Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мобільного застосунку для підтримки діяльності благодійних організацій»

Виконав: студент 4 курсу групи 2ПІ-20Б
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Лісник В. І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Майданюк В.П.
(прізвище та ініціали)

Рецензент:, к.т.н., доц. каф ОТ Кожемяко А. В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ _____
« 10 » червня 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ліснику Владиславу Ігоровичу

1. Тема роботи – Розробка мобільного застосунку для підтримки діяльності благодійних організацій.

Керівник роботи: Майданюк Володимир Павлович, к. т. н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова розробки JavaScript, Windows 11 23H2, алгоритми авторизації користувачів та захисту їх даних.

4. Зміст текстової частини: вступ; аналіз розвитку мобільних систем для підтримки благодійних організацій та постановка задач розробки, розробка моделей та алгоритмів програмного продукту, розробка програмних модулів реалізації мобільного застосунку, тестування застосунку; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: приклади аналогів; модель роботи застосунку; структурна схема застосунку; блок-схеми алгоритмів роботи застосунку; графічний інтерфейс застосунку; тестування застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Майданюк В.П., к.т.н, доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задачі	14.03.24 – 25.03.24	Викон.
2	Розробка моделей, структури та алгоритмів	26.03.24 – 10.04.24	Викон.
3	Розробка програмного забезпечення	11.04.24 – 28.04.24	Викон.
4	Тестування застосунку	29.04.24 – 15.05.24	Викон.
5	Оформлення матеріалів до захисту БДР	16.05 – 30.05.24	Викон.

Студент _____ (підпис) **В. І. Лісник** (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ (підпис) **В.П. Майданюк** (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.4

Лісник В. І. Розробка мобільного застосунку для підтримки діяльності благодійних організацій: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – Іінженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 114 с. На укр. мові. Бібліогр.: 22 назв; рис.: 27; табл. 2.

У бакалаврській дипломній роботі проведено аналіз стану мобільних систем для благодійних організацій, доведено доцільність розробки власної програмної реалізації, зважаючи на порівняльний аналіз аналогів.

У ході виконання роботи розроблено мобільний застосунок для підтримки діяльності благодійних організацій, що включає в себе модулі для оперування, перегляду, пошуку та фільтрації волонтерських проектів та графічного інтерфейсу. Додаток призначений для спрощення та збільшення ефективності діяльності благодійних організацій.

Мобільний застосунок розроблено з використанням технології React Native та мови програмування JavaScript та середовища Visual Studio Code, для зберігання даних використано сервіс Hygraph із мовою запитів GraphQL. Для створення зручного та інтуїтивно зрозумілого інтерфейсу та захисту даних користувачів було використано фреймворк Expo.

Розроблені програмні модулі можуть бути використані розробниками для створення різних мобільних систем.

Ключові слова: благодійні організації, мобільний застосунок, фреймворк.

ANNOTATION

UDC 004.4

Lisnyk V. I. Development of a mobile application to support the activities of charitable organisations: bachelor's thesis in speciality 121 Software Engineering, educational programme - Software Engineering. Vinnytsia: VNTU, 2024. 114 p. In Ukrainian. Bibliography: 22 titles; Figures: 27; Table 2.

The bachelor's thesis analyses the state of mobile systems for charitable organisations. The expediency of developing our own software implementation was proved, based on a comparative analysis of analogues.

In the course of the work, a mobile application was developed to support the activities of charitable organisations, including modules for operating, viewing, searching and filtering volunteer projects and a graphical interface. The application is designed to simplify and increase the efficiency of charitable organisations.

The mobile application was developed using React Native technology, the JavaScript programming language and the Visual Studio Code environment, and the Hygraph service with the GraphQL query language was used to store data. The Expo framework was used to create a user-friendly and intuitive interface and protect user data.

The developed software modules can be used by developers to create various mobile systems.

Keywords: charitable organisations, mobile application, framework.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний світ благодійних організацій вимагає ефективних інструментів для координації, спілкування та підвищення ефективності виконання процесів. Мобільні застосунки стають невід'ємною частиною цього середовища, надаючи можливість зручного та швидкого доступу до ресурсів та інформації. Однак існуючі рішення часто не відповідають потребам благодійних організацій, оскільки вони не забезпечують достатньої підтримки для специфічних завдань та процесів, які стоять перед організаціями.

У відповідь на ці виклики, актуальним стає розробка програмного забезпечення мобільного застосунку, спеціально адаптованого для потреб благодійних організацій. Такий застосунок міг би значно полегшити організацію та координацію волонтерських заходів, автоматизувати процеси збору даних, комунікації та взаємодії з учасниками [1].

Розробка такого мобільного застосунку є важливим кроком у напрямку покращення діяльності благодійних організацій. Він надасть зручний інструмент для управління ресурсами, волонтерами та проектами, що сприятиме підвищенню ефективності та результативності їхньої роботи. Така система створить сприятливі умови для активної участі волонтерів та розвитку волонтерського руху в цілому. Тому розробка мобільного застосунку для благодійних організацій є не лише актуальною, але й необхідною для вдосконалення їхньої діяльності та забезпечення більш ефективного впливу на соціальні процеси.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення комунікативних можливостей користувачів та організацій у сфері благодійності за рахунок створення мобільного застосунку для підтримки діяльності

благодійних організацій. Застосунок сприятиме ефективній координації заходів, забезпечуючи швидкий доступ до ресурсів та інформації.

Основними задачами дослідження є:

- розробка бази даних для зберігання користувацьких даних та інформації про проекти благодійних організацій;
- розробка інтерфейсу користувача, що забезпечить зручну взаємодію з застосунком та легкий доступ до необхідної інформації;
- розробка модуля авторизації користувачів для забезпечення безпеки та додаткових функцій;
- розробка модуля для перегляду інформації про волонтерські заходи та проекти;
- розробка модуля фінансової допомоги.

Об’єкт дослідження - процес розробки мобільного застосунку для підтримки діяльності благодійних організацій.

Предмет дослідження - методи та програмні засоби розробки мобільного застосунку для підтримки діяльності благодійних організацій.

Методи дослідження. У процесі досліджень використовувались: методи розробки кросплатформного додатку під операційні системи Android та iOS; методи побудови взаємодії застосунку із сервером за допомогою CMS сервісу Hygraph для реалізації швидкої ефективної та гнучкої комунікації клієнту та серверу; методи побудови структури застосунку з підтримкою масштабованості; методи розробки графічного інтерфейсу застосунку за допомогою інструменту Figma задля створення зручного та адаптивного графічного інтерфейсу користувача; методи тестування здля забезпечення коректності роботи застосунку.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод підримки діяльності благодійних організацій, у якому, на відміну від існуючих, для завантаження контенту використано CMS сервіс Hygraph, та мобільний застосунок для комунікації з

спонсорами, що дозволило підвищити рівень взаємодії благодійних організації та спонсорів.

2. Подальшого розвитку отримав метод створення мобільних застосунків для підримки діяльності благодійних організацій, у якому, на відміну від існуючих, використано технологію React Native, фрейворк Expo та мову програмування JavaScript, що спрощує розробку та забезпечує кросплатформність застосунку.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби мобільного застосунку для підтримки діяльності благодійних організацій.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: обґрунтовано доцільність розробки мобільного за стосунку для автоматизації процесів діяльності волонтерських організацій [2]; обгрунтовано використання cms-сервісів для розробки серверної частини мобільних застосунків [3].

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)»; LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024).

Публікації. Основні результати досліджень опубліковано в 2 наукових працях у матеріалах конференцій.

1 АНАЛІЗ РОЗВИТКУ МОБІЛЬНИХ СИСТЕМ ДЛЯ ПІДТРИМКИ БЛАГОДІЙНИХ ОРГАНІЗАЦІЙ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз стану мобільних систем створених для підтримки благодійних організацій

У сучасному світі спостерігається зростання популярності застосунків, що спрямовані на покращення діяльності благодійних організацій. Ці тенденції відображають, що організації віддають перевагу зручності, доступності та швидкості у виконанні своїх завдань. Мобільні застосунки дозволяють волонтерам отримувати доступ до інформації та здійснювати роботу з будь-яких пристроїв, що має підключення до Інтернету, що є вкрай важливим в сучасному світі.

Сьогодні, існують різні програмні застосунки для підтримки діяльності благодійних організацій, проте вони часто мають обмежену функціональність та не забезпечують врахування всіх потреб організацій. Більшість існуючих застосунків є обмеженими в базових функціх та мають складний інтерфейс або неінтуїтивні функції, що ускладнює їх використання для користувачів. Такі застосунки недостатньо враховують різноманітні потреби благодійних організацій та спонсорів.

В результаті, розвиток більш інтегрованих та функціональних мобільних застосунків може сприяти підтримці діяльності благодійних організацій, а також забезпечити ширший спектр сервісів, які відповідають конкретним потребам організацій. Такі застосунки можуть включати в себе функції підтримки діяльності, взаємодію з волонтерами та збір інформації.

Існуючі рішення показують, що зараз існують обмежені програмні застосунки для підтримки діяльності благодійних організацій. Застосунки мають вагомі недоліки, вони часто мають обмежений функціонал, що не враховує всіх потреб організацій. Більшість з них мають складний інтерфейс або неінтуїтивні функції, що ускладнює їх використання. Також, багато застосунків обмежені лише до передачі базової інформації, не забезпечуючи повноцінної взаємодії між

користувачами. Крім того, деякі застосунки не забезпечують достатнього рівня безпеки особистих даних користувачів.

Для вирішення цих проблем необхідно розробити більш інтегрований та функціональний мобільний застосунок. Він має бути спрямований на полегшення діяльності організацій та забезпечення їх ефективного функціонування. Застосунок повинен включати широкий функціонал, відповідати потребам організацій та бути легкими у використанні. Також, він повинен забезпечувати безпеку та конфіденційність особистих даних користувачів.

В результаті, визначено, що є потреба у створенні власного мобільного застосунку, який сприятиме покращенню діяльності благодійних організацій, полегшить комунікацію та співпрацю між їх учасниками і забезпечить захист особистих даних.

1.2 Порівняльний аналіз аналогів

Під час проведення досліджень аналогів до розроблюваного програмного застосунку було виявлено, що подібних додатків наразі існує досить небагато. В якості аналогів було обрано веб-сайти «СпівДія», «VolunteerUP» та «volunter.org».

«VolunteerUP» - це веб-сайт, спрямований на полегшення волонтерської діяльності [4]. Основною метою платформи є забезпечення доступу до різноманітних волонтерських можливостей та сприяння залученню нових волонтерів до різних соціальних та благодійних проектів.

Основні особливості «VolunteerUP»:

Платформа дозволяє волонтерам швидко знаходити події, проекти та акції, які відповідають їхнім інтересам та готовності допомагати.

Волонтери можуть зареєструватися на бажані події безпосередньо через платформу, що спрощує процес організації та участі у волонтерських заходах.

Веб-застосунок «VolunteerUP» представлено на рисунку 1.1.

Рисунок 1.1 – Веб-застосунок «VolunteerUP»

Організатори можуть розміщувати оголошення про свої події, збираючи команди волонтерів та координуючи їх діяльність. Волонтери можуть спілкуватися між собою та обмінюватися досвідом.

«СпівДія» - це онлайн-платформа, яка створена для залучення волонтерів до участі у соціальних та культурних проектах [5]. Головна мета Спів дія полягає в тому, щоб кожен волонтер міг знайти проєкт, який відповідає його інтересам та можливостям.

Веб-застосунок представлено на рисунку 1.2.

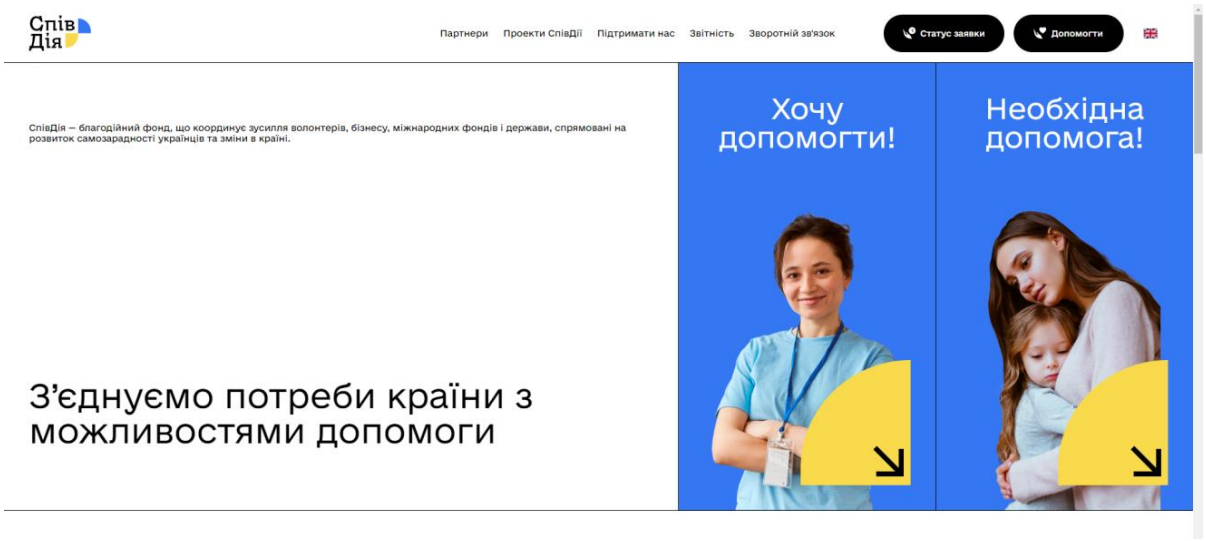

Рисунок 1.2 – Веб-застосунок «СпівДія»

Основні функції «СпівДія»:

Платформа пропонує широкий вибір проєктів для участі, від благодійних акцій до екологічних заходів та культурних подій.

Волонтери можуть швидко зареєструватися на бажані проєкти та отримати всю необхідну інформацію щодо участі.

Волонтери можуть обговорювати свої ідеї, ділитися досвідом та координувати свою діяльність через спеціальні форуми та чати.

Волонтери можуть відстежувати свій внесок у різні проєкти та отримувати звіти про результати своєї діяльності.

«ВолонтерОрг» - це онлайн-платформа, яка об’єднує волонтерів та організації, що потребують допомоги [6]. Основна мета платформи - сприяти залученню волонтерів до різних соціальних та благодійних ініціатив.

Веб-застосунок представлено на рисунку 1.3.

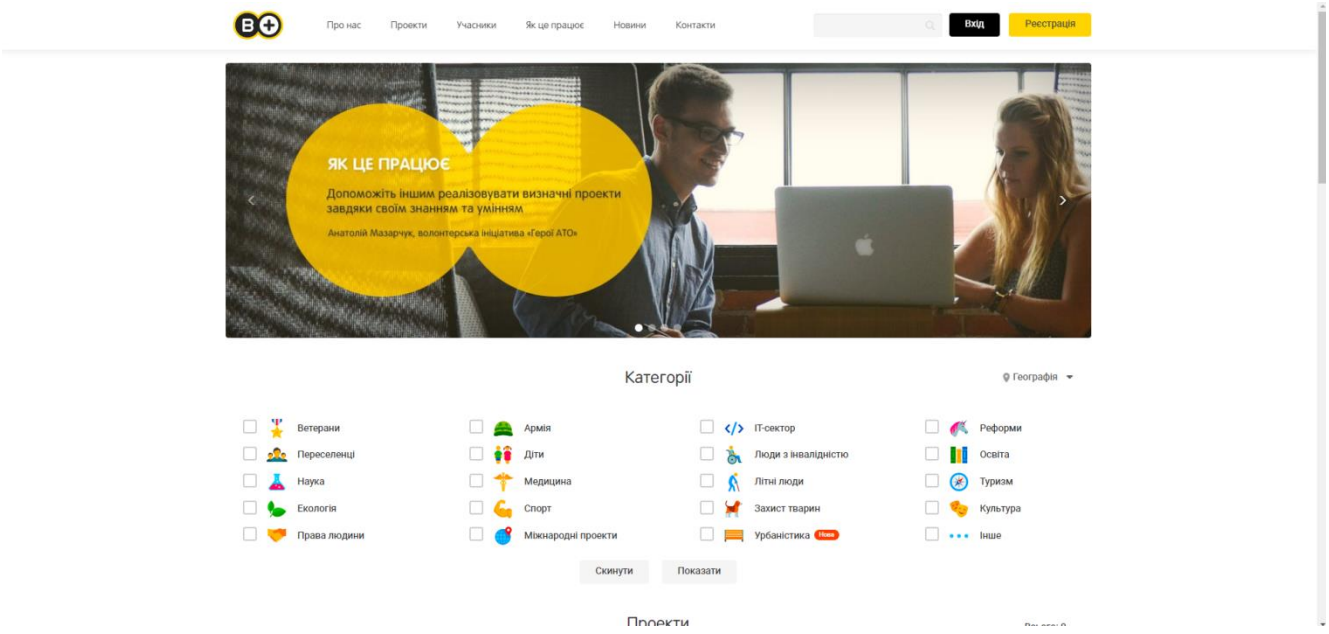

Рисунок 1.3 – Веб-застосунок «ВолонтерОрг»

Основні можливості «Волонтер.орг»:

Волонтери можуть шукати волонтерські можливості за різними критеріями, включаючи тематику, регіон та тип діяльності.

Волонтери можуть зареєструватися на бажані проєкти та отримати всю необхідну інформацію щодо участі.

Організатори можуть розміщувати інформацію про свої проєкти та запрошувати волонтерів до участі.

Волонтери можуть спілкуватися з організаторами та іншими волонтерами, обговорювати плани та ділитися досвідом через спеціальні чати та форуми.

Щоб підсумувати порівняння аналогів приведено таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	VolunteerUP	СпівДія	ВолонтерОрг	Власна реалізація
Можливість авторизації	+	+	+	+
Розбиття проектів на категорії	-	-	+	+
Відстеження прогресу	-	+	-	+
Можливість разової фінансової підтримки	-	+	+	+
Інтеграція з соц. мережами	-	-	-	+

Відповідно до таблиці порівняльних характеристик розробка власних мобільного застосунку для підтримки діяльності благодійних організацій є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити розширені можливості.

Отже, розробка мобільного застосунку для підтримки діяльності благодійних організацій відкриває широкі можливості для покращення благодійності, а саме, волонтерства, завдяки ефективною реалізацією проєктної діяльності з додатковим функціоналом.

1.3 Аналіз методів розв’язання задачі

Розробка мобільного застосунку для підтримки діяльності благодійних організацій вимагає ретельного аналізу та вибору ефективних методів розв'язання.

Цей процес можна поділити на кілька етапів:

Для початку проводиться аналіз існуючих методів, які вже використовуються в сфері розробки мобільних застосунків для благодійних організацій. Цей аналіз включає огляд програмних продуктів на ринку, які пропонують інтерактивні системи для волонтерів. Також досліджуються наукові статті, публікації та інші джерела, що стосуються розробки застосунків для благодійних організацій.

Далі, проводиться дослідження потреб та вимог потенційних користувачів. Це включає аналіз типових сценаріїв використання, визначення основних функціональних потреб користувачів та їхніх очікувань від мобільного застосунку для благодійних організацій. Для цього можуть проводитися опитування та фокус-групи серед представників цільової аудиторії.

