисунок 2.8 – Екран створення звички у застосунку Habit

Користувача зустрічає екран з багатьма параметрами, як для такої простої звички читати книгу, втім розберемо кожен з цих параметрів. Назва книги вже є – Read a book, далі йде опис, що є опціональним полем до заповнення, далі колір, група, тип звички, період цілі, яка ціль на день, часовий проміжок виконання звички. Далі йде секція з нагадуваннями, що дає змогу персоналізувати нагадування саме під користувача та його потреби, що є досить добре, також є можливість налаштувати рингтон. В кінці також є різні параметри, які дають змогу налаштувати тип статистики: кругова чи у вигляді графіка. Дата початку та кінця звички.

Підсумовуючи, можна сказати що екран зі створенням звички включає в себе надто багато параметрів, які можуть здаватись не потрібними для користувача на першому етапі знайомства із застосунком, але через те, що, це

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

звичка яка уже створена, тобто налаштована за користувача, лише потрібно натиснути “зберегти” і звичка створюється, що є дуже зручним.

Звичка читати книгу щодня була успішно створена, перейдемо в неї для того, щоб переглянути наявний функціонал. Що ж так як звичка читати книгу вимірюється в хвилинах, застосунок пропонує нам відразу поставити таймер на 30 хвилин, що збігається саме з кількістю хвилин на день. Додаток надає змогу: змінити стиль таймеру, поставити свою мелодію та поставити мелодію на завершення таймеру, що робить його досить кастомізованим таймером. “Мето” в перекладі з англійської це пам’ятка, тобто є можливість до звички додати, просто текст або емоджі, що робить цю ідею досить цікавою[31]. Якщо подумати, пам’ятка що ти читав якогось дня, може бути досить корисною.

Насамкінець користувач може переглянути статистику по цій звичці. Статистика є досить обширною, можна переглядати навіть кількість хвилин читання за поточний місяць, переглядати збережені пам’ятки та інше, що робить цей функціонал дуже потрібним, для користувачів, які дуже захоплені ідеєю переглядати свою статистику щоденно.

Пристімо, ввели значення, що 90 хвилин сьогодні читали книгу, тоді спробуємо занести це значення. Після вибору кількості хвилин та натискання на кнопку “зберегти”, ми можемо спостерігати гарний ефект з безліччю конфеті. Екран із виконаною звичкою на сьогодні зображений на рисунку 2.9.

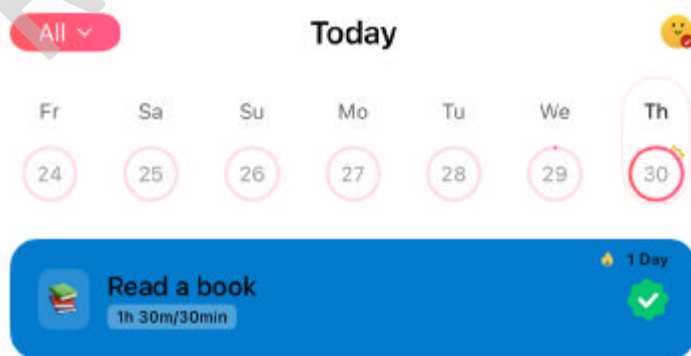


Рисунок 2.9 – Екран з виконаною звичкою у застосунку Habit

Звичка змінила свій колір зафарбування, тобто зафарбувала в синій колір. Також ми бачимо що у користувача з’явився напис “1 Day” та емодзі вогню біля

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

нього, що означає, що сьогодні перший день так званого стріку. Streak в перекладі з англійської означає “смужка”, тобто користувач почав будувати смугу виконаних днів своєї звички, що виглядає досить захопливо[32]. Також не можна не відзначити зелену іконку що з’явилась після того як користувач вніс дані. Найімовірніше, це означає що день пройшов успішно, і ця звичка була виконана.

На наступному екрані ймовірніше за все, це екран зі статистикою звичок користувача, поглянемо на неї ближче. Екран, що зображує статистику обраної звички зображений на рисунку 2.10.

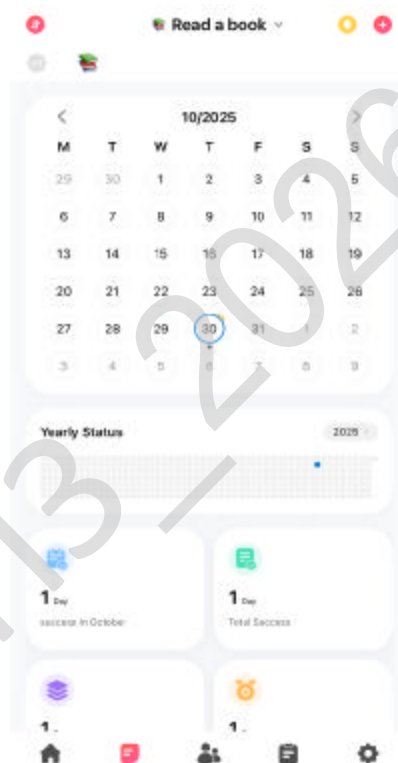


Рисунок 2.10 – Статистика обраної звички у застосунку Habit

На цьому екрані можемо спостерігати календар для поточного місяця, що є досить зручним, також бачимо річний статус звички у вигляді 365 клітинок, що зафарбовуються у колір звички, що був обраний при створенні. А також є чотири блоки, які теж відображають статистику. Перший блок – це скільки було успішних днів в поточному місяці. Відразу другий блок, що йде за першим по горизонталі це – скільки днів було успішних загалом, за весь період. Третій блок, який зафарбований фіолетовим кольором, означає яка поточна смуга успішних

днів. Четвертий блок, відображає найкращу смугу успішних днів, вимірювану саме в днях. П'ятий і шостий блоки відображають обсяг хвилин за поточний місяць та весь час, але не просто у форматі хвилин, а у форматі годин та імовірно далі якщо годин буде забагато – у форматі днів. Сьомий блок відображає загальну кількість обсягу, тобто хвилин поділену на кількість загальних днів. У випадку користувача, який користується застосунком всього лише два дні – це 45 хвилин та три секунди. Останній блок, це відсоткова кількість виконаних днів до загальних. У випадку користувача – це 50,17%. Якщо прогорнути вниз, можна побачити ще два графіка, розділ з пам'ятками та три основних кнопки: редагувати, архівувати та видалити.

Якщо з екраном статистики звичкою все зрозуміло, то наступний екран в меню це групи. Під групами напевно маються на увазі групи друзів, які можуть разом будувати звички. Система запрошень побудована навколо iCloud Sync, що робить це складним для людей, що не мають бажання користуватись iCloud Sync. На думку автора, це не є добре, тому що запросити друга чи подругу повинно бути легко і реалізовуватися всього лише кількома кліками, без третіх додатків, тим не менш цей функціонал в додатку присутній.

Наступний екран зі звітами, але він заблокований для звичайних користувачів, без преміум-підписки зображений на рисунку 2.11.

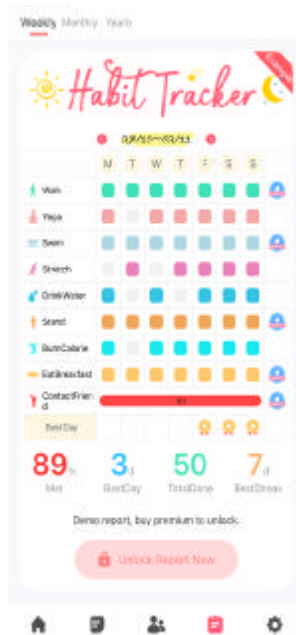


Рисунок 2.11 – Заблокований екран зі звітами у застосунку Habit

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Останнім є екран налаштувань, в якому можна змінити ім'я та фото. Також можна прочитати надихаючу цитату дня, переглянути рівні стресу, та настрої протягом тижня. Додатково можна налаштувати тему, тему віджета, функцію сторінки, динамічний острів та iCloud Sync. Додатково присутня секція з порадами по використанні застосунку, відповідями на часто запитувані питання, можливістю поділитися застосунком у соціальних мережах та інше.

Підсумовуючи Habit, можна сказати що додаток має приємний інтерфейс, достатню швидкість відповіді на кліки та перегляд сторінок та обширну статистику. Хоча також є і мінуси, які полягають в багатьох параметрах для створення звички, хоча і компенсуються багатьма опціональними параметрами, які і дають змогу швидко створити звичку, інтеграція користувача із стороннім сервісом iCloud Sync, робить процес запрошення друга більш складним, ніж він міг би бути. Недоступність звітів, на думку автора, не є мінусом, тому що кожен додаток додає обмеження для того, щоб хоча б якось монетизувати свою працю. Загалом додаток досить цікавий, і вартий того щоб його спробувати.

HabitMate

При відкриванні та наданні деяких дозволів HabitMate, користувача зустрічає екран в якому йдеться про те, що HabitMate це список завдань, який допоможе сформувати хороші звички. Основною ціллю є досягти 100% виконання звичок. Далі пропонують придбати PRO-версію, але продовжимо поки без неї. Отож, після проходження декількох вступних екранів, користувач бачить перед собою мінімалістичний дизайн застосунку, зображений на рисунку 2.12[33].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

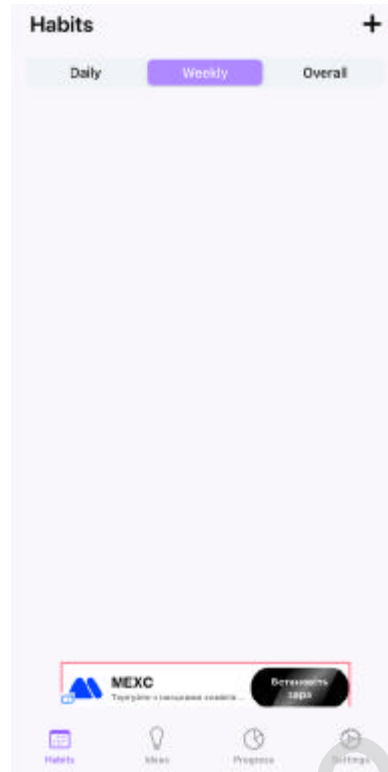


Рисунок 2.12 – Мінімалістичний дизайн застосунку HabitMate

Знизу можна спостерігати навігацію по наступним сторінкам: звички, ідеї, прогрес та налаштування, досить мінімалістично і зрозуміло. У верхній частині екрану можна бачити “Habits” та іконку “+”, трохи нижче відразу розташовані три вкладки: щоденні, щотижневі та загальні.

Productive

Після першого ж відкриття нас зустрічає серія дозволів та питань, через які необхідно пройти перш ніж можна буде користуватись додатком. Перше питання сформульовано наступним чином: “Як ви оцінюєте рівень своєї енергії протягом дня?”, відразу наступне питання: “Наскільки ви задоволені способом свого життя?”, наступне: “Чи часто ви прокрастинуете?”, ще одне: “Чи важко вам сфокусуватись?”, і останнє: “Що ви очікуєте досягти за допомогою Productive?”. Отож після відповіді на питання, користувачу пропонують обрати першу звичку з наявних, зображений на рисунку 2.13[34].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

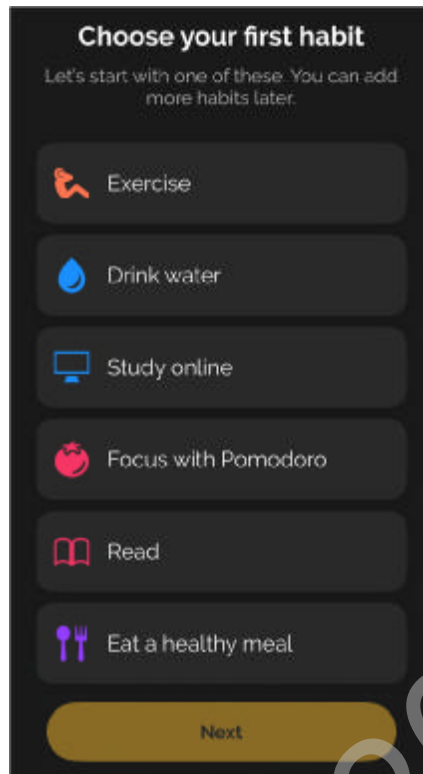


Рисунок 2.13 – Вибір першої звички у застосунку Productive

Припустімо, користувач обрав таку ж звичку як читання, що і в минулому застосунку. Після натискання на кнопку “Далі”, користувача навчають як відмічати новостворену звичку, а саме якщо провести вправо, то звичка відзначається як успішно виконана, і навпаки якщо вліво потягнути, повернемось до початкового стану, тобто незакінченої звички.

Після вибору звички “читати щодня” користувач потрапляє на екран налаштування, який має деяку схожість із застосунком “Habit”. З екрану, що зображений на рисунку 2.14, можна побачити налаштування схожі до “Habit” також на моменті створення звички. Розберемо кожен з цих параметрів окремо. Назва звички, Productive поставило “Read”, що, варто зазначити, не є зовсім коректним з погляду звички, тому що слово read в перекладі з англійської читати, тобто є дієсловом, а не іменником. Натомість, зрозуміліше було б зробити, як у застосунку “Habit” – reading, тобто в значення іменника читання. А втім, продовжимо огляд, наступним полем іде поле з описом звички, за замовчуванням є порожнім та опціональним. Можливість вибору іконки, кольору та звуку звички також присутні.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Наступним блоком налаштувань є повторюваність звички, за замовчуванням повторюється кожен день та дата кінцевого дня звички, за замовчуванням відсутня. Одиниці вимірювання за замовчуванням відрізняються від тих що в “Habit”, тобто звичка буде вимірюватись у сторінках на день, що є досить теж хорошим вибором. Останнім блоком в цих налаштуваннях при створенні звички є блок з нагадуваннями. За замовчуванням обраний вечір, тобто час коли буде надсилатись push-повідомлення. Але є вибір налаштувати це повністю під потреби користувача, а саме обрати час, локацію, і навіть після якої дії надсилати нотифікацію, що безсумнівно, є чудовою опцією.

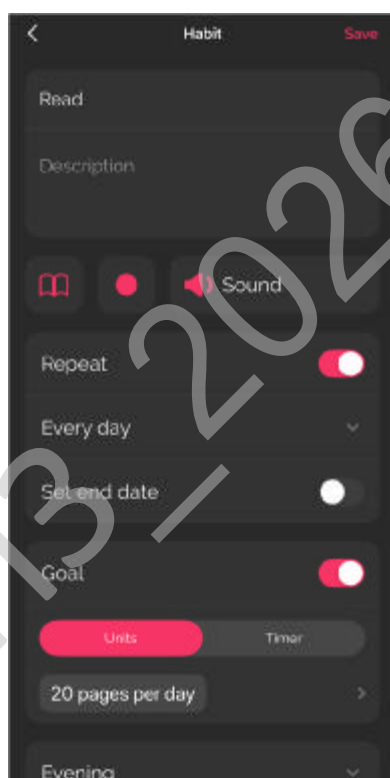


Рисунок 2.14 – Налаштування звички у застосунку Productive

Після створення звички, користувач переходить на загальну сторінку “Today”, де відображені всі звички на сьогодні, що зображений на рисунку 2.15. З цього екрану можна побачити, що на сьогодні користувач має лише читання і все. Для того, щоб виконати звичку потрібно, прогорнути натиснувши на звичку вліворуч, тоді з’явиться три опції: занести одну прочитану сторінку, занести десять прочитаних сторінок або виконати загалом, що у випадку користувача є двадцять сторінок відразу. Скористаймося останньою опцією, і виконаємо

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

відразу цю звичку, і побачимо що звичка зникла з екрану, тому що система порахувала її вже виконаною, що є безсумнівно правильним кроком.

