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
“____” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ковальов В. О.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є. В.
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

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

Ковальову Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів*

керівник роботи *Мелешко Єлизавета Владиславівна, д-р техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №13-02 від 05.01.2023 року

2. Строк подання студентом роботи до захисту *20.05.2023 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для авторизації з застосуванням QR-кодів*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

5. 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	*2 аркуша*

6. Дата видачі завдання « 05 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	22.05.2023 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ковальов В.О. Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначено для системи кібербезпеки автентифікації з застосуванням QR-кодів.

Метою роботи є створення системи кібербезпеки для автентифікації з застосуванням QR-кодів.

Результат роботи – програмна реалізація системи кібербезпеки для автентифікації з застосуванням QR-кодів.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11 та мобільних пристроях з ОС Android.

Програму розроблено на мові програмування JavaScript.

Ключові слова: кібербезпека, авторизація, QR-коди

ABSTRACT

Kovalov V.O. Cybersecurity system software for authentication using QR codes. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi 2023.

In this bachelor's qualification work, software that is designed for system for authentication using QR-codes was developed.

The purpose of the development is the software of the system for authentication using QR-codes.

The result of the work is the software implementation of the system for authentication using QR-codes.

In the process of working on a software model an analysis of existing hardware and software was performed. All components of the software developed are fully described.

A user-friendly interface is developed. The instructions for working with the software are provided.

The program can be used on an IBM PC running Windows 10/11 and mobile devices with Android OS.

The program was developed in the JavaScript programming language.

Keywords: cybersecurity, authorization, QR-codes

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

ОС – операційна система

QR-код – матричний код (двовимірний штрих-код), розроблений представлений японською компанією «Denso-Wave».

ВСТУП

Сьогодні, коли технології активно змінюють наше життя та бізнес-процеси, захист інформації є однією з найбільш актуальних проблем. У зв'язку з цим зростає значимість систем кібербезпеки, які забезпечують захист від несанкціонованого доступу до інформації та недозволеного використання даних. Одним з ефективних засобів забезпечення безпеки доступу до електронної інформації та облікових записів користувачів у різних комп'ютерних системах та мережах є системи автентифікації з використанням QR-кодів.

Програмне забезпечення для авторизації з використанням QR-кодів може бути використане в різних сферах, включаючи банківський сектор, електронну комерцію, системи електронного документообігу та інші області, де важливо забезпечити швидкий та безпечний доступ до ресурсів. За допомогою QR-кодів можна забезпечити захист від несанкціонованого доступу до конфіденційної інформації, а також від маніпулювання даними від імені іншої особи. Їх можна використовувати для аутентифікації, наприклад, на одному з етапів двохетапної аутентифікації, що прискорює процес, оскільки не треба вводити паролі або коди з SMS.

Мета й завдання дослідження. Метою роботи є розробка програмного забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– Дослідження існуючих методів автентифікації з застосуванням QR-кодів.

– Розробка алгоритмів для автентифікації з застосуванням QR-кодів.

– Програмна реалізація системи автентифікації з застосуванням QR-кодів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автентифікації з застосуванням QR-кодів. Отже, розробка та впровадження програмного забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів є актуальною задачею, яка потребує вирішення у цій кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів призначене для забезпечення безпечної автентифікації користувачів у різних сферах діяльності. QR-коди є одним з ефективних засобів передачі інформації, що дозволяє легко і швидко ввести дані у пристрій з камерою, такий як смартфон або планшет. Система кібербезпеки з QR-кодами може використовуватись для автентифікації користувачів у різних веб-сервісах, мобільних додатках, системах електронного голосування, електронних касах та інших системах, які потребують забезпечення високого рівня безпеки.

Основним призначенням програмного забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів є забезпечення захисту особистої інформації користувачів від зловмисників, які можуть намагатися здійснити несанкціонований доступ до системи. Програмне забезпечення використовує безпечні методи шифрування даних та автентифікації, що гарантує безпеку процесу автентифікації та унеможливлює несанкціонований доступ до інформації користувача.

Крім того, програмне забезпечення системи кібербезпеки з QR-кодами дозволяє забезпечити швидку і просту двохетапну авторизацію користувачів без необхідності запам’ятовування додаткових паролів та кодів з SMS. Користувач може легко отримати QR-код для автентифікації на своєму мобільному пристрої та сканувати його для входу в систему.

Крім того, система може надавати інформацію про час та місцезнаходження, де виконувалася авторизація з використанням QR-коду, що дозволяє відстежувати потенційні спроби несанкціонованого доступу до облікових записів користувачів та забезпечувати належний рівень кібербезпеки.

1.2 Область застосування

Розглянемо основні можливі сфери застосування програмного забезпечення системи кібербезпеки для автентифікації з використанням QR-кодів:

– *Банківський сектор*: розроблюване програмне забезпечення може бути використане для авторизації клієнтів в онлайн-банкінгу з застосуванням QR-кодів. Вони можуть бути згенеровані на стороні банку та надіслані на мобільний пристрій користувача, що забезпечує безпеку його даних при вході в систему онлайн-банкінгу.

– *Сфера торгівлі*: розроблюване програмне забезпечення може бути використане для авторизації користувачів на сайтах електронної комерції, що забезпечує захист конфіденційної інформації користувача та знижує ризик шахрайства.

– *Сфера логістики*: розроблюване програмне забезпечення може бути використане для авторизації користувачів на сайтах доставки товарів та забезпечення безпеки транзакцій, здійснених з використанням мобільних пристроїв.

– *Медична галузь*: розроблюване програмне забезпечення може бути використане для авторизації медичних працівників у системах електронної медичної інформації, забезпечуючи безпеку конфіденційної інформації пацієнтів та запобігаючи несанкціонованому доступу до даних.

– *Обмін даними*: розроблюване програмне забезпечення може бути використане в месенжерах для забезпечення безпеки обміну повідомленнями. Наприклад, додаток може забезпечити авторизацію користувача з використанням QR-кодів перед початком обміну повідомленнями з іншим користувачем. Додаток може забезпечити двофакторну аутентифікацію з використанням QR-кодів для підтвердження ідентичності користувача, що зменшить ризик несанкціонованого доступу до конфіденційних даних.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

QR код – це монохромна картинка, на якій деякі пристрої (наприклад, смартфон зі спеціальним додатком) розпізнають текст. Цим текстом може бути не лише проста фраза, а й, хоч це й не входить до офіційної специфікації, посилання, номера телефону чи візитної картки. Такі коди найчастіше використовують, щоб закодувати посилання та роздрукувати його на плакаті чи візитці.

Процес генерації QR-коду в загальному вигляді поділяється на такі кроки:

1. Аналіз даних – стандарт QR має 4 режими кодування тексту – числовий, буквено-цифровий, байтовий і кандзі. Кожен режим кодує текст як рядок бітів. Кожен режим використовує інший метод для перетворення тексту в біти, і кожен метод кодування оптимізований для кодування даних за допомогою найкоротшого можливого рядка бітів. Першим кроком є аналіз того, який режим найбільш оптимальний

2. Кодування даних – після вибору режиму кодування наступним кроком є кодування тексту в рядок бітів, який розділено на кодові слова даних, кожне з яких має довжину 8 бітів.

3. Кодування виправлення помилок – цей процес використовуватиме рядок бітів даних для генерації кодових слів виправлення помилок за допомогою виправлення помилок Ріда-Соломона. Це робиться для того, щоб сканер міг визначити, чи правильні дані, порівнюючи кодові слова даних і кодові слова виправлення помилок.

4. Структура (Остаточне повідомлення) – дані та кодові слова виправлення помилок розташовані в належному порядку в блоках, і ці блоки

мають відповідати специфікації QR-коду.

5. Розміщення модуля в матриці – цей крок розмістить порядок кроку 4 у матриці QR-коду певним чином. Алгоритм розмістить шаблони, які є загальними у всіх QR-кодах, наприклад прямокутники в трьох кутах, як це зазначено в специфікації матриці.

6. Маскування даних – деякі візерунки в матриці QR-коду ускладнюють читання коду, тому алгоритм визначатиме, які шаблони масок призводять до найменше небажаних рис, змінюючи QR-код відповідно до певного шаблону. Остаточною матрицею буде матриця з найменшим штрафним балом.

7. Інформація про формат і версію – це необов'язково, але інформація про версію додається до QR-коду. Пікселі формату містять таку інформацію, як керування версіями та шаблон маски.

Алгоритми генерації QR-кодів – це набори правил і кроків, які визначають, як перетворити дані на двовимірний штрих-код, що складається з чорних і білих квадратиків. Всі вони засновані на розглянутих вище етапах генерації QR-кодів. Існує кілька стандартів та специфікацій для QR-кодів, але найпоширенішим є стандарт ISO/IEC 18004:2015, який описує алгоритм генерації QR-коду в наступних етапах:

– Кодування даних: на цьому етапі вибирається тип даних, які будуть закодовані у QR-коді, і перетворюються в бітову послідовність за допомогою відповідної таблиці кодування. Наприклад, для URL-адреси можна використовувати побайтове кодування, а для тексту – літерно-цифрове або кодування коду.

– Додавання службової інформації та заповнення: на цьому етапі додається службова інформація до бітової послідовності, такої як індикатор режиму, кількість символів, індикатор версії та рівня корекції помилок. Ця інформація потрібна визначення параметрів QR-коду під час його зчитуванні. Також додаються спеціальні символи заповнення для доповнення бітової послідовності до потрібної довжини, яка залежить від версії QR-коду (від 1 до 40)

та рівня корекції помилок (від L до H).

– Поділ інформації на блоки: на цьому етапі бітова послідовність поділяється на блоки по 8 біт (байти), які будуть використовуватись для створення байтів корекції помилок. Кількість блоків залежить від версії та рівня корекції QR-коду.

– Створення байтів корекції помилок: цьому етапі кожного блоку даних створюється певну кількість байтів корекції помилок з допомогою алгоритму Ріда-Соломона. Ці байти дозволяють відновити дані у разі пошкодження QR-коду. Кількість байтів корекції також залежить від версії та рівня корекції QR-коду.

– Сформування остаточної бітової послідовності: цьому етапі з'єднуються блоки даних і блоки корекції у порядку. Ця послідовність буде матрицею з чорних і білих квадратиків - модулів QR-коду.

– Додавання стандартних елементів дизайну QR-коду: на цьому етапі додаються до матриці стандартні елементи дизайну QR-коду, такі як маркери позиціонування, роздільники, вирівнюючі маркери і маскувальні шаблони. Ці елементи необхідні для полегшення розпізнавання QR-коду сканером. Маскувальний шаблон застосовується для усунення небажаних шаблонів у матриці, таких як великі області одного кольору або діагональні лінії.

Читання даних QR-коду:

1. Користувач сканує QR-код камерою мобільного телефону.
2. Сканер камери телефону спочатку визначає три позиційні маркери в кутах зображення (кола).
3. Сканер починає сканування з квадрата в нижньому правому куті зображення (стрілка), індикатора режиму. Індикатор режиму – це група з чотирьох модулів даних, які повідомляють сканеру, який тип даних надсилається (числові, буквено-цифрові або байтові).
4. Потім він продовжує сканування до індикатора кількості символів, групи з восьми модулів даних, розташованих над індикатором режиму. Це

повідомляє про кількість символів, вбудованих у код.

5. Сканер продовжує роботу вздовж усіх модулів даних, завершуючи, коли він натискає індикатор кінця.

6. Продовжується сканування модулів виправлення помилок. Це математичні формули, які, по суті, допомагають створити резервну копію коду в разі пошкодження QR-коду.

Авторизація через QR-код

Аутентифікація за допомогою QR-коду – це тип автентифікації за допомогою телефону як маркера, який допомагає ідентифікувати користувача. Традиційні QR-коди допомагають організаціям реалізувати багатофакторну автентифікацію для своїх користувачів, використовуючи QR-код для генерації одноразового пароля (TOTP) на основі часу.

Паролі, однак, не є найбезпечнішим методом автентифікації, оскільки їх можна легко піддати фішингу для таких атак, як захоплення облікового запису. Їх також важко запам'ятати і це може зірвати роботу користувача.

Безпечність авторизації через QR-коди

Хоча самі QR-коди неможливо зламати, кіберзлочинці все частіше створюють підроблені QR-коди, які спрямовують користувачів на шкідливі веб-сайти, що призводить до шахрайства та крадіжки особистих даних. На жаль, виявити ці шахрайські QR-коди неможливо. Організації, які розглядають впровадження автентифікації за допомогою QR-коду, повинні розуміти ці ризики та використовувати надійне джерело для створення QR-коду, а не покладатися на безкоштовний онлайн-генератор. QR-коди, які використовують біометричну автентифікацію у своїй технології, також можуть зменшити ризики безпеки.

Плюси та мінуси автентифікації з QR-кодом

Автентифікація за допомогою QR-коду, особливо якщо вона здійснюється через надійне джерело, має ряд переваг.

Плюси автентифікації з QR-кодом

Кілька основних переваг автентифікації за допомогою QR-коду

включають:

– **Універсальність:** QR-коди можуть вміщувати багато типів і великі обсяги даних. Як наслідок, їх можна використовувати для широкого спектру продуктів у різних галузях промисловості.

– **Доступ до більшої кількості даних:** оскільки QR-коди усувають конфлікти, вони можуть бути ефективним способом для організацій збирати клієнтські та маркетингові дані. Потім вони можуть використовувати дані, наприклад, для вимірювання ефективності певних маркетингових кампаній.

– **Усунення незручностей:** QR-коди створюють зручний досвід для користувачів, які не хочуть відчувати незручності, пов’язані із запам’ятовуванням різних паролів.

Мінуси автентифікації з QR-кодом:

На додаток до згаданих проблем безпеки, QR-коди також мають інші недоліки.

– **Обмежено зручними для технологій користувачами та середовищами:** хоча QR-коди створюють зручність для багатьох користувачів, не всі користувачі завжди мають під рукою мобільний телефон або розуміють технологію достатньо добре, щоб швидко й ефективно ним користуватися. Вони також неефективні в місцях без надійного з’єднання Wi-Fi.

– **Конфіденційність клієнтів:** обмін величезними обсягами даних створює занепокоєння щодо конфіденційності клієнтів. Під час впровадження QR-технології організації повинні переконатися, що вони діють відповідно до правил збору даних, згоди та конфіденційності свого регіону.

Методи авторизації з QR-кодами

Існують різні способи підтвердження особистості або доступу до веб-ресурсів за допомогою QR-кодів, які містять унікальну інформацію про користувача або сесію.

Методи входу в браузер – ці методи дозволяють користувачеві увійти на веб-сайт або програму, просканувавши QR-код на екрані комп'ютера за

допомогою мобільного пристрою. Це зручно, коли користувач не хоче вводити логін та пароль на клавіатурі або коли він не має доступу до свого облікового запису на комп'ютері.

Методи двофакторної аутентифікації – ці методи дозволяють користувачеві підтвердити свою особу чи доступ до ресурсу, просканувавши QR-код, який містить одноразовий пароль або токен. Це підвищує безпеку, тому що для входу потрібне не тільки знання логіну та пароля, але й наявність мобільного пристрою з QR-кодом.

Методи оплати за QR-кодом: – ці методи дозволяють користувачеві оплатити товар або послугу, просканувавши QR-код, який містить інформацію про продавця, суму та спосіб оплати. Це зручно, коли користувач не хоче використовувати готівку чи банківські картки, а також для прискорення процесу оплати.

У цій роботі було досліджено та реалізовано саме метод двофакторної аутентифікації за допомогою QR-кодів.

Етапи двофакторної аутентифікації за допомогою QR-кодів:

1. Завантаження та встановлення додатоку-автентифікатора на мобільний пристрій.

2. Вхід до облікового запису користувача на веб-сайті або програмі, яка підтримує двофакторну автентифікацію за допомогою QR-кодів.

3. Перехід до параметрів безпеки або двофакторної автентифікації та вибір опції, що відповідає функції автентифікації через QR-код (у різних програмах назви можуть бути різними).

4. Сканування QR-коду, який з'явиться на екрані комп'ютера, за допомогою додатку-автентифікатора на мобільному пристрої. Це дозволить зв'язати програму та обліковий запис за допомогою секретного ключа.

5. Введення коду, який з'явиться в автентифікаторі, на веб-сайті або програмі для підтвердження налаштування двофакторної автентифікації.