Наступним кроком є вибір технологій для розробки мобільного застосунку. Це включає вибір платформи розробки, мови програмування, бази даних та інших технічних аспектів. Також важливо врахувати можливість масштабування та підтримки платформ.

Після вибору технологій розроблюється архітектура мобільного застосунку. Визначаються структура, модулі та компоненти. Особлива увага приділяється розробці зручного та ефективного інтерфейсу користувача, який відповідає потребам цільової аудиторії.

Після проектування архітектури переходимо до розробки мобільного застосунку. Це включає програмування функціоналу та розробку інтерфейсу. Після завершення розробки проводиться тестування для перевірки правильності та стабільності роботи застосунку.

Останнім етапом є впровадження мобільного застосунку серед волонтерської спільноти та підтримка користувачів. Це включає вирішення можливих проблем, оновлення програмного забезпечення та відповідь на запитання користувачів.

В результаті, розглянувши кожен метод, переваги та недоліки, було вирішено розробити мобільний застосунок для підтримки діяльності благодійних організацій, який враховуватиме основні вимоги та потреби цільової аудиторії.

1.4 Постановка задачі

Для успішної розробки мобільного застосунку для підтримки діяльності благодійних організацій необхідно вирішити наступні завдання:

1. Дослідження та аналіз потреб та очікувань спільнот щодо мобільного застосунку:

– Провести аналіз потреб та очікувань волонтерської спільноти щодо мобільного застосунку.

– Виявити ключові вимоги та функціональність, які повинні бути включені в застосунок.

2. Визначення основних функціональних потреб та встановлення пріоритетів користувачів:

– Сформулювати основні функції та функціональні вимоги до застосунку.

– Встановити пріоритети за допомогою аналізу важливості функцій для користувачів.

3. Вибір платформи розробки:

– Обрати платформу для розробки мобільного застосунку (наприклад, Android, iOS або кросплатформна).

– Врахувати особливості та потреби цільової аудиторії при виборі платформи.

4. Розробка структури додатку:

– Спроектувати структуру додатку, включаючи його модулі та компоненти.

– Забезпечити оптимальну функціональність та легкість використання додатку.

5. Написання коду для реалізації необхідної функціональності:

– Розробити програмний код, який відповідає визначеним вимогам та функціональності.

– Забезпечити ефективну роботу додатку та відповідність стандартам програмування.

6. Розробка інтерфейсу користувача:

– Створити інтуїтивно зрозумілий та зручний інтерфейс для користувачів застосунку.

– Врахувати дизайн-принципи та потреби користувачів для покращення взаємодії з додатком.

7. Проведення комплексного тестування:

– Виконати тестування додатку на різних платформах для перевірки стабільності та ефективності його роботи.

– Виявити та виправити будь-які помилки або недоліки у роботі додатку.

В результаті, виконання цих завдань дозволить створити високоякісний мобільний застосунок, який відповідає потребам благодійних організацій та забезпечує їх підтримку.

1.5 Висновки

У першому розділі розглянуто стан мобільних систем для підтримки діяльності благодійних організацій, застосунки СпівДія, ВолонтерОрг та VolunteerUp.

Проаналізувавши переваги та недоліки існуючих програмних засобів, було сформульовано завдання для подальшої розробки власного мобільного застосунку для підтримки діяльності благодійних організацій. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації сформовано перелік задач, які необхідно виконати для розробки власної мобільної системи.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Аналіз вхідних даних системи є важливим етапом у розробці мобільного застосунку для підтримки діяльності благодійних організацій. Цей процес полягає у ретельному розгляді та розумінні різних типів інформації, яка може надходити до системи.

Особиста інформація користувача є основним типом вхідних даних, яку варто розглянути. Ця інформація включає «Google» аккаунт користувача. Ці дані не лише допомагають ідентифікувати користувача, та й створюють можливість прямого зворотнього зв’язку користувача із розробником.

Також потрібно враховувати дані про пропозиції та скарги користувачів. Ця інформація є дуже цінною, оскільки дозволяє швидко виявити проблеми та недоліки застосунку. Ретельний аналіз цих даних дозволяє не лише вчасно реагувати на проблеми, але й вдосконалювати застосунок та підвищувати зацікавленість у ньому користувачів.

В результаті, аналіз вхідних даних допомагає зрозуміти потреби користувачів, оптимізувати використання ресурсів та покращувати якість. Це важливий етап у процесі розробки веб додатку, оскільки від правильного аналізу даних залежить успішність та ефективність роботи програмного забезпечення.

2.2 Розробка моделі системи

Для кращого розуміння роботи модулів мобільного застосунку для підтримки діяльності благодійних організацій було вирішено розробити модель роботи системи на прикладі UML діаграм [7].

Діаграма діяльності – це графічне зображення послідовності дій або процесів у системі [8]. Вона дозволяє візуалізувати, як об'єкти взаємодіють між собою в рамках певного процесу або діяльності. Кожна діяльність

представляється у вигляді прямокутника, що містить назву дії, і вказує на конкретну операцію чи функцію, що виконується.

Велика перевага діаграм діяльності полягає в їх зрозумілості і простоті сприйняття, що дозволяє швидко зрозуміти послідовність подій. Вони широко використовуються при аналізі та проектуванні бізнес-процесів, системного аналізу та в управлінні проектами.

Діаграми діяльності допомагають ідентифікувати можливі проблеми в процесах та здійснювати їх вдосконалення.

Діаграму діяльності мобільного застосунку для підтримки благодійних організацій відображено на рисунку 2.1.

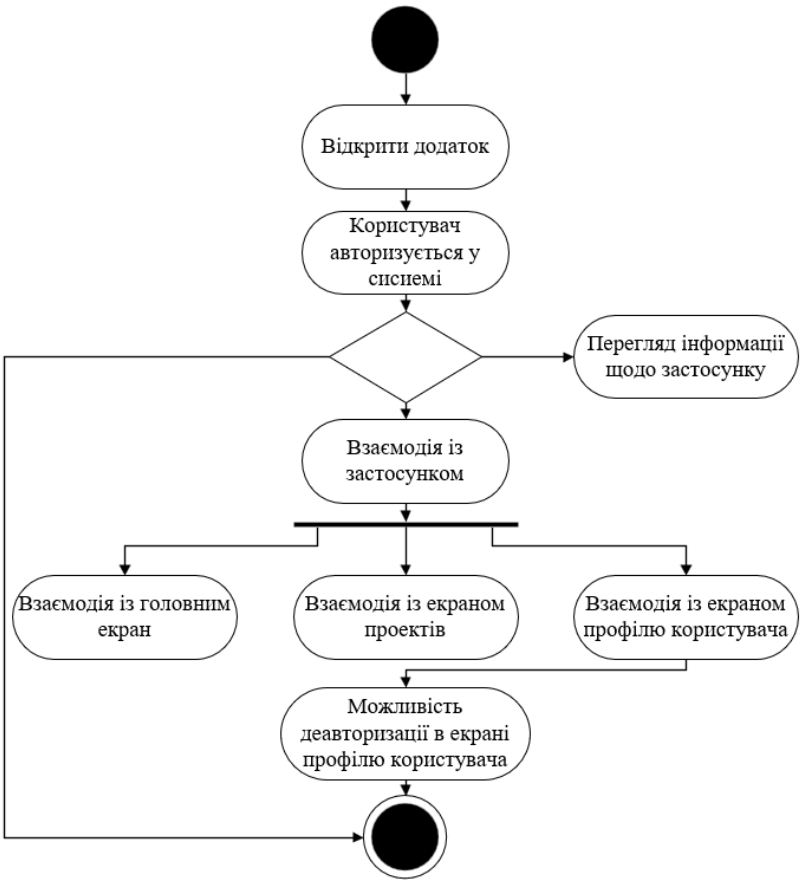

Рисунок 2.1 – Діаграма діяльності мобільного застосунку

Згідно вказаної діаграми користувач для початку взаємодії із програмним застосунком повинен авторизуватися у системі, у іншому разі користувач зможе переглянути інформацію про застосунок або вийти.

Наступним кроком буде перенесення користувача на головний екран застосунку. При переході на головний екран застосунку перед користувачем відкриється навігаційне меню для кращого розуміння основних екранів застосунку.

Після цього користувач матиме змогу перейти до потрібного йому екрана: «Головний екран», «Екран проектів», або «Екран профілю користувача».

Після успішної взаємодії із функціями застосунку, користувач зможе деавторизуватись на екрані профілю користувача та закінчити роботу із застосунком.

Розробка діаграми діяльності була виконана у програмному забезпеченні Microsoft Visio [9].

2.3 Розробка структури системи

Для розробки мобільного застосунку повинна бути розроблена його структура, поділена на загальні модулі, які визначатимуть послідовність дій користувача в застосунку. Подальшого розвитку отримав метод підримки діяльності благодійних організацій, у якому, на відміну від існуючих, для завантаження контенту використано CMS сервіс Hygraph, та мобільний застосунок для комунікації з спонсорами, що дозволило підвищити рівень взаємодії благодійних організації та спонсорів.

Згідно цього, перш за все, користувач проходить авторизацію або переглядає інформацію про застосунок, після цього, користувач потрапляє на головний екран застосунку. Далі, користувач матиме можливість перейти до рекламних інтеграцій, категорій проектів, списку останніх проектів, або ж здійснити пошук проектів за назвою.

Також, перед користувачем візуалізовано меню навігації, що складається з трьох сторінок: «Головна», «Проекти» та «Профіль. У випадку, якщо користувач залишиться на головному екрані, він бачитиме навігаційне меню.

Крім того, на головному екрані є доступними корисні функції: слайдера з рекламними інтеграціями, категоріями, та останніми проектами. Такий підхід

дозволяє користувачу швидко знаходити необхідну інформацію та зручно користуватися додатком.

При переході на екран «Проекти» користувач побачить усі створені волонтерські проекти. Таким чином, запускається модуль перегляду проектів та доступ до функцій.

Якщо користувач перейде до екрану «Профіль», він матиме можливість переглянути свої данні, фото, навігаційне меню, також користувач зможе надіслати відгуки та побажання для покращення роботи застосунку і деавторизуватись із системи.

Також, було створено діаграму, на якій відображено структуру мобільного застосунку для підтримки діяльності благодійних організацій.

Структуру мобільного застосунку відображено на рисунку 2.2 .

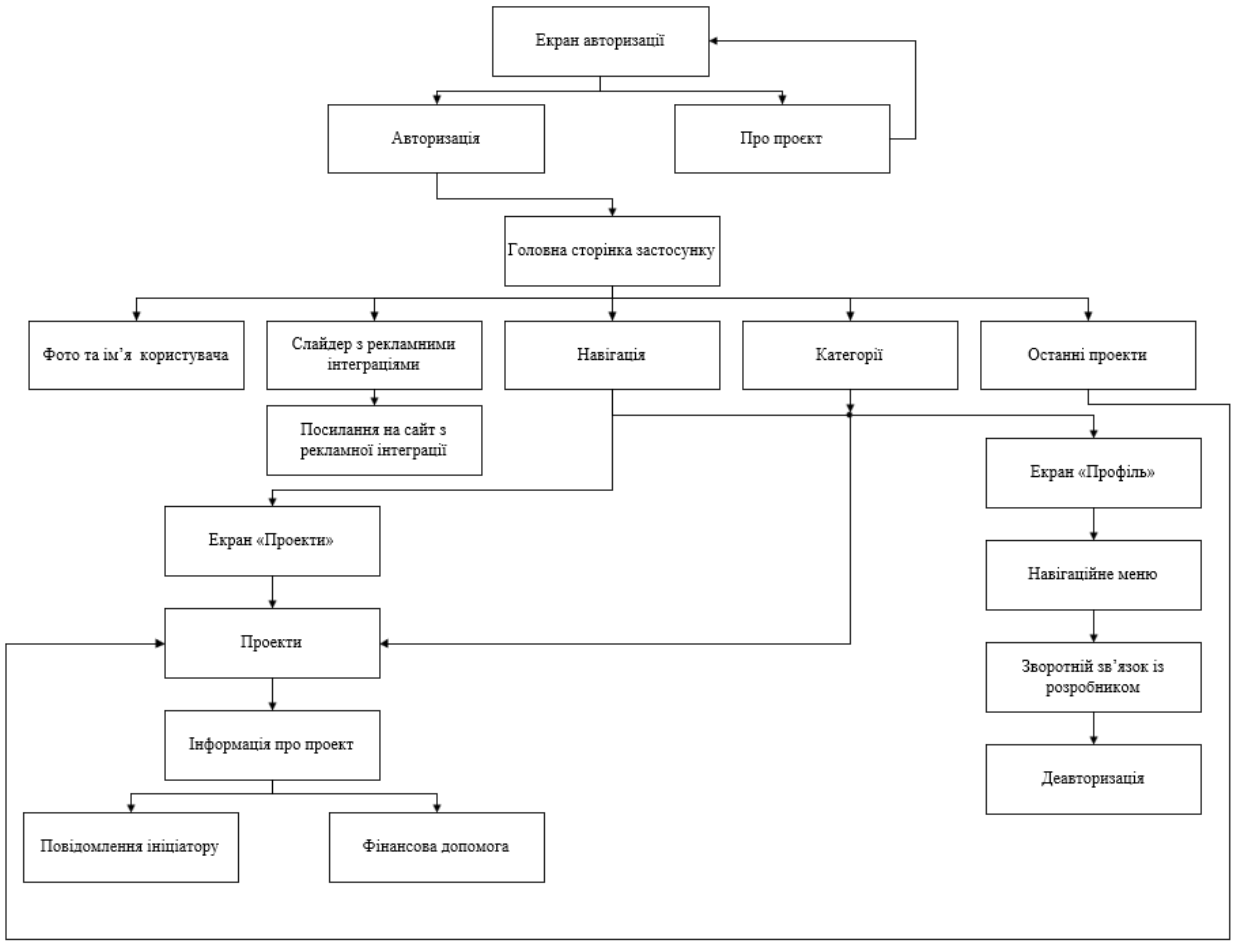

Рисунок 2.2 – Структура мобільного застосунку

Створення архітектури мобільного застосунку включає в себе визначення структури застосунку та послідовності переходів між його екранами.

Отже, визначення структури застосунку є важливою складовою процесу розробки, що дозволяє забезпечити зручну та інтуїтивну навігацію для користувачів.

2.4 Розробка алгоритму захисту даних користувачів

Для забезпечення конфіденційності та безпеки даних користувачів у мобільних додатках було розроблено алгоритм захисту, базований на бібліотеці Expo secure-store. Цей алгоритм складається з двох основних механізмів: SecureStore та Crypto, які спільно забезпечують шифрування та зберігання конфіденційних даних.

SecureStore використовується для зберігання конфіденційних даних у вигляді пар ключ-значення на мобільному пристрої. Ключі та значення зберігаються у вигляді рядків, що дозволяє зручно організувати доступ до них.

Crypto відповідає за шифрування та розшифрування даних перед їхнім збереженням у SecureStore. Використовується асиметричне шифрування RSA для забезпечення високого рівня безпеки.

Алгоритм роботи виглядає наступним чином:

При запуску додатка генерується пара ключів RSA - публічний та приватний. Публічний ключ використовується для шифрування даних, а приватний - для їхнього розшифрування.

Конфіденційні дані перед збереженням у SecureStore шифруються з використанням публічного ключа.

Зашифровані дані зберігаються у SecureStore у вигляді пар ключ-значення, забезпечуючи їхню безпеку під час зберігання на пристрої.

При необхідності отримати дані, вони зчитуються з SecureStore.

Зчитані зашифровані дані розшифровуються з використанням приватного ключа.

Розшифровані дані повертаються у додаток, готові до використання.

Блок-схема алгоритму захисту даних користувачів зображена на рисунку 2.3.

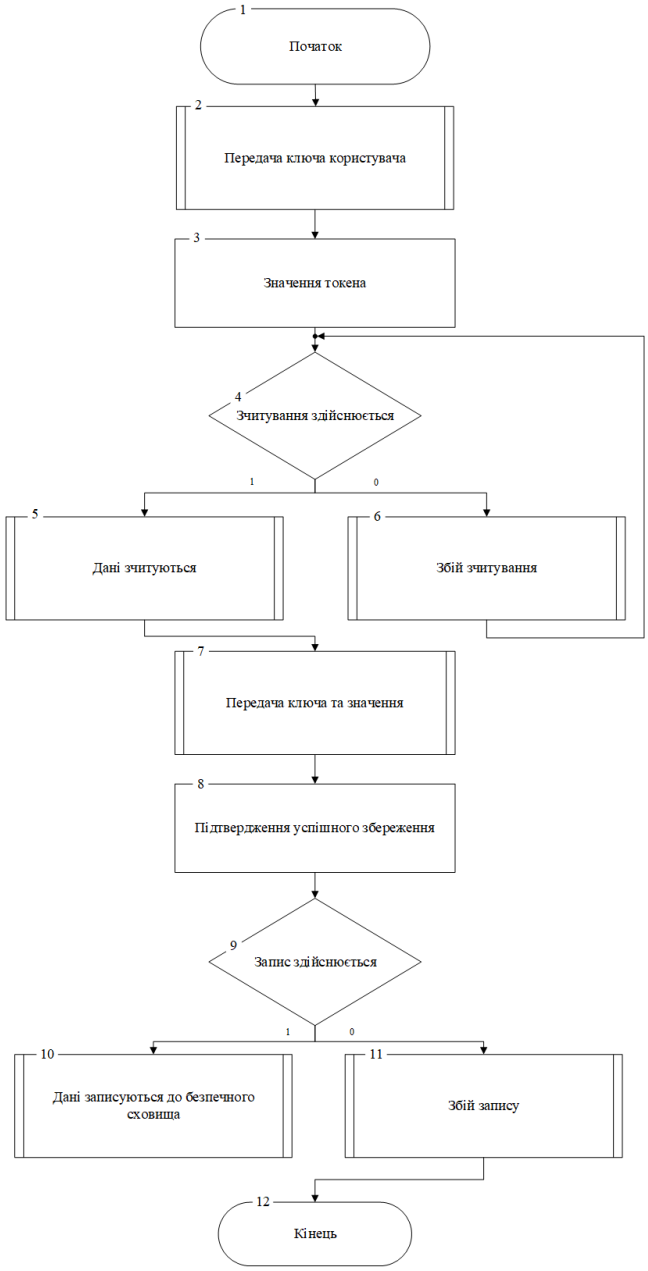

Рисунок 2.3 – Блок-схема алгоритму захисту даних користувачів

Отже, алгоритм забезпечує високий рівень захисту конфіденційних даних користувачів, не лише шифруючи їх, але й забезпечуючи безпечне зберігання на мобільному пристрої, що робить його ідеальним вибором для застосування у мобільних застосунках.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмних засобів. Було розроблено модель роботи програмного застосунку за допомогою UML діаграм та визначено структурну схему мобільного застосунку. Також, було розроблено алгоритм захисту даних користувачів мобільного застосунку для підтримки діяльності благодійних організацій.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Варіантний аналіз і обгрунтування вибору засобів для реалізації програмного забезпечення

При розробці програмного забезпечення мобільного застосунку для пітримки діяльності благодійних організацій важливо обрати найбільш оптимальні технології для розробки кожного модуля програмних засобів.