Також звички можна фільтрувати за частинами доби: ранок, обід та вечір, що робить це зручним якщо користувач має багато звичок розділених за частиною дня. У верхній частині екрану можна спостерігати міні-календар, як і за аналогією з “Habit”. Календар інформує, про дні тижня та має на меті, підказати що сьогодні за день.

Два елементи що також є на цьому екрані, але не є частиною меню це налаштування та список звичок. Список звичок, це просте модальне вікно, яке показує створені звички. Налаштування застосунку показують не тільки всі можливі налаштування застосунку починаючи від основних та закінчуючи нотифікаціями, а також є можливість, написати відгук на платформі App Store, написати відгук розробникам додатку. Прочитати про політику конфіденційності, спробувати преміум безкоштовно, увійти в акаунт за допомогою вебсайту. Переглянути можливості використання додатку разом з Siri. Поділитись застосунком з іншими та інше.

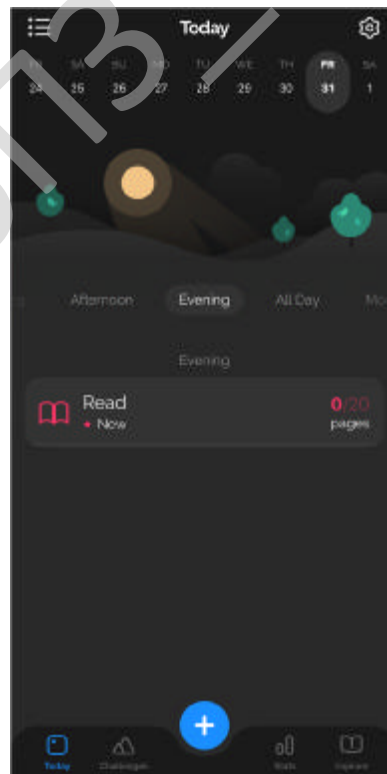


Рисунок 2.15 – Загальна сторінка “Today” застосунку Productive

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо перейти на сторінку “Challenges”, що зображена на рисунку 2.16 з меню, то можна спостерігати наступний інтерфейс та функціонал. Можна побачити декілька викликів, а саме: “Find Happiness Challenge”, “Flow Challenge”, “Intermittent Fasting Challenge”, “Italian Excursion Challenge”, “Fresh Morning Challenge” та “Bedtime Routine Challenge”. “Find Happiness Challenge” – як можна зрозуміти з назви цього виклику, це пошук щастя. До цього виклику приєдналися 10 людей, і він все відкритий протягом більше ніж шести годин.

Після натискання на кнопку “Приєднатись зараз”, користувач спостерігає короткий опис виклику і основну кнопку для доєднання до виклику, але вона не є зовсім активною, поки користувач не є преміумом, що безумовно є мінусом, тому що, як було зазначено в першому розділі соціальний чинник застосунку, може зіграти важливу роль, і це повинно бути доступним для всіх охочих на думку автора. Втім, всі інші виклики також є недоступними і мають опис теж.

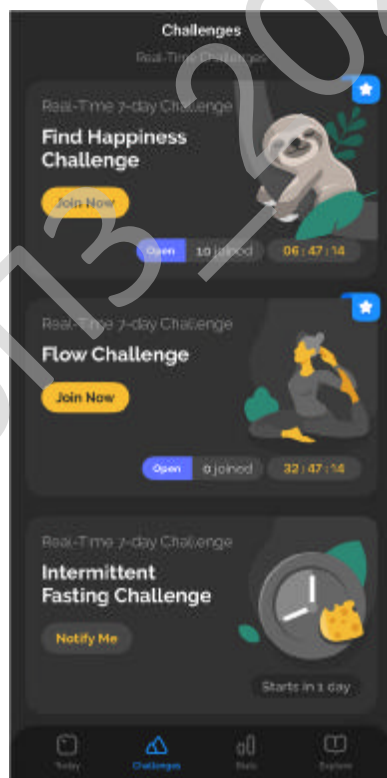


Рисунок 2.16 – Сторінка “Challenges” у застосунку Productive

Наступною є сторінка “Stats”, яка зображена на рисунку 2.17 виглядає наступним чином. Але користувач практично не має змоги переглянути цю

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

сторінку, через те, що вона теж є доступною тільки для преміум користувачів. Отже перейдемо до наступної сторінки.

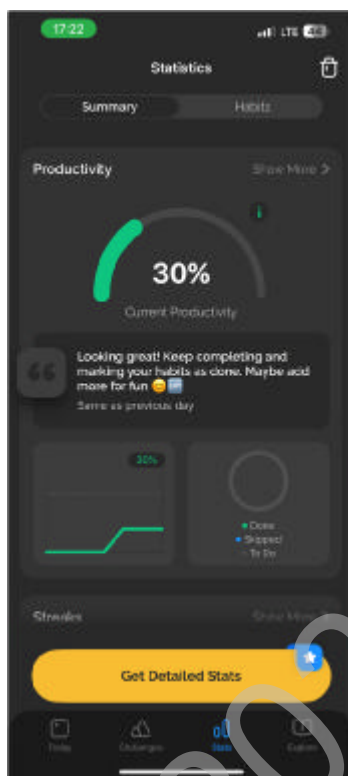


Рисунок 2.17 – Сторінка “Stats” застосунку Productive

Останньою у застосунку є екран “Explore”, який є досить обширний і має наступний вигляд. На екрані є дві вкладки це статті та програми. Сторінка “Explore” зображена на рисунку 2.18.

Почнемо з огляду вкладки статті. Перш за все є імплементований пошук по назві статті, що є зручним, наступне це класифікація статей по різних категоріям, а саме: всі, вода, вправи, медитація та корисне харчування. Загалом сторінка має багато цікавих та пізнавальних статей з практичними порадами, що можуть допомогти користувачеві в будіванні корисних звичок.

Друга вкладка – це програми, де перелічені чотири програми. Наприклад, програма, “Як бути продуктивним” або “Як вивчити нову навичку”. Програми мають певний термін їх виконання, що вимірюється у днях. На сторінці окремої програми є короткий опис, про що сама програма, також є основні переваги цієї програми, і також в кінці є список схожих програм, які також можуть сподобатись користувачеві. Приєднатись до програми, можна без преміума.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

Один день з програми виглядає наступним чином. Є певне завдання на день, яке потрібно виконати, є опис задачі та можливість позначити його як виконане, досить простий функціонал. Є деяка стаття, що може допомогти у виконанні раніше згаданого завдання. Це і є один день з програми. Справді, програма є досить короткою, незрозуміло чому було реалізовано саме таким чином.

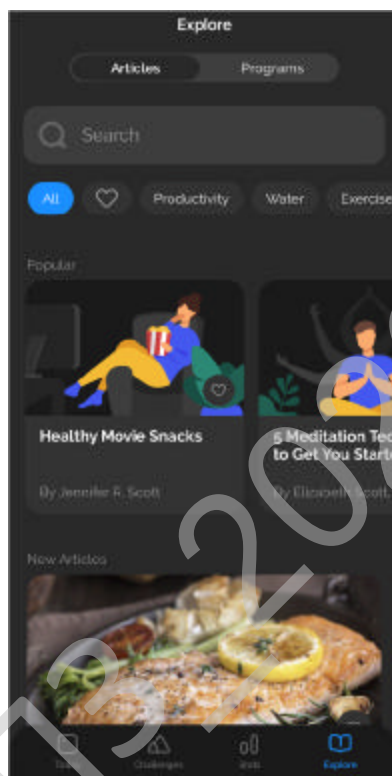


Рисунок 2.18 – Сторінка “Explore” застосунку Productive

Підсумовуючи, можна сказати, що додаток має чудовий UI/UX, який вирізняє його поміж інших, які є на ринку, також Productive має чудову систему звичок, схожу з Habit. Є виклики та статистика, але розроблені лише для преміум користувачів, що не дало можливість використати додаток на повну. Насправді, додаток має небагато функціоналу, ще й доступного для звичайного користувача, що робить його більш вразливим перед конкурентами, які мають те ж саме, функціонал, що і Productive, але безкоштовний, тому можна сказати, що додаток не розкрив принаймні тих переваг, які є доступними в багатьох інших додатках, що не дало змогу звичайному користувачеві насолодитися додатком сповна.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Дослідження доцільності використання подібних технологій для систем інтелектуального аналізу даних

Хоча використані технології у проєкті Habitica та подібних відкривають нові можливості перед користувачами, потрібно також зважати на доцільність та темп розробки в конкретному випадку, а саме у випадку із системою інтелектуального аналізу даних. З одного боку, технології використані є цілком виправданими для таких проєктів як Habitica, але це всього лише високорівневий огляд аналогічного додатка, тому перш ніж будувати систему інтелектуального аналізу даних, потрібно дати відповіді на наступні питання:

1. Яку платформу обрати? Вибір платформи – один з найважливіших, зазвичай це Web, мобільні платформи або десктопні платформи.
2. Які мови програмування або технології можуть підійти для такого програмного забезпечення? Мови програмування або ж технології, які підходять для такої платформи та типу завдань, є важливим питанням.
3. Який досвід має команда з переліченими мовами програмування або технологіями? Врешті-решт, досвід команди з цими технологіями відіграватиме не менш важливу роль, ніж самі технології.
4. Яка підтримка спільноти для обраних технологій? Обрати мертву технологію або мову програмування, яка вже ніким не використовується та у якої немає спільноти, взагалі немає доцільності, тому потрібно також важливо поставитися до цього питання.
5. Де буде розгортатися програмне забезпечення? Введення програмного забезпечення у дію, означає розгортання його на хостингу, що надає можливості використання VPS, тому потрібно дуже обережно вибрати хостинг або сервіс, а також звернути увагу на цінову політику.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення системи інтелектуального аналізу даних для управління звичками людини було обрано наступні технології та мови програмування. Технології, що були обрані, відповідають критеріям, таким як: надійність, підтримка зі сторони спільноти, зручність в розробці та практичність, перевірена багатьма проєктами. Щодо мов програмування, то основним критерієм вибору мови програмування є досвід роботи команди з нею. Перейдемо до списку мов програмування та технологій:

1. PHP (Laravel)

Мова програмування PHP пройшла досить тернистий шлях, починаючись по суті як шаблонізатор, ця мова дійшла до рівня повноцінної мови програмування що є, мабуть, однією з найпопулярніших мов програмування у світі, через її легкий поріг входу, легкість написання коду та C-подібний синтаксис. У випадку клієнт-серверної моделі, PHP чудово підходить для цього завдання, допоки не з'явиться серйозне навантаження. Варто зазначити, що хоча PHP вважають досить повільною мовою програмування, на досвіді автора, який працював з PHP більше ніж 4 роки, PHP є чудовим вибором для більшості стартапів через легкість написання коду та спільноти, яка побудована навколо цієї мови.

І навіть якщо навантаження серверів з PHP виросте кратно, завжди можна підключити модуль Swoole, що кратно пришвидшить обробку HTTP-запитів[20]. Laravel – це вебфреймворк, написаний для мови програмування PHP, що робить розробку вебдодатків на PHP неймовірно простою і веселою для розробника. Багата екосистема Laravel надає багато готових рішень та пакетів, що дозволяють рухатися швидше, ніж на інших мовах програмування таких як Java, C# і навіть Node.js. Таким чином, Laravel робить розробника надзвичайно продуктивним[21].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

2. JavaScript (React)

Сьогодні, майже неможливо уявити вебсайт, який би не використовував JavaScript як мову програмування для інтерактивної частини сайту. Будучи розробленою за 10 днів, для того, щоб замінити Java та додати інтерактивність для ранніх вебсайтів того часу, наразі JavaScript одна з найпопулярніших мов світу, а також домінує у світі веброботки[22]. Багато хто вважає, що React є JavaScript фреймворком, але насправді це лише бібліотека, яка надає змогу написати нативні користувацькі вебінтерфейси. React початково розроблений Facebook (Meta) в 2011 році, а потім ставши опенсорс проєктом є одним з найпопулярніших проєктів на GitHub. React має безліч корисних бібліотек, що робить життя розробника простіше, а код більш зрозумілим[23].

3. PostgreSQL

PostgreSQL – потужна, з відкритим вихідним кодом, об'єктно-реляційна база даних з довгою історією, яка зарекомендувала себе як надійну та продуктивну базу даних. PHP чудово інтегрується з PostgreSQL через модуль pdo_pgsql[24].

4. AWS

AWS (Amazon Web Services) – це одна з найбільших світових цифрових хмар, що дозволяє своїм клієнтам будувати усе, що вони тільки можуть собі уявити. AWS надає один з найбільших варіантів вибору хмарних можливостей та експертизи для найбільш сучасної глобальної інфраструктури у світі, з однією з найкращих у своїй індустрії безпекою, надійністю та швидкодією. Справді, AWS має багато переваг, починаючи від спектру послуг, які вони надають для побудови вебдодатків закінчуючи складними сервісами, що ідеально підходять для високонавантажених сервісів[25].

5. Git / GitHub

Git у поєднанні з GitHub є чудовою комбінацією, не тільки для того, щоб зберігати та керувати кодовою базою, але й для того, щоб потім відкрити вихідних код, що дозволить іншим приєднатись до проєкту також. Ще однією перевагою GitHub є GitHub Actions, яке надає змогу будувати ефективні CI/CD

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

конвеєри та доставляти код швидше й безпечніше, а також має наступні переваги:

– Налаштування досить просте. Справді, через те, що GitHub Actions побудований розробниками для розробників, у більшості випадків налаштування та підтримку CI/CD-пайплайн у GitHub Actions є досить простим. Не потрібно налаштовувати вебхуки, купувати серверне обладнання та перейматись за безпеку. Усе, що потрібно це лише помістити файл у корінь проєкту.

– Відповідати на будь-який вебхук у GitHub. Через те, що, GitHub Actions повністю інтегрований з GitHub, можна налаштувати будь-який хук, щоб привести в дію повний цикл CI/CD-пайплайн. Це включає в себе: пул-реквести, помилки проєктів, коментарі до пул-реквестів, а також включає вебхуки з додатків, які інтегровані з репозиторієм.

– Перевикористовувані робочі процеси створені спільнотою. GitHub має багато готових воркфлоу, створені великою спільнотою, що дає змогу швидко підібрати потрібний та зекономити час.

– Підтримка для будь-якої платформи, мов програмування та хмари. GitHub Actions є агностичним, якщо це стосується платформи, мови програмування або хмари, тобто це означає, що можна використовувати будь-яку хмару, мову програмування або платформу.