6. Зберігання резервного коду або ключа, який надасть веб-сайт або

програма для відновлення доступу до облікового запису у разі втрати або зміни мобільного пристрою.

Огляд програм, що використовують QR-код для авторизації

Існує безліч програм, які використовують авторизацію за QR-кодом для різних цілей. Нижче розглянуті деякі з них.

WhatsApp Web – це веб-версія популярного месенджера WhatsApp, яка дозволяє обмінюватись повідомленнями та файлами з комп'ютера. Для входу до WhatsApp Web потрібно відкрити сайт https://web.whatsapp.com/ на комп'ютері та відсканувати QR-код з телефону, на якому встановлено програму WhatsApp. Таким чином, встановлюється зв'язок між телефоном та комп'ютером, і можна використовувати всі функції WhatsApp на великому екрані.

Google Authenticator – це програма для двофакторної аутентифікації, яка генерує одноразові коди для входу в різні сервіси Google та інші сайти. Для налаштування Google Authenticator потрібно відсканувати QR-код із сайту або програми, яка підтримує двофакторну автентифікацію. Таким чином, додається обліковий запис Google Authenticator, і можна отримувати коди для підтвердження особистості при вході.

Microsoft Authenticator – цей мобільний додаток від Microsoft пропонує аналогічні функції двофакторної автентифікації, що й Google Authenticator, і також використовує QR-коди для швидкої настройки.

Authy – це ще один популярний додаток для двофакторної автентифікації, який підтримує авторизацію за допомогою QR-кодів. Він сумісний з різними онлайн-платформами та послугами. Authy дозволяє пройти двофакторну автентифікацію за допомогою програми для пристроїв під керуванням iOS або Android, для ПК, а також за допомогою SMS-повідомлення.

LastPass Authenticator – це мобільний додаток, який пропонує двофакторну автентифікацію для користувачів LastPass, програми для зберігання паролів, розроблена компанією. Він використовує QR-коди для настройки авторизації на різних веб-сайтах.

Paytm – це індійський сервіс для електронних платежів та переказів грошей. Для оплати товарів або послуг за допомогою Paytm потрібно відсканувати QR-код продавця або постачальника послуг із телефону, на якому встановлено програму Paytm. Таким чином відбувається переказ грошей з балансу Paytm або прив'язаної банківської картки на рахунок одержувача.

Створення візирунків на QR-кодах

Створення візерунків у QR-кодах – це спосіб зміни зовнішнього вигляду QR-кодів за допомогою різних елементів дизайну, таких як кольори, форми, логотипи тощо. Мета цих алгоритмів – зробити QR-коди більш привабливими і такими, що запам'ятовуються для користувачів, не погіршуючи при цьому їх читаність і функціональність. Існує декілька типів алгоритмів створення візерунків у QR-кодах, наприклад:

– *Алгоритми зміни кольору* – ці алгоритми дозволяють вибирати колір фону та модулів QR-коду, а також застосовувати градієнти та тіні. При виборі кольору важливо враховувати контрастність та яскравість, щоб QR-код залишався читаним для сканерів.

– *Алгоритми зміни форми* – ці алгоритми дозволяють замінювати стандартні квадратні модулі QR-коду на інші геометричні фігури, такі як круги, трикутники, зірки тощо. При виборі форми важливо враховувати розмір та відстань між модулями, щоб QR-код залишався читаним для сканерів.

– *Алгоритми додавання логотипу* – ці алгоритми дозволяють вставляти зображення або текст у центр QR-коду, щоб збільшити його впізнаваність та прихильність до бренду. При додаванні логотипу важливо враховувати його розмір та прозорість, щоб не перекрити надто багато модулів QR-коду та не порушити його читання для сканерів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

При дослідженні найближчих програм-аналогів було визначено, що найкращим рішенням для автоматизації процесу обліку відвідуваності студентами занять є система, яка інтегрує у собі веб-орієнтоване програмне рішення та програмні рішення для мобільних платформ, тож програмне забезпечення для обліку відвідування занять студентами було вирішено реалізувати за допомогою мов програмування php та JavaScript, а в якості бази даних використати MySQL.

JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд Web-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ, всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманний «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

PHP (англ. PHP: Hypertext Preprocessor - PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools - скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP - проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконуються вже на стороні клієнта.

СУБД (MySQL) - програма, яка організує роботу з базами даних (на сервері). MySQL - вільна реляційна система управління базами даних. Розробку та підтримку MySQL здійснює корпорація Oracle. Продукт розповсюджується під GNU General Public License.

MySQL є ідеальним рішенням для малих і середніх веб-додатків. Входить до складу серверів WAMP, AppServ, LAMP і портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL постачається із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць.

WEB-сервер (Apache) - програма, яка оброблює запити браузерів. Web-сервер Apache є самостійним, некомерційним, вільно розповсюджуваним продуктом. Продукт підтримує безліч можливостей, багато з яких реалізовані як скомпільовані модулі, які розширюють основні функціональні можливості. Вони різняться від серверної підтримки мов програмування до схем аутентифікації. Існують інтерфейси для підтримки мов програмування Perl, Python, Tcl і PHP.

Популярні методи стискування на Apache включають зовнішній модуль mod_gzip, створений для зменшення розміру веб-сторінок, переданих по HTTP.

Функції віртуального хостингу дозволяють одній інсталяції Apache обслуговувати різні веб-сайти. Apache передусім використовується для передачі через HTTP статичних та динамічних Web-сторінок у всесвітній павутині. Багато Web-додатків спроектовано, зважаючи на середовище і можливості, які надає цей Web-сервер.

Apache зіграв ключову роль у початковому зростанні всесвітньої павутини, і продовжує бути найпопулярнішим у світі Web-сервером, де-факто платформою, на яку орієнтуються інші Web-сервери.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки авторизації з застосуванням QR-кодів.

В процесі розробки бакалаврської роботи необхідно виконати наступні за:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей, результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи, розробити функціональну та структурну схеми системи;

в) розробити алгоритми для реалізації програмного забезпечення стосовно теми роботи, побудувати блок-схеми алгоритмів програми та підпрограм;

г) розробити програмне забезпечення системи, що дозволить реалізувати задачу, поставлену технічним завданням задачу;

д) організувати інтерфейс користувача та обробку виключних ситуацій;

е) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

ж) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Використання QR-коду зробить програмний продукт набагато ефективним та зручним.

У наш час QR-код можна зустріти на рекламних щитах, на вулицях міста, в магазинах, на упаковках продуктів харчування та на будь-якій іншій продукції. Тому що в QR-коді може бути різноманітна інформація: літери, цифри, знаки. У цій технології є різні варіанти зміни тексту. Завдяки цьому факту пристрій, який сканує QR-код, безпомилково визначає інформацію, що зберігає. Найбільш поширені формати, які можуть зберігатися в QR-коді:

- Інтернет-адреса. QR-код, що зберігає посилання на інтернет-ресурс найчастіше знаходиться в газетах і на рекламних оголошеннях. Після розпізнавання даного коду пристрій, що зчитує, відкриє користувачеві сайт. Тим самим позбавляючи користувача ретельно вводити інтернет-адресу в рядок браузера. Сайт може містити, наприклад, інформацію про будь-який продукт або більшу інформацію про будь-яку статтю.

- Контактні дані. QR-код, який містить ту чи іншу контактну інформацію, найчастіше можна побачити на візитках. Просканувавши QR-код із цією інформацією, користувач може зберегти лічені дані на мобільний пристрій. Таким чином, користувач заощаджує час на введення контактних даних.

- Адреса електронної пошти. QR-код може містити адресу електронної пошти та ім’я адресата. Вважаючи QR-код з такою інформацією користувачеві не знадобиться вручну, вводити адресу, щоб надіслати електронний лист, а треба після зчитування QR-коду натиснути на одну кнопку і лист буде надіслано.

- SMS. Часто для участі в акції слід надіслати SMS. QR-код позбавить необхідності набору номера телефону і самого тексту повідомлення. Необхідно

рахувати QR-код і на дисплеї телефону побачите готове повідомлення, після чого слід натиснути кнопку «Надіслати».

- Геодані. QR-код може містити геодані. Це дозволяє переглянути розташування об’єкта, що цікавить, на карті, наприклад, в "Картах Google"".

- Текст. QR-код, який зберігає текст, може бути придатний для різних цілей. Наприклад, для зберігання вірша.

- Телефонні номери. QR-код може зберігати номер телефону. При скануванні такої інформації користувач може зберегти номер телефону, а також відразу зробити дзвінок, якщо програма-сканер дозволяє.

Усього існує 40 версій QR-коду. Перша має розмір 21x21 пікселів, кожна наступна на 4 більше. Максимальний розмір QR-коду: 177x177 пікселів. "Інформація в QR-коді розташовується у двох напрямках - як по горизонталі, так і по вертикалі, у той час як у штрих-коді інформація розміщується тільки в одному напрямку. На відміну від звичайних штрих-кодів, що зберігають максимум 20 символів, QR -код здатний зберігати в багато разів більше інформації. Це практично всі типи даних: цифрові та літерні знаки, кілька різновидів ієрогліфів, символи і т.д ".

Максимальна кількість символів, які розміщуються в один QR-код:

- цифри – 7089;

- цифри та літери (включаючи кирилицю) – 4296;

- двійковий код – 2953 байт;

- ієрогліфи – 1817.

Ще одна перевага QR-коду - його здатність відновлювати інформацію, що міститься в ньому. Навіть якщо символ частково забруднений або пошкоджений, за допомогою системи корекції помилок на базі кодів Ріда-Соломона , який розглянутий нижче, відновлення підлягає до 30% кодових слів. У таблиці 3.1 значення якості корекції помилок QR-коду в залежності від рівня.

Таблиця 3.1 – Якість корекції помилок QR-коду

Рівень L	~ 7%
Рівень M	~ 15%
Рівень Q	~ 25%
Рівень H	~ 30%

Будь-який кодовий символ QR будується з номінально квадратних модулів, що знаходяться в квадратній множині, і що складаються з областей: quiet zona (тиха зона), position detection patterns (зразки виявлення положення), separators for position detection patterns (сепаратори для зразків виявлено положення), timing patterns (тимчасові шаблони), alignment patterns (вирівнюючі шаблони), format information (інформація про формат), versionin formation (інформація про версію), data and error correction code words (інформація та кодові слова для виправлення помилок).

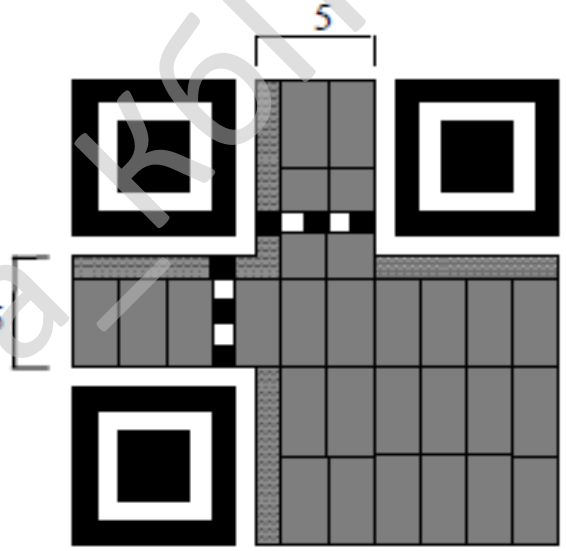

Рисунок 3.3 – Перша версія в 21 модуль

Існує сорок розмірів кодового символу QR, званого Версією 1, Версією 2... Версія 40. Версія 1 вимірює 21 модуль x 21 модуль, Версія 2 вимірює 25 модулів x 25 модулів, і так далі збільшуються в кроках 4 модулів за бік до Версії 40 , Що

вимірює 177 модулів х 177 модулів.

На рисунку 3.3-3.8 наочно показаний матричний спосіб розбиття картинки коду. Блоки необов'язково повинні бути квадратними, у версіях вищого рівня мають ще більш хитромудру форму і їх щільність збільшується. Розташування блоків з даними може змінюватись один відносно один. Щоб розподілити блоки з даними рівномірно по полю, використовуються маски.