Подальшого розвитку отримав метод створення мобільних застосунків для підримки діяльності благодійних організацій, у якому, на відміну від існуючих, використано технологію React Native, фрейворк Expo та мову програмування JavaScript, що спрощує розробку та забезпечує кросплатформність застосунку.

React Native – це фреймворк для розробки мобільних додатків, що дозволяє використовувати JavaScript та React для побудови кросплатформених додатків [10]. Він дозволяє розробникам створювати додатки для iOS та Android, використовуючи одну кодову базу. React Native використовує компоненти, що дозволяє просто створювати інтерфейси додатків. Один з ключових плюсів React Native – це можливість гарячого перезавантаження, що дозволяє швидко бачити зміни в коді без перекомпіляції додатку [11].

Фреймворк Expo – це надбудова над React Native, яка надає додаткові можливості та спрощує процес розробки мобільних додатків [12]. Використання Expo дозволяє швидше розпочати роботу над проектом, оскільки він автоматично конфігурує проект та забезпечує доступ до багатьох корисних бібліотек та інструментів. Один з головних плюсів Expo – це можливість розробляти додатки без необхідності налаштовувати середовище розробки для кожної платформи окремо [13]. В Expo також є багато готових компонентів, які дозволяють швидко створювати складні інтерфейси. Expo надає зручний інтерфейс командного рядка для роботи з проектом, включаючи можливість запуску додатків на емуляторах або реальних пристроях. Крім того, Expo надає можливість легко взаємодіяти з нативними API, що дозволяє розширювати

функціональність додатків. Ще однією перевагою Expo є можливість швидко розгорнути додаток для тестування та відладки через інтернет, що спрощує співпрацю з іншими розробниками та тестерами. React Native та фреймворк Expo мають низку значних переваг, які роблять їх важливими інструментами для розробки кросплатформених мобільних додатків. По-перше, вони дозволяють використовувати один і той же код для розробки додатків як для iOS, так і для Android, що значно економить час та ресурси розробки. Для команд, які мають обмежений бюджет або час, це особливо цінно. Друга важлива перевага полягає в тому, що React Native і Expo використовують JavaScript, мову, яка є дуже поширеною та зрозумілою для багатьох розробників, що спрощує набір команди для проекту.

Ще одна перевага – це відкритість та активна спільнота розробників, яка підтримує ці технології. За допомогою документації, статей та інших ресурсів від спільноти, розробники можуть легко знайти рішення для своїх проблем або поради щодо розробки.

Крім того, React Native та Expo забезпечують можливість гарячого перезавантаження, що дозволяє бачити зміни в коді миттєво на віддаленому пристрої без перекомпіляції. Це значно прискорює процес розробки та відладки.

Ще однією важливою перевагою є широкий вибір готових компонентів та бібліотек, доступних для React Native та Expo, що дозволяє прискорити розробку та зменшити кількість написаного коду.

Використання Expo спрощує підготовку середовища для розробки, що особливо корисно для початківців, а також забезпечує швидке розгортання та тестування додатків. Також варто зазначити, що Expo дозволяє взаємодіяти з нативними функціями пристрою без необхідності вивчати платформено-залежні API.

Загалом, React Native та Expo – це потужні технології, які дозволяють розробникам швидко створювати кросплатформні мобільні додатки з високою продуктивністю та зручним інструментарієм.

Для обумовлення серверної частини мобільного застосунку було використано CMS сервіс Hygraph та мову запитів GraphQL.

GraphQL – це мова запитів та середовище виконання запитів для взаємодії з серверами даних [14]. Одна з головних переваг GraphQL полягає в тому, що вона дозволяє користувачам запитувати саме ті дані, які їм потрібні, без зайвих запитів або отримання зайвих даних [15]. У порівнянні з традиційним REST API, де кожен ендпоінт відповідає за певний ресурс або операцію, GraphQL дозволяє користувачам визначати структуру та обсяг даних, які вони отримують.

Ще однією важливою перевагою GraphQL є те, що вона дозволяє користувачам здійснювати запити за декількома зв'язаними даними одночасно, з одного запиту. Це називається "запитами на глибокий об'єкт", що дозволяє ефективно отримувати всі необхідні дані за один раз, замість кількох окремих запитів.

GraphQL також має вбудовану систему типів даних, що дозволяє описувати структуру даних, що повертається сервером. Це робить GraphQL дуже гнучкою та підтримує автоматичну валідацію запитів, що дозволяє виявляти помилки на етапі розробки. Крім того, GraphQL дозволяє використовувати директиви для модифікації та фільтрації даних, що отримуються в результаті запиту. Це дає можливість динамічно змінювати поведінку запитів на основі умов або контексту. Вирішальним, є те, що GraphQL дозволяє реалізовувати підписку на дані, що дозволяє клієнтам отримувати оновлення даних в реальному часі, без необхідності постійно повторювати запити. Це особливо важливо для реал-тайм застосунків, таких як мобільного застосунку для підтримки діяльності благодійних організацій.

Загалом, GraphQL – це потужний інструмент для створення ефективних та гнучких API, які дозволяють користувачам отримувати точно ті дані, які їм потрібні, та працювати з ними ефективно.

Отже, вибір технологій для розробки кросплатформного мобільного застосунку для підтримки діяльності благодійних організацій, а також мови запитів GraphQL, має вирішальне значення для створення ефективного і точного

інструменту для обумовлення підтримки діяльності благодійних організацій. Поєднання цих інструментів надає комплексний підхід до розробки, що дозволяє створювати продукт з високим рівнем функціональності, надійності та зручного користування.

3.2 Розробка модуля авторизації користувачів

При розробці модуля для авторизації користувачів важливо враховувати ряд технічних аспектів. Варто зауважити, що має бути забезпечений значний рівень безпеки даних користувачів. Авторизація користувачів у застосунку розроблена із допомогою Clerk технології. Авторизація через Clerk в React Native Expo застосунок – це спосіб забезпечити безпеку та зручність для користувачів [16]. Спершу треба встановити спеціальні пакети для цього. Потім налаштувати Clerk SDK – він допомагає застосунку спілкуватися з Clerk. Clerk має готові компоненти для входу, реєстрації та інших дій.

Відображення авторизованих користувачів здійснюється у самій системі Clerk, що відображено на рисунку 3.1.

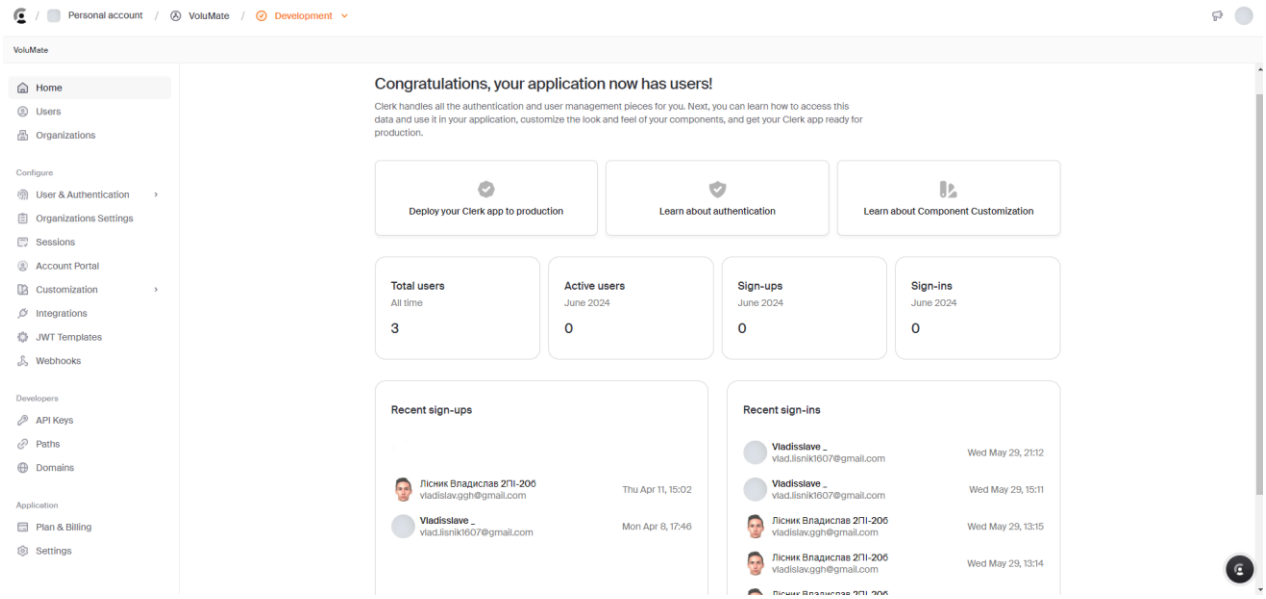

Рисунок 3.1 – Система Clerk для обліку користувачів застосунку

Спочатку здійснюється перевірка, чи користувач увійшов у систему, і відповідно показує навігаційний контейнер або екран входу.

Якщо користувач увійшов, він бачить основний інтерфейс додатку; якщо ні - екран для входу.

Для відкриття сторінки авторизації за допомогою облікового запису Google було використано створення відповідного запита в браузер за допомогою React Hook.

React Hook використовує бібліотеку expo-web-browser для попереднього нагріву та охолодження веб-браузера. При імпортуванні React та expo-web-browser, створюється користувацький хук useWarmUpBrowser.

Завантаження браузера із вибором облікового запису для авторизації відображено на рисунку 3.2.

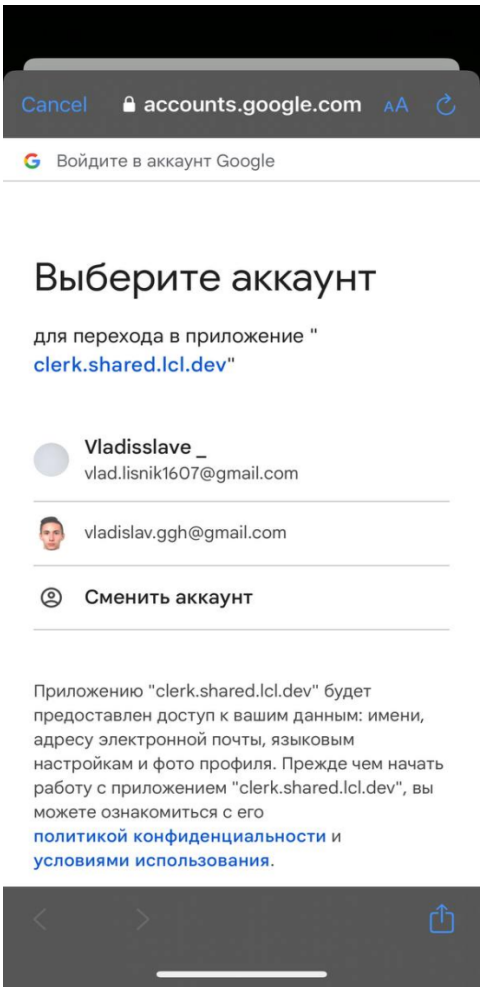

Рисунок 3.2 – Завантаження браузера із вибором облікового запису

Тут використовується ефект React, що автоматично викликається після рендерингу компонента. В цьому ефекті викликається функція WebBrowser.warmUpAsync(), яка попередньо завантажує браузер для швидшого відкриття веб-сторінок.

Коли компонент демонтується, викликається функція WebBrowser.coolDownAsync(), що охолоджує браузер. Це допомагає зекономити ресурси пристрою. Загалом, цей хук допомагає оптимізувати використання веб-браузера в застосунку.

Також, щоб користувач міг без виникнення помилок авторизовуватись в застосунку було розроблено спеціалізовані функції.

Для їх розробки було використано компонент Login в React Native, який використовує OAuth для авторизації користувачів через Google. Він використовує функцію WebBrowser.maybeCompleteAuthSession() для завершення OAuth сесії. Також використовується хук useWarmUpBrowser() для попереднього нагріву браузера.

Далі використовується хук useOAuth з параметром oauth_google для ініціалізації OAuth потоку. Коли користувач натискає кнопку, викликається функція onPress, яка запускає OAuth потік.

Якщо сесія успішно створена, вона стає активною. Якщо ж ні, то можна використовувати signIn або signUp для наступних кроків, таких як багатофакторна автентифікація. У разі помилки виводиться повідомлення про помилку в консоль.

Основним в авторизації користувачів є захист їх даних. Clerk також дозволяє захистити приватні сторінки від несанкціонованого доступу. Це допомагає забезпечити безпеку та легкість використання для користувачів. Також, користувачі не повинні запам'ятовувати нові дані для кожного додатку. Все є в одному місці, і це є досить зручним доповненням.

Дані користувачів при авторизації зразу ж заносяться в безпечне сховище перетворюючи їх у токени [17].

Деавторизація користувачів відбувається за допомогою простого завершення їх сессії для повторної авторизації.

Отже, система авторизації користувача працює самостійно. Це значно полегшує процес авторизації, роблячи його швидшим і зручнішим для користувачів. Такий підхід спрощує доступ користувачів до застосунку і забезпечує більшу зручність у використанні сервісу.

3.3 Розробка модуля для перегляду інформації про волонтерські заходи та проекти

Головною задачею модуля перегляду інформації про волонтерські заходи та проекти є обумовлення взаємодії користувачів із волонтерськими проектами, їх фінансування та покращення.

Для реалізації цієї задачі було створено файл навігатор та декілька файлів взаємодії з проектами.

Не менш важливим є файл який реалізує відображення останніх проектів на головному екрані застосунку.

Файл являє собою компонент ProjectListItemSmall в React Native, який відображає інформацію про проект. Компонент приймає об’єкт project як вхідний параметр і відображає його зображення, назву, контактну особу та категорію.

Коли користувач натискає на компонент, він переходить на сторінку деталей проекту. Стилі для компонента визначаються в об’єкті styles, який створюється за допомогою StyleSheet.create().

Вся область реалізації відображення останніх проектів на головному екрані застосунку являє собою область з ефектом дотику, що переносить користувача на сторінку з інформацією щодо проекту.

Уся інформація зображена на області останніх проектів дістається з бази даних за допомогою запиту, який являє собою перелік потрібної користувачу інформації.

Також задля гарного візуального сприйняття було використано та застосовано велика кількість стилів.

Для реалізації отримання списку проектів з бази даних було створено файл ProjectList.jsx, що за допомогою запиту отримує список усіх проектів з їх наповненням.

Список проектів знаходиться у базі даних cms-сервісу Hygraph, інтерфейс сервісу із завантаженими проєктами відображено на рисунку 3.3.

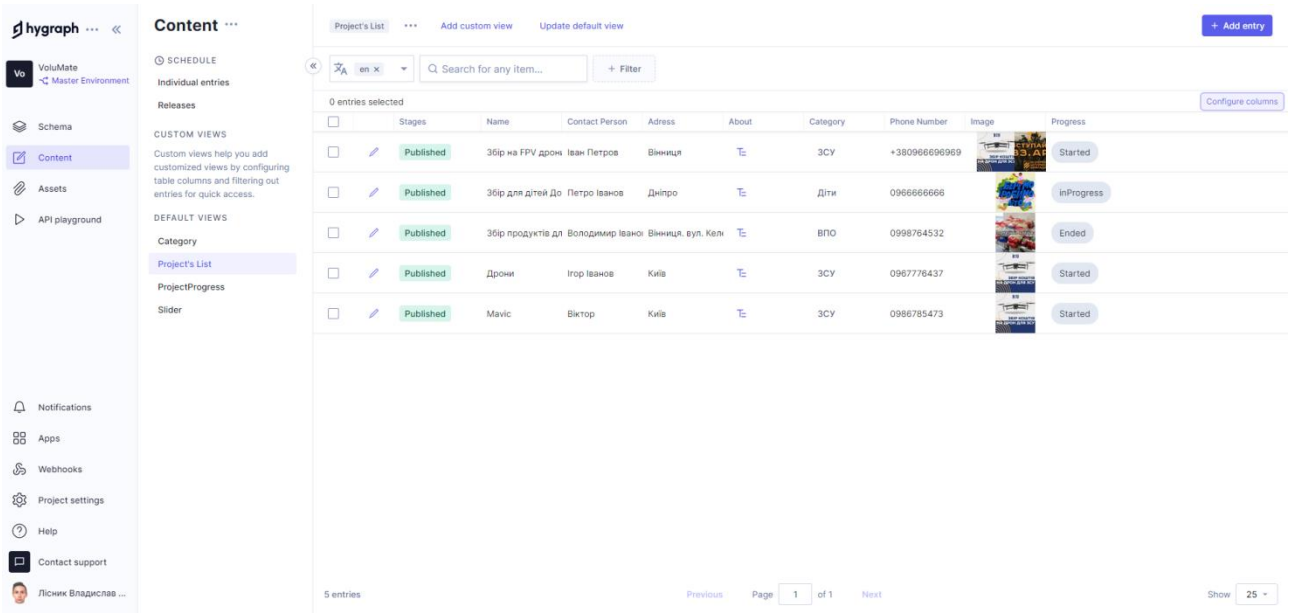

Рисунок 3.3 – Інтерфейс сервісу для управління контентом із завантаженими проєктами

Екран з проектами складається зі списку усіх створених проектів що мають статус «Розпочатий» або «Активний», закінчені проекти автоматично видаляються зі списку.

Щоб реалізувати надсилання повідомлення до ініціатора проекту було використано бібліотеку Linking в React Native.

Відправку повідомлень реалізовано за допомогою схеми mailto - це URI-схема, яка використовується для відкриття стандартного поштового клієнта на пристрої з можливістю створення нового листа. У React Native для відкриття поштового клієнта та створення нового листа за допомогою mailto,

використовується метод Linking. Це відкриє поштовий клієнт пристрою з вже заповненим адресою електронної пошти.

Для реалізації фінансової допомоги було використано технологію Stripe React Native. Stripe - це платіжна платформа, яка надає інструменти для прийому платежів через Інтернет.

Інтерфейс платіжної платформи з налаштованим проєктом відображено на рисунку 3.4.

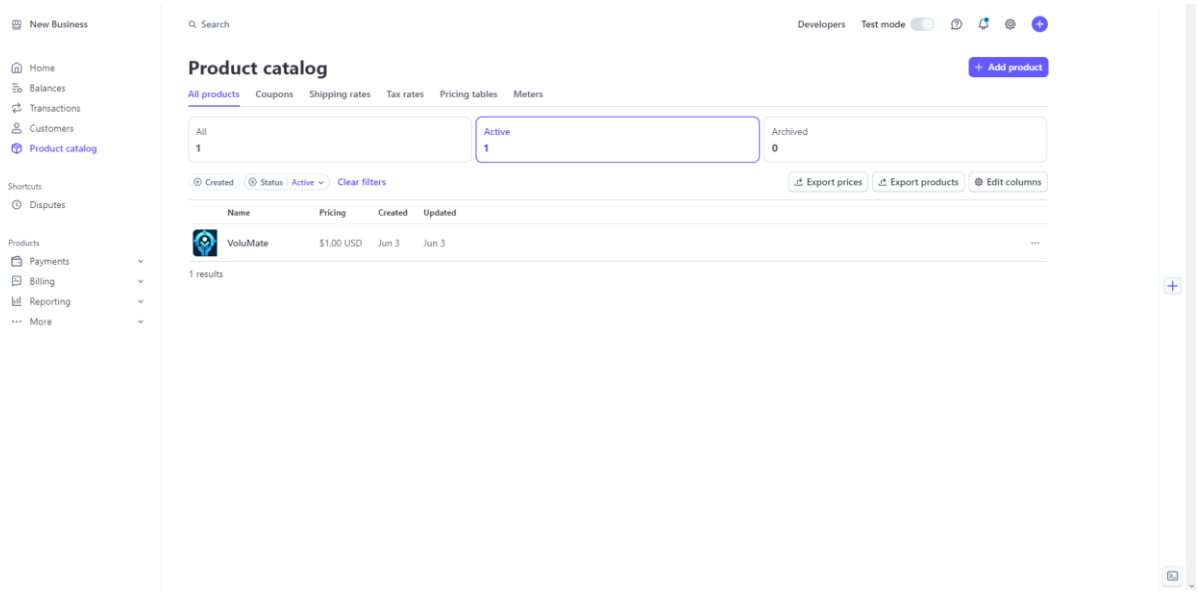

Рисунок 3.4 – Інтерфейс платіжної платформи з інтеграцією проєку

У React Native використання Stripe дозволяє розробникам інтегрувати функціональність платежів безпосередньо в мобільні додатки. Бібліотека дозволяє інтегрувати платіжні функції, такі як прийом кредитних карт або розрахунки, а також обробляти платежі з безпекою на бекенді Stripe.