2.3 Розгорнута постановка завдання

Отримавши всі вхідні дані та конкретні деталі реалізації цього програмного забезпечення, можна перейти до постановки завдання. Розробка ПЗ інтелектуального аналізу у системах управління звичками людини передбачає створення програмного забезпечення на мовах програмування PHP, JS та їхніми відповідними фреймворками: Laravel та React. Важливим моментом при створенні такого типу ПЗ є простота, доступність, інтелектуальний аналіз даних та швидкість відповіді на запит.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнтською частиною ПЗ виступає Telegram із ТМА, що створить відчуття нібито це нативний додаток для мобільного пристрою. Саме такий підхід є по-справжньому простим та доступним у використанні. Користувачеві додатку не потрібно нічого встановлювати додатково, лише потрібно мати встановлений Telegram.

Щодо зберігання даних користувачів, то тут пропонується використання касту типу “encrypted” для певних таблиць, як от таблиця звичок, що зберігатиме назву та опис цієї звички.

На завершальному етапі проводиться тестування та розгортання проєкту на AWS-сервісі. Варто також зазначити, що тестування буде проводитись не лише на завершальному етапі, але й після кожного новоствореного функціоналу, за допомогою мануального тестування або автоматичного за допомогою написання тест-кейсів для спеціальної бібліотеки. Найкращим методом розробки програмного забезпечення є TDD. TDD – це така методологія розробки, при якій для кожного функціоналу додатку, який планується створити, пишеться спочатку тест-кейс, а потім і реалізація для проходження цього тест-кейсу.

Підсумовуючи, у розділі було проведено аналіз існуючих програмних рішень наступних аналогічних систем: Habitica, Way of Life, Break Bad Habit, Habit, Productive та HabitMate. Одну з аналогічних систем, а саме Habitica вдалося ретельно дослідити технічні можливості системи, а саме технології, які були покладені в основу даного програмного продукту. Також було досліджено та обґрунтовано вибір майбутніх засобів та мов програмування для розробки ПЗ інтелектуального аналізу даних для управління звичками людини. Насамкінець, була описана розгорнута постановка завдання, де прописані всі вимоги до реалізації проєкту.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ

При проєктуванні такого типу програмного забезпечення як системи інтелектуального аналізу даних, основним завданням є проєктування цієї системи, яка полягає в низці складних рішень, які потрібно прийняти щодо певних технологій.

В основі інфраструктури даного вебмодулю було обрано Linux разом з дистрибутивом Ubuntu як операційну систему, оскільки серверна частина цієї системи є критично важливою, і повинна забезпечувати надійність, стабільність, швидкість та забезпечувати приємний досвід розробника при роботі з нею. Ubuntu є не лише популярним дистрибутивом, а світовим стандартом, який використовується для більшості вебсерверів по всьому світу.

Це рішення дозволяє отримати досить стабільне середовище для роботи nginx, PHP-FPM, Redis, PostgreSQL та інших компонентів, таких як додаткових бібліотек, які цей дистрибутив має в достатній кількості.

Взагалі, Laravel, тобто вебфреймворк який реалізовуватиме API нашого вебмодулю, написаний серверною мовою програмування не може і не повинен напряду взаємодіяти із зовнішнім трафіком, оскільки це лише створює зайве навантаження на шар, в якому знаходиться вебмодуль, тому на рівні вебшару, було прийнято рішення обрати nginx, як reverse-проху для розділення відповідальностей між транспортним шаром та шаром вебмодулю. Задача reverse-проху, яким виступає вебсервер nginx, досить проста це взаємодіяти із зовнішнім трафіком, який приходить через мережу Internet, а саме через порти 80 та 443, і передати цей запит далі на вхід PHP-FPM процесу, тобто нашому вебмодулю - Laravel додатку. Крім того, nginx має додатковий шар захисту між клієнтом і Laravel, що підвищує контроль над мережею та спрощує майбутнє масштабування.

Як основний вебфреймворк для серверної частини вебмодулю було обрано Laravel. У системах інтелектуального аналізу даних штучний інтелект

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

є лише частиною проєкту, основна складність зазвичай знаходиться саме в бізнес-логіці додатку: авторизація, аутентифікація, ролі доступу, логування, інтегрування зі сторонніми бібліотеками, і в останньому дуже велику роль грає популярність вебфреймворку серед розробників.

Laravel дає змогу дуже швидко будувати подібні системи, завдяки вже готовій екосистемі, яка будувалась роками навколо цього вебфреймворку. Також досить важливим критерієм вибору цього вебфреймворку стала система безпеки, що є надважливим аспектом при проєктуванні API для клієнтів.

Для основного сховища даних була обрана зріла база даних PostgreSQL, оскільки система інтелектуального аналізу даних, потребує не просто бази даних, а потужної аналітичної платформи з високою консистентністю, складними SQL-операціями та підтримкою навіпструктурованих даних. PostgreSQL забезпечує повноцінну ACID модель, що важливо для роботи із користувацькими даними, історією інтелектуальних аналізів, логами, тощо. Але ключовою причиною вибору стала його здатність ефективно працювати зі складною аналітикою. PostgreSQL має потужну підтримку функцій агрегацій, віконних функцій, потужної системи індексування, бінарний тип даних JSON, а також багато іншого.

Використання одного із сервісів компанії OpenAI – Responses API є досить стратегічним рішенням з точки зору оптимізації ресурсів, тому що побудова власної ML-інфраструктури потребувала б окремої команди інженерів із машинного навчання, неймовірно дорогої GPU-інфраструктури та постійної підтримки створеної моделі машинного навчання. Натомість, OpenAI API дозволяє швидко інтегрувати функціонал інтелектуального аналізу практично відразу, концентруючись не на побудові ML-інфраструктури, а на створенні цінності продукту. Це рішення також значно спрощує масштабування, оскільки вся складність, яка пов'язана з машинним навчанням та моделями генеративного штучного інтелекту повністю делегується зовнішньому провайдеру.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

Програмне забезпечення системи інтелектуального аналізу даних для управління звичками людини, реалізовуватиме клієнт-серверну модель, таким чином, створюючи підґрунтя для масштабування проєкту в подальшому. У ролі клієнту, може виступати як вебсайт створений за допомогою бібліотеки React для створення користувацьких вебінтерфейсів та комунікації із сервером через захищений протокол HTTPS, так і додаток для відправлення запитів через протокол HTTP - Postman. Вебмодуль уособлюватиме швидкість та надійність завдяки вебсерверу nginx та вебфреймворку Laravel, написаному мовою програмування PHP, стабільної версії 8.4, що має багато сучасного функціоналу як от асиметрична видимість властивостей або хуки для властивостей класу.

Варто зазначити, що мета покладеного в основу проєкту вебмодулю, полягає в наданні, саме так званого API, тобто технологія якою буде реалізовуватись клієнт не є важливою, єдине вона повинна підтримувати протокол передачі даних HTTP та формат передачі даних JSON.

Таким чином, можна з впевненістю сказати, що вебмодуль є агностичним до вибору технології, що використовуватиметься в клієнтній частині ПЗ. Це відкриває неймовірний простір можливостей для розробників, що використовуватимуть цей вебмодуль в своїх проєктах.

Задля реалізації належної швидкості вебмодулю, було покладено в його основу швидкий вебсервер nginx, який написаний на швидкому та оптимізованому C++, в зв'язці з Laravel є потужним рішенням для реалізації вебдодатків, що надають API.

Насамкінець, даний стек був обраний не тільки з перспективи масштабування, простоти роботи з ними, чи зрілості цих технологій, а також із простоти цих технологій у використанні, досвід команди, також є вирішальною частиною вибору деяких технологій, що значно спрощує розробку в подальшому, і комунікацію всередині команди, навіть, якщо команда складається із одного розробника.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

3.1 Опис функціонування системи

Перш за все, кінцеві користувачі можуть взаємодіяти через вебсайт, який уособлює клієнт у клієнт-серверній моделі або через Postman. Варто зазначити, що технічно сайт не є прямим клієнтом, а лише зручною оболонкою для пересічного користувача. Насправді, клієнтом можуть бути звичайні HTTP-запити через cURL до API вебмодуля. Вебсайт може бути написаний за допомогою бібліотеки для створення користувацьких інтерфейсів – React або іншим сучасним фреймворком, що використовується на фронтенді – Angular, Svelte, Vue.js та інші.

Маючи необхідний клієнт, наприклад, cURL або додаток Postman можна робити запити до вебмодуля. По-перше, необхідно авторизуватись та отримати access token, для того щоб в подальшому мати змогу взаємодіяти із системою, звертатися до інших ресурсів, таких як інформація про користувача або ж список звичок. По-друге, після того як JWT-токен був успішно отриманий, користувач має змогу створювати, редагувати, переглядати та видаляти звички, створювати нотатки до звички та згодом ініціювати процес інтелектуального аналізу даних.

Json Web Tokens (JWT) – це відкритий стандарт для створення токенів доступу на основі відкритого формату JSON. Стандарт JWT є перевірений часом і зарекомендував себе як надійний стандарт, що використовуються в сучасних вебдодатках, що реалізують API. Алгоритм шифрування, що покладений в основу може бути HMAC та SHA256 або RSA разом з SHA256, що робить його досить безпечним для використання в реальних додатках, через алгоритми шифрування, що зарекомендували себе як перевірені часом.

Таким чином, маючи принаймні встановлену бібліотеку cURL, користувач може робити запити до вебмодуля та комунікувати з ним або можна встановити більш зрозумілий користувачу GUI – Postman, що дає змогу переглядати запити та відповіді у гарному вигляді, як це зображено на рисунку 3.1.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

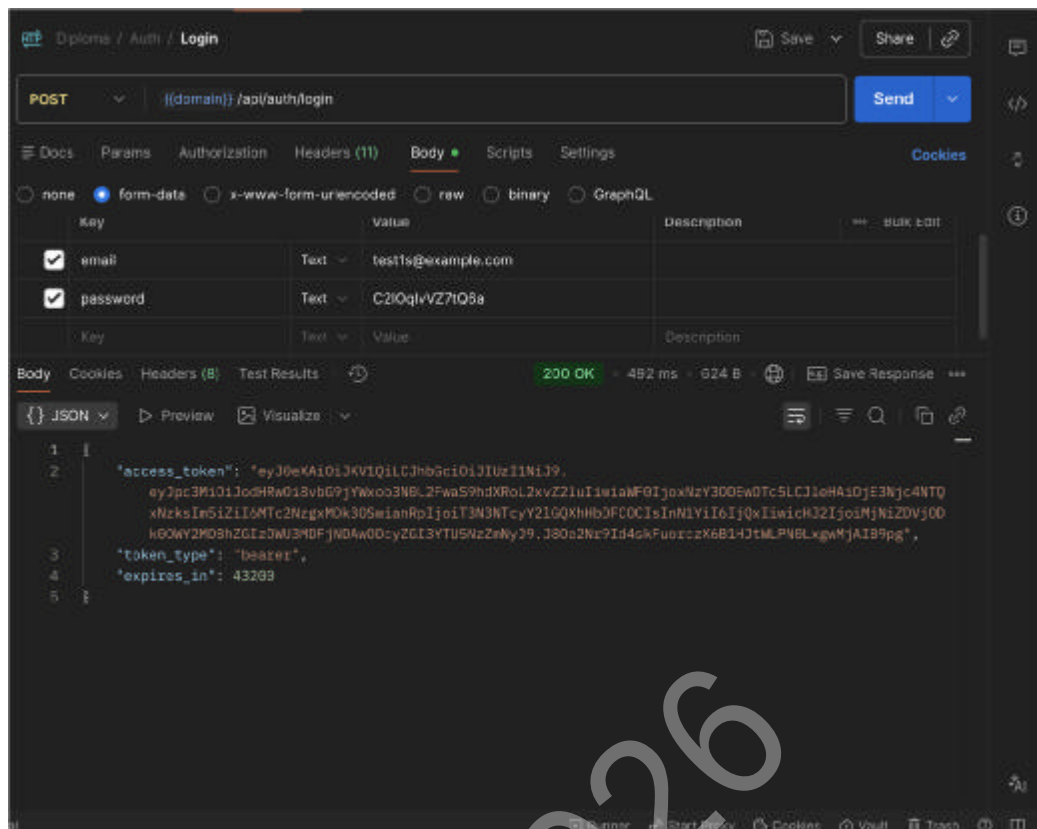


Рисунок 3.1 – Графічний інтерфейс користувача в Postman

Комунікація із сервером відбувається через захищений протокол HTTPS, який де-факто є стандартом безпеки у мережі Інтернет. Варто відзначити, що дані між клієнтом і сервером передаються у форматі JSON. Вся архітектура реалізує інтерфейс RESTful API, який де-факто якнайкраще підходить для клієнт-серверної моделі.

Вебмодуль являє собою вебсервер та основний додаток, написаний на мові програмування PHP з використанням вебфреймворку PHP. База даних, з якою взаємодіє основний додаток є реляційною, що також реалізовує клієнт-серверну модель, але спілкування відбувається через протокол TCP.

Інтелектуальний аналіз даних відбувається через один із API-сервісів від компанії OpenAI – Responses API. Responses API використовує RESTful інтерфейс для зручного та швидкого обміну даними через захищений протокол HTTPS.

3.2 Розробка структурної схеми

Структурна схема є одним із найважливіших аспектів при проектуванні будь-якого сучасного ІТ-проєкту або програмного продукту, що надає змогу відобразити найголовніші компоненти системи та їхню взаємодію один з одним. Перш за все структурна схема повинна показувати принцип роботи даного ІТ-проєкту, а саме як він працює всередині, та його взаємодія з іншими компонентами. Структурні зв'язки призначені для того, щоб показати взаємодію між різними блоками всередині схеми. Структурна схема для програмного забезпечення системи інтелектуального аналізу даних для управління звичками людини представлена на рисунку 3.2.