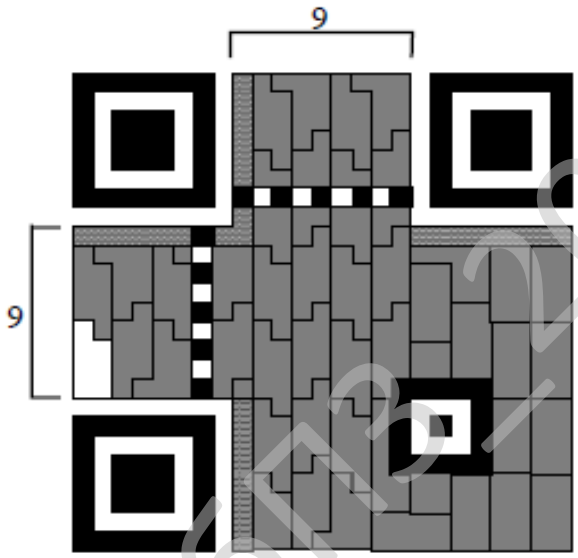

Рисунок 3.4 – Друга версія в 25 модулів

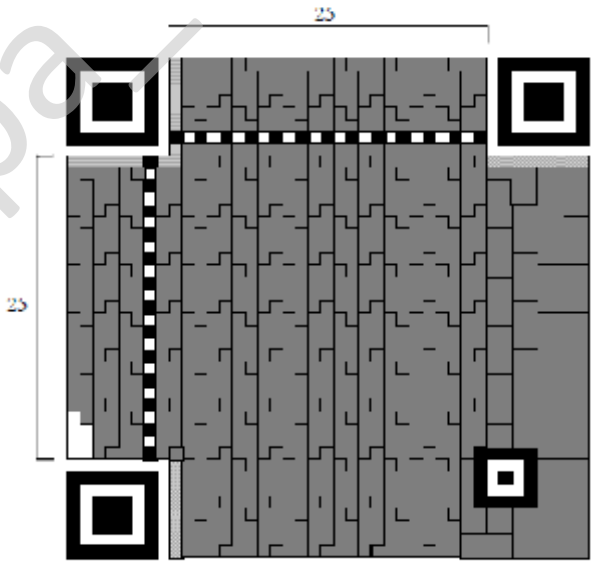

Рисунок 3.5 – Шоста версія в 41 модуль

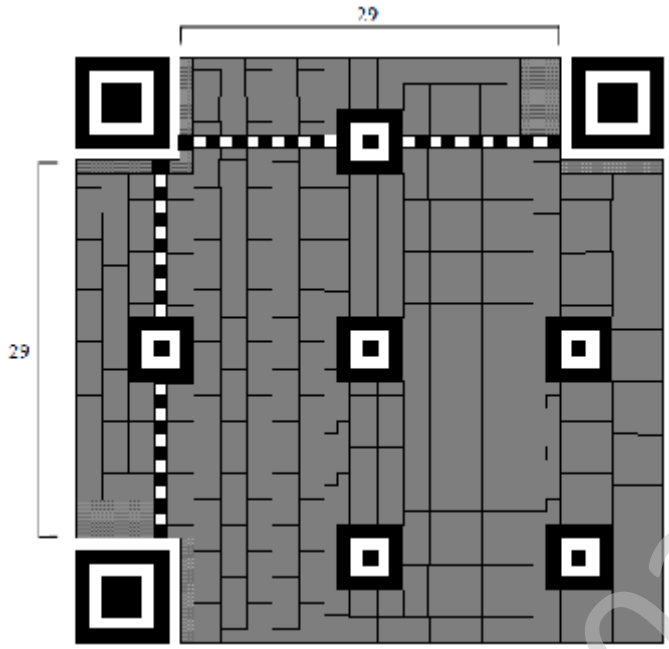

Рисунок 3.6 – Сьома версія в 45 модулів

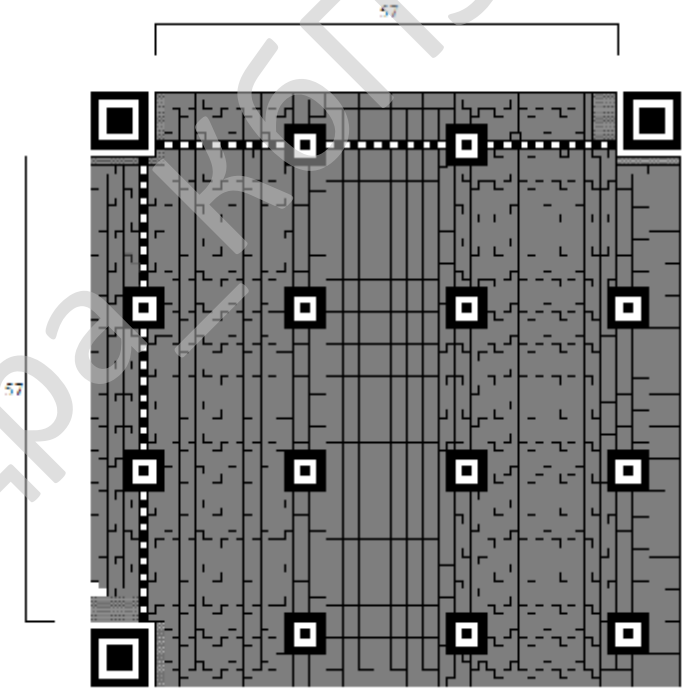

Рисунок 3.7 – Чотирнадцята версія в 73 модулі

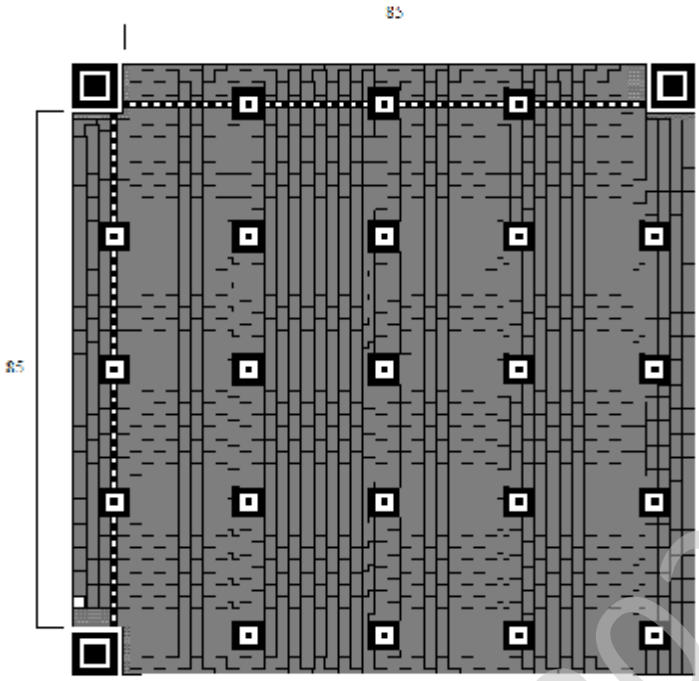

Рисунок 3.8 – Двадцять перша версія в 101 модуль

Для зчитування закладеної інформації в QR-коді він будується за певним шаблоном. Розглянемо його складові докладніше.

Шаблон пошуку – Finder pattern. Шаблон пошуку (зразок шукача) має складатися із трьох ідентичних «Зразків Виявлення Положення». Вони розташовуються у верхньому лівому, верхньому правому та нижньому лівому кутах символу як показано на рис. 3.3-3.8. Вони можуть розглядатися як три додані квадрати, і побудовані з трьох модулів: темний 7x7, світлий 5x5 і темний 3x3. Відношення ширин модуля 1:1:3:1:1, як показано на рисунку 9. Ідентифікація трьох «Зразків Виявлення Положення» однозначно визначає місцезнаходження та орієнтацію символу у полі зору.

Вирівнюючий шаблон – Alignment pattern. Кожен шаблон вирівнювання або напряму складається з трьох квадратів, розташованих один в одному: темного розмірність 5x5 модулів, світлого 3x3 та одного центрального темного. Цей шаблон використовується, починаючи з другої версії (що вища версія, тим вирівнюючих шаблонів більше).

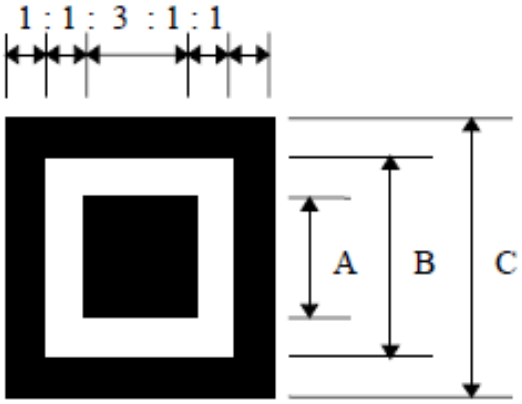

Рисунок 3.9 – Структура шаблону пошуку
(A – 3 модулі, B – 5 модулів, C – 7 модулів)

Шаблон QR-коду, що вирівнює, допомагає при декодуванні. Декодер за допомогою цих шаблонів перетворює перекошений QR-код у віртуальну сітку даних. На рис. 3.10 зображено зчитування QR-коду, де 1,2,3 - шаблон пошуку (Finder pattern), а 4 - шаблон, що вирівнює (Alignment pattern).

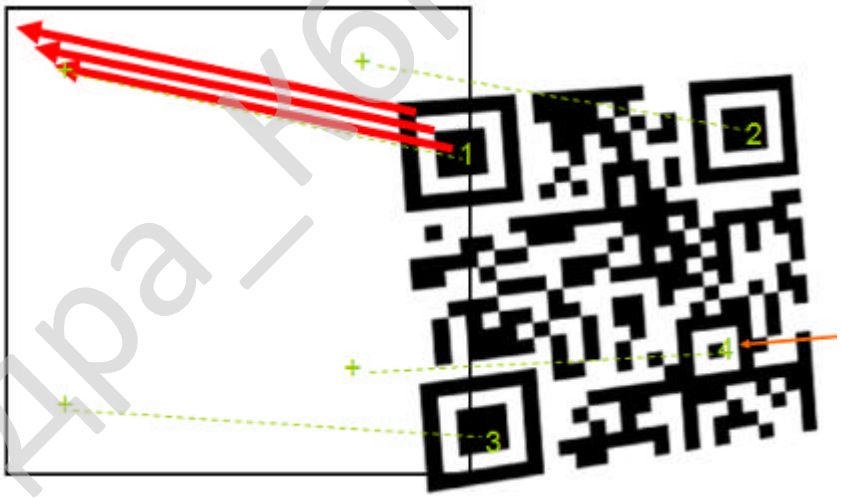

Рисунок 3.10 – Зчитування QR-коду, де 1,2,3 - шаблон пошуку (Finder pattern), а 4 - шаблон, що вирівнює (Alignment pattern)

Рисунок 3.11 - Структури символу

Різний колір має різну функціональність, а саме: зелений – тиха зона, помаранчевий – інформація о версії, червоний – інформація о форматі, сірий – інформація та кодові слова для виправлення помилок, фіолетовий – тимчасові шаблони та чорний - шаблон пошуку та вирівнювання.

Таблиця 3.2 – Комбінація масок

Маска	**Стан**
000	(i+ j) mod 2 = 0
001	i mod 2 = 0
010	j mod 3 = 0
011	(i + j) mod 3 = 0
100	((i div 2) +(j div 3)) mod 2 = 0
101	(i j) mod 2 +(i j) mod 3 = 0
110	((i j) mod 2 +(i j) mod 3) mod 2 = 0
111	((i j) mod 3 + (i + j) mod 3) mod 2 = 0

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розроблюваної системи.

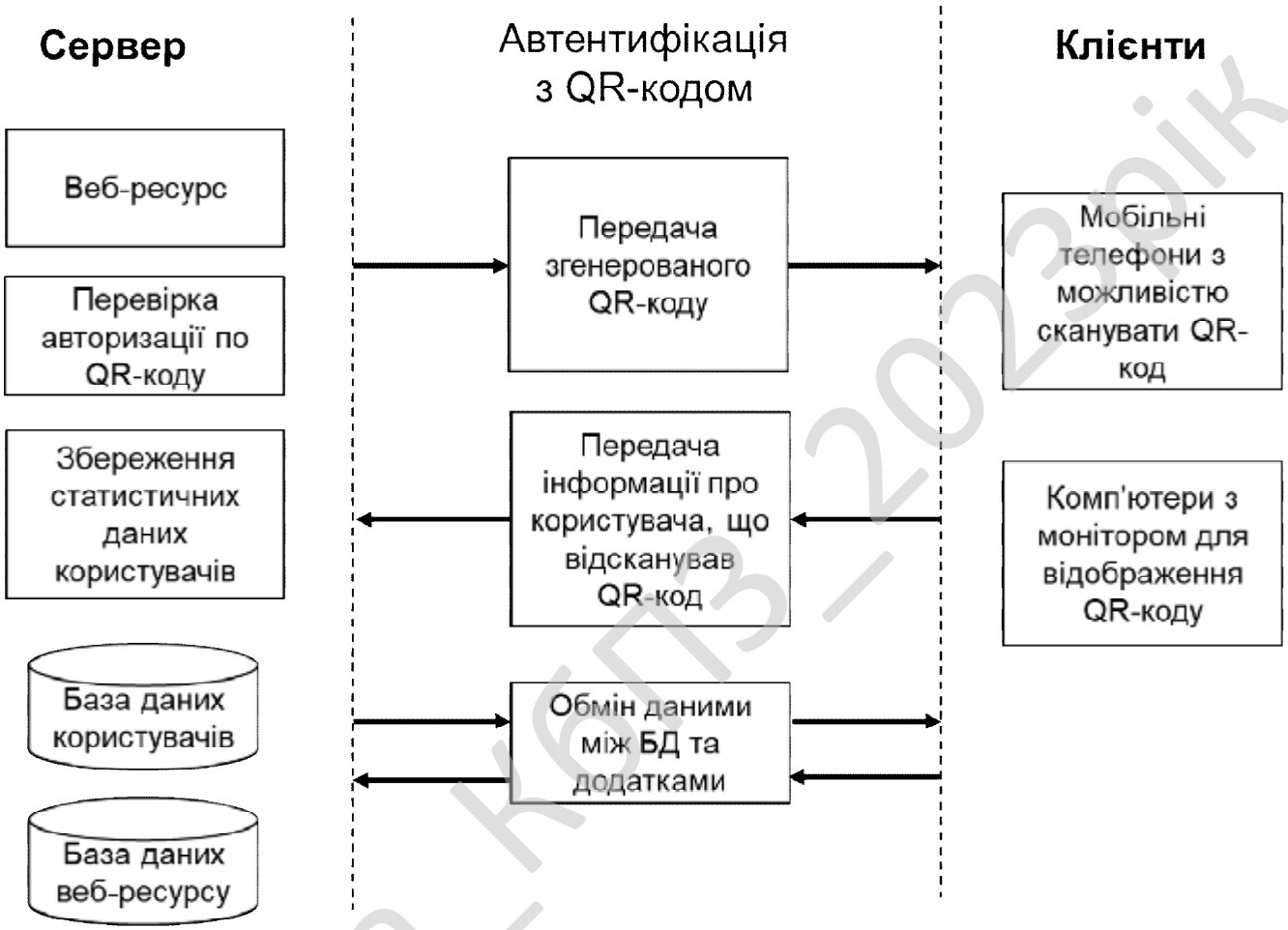

Рисунок 3.1 – Структурна схема системи

Розглянемо компоненти структурної схеми. Вона складається з серверного та клієнтських додатків та процесу авторищації з QR-кодом.

Блок серверу складається з наступних елементів:

– Веб-ресурс;

– Перевірка авторизації по QR-коду;

– Збереження статистичних даних користувачів;

– База даних користувачів;

– База даних веб ресурсів.

Блок клієнтів складається з наступних елементів:

– Мобільні телефони з можливістю сканувати QR-коди;

– Комп’ютери з монітором для відображення QR-коду.

Блок процесу авторизації з QR-кодами складається з таких елементів:

– Передача згенерованого QR-коду;

– Передача інформації про користувача, що відсканував QR-код;

– Обмін даними між БД та додатками.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема розроблюваної системи.

Рисунок 3.2 – Функціональна схема системи

Як видно з рисунку, розроблювана системи складається з таких основних частин:

1. Серверна частина програмного забезпечення.

2. Клієнтська частина програмного забезпечення.

До серверної частини програмного забезпечення відносяться наступні модулі:

1. Модуль реєстрації та авторизації користувачів адміністратором.

2. Модуль надання користувачу послуг веб-ресурсу.

3. Модуль роботи з базою даних користувачів.

4. Модуль роботи з базою даних веб-ресурсу.

5. Модуль роботи із статистичними даними користувачів.

До клієнтської частини програмного забезпечення відносяться наступні модулі:

1. Модуль реєстрації та авторизації користувача.

2. Модуль роботи користувача у системі.

У модулі роботи користувача у системі наяні наступні функції:

– Функції наявні для мобільного додатку.

– Функції наявні для додатку ПК.

Функції наявні для мобільного додатку реалізують наступні можливості:

– Авторизація з логіном та паролем.

– Сканування QR-коду з монітору комп’ютера.

– Відправка даних на сервер.

Функції наявні для додатку ПК реалізують наступні можливості:

– Запит на авторизацію.

– Одержання та відображення QR-коду.

– Отримання доступу до веб-ресурсу.

3.4 Розробка діаграми процесів

На рисунку 3.3 зображена діаграма процесів розроблюваної системи для серверної частини програмного забезпечення.

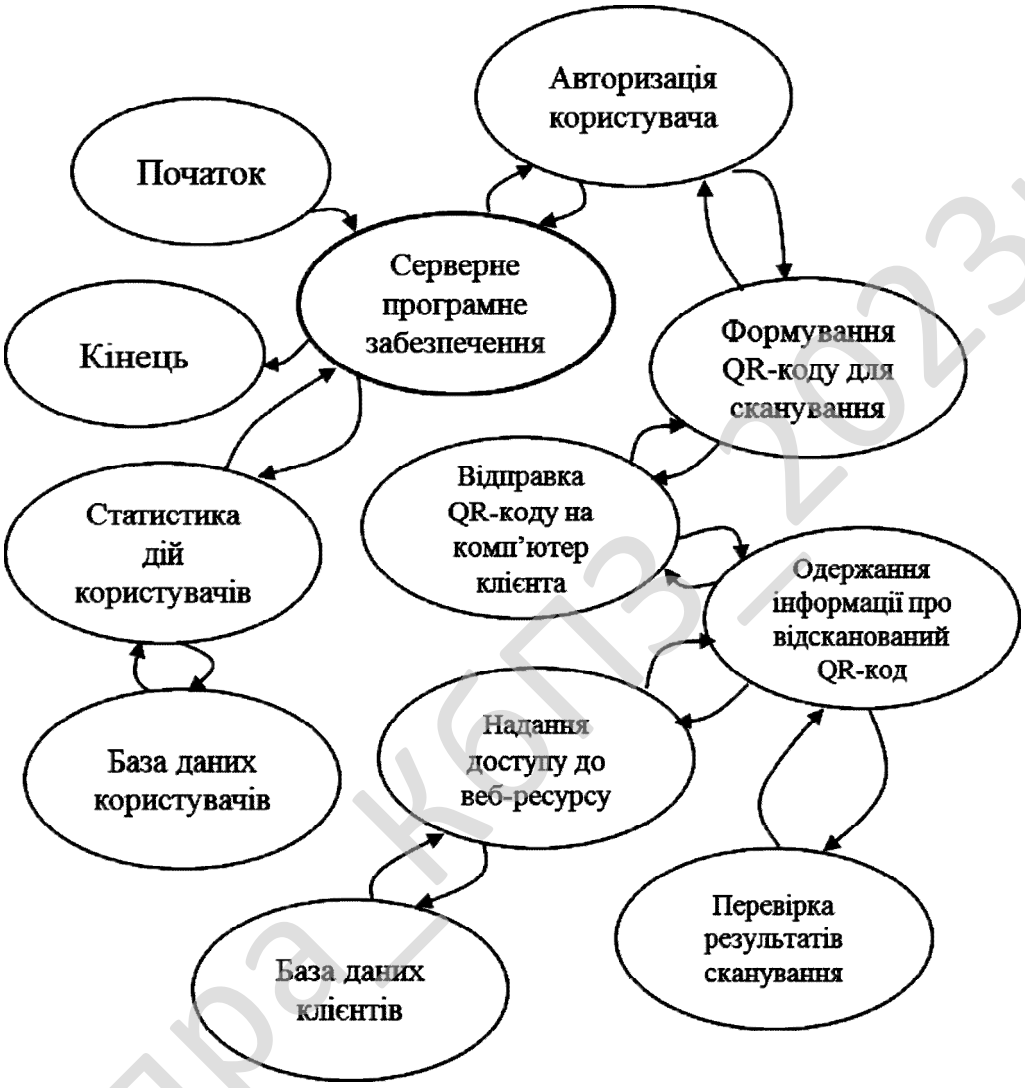

Рисунок 3.3 – Діаграма процесів системи для серверного додатку

Діаграма процесів системи для серверного додатку складається з таких процесів:

– Авторизація користувача.