Завдяки цьому, користувачі можуть здійснювати покупки чи оплачувати послуги прямо з мобільного додатка, забезпечуючи зручність і безпеку.

Отже, було створено модуль для перегляду інформації про волонтерські заходи та проекти та всі побічні його екрани та функції. Наявність додаткових функцій, таких як, зворотній зв’язок з ініціатором проекту та можливість

фінансової підтримки сприяє покращенню функціональності перед користувачем.

3.4 Розробка інтерфейсу

При розробці інтерфейсу програмного застосунку було приділено увагу його зрозумілості та зручності для користувача [18]. Особливо було виділено цілі розробити візуальтно привабливий та зручний інтерфейс, який відповідає цілям та цільовій аудиторії додатку.

Попередньо, інтерфейс застосунку було розроблено у Figma. Figma - це веб-інструмент для дизайну та прототипування інтерфейсів, який здобув широку популярність серед дизайнерів та розробників мобільних застосунків. Його основна перевага - доступність онлайн, що дозволяє працювати над проектом в будь-якому місці та на будь-якому пристрої без необхідності встановлення програмного забезпечення.

Інтерфейс інструменту Figma відображено на рисунку 3.5.

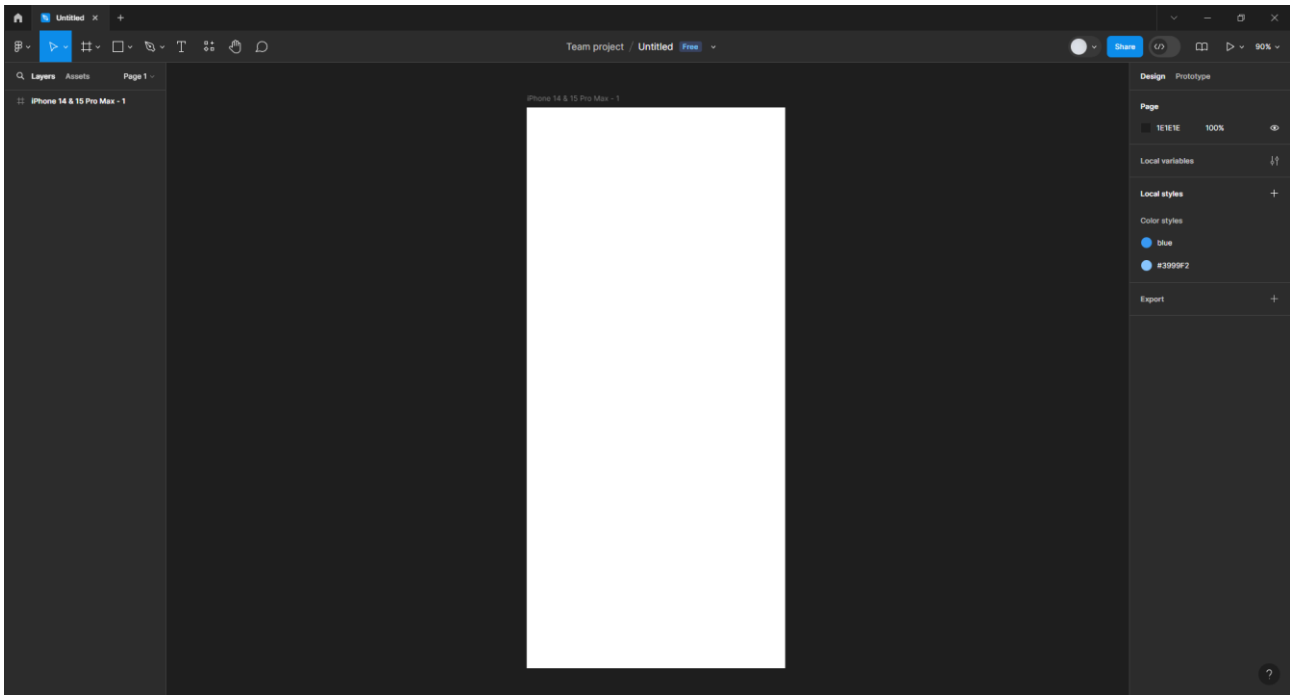

Рисунок 3.5 – Інтерфейс інструменту Figma

Figma відрізняється від інших інструментів своєю колаборативністю. Дизайнери, розробники та інші учасники проекту можуть спільно працювати над дизайном, бачити зміни в реальному часі та комунікувати між собою. Це значно спрощує процес спільної роботи та дозволяє швидше досягати результатів.

У Figma зручний інтерфейс, який дозволяє легко створювати макети, макетирувати елементи та створювати інтерактивні прототипи. Завдяки багатому набору інструментів і можливості імпортувати ресурси, такі як зображення та іконки, Figma надає безліч можливостей для творчості.

Ще одна важлива риса Figma - це можливість створення компонентів та стилів, які можна повторно використовувати. Це дозволяє створювати єдинообразний дизайн та ефективно керувати змінами в проекті.

Однією з ключових переваг Figma є його можливість працювати з векторною графікою, що робить інтерфейси, створені в Figma, масштабованими та гнучкими.

Це дозволяє створювати дизайни, які без втрат якості можуть адаптуватися до різних розмірів екранів мобільних пристроїв, що є критичним для розробки мобільних додатків.

Ще однією корисною функцією є можливість створювати інтерактивні прототипи, які дозволяють перевіряти функціональність та навігацію в додатку перед розробкою.

Інтерактивність дозволяє команді та клієнтам краще зрозуміти, як буде взаємодіяти користувач з додатком в реальному житті, що сприяє виправленню помилок та вдосконаленню дизайну ще на ранніх етапах розробки.

Figma використовується веб-дизайнерами, UI/UX-дизайнерами, графічними дизайнерами, продукт-менеджерами, розробниками та іншими фахівцями в галузі цифрового дизайну. Вона стала особливо популярною серед online команд, що працюють віддалено, оскільки Figma полегшує спільну роботу та обмін ідеями.

Крім того, Figma має вбудовані можливості спільної роботи з розробниками, такі як зручний експорт коду CSS для дизайну та можливість генерувати документацію для розробників мобільних додатків.

Це сприяє збереженню єдинообразного дизайну та полегшує процес інтеграції дизайну з розробкою.

Плагін для експорту коду CSS для дизайну застосунку відображено на рисунку 3.6.

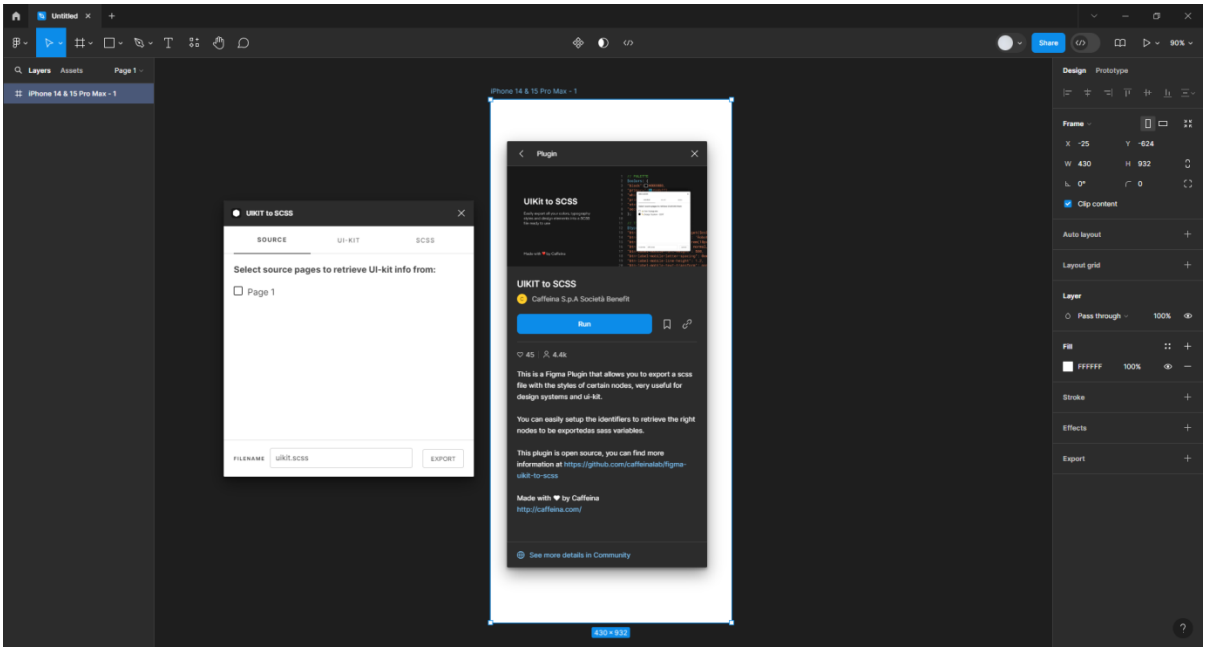

Рисунок 3.6 – Використання плагінів для еспротування стилів

Загальна доступність та потужні можливості Figma роблять його ідеальним інструментом для розробки інтерфейсів мобільних застосунків, дозволяючи командам швидко та ефективно працювати над проектами, спільно втілюючи ідеї та вдосконалюючи дизайн.

Основним кольором інтерфейсу обрано блакитний, оскільки даний колір є досить актуальним відносно теми та легким для сприйняття користувачами, другорядними є легкий відтінок блакитного та сірий. Також допоміжними кольорами обрано білий та чорний, оскільки вони контрастують один одному.

Отже, Figma має безліч переваг для роботи, включно з можливістю роботи в режимі реального часу, спільною роботою, доступністю та зручністю використання. Вона дає змогу створювати компоненти, спрощуючи процес дизайну та повторного використання елементів інтерфейсу. Крім того, Figma дає змогу створювати прототипи та робити презентації своїх проєктів. Figma не обмежується роботою тільки з векторною графікою, а й підтримує використання растрових зображень. Це може бути особливо корисним для дизайнерів, які хочуть створити інтерактивні прототипи з використанням реалістичних зображень.

При відкритті мобільного застосунку, користувача зустрічає його початковий екран застосунку відображено на рисунку 3.7.

Рисунок 3.7 – Початковий екран

При запуску застосунку користувача зустрічає початковий екран, де користувач може авторизуватись за допомогою облікового запису «Google».

Коли користувач натисне кнопку «Про проект» перед ним відобразиться екран інформації про застосунок.

Якщо користувач натисне кнопку «Авторизація», він буде перенаправлений на екран вибору облікового запису для входу.

Екран вибору облікового запису для входу відображена на рисунку 3.8.

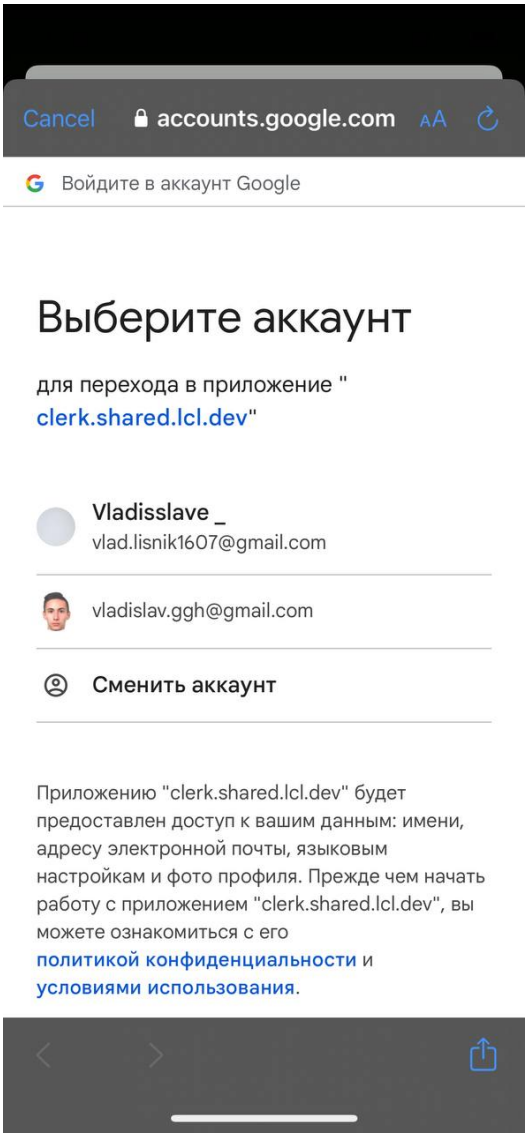

Рисунок 3.8 – Екран вибору облікового запису

Після того, як користувач авторизується у системі, його вітає головний екран застосунку.

Цей екран є ключовим елементом у формуванні ідентичності та визначенні враження від використання додатку.

Головний екран застосунку відображено на рисунку 3.9.

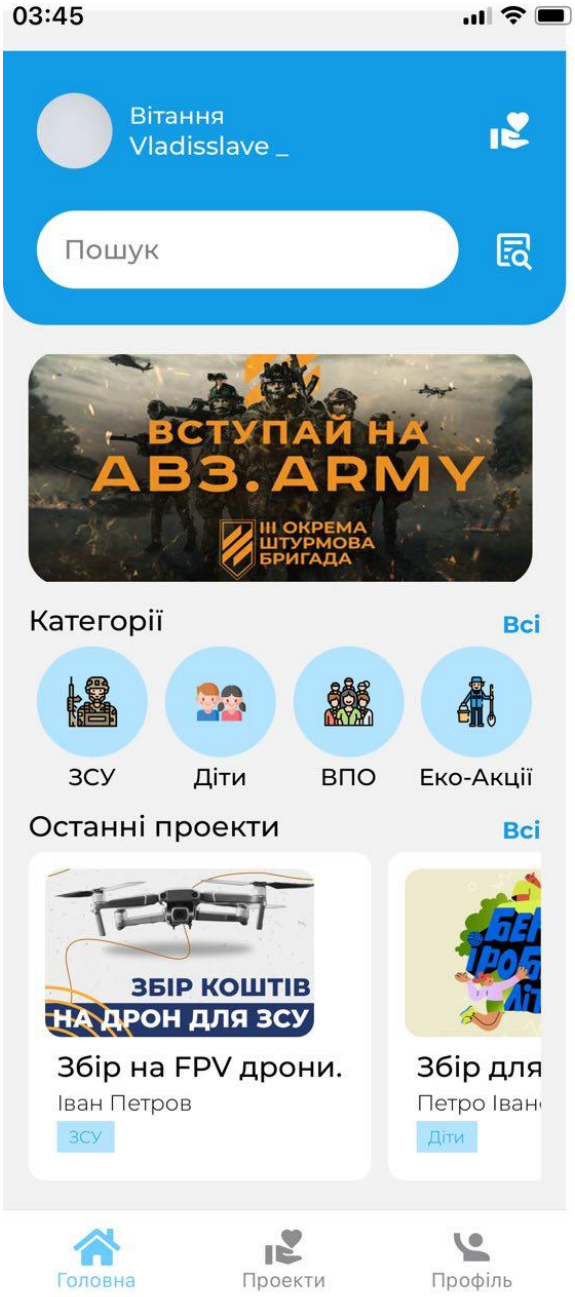

Рисунок 3.9 – Головний екран застосунку

Головний екран застосунку зустрічатиме користувача багатим інтерфейсом, що відкриє для користувача та інтуїтивно-зрозуміло відобразить основне наповнення застосунку, а також привітає його та зобразить фото та ім’я.

В загальному, застосунок складається з трьох основних екранів: «Головна», «Проекти» та «Профіль».

На екрані «Головна» - знаходиться найбільша частина наповнення усього застосунку, що складається з обумовленого пошуку за назвою проекту, слайдера, що відображає рекламні інтеграції, списком категорій та останніми волонтерськими проектами.

При натисканні на фото слайдера, користувач буде перенесений на сайт відносно рекламної інтеграції.

Після переходу за певною категорією перед користувачем буде відображено список волонтерських проектів відносно тієї категорії, що відображено на рисунку 3.10.

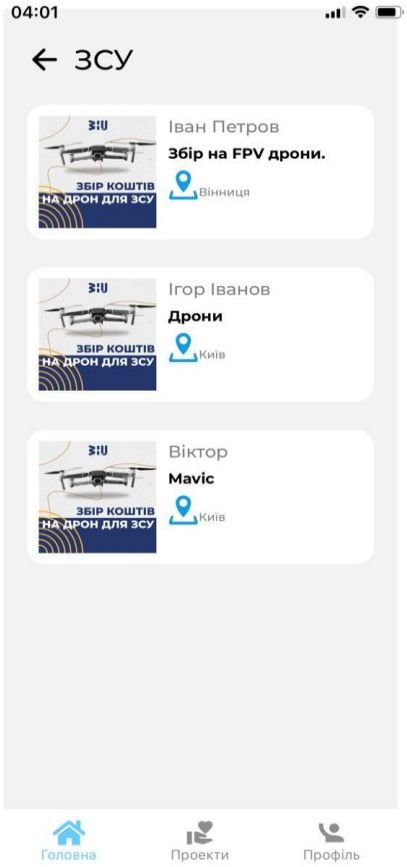

Рисунок 3.10 – Список проектів відносно категорії

У певній категорії відображається список проектів, їх фото, контактне лице та локація.

Після переходу на певний волонтерський проект перед користувачем відобразиться повна інформація про сам проект.

Екран інформації про проект відображено на рисунку 3.11.

Рисунок 3.11 – Екран інформації про проект

Після перегляду інформації щодо проекту користувач матиме змогу задати запитання.

Натиснувши кнопку «Повідомлення» для уточнення інформації або при виникненні запитань відкривається вікно електронної пошти, що відображено на рисунку 3.12.

Повідомлення буде надіслано на електронну скриньку контактної особи зазначеної у проекті.