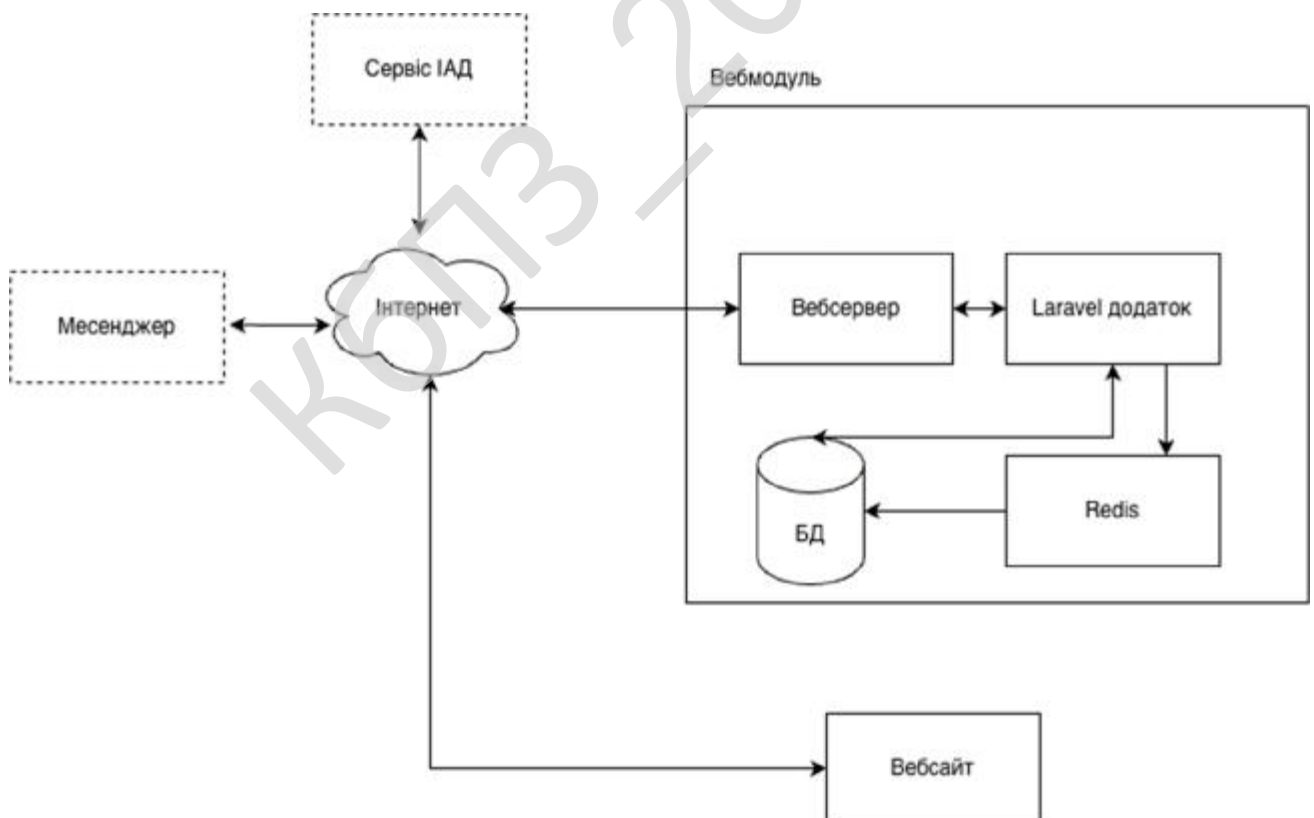


Рисунок 3.2 – Структурна схема для програмного забезпечення інтелектуального аналізу даних для систем управління звичками людини

До складу цієї структурної схеми входять наступні функціональні елементи: месенджер, вебсайт, сервіс, що надає можливості ШІ для ІАД, вебмодуль, який складається з: вебсерверу, Laravel-додатку, Redis та БД.

Зв'язок між компонентами на схемі відбувається через мережу Інтернет, а саме захищений протокол HTTPS. Користувач, який відвідує вебсайт, звертається через HTTPS до вебмодуля, тобто до сервера, який перенаправить запит до центру прийняття рішень, а саме Laravel-додатку. Laravel-додаток, завдання якого полягає в обробці користувацького запиту, може зберігати дані для персистентного використання у реляційній базі даних PostgreSQL. Також існує певний зв'язок між Laravel додатком та Redis, який має за мету, на відміну від PostgreSQL збереження короткотривалих даних, для їх швидкого доступу.

Для впровадження штучного інтелекту для інтелектуального аналізу даних буде використано один із сервісів OpenAI – Responses API. Зв'язок між додатком та сервісом від OpenAI відбувається також через захищений протокол HTTPS. Завдання Responses API полягає в обробці даних, надісланих вебдодатком та надсиланні відповіді назад вебдодатку.

Сервіс, що надає можливості генеративного штучного інтелекту, що і реалізовує покладений в основу праці інтелектуальний аналіз даних є один із сервісів, що надає компанія OpenAI – ResponsesAPI. Насправді, компанія OpenAI, що є творцем Generative Pretrained Transformer (GPT), зарекомендувала себе як лідера на ринку генеративного штучного інтелекту, не лише через одні з найкращих нейронних мереж в світі, але й через їхню платформу для розробників – OpenAI API Platform, що надає гнучкий, уніфікований та підтримуваний програмний інтерфейс, для реалізації своїх додатків на основі першокласних моделей нейронних мереж, таких як GPT 5, GPT 5.4 тощо. Документація, що представлена на OpenAI API Platform, є також інтерактивною, підтримує декілька мов програмування, надає сучасні приклади роботи з нею та багато іншого, що робить її оптимальним варіантом для використання у власних вебдодатках.

Месенджер, який представлений на схемі, також може потенційно виступати у ролі клієнта, який звертатиметься до вебмодуля. Варто зазначити, що не кожен месенджер має змогу вбудовувати вебсайт у нативний додаток месенджеру. Насправді, вся технологія зводиться до того, що месенджер надає змогу створити свій додаток всередині нього, лише створивши вебдодаток та вбудувавши його в нього належним чином. Тобто потрібно лише реалізувати вебсайт, у вигляді клієнта і вбудувати його в нативний додаток месенджеру.

3.3 Розробка функціональної схеми

Функціональна схема є невід’ємною частиною сучасних успішних ІТ-проектів. На цій схемі зображені основні функціональні зв’язки в системі та функції, що має наявна система. Також вона є графічним способом візуалізації процесів, що відбуваються всередині системи та потоків вхідних та вихідних даних. Функціональна схема повинна забезпечити високорівний огляд, того як кожна частина системи взаємодіє з іншими та як це впливає на загальний результат.

Функціональна схема програмного забезпечення системи інтелектуального аналізу даних для управління звичками людини, повинна розкрити в деталях структурну схему, що була розроблена раніше, але від самого початку і до кінця, інакше кажучи, від запиту користувача через мережу Internet до відповіді, часто цю модель називають Request-Response моделлю вебдодатку. В схемі представлений вебсервер nginx, додаток написаний за допомогою фреймворку Laravel, комунікація зі сторонніми сховищами такими як Redis і база даних PostgreSQL та сторонній сервіс від OpenAI – Responses API.

Функціональна схема програмного забезпечення системи інтелектуального аналізу даних для управління звичками людини представлена на рисунку 3.3.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

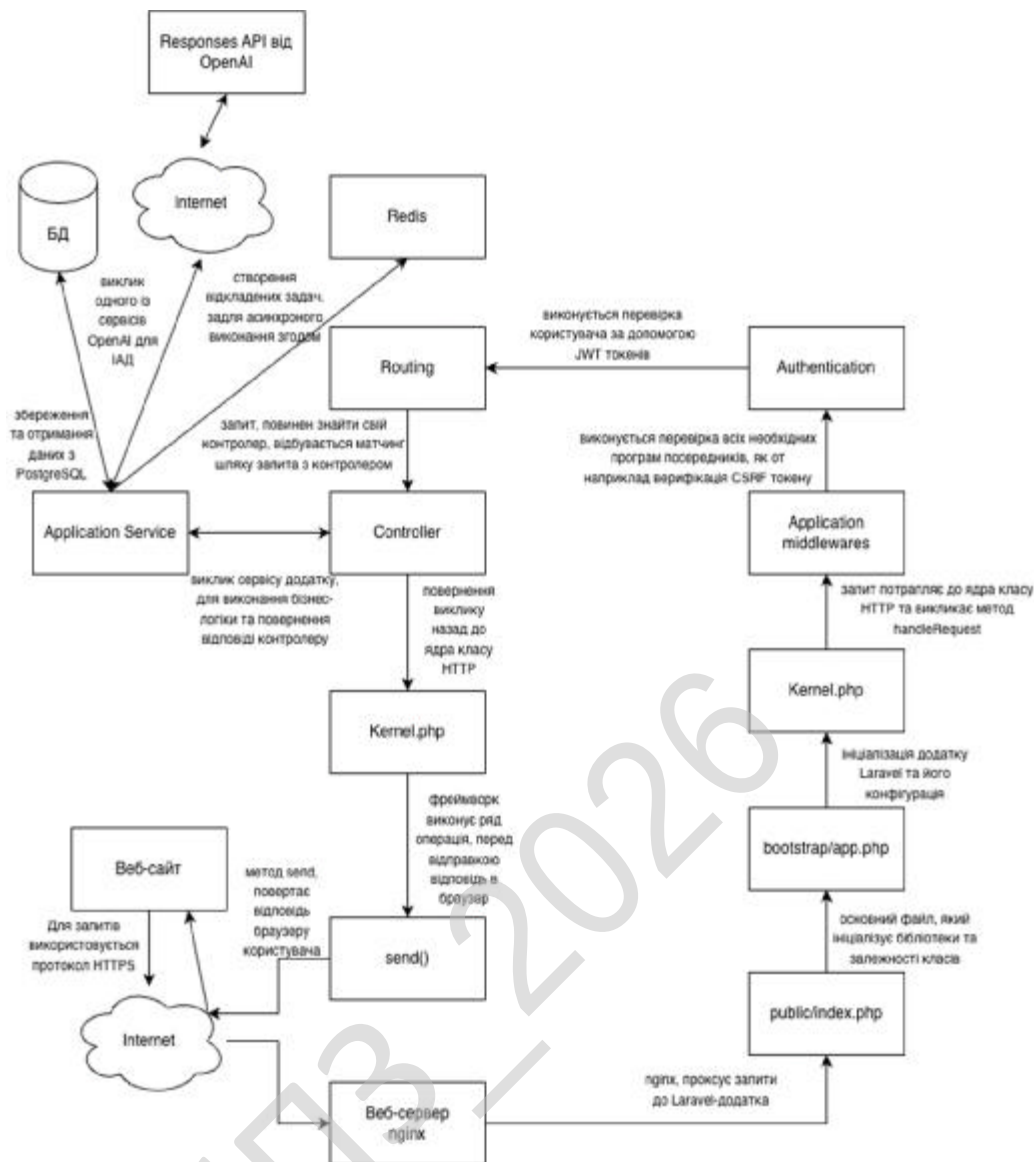


Рисунок 3.3 – Функціональна схема програмного забезпечення системи інтелектуального аналізу даних для управління звичками людини

Функціональна схема повинна відображати наступну ідею. Користувач відвідує клієнт, що може виступати як вебсайт, написаний на HTML, CSS та JavaScript або месенджер, що підтримує вбудований вебсайт в додаток або навіть виконання запитів через додаток Postman.

Далі клієнт робить запит до вебмодуля, який своєю чергою є вебсервером, таким як nginx. Вебсервер, що розміщений, на AWS EC2 екземплярі, інакше кажучи приватному сервері, що виконується під операційною системою Linux та дистрибутивом Ubuntu, виступає в ролі reverse-proxu. Завдання вебсервера це отримати запит та перенаправити його

на один з php-файлів – `public/index.php`, що знаходиться в корені проєкту, який в свою чергу ініціалізує типовий додаток створений за допомогою фреймворку Laravel. Файл `public/index.php`, завантажує всі необхідні бібліотеки та класи через Composer і потім отримує готовий екземпляр Laravel-додатку із файлу `bootstrap/app.php`. Перший крок Laravel-додатку полягає у тому, щоб створити сам додаток та сервісний контейнер.

Початкова ініціалізація або bootstrapping дуже поширена практика серед сучасних вебфреймворків, і є невід’ємною частиною успішних вебдодатків, тому що дозволяє ініціалізувати всі необхідні компоненти вебфреймворку, налаштувати сервісний контейнер, а також багато іншого. Важливою частиною початкової ініціалізації Laravel є сервісні провайдери. Спочатку створюється екземпляр вебфреймворку, тобто нашого додатку, з всіма зареєстрованими сервіс провайдерами, де саме і може бути реалізована частина сервісних контейнерів, які потім можна буде використати в додатку, просто звернувшись до екземпляру додатку, та отримати необхідний клас із сервісного контейнеру, що є дуже зручним для розробника, і також використовується в цьому проєкті.

Сервісний контейнер в Laravel – це потужний інструмент для управління залежностями класів та виконання ін’єкції залежностей. Ін’єкція залежностей, може виявитись складною спочатку, але в кінці кінців, все зводиться до того, що фреймворк дивиться, де використовується в параметрах методів класу інші залежності, наприклад інші об’єкти класів, і робить “ін’єкцію” цих об’єктів прямо в метод цього класу, що робить можливим подальше використання цих об’єктів в тілі цього методу, таким чином не потрібно створювати об’єкти класів вручну, достатньо лише прописати їх в параметрах методу, а сервісний контейнер подбає про це сам.

Наступним кроком, вхідний запит потрапляє до класу HTTP ядра: `\Illuminate\Foundation\Http\Kernel` а потім, і до самого методу класу: `handleRequest()`[35]. Варто зазначити, що клас HTTP ядра зберігає масив з усіма необхідними функціями, які запускаються перед тим, як запит виконується. Одним з цих прикладів є конфігурація самого Laravel додатку.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі, клас HTTP ядра починає пропускати запит через набір програм-посередників (middlewares) або ж інакше кажучи, класів фільтрів, далі в роботі буде використовуватись термін middleware.

Middleware – це такий механізм, що використовується в багатьох сучасних вебдодаток, його суть полягає в інспектуванні та фільтруванні HTTP запитів до вебдодатку. Наприклад, Laravel надає middleware для верифікації того, чи автентифікований користувач, і в залежності від статусу користувача перенаправити його на сторінку автентифікації, або далі продовжити виконання запиту.

Варто також відзначити, що метод класу HTTP ядра `handle()` приймає на вхід об'єкт, який саме репрезентує поточний запит: `Request` та завжди повертає назад об'єкт класу `Response`, що говорить про те, що у кожного запиту повинна бути відповідь, часто цю модель називають `Request-Response`[36]. Справді, клас HTTP ядра є такою собі чорною коробкою, що відображає наш додаток. Якщо ми передамо HTTP-запит то завжди отримаємо HTTP-відповідь.

Щойно додаток, ініціалізувався зі всіма необхідними параметрами та класами, клас який відображає запит `Request` передається обробці класу маршрутизатора, який обере необхідний шлях до контролера, який саме оброблюватиме запит. Також якщо існують якісь `middlewares` вони також спочатку запустяться перед класом контролера.

Щойно маршрут буде оброблений методом класу контролеру, об'єкт класу `Response` повертається назад до оброблювача запитів фреймворку, тобто до класу HTTP ядра `Kernel`, який в свою чергу готує відповідь перед відправкою клієнтові, додаючи необхідні заголовки відповіді, або якимось іншим чином модифікуючі, цю відповідь і насамкінець метод `send()` надсилає відповідь на запит клієнту. Це і є уособленням повно життєвого циклу вебфреймворку `Laravel`, що використовується в даному проєкті, розуміння того як працює життєвий цикл є дуже корисним, тому що вебфреймворк перестає здаватись чимось магічним для розробника, а навпаки стає потужним інструментом в його інструментарію.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

3.4 Розробка діаграми процесів.

Одним з не менш, важливих інструментів для візуалізації, аналізу та документування всіх процесів є діаграма процесів, що відбуваються в рамках системи. Вона дозволяє явно показати зв'язки на різних етапах, а також те, як кожен елемент процесу впливає на загальний результат. Зазвичай діаграму використовують для детального відображення та опису алгоритмів, потоків даних між різними компонентами всередині цієї системи.

Діаграма процесів для програмного забезпечення системи інтелектуального аналізу даних для управління звичками представлена на рисунку 3.4.