– Формування QR-коду для сканування.

– Відправка QR-коду на комп'ютер клієнта.

– Одержання інформації про відсканований QR-код.

– Перевірка результатів сканування.

– Надання доступу до веб-ресурсу.

– База даних клієнтів.

– Статистика дій користувачів.

– База даних користувачів.

На рисунку 3.4 зображена діаграма процесів розроблюваної системи для серверної частини програмного забезпечення.

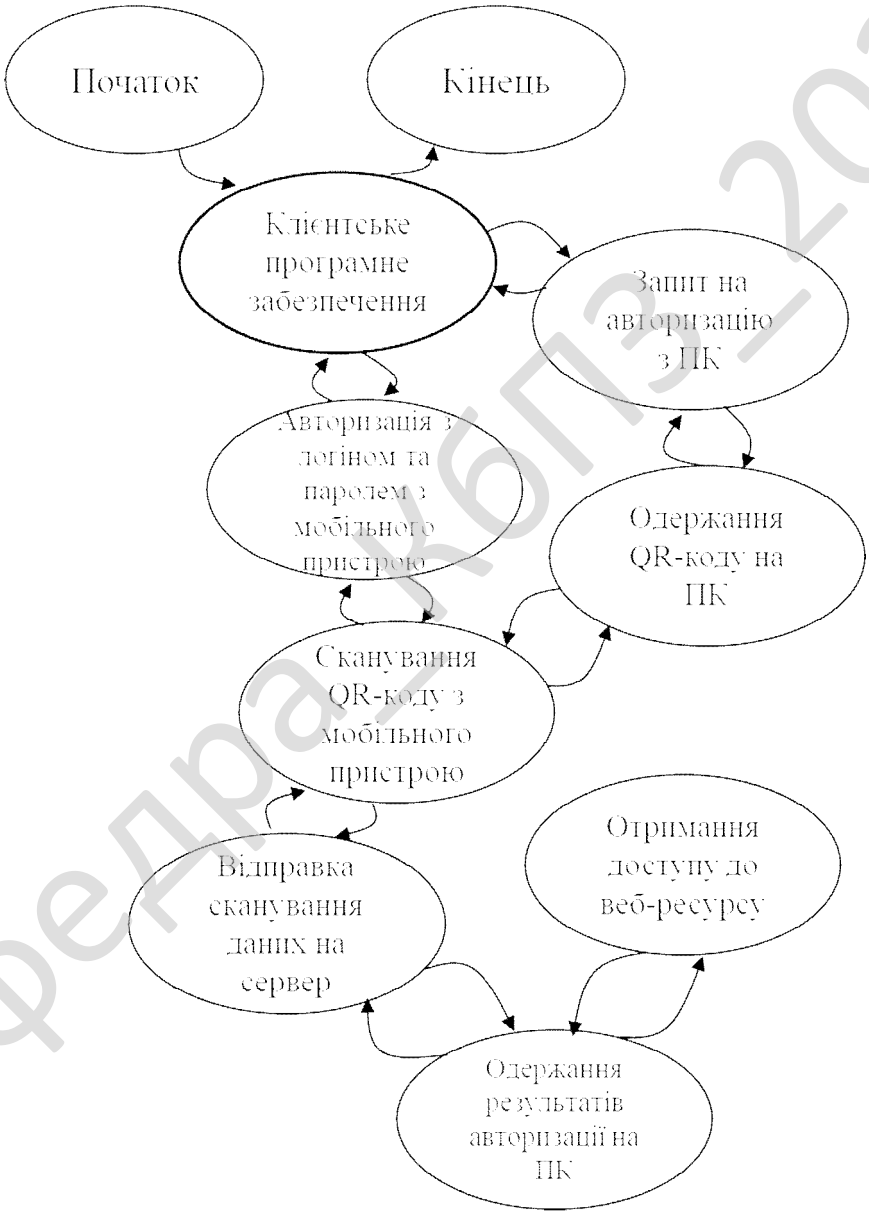

Рисунок 3.4 – Діаграма процесів системи для клієнтського додатку

Діаграма процесів системи для клієнтського додатку складається з таких процесів:

– Запит на авторизацію з ПК.
– Одержання QR-коду на ПК.
– Авторизація з логіном та паролем з мобільного пристрою.
– Сканування QR-коду з мобільного пристрою.
– Відправка сканування даних на сервер.
– Одержання результатів авторизації на ПК.
– Отримання доступу до веб-ресурсу.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.4 зображена блок-схема роботи основної програми.

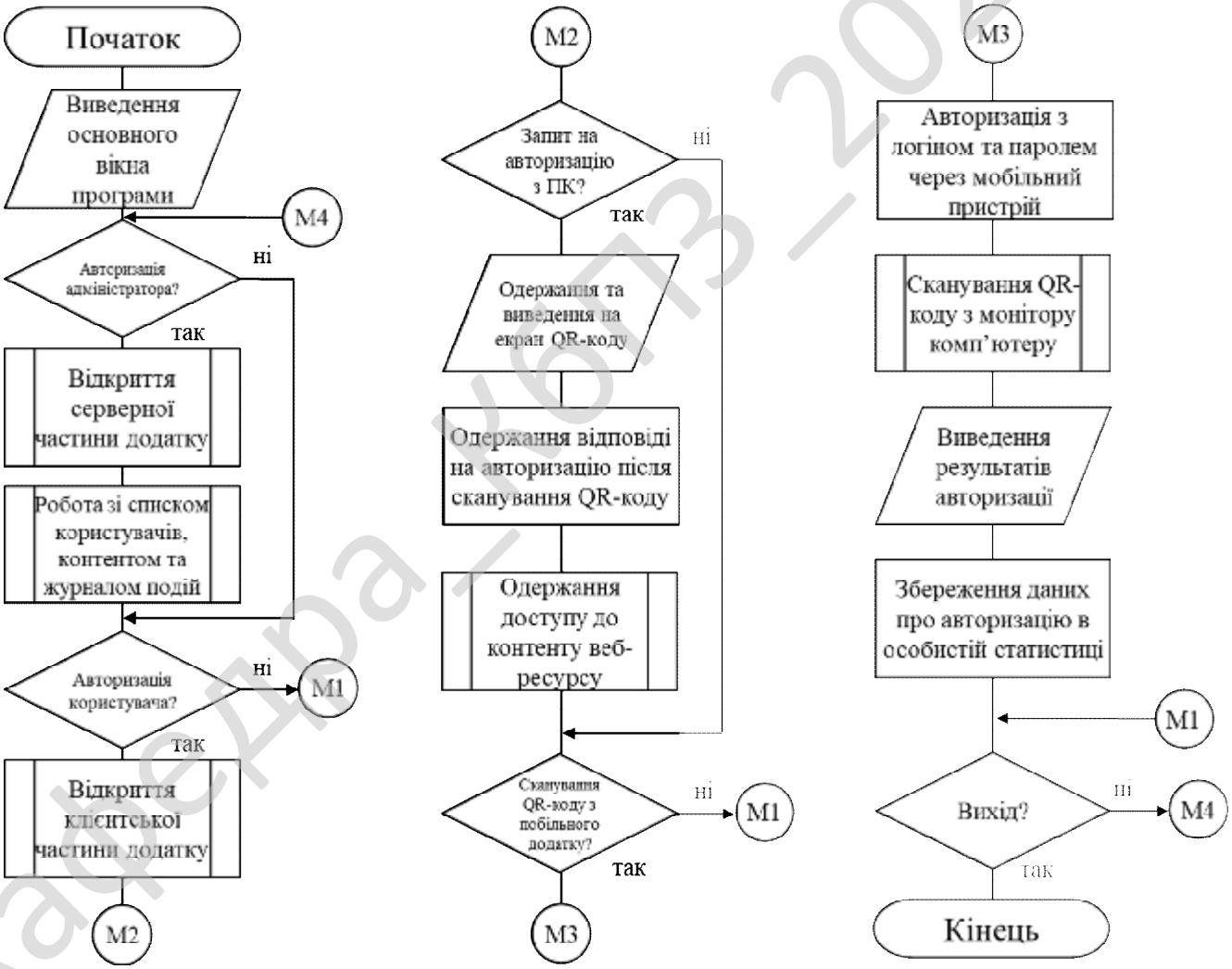

Рисунок 4.1 – Блок-схема роботи основної програми

Як видно з рисунку, для роботи програми потрібні серверний та клієнтські додатки. Сервер створює QR-коди для двофакторної авторизації.

QR-код предсталяється матричним кодом – двовимірний штрих-код. Основна перевага QR-коду – це легке розпізнавання сканувальним обладнанням (в тому числі й фотокамерою мобільного телефона), що дає можливість використання в торгівлі, на виробництві, в логістиці.

На відміну від старого штрих-коду, який сканують тонким променем, QR-код визначається сенсором як двовимірне зображення. Три квадрати в кутах зображення та менші синхронізувальні квадратики по всьому коду дозволяють нормалізувати розмір зображення і його орієнтацію, а також кут, під яким сенсор розташований до поверхні зображення. Точки переводяться в двійкові числа з перевіркою контрольних сум.

Побудова матриці QR-коду – це етап алгоритму генерації QR-коду, на якому створюється двовимірне зображення QR-коду з бітової послідовності. Для побудови матриці QR-коду виконуються такі дії:

– Створюється пуста матриця QR-коду заданого розміру. Розмір матриці залежить від версії QR-коду і дорівнює 17+4* версія модулів по кожній стороні. Наприклад, для версії 1 розмір матриці дорівнює 21 х 21 модулів, а для версії 40 - 177 х 177 модулів.

– У кутах матриці малюються маркери, що позиціонують, – квадрати з чорних і білих модулів, які служать для визначення положення та орієнтації QR-коду при скануванні. Позиціонують маркери мають розмір 7 х 7 модулів і оточені білим роздільником шириною один модуль.

– Навколо маркерів, що позиціонують, малюються маркери вирівнювання – менші квадрати з чорних і білих модулів, які служать для компенсації спотворень QR-коду при скануванні. Маркери вирівнювання мають розмір 5 х 5 модулів та оточені білим роздільником шириною в один модуль. Кількість та розташування маркерів вирівнювання залежать від версії QR-коду та задаються спеціальною таблицею.

– У правому нижньому куті матриці малюється маркер версії – прямокутник з чорних і білих модулів, який містить інформацію про версію QR-коду. Маркер версії має розмір 6 х 3 модулів і дублюється з двох сторін від маркера, що позиціонує. Маркер версії є тільки в QR-кодах версії 7 і вище.

– Уздовж меж матриці малюються маркери формату – лінії з чорних та білих модулів, які містять інформацію про рівень надійності та обрану маску QR-коду. Маркери формату мають довжину в 15 модулів і дублюються з двох сторін від маркерів, що позиціонують. Маркери формату захищені від помилок кодом БЧХ.

– У вільні осередки матриці записуються біти даних та корекції помилок по спіральному порядку праворуч наліво. При цьому пропускаються осередки, зайняті позиціонуючими маркерами, роздільниками, маркерами вирівнювання, маркерами версії та маркерами формату.

– Після запису всіх бітів даних та корекції помилок застосовується маскування – інвертування кольору деяких модулів залежно від їх координат. Мета маскування – зменшити складність зображення QR-коду та уникнення.

Найменший QR-код (версія 1) має розмір 21 × 21 піксель (без урахування полів), найбільший (версія 40) — 177 × 177 пікселів.

Максимальна кількість символів, які вміщаються в один QR-код:

– Цифри – 7089.

– Цифри і букви (включаючи кирилицю) – 4296.

– Двійковий код – 2953 байт.

– Ієрогліфи – 1817.

Також існують «псевдокодування»: завдання методу кодування в даних, розбиття довгого повідомлення на кілька кодів тощо.

Для виправлення помилок застосовується код Ріда-Соломона з 8-бітним кодовим словом. Є чотири рівня надмірності: 7, 15, 25 і 30 %. Завдяки виправленню помилок вдається нанести на QR-код малюнок і все одно залишити його читабельним.

Щоб у коді не було елементів, здатних заплутати сканер, область даних XOR'иться зі спеціальною маскою. Коректно працюючий кодер повинен перебрати всі варіанти масок, порахувати штрафні очки для кожної області та вибрати найбільш вдалу.

На рисунку 4.2 наведено блок-схеми читання QR-коду.

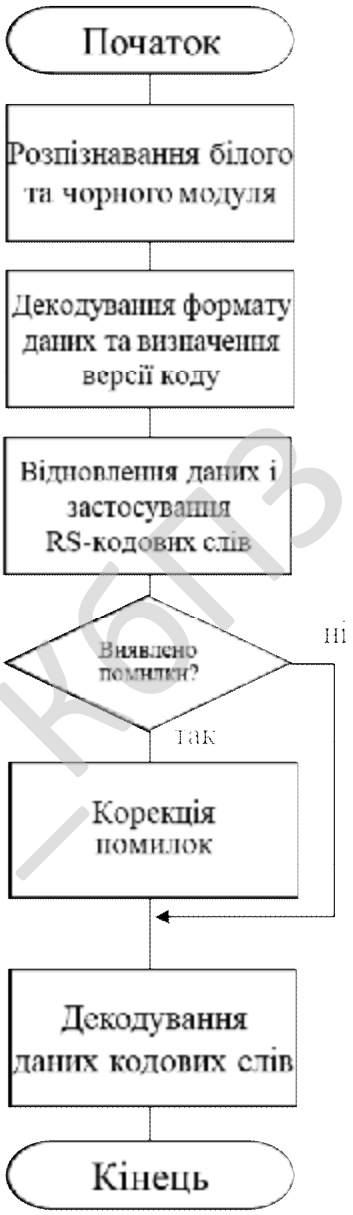

Рисунок 4.2 – Блок-схема підпрограми сканування QR-коду

Сканування QR-коду – це спосіб розпізнавання та декодування даних з двовимірного коду, що складається з чорних та білих модулів. Для сканування QR-коду необхідно мати камеру та спеціальну програму на мобільному пристрої. Для сканування QR-коду виконуються такі дії:

– Відкриється камера або програма для сканування QR-кодів.

– Наводиться камера на QR-код таким чином, щоб він сфокусувався на центрі екрана. Якщо фронтальна камера увімкнена, спочатку перемикається на основну камеру.

– Коректування положення камери. Усі чотири кути QR-коду мають бути видні на екрані.

– Очікування, коли код буде відскановано. Якщо QR-код повністю відображається на екрані, його сканування розпочнеться негайно.

– Зчитується вміст QR-коду.

Для декодування QR-коду виконуються зворотні операції алгоритму генерації QR-коду, наприклад:

– Зчитується інформація про формат та версію QR-коду.

– Застосовується зворотне маскування – інвертування кольору деяких модулів залежно від координат і обраної маски QR-коду.

– Зчитуються біти даних та корекції помилок із вільних осередків матриці QR-коду по спіральному порядку зліва направо. При цьому пропускаються осередки, зайняті позиціонуючими маркерами, роздільниками, маркерами вирівнювання, маркерами версії та маркерами формату.

– Відновлюються біти даних з бітів корекції помилок.

– Збираються блоки даних та корекції помилок в одну бітову послідовність відповідно до рівня надійності QR-коду.

– Видаляються біти службової інформації та заповнення з бітової послідовності.

– Декодуються дані з бітової послідовності за допомогою одного із чотирьох режимів: цифрового, літерно-цифрового, побайтового або кандзі.

Зчитуються службові біти, такі як індикатор режиму, індикатор довжини та термінатор.