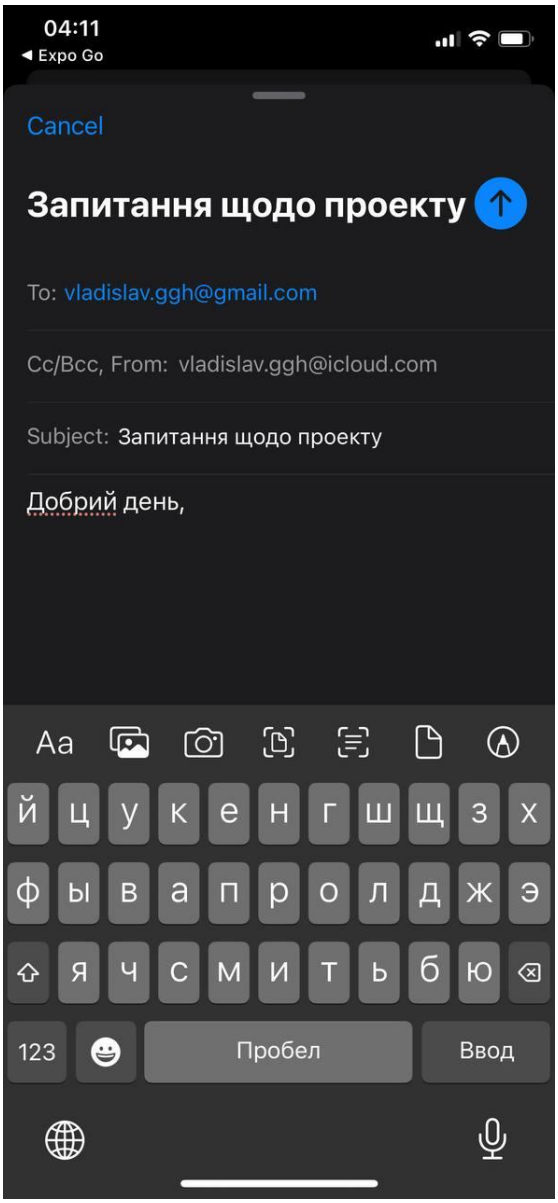

Рисунок 3.12 – Відправка користувачем повідомлення при виникненні запитань щодо проекту

Також користувач має змогу здійснити фінансову допомогу, натиснувши кнопку «Приєднатися», тим самим долучитись до певного збору.

Після натискання користувач буде перенесений на сторінку, де потрібно буде заповнити свою електронну адресу та поле платіжної інформації для здійснення фінансової допомоги та коментару до неї, що відображено на рисунку 3.13.

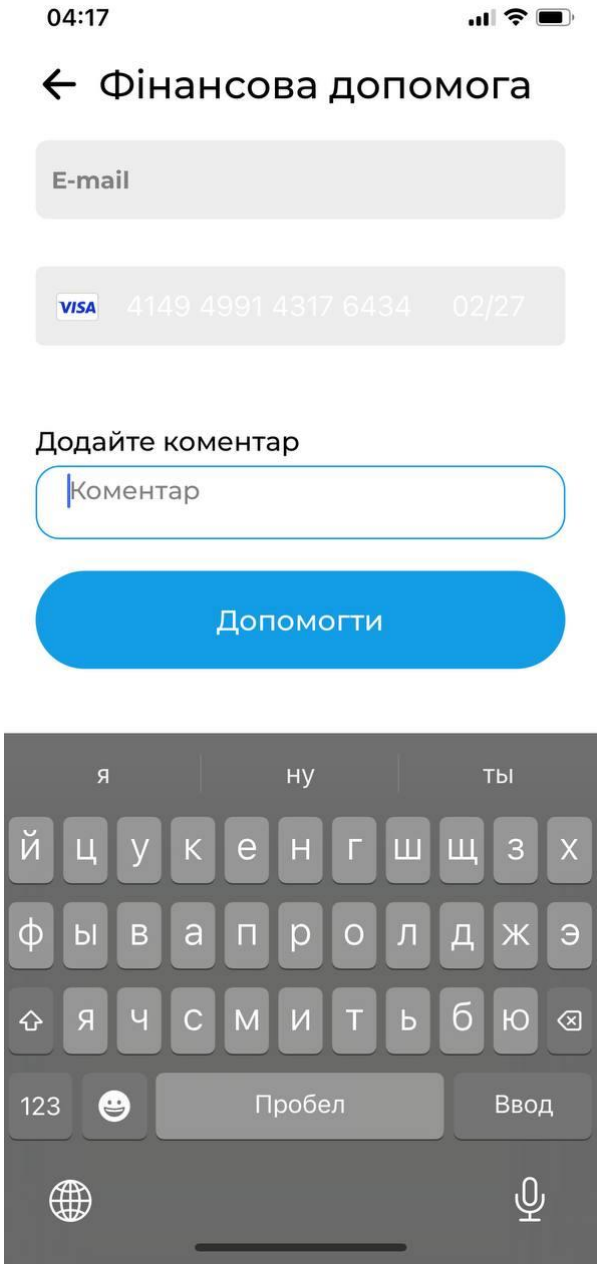

Рисунок 3.13 – Екран фінансової допомоги

Закінчується головний екран відображенням списку останніх волонтерських проектів.

Список останніх волонтерських проектів відображено на рисунку 3.14.

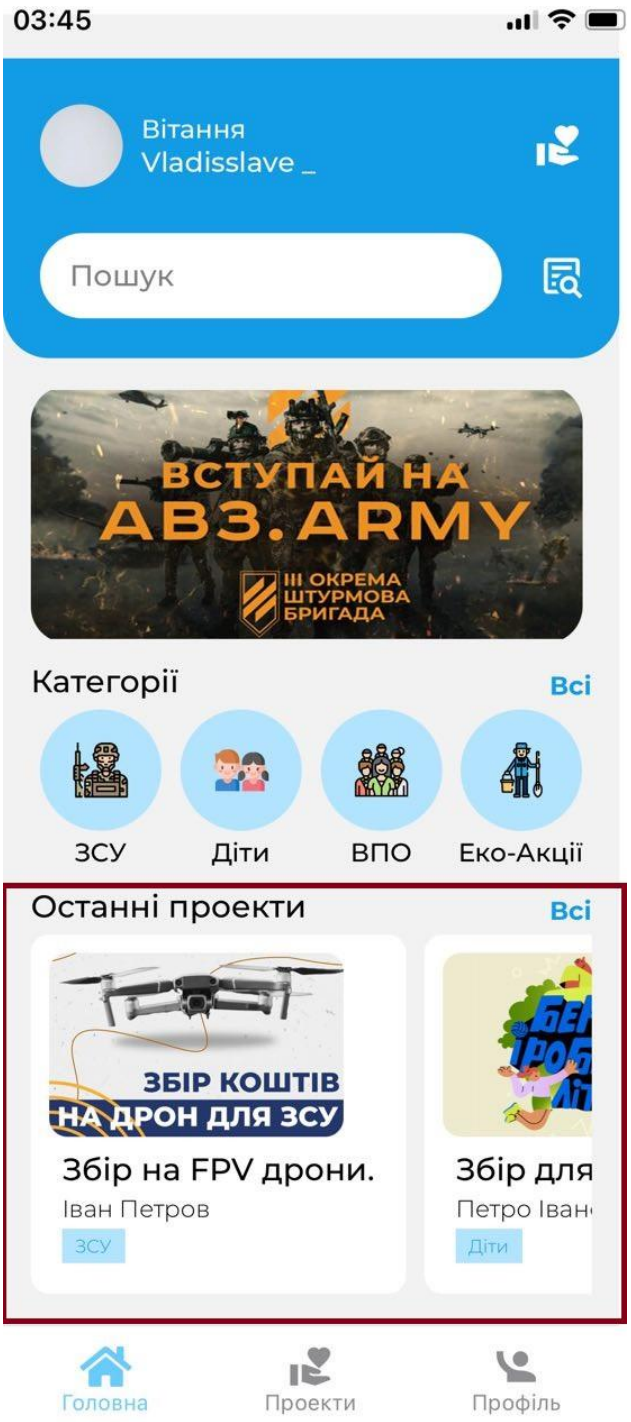

Рисунок 3.14 – Відображення списку останніх проектів

Наступним у навігації застосунку є екран «Проекти», на якому відображено весь список проектів з їх назвами, контактними особами, маркером категорії, та маркером їх прогресу.

Екран «Проекти» мобільного застосунку відображено на рисунку 3.15.

Рисунок 3.15 – Екран «Проекти»

Третім, також основним з головних екранів – є екран «Профіль», у якому відображено профіль користувача та відображене навігаційне меню застосунку.

Також, екран включає зворотній зв’язок із розробником та вихід із облікового запису.

Екран профілю користувача відображено на рисунку 3.16.

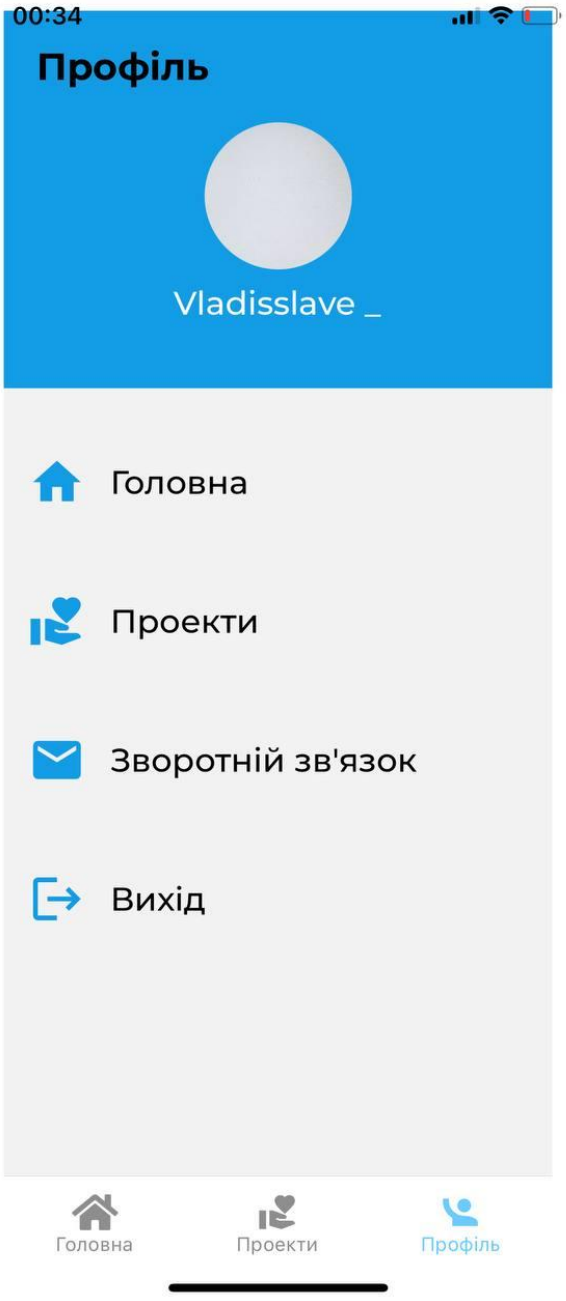

Рисунок 3.16 – Екран профілю користувача

На екрані профілю користувача знаходиться меню, у якому можна перейти на екран «Головна», «Проекти, а також написати розробнику листа, або деавторизуватись в застосунку.

Зворотній зв’язок із розробником відображено на рисунку 3.17.

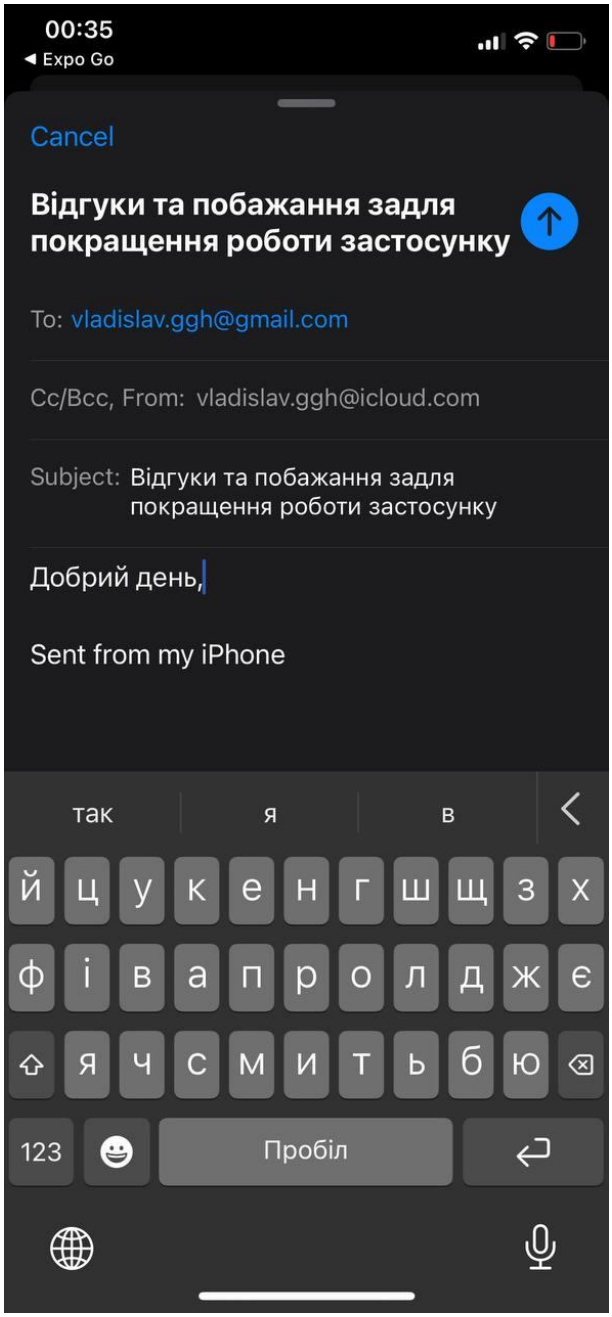

Рисунок 3.17 – Зв’язок із розробником

Попереднє визначення з інтерфейсом застосунку було проведено в дизайн-інструменті Figma, безпосередньо розробка графічного інтерфейсу програмного застосунку була проведена у середовищі розробки Visual Studio Code з використанням технології React Native та мови програмування JavaScript.

3.5 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки програмних модулів. Виконавши аналіз програмного застосунку та його задач було обрано комбінацію технологій, а саме React Native з фреймворком Expo та мова запитів GraphQL. У розділі також було описано розробку основних програмних модулів та додаткових функцій зворотного зв’язку із ініціаторами проекту та реалізацією можливості фінансової підтримки проєкту. Додатково, було описано розробку інтерфейсу застосунку.

4 ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1 Тестування системи

Тестування програмного забезпечення – це процес перевірки програмного продукту з метою виявлення помилок, дефектів або неполадок та перевірки відповідності вимогам замовника [19].

Основна мета тестування – забезпечити якість та надійність програмного продукту перед його випуском у виробництво або початком експлуатації.

Перед тестуванням програми на смартфоні з операційною системою Android або iOS необхідно встановити тестову версію додатка.

Також важливо підготувати документацію, включаючи вимоги та план тестування. Процес тестування складається з кількох етапів.

Перший – функціональне тестування, яке перевіряє правильність роботи програми, включаючи введення та виведення даних і відповідність бізнес-логіці [20].

Другий етап – тестування відмов, спрямоване на виявлення дефектів та здатність програми відновлювати роботу після виникнення помилок [21]. Після цього проводиться тестування продуктивності, що включає вимірювання швидкості виконання операцій.

Також важливим етапом є тестування сумісності, що перевіряє взаємодію програми з різними операційними системами та апаратними платформами [22].

Під час першого етапу тестування програмного забезпечення перевіряється швидкість відповіді мобільного застосунку на різні запити.

Тестування швидкості виконання запитів мобільного застосунку дозволяє виявити та виправити можливі проблеми з продуктивністю, які можуть вплинути на користувацький досвід.

Це дозволяє забезпечити швидку реакцію додатка на вхідні дані та оптимізувати його роботу для забезпечення оптимальної продуктивності.

Тестування швидкості завантаження компонентів та обробки запитів відображено на рисунку 4.1.

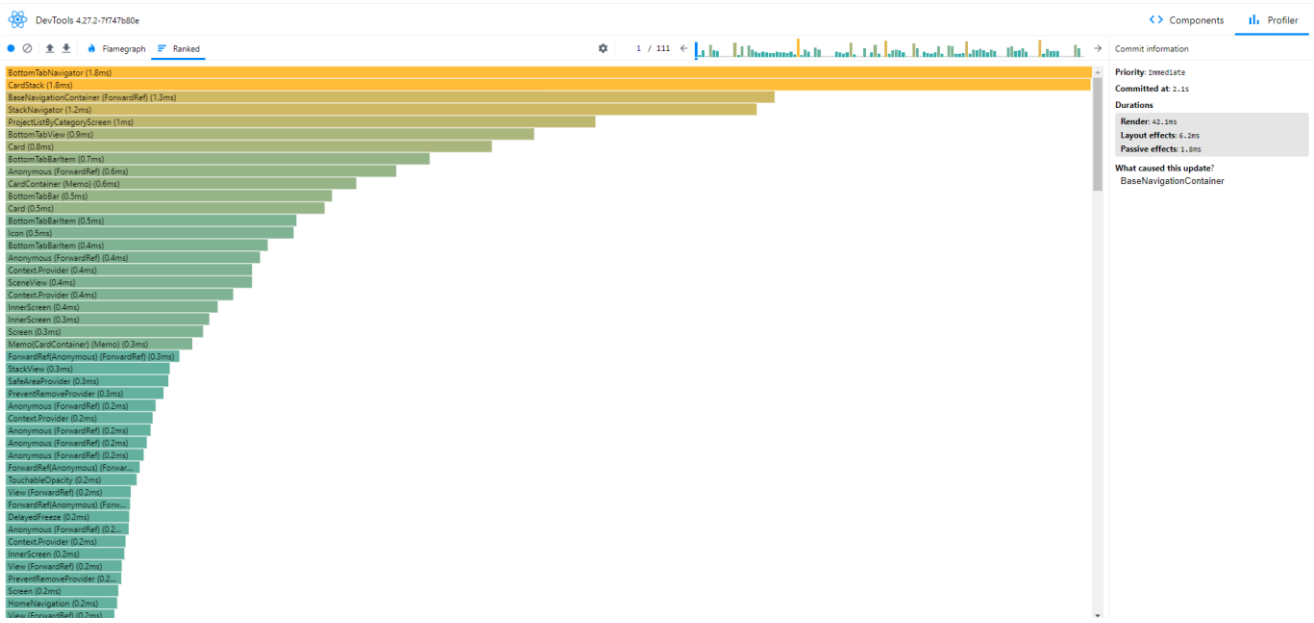

Рисунок 4.1 – Тестування швидкості завантаження компонентів та обробки запитів

Після проведення тестування швидкості завантаження компонентів та обробки запитів було прийнято рішення провести додаткове тестування, спрямоване на перевірку реакції програмного забезпечення на відсутність вхідних даних.

Це було зроблено з метою визначення, як програма реагує на таку ситуацію та чи забезпечує вона користувача відповідними повідомленнями або інструкціями. У результаті цього тестування було виявлено, що програмне забезпечення обробляє такий сценарій використання користувачем та надає інформацію про можливий шлях рішення.

Важливим аспектом проведеного додаткового тестування було визначення того, наскільки ефективно програма впоралася з ситуацією відсутності вхідних даних та чи були надані вичерпні інструкції користувачу. Це допомагає забезпечити, що навіть у складних або непередбачуваних сценаріях користувачі будуть знати, як діяти.

Тестування застосунку на реакцію на відсутність вхідних даних відображено на рисунку 4.2.