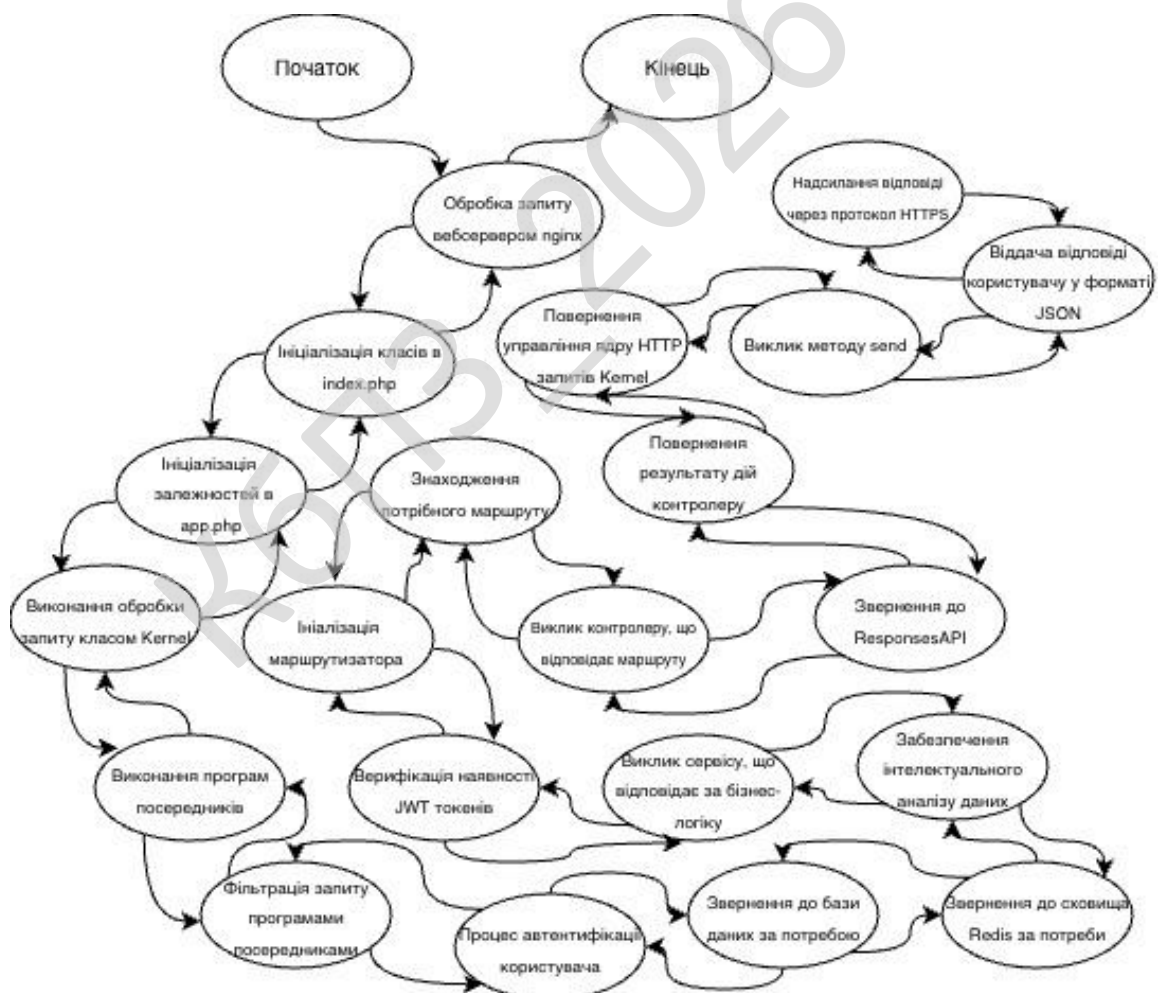


Рисунок 3.4 – Діаграма процесів програмного забезпечення інтелектуального аналізу даних для систем управління звичками людини

Початок діаграми процесів наведеної вище починається з обробки вхідного запиту вебсервером nginx. Завдання вебсерверу nginx, який виступає в ролі reverse proxy – це проксувати запит до PHP-FPM процесу, іншими словами додатку Laravel. Тобто всі запити, які сприймає nginx, проксуються до Laravel-додатку.

FastCGI Process Manager (FPM) – це одна з потужних технологій, що лежить в основі сучасного PHP, що дозволяє використовувати PHP як основну мову програмування для високонавантажених вебдодатків.

Наступним кроком, запит завжди потрапляє до файлу, який знаходиться у теці public/index.php. Насправді файл index.php не містить великої кількості коду, він лише ініціалізує сам фреймворк Laravel. Ще однією важливою особливістю цього файлу, є те що він завантажує автозавантажувач класів Composer, за допомогою якого всі залежності та бібліотеки додаються під час виконання коду.

Після цього відбувається налаштування фреймворку у файлі bootstrap/app.php, де створюється контейнер залежностей після чого запит потрапляє до методу handleRequest класу Illuminate\Foundation\Http\Kernel, де починається обробка запиту на рівні фреймворку. Перш ніж виконати запит і передати його до коду розробника додатка, Laravel запускає ряд класів, що мають за мету ініціалізувати додаток, а саме до них входять: ініціалізація змінних оточення, завантаження конфігурації додатку, налаштування обробників помилок, реєстрація фасадів, реєстрація провайдерів та введення в дію провайдерів.

Після цього запит проходить вже серію програм-посередників, які створені розробником додатку або ж самим фреймворком. Наприклад Illuminate\Foundation\Http\Middleware\ValidateCsrfToken, який за замовчування увімкнений до групи web, і виконує перевірку на токен CSRF в запиті, і порівнює з тим, що знаходиться в сесії для цього користувача, який зробив запит. Якщо токени співпадають, це означає, що користувач той за кого себе видає. Або ж інший приклад, де потрібно перевірити чи користувач

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

автентифікований, для цього теж можна скористатись програмою-посередником, яку надає сам фреймворк – auth. Якщо користувач не є автентифікованим, він отримує сторінку 401, якщо ж навпаки, то запит проходить далі.

Cross-site request forgery (CSRF) – це такий вид атаки, де...

Після того, як запит був оброблений, всіма програмами посередниками, запит потрапляє до маршрутизатору, і повинен виконуватись контролером, якщо такий маршрут на шляху запиту знайшовся. Якщо маршруту за таким шляхом запиту не знайшлося, фреймворк поверне помилку 404 – сторінку не знайдено.

Бізнес-клас повинен зберігати в собі всю логіку, про те, як виконувати запит від користувача. Він навіть може звернутися до одного з сервісів OpenAI – Responses API, щоб забезпечити інтелектуальний аналіз даних для користувача. Також дуже часто в подібних класах міститься логіка взаємодії з базою даних або іншим сховищем, як от Redis. Такий підхід, де бізнес-логіка застосунку виноситься в окремі шари вебдодатку зарекомендувала себе часом, і відповідає принципам, що покладені в базових принципах об'єктно-орієнтованого програмування та дизайну класів – SOLID.

Перша буква S цього акроніму, говорить про те, що клас повинен мати одну і тільки одну причину для внесення змін, інакше кажучи клас повинен мати єдину відповідальність. O – це принцип відкритості закритості, що відповідає проектуванню класів, і робить їх гнучкими у використанні. L – принцип підстановки Барбари Лісков. I – принцип, що наголошує на розділенні великого інтерфейсу на менші. D – високорівневі класи не повинні залежати від низькорівневних обидва повинні залежати від абстракцій.

На схемі, також вказана база даних Redis з типом ключ-значення, що може використовуватись за потреби для кешування результатів із бази даних, або навіть цілих відповідей у форматі JSON. Таким чином, Redis – це потужний інструмент, який використовується в багатьох високо навантажених вебдодатках сьогодні. Redis має відкритий вихідний код написаний мовою

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

програмування C, що є досить близькою до процесору, що і робить його таким швидким. Також варто зазначити, що суть Redis полягає в тому, що на відмінну від традиційних реляційних баз даних, які зберігають дані на диску, Redis зберігає їх в Random Accessed Memory (RAM) комп'ютеру.

Основний алгоритм роботи інтелектуального аналізу даних, а саме агрегуванні та отриманні звичок із бази даних PostgreSQL, формуванні запиту та відправлення його до одного із сервісів OpenAI – Reponses API реалізовується саме в бізнес-класі: HabitsService, методі analyze на вхід якому потрібно передати існуючу звичку користувача, для якої виконуватиметься інтелектуальний аналіз даних.

Після того, як бізнес-клас виконав всю необхідну логіку, він може повернути відповідь контролеру, який своєю чергою поверне відповідь браузеру, у вигляді HTML або JSON. Але перед тим, як відповідь остаточно надійде до браузера, відповідь може знову пройти через деякі програми-посередники, що інколи також може бути зручним. Наприклад, перед тим як відправити остаточно відповідь користувачу, потрібно також надіслати лист на електронну адресу користувача або створити асинхронну задачу, тощо.

Підсумовуючи, у розділі було описано функціонування системи, а саме які технології будуть використані, як клієнт буде взаємодіяти із сервером, протоколи та інші важливі технології. Розроблено структурну схему, яка покладена в основу ПЗ, де прописані всі структурні зв'язки між компонентами, такими як вебсайт, сервіс ІАД, месенджер та вебмодуль. На основі блок-схеми, що представляє структурну схему, розроблено та описано функціональну схему, що описує детально функціональні зв'язки в ПЗ, а також функції/класи, що будуть використані. Насамкінець для наочного представлення функціонування системи, розроблена діаграма процесів, що описує процеси на всіх рівнях додатку.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Для реалізації ПЗ інтелектуального аналізу даних було ретельно обрано наступний технологічний стек: PHP, Laravel, Docker, PostgreSQL, nginx та Ubuntu. PHP є зрілою мовою програмування з довгою історією, проте, активно розвивається і не збирається здавати свої позиції. Laravel є одним з найсучасніших та найактивніш підтримуваних фреймворків серед розробників, що робить його чудовим вибором для цього типу ПЗ. Складно уявити сучасну локальну розробку без технології контейнеризації, тому було обрано Docker, щоб забезпечити належний розробницький досвід. Інші технології такі як PostgreSQL, nginx та Ubuntu також були ретельно відібрані для цього ПЗ, так як саме вони пройшли довгу історію розробки та зарекомендували себе як надійні інструменти для розробки вебдодатків серед розробників всього світу. Насамкінець, на основі розробленої структурної та функціональної схем та діаграми процесів в розділі 3.1, 3.2 та 3.3 відповідно можна детально розробити блок-схеми та зробити опис алгоритмів функціонування системи на кожному з етапів.

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Блок-схема інтелектуального аналізу даних звички людини зображена на рисунку 4.1.

Процес розпочинається з того, що відправляється запит методом POST по наступному відносному шляху: `/api/habits/{uuid}/get-analysis`, що очікує параметром універсально-унікальний ідентифікатор звички. Існує декілька типів ідентифікаторів, які можуть використовуватися на рівні бази даних, щоб визначити запис як унікальний. Це зростаюче числове представлення або ж

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

UUID. Зазвичай, у досить великих вебдодатках використовують саме UUID, тому що це надійно та приватно. UUID вирішує декілька проблем, а саме унікальність ключа та його приватність.

Розробники UUID стверджують, що майже неможливо, щоб UUID повторився, тому це робить його дуже корисним для великої кількості записів. На відміну від традиційних натуральних чисел, якими можуть бути представлені ідентифікатори записів у таблицях, UUID дає приватність цих записів. Можна навести досить простий приклад, коли ідентифікатор запису міститься в URL сайту, до прикладу: /api/users/123. 123 – це і є ідентифікатор, по якому згодом можна знайти цей запис у базі даних, також недоброчесні користувачі можуть легко дізнатись інформацію про наступні ідентифікатори, такі як: 1, 2, 3, ..., 124, що робить це не завжди доречним, для деяких вебдодатків з метою безпеки, але з UUID таких проблем немає. З UUID попередній приклад може бути представлений як: /api/users/0648dff9-91a0-4b6e-861b-ac5fe826a240.

Після того, як запит був сформований та відправлений, наступним кроком є перевірка, чи існує саме ця звичка з цим UUID в базі даних в таблиці habits, якщо так, переходимо до наступного кроку, якщо ж ні, алгоритм завершується з помилкою 404 Not Found.

Щойно звичка була знайдена, потрібно перевірити чи користувач є авторизованим для дії над нею, інакше кажучи, чи є користувач власником цієї звички. Якщо користувач є творцем переданої звички, значить потрібно перейти до наступного кроку, якщо ж ні повернути помилку 403 Forbidden та завершити алгоритм.

Для якісного інтелектуального аналізу даних звички людини, потрібно мати набір даних, для того, щоб сервіс від OpenAI, зміг надати якісний та ґрунтовний аналіз і дати поради щодо звички, тому було прийнято рішення встановити мінімальну кількість нотаток до звички – 10. Інакше кажучи, користувач повинен створити 10 нотаток, що відносяться до цієї звички.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Число 10 було підібрано емпіричним шляхом, тому що якщо нотаток до цієї звички менше 10, інтелектуальний аналіз даних може бути не таким потужним, тому що даних не вистачає, і таким чином, штучному інтелекту буде складно знайти поведінкові патерни, що знаходяться в цій звичці. Інша причина, це економічна, а саме відправлення запиту до сервісу від OpenAI, коштуватиме певних грошей. Хоча і в не великому масштабі, це не багато, але якщо масштабувати це на десятки тисяч користувачів, це може виявитись зовсім не ефективно з точки зору економії коштів.

Нотатки діляться на два типи: позитивні та негативні. Щойно користувач має 10 нотаток, відправляється запит на створення відповіді від сервісу OpenAI, Responses API, якщо ж 10 нотаток немає ще, аналіз не може бути виконаний і на цьому алгоритм завершується з помилкою 422 Unprocessable.

Натомість, якщо 10 нотаток у користувача є, відповідь від сервісу Responses API, може бути приблизно наступною:

```
{  
  "analysis": "1. Ключові патерни успіху\n- Ранній підйом і негайний старт без телефону.\n- Чіткий план на день і розуміння що робити та навіщо.\n- Початок із найскладнішого завдання дає інерцію для решти..."  
}
```

Як можемо спостерігати з відповіді у форматі JSON, сервіс зі штучним інтелектом від OpenAI, зміг проаналізувати 10 нотаток, які були прикріплені до цієї звички, і надати ґрунтовну відповідь, для подальшого покращення цієї звички користувачем. На цьому алгоритм завершується зі статусом 200 OK.