Розглянемо реалізацію деяких з найбільш важливих модулів системи:

Модуль роботи з QR-кодами:

```
import React, {Component} from 'react';
import QrReader from 'react-qr-reader';
import {Container, Button, Icon, Message} from 'semantic-ui-react';

export default class MainPage extends Component {
  constructor(props) {
    super(props);

    this.state = {
      showQR: false,
      delay: 300,
      result: '',
    };
  }

  triggerQr() {
    const {showQR} = this.state;

    this.setState({
      showQR: !showQR,
    });
  };

  handleError(err) {
    console.error(err);
  }

  handleScan(data) {
    if (data) {
      this.setState({
        result: data,
        showQR: false,
      });
    }
  }

  renderResult(result) {
    return (
      <Message positive>
        <Message.Header>QR was read</Message.Header>
        <p>
          Go to <b><a href={result}>{result}</a></b>.
        </p>
      </Message>
    );
  }

  render() {
    const {showQR, delay, result} = this.state;

    return (
      <Container className='center aligned' fluid>
        <Button
          onClick={this.triggerQr.bind(this)}
```

```
          color='teal' icon='qrcode'
          size='massive'
          content='Scan QR code'/>

        {result ? this.renderResult(result) : null}
        {showQR ? <QrReader
          className='qr-reader'
          delay={delay}
          onError={this.handleError.bind(this)}
          onScan={this.handleScan.bind(this)}
        /> : null}
      </Container>
    );
  }
}
```

Модуль реєстрації користувача:

```
import React, {Component} from 'react';
import {connect} from 'react-redux';
import {Button, Form, Grid, Message, Segment} from 'semantic-ui-react';
import * as authActions from '@/client/store/actions/auth.actions';
import {Logo} from '../assets';

class LoginForm extends Component {

  state = {
    data: {
      email: null,
      password: null,
    },
    errors: {},
  };

  handleSubmit = async (e) => {
    e.preventDefault();

    const {login} = this.props;

    await this.setStateAsync({loading: true});

    const resp = await login(this.state.data);
    const errors = {};

    if (resp.err) {
      if (resp.err.message) {
        errors.common = resp.err.message;
```

```
        }
        // multiple errors
        else if (resp.body.details) {
          resp.body.details.forEach(err => {
            errors[err.context.key] = err.message;
          });
        }
        this.setState({errors});
      } else {
        // TODO: redirect
      }

      this.setState({errors, loading: false});
    };

    handleInputChange = e => {
      this.setState({
        data: {
          ...this.state.data,
          [e.target.name]: e.target.value,
        },
      });
    };

    setStateAsync(state) {
      return new Promise((resolve) => {
        this.setState(state, resolve);
      });
    }

    render() {

      const {loading} = this.props;
      const {data, errors} = this.state;
      const     errorMessage     =     Object.entries(errors).map(err     =>
err[1]).join('\n');

      return (
        <div className='login-form'>
          <style>{`
        body > div,
        body > div > div,
```

```
            body > div > div > div.login-form {
              height: 100%;
            }
          `}</style>

            <Grid textAlign='center' style={{height: '100%'}}
                  verticalAlign='middle'>
              <Grid.Column style={{maxWidth: 450}}>

                <h2>
                  <img className="logo" src={Logo} alt=""/>
                </h2>

                {errorMessage && <Message error content={errorMessage}/>}

                <Form              size='large'              loading={loading}
onSubmit={this.handleSubmit}
                      error={errors.length > 0}>
                  <Segment stacked>
                    <Form.Input
                      fluid
                      icon='user'
                      iconPosition='left'
                      placeholder='E-mail address'
                      name='email'
                      defaultValue={data.email}
                      error={!!errors.email}
                      onChange={this.handleInputChange}
                    />
                    <Form.Input
                      fluid
                      icon='lock'
                      iconPosition='left'
                      placeholder='Password'
                      type='password'
                      name='password'
                      defaultValue={data.password}
                      error={!!errors.password}
                      onChange={this.handleInputChange}
                    />
                    <Button color='teal' fluid size='large'>
                      Login
```

```
                </Button>
              </Segment>

            </Form>
          </Grid.Column>
        </Grid>
      </div>
    );
  }
}

export default connect(state => {
  return {
    loading: state.auth.loading,
  };
}, authActions)(LoginForm);
```

Функція навігації по додатку:

```
import React, {Component} from 'react';
import {Link, NavLink} from 'react-router-dom';
import {Icon, Modal, Button, Header} from 'semantic-ui-react';

class ModalNav extends Component {
  constructor(props) {
    super(props);

    this.state = {
      modalOpen: false,
    };

    this.handleOpen = () => this.setState({modalOpen: true});
    this.handleClose = () => this.setState({modalOpen: false});
  }

  render() {
    return (
      <React.Fragment>
        <Modal centered={false}
               basic
               className="burger-nav"
               name="bars"
```

```
                    open={this.state.modalOpen}
                    trigger={<Icon name="bars" className="burger-ico"
                                onClick={this.handleOpen}/>
                    }>
                <Modal.Content>
                  <Menu onClick={this.handleClose.bind(this)}/>
                  <ul className='main-nav'>
                    <li>
                      <span onClick={this.handleClose}>Close</span>
                    </li>
                  </ul>
                </Modal.Content>
              </Modal>
            </React.Fragment>
          );
        }
      }

      class Menu extends Component {

        static defaultProps = {
          onClick: () => {
          },
        };

        render() {
          return (
            <ul className='main-nav'>
            <li>
                <Link  onClick={this.props.onClick}  to='/'><Icon  name="qrcode"
/>Home</Link>
            </li>
            <li>
                <NavLink       onClick={this.props.onClick}       to='/statistic'
activeClassName="active"><Icon name="line graph" />Statistic</NavLink>
            </li>
            <li>
                <NavLink       onClick={this.props.onClick}       to='/subjects'
activeClassName="active"><Icon name="book" />Subjects</NavLink>
            </li>
            <li>
```

```
                <NavLink         onClick={this.props.onClick}         to='/users'
activeClassName="active"><Icon name="users" /> Users</NavLink>
            </li>
            <li>
                <NavLink         onClick={this.props.onClick}         to='/user/'
activeClassName="active"><Icon name="add user" /> Add User</NavLink>
            </li>
            <li>
                <NavLink        onClick={this.props.onClick}        to='/subject/'
activeClassName="active"><Icon name="add" /> Add Subject</NavLink>
            </li>
            <li>
                <NavLink          onClick={this.props.onClick}          to='/faq'
activeClassName="active"><Icon name="info" /> FAQ</NavLink>
            </li>
              <li>
                <NavLink to='/logout'><Icon name="log out" /></NavLink>
              </li>
            </ul>);
        }
      }

      export default class Navigation extends Component {
        constructor(props) {
          super(props);

          this.state = {
            mobile: window.screen.width < 500,
          };
        }

        render() {
          const {mobile} = this.state;

          return (
            <React.Fragment>
              {!mobile ? <Menu/> : <ModalNav/>}
            </React.Fragment>
          );
        }
      }
```

4.2 Захист розробленого програмного забезпечення

Захист розроблюваного програмного забезпечення від несанкціонованого копіювання та розповсюдження пропонується здійснбвати за допомогою обфускації коду. Обфускація є надійною стратегією для збереження інтелектуальної власності на програмний код та забезпечення конфіденційності інформації. Обфускація передбачає зміну вихідного коду програми таким чином, що його стає важко зрозуміти та аналізувати людям, але при цьому програма продовжує працювати коректно.

Одним із способів обфускації коду є перейменування змінних, функцій та класів у неінформативні назви, які ускладнюють розуміння коду. Це також може включати в себе видалення коментарів та форматування, що полегшують читання коду.

Крім того, можна використовувати техніки обфускації, які приховують алгоритми та структури даних, за допомогою різних методів, таких як вбудовування коду, контроль потоку, та криптографічних перетворень. Це може значно ускладнити процес відтворення логіки програми та витіків її коду.

Обфускація може застосовуватися до коду JavaScript, CSS та HTML, який використовується на веб-сайті. Це допоможе запобігти копіюванню та розповсюдженню коду, а також захистити від несанкціонованого використання частин веб-сайту.

Важливо підкреслити, що обфускація не гарантує повного захисту коду від зловмисників, але суттєво ускладнює процес його аналізу та копіювання. Бажано використовувати її разом з іншими методами захисту, такими як, наприклад, зашифровані з’єднання, системи авторизації та сильні паролі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення призначене для системи автентифікації з застосуванням QR-кодів. Скришоти роботи розробленого забезпечення представлені на рисунках 5.1-5.4.

Користувачів розробленого додатку можна поділити на дві категорії:

1. Адміністратор – користувач, у якого є доступ до адміністративних функцій веб-сайту чи додатку, який захищаємо двофакторною авторизацією. Адміністратор може переглядати статистику дій користувачів та налаштовувати параметри сайту, зокрема, й параметри авторизації.

2. Звичайні користувачі – можуть авторизуватися на веб-ресурсі з використанням двофакторної авторизації зі скануванням QR-кодів.

Алгоритм дій користувача для двофакторної авторизації з QR-кодами:

Реєстрація в системі: користувач повинен зареєструватися в системі і налаштувати свій обліковий запис, додати свій мобільний телефон і включити двофакторну авторизацію.

Одержання QR-коду на комп'ютер: запит на авторизацію з комп'ютера, система генерує QR-код для цього облікового запису, QR-код відображається на моніторі комп'ютера та може бути сканований мобільним додатком.

Сканування QR-коду: користувач сканує QR-код за допомогою мобільного додатку, в якому увійшов у свій обліковий запис з використанням логіну та паролю.

Отримання коду: після сканування QR-коду додаток генерує одноразовий код, який користувач повинен ввести на сторінці авторизації.

Підтвердження: після введення одноразового коду підтверджується авторизація в системі і користувач має доступ до свого облікового запису на веб-ресурсі.

Дії, які треба виконати для встановлення системи на сервер:

1. Встановити Docker.

2. Встановити залежності командою у терміналі: `npm install`

3. Перейти в директорію проекту – у файлі packages.js можна побачити команди, необхідні для запуску преокту.

4. Послідовно запустити команди з packages.js:

```
docker:start
server
client
```

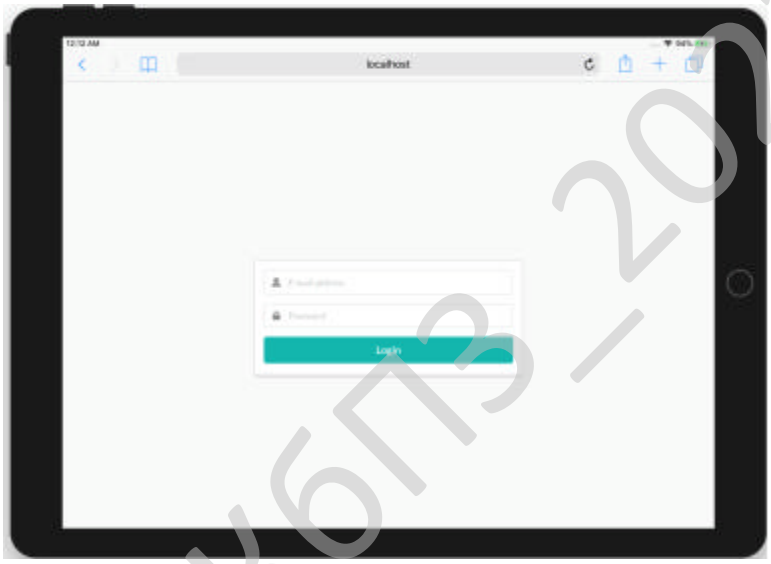

Рисунок 5.1 – Авторизація в розробленій системі

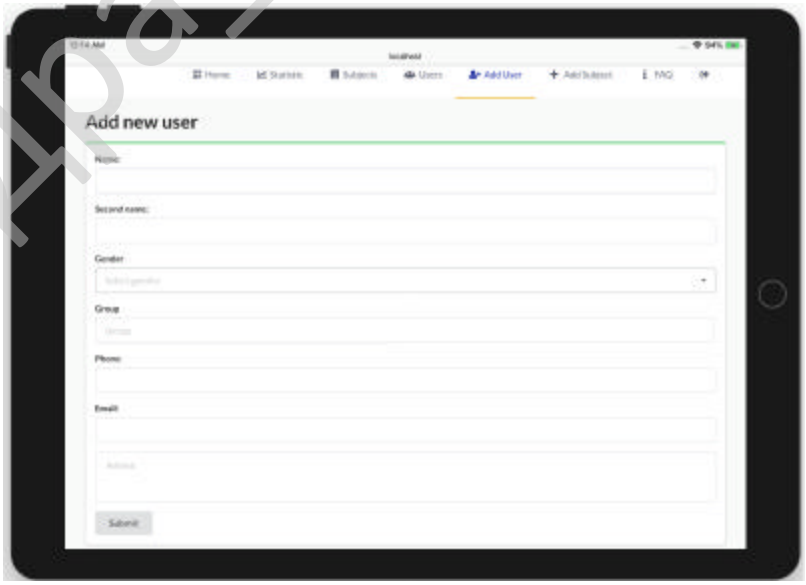

Рисунок 5.2 – Додавання адміністратором нового користувача в систему

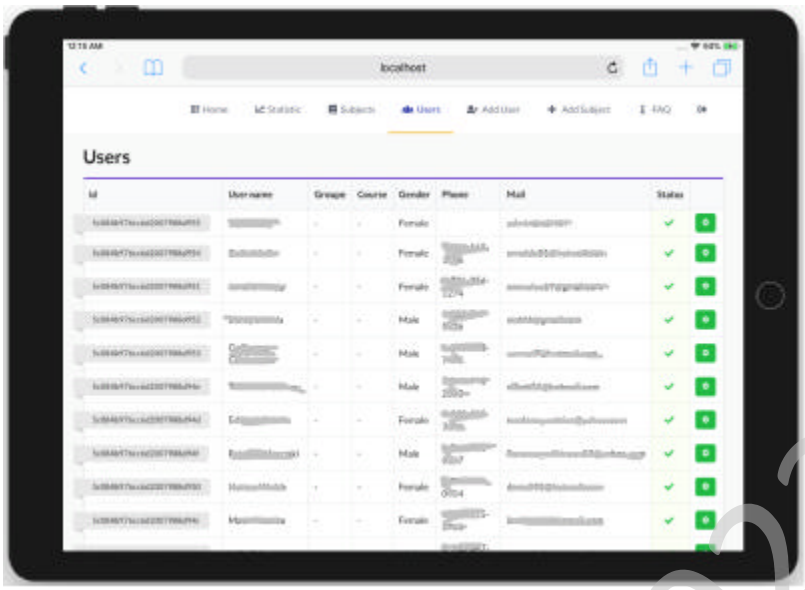

Рисунок 5.3 – Перегляд наявних користувачів в розробленій системі

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання роботи, призначено для системи автентифікації з застосуванням QR-кодів.

Для вирішення поставленої мети було поставлено та реалізовано наступні задачі:

– Дослідження існуючих систем автентифікації з застосуванням QR-кодів.

– Дослідження методів створення та сканування QR-кодів.

– Програмна реалізація системи автентифікації з застосуванням QR-кодів.

Розроблені під час виконання роботи алгоритми дозволяють успішно вирішувати завдання створення системи авторизації з застосуванням QR-кодів.

Система двофакторної авторизації з застосуванням QR-кодів є ефективним і надійним рішенням для захисту від несанкціонованого доступу до веб-ресурсів. Використання QR-кодів для авторизації забезпечує швидкий і простий спосіб перевірки легітимності користувача, що зменшує ризики, пов'язані з введенням невірних паролів та інших форм аутентифікації.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня JavaScript. При розробці використано об’єктно-орієнтований підхід у програмуванні. Це дозволило прискорити процес розробки та зробити код зрозумілим та масштабованим. Розроблене програмне забезпечення поділяється на клієнтську та серверну частини.

Програма може використовуватися під управлінням операційних систем Windows 10/11 та Android.

Розроблене програмне забезпечення можна застосовувати у різних додатках та веб-ресурсах, на яких треба захистити доступ до облікових записів користувачів двофакторною авторизацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цеслів О.В. WEB-програмування : навч. посібнник / О.В. Цеслів ; М-во освіти і науки, молоді та спорту України, Нац. техн. ун-т України “Київ. політехн. ін-т”. – Київ : НТУУ “КПІ”, 2011. – 296, с. .

2. Куленко М.Я. Основи графічного дизайну : підручник для студентів вищих навч. закладів / Михайло Куленко; МОНУ; Київський нац. ун-т будівництва і архітектури. – 2-ге вид., виправл. та доп. – Київ : Кондор, 2007. – 492с.

3. Пасічник О. Г., Пасічник О. В., Стеценко І. В. Основи веб-дизайну: Навч. посіб. -К.: Вид. група BHV. 2011р. -336 с.

4. Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-програмування» : для студ. денної та заоч. форми навч. спец. 123 «Комп’ютерна інженерія» та 125 «Кібербезпека» / уклад. Є. В. Мелешко, Л. В. Константинова ; М-во освіти і науки України, Кіровоград. нац. техн. ун-т, каф. прог. та захисту інформації. – Кропивницький : КНТУ, 2016. – 81 с.

5. Мелешко, Є. В. Алгоритми та структури даних : навч. посіб. / Є. В. Мелешко, М. С. Якименко, Л. І. Поліщук ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : Лисенко В.Ф., 2019. – 156 с.

6. Основи захисту інформації : метод. вказ. до викон. лаб. робіт для студ. за спец. 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”/ [уклад. : О. А. Смірнов, Є. В. Мелешко, О. К. Конопліцька-Слободенюк, В. Д. Хох, С. А. Смірнов] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2017. – 53 с.

7. Web-програмування. Частина 1 (frontend) : навч. посіб. / В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 208 с.

8. Методичні рекомендації до виконання лабораторних робіт з дисципліни «Web-програмування» для студентів денної та заочної форми навчання спеціальностей 123 «Комп'ютерна інженерія», 125 «Кібербезпека» та 122 «Комп'ютерні науки» / [уклад. : Є. В. Мелешко, В. В. Босько, Л. В. Константинова] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2023. - 87 с.

9. Тимошенко, К. О. Сучасні інструменти JS – розробника / К. О. Тимошенко, О. Є. Тесленко // Перспективні напрямки розвитку сучасних інформаційних систем та технологій : зб. тез доп. всеукр. наук.-практ. студ. конференція, 18 квіт. 2018р., м. Кропивницький. - Кропивницький : ЦНТУ, 2018. - С. 15-16.

10. Фесечко, Д. В. Дослідження та програмна реалізація веб-сайту компанії засобами фреймворку AngularJS : кваліфікаційна магістерська робота : спец. 123 «Комп'ютерна інженерія» / наук. кер. В. В. Босько ; Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2021. - 142 с.

11. Бушуєв, Р. В. Дослідження та програмна реалізація застосування об'єктно-орієнтованих баз даних в ІС : кваліфікаційна магістерська робота : спец. 123 "Комп'ютерна інженерія" / наук. кер. В. В. Босько ; Центральноукраїн. нац. тех. ун-т. - Кропивницький : ЦНТУ, 2022. - 171 с.

12. Демидов, З. Г. Основні види кібератак на WEB-сайти / З.Г. Демідов // Актуальні питання протидії кіберзлочинності та торгівлі людьми : зб. матеріалів Всеукр. наук.-практ. конф. (м. Харків, 15 листоп. 2017 р.) / МВС України, Харк. нац. ун-т внутр. справ; Координатор проектів ОБСЄ в Україні. – Харків : ХНУВС, 2017. – С. 131-134.

13. Cantelon, M., Harter, M., Holowaychuk, T. J., & Rajlich, N. (2014). Node. js in Action (pp. 17-20). Greenwich: Manning.

14. Green, B., & Seshadri, S. (2013). AngularJS. " O'Reilly Media, Inc.".

15. Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 80-83.

16. Syed, B. (2014). Beginning Node. js. Apress.

17. Mardan, A., Mardan, & Corrigan. (2018). Practical Node. js. Apress.

18. Teixeira, P. (2012). Professional Node. js: Building Javascript based scalable software. John Wiley & Sons.

19. Doglio, F., Doglio, & Corrigan. (2018). REST API Development with Node. js (Vol. 331). Apress.

20. Herron, D. (2020). Node. js Web Development: Server-side web development made easy with Node 14 using practical examples. Packt Publishing Ltd.

21. Pasquali, S. (2013). Mastering Node. js. Packt Publishing Ltd.

22. Позднякова, Т. (2020). QR-КОДИ: ЇХ СТВОРЕННЯ ТА ВИКОРИСТАННЯ. New pedagogical thought, 101(1), 36-42.

23. Засадна, Х. О. (2014). QR-кодування та альтернативні технології. Фінансовий простір, (3), 103-110.

24. Горовая, Н. М. (2021). Двофакторна система аутентифікації корпоративного середовища університету.

25. Ковальчук, К. В. (2019). Метод та засіб автентифікації користувачів корпоративних мереж.

26. Ляшенко, В. О. ВАРІАНТ СИСТЕМИ ЕЛЕКТРОННОЇ ІДЕНТИФІКАЦІЇ ТА АВТЕНТИФІКАЦІЇ НА ОСНОВІ 2D (QR) КОДУ. Редакційна колегія, 61.

27. Колос, В. Ю. (2020). Дослідження методів аутентифікації користувачів в інформаційно-комунікаційних системах.

28. Котенко, Б. М. (2021). Додаток для створення та розпізнавання QR-коду на основі електронного цифрового підпису (Bachelor's thesis, КПІ ім. Ігоря Сікорського).

29. Ткачук, Г. В. (2018). Аналіз безкоштовних програмних засобів для веб-програмування.

30. Двірничук, К. В., & Вацек, Д. О. (2022). ВЕБ-ПРОГРАМУВАННЯ ТА ВЕБ-ДИЗАЙН.

31. Mehlhorn, K., Sanders, P., & Sanders, P. (2008). Algorithms and data structures: The basic toolbox (Vol. 55, p. 56). Berlin: Springer.

32. Alsuwaiyel, M. H. (2021). Algorithms: design techniques and analysis (Vol. 15). World Scientific.

33. Crockford, D. (2008). JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.".

34. Щербакова, М. Є., & Щербаков, Є. В. (2014). Функціональні особливості JavaScript-додатків. Вісник Східноукраїнського національного університету імені Володимира Даля, (10), 142-146.

35. Кучминда, Р. М. (2021). Розробка інтерфейсу веб-додатку для візуалізації результатів стороннього інтерфейсу прикладного програмування з використанням мови JavaScript (Master's thesis, ТНТУ ім. І Пулюя).

36. Глущук, А. І., & Баленко, О. І. (2022). Javascript як мова розробки мобільних додатків (Doctoral dissertation, ТОВ ВПП" Контраст").

37. Онлайн-підручник з JavaScript. URL: http://www.w3schools.com/js/

38. Онлайн-підручник з HTML. URL: http://www.w3schools.com/html/

39. Романюк О. Н., Катєльніков Д. І., Косовець О. П. Веб-дизайн і комп’ютерна графіка. Навчальний посібник. Вінниця: ВНТУ, 2007. 142 с.

40. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

41. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

42. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

43. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

44. Lutz M. Learning Python, 5th Edition Fifth Edition. - O'Reilly Media, 2016. - 1643 p.

45. Lutz M. Python: Pocket Reference Fourth Edition. - O'Reilly Media, 2016. -

210 p.

46. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter 3rd Edition. - O'Reilly Media, 2022. - 579 p.

47. Fenner M. Machine Learning with Python for Everyone (Addison-Wesley Data & Analytics Series) 1st Edition, Kindle Edition. - Addison-Wesley Professional, 2019. – 586 p.

48. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення // Навчальний посібник. – Кіровоград: Вид. КНТУ, 2012. – 409 с.

49. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 p.

50. Ullman J.D., Aho A.V., Hopcroft J.E. The Design and Analysis of Computer Algorithms - International Economy Edition Paperback. – Pearson education, 1905. – 470 p.

51. Jain H. Data Structures & Algorithms In Go. – Hemant Jain, 2022. – 584 с.

52. Rocca La M. Advanced Algorithms and Data Structures. – Manning, 2021. – 768 p.

53. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 с.

54. Adam D. Scott Building Web Apps for Everyone. Copyright, O'Reilly Media, 2016, 245 p.

55. David Upton. CodeIgniter for Rapid PHP Application Development. Packt Publishing, 2007. 244 p.

56. Casteleyn S., Daniel F., Dolog P., Matera M. Engineering Web Applications. Berlin: Springer-Verlag, 2009, 363p.

57. Cody Lindley. Front-End Developer Handbook 2017. 2017. URL: https://frontendmasters.gitbooks.io/front-end-handbook-2017/content.

58. Матвієнко О.В., Бородкіна І.Л. Internet - технології: проектування Web-

сторінки : навч. посіб. Київ : Альтерпрес, 2003, 132 с.

59. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Компютерні мережі : навч. посіб. Львів : Магнолія, 2013, 250 с.

60. Oliver James. Html & CSS is hard (But it doesn't have to be). A friendly web development tutorial for complete beginners. 2017. URL: https://internetingishard.com/html-and-css.

61. Jeremy Thomas. MarkSheet. A free HTML and CSS tutorial. 2015-2017. URL: https://marksheet.io.

62. Shay Howe. Learn To Code HTML & CSS. Develop & Style Websites. 2014-2017. URL: https://learn.shayhowe.com/html-css

63. Shay Howe. Learn To Code Advanced HTML & CSS. Develop & Style Websites. 2014-2017. URL: https://learn.shayhowe.com/advanced-html-css.

64. Shklar L., Rosen Wiley R. Web Application Architecture: Principles, Protocols and Practices, 2009 – 440 p.

65. Welling L., Thomson L. PHP and MySQL Web Development (4th Edition), Addison-Wesley Professional, 2008 – 1008 p.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-125.23.0009.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Ковальов В.О.			Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Мелешко Є.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КБ-19	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для автентифікації з застосуванням QR-кодів.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему кібербезпеки для автентифікації з застосуванням QR-кодів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11 та з ОС Android.

5.8.1 Обладнання

Комп’ютер Intel Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування JavaScript.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи.
– Функціональна схема системи.
– Діаграма процесів.
– Блок-схема алгоритму роботи програми.
– Пояснювальна записка.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 20.05.2023 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист . .2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Є.В. Мелешко

Програмне забезпечення системи кібербезпеки для автентифікації з застосуванням QR-кодів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 29

Літера: РП

Кропивницький – 2023 року

Серверна частина розробленого програмного забезпечення

// Файл index.js

```
import Express from 'express';
import mongoose from 'mongoose';
import bodyParser from 'body-parser';
import compression from 'compression';
import jwt from 'express-jwt';
import cors from 'cors';

import config from './config';
import routes from './routes';

// Set native promises as mongoose promise
mongoose.Promise = global.Promise;

// MongoDB Connection
// if (process.env.NODE_ENV !== 'test') {
mongoose.connect(config.mongoURL, {
  useNewUrlParser: true,
}, (error) => {
  if (error) {
    console.error('Please make sure Mongodb is installed and running!'); //
eslint-disable-line no-console
    throw error;
  }
});
// }

const app = Express();

// Set Development modes checks
const isDevMode = process.env.NODE_ENV === 'development' || false;
const isProdMode = process.env.NODE_ENV === 'production' || false;

app.use(webpackHotMiddleware(compiler));
// }

if (isProdMode) {
  app.use(cors());
  app.use(compression());
}

app.use(bodyParser.json());

// app.use(Express.static(path.resolve(__dirname, '../dist/client')));

// ------------------------------------------------
// Routes
// ------------------------------------------------

app.use('/api', routes);

app.use(
  jwt({secret: config.jwt.secret}).unless({
    path: [
      '/api/auth/signup',
      '/api/auth/login',
      '/api/auth/forgot-password',
      '/api/auth/reset-password',
    ],
  }),
);

app.use((err, req, res, next) => {
  if (err.name === 'UnauthorizedError') {
```

```
    res.status(401).send('Missing authentication credentials.');
  }
});

app.listen(config.port, error => {
  if (!error) {
    console.log(`App is running on port: ${config.port}!`); // eslint-disable-
line
  }
});

module.exports = app;
```

// seed.service.js

```
import User from '@/server/models/user.model';
import Group from '@/server/models/group.model';
import Subject from '@/server/models/subject.model';
import Stat, {TYPE_CHECKIN} from '@/server/models/stat.model';
import faker from 'faker';
import sample from 'lodash/sample';

export async function seedRoute(req, res) {
  console.log('Seed users...');
  await seedUsers(10);

  console.log('Seed Groups...');
  await seedGroups();

  console.log('Seed Subjects...');
  await seedSubjects();

  console.log('Seed Stats...');
  await seedStats(1000);

  console.log('ok...');

  return res.send('ok');
}

/**
 * Seed Users
 */
export async function seedUsers(total = 100) {

  console.log('Clear users...');
  await User.remove({});

  const items = [
    // presets
    {
      firstName: 'Super',
      lastName: 'Man',
      email: 'admin@ad.min',
      password: 'admin',
    },
  ];

  for (let i = 0; i < total; i++) {
    items.push(new User({
      firstName: faker.name.firstName(),
      lastName: faker.name.lastName(),
      locale: faker.random.locale(),
      gender: faker.random.number({max: 2}),
      email: faker.internet.email(),
      phone: faker.phone.phoneNumberFormat(2),
      password: 'secret',
    }));
```

```
  }

  await User.create(items, (error) => {
    if (error) {
      throw error;
    }
  });
}

/**
 * Seed Groups
 */
export async function seedGroups() {
  console.log('Clear Groups...');
  await Group.remove({});

  const items = [
    {
      name: 'B52',
      description: 'test',
    },
  ];

  await Group.create(items, (error) => {
    if (error) {
      throw error;
    }
  });
}

/**
 * Seed Subjects
 */
export async function seedSubjects() {

  console.log('Clear Subjects...');
  await Subject.remove({});

  const items = [
    {
      name: 'Math',
      requiredHours: 40,
    },
    {
      name: 'Biology',
      requiredHours: 20,
    },
    {
      name: 'Chemistry',
      requiredHours: 55,
    },
      {
      name: 'Programing',
      requiredHours: 20,
    },
      {
      name: 'Religion',
      requiredHours: 15,
    },
      {
      name: 'JavaScript',
      requiredHours: 30,
    },
  ];

  await Subject.create(items, (error) => {
    if (error) {
      throw error;
    }
```

```
  });
}

/**
 * Seed Stats
 */
export async function seedStats(total = 100) {

  console.log('Clear Stats...');
  await Stat.remove({});

  const items = [];

  const randomUsers = await User.aggregate([
    {$sample: {size: 100}},
  ]);

  const randomSubjects = await Subject.aggregate([
    {$sample: {size: 100}},
  ]);

  for (let i = 0; i < total; i++) {
    const user = sample(randomUsers);
    const subject = sample(randomSubjects);
    items.push(new Stat({
      type: TYPE_CHECKIN,
      user: user._id,
      subject: subject._id,
      createdAt: faker.date.past(),
    }));
  }

  await Stat.create(items, (error) => {
    if (error) {
      throw error;
    }
  });
}
```

// user.service.js

```
import Joi from 'joi';
import bcrypt from 'bcryptjs';

export function encryptPassword(palinText) {
  const salt = bcrypt.genSaltSync(10);
  return bcrypt.hashSync(palinText, salt);
}

export function comparePassword(plainText, encrypedPassword) {
  return bcrypt.compareSync(plainText, encrypedPassword);
}

export function validateSignup(body) {
  const schema = Joi.object().keys({
    firstName: Joi.string().required(),
    lastName: Joi.string().required(),
    email: Joi.string().email().required(),
    password: Joi.string().required(),
    role: Joi.number().integer(),
  });
  const {value, error} = Joi.validate(body, schema);
  if (error && error.details) {
    return {error};
  }
  return {value};
}

export function validateLogin(body) {
```

```
  const schema = Joi.object().keys({
    email: Joi.string().email().required(),
    password: Joi.string().required(),
  });
  const {value, error} = Joi.validate(body, schema);
  if (error && error.details) {
    return {error};
  }
  return {value};
}
```