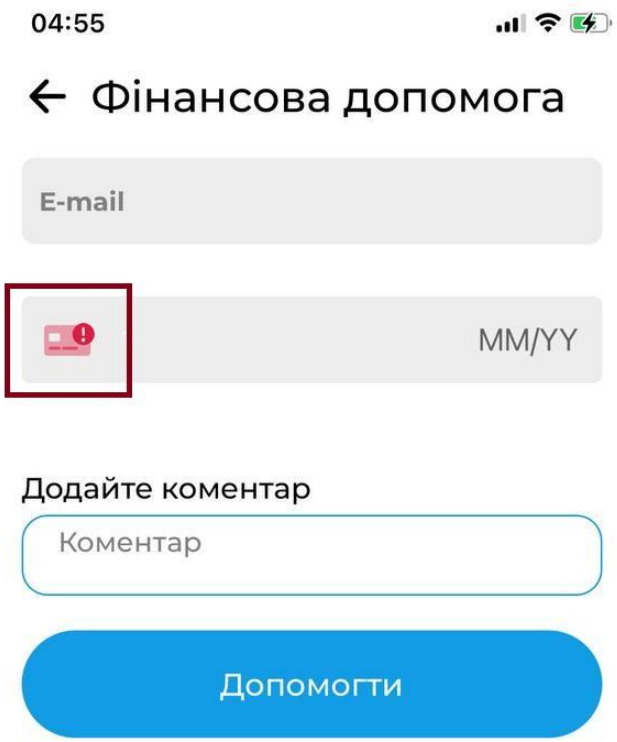

Рисунок 4.2 – Реакція застосунку на відсутність вхідних даних

У цьому випадку застосунок змінив вигляд іконки картки на червоний із знаком оклику, що пояснює неправильне введення даних.

Якщо користувач введе неправильні дані в графу, застосунок повідомить про це повідомленням, що відображено на рисунку 4.3.

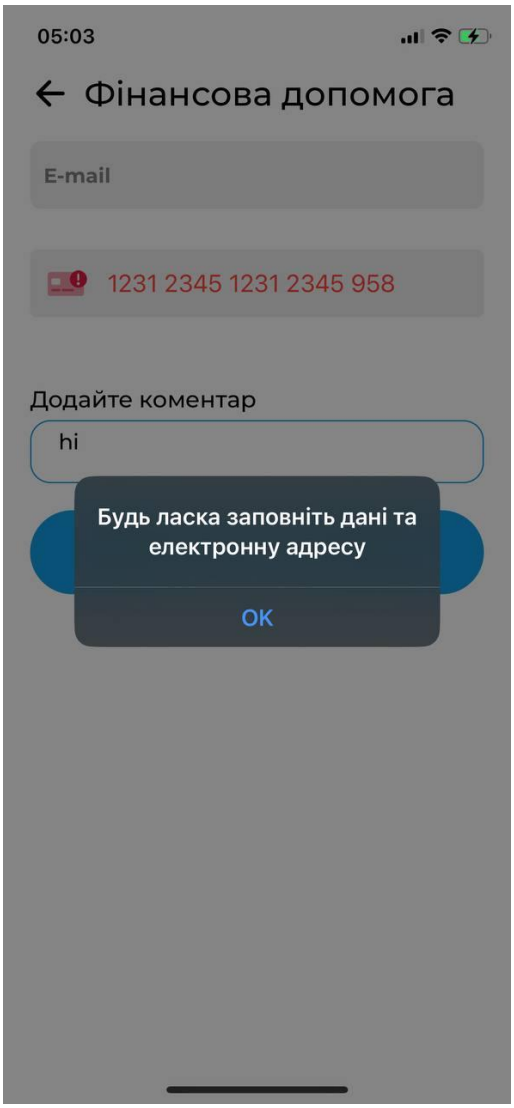

Рисунок 4.3 – Реакція застосунку на відсутність вхідних даних

Результати тестування підтвердили, що програмне забезпечення виявляє відсутність вхідних даних та надає відповідні повідомлення або інструкції.

Це дозволяє користувачам ефективно керувати ситуацією. Також, це допомагає забезпечити позитивний досвід користувача та зменшує ризик виникнення помилок або неприємних ситуацій при взаємодії з програмою.

У рамках тестування програмного забезпечення також було проведено аналіз його сумісності з різними версіями операційних систем та мінімальною конфігурацією апаратного забезпечення.

Зокрема, програмний продукт був перевірений на сумісність з наступними операційними системами: Android 10 та iOS 17.4.1. Під час тестування використовувалися як високопродуктивні iOS пристрої, так і застарілі Android пристрої.

Результати тестування показали успішну роботу програмного забезпечення як на потужних, так і на менш потужних пристроях та різних операційних платформах, що вказує на його високу оптимізацію та ефективність роботи.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації розміщено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Мінімальні	Рекомендовані
Процесор	З тактовою частотою 1,5-2ГГц	З тактовою частотою 2-2,5 ГГц
Оперативна пам'ять	2 ГБ RAM	4 ГБ RAM
Вільний простір у сховищі	1 ГБ	2 ГБ
Батарея	Не менше 1500 мАг	Більше 2000 мАг
Операційна система	Android 8.0, iOS 12	Android 10, iOS 14
Дисплей	Роздільна здатність 720p, діагональ 5-6 дюймів	Роздільна здатність 1080p, діагональ 5,5-6,5 дюймів

Після запуску та авторизації у мобільному застосунку, користувача вітає головний екран застосунку. Цей екран є ключовим елементом у формуванні ідентичності та визначенні враження від використання додатку.

Головний екран застосунку зустрічатиме користувача багатим інтерфейсом, що відкриє для користувача та інтуїтивно-зрозуміло відобразить основне наповнення застосунку, а також привітає його та зобразить фото та ім’я.

В загальному, застосунок складається з трьох основних екранів: «Головна», «Проекти» та «Профіль».

Основним з головних екранів – є екран «Профіль», у якому відображено профіль користувача та відображене навігаційне меню застосунку.

Також, екран включає зворотній зв’язок із розробником та вихід із облікового запису.

На екрані профілю користувача знаходиться навігаційне меню, у якому можна перейти на екран «Головна», «Проекти, а також написати розробнику листа, або деавторизуватись в застосунку.

На екрані «Проекти» перед користувачем буде відображено усі не завершені проєкти не залежно від категрій, із виділеним індикатором завершеності.

При натисканні на проект користувач буде перенесений на інформаційну сторінку проєкту, де він зможе дізнатися більше про проєкт, задати запитання шляхом надсилання електронноло листа ініціатору збору, або фінансово підтримати проєкт.

На екрані «Головна» - знаходиться найбільша частина наповнення усього застосунку, що складається з обумовленого пошуку за назвою проекту, слайдера, що відображає рекламні інтеграції, списком категорій та останніми волонтерськими проектами.

При натисканні на фото слайдера, користувач буде перенесений на сайт відносно рекламної інтеграції.

Після переходу за певною категорією перед користувачем буде відображено список волонтерських проектів відносно тієї категорії.

Головний екран застосунку відображено на рисунку 4.4.

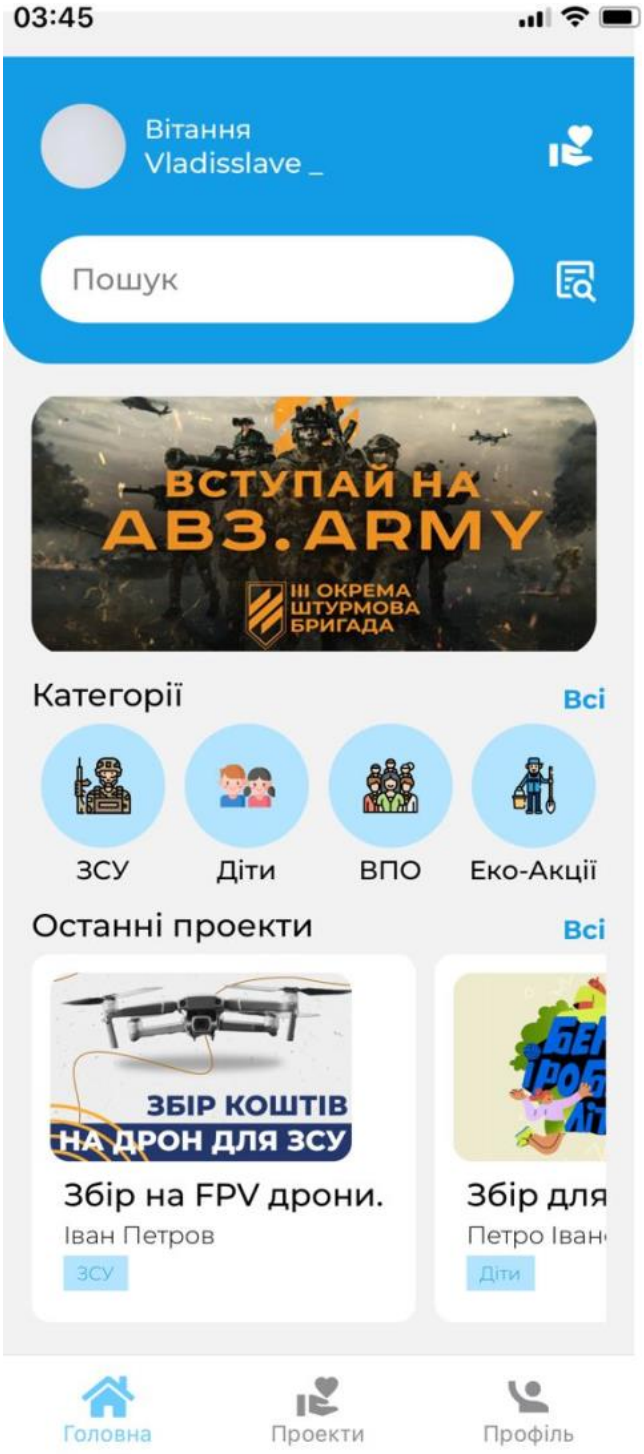

Рисунок 4.4 – Головний екран мобільного застосунку

4.3 Висновки

У четвертому розділі проведено тестування програмних модулів мобільного застосунку для підтримки діяльності благодійних організацій. Перевірено реакцію програмних засобів на помилкові ситуації, вхідні дані різних форматів. Також, розроблено інструкцію користувача по початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській дипломній роботі розроблено модулі мобільного застосунку для підтримки діяльності благодійних організацій організацій. Для розробки використовувалося середовище програмування Visual Studio Code.

Під час виконання роботи проведено аналіз сучасного стану проблеми. Розглянуто основні аналоги мобільної системи, виявлені їхні недоліки та порівняно з власним розробленим продуктом. На основі цього порівняння сформульовані основні завдання бакалаврської дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мов програмування JavaScript, її технології React Native та фреймворку Expo, а також мови запитів GraphQL.

У бакалаврській дипломній роботі було зроблено наступне:

- розроблено модуль авторизації користувачів для забезпечення безпеки та додаткових функцій;
- розроблено модуль для перегляду інформації про волонтерські заходи та проекти;
- розробка модуль зворотного зв’язку та фінансової допомоги.
- розроблено зручний та адаптивний графічний інтерфейс програми;
- проведено тестування програмного забезпечення згідно поставлених задач.

Розроблено схеми навігації програмного застосунку. Також, розроблено модель системи з використанням UML діаграм.

Тестування застосунку довело повну працездатність мобільної системи та відповідність поставленому технічному завданню. Також, розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Актуальність розробки застосунків для волонтерства [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegwh.

2. Лісник В.І. Мобільний застосунок для автоматизації процесів діяльності волонтерських організацій/ В.П. Майданюк, В.І. Лісник // Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegwr.

3. Лісник В. І. Актуальність розробки серверної частини мобільних застосунків за допомогою cms-орієнтованих сервісів/ В.П. Майданюк, В.І. Лісник // Матеріали міжнародної науково-практичної інтернет конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegwu.

4. VolunteerUP [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegxh.

5. СпівДія [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegxq.

6. ВолонтерОрг [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegxt.

7. UML diagram [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegxx.

8. UML activity diagram [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegyf.

9. Visio [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegyl.

10. React Native [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegyv.

11. Benefits React Native [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegze.

12. Expo framework [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzi.

13. Advantages of the Expo [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzl.

14. GraphQL [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzp.

15. Benefits of GraphQL [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzs.

16. Clerk [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzv.

17. Expo Secure Store [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzy.

18. Robert C. Martin Series. Functional Design: Principles, Patterns, and Practices. 1st Edition, 2023. 384 p.

19. Software testing [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uehab.

20. Functional testing [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uehag.

21. Failure testing [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tkely.

22. Compatibility testing [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/ueham.

58

ДОДАТОК А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

"12" березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка мобільного застосунку для підтримки діяльності благодійних організацій» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доцент В.П. Майданюк

"_12_" _____березня_____ 2024 р.

Виконав:

_____ студент гр. 2ПІ-20б В.І. Лісник

"_12_" _____березня_____ 2024 р.

Вінниця ВНТУ - 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільного застосунку для підтримки діяльності благодійних організацій».

Галузь застосування – мобільні технології.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є підвищення комунікативних можливостей користувачів та організацій у сфері благодійності за рахунок створення мобільного застосунку для підтримки діяльності благодійних організацій. Застосунок сприятиме ефективній координації заходів, забезпечуючи швидкий доступ до ресурсів та інформації.

Основними задачами дослідження є:

- розробка бази даних для зберігання користувацьких даних та інформації про проекти благодійних організацій;
- розробка інтерфейсу користувача, що забезпечить зручну взаємодію з застосунком та легкий доступ до необхідної інформації;
- розробка модуля авторизації користувачів для забезпечення безпеки та додаткових функцій;
- розробка модуля для перегляду інформації про волонтерські заходи та проекти;
- розробка модуля фінансової допомоги.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

1. React Native [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegyv.
2. Benefits React Native [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegze.
3. Expo framework [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzi.
4. Advantages of the Expo [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzl.
5. GraphQL [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/uegzp.
6. Robert C. Martin Series. Functional Design: Principles, Patterns, and Practices. 1st Edition, 2023. 384 p.

5. Технічні вимоги

- середовище розробки – Visual Studio Code;
- мова програмування – JavaScript;
- вхідні дані: інформація про благодійні організації, контент для створення проектів, обліковий запис користувача;
- вихідні дані: застосунок із графічним користувацьким інтерфейсом для перегляду проектів.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задачі	14.03.24 – 25.03.24
2	Розробка моделей, структури та алгоритмів	26.03.24 – 10.04.24
3	Розробка програмного забезпечення	11.04.24 – 28.04.24
4	Тестування застосунку	29.04.24 – 15.05.24
5	Оформлення матеріалів до захисту БДР	16.05 – 30.05.24

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

ДОДАТОК Б

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка мобільного застосунку для підтримки діяльності благодійних організацій

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент кафедри ПЗ, Майданюк Володимир Павлович.

Оригінальність	90,6
Схожість	9,4

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Лісник В.І.

Керівник роботи _____ Майданюк В.П.

ДОДАТОК В

(довідниковий)

Лістинг програмного коду мобільного застосунку

Модуль Login.jsx

```
import {
  View,
  Text,
  Image,
  StyleSheet,
  TouchableOpacity,
  Button,
} from "react-native";
import React from "react";
import Colors from "../../Utils/Colors";
import * as WebBrowser from "expo-web-browser";
import { useWarmUpBrowser } from "../../hooks/warmUpBrowser";
import { useOAuth } from "@clerk/clerk-expo";
import { Ionicons, Entypo, AntDesign } from "@expo/vector-icons";

WebBrowser.maybeCompleteAuthSession();

export default function Login() {
  useWarmUpBrowser();

  const { startOAuthFlow } = useOAuth({ strategy: "oauth_google" });
```

```
const onPress = React.useCallback(async () => {
  try {
    const { createdSessionId, signIn, signUp, setActive } =
      await startOAuthFlow();

    if (createdSessionId) {
      setActive({ session: createdSessionId });
    } else {
      // Use signIn or signUp for next steps such as MFA
    }
  } catch (err) {
    console.error("OAuth error", err);
  }
}, []);

return (
  <View style={{ alignItems: "center" }}>
    <Image
      source={require("./../../../assets/images/logo.png")}
      style={styles.logoImage}
    />
    <View style={styles.subContainer}>
      <Text
        style={{ fontSize: 27, color: Colors.WHITE, textAlign: "center" }}
      >
        <Text style={{ fontFamily: "Montserrat-Bold" }}>
          {" "}
          Моніторинг проектів, зв'язок із організаціями
```

```
          </Text>
          {""}
        </Text>
        <Text
          style={{
            fontSize: 14,
            color: Colors.WHITE,
            textAlign: "center",
            marginTop: 20,
            fontFamily: "Montserrat-Medium",
          }}
        >
          Єдина платформа для організації благодійності VoluMate - все в одному
          місці!
        </Text>

        <TouchableOpacity style={styles.button} onPress={onPress}>
          <Text
            style={{
              textAlign: "center",
              fontSize: 17,
              color: Colors.PRIMARY,
              fontFamily: "Montserrat-Medium",
            }}
          >
            Авторизація
          </Text>
```

```
            </TouchableOpacity>

            <TouchableOpacity style={styles.button1}>
              <Text
                style={{
                  textAlign: "center",
                  fontSize: 17,
                  color: Colors.PRIMARY,
                  fontFamily: "Montserrat-Medium",
                  alignItems: "center",
                }}
              >
                <AntDesign name="infocirlceo" size={12} color="black" /> Про проект
              </Text>
            </TouchableOpacity>
          </View>
        </View>
      );
    }

    const styles = StyleSheet.create({
      logoImage: {
        width: 220,
        height: 250,
        marginTop: 20,
        borderWidth: 2,
        borderColor: Colors.BLACK,
```

```
    borderRadius: 20,
  },
  subContainer: {
    width: "112%",
    backgroundColor: Colors.PRIMARY,
    height: "70%",
    marginTop: 10,
    borderTopLeftRadius: 30,
    borderTopRightRadius: 30,
    padding: 30,
  },
  button: {
    padding: 18,
    backgroundColor: Colors.WHITE,
    borderRadius: 99,
    marginTop: 85,
  },
  button1: {
    padding: 18,
    backgroundColor: Colors.WHITE,
    borderRadius: 99,
    marginTop: 50,
  },
});
```

Модуль TabNavigation.jsx

```
import { View, Text } from "react-native";
import React from "react";
```

```
import Colors from "../Utils/Colors";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import ProfileScreen from "../Screens/ProfileScreen/ProfileScreen";
import { FontAwesome5 } from "@expo/vector-icons";
import { MaterialIcons } from "@expo/vector-icons";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import HomeNavigation from "./HomeNavigation";
import ProjectNavigation from "./ProjectNavigation";

const Tab = createBottomTabNavigator();

export default function TabNavigation() {
  return (
    <Tab.Navigator
      screenOptions={{
        headerShown: false,
        tabBarActiveTintColor: Colors.TABBAR,
      }}
    >
      <Tab.Screen
        name="Home"
        component={HomeNavigation}
        options={{
          tabBarLabel: ({ color }) => (
            <Text style={{ color: color, fontSize: 12, marginTop: -10 }}>
              Головна
            </Text>
          ),
```

```
    tabBarIcon: ({ color, size }) => (
      <FontAwesome5 name="home" size={size} color={color} />
    ),
  }}
/>
<Tab.Screen
  name="Projects"
  component={ProjectNavigation}
  options={{
    tabBarLabel: ({ color }) => (
      <Text style={{ color: color, fontSize: 12, marginTop: -10 }}>
        Проекти
      </Text>
    ),
    tabBarIcon: ({ color, size }) => (
      <MaterialIcons
        name="volunteer-activism"
        size={size}
        color={color}
      />
    ),
  }}
/>
<Tab.Screen
  name="Profile"
  component={ProfileScreen}
  options={{
    tabBarLabel: ({ color }) => (
```

```
          <Text style={{ color: color, fontSize: 12, marginTop: -10 }}>
            Профіль
          </Text>
        ),
        tabBarIcon: ({ color, size }) => (
          <MaterialCommunityIcons
            name="human-greeting-variant"
            size={size}
            color={color}
          />
        ),
      }}
    />
  </Tab.Navigator>
 );
}
```