Аналіз було здійснено за допомогою сервісу штучного інтелекту від OpenAI, а саме Responses API. Хоча модель, що була використана в цій системі – це була GPT-5, також можна використовувати і новіші моделі, все залежить від задачі та від бюджету проекту. В нашому випадку GPT-5 показує, хороші результати інтелектуального аналізу даних, хоча існують і новіші моделі, які перевершують GPT-5 в багатьох оцінках від незалежних експертів, тим не менш, я вважаю, що дана модель, є доцільною для використання в системі інтелектуального аналізу даних.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

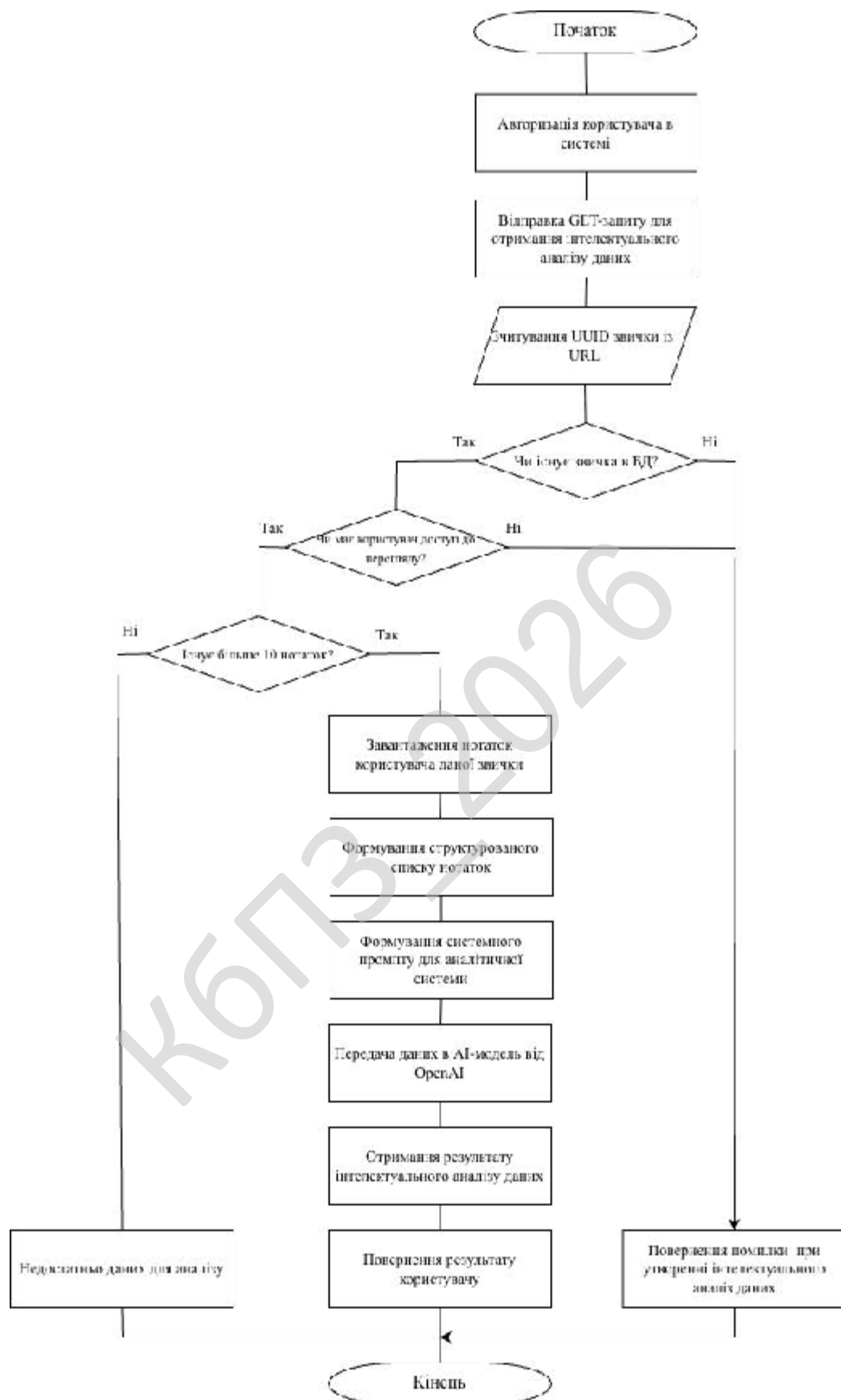


Рисунок 4.1 – Основна блок-схема програмного забезпечення системи інтелектуального аналізу даних

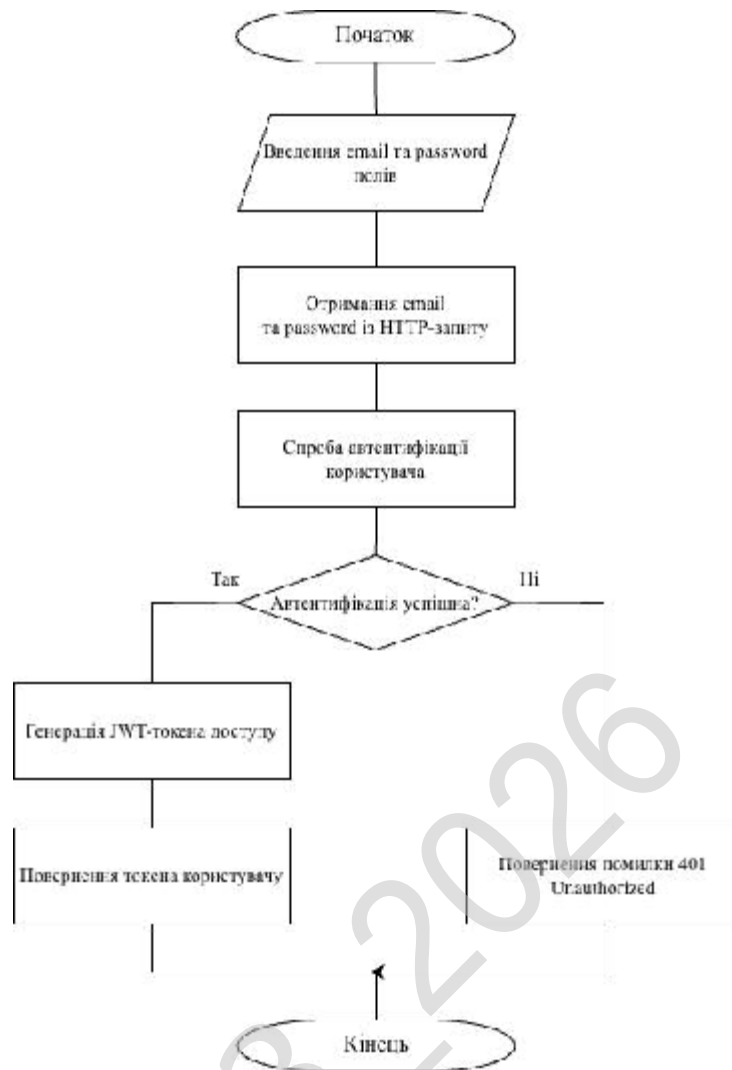


Рисунок 4.2 – Блок-схема алгоритму автентифікації користувача за допомогою JWT-токена

Опис алгоритму автентифікації користувача за допомогою JWT-токена, відображено на рисунку 4.2. У сучасних вебдодатках майже не обійтись без механізму автентифікації користувачів. За основу алгоритму була взята бібліотека `tymondesigns/jwt-auth`[38], яка надає можливості використання JWT-токенів, використовуючи фреймворк `Laravel`.

JSON Web Tokens – це стандарт, який надає змогу передавати інформацію у вигляді JSON об’єктів через мережу Інтернет надійно. JWT-токени можуть бути підписані, використовуючи наступні алгоритми: HMAC або приватний/публічний ключі RSA або ECDSA[39].

Найбільш популярним місцем використання JWT токенів, є саме аутентифікація. Досить лише згенерувати JWT токен, і кожного разу користувачу його передавати в HTTP-заголовок Authorization з типом Bearer для того, щоб отримати доступ до захищених ресурсів.

JWT токен складається з трьох частин: заголовок, тіла та підпису, всі три частини з'єднані між собою символом звичайної “.”. Заголовок, зазвичай складається з двох частин, а саме типу токена та обраного алгоритму для підпису. Тіло містить в собі, так звані заяви. Заяви – це твердження про дану сутність, як от користувач. Є всього лише три типи заяв: публічні, приватні та зареєстровані.

Підпис формується наступним чином. Наприклад було обрано алгоритм HMAC SHA256, тоді підпис складатиметься із заголовку, який закодований як Base64 рядок та тіла, що також буде закодовано як Base64, потім обидві частини з'єднуються символом “.”. Останнім параметром функції, яка реалізовує алгоритм, потрібно передати секретний рядок, що базово означає ключ, який зберігається тільки у власника, і не піддається розголошенню.

Приклад

JWT-токена:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOi8vbG9jYXRob3N0L2FwaS9hdXRoL2xvZ2luIiwiaWF0IjoxNzY1Mjc2MDkzLCJleHAiOjE5NjUyNzk2OTMsIm5iOiI6MTc2NTI3NjA5MywianRpIjoiQjhkQ1hEZWRrSGx2Q0w3bSIsInN1YiI6IjM5IiwicHJ2IjoiMjNiZDVjODk0OWY2MDBhZGIzOWU3MDFjNDAwODcyZGI3YTU5NzZmNyJ9.A8ZCfodU01x0r1iIj7L9X5ogK3vx3umIa4t-IDwremE

Таким чином, алгоритм, що зображений на рисунку 4.2, починає свою роботу, з того що на вхід передаються два поля email та password, після цього перевіряється чи існує такий користувач вже в системі, якщо ні помилка 401, повертається користувачу із повідомленням “Unauthorized”. Якщо такий користувач існує з такою електронною адресою, наступним кроком, перевіряється чи співпадає введений пароль з тим, що зберігається в базі даних. Якщо ні, помилка 401 повертається користувачу. Якщо так, для цього користувача, відбудеться процес генерації JWT токена через метод `JWT::fromUser()`, далі згенерований токен буде повернутий користувачу у вигляді JSON об’єкта з наступними полями: `access_token`, `token_type`, `expires_at`.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

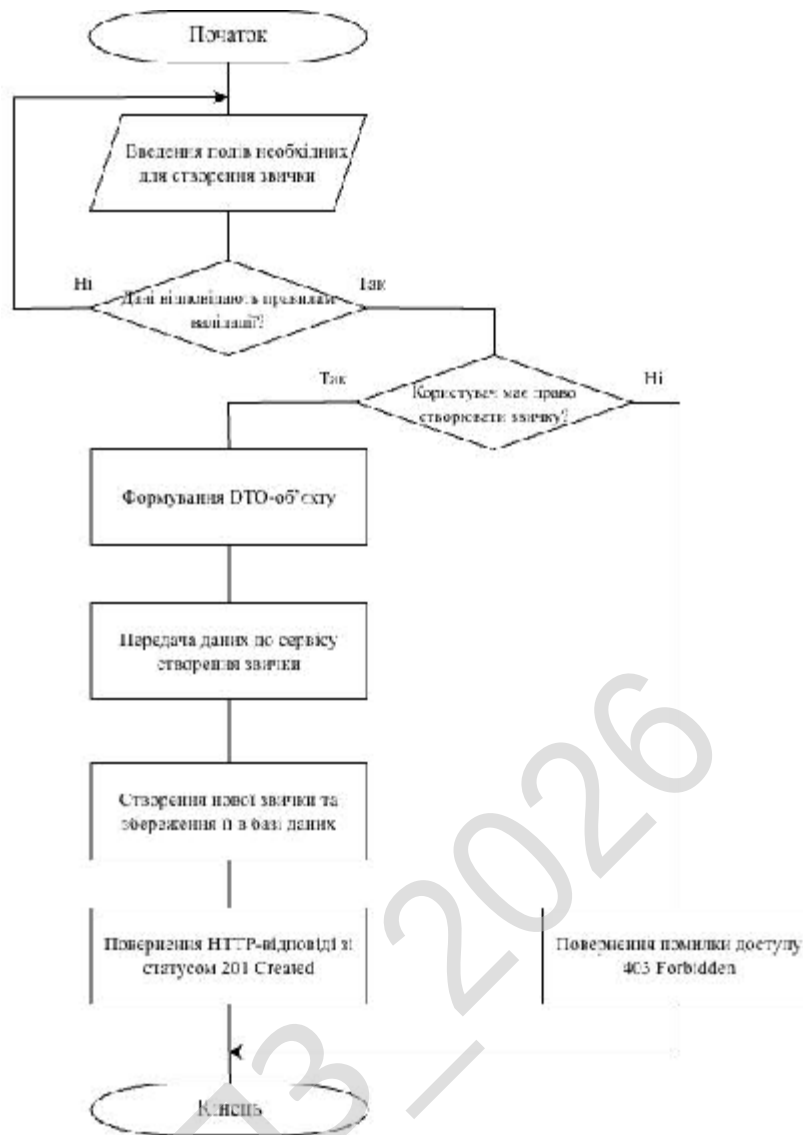


Рисунок 4.3 – Блок-схема алгоритму створення нової звички

На рисунку 4.3 зображений алгоритм створення нової звички користувачем в програмному забезпеченні інтелектуального аналізу даних для управління звичками людини. Погляньмо на нього більш детально.

Спочатку, користувач, який користується клієнтом, наприклад вебсайтом, робить POST-запит на наступний ендпоінт. Ендпоінтом часто називають шлях до якогось ресурсу API, у нашому випадку це – /api/habits. POST – це насправді метод, яким відправляється HTTP-запит, існує конвенція використання правильних методів, для ендпоінтів, до прикладу, щоб просто отримати дані від сервера через API використовується метод GET, для створення POST, часткова модифікація ресурсу PATCH, повна заміна

існуючого ресурсу на інший PUT, для видалення використовується метод DELETE. Варто зазначити, що також існують інші методи, які не будуть розглядатися в цій кваліфікаційній роботі.

Щойно сервер отримав запит із методом POST, запускається процес перевірки автентифікації користувача. Якщо користувач немає відповідного ключу доступу, відомого як `access_token`, або він буде недійсним, користувач отримає помилку 401 Unauthorized, і на цьому алгоритм завершується.

Якщо користувач має дійсний ключ доступу до системи, система передасть виконання контролеру `HabitController::store()`, який відповідає за створення звички, і розпочне процес валідації вхідних параметрів. Справді, звичка в системі, повинна містити в собі назву та тип, це є мінімально-обов'язкові поля для заповнення, якщо ж запит їх не міститиме, система поверне помилку 422 Unprocessable Entity, що означає, що переданий вміст тіла запиту, не відповідає певним правилам валідації і обробки запиту. Дійсно, якщо передати JSON об'єкт без обов'язкових полів: `name` та `type`. Запит буде відхилено.

Варто зазначити, що статуси, які використовуються для надсилання відповідей з серверу, відповідають загальноприйнятій конвенції. Інформаційні відповіді знаходяться у діапазоні: 100-199. Успішні статуси відповідей у діапазоні: 200-299. Статуси перенаправлень, з однієї сторінки на інший ресурс у діапазоні: 300-399. Помилки клієнта, яку ми розглянули вище та інші у діапазоні: 400-499. Помилки суто сервера у діапазоні: 500-599[41].

Для успішного створення звички, яка має тип “кількість” відповідає наступний коректний payload зображений на рисунку 4.4.

Key	Value
☒ name	Text Text
☒ description	Text Test habit
☒ type	Text count
☒ type_payload[target]	Text 10
☒ type_payload[measure_unit]	Text kilometers

Рисунок 4.4 – коректний payload для створення звички в Postman

З переданим вище payload створиться звичка і повернеться наступний JSON об'єкт, який зможе прочитати клієнт, у нашому випадку це вебсайт, написаний за допомогою бібліотеки React:

```
{
  "uuid": "df1a303a-5267-4370-87ed-c8cdeaf3461f",
  "habitable_type": "App\\Models\\CountHabit",
  "habitable_id": "beb62ec9-dfaf-4709-9b8c-6fb8d10d746d",
  "name": "Test",
  "description": "Test habit",
  "starts_at": null,
  "ends_at": null,
  "started_at": null,
  "ended_at": null,
  "user_id": 40,
  "created_at": "2026-03-03T11:39:36.000000Z",
  "updated_at": "2026-03-03T11:39:36.000000Z",
  "habitable": {
    "uuid": "beb62ec9-dfaf-4709-9b8c-6fb8d10d746d",
    "target": 10,
    "measure_unit": "kilometers",
    "created_at": "2026-03-03T11:39:36.000000Z",
    "updated_at": "2026-03-03T11:39:36.000000Z"
  }
}
```

4.2 Захист розробленого програмного забезпечення

Захист від несанкціонованого доступу до API ПЗ здійснюється за допомогою системи аутентифікації та JWT-токенів. Перед тим як почати використовувати ПЗ, користувач відправляє запит з методом POST на маршрут /login, для отримання JWT-токена, з яким надалі можна отримати доступ і до інших ресурсів, таких як звички та інтелектуальний аналіз даних.