// Файл auth.controller.js

```
import User, {ROLE_USER} from '@/server/models/user.model';
import {
  encryptPassword,
  comparePassword,
  validateLogin,
  validateSignup,
} from '@/server/services/user.service';

import jwt from 'jsonwebtoken';
import config from '../config';

export async function me(req, res) {
  const user = await User.findById(req.user.id);
  // TODO: refactory
  if (user) {
    user.password = undefined;
  }
  return res.status(200).json(user);
}

/**
 * @param {Object} req
 * @param {Object} res
 * @returns {Promise<void>}
 */
export async function logout(req, res) {
  // TODO: revoke token ?
  // res.redirect('/');
  res.status(200).send({auth: false, token: null});
}

/**
 * @param {Object} req
 * @param {Object} res
 * @returns {Promise<void>}
 */
export async function login(req, res) {
  try {
    const {value, error} = validateLogin(req.body);
    if (error) {
      return res.status(400).json(error);
    }
    const user = await User.findOne({email: value.email});
    if (user && comparePassword(value.password, user.password)) {
      const token = jwt.sign({id: user._id}, config.jwt.secret,
        {expiresIn: config.jwt.expiresIn});
      return res.json({
        user,
        token,
        expiresIn: config.jwt.expiresIn,
      });
    } else {
      return res.status(400).json({message: 'Wrong email or password!'});
    }
  } catch (err) {
```

```
    return res.status(500).send(err);
  }
}

/**
 * @param {Object} req
 * @param {Object} res
 * @returns {Promise<void>}
 */
export async function register(req, res) {
  try {
    const {value, error} = validateSignup(req.body);
    if (error) {
      return res.status(400).json(error);
    }
    const encryptedPass = encryptPassword(value.password);
    const user = await User.create({
      email: value.email,
      firstName: value.firstName,
      lastName: value.lastName,
      password: encryptedPass,
      role: value.role || ROLE_USER,
    });
    return res.json({success: true});
  } catch (err) {
    return res.status(500).send(err);
  }
}

export async function forgotPassword(req, res) {
  // TODO: ...
  res.status(401).json({
    message: 'Not implemented',
  });
}

export async function resetPassword(req, res) {
  // TODO: ...
  res.status(401).json({
    message: 'Not implemented',
  });
}
```

// Файл stats.controller.js

```
import Stat from '@/server/models/stat.model';

/**
 * Search entities
 *
 * @param req
 * @param res
 * @returns void
 */
export function search(req, res) {
  const query = Stat.find({
      // TODO: filters
    }).
    populate({path: 'user', select: ['firstName', 'lastNam', 'email',
'gender']}).
    populate({path: 'subject', select: ['name']})
  ;
  query.exec((err, items) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({items});
  });
}
```

// Файл subject.controller.js

```
import Subject from '@/server/models/subject.model';
import sanitizeHtml from 'sanitize-html';

/**
 * Search entities
 *
 * @param req
 * @param res
 * @returns void
 */
export function search(req, res) {
  const query = Subject.find({
    // TODO: filters
  });
  query.sort('-createdAt').exec((err, items) => {
    if (err) {
      res.status(500).send(err);
    }
    // TODO: paginate
    res.json({ items });
  });
}

/**
 * View entity
 *
 * @param req
 * @param res
 * @returns void
 */
export function view(req, res) {
  const { id } = req.params;
  Subject.findOne({ _id: id }).exec((err, data) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({ data });
  });
}

/**
 * Save entity
 *
 * @param req
 * @param res
 * @returns void
 */
export function store({ body }, res) {
  if (!body.name) {
    res.status(403).end();
  }

  const subject = new Subject(body);
  subject.name = sanitizeHtml(body.name);
  subject.description = sanitizeHtml(body.description);

  subject.save((err, status) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({ status });
  });
}

/**
```

```
 * Delete entity
 *
 * @param req
 * @param res
 * @returns void
 */
export function remove(req, res) {
  const { id } = req.params;
  Subject.findOne({ _id: id }).exec((err, entity) => {
    if (err) {
      res.status(500).send(err);
    }
    entity.remove(() => {
      res.status(200).end();
    });
  });
}
```

// Файл user.controller.js

```
import User from '@/server/models/user.model';
import sanitizeHtml from 'sanitize-html';

/**
 * Search entities
 *
 * @param req
 * @param res
 * @returns void
 */
export function search(req, res) {

  const query = User.find({
    // TODO: filters
  });

  query.sort('-createdAt').exec((err, items) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({ items });
  });
}

/**
 * View entity
 *
 * @param req
 * @param res
 * @returns void
 */
export function view({ params }, res) {
  const { id } = params;
  User.findOne({ _id: id })
  .populate('stats')
  .exec((err, data) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({ data });
  });
}

/**
 * Save entity
 *
 * @param req
 * @param res
```

```
 * @returns void
 */
export function store({ body }, res) {

  // TODO: validators

  if (!body.firstName ||
    !body.lastName ||
    !body.email ||
    !body.password) {
    res.status(403).end();
  }

  const user = new User(body);

  user.firstName = sanitizeHtml(body.firstName);
  user.lastName = sanitizeHtml(body.lastName);
  user.email = sanitizeHtml(body.email);
  user.password = body.password;

  user.save((err, status) => {
    if (err) {
      res.status(500).send(err);
    }
    res.json({ status });
  });
}

/**
 * Delete entity
 *
 * @param req
 * @param res
 * @returns void
 */
export function remove(req, res) {
  const { id } = req.params;
  User.findOne({ _id: id }).exec((err, entity) => {
    if (err) {
      res.status(500).send(err);
    }
    entity.remove(() => {
      res.status(200).end();
    });
  });
}
```

// Файл group.model.js

```
import mongoose from 'mongoose';

const Schema = mongoose.Schema;

const GroupSchema = new Schema({
  name: {type: String, required: true},
  description: {type: String},
  author: { type: Schema.Types.ObjectId, ref: 'User' },
  createdAt: {type: Date, default: Date.now},
});

export default mongoose.model('Group', GroupSchema);
```

// Файл stat.model.js

```
import mongoose from 'mongoose';

const Schema = mongoose.Schema;

export const TYPE_CHECKIN = 'checkin';
```

```
export const TYPE_CHECKOUT = 'checkout';
// TODO: more types ...

const StatSchema = new Schema({
  type: {type: String, required: true},
  description: {type: String},
  subject: {type: Schema.Types.ObjectId, ref: 'Subject'},
  user: {type: Schema.Types.ObjectId, ref: 'User'},
  // meta: {type: Object},
  createdAt: {type: Date, default: Date.now},
});

export default mongoose.model('Stat', StatSchema);
```

// Файл subject.model.js

```
import mongoose from 'mongoose';

const Schema = mongoose.Schema;

const SubjectSchema = new Schema({
  name: {type: String, required: true},
  description: {type: String},
  requiredHours: {type: Number, default: 0},
  duration: {type: Number, default: 0},
  createdAt: {type: Date, default: Date.now},
});

export default mongoose.model('Subject', SubjectSchema);
```

// Файл user.model.js

```
import mongoose from 'mongoose';
// import { autoIncrement } from 'mongoose-plugin-autoinc';
// import bcrypt from 'bcryptjs';

import {encryptPassword} from '@/server/services/user.service';

const Schema = mongoose.Schema;

// TODO: move to constants

export const ROLE_USER = 0;
export const ROLE_MANAGER = 1;
export const ROLE_ADMIN = 999;

export const STATUS_BLOCKED = -1;
export const STATUS_ACTIVE = 1;
export const STATUS_INACTIVE = 0;

export const GENDER_MALE = 1;
export const GENDER_FEMALE = 2;

const UserSchema = new Schema({

  email: {
    type: String,
    trim: true,
    unique: true,
    index: true,
    lowercase: true,
    required: true,
  },

  firstName: {
    type: String,
    trim: true,
    required: true,
  },
```

```
  lastName: {
    type: String,
    trim: true,
    required: true,
  },

  password: {
    type: String,
    default: '',
  },

  gender: {
    type: Number,
    default: 0,
  },

  role: {
    type: String,
    default: ROLE_USER,
  },

  group: {
    type: String,
    default: '',
  },

  phone: {
    type: String,
    default: '',
  },

  // Date Of Birth
  dob: {
    type: Date,
    default: null,
  },

  resetPasswordToken: String,
  resetPasswordExpires: Date,

  verified: {
    type: Boolean,
    default: false,
  },

  verifyToken: {
    type: String,
  },

  status: {
    type: Number,
    default: STATUS_ACTIVE,
  },

  studyDate: {
    type: Date,
    default: null,
  },

  createdAt: {
    type: Date,
    default: Date.now,
  },

  stats: [{type: Schema.Types.ObjectId, ref: 'Stat'}],

});
```

```
/**
 * Password hashing
 */
UserSchema.pre('save', function(next) {
  if (this.isModified('password')) {
    this.password = encryptPassword(this.password);
  }
  return next();
});

/**
 * Pick is only some fields of object
 *
 * http://mongoosejs.com/docs/api.html#document_Document-toObject
 *
 */
UserSchema.methods.pick = function(props, model) {
  return _.pick(model || this.toJSON(), props || [
    'email',
    'firstName',
    'lastName',
    'role',
    'group',
    'dob',
    'phone',
    'status',
  ]);
};

// UserSchema.plugin(autoIncrement, 'User');

export default mongoose.model('User', UserSchema);
```

Клієнтська частина розробленого програмного забезпечення

// Файл index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import store from './store';
import {Provider} from 'react-redux';

ReactDOM.render(
  <Provider store={store}>
    <App/>
  </Provider>, document.getElementById('root'),
);
```

// Файл App.js

```
import React, {Component} from 'react';
import {HashRouter} from 'react-router-dom';
import {connect} from 'react-redux';

import MetaTags from 'react-meta-tags';

import {Routes, AuthRoutes} from './routes';

import Header from './common/Header';
import ProgressLine from './common/ProgressLine';

import './assets/scss/main.scss';

class App extends Component {
  render() {
    const isAuthenticated = !!this.props.token;
    return (
      <HashRouter>
        <div className="shell">
          <MetaTags>
            <title>Observer</title>
            <meta name="viewport"
                  content="width=device-width, initial-scale=1"/>
          </MetaTags>

          <ProgressLine/>

          {isAuthenticated ? (
            <React.Fragment>
              <Header/>
              <div className="content">
                <Routes/>
              </div>
            </React.Fragment>
          ) : (
            <AuthRoutes/>
          )}

        </div>
      </HashRouter>
    );
  }
}

export default connect(state => {
  return {
    token: state.auth.token,
  };
})(App);
```

// Файл Navigation.js

```
import React, {Component} from 'react';
import {Link, NavLink} from 'react-router-dom';
import {Icon, Modal, Button, Header} from 'semantic-ui-react';

class ModalNav extends Component {
  constructor(props) {
    super(props);

    this.state = {
      modalOpen: false,
    };

    this.handleOpen = () => this.setState({modalOpen: true});
    this.handleClose = () => this.setState({modalOpen: false});
  }

  render() {
    return (
      <React.Fragment>
        <Modal centered={false}
               basic
               className="burger-nav"
               name="bars"
               open={this.state.modalOpen}
               trigger={<Icon name="bars" className="burger-ico"
                             onClick={this.handleOpen}/>
               }>
          <Modal.Content>
            <Menu onClick={this.handleClose.bind(this)}/>
            <ul className='main-nav'>
              <li>
                <span onClick={this.handleClose}>Close</span>
              </li>
            </ul>
          </Modal.Content>
        </Modal>
      </React.Fragment>
    );
  }
}

class Menu extends Component {

  static defaultProps = {
    onClick: () => {
    },
  };

  render() {
    return (
      <ul className='main-nav'>
      <li>
          <Link onClick={this.props.onClick} to='/'><Icon name="qrcode"
/>Home</Link>
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/statistic'
activeClassName="active"><Icon name="line graph" />Statistic</NavLink>
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/subjects'
activeClassName="active"><Icon name="book" />Subjects</NavLink>
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/users'
activeClassName="active"><Icon name="users" /> Users</NavLink>
```

```
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/user/'
activeClassName="active"><Icon name="add user" /> Add User</NavLink>
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/subject/'
activeClassName="active"><Icon name="add" /> Add Subject</NavLink>
      </li>
      <li>
          <NavLink onClick={this.props.onClick} to='/faq'
activeClassName="active"><Icon name="info" /> FAQ</NavLink>
      </li>
        <li>
          <NavLink to='/logout'><Icon name="log out" /></NavLink>
        </li>
      </ul>);
  }
}

export default class Navigation extends Component {
  constructor(props) {
    super(props);

    this.state = {
      mobile: window.screen.width < 500,
    };
  }

  render() {
    const {mobile} = this.state;

    return (
      <React.Fragment>
        {!mobile ? <Menu/> : <ModalNav/>}
      </React.Fragment>
    );
  }
}
```

// Файл ProgressLine.js

```
import React, {Component} from 'react';
import Progress from 'react-progress-2';
import {connect} from 'react-redux';

class ProgressLine extends Component {
  componentWillReceiveProps(nextProps) {
    if (nextProps.state) {
      Progress.show();
    } else {
      Progress.hide();
    }
  }

  render() {
    return (
      <Progress.Component/>
    );
  }
}

function mapStateToProps(state) {
  return {
    state: state.app.loading,
  };
}

export default connect(mapStateToProps)(ProgressLine);
```

// Файл Header.js

```
import React, {Component} from 'react';
import {Link} from 'react-router-dom';
import Navigation from './Navigation';
import {Logo} from '../assets';

export default class Header extends Component {
  render() {
    return (
      <div className="header">
        <Link className="logo" to="/">
          <img src={Logo} alt=""/>
        </Link>
        <Navigation/>
      </div>
    );
  }
}
```

// Файл login.js

```
import React, {Component} from 'react';
import {connect} from 'react-redux';
import {Button, Form, Grid, Message, Segment} from 'semantic-ui-react';
import * as authActions from '@/client/store/actions/auth.actions';
import {Logo} from '../assets';

class LoginForm extends Component {

  state = {
    data: {
      email: null,
      password: null,
    },
    errors: {},
  };

  handleSubmit = async (e) => {
    e.preventDefault();

    const {login} = this.props;

    await this.setStateAsync({loading: true});

    const resp = await login(this.state.data);
    const errors = {};

    if (resp.err) {
      if (resp.err.message) {
        errors.common = resp.err.message;
      }
      // multiple errors
      else if (resp.body.details) {
        resp.body.details.forEach(err => {
          errors[err.context.key] = err.message;
        });
      }
      this.setState({errors});
    } else {
      // TODO: redirect
    }

    this.setState({errors, loading: false});
  };

  handleInputChange = e => {
    this.setState({
```

```
    data: {
      ...this.state.data,
      [e.target.name]: e.target.value,
    },
  });
};

setStateAsync(state) {
  return new Promise((resolve) => {
    this.setState(state, resolve);
  });
}

render() {

  const {loading} = this.props;
  const {data, errors} = this.state;
  const errorMessage = Object.entries(errors).map(err => err[1]).join('\n');

  return (
    <div className='login-form'>
      <style>{`
    body > div,
    body > div > div,
    body > div > div > div.login-form {
      height: 100%;
    }
  `}</style>

      <Grid textAlign='center' style={{height: '100%'}}
            verticalAlign='middle'>
        <Grid.Column style={{maxWidth: 450}}>

          <h2>
            <img className="logo" src={Logo} alt=""/>
          </h2>

          {errorMessage && <Message error content={errorMessage}/>}

          <Form size='large' loading={loading} onSubmit={this.handleSubmit}
                error={errors.length > 0}>
            <Segment stacked>
              <Form.Input
                fluid
                icon='user'
                iconPosition='left'
                placeholder='E-mail address'
                name='email'
                defaultValue={data.email}
                error={!!errors.email}
                onChange={this.handleInputChange}
              />
              <Form.Input
                fluid
                icon='lock'
                iconPosition='left'
                placeholder='Password'
                type='password'
                name='password'
                defaultValue={data.password}
                error={!!errors.password}
                onChange={this.handleInputChange}
              />
              <Button color='teal' fluid size='large'>
                Login
              </Button>
            </Segment>

          </Form>
```

```
          </Grid.Column>
        </Grid>
      </div>
    );
  }
}

export default connect(state => {
  return {
    loading: state.auth.loading,
  };
}, authActions)(LoginForm);
```

// Файл logout.js

```
import React, {Component} from 'react';
import {connect} from 'react-redux';
import {Redirect} from 'react-router-dom';
import * as authActions from '@/client/store/actions/auth.actions';

class LogoutPage extends Component {
  componentWillMount() {
    this.props.logout();
  }

  render() {
    return (
      <Redirect to="/"/>
    );
  }
}

export default connect(null, {
  logout: authActions.logout,
})(LogoutPage);
```

// Файл main-page.js