Модуль HomeScreen.jsx

```
import { View, Text } from "react-native";
import React from "react";
import Header from "./Header";
import Slider from "./Slider";
import Categories from "./Categories";
import ProjectList from "./ProjectList";
import { SafeAreaView } from "react-native-safe-area-context";

export default function HomeScreen() {
 return (
```

```
    <SafeAreaView style={{ flex: 1 }}>
      <Header />
      <View style={{ padding: 15 }}>
        <Slider />
        <Categories />
        <ProjectList />
      </View>
    </SafeAreaView>
  );
}
```

Модуль Header.jsx

```
import {
  View,
  Text,
  Image,
  StyleSheet,
  TextInput,
  TouchableOpacity,
} from "react-native";
import React from "react";
import { useUser } from "@clerk/clerk-expo";
import Colors from "../../Utils/Colors";
import { MaterialIcons } from "@expo/vector-icons";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import { useNavigation } from "@react-navigation/native";

export default function Header() {
```

```
const { user, isLoading } = useUser();
const navigation = useNavigation();
return (
  user && (
    <View style={styles.container}>
      <View style={styles.profileMainContainer}>
        <TouchableOpacity
          style={styles.profileContainer}
          onPress={() => navigation.navigate("Profile")}
        >
          <Image source={{ uri: user?.imageUrl }} style={styles.userImage} />
          <View>
            <Text
              style={{ color: Colors.WHITE, fontFamily: "Montserrat-Medium" }}
            >
              Вітання
            </Text>
            <Text
              style={{
                color: Colors.WHITE,
                fontSize: 15,
                fontFamily: "Montserrat-Medium",
              }}
            >
              {user?.fullName}
            </Text>
          </View>
        </TouchableOpacity>
```

```
          <TouchableOpacity onPress={() => navigation.navigate("Projects")}>
            <MaterialIcons name="volunteer-activism" size={26} color="white" />
          </TouchableOpacity>
        </View>
        <View style={styles.searchBarContainer}>
          <TextInput
            placeholder="Пошук"
            placeholderTextColor={Colors.GRAY}
            style={styles.texInput}
          />
          <TouchableOpacity>
            <MaterialCommunityIcons
              name="text-box-search-outline"
              style={styles.searchButton}
              size={26}
              color={Colors.WHITE}
            />
          </TouchableOpacity>
        </View>
      </View>
    )
  );
}

const styles = StyleSheet.create({
  container: {
    width: "100%",
    padding: 20,
```

```
  paddingTop: 25,
  backgroundColor: Colors.PRIMARY,
  borderBottomLeftRadius: 25,
  borderBottomRightRadius: 25,
 },
 profileMainContainer: {
  display: "flex",
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-between",
 },
 profileContainer: {
  display: "flex",
  flexDirection: "row",
  alignItems: "center",
  gap: 10,
 },
 texInput: {
  padding: 7,
  paddingHorizontal: 16,
  backgroundColor: Colors.WHITE,
  borderRadius: 99,
  width: "85%",
  fontSize: 16,
  fontFamily: "Montserrat-Medium",
 },
 searchBarContainer: {
  marginTop: 25,
```

```
    display: "flex",
    flexDirection: "row",
    gap: 10,
    marginBottom: 2,
  },
  searchButton: {
    backgroundColor: Colors.PRIMARY,
    padding: 10,
    borderRadius: 8,
  },
  userImage: {
    width: 45,
    height: 45,
    borderRadius: 99,
  },
});
```

Модуль Slider.jsx

```
import {
  View,
  Text,
  StyleSheet,
  FlatList,
  Image,
  TouchableOpacity,
} from "react-native";
import React, { useEffect, useState } from "react";
import GlobalApi from "../../Utils/GlobalApi";
```

```
import Heading from "../../Components/Heading";
import { Linking } from "react-native";

export default function Slider() {
  const [slider, setSlider] = useState([]);
  useEffect(() => {
    getSlider();
  }, []);
  const getSlider = () => {
    GlobalApi.getSlider().then((resp) => {
      setSlider(resp?.sliders);
    });
  };

  return (
    <View>
      <FlatList
        data={slider}
        horizontal={true}
        showsHorizontalScrollIndicator={false}
        renderItem={({ item, index }) => (
          <TouchableOpacity
            style={{ marginRight: 20 }}
            onPress={() => {
              if (index === 0) {
                Linking.openURL("https://ab3.army/");
              } else if (index === 1) {
                Linking.openURL("http://surl.li/tjgop");
```

```
            }
           }}
          >
           <Image
            source={{ uri: item?.image?.url }}
            style={styles.sliderImage}
           />
          </TouchableOpacity>
        )}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  heading: {
    fontSize: 20,
    fontFamily: "Montserrat-Medium",
    marginBottom: 5,
  },
  sliderImage: {
    width: 300,
    height: 140,
    borderRadius: 20,
    objectFit: "contain",
    marginBottom: 10,
  },
});
```

Модуль Categories.jsx

```
import {
  View,
  Text,
  FlatList,
  Image,
  StyleSheet,
  TouchableOpacity,
} from "react-native";
import React, { useEffect, useState } from "react";
import GlobalApi from "../../Utils/GlobalApi";
import Heading from "../../Components/Heading";
import Colors from "../../Utils/Colors";
import { useNavigation } from "@react-navigation/native";

export default function Categories() {
  const [categories, setCategories] = useState([]);

  const navigation = useNavigation();

  useEffect(() => {
    getCategories();
  }, []);

  const getCategories = () => {
    GlobalApi.getCategories().then((resp) => {
```

```
    setCategories(resp?.categories);
  });
};

return (
  <View>
    <Heading text={"Категорії"} isViewAll={true} />
    <FlatList
      data={categories}
      numColumns={4}
      renderItem={({ item, index }) =>
        index <= 3 && (
          <TouchableOpacity
            style={styles.container}
            onPress={() =>
              navigation.push("project-list", {
                category: item.name,
              })
            }
          >
            <View style={styles.iconContainer}>
              <Image
                source={{ uri: item.image?.url }}
                style={{ width: 30, height: 30 }}
              />
            </View>
            <Text
              style={{
```

```
                fontSize: 14,
                fontFamily: "Montserrat-Medium",
                marginTop: 3,
              }}
            >
              {item?.name}
            </Text>
          </TouchableOpacity>
        )
      }
    />
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  alignItems: "center",
 },
 iconContainer: {
  backgroundColor: Colors.UNDERPRIMARY,
  padding: 18,
  borderRadius: 99,
 },
});
```

Модуль ProjectList.jsx

```
import { View, Text, FlatList, Image } from "react-native";
import React, { useEffect, useState } from "react";
import Heading from "../../Components/Heading";
import GlobalApi from "../../Utils/GlobalApi";
import ProjectListItemSmall from "./ProjectListItemSmall";

export default function ProjectList() {
  const [projectList, setProjectList] = useState([]);

  useEffect(() => {
    getProjectList();
  }, []);

  const getProjectList = () => {
    GlobalApi.getProjectList().then((resp) => {
      setProjectList(resp.projectsLists);
    });
  };

  return (
    <View style={{ marginTop: 10 }}>
      <Heading text={"Останні проекти"} isViewAll={true} />
      <FlatList
        data={projectList}
        horizontal={true}
        showsHorizontalScrollIndicator={false}
        renderItem={({ item, index }) => (
          <View style={{ marginRight: 10 }}>
```

```
            <ProjectListItemSmall project={item} />
          </View>
        )}
      />
    </View>
  );
}
```

Модуль ProjectListItemSmall.jsx

```
import { View, Text, Image, StyleSheet, TouchableOpacity } from "react-native";
import React from "react";
import Colors from "../../Utils/Colors";
import { useNavigation } from "@react-navigation/native";

export default function ProjectListItemSmall({ project }) {
  const navigation = useNavigation();
  return (
    <TouchableOpacity
      style={styles.container}
      onPress={() => {
        navigation.push("project-detail", {
          project: project,
        });
      }}
    >
      <Image source={{ uri: project?.image[0].url }} style={styles.image} />
      <View style={styles.infoContainer}>
```

```
        <Text style={{ fontSize: 17, fontFamily: "Montserrat-Medium" }}>
          {project?.name}
        </Text>
        <Text style={{ fontSize: 13, fontFamily: "Montserrat-Light" }}>
          {project?.contactPerson}
        </Text>
        <Text
          style={{
            fontSize: 10,
            fontFamily: "Montserrat-Light",
            padding: 3,
            color: Colors.PRIMARY,
            backgroundColor: Colors.UNDERPRIMARY,
            borderRadius: 3,
            alignSelf: "flex-start",
            paddingHorizontal: 7,
          }}
        >
          {project?.category.name}
        </Text>
      </View>
    </TouchableOpacity>
  );
}

const styles = StyleSheet.create({
  container: {
    padding: 10,
```

```
    backgroundColor: Colors.WHITE,
    borderRadius: 10,
  },
  infoContainer: {
    padding: 7,
    display: "flex",
    gap: 3,
  },
  image: {
    width: 160,
    height: 100,
    borderRadius: 10,
  },
});
```

Модуль ProjectDetailsScreen.jsx

```
import {
  View,
  Text,
  Image,
  TouchableOpacity,
  StyleSheet,
  ScrollView,
  Modal,
  Linking,
} from "react-native";
import React, { useEffect, useState } from "react";
import { useNavigation, useRoute } from "@react-navigation/native";
```

```
import { Ionicons, Entypo, AntDesign } from "@expo/vector-icons";
import Colors from "../../Utils/Colors";
import ProjectPhotos from "./ProjectPhotos";
import ProjectAbout from "./ProjectAbout";
import HelpModal from "./HelpModal";
import { StripeProvider } from "@stripe/stripe-react-native";

export default function ProjectDetailsScreen() {
  const param = useRoute().params;
  const [project, setProject] = useState(param.project);
  const navigation = useNavigation();
  const [showModal, setShowModal] = useState(false);

  useEffect(() => {}, []);
  const onMessageBtnClick = () => {
    Linking.openURL(
      "mailto:" +
        "vladislav.ggh@gmail.com" +
        "?subject=Запитання щодо проекту&body=Добрий день,"
    );
  };
  return (
    project && (
      <StripeProvider
publishableKey="pk_test_51PABK604MubCHuYuYx4aXQvAGo15cC23feDiLNXS
5jQsaofjuy2xlUxm0hFvCrKGNoJL5pJLE1UgVyYbXhCh9B1200yWow59rP">
        <View>
          <ScrollView style={{ height: "92%" }}>
            <TouchableOpacity
```

```
  style={styles.backButtonContainer}
  onPress={() => navigation.goBack()}
>
  <Ionicons name="arrow-back-outline" size={30} color="black" />
</TouchableOpacity>
<Image
  source={{ uri: project?.image[0]?.url }}
  style={{ width: "100%", height: 225 }}
/>
<View style={styles.infoContainer}>
  <Text style={{ fontFamily: "Montserrat-Bold", fontSize: 22 }}>
    {project?.name}
  </Text>
  <View style={styles.subContainer}>
    <Text
      style={{
        fontFamily: "Montserrat-Medium",
        color: Colors.PRIMARY,
        fontSize: 20,
      }}
    >
      {project?.contactPerson}
    </Text>
    <Text
      style={{
        color: Colors.PRIMARY,
        backgroundColor: Colors.UNDERPRIMARY,
        padding: 5,
```

```
      borderRadius: 6,
      fontSize: 16,
    }}
  >
    {project?.category.name}
  </Text>
</View>

<Text style={{ fontSize: 17, fontFamily: "Montserrat-Medium" }}>
  <Entypo name="location" size={25} color={Colors.PRIMARY} />
  {project?.adress}
</Text>

<View
  style={{
    borderWidth: 0.5,
    borderColor: Colors.GRAY,
    marginTop: 10,
    marginBottom: 10,
  }}
></View>

<ProjectAbout project={project} />

<View
  style={{
    borderWidth: 0.5,
    borderColor: Colors.GRAY,
```

```
        marginTop: 10,
        marginBottom: 10,
      }}
    ></View>

    <ProjectPhotos project={project} />
  </View>
</ScrollView>
<View
  style={{ display: "flex", flexDirection: "row", margin: 3, gap: 8 }}
>
  <TouchableOpacity
    style={styles.messageButton1}
    onPress={() => onMessageBtnClick()}
  >
    <Text
      style={{
        textAlign: "center",
        fontFamily: "Montserrat-Medium",
        fontSize: 16,
        color: Colors.PRIMARY,
      }}
    >
      Повідомлення
    </Text>
  </TouchableOpacity>
  <TouchableOpacity
    style={styles.messageButton2}
```

```
            onPress={() => setShowModal(true)}
          >
            <Text
              style={{
                textAlign: "center",
                fontFamily: "Montserrat-Medium",
                fontSize: 16,
                color: Colors.WHITE,
              }}
            >
              Приєднатися
            </Text>
          </TouchableOpacity>
        </View>

        <Modal animationType="slide" visible={showModal}>
          <HelpModal hideModal={() => setShowModal(false)} />
        </Modal>
      </View>
    </StripeProvider>
  )
 );
}
const styles = StyleSheet.create({
 backButtonContainer: {
   position: "absolute",
   zIndex: 10,
   padding: 20,
```

```
},
infoContainer: {
  padding: 20,
  display: "flex",
  gap: 7,
},
subContainer: {
  display: "flex",
  flexDirection: "row",
  gap: 6,
  alignItems: "center",
  flex: 1,
},
messageButton1: {
  padding: 15,
  backgroundColor: Colors.WHITE,
  borderWidth: 1,
  borderColor: Colors.PRIMARY,
  borderRadius: 99,
  flex: 1,
},
messageButton2: {
  padding: 15,
  backgroundColor: Colors.PRIMARY,
  borderWidth: 1,
  borderColor: Colors.PRIMARY,
  borderRadius: 99,
  flex: 1,
```

```
  },
});
```

Модуль ProfileScreen.jsx

```
import {
  View,
  Text,
  Image,
  FlatList,
  TouchableOpacity,
  Linking,
} from "react-native";
import React from "react";
import { useUser, useSession } from "@clerk/clerk-expo";
import Colors from "../../Utils/Colors";
import { MaterialIcons } from "@expo/vector-icons";
import { useNavigation } from "@react-navigation/native";

export default function ProfileScreen() {
  const { user } = useUser();
  const navigation = useNavigation();
  const { session } = useSession();
  const handleSignOut = () => {
    session.end();
  };
  const onMessageBtnClick = () => {
    Linking.openURL(
      "mailto:" +
```

```
        "vladislav.ggh@gmail.com" +
        "?subject=Відгуки та побажання задля покращення роботи застосунку&body=Добрий день,"
      );
    };
    const profileMenu = [
      {
        id: 1,
        name: "Головна",
        icon: "home",
        screen: "Home",
      },
      {
        id: 2,
        name: "Проекти",
        icon: "volunteer-activism",
        screen: "Projects",
      },
      {
        id: 3,
        name: "Зворотній зв'язок",
        icon: "mail",
        action: onMessageBtnClick,
      },
      {
        id: 4,
        name: "Вихід",
        icon: "logout",
```

```
    action: handleSignOut,
  },
];

return (
  <View>
    <View
      style={{
        padding: 20,
        paddingTop: 20,
        backgroundColor: Colors.PRIMARY,
      }}
    >
      <Text style={{ fontSize: 23, fontFamily: "Montserrat-Bold" }}>
        Профіль
      </Text>
      <View
        style={{
          display: "flex",
          justifyContent: "center",
          alignItems: "center",
          padding: 20,
        }}
      >
        <Image
          source={{ uri: user.imageUrl }}
          style={{ width: 90, height: 90, borderRadius: 99 }}
        />
```

```
    <Text
      style={{
        fontSize: 20,
        marginTop: 8,
        fontFamily: "Montserrat-Medium",
        color: Colors.WHITE,
      }}
    >
      {user.fullName}
    </Text>
  </View>
</View>
<View style={{ paddingTop: 40 }}>
  <FlatList
    data={profileMenu}
    renderItem={({ item, index }) => (
      <TouchableOpacity
        style={{
          display: "flex",
          flexDirection: "row",
          alignItems: "center",
          gap: 15,
          marginBottom: 50,
          paddingHorizontal: 15,
        }}
        onPress={() => {
          if (item.screen) {
            navigation.navigate(item.screen);
```

```
            } else if (item.action) {
              item.action();
            }
          }}
        >
          <MaterialIcons
            name={item.icon}
            size={35}
            color={Colors.PRIMARY}
          />
          <Text
            style={{
              fontFamily: "Montserrat-Medium",
              fontSize: 20,
            }}
          >
            {item.name}
          </Text>
        </TouchableOpacity>
      )}
    />
  </View>
 </View>
 );
}
```

Модуль GlobalApi.js

```
import { request, gql } from "graphql-request";
```

```
const MASTER_URL =
  "https://api-eu-central-1-shared-euc1-
02.hygraph.com/v2/clusfga9e0me307wevslds90h/master";

const getSlider = async () => {
  const query = gql`
    query GetSlider {
      sliders {
        id
        name
        image {
          url
        }
      }
    }
  `;
  const result = await request(MASTER_URL, query);
  return result;
};

const getCategories = async () => {
  const query = gql`
    query GetCategory {
      categories {
        id
        name
        image {
```

```
        url
      }
    }
  }
 `;
 const result = await request(MASTER_URL, query);
 return result;
};

const getProjectList = async () => {
 const query = gql`
   query getProjectList {
     projectsLists {
       id
       name
       phoneNumber
       contactPerson
       category {
         name
       }
       image {
         url
       }
       adress
       about
       progress
     }
   }
```

```
  `;
  const result = await request(MASTER_URL, query);
  return result;
};

const getProjectListByCategory = async (category) => {
  const query =
    gql`
    query getProjectList {
      projectsLists(where: { category: { name: "` +
    category +
    `" } }) {
        id
        name
        phoneNumber
        contactPerson
        category {
          name
        }
        image {
          url
        }
        adress
        about
        progress
      }
    }
  `;
```

```
  const result = await request(MASTER_URL, query);
  return result;
};

export default {
  getSlider,
  getCategories,
  getProjectList,
  getProjectListByCategory,
};
```