Процес генерації та підтвердження, зображений на рисунку 4.2 Алгоритм автентифікації користувача за допомогою JWT-токена. Клас що відповідає за цей процес називається AuthController.php, який можна знайти у додатку Б. AuthController::login виконує процес ідентифікації користувача, і якщо

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

користувач успішно ідентифікований, метод поверне результат у форматі JSON, в якому міститиметься поле “access_token”, що і представляє собою JWT-токен.

Наступним кроком після автентифікації користувача є процес авторизації. Авторизація відбувається в класі HabitPolicy.php, в якому містяться описані методи, такі як view, viewAny, create, update та delete, що саме і визначають логіку доступу автентифікованого користувача до ресурсу. Наприклад, будь-який автентифікований користувач системи може створити звички, без винятку, але якщо користувач має намір видалити звичку, він може це зробити лише для своєї звички, тобто немає змоги видалити звичку іншого користувача. Варто зазначити, що перед перевіркою логіки авторизації користувача, потрібно впевнитись, що користувач автентифікований, для цього використовується програма-посередник “auth:api”, яку визначають для певної множини маршрутів, як от маршрути, що відповідають за створення, перегляд, оновлення та видалення звички. Таким чином, кожен метод HabitPolicy.php використовується у відповідному методі контролеру HabitController.php, для коректної авторизації дій користувача в системі щодо до ресурсу, як от звички.

Останнім але не менш важливим аспектом захисту ПЗ є шифрування користувацьких даних.

Laravel надає простий інтерфейс для шифрування та дешифрування тексту через OpenSSL, використовуючи AES-128/256. Всі зашифровані повідомлення підписані за допомогою message authentication code (MAC), що унеможливорює модифікування або втручання в процес, відколи повідомлення було зашифроване[43].

Advanced Encryption Standard (AES) – це симетричний алгоритм блочного шифрування, в якому розмір блоку завжди становить 128 біт, а розмір ключа може бути 128/192/256 біт. Так само як і DES, AES – це підстановочно-перестановочна системи, що може включати різну кількість раундів, залежно від довжини ключа, тобто у випадку з 128-бітним ключем, буде використано

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

10 раундів, у випадку з 256-бітним ключем, максимальна кількість раундів – 14. На вхід подається 128-й бітний блок відкритого тексту а на виході виходить 128-бітний блок зашифрованого тексту. Шифрування відбувається завдяки раундам, в яких відбуваються всі необхідні перестановки та підстановки. Дешифрування працює майже як і шифрування, але в зворотному напрямку[44].

Використання цього методу шифрування значно підвищує рівень кібербезпеки, створюючи надійний захист даних від несанкціонованого доступу.

Використаний алгоритм шифрування AES-256 був застосований для класів-моделей Habit.php та HabitNote.php. Кожна з моделей має властивість класу \$casts, в якій можна вказати, яке поле в таблиці, що відображає ця модель за архітектурою Object-Relational Mapper (ORM).

Таким чином, деякі з полів таблиць будуть надійно зашифровані, як от таблиця habits, яка безпосередньо містить особисті дані користувачів, як от: name, description та інші, будуть зберігати незрозумілі рядки, що унеможливилюватиме дешифрування без ключа APP_KEY, який надійно зберігається на сервері, у файлі .env.

У цьому розділі було реалізовано технічну складову ПЗ, а саме встановлення та ініціювання проєкту за допомогою технології Docker, дозволяє здійснити контейнеризацію ПЗ для локальної розробки. Таким чином було встановлено PHP 8.4, PostgreSQL та інші технології. Встановлено фреймворк Laravel з усіма необхідними бібліотеками.

Було імплементовано систему автентифікації за допомогою JWT-токенів, що дало змогу ідентифікувати кожного користувача як суб'єкта в системі.

Система авторизації разом з підходом Policy, що надає фреймворк Laravel дало змогу авторизувати дії користувачів в системі, таким чином обмежуючи доступ до ресурсів інших користувачів, таких як звички.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтелектуальний аналіз даних був імплементований за рахунок бібліотеки `openai-php/laravel` та сервісу від OpenAI – Responses API. Перш за все бібліотека дала доступ до зручних класів-ресурсів таких як Responses, що представляє собою обгортку над API від OpenAI, що реалізована на мові програмування PHP та інтегровану з фреймворком Laravel. Для отримання відповіді (response) від сервісу OpenAI в методі `HabitsService::analyze` було передано наступні параметри: назву моделі, системний промпт для моделі та нотатки користувача для цієї звички таким чином модель має необхідні вхідні дані для формування відповіді.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ЕКСПЛУАТАЦІЮ

Впровадження ПЗ інтелектуального аналізу даних для управління звичками людини передбачає низку підготовчих дій, для подальшої інтеграції у реальному середовищі.

Перед налаштуванням реального середовища для імплементованого вебмодуля були виконані такі підготовчі кроки:

1. Виконано мануальну перевірку кожного компонента системи, кожен ендпоінт API вебмодуля був належно протестований.
2. Створення Git репозиторію та завантаження коду на GitHub.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

Після того, як були виконані підготовчі кроки, можна перейти до створення та налаштування акаунту на платформі AWS:

1. Створення master-акаунту на платформі AWS.
2. Створення окремого акаунту з належними правами.
3. Налаштування MFA.

Маючи створений акаунт на цій платформі, можна перейти до створення EC2-екземпляру, який є VPS та встановлення всіх необхідних пакетів на ньому. На рисунку 5.1 зображено екран створення EC2-екземпляру в інтерфейсі від AWS. Щоб налаштувати необхідний EC2-екземпляр потрібно, встановити наступні параметри:

1. AMI – Ubuntu Server 24.04 LTS (HVM), SSD Volume Type.
2. Architecture – 64-bit (x86).
3. Instance type – t3.micro.
4. Key pair name – відповідний SSH-ключ для подальшого безпечного з'єднання з вебмодулем.

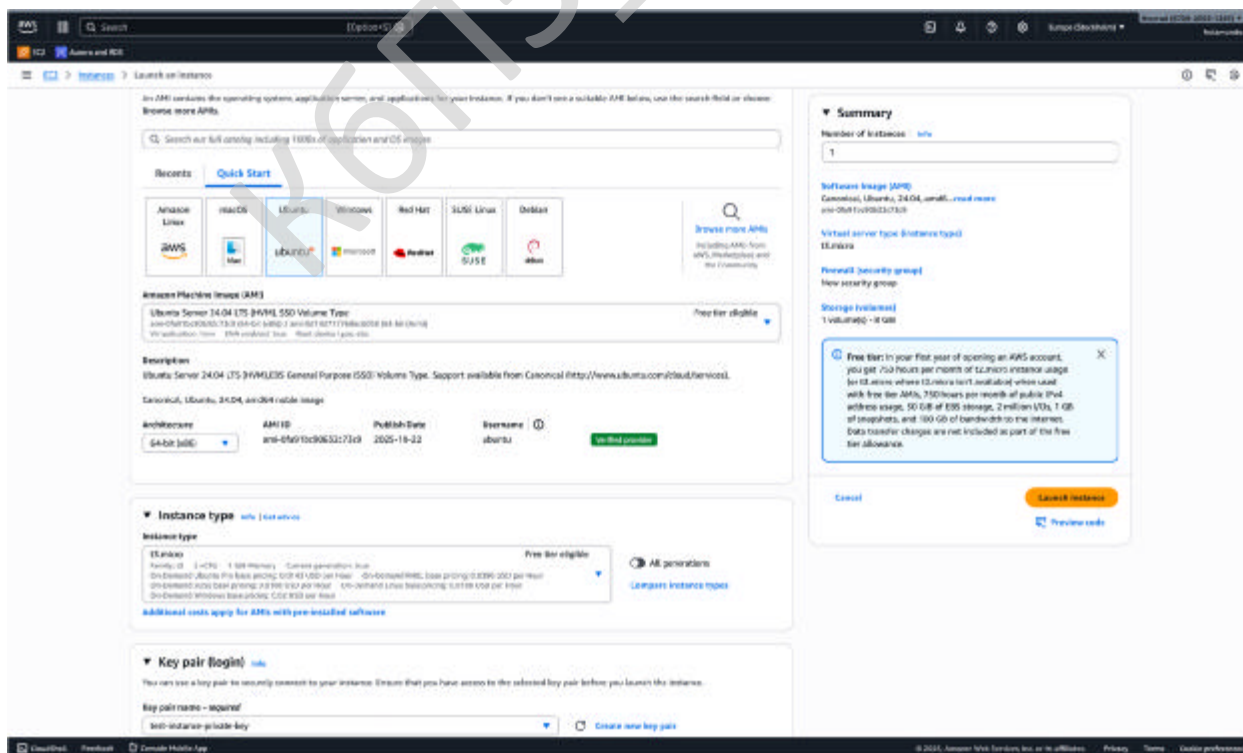


Рисунок 5.1 – Інтерфейс створення EC2-екземпляру на платформі AWS

Далі потрібно створити Amazon RDS, що виступає в ролі реляційної бази даних для вебмодуля. Інтерфейс створення RDS представлений на рисунку 5.2. Наступні параметри були обрані при створенні RDS, для ефективного використання ресурсів:

1. Database creation method – Full configuration
2. Engine type – PostgreSQL.
3. Engine version – 17.6-R2.
4. Templates – Free Tier.
5. DB instance identifier – diploma.
6. Credentials manager – Managed in AWS Secrets Manager.
7. Instance type – db.t4g.micro.
8. Storage – general purpose SSD (gp2).
9. Database authentication – password authentication.

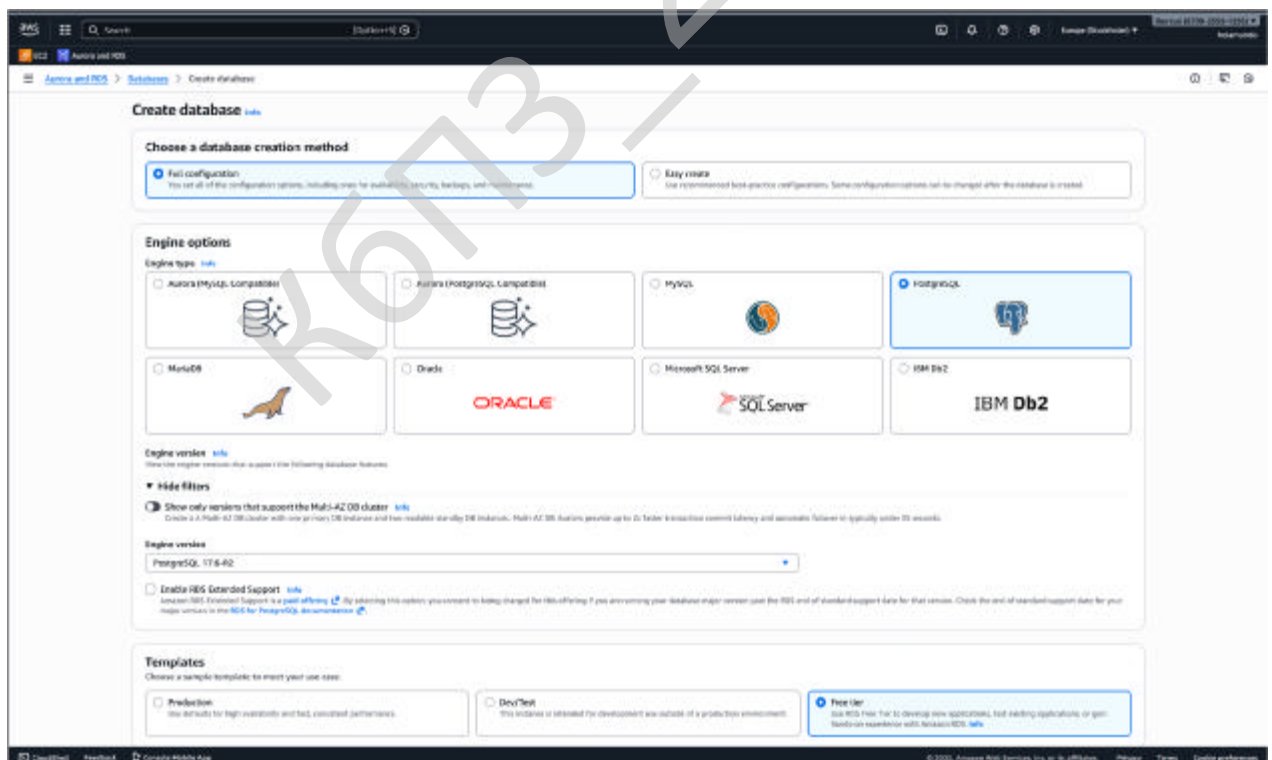


Рисунок 5.2 – Інтефейс створення RDS на платформі AWS

Після створення EC2 та RDS можна перейти до налаштування вебмодуля, а саме встановлення вебсервера та налаштування production середовища для фреймворку Laravel. Для початку потрібно під'єднатися до EC2 через SSH з'єднання. Щойно підключення було встановлено, можна перейти до оновлення та встановлення необхідних пакетів. Перш за все потрібно оновити всі пакети в системі, виконавши звичайну команду на Ubuntu: `sudo apt update`, потім відразу перейти до встановлення вебсерверу `nginx` та `php-fpm` разом із усіма іншими пакетами з файлу `Dockerfile`.

Щойно всі пакети для PHP та вебсервер встановлений залишається лише під'єднатись до Git репозиторію, завантажити репозиторій, та розмістити його за абсолютним шляхом: `/var/www/html`, оскільки він є стандартним для `nginx`. Після цього, потрібно оновити конфігурацію `nginx` та замінити її на конфігурацію, яка використовувалась для локальної розробки, лише з деякими відмінностями, як от `server_name` – вказати IPv4-адресу, що надає AWS для цього EC2.

Останніми кроками, були встановлення всіх необхідних пакетів проєкту за допомогою пакетного менеджера `Composer`. Потім генерація ключа додатку через `php artisan key:generate` та внесення змін у `.env` файл, слід бути особливо уважним з наступними полями: `APP_ENV` та `APP_DEBUG`. Поля повинні мати наступні значення: `production` та `false` відповідно. Насамкінець потрібно виконати міграції для створення необхідних таблиць в базі даних за допомогою команди `php artisan migrate`.