```
import React, {Component} from 'react';
import QrReader from 'react-qr-reader';
import {Container, Button, Icon, Message} from 'semantic-ui-react';

export default class MainPage extends Component {
  constructor(props) {
    super(props);

    this.state = {
      showQR: false,
      delay: 300,
      result: '',
    };
  }

  triggerQr() {
    const {showQR} = this.state;

    this.setState({
      showQR: !showQR,
    });
  };

  handleError(err) {
    console.error(err);
  }

  handleScan(data) {
    if (data) {
      this.setState({
```

```
        result: data,
        showQR: false,
      });
    }
  }

  renderResult(result) {
    return (
      <Message positive>
        <Message.Header>QR was read</Message.Header>
        <p>
          Go to <b><a href={result}>{result}</a></b>.
        </p>
      </Message>
    );
  }

  render() {
    const {showQR, delay, result} = this.state;

    return (
      <Container className='center aligned' fluid>
        <Button
          onClick={this.triggerQr.bind(this)}
          color='teal' icon='qrcode'
          size='massive'
          content='Scan QR code'/>

        {result ? this.renderResult(result) : null}
        {showQR ? <QrReader
          className='qr-reader'
          delay={delay}
          onError={this.handleError.bind(this)}
          onScan={this.handleScan.bin(this)}
        /> : null}
      </Container>
    );
  }
}
```

// Файл statistic-list.js

```
import React, {Component} from 'react';
import {Table, Label, Icon, Header, Dimmer, Loader} from 'semantic-ui-react';
import moment from 'moment';
import {fetchStart, fetchFinish} from '../store/actions/app.actions';
import api from '@/client/services/api';

import {connect} from 'react-redux';

class StatisticList extends Component {

  state = {
    stats: [],
  };

  async componentDidMount() {
    this.props.fetchStart();
    const resp = await api.endpoints.getStats();
    if (resp.ok) {
      this.setState({stats: resp.body.items});
    }
    this.props.fetchFinish();
  }

  renderStats() {
    const {stats} = this.state;
    const statsList = [];
```

```
    stats.forEach((stat, index) => {
      statsList.push(
        <Table.Row key={stat._id}>
          <Table.Cell><Label ribbon>{index}</Label></Table.Cell>
          <Table.Cell>{stat.user.firstName}</Table.Cell>
          <Table.Cell>{stat.subject.name}</Table.Cell>
          <Table.Cell>{moment(stat.createdAt).format('ll')}</Table.Cell>
        </Table.Row>,
      );
    });

    return statsList;
  }

  render() {
    const {stats} = this.state;

    return (
      <React.Fragment>
        <Header as='h1'>Statistic</Header>
        <Table celled selectable compact='very' color="teal">
          <Table.Header>
            <Table.Row>
              <Table.HeaderCell width={1}></Table.HeaderCell>
              <Table.HeaderCell>User ID</Table.HeaderCell>
              <Table.HeaderCell>Subject</Table.HeaderCell>
              <Table.HeaderCell>Date of visiting</Table.HeaderCell>
            </Table.Row>
          </Table.Header>

          <Table.Body>
            {stats.length ? this.renderStats() : <Table.Row>
              <Table.Cell colSpan={4} textAlign="center"> No Statistic
Data</Table.Cell>
            </Table.Row>}
          </Table.Body>
        </Table>
      </React.Fragment>
    );
  }
}

function mapStateToProps(state) {
  return {
    loading: state.loading,
  };
}

const mapDispatchToProps = {
  fetchFinish,
  fetchStart,
};

export default connect(mapStateToProps, mapDispatchToProps)(StatisticList);
```

// Файл subject-add.js

```
import React, {Component} from 'react';
import {Redirect} from 'react-router-dom';
import {Form, Header, Segment, Modal} from 'semantic-ui-react';
import {connect} from 'react-redux';
import {fetchStart, fetchFinish} from '../store/actions/app.actions';

class SubjectAdd extends Component {
  constructor(props) {
    super(props);
    this.state = {
      subject: {},
    };
```

```
  }

  onChangeHandler(e) {
    const {subject} = this.state;
    const newSubjectData = Object.assign(subject,
      {[e.target.name]: e.target.value});

    this.setState({
      subject: newSubjectData,
    });
  }

  onSubmit() {
    const {subject} = this.state;

    this.props.fetchStart();
    fetch('/api/subjects/', {
      method: 'POST',
      body: subject,
    }).then(function(response) {
      return response.json();
    }).then((data) => {
      <Redirect to='/subjects'/>;
    });
  }

  render() {
    const {subject} = this.state;

    return (
      <React.Fragment>
        <Header as='h1'>Add new subject</Header>
        <Segment color='green'>
          <Form onSubmit={this.onSubmit.bind(this)}>
            <Form.Field>
              <label>Subject name: </label>
              <input
                onChange={this.onChangeHandler.bind(this)}
                name='name'
                value={subject.firstName}
              />
            </Form.Field>

            <Form.Button>Submit</Form.Button>
          </Form>
        </Segment>
      </React.Fragment>
    );
  }
}

const mapDispatchToProps = {
  fetchFinish,
  fetchStart,
};

export default connect(null, mapDispatchToProps)(SubjectAdd);
```

// Файл subject-list.js

```
import React, {Component} from 'react';
import {Table, Label, Icon, Header, Dimmer, Loader} from 'semantic-ui-react';
import {fetchStart, fetchFinish} from '../store/actions/app.actions';

import {connect} from 'react-redux';
import api from '@/client/services/api';

class SubjectList extends Component {
    constructor(props) {
```

```
        super(props);

        this.state = {
            subjects: [],
        };
    }

    async componentDidMount() {
        this.props.fetchStart();
        const resp = await api.endpoints.getSubjects();
        if (resp.ok) {
            this.setState({subjects: resp.body.items});
        }
        this.props.fetchFinish();
    }

    renderSubjectList(sub, index) {
        return (
            <Table.Row key={sub._id}>
                <Table.Cell><Label ribbon>{index}</Label></Table.Cell>
                <Table.Cell>{sub._id}</Table.Cell>
                <Table.Cell>{sub.name}</Table.Cell>
                <Table.Cell>{sub.requiredHours}</Table.Cell>
                <Table.Cell>{sub.duration}</Table.Cell>
            </Table.Row>
        );
    }

    render() {
        const {subjects} = this.state;

        return (
            <React.Fragment>
                <Header as='h1'>Subjects </Header>
                <Table celled selectable compact='very' color="teal">
                    <Table.Header>
                        <Table.Row>
                            <Table.HeaderCell width={1}></Table.HeaderCell>
                            <Table.HeaderCell width={1}>id</Table.HeaderCell>
                            <Table.HeaderCell>Subject</Table.HeaderCell>
                            <Table.HeaderCell>Required Hours</Table.HeaderCell>
                            <Table.HeaderCell>Duration</Table.HeaderCell>
                        </Table.Row>
                    </Table.Header>

                    <Table.Body>
                        {subjects.length ? subjects.map(this.renderSubjectList)
:
                            <Table.Row>
                                <Table.Cell colSpan={5} textAlign="center"> No
Subjects
                                    Data</Table.Cell>
                            </Table.Row>}
                    </Table.Body>
                </Table>
            </React.Fragment>
        );
    }
}

function mapStateToProps(state) {
    return {
        loading: state.loading,
    };
}

const mapDispatchToProps = {
    fetchFinish,
    fetchStart,
```

```
};

export default connect(mapStateToProps, mapDispatchToProps)(SubjectList);
```

// Файл user-add.js

```
import React, {Component} from 'react';
import {Redirect} from 'react-router-dom';
import {Form, Header, Segment, Modal} from 'semantic-ui-react';
import {connect} from 'react-redux';
import {fetchStart, fetchFinish} from '../store/actions/app.actions';

class UserAdd extends Component {
  constructor(props) {
    super(props);

    this.state = {
      user: false,
    };

    this.id = this.props.match.params.id ? this.props.match.params.id : '';
  }

  componentDidMount() {
    if (this.id) {
      this.props.fetchStart();
      fetch('/api/users/' + this.id).then(function(response) {
        return response.json();
      }).then((data) => {
        this.setState({
          user: data.data,
        });

        this.props.fetchFinish();
      });
    }
  }

  onChangeHandler(e) {
    const {user} = this.state;
    const newUserData = Object.assign(user, {[e.target.name]: e.target.value});

    this.setState({
      user: newUserData,
    });
  }

  onSubmit() {
    const {user} = this.state;

    this.props.fetchStart();
    fetch('/api/users/' + this.id, {
      method: 'POST',
      body: user,
    }).then(function(response) {
      return response.json();
    }).then((data) => {
      <Redirect to='/users'/>;
    });
  }

  render() {
    const options = [
      {key: 'm', text: 'Male', value: '0'},
      {key: 'f', text: 'Female', value: '1'},
    ];
    const {user} = this.state;

    return (
```

```
    <React.Fragment>
      <Header as='h1'>{user ? 'Edit user' : 'Add new user'}</Header>
      <Segment color='green'>
        <Form onSubmit={this.onSubmit.bind(this)}>
          <Form.Field>
            <label>Name: </label>
            <input
              onChange={this.onChangeHandler.bind(this)}
              name='firstName'
              value={user.firstName}
            />
          </Form.Field>
          <Form.Field>
            <label>Second name:</label>
            <input
              onChange={this.onChangeHandler.bind(this)}
              name='lastName'
              value={user.lastName}
            />
          </Form.Field>
          <Form.Field>
            <Form.Select fluid label='Gender' defaultValue={user.gender}
                         options={options} placeholder='Select gender'/>
          </Form.Field>
          <Form.Group widths='equal'>
            <Form.Input
              onChange={this.onChangeHandler.bind(this)}
              fluid
              label='Group'
              value={user.group}
              placeholder='Group'
            />
            <Form.Input
              onChange={this.onChangeHandler.bind(this)}
              fluid
              label='Course'
              value={user.group}
              placeholder='Course'
            />
          </Form.Group>
          <Form.Field>
            <label>Phone:</label>
            <input
              name='phone'
              value={user.phone}
              onChange={this.onChangeHandler.bind(this)}
            />
          </Form.Field>
          <Form.Field>
            <label>Email:</label>
            <input
              name='email'
              value={user.email}
              onChange={this.onChangeHandler.bind(this)}
            />
          </Form.Field>
          <Form.Field>
            <Form.TextArea
              placeholder='Adress'
              name='adress'
              value={user.adress}
              onChange={this.onChangeHandler.bind(this)}
            />
          </Form.Field>
          <Form.Button>Submit</Form.Button>
        </Form>
      </Segment>
    </React.Fragment>
  );
```

```
  }
}

const mapDispatchToProps = {
  fetchFinish,
  fetchStart,
};

export default connect(null, mapDispatchToProps)(UserAdd);
```

// Файл user-list.js

```
import React, {Component} from 'react';
import {Table, Label, Icon, Header, Button} from 'semantic-ui-react';
import {Link} from 'react-router-dom';
import {connect} from 'react-redux';
import {
  fetchFinish,
  fetchStart,
} from '../store/actions/app.actions';
import api from '@/client/services/api';

class UserList extends Component {

  state = {
    users: [],
  };

  async componentDidMount() {
    this.props.fetchStart();
    const resp = await api.endpoints.getUsers();
    if (resp.ok) {
      this.setState({users: resp.body.items});
    }
    this.props.fetchFinish();
  }

  renderUser() {
    const {users} = this.state;
    const userList = [];

    users.forEach((user) => {
      userList.push(
        <Table.Row key={user._id}>
          <Table.Cell><Label ribbon>{user._id}</Label></Table.Cell>
          <Table.Cell>{user.firstName} {user.lastName}</Table.Cell>
          <Table.Cell>{user.group ? user.group : '-'}</Table.Cell>
          <Table.Cell>{user.studyDate ? user.studyDate : '-'}</Table.Cell>
          <Table.Cell>{user.gender === 1 ? 'Male' : 'Female'}</Table.Cell>
          <Table.Cell>{user.phone}</Table.Cell>
          <Table.Cell>{user.email}</Table.Cell>
          <Table.Cell
            positive={user.status === 1 ? true : false}
            negative={user.status !== 1 ? true : false}
            textAlign="center">
            {user.status === 1 ? <Icon color="green" name="checkmark"/> : null}
          </Table.Cell>
          <Table.Cell textAlign="center">
            <Link to={'user/' + user._id}><Button icon='setting' color='green'
                                                  size='mini'/></Link>
          </Table.Cell>
        </Table.Row>,
      );
    });

    return userList;
  }

  render() {
```

```
    const {users} = this.state;

    return (
      <React.Fragment>
        <Header as='h1'>Users</Header>
        <Table celled selectable compact='very' color="violet">
          <Table.Header>
            <Table.Row>
              <Table.HeaderCell width={1}>id</Table.HeaderCell>
              <Table.HeaderCell>User name</Table.HeaderCell>
              <Table.HeaderCell>Groupe</Table.HeaderCell>
              <Table.HeaderCell>Course</Table.HeaderCell>
              <Table.HeaderCell>Gender</Table.HeaderCell>
              <Table.HeaderCell>Phone</Table.HeaderCell>
              <Table.HeaderCell>Mail</Table.HeaderCell>
              <Table.HeaderCell width={1}>Status</Table.HeaderCell>
              <Table.HeaderCell width={1}></Table.HeaderCell>
            </Table.Row>
          </Table.Header>

          <Table.Body>
            {users.length ? this.renderUser() :
              <Table.Row>
                <Table.Cell colSpan={8} textAlign="center"> No User
                  Data</Table.Cell>
              </Table.Row>}
          </Table.Body>
        </Table>
      </React.Fragment>
    );
  }
}

function mapStateToProps(state) {
  return {
    loading: state.app.loading,
  };
}

const mapDispatchToProps = {
  fetchFinish,
  fetchStart,
};

export default connect(mapStateToProps, mapDispatchToProps)(UserList);
```

// Файл user-settings.js

```
import React, {Component} from 'react';
import {Table, Label, Icon, Header} from 'semantic-ui-react';

export default class UserSettings extends Component {
  render() {
    return (
      <React.Fragment>
        <Header as='h1'>Users Settings</Header>
        123
      </React.Fragment>
    );
  }
}
```

// Файл api.js

```
import React from 'react';
import Frisbee from 'frisbee';
import store from '@/client/store';
import {END_SESSION} from '@/client/store/reducers/auth.reducer';
```

```
const api = new Frisbee({
  baseURI: '/api',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json',
  },
});

// TODO: refactory
store.subscribe(function() {
  const {auth} = store.getState();
  if (auth.token) {
    api.jwt(auth.token);
  }
});

api.interceptor.register({
  response(resp) {
    const {status} = resp;
    if (status === 401) {
      // Force logout when token expired
      store.dispatch({type: END_SESSION});
      // window.location.href = '#/login';
    }
    return resp;
  },
});

// Api Endpoints

api.endpoints = {
  me() {
    return api.get('me');
  },
  getUsers() {
    return api.get('users');
  },
  getStats() {
    return api.get('stats');
  },
  getSubjects() {
    return api.get('subjects');
  }
};

export default api;
```

// Файл app.actions.js

```
import {FETCH_START, FETCH_FINISH} from '../reducers/app.reducer';
// import api from '@/client/services/api';

export function fetchFinish() {
  return dispatch => {
    dispatch({
      type: FETCH_FINISH,
    });
  };
}

export function fetchStart() {
  return dispatch => {
    dispatch({
      type: FETCH_START,
    });
  };
}
```

// Файл auth.actions.js

```
import {
  LOGIN_PENDING,
  LOGIN_SUCCESS,
  LOGIN_ERROR,
  END_SESSION,
} from '../reducers/auth.reducer';
import api from '@/client/services/api';

export function logout() {
  return async dispatch => {
    dispatch({type: END_SESSION});
    return Promise.resolve(true);
  };
}

export function login(body) {
  return async dispatch => {
    try {
      dispatch({type: LOGIN_PENDING});
      const resp = await api.post('auth/login', {body});

      if (resp.ok) {
        dispatch({
          type: LOGIN_SUCCESS, payload: resp.body,
        });
      } else {
        dispatch({
          type: LOGIN_ERROR, payload: resp.err,
        });
      }
      return resp;
    } catch (error) {
      dispatch({type: LOGIN_ERROR});
      return error;
    }
  };
}
```

// Файл app.reducer.js

```
export const FETCH_FINISH = 'FETCH_FINISH';
export const FETCH_START = 'FETCH_START';

const initialState = {
  loading: false,
};

export default function loading(state = initialState, action) {
  switch (action.type) {
    case FETCH_START:
      return {
        ...state,
        loading: true,
      };
    case FETCH_FINISH:
      return {
        ...state,
        loading: false,
      };
    default:
      return state;
  }
}
```