Модуль StripeApp.js

```
import {
  View,
  Text,
  StyleSheet,
  TextInput,
  Button,
  Alert,
  TouchableOpacity,
} from "react-native";
import React, { useState } from "react";
import Colors from "./Colors";
import { CardField, useConfirmPayment } from "@stripe/stripe-react-native";
import Heading from "../Components/Heading";

const API_URL = "http://localhost:5500";
```

```
const StripeApp = () => {
  const [email, setEmail] = useState();
  const [cardDetails, setCardDetails] = useState();
  const { confirmPayment, loading } = useConfirmPayment();
  const [note, setNote] = useState();

  const handlePayPress = async () => {
    if (!cardDetails?.complete || !email) {
      Alert.alert("Будь ласка заповніть дані та електронну адресу");
      return;
    }
    const billingDetails = {
      email: email,
    };
    try {
      const { clientSecret, error } = await fetchPaymentIntentClientSecret();
      if (error) {
        console.log("Неможливо провести оплату");
      } else {
        const { paymentIntent, error } = await confirmPayment(clientSecret, {
          type: "Card",
          billingDetails: billingDetails,
        });
        if (error) {
          Alert.alert(`Помилка оплати ${error.message}`);
        } else if (paymentIntent) {
          Alert.alert("Оплата успішна");
          console.log("Оплата успішна", paymentIntent);
```

```
      }
    }
  } catch (e) {
    console.log(e);
  }
};

const fetchPaymentIntentClientSecret = async () => {
  const response = await fetch(`${API_URL}/create-payment-intent`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
  });
  const { clientSecret, error } = await response.json();
  return { clientSecret, error };
};

return (
  <View>
    <View>
      <TextInput
        autoCapitalize="none"
        placeholder="E-mail"
        placeholderTextColor={Colors.GRAY}
        keyboardType="email-address"
        keyboardAppearance="default"
        onChange={(value) => setEmail(value.nativeEvent)}
```

```
    style={styles.input}
   />
   <CardField
    postalCodeEnabled={true}
    placeholders={{ number: "6969 6969 6969 6969" }}
    cardStyle={styles.card}
    style={styles.cardContainer}
    onChange={(cardDetails) => {
     setCardDetails(cardDetails);
    }}
   />
  </View>
  <View style={{ paddingTop: 20 }}>
   <Heading text={"Додайте коментар"} />
   <TextInput
    placeholder="Коментар"
    placeholderTextColor={Colors.GRAY}
    numberOfLines={4}
    multiline={true}
    style={styles.noteTextArea}
    onChange={(text) => setNote(text)}
   />
  </View>
  <TouchableOpacity
   onPress={handlePayPress}
   disabled={loading}
   style={styles.confirmButton}
  >
```

```
        <Text
          style={{
            textAlign: "center",
            fontFamily: "Montserrat-Medium",
            fontSize: 18,
            color: Colors.WHITE,
          }}
        >
          Допомогти
        </Text>
      </TouchableOpacity>
    </View>
  );
};

export default StripeApp;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "center",
    alignItems: "center",
  },
  noteTextArea: {
    borderWidth: 1,
    borderRadius: 15,
    textAlignVertical: "top",
    padding: 20,
```

```
    fontSize: 16,
    fontFamily: "Montserrat-Medium",
    borderColor: Colors.PRIMARY,
  },
  confirmButton: {
    padding: 20,
    borderRadius: 99,
    backgroundColor: Colors.PRIMARY,
    color: Colors.WHITE,
    marginTop: 20,
  },
  input: {
    backgroundColor: Colors.LIGHT_GRAY,
    borderRadius: 8,
    fontSize: 15,
    height: 50,
    padding: 10,
    marginTop: 20,
    fontFamily: "Montserrat-Bold",
  },
  card: {
    backgroundColor: Colors.LIGHT_GRAY,
  },
  cardContainer: { height: 50, marginVertical: 30 },
});
```

ДОДАТОК Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ БЛАГОДІЙНИХ ОРГАНІЗАЦІЙ

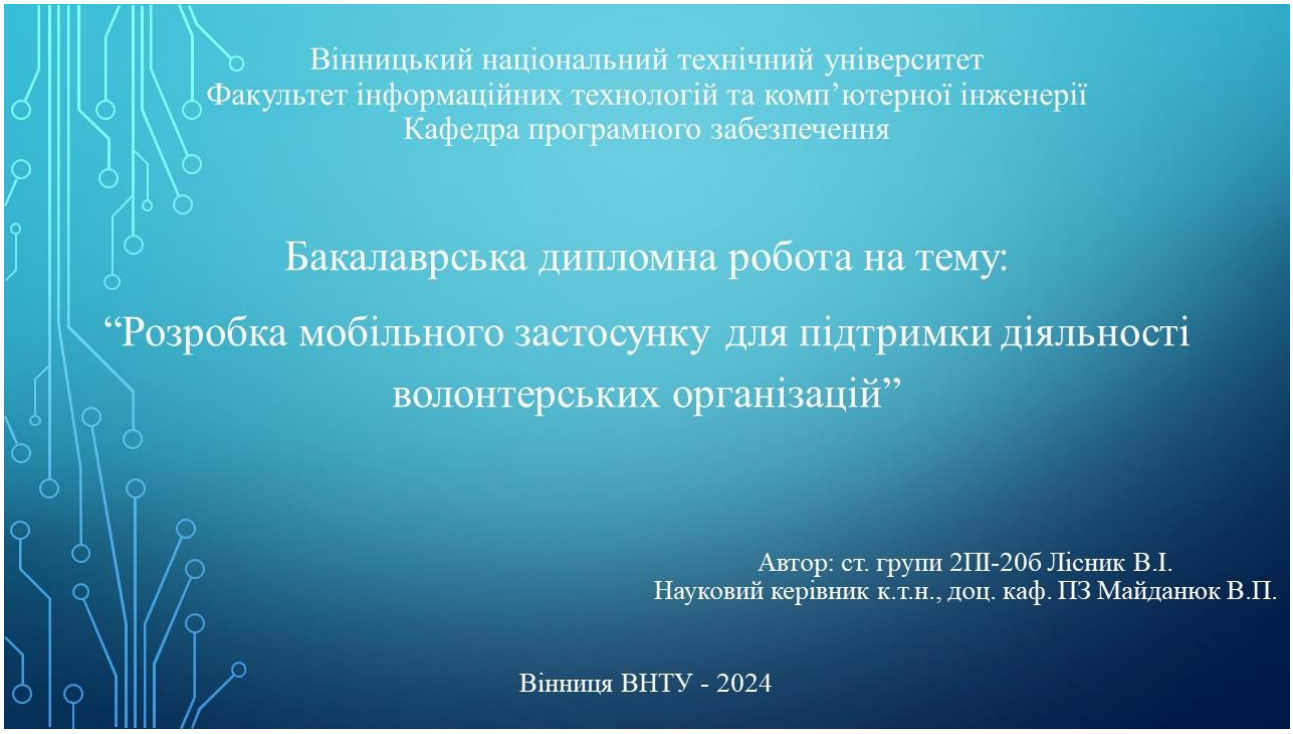

Рисунок Г.1 – Титульний слайд

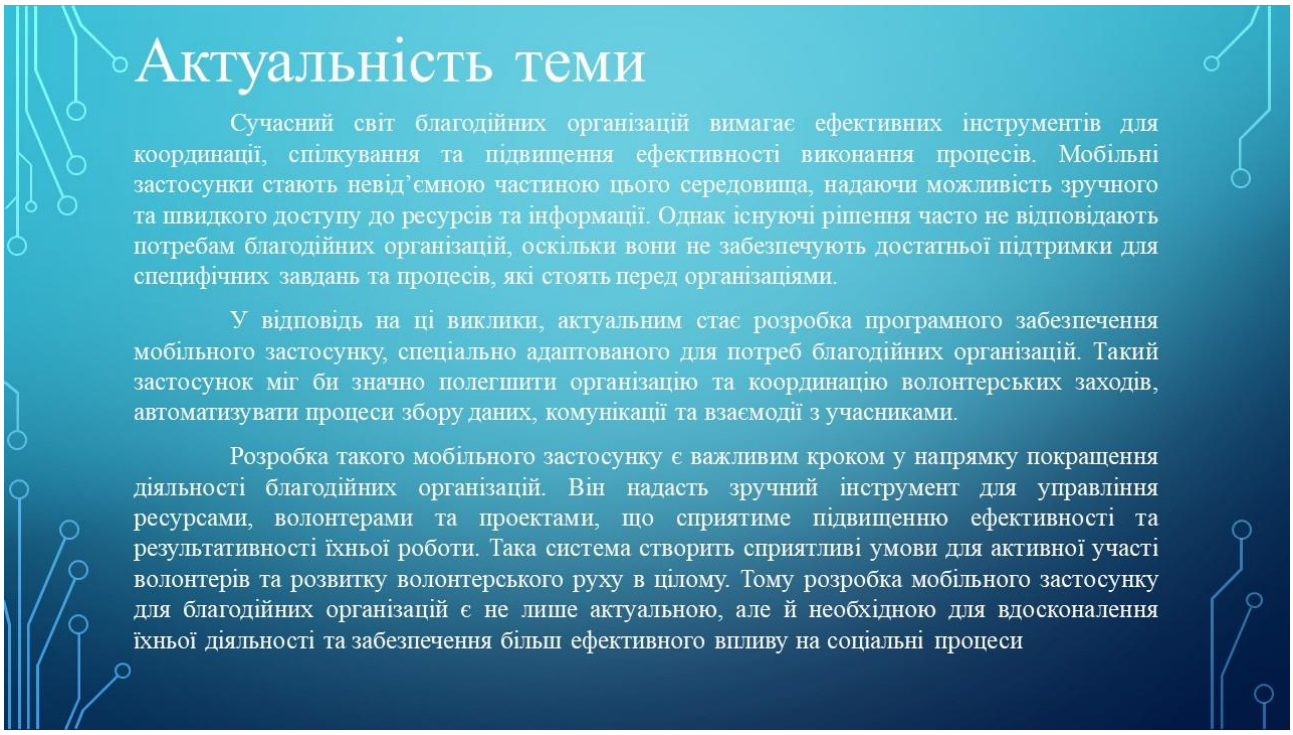

Рисунок Г.2 – Актуальність теми

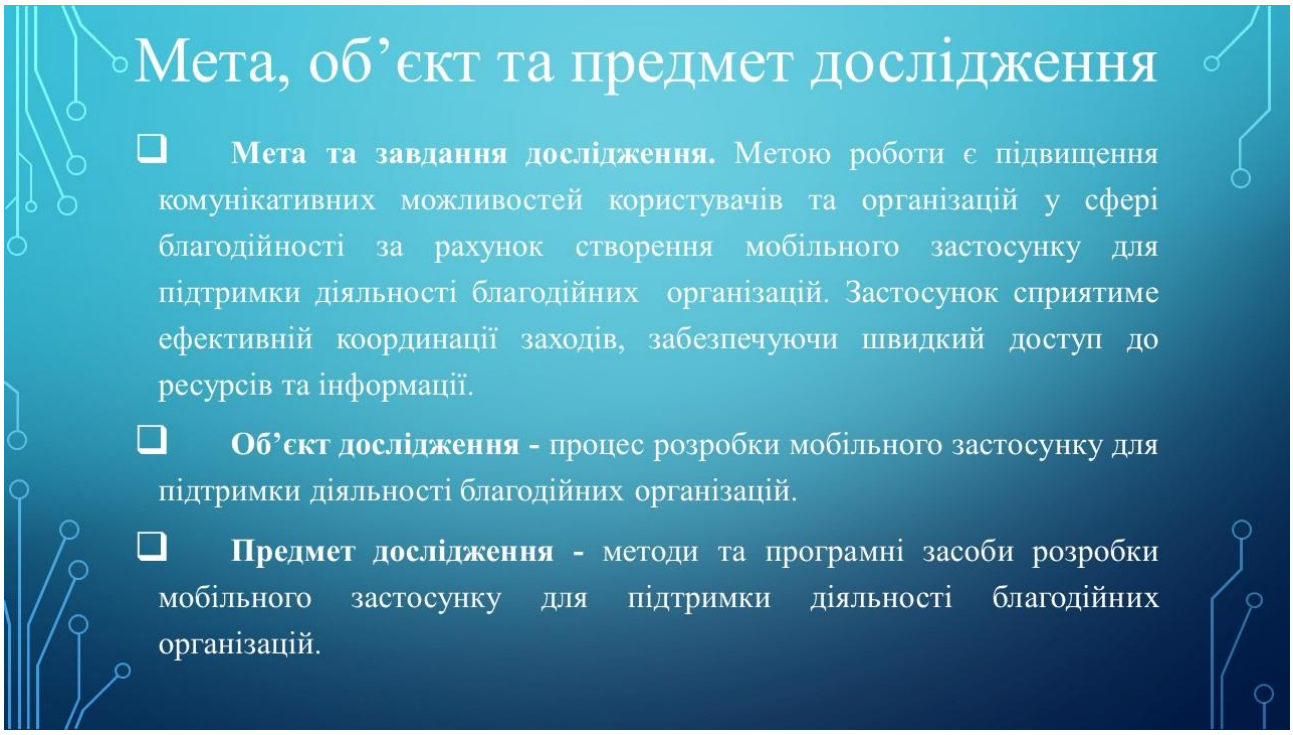

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

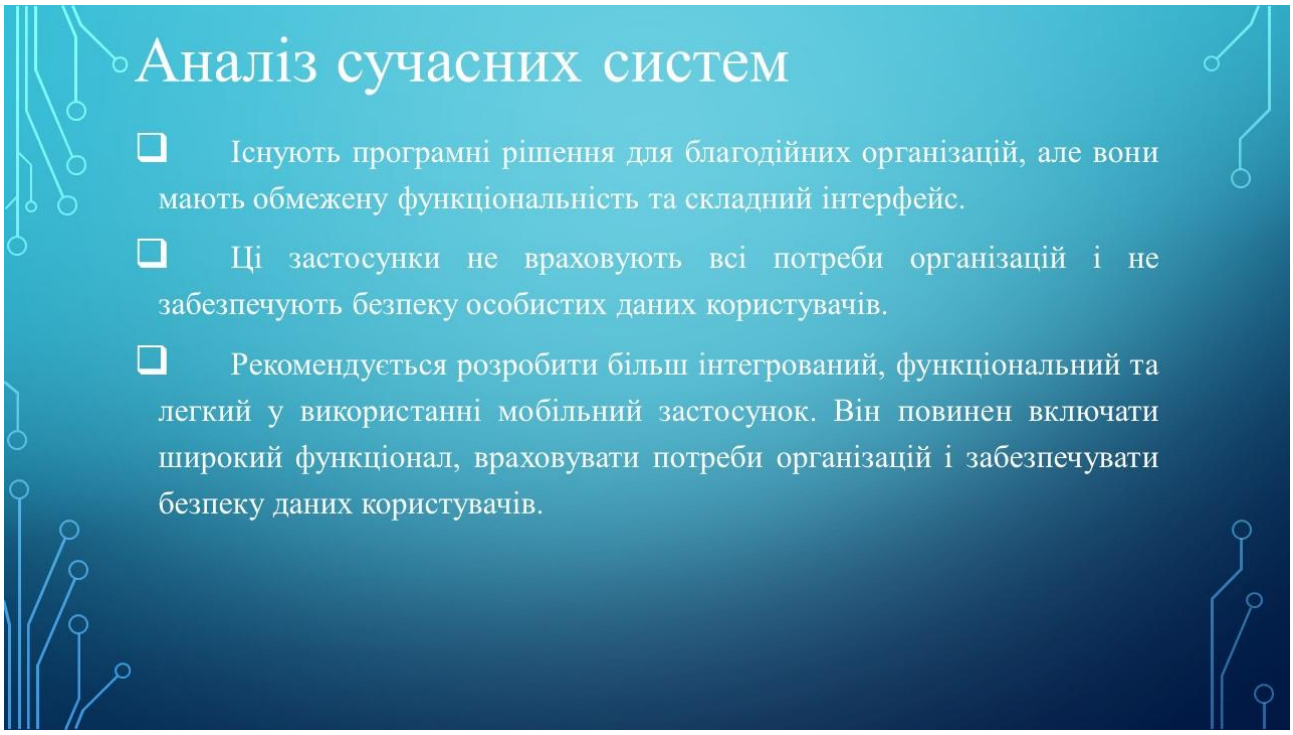

Рисунок Г.4 – Аналіз сучасних систем

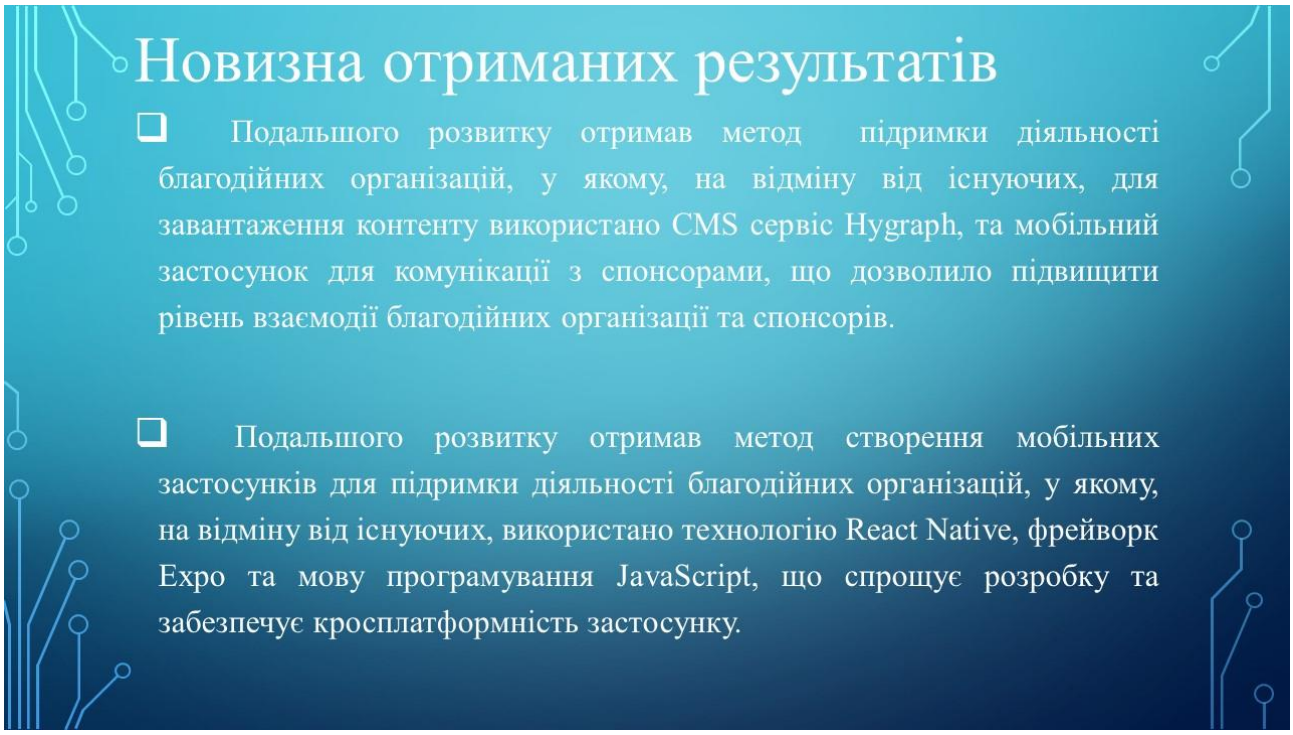

Рисунок Г.5 – Новизна отриманих результатів