Насамкінець, потрібно переконатися в коректності роботи вебмодуля перейшовши за IPv4 адресою EC2-екземпляра та використати протокол HTTP. Після цього перевірити всі інші ендпоінти системи, такі як автентифікація та створення звичок разом з інтелектуальним аналізом даних. На рисунку 5.3 зображена відповідь від впровадженої системи у реальному середовищі в `Postman`.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

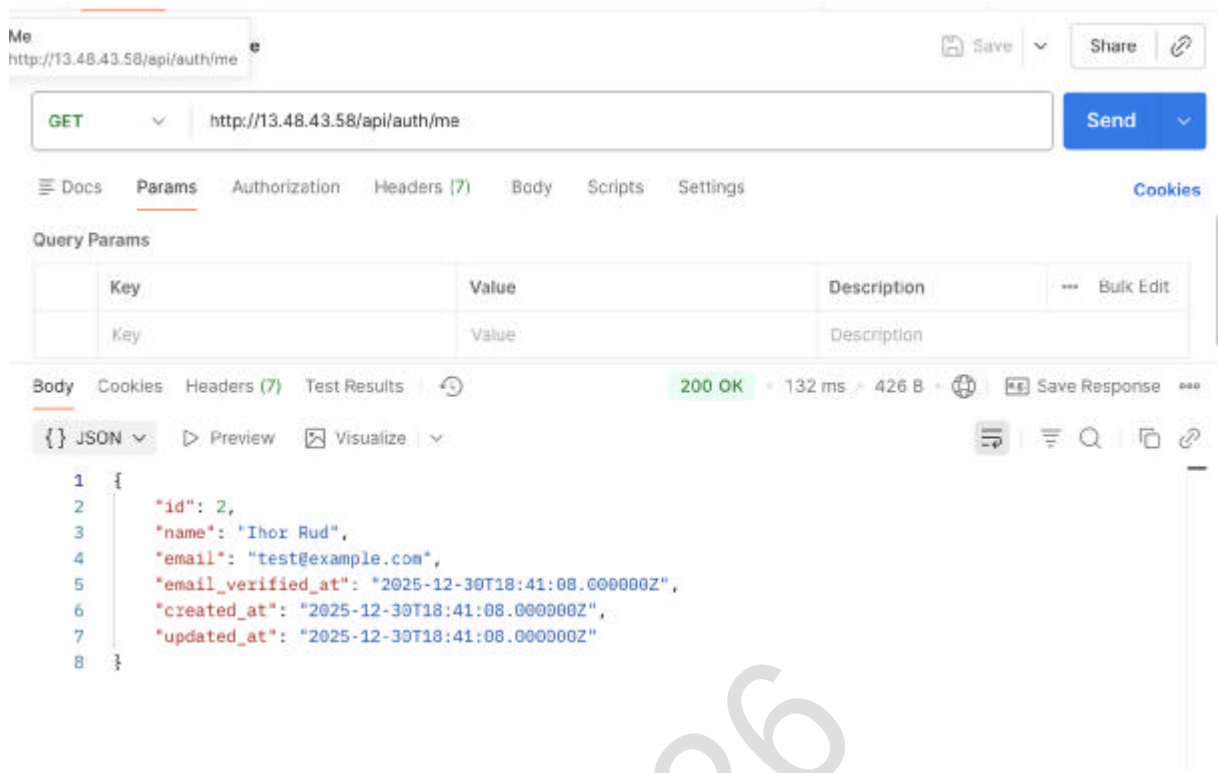


Рисунок 5.3 – Отримання відповідь від реального впровадженної системи в реальне середовище

Розгортання сучасного вебдодатку є одним із найважливіших етапів сучасних ІТ-проектів. У цьому розділі вдалося повністю налаштувати вебмодуль та створити екземпляр EC2, базу даних сервісу AWS RDS із встановленим PostgreSQL, а також виконати цілу низку підготовчих кроків, щоб налаштувати та впровадити систему ПЗ інтелектуального аналізу даних для управління звичками людини на одному з найбільш популярних хмарних сервісів – Amazon Web Services. Екземпляр EC2 надає можливість використання вебсерверу nginx та ядро ОС Linux із дистрибутивом Ubuntu. Екземпляр RDS надає можливості використання сучасної бази даних PostgreSQL, яка має відкритий вихідний код. Після цього було ретельно протестовано систему в реальному середовищі за допомогою Postman, який надає можливості зручнішого графічного інтерфейсу (GUI) для відправки HTTP-запитів до вебдодатку, і таким чином пришвидшуючи тестування.

К6ПЗ-2026

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ОСНОВНІ ВИСНОВКИ

За мету роботи ставилось завдання розробки ПЗ для підвищення ефективності набування нових звичок та позбавлення шкідливих звичок шляхом інтелектуального аналізу даних. Оскільки штучний інтелект стрімко розвивається в усіх сферах життя, генеративний ШІ чудово підходить як інструмент інтелектуального аналізу даних. Створення цього ПЗ націлене на підвищення ефективності процесу управління звичками людини.

Потреба у програмному забезпеченні, яке перш за все націлене на інтелектуальний аналіз даних у системах управління звичками людини може стати не тільки-но революційним, а й монетизованим згодом. ПЗ, яке буде побудоване повинно включати в себе також соціальний чинник, тобто взаємодія з іншими людьми. Інтелектуальний аналіз даних можна розширити не тільки для однієї людини, а для групи людей. Люди повинні мати змогу об'єднуватися разом, для набуття корисних звичок та аналізу своїх звичок.

Надано визначення що таке цільова аудиторія, зробити мінімальний аналіз цільової аудиторії для ПЗ інтелектуального аналізу даних для управління звичками людини, а також розібрати доцільні та схожі платформи, для розгортання схожих додатків. Було оглянуто три великі платформи, а саме: App Store, Google Play та Telegram Mini App, включаючи можливості, що надає кожна з цих платформ.

Було проведено аналіз існуючих програмних рішень наступних аналогічних систем: Habitica, Way of Life, Break Bad Habit, Habit, Productive та HabitMate. Одну з аналогічних систем – Habitica вдалося ретельно дослідити технічні характеристики системи, а саме технології, які були покладені в основу даного програмного продукту. Також було досліджено та обґрунтовано вибір майбутніх засобів та мов програмування для розробки ПЗ інтелектуального аналізу даних для управління звичками людини. Насамкінець, була описана розгорнута постановка завдання, де прописані всі вимоги до реалізації ПЗ.

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Було описано функціонування системи, а саме які технології будуть використані, як клієнт буде взаємодіяти із сервером, протоколи та інші важливі технології. Було розроблено структурну схему, яка покладена в основу ПЗ, де прописані всі структурні зв'язки між компонентами, такими як: вебсайт, сервіс ІАД, месенджер та вебмодуль. На основі діаграми, що представляє структурну схему, було розроблено та описано функціональну схему, що описує детально функціональні зв'язки в ПЗ, а також функції/класи, що будуть використані. Насамкінець, для наочного представлення функціонування системи, було розроблена діаграму процесів, що описує процеси на всіх рівнях додатку.

Інтелектуальний аналіз даних був імплементований за рахунок бібліотеки `openai-php/laravel` та сервісу від OpenAI – Responses API. Перш за все бібліотека дала доступ до зручних класів-ресурсів таких як Responses що є обгорткою над API від OpenAI, що реалізована мовою програмування PHP та інтегрованої з фреймворком Laravel. Для отримання відповіді (response) від сервісу OpenAI в методі `HabitsService::analyze` було передано наступні параметри: назву моделі, системний промпт для моделі та нотатки користувача для цієї звички, таким чином модель має необхідні вхідні дані для формування відповіді.

Вдалося виконати цілу низку підготовчих кроків, для того, щоб налаштувати та впровадити систему ПЗ інтелектуального аналізу даних для управління звичками людини на одному з найбільш популярних хмарних сервісів – AWS.

Упровадження такого ПЗ надає змогу підвищити ефективність при впровадженні нових звичок та позбаванні шкідливих звичок, шляхом реалізованого інтелектуального аналізу даних за допомогою ШІ. Частина результатів цієї бакалаврської роботи апробована і схвалена на VII Міжнародній конференції молодих учених «Актуальні питання розвитку інформаційних технологій» (м. Дніпро, 19 листопада 2025 р.), а також опублікована[39].

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ДЖЕРЕЛ

1. Mriya.run. *Mriya.run*. URL: <https://mriya.run/post/zvicki> (дата звернення: 18.02.2026).
2. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ. URL: <https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6688/1049-58.pdf> (дата звернення: 18.02.2026).
3. Портрет ідеального клієнта: як правильно аналізувати аудиторію. *Школа бізнесу «Нова пошта»*. URL: <https://online.novaposhta.education/blog/portret-idealnogo-klienta-yak-pravilno-analizuvati-auditoriyu> (дата звернення: 20.02.2026).
4. Apple Developer Program. *Apple Developer*. URL: <https://developer.apple.com/programs/> (дата звернення: 21.02.2026).
5. Google Play. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Google_Play (дата звернення: 21.02.2026).
6. Telegram Messenger. *Telegram*. URL: <https://telegram.org/> (дата звернення: 23.02.2026).
7. Habitica – Gamify Your Life. *Habitica*. URL: <https://habitica.com/static/home> (дата звернення: 24.02.2026).
8. HabitRPG/habitica. *GitHub*. URL: <https://GitHub.com/HabitRPG/habitica> (дата звернення: 24.02.2026).
9. Express.js. *Wikipedia*. URL: <https://en.wikipedia.org/wiki/Express.js> (дата звернення: 24.02.2026).
10. MIT License. *Wikipedia*. URL: https://en.wikipedia.org/wiki/MIT_License (дата звернення: 24.02.2026).
11. What is an API communication protocol? *Sendbird*. URL: <https://sendbird.com/learn/what-is-an-api-communication-protocol> (дата звернення: 24.02.2026).

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		70

12. JSON. *JSON.org*. URL: <https://www.json.org/json-en.html> (дата звернення: 25.02.2026).
13. XML introduction. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/XML/Guides/XML_introduction (дата звернення: 25.02.2026).
14. Node.js. *Node.js*. URL: <https://nodejs.org/en> (дата звернення: 25.02.2026).
15. MongoDB. *MongoDB*. URL: <https://www.mongodb.com/> (дата звернення: 25.02.2026).
16. gulpjs/gulp. *GitHub*. URL: <https://GitHub.com/gulpjs/gulp> (дата звернення: 25.02.2026).
17. Git. *Git*. URL: <https://git-scm.com/> (дата звернення: 25.02.2026).
18. GitHub. *Wikipedia*. URL: <https://en.wikipedia.org/wiki/GitHub> (дата звернення: 25.02.2026).
19. Find and fix problems in your code – ESLint. *ESLint*. URL: <https://eslint.org/> (дата звернення: 25.02.2026).
20. PHP: Introduction – Manual. *PHP Manual*. URL: <https://www.php.net/manual/en/intro.swoole.php> (дата звернення: 26.02.2026).
21. Laravel – The Framework for Web Artisans. *Laravel*. URL: <https://laravel.com/> (дата звернення: 26.02.2026).
22. A brief history of JavaScript. *Deno*. URL: <https://deno.com/blog/history-of-javascript> (дата звернення: 26.02.2026).
23. React. *React*. URL: <https://react.dev/> (дата звернення: 26.02.2026).
24. PostgreSQL: The world's most advanced open source database. *PostgreSQL*. URL: <https://www.postgresql.org/> (дата звернення: 26.02.2026).
25. What is AWS? *Amazon Web Services*. URL: <https://aws.amazon.com/what-is-aws/> (дата звернення: 26.02.2026).
26. How to build a CI/CD pipeline with GitHub Actions in four simple steps. *GitHub Blog*. URL: <https://GitHub.blog/enterprise-software/ci-cd/build-ci-cd-pipeline-GitHub-actions-four-steps/> (дата звернення: 26.02.2026).

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

27. Way of Life – an elegant habit tracker that actually works! *Way of Life*. URL: <https://wayoflifeapp.com/> (дата звернення: 28.02.2026).

28. Break bad habit. *App Store*. URL: <https://apps.apple.com/us/app/bad-habit-break/id1658588991> (дата звернення: 01.03.2026).

29. Habit Tracker. *App Store*. URL: <https://apps.apple.com/us/app/habit-tracker/id1438388363> (дата звернення: 01.03.2026).

30. Пошук: мемо. *Англійсько-українські словники*. URL: <https://e2u.org.ua/s?w=memo&dicts=all&highlight=on> (дата звернення: 01.03.2026).

31. Пошук: streak. *Англійсько-українські словники*. URL: <https://e2u.org.ua/s?w=streak&dicts=all&highlight=on> (дата звернення: 03.03.2026).

32. Habit Tracker by Habit Mate. *App Store*. URL: <https://apps.apple.com/us/app/habit-tracker-by-habit-mate/id1541190629> (дата звернення: 03.03.2026).

33. Productive – Habit Tracker. *App Store*. URL: <https://apps.apple.com/us/app/productive-habit-tracker/id983826477> (дата звернення: 03.03.2026).

34. Request Lifecycle. *Laravel Documentation*. URL: <https://laravel.com/docs/12.x/lifecycle> (дата звернення: 10.03.2026).

35. Request–response. *Wikipedia*. URL: <https://en.wikipedia.org/wiki/Request%E2%80%93response> (дата звернення: 14.03.2026).

36. TCP. *Вікіпедія*. URL: <https://uk.wikipedia.org/wiki/TCP> (дата звернення: 14.03.2026).

37. JSON Web Token Authentication for Laravel & Lumen. *GitHub*. URL: <https://GitHub.com/tymondesigns/jwt-auth> (дата звернення: 14.03.2026).

38. JSON Web Token Introduction. *jwt.io*. URL: <https://www.jwt.io/introduction> (дата звернення: 16.03.2026).

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		72

39. Рудь І. В., Доренський О. П. Концептуальна модель системи інтелектуального аналізу даних для управління звичками людини. *Збірник тез VII Міжнародної конференції молодих учених «Актуальні питання розвитку інформаційних технологій»*. Дніпро, 19 листопада 2025 р. Дніпро : ДВНЗ «Приазовський державний технічний університет», 2025. С. 118–119. URL: <https://dspace.kntu.kr.ua/handle/123456789/17920>.

40. HTTP response status codes. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status> (дата звернення: 20.03.2026).

41. Eloquent: Mutators & Casting. *Laravel Documentation*. URL: <https://laravel.com/docs/12.x/eloquent-mutators#encrypted-casting> (дата звернення: 26.03.2026).

42. Universally unique identifier. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Universally_unique_identifier (дата звернення: 26.03.2026).

43. Encryption. *Laravel Documentation*. URL: <https://laravel.com/docs/12.x/encryption> (дата звернення: 12.04.2026).

44. Лекція 11. Сучасні методи шифрування (AES). *Система дистанційного навчання КНТУ*. URL: [moodle.kntu.kr.ua/pluginfile.php/24925/mod_resource/content/1/Лекція%20№%201.1.%20Сучасні%20симетричні%20алгоритми%20шифрування%20\(AES\).pdf](https://moodle.kntu.kr.ua/pluginfile.php/24925/mod_resource/content/1/Лекція%20№%201.1.%20Сучасні%20симетричні%20алгоритми%20шифрування%20(AES).pdf) (дата звернення: 12.04.2026).

45. Deploying a Laravel application to Elastic Beanstalk. URL: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/php-laravel-tutorial.html> (дата звернення 13.04.2026)

46. OpenAI API Platform Documentation. URL: <https://developers.openai.com/api/docs> (дата звернення 15.04.2026)

47. Laravel AI SDK. URL: <https://laravel.com/docs/13.x/ai-sdk> (дата звернення 15.04.2026)

					ВКРБ-122.26.0026.00.00.ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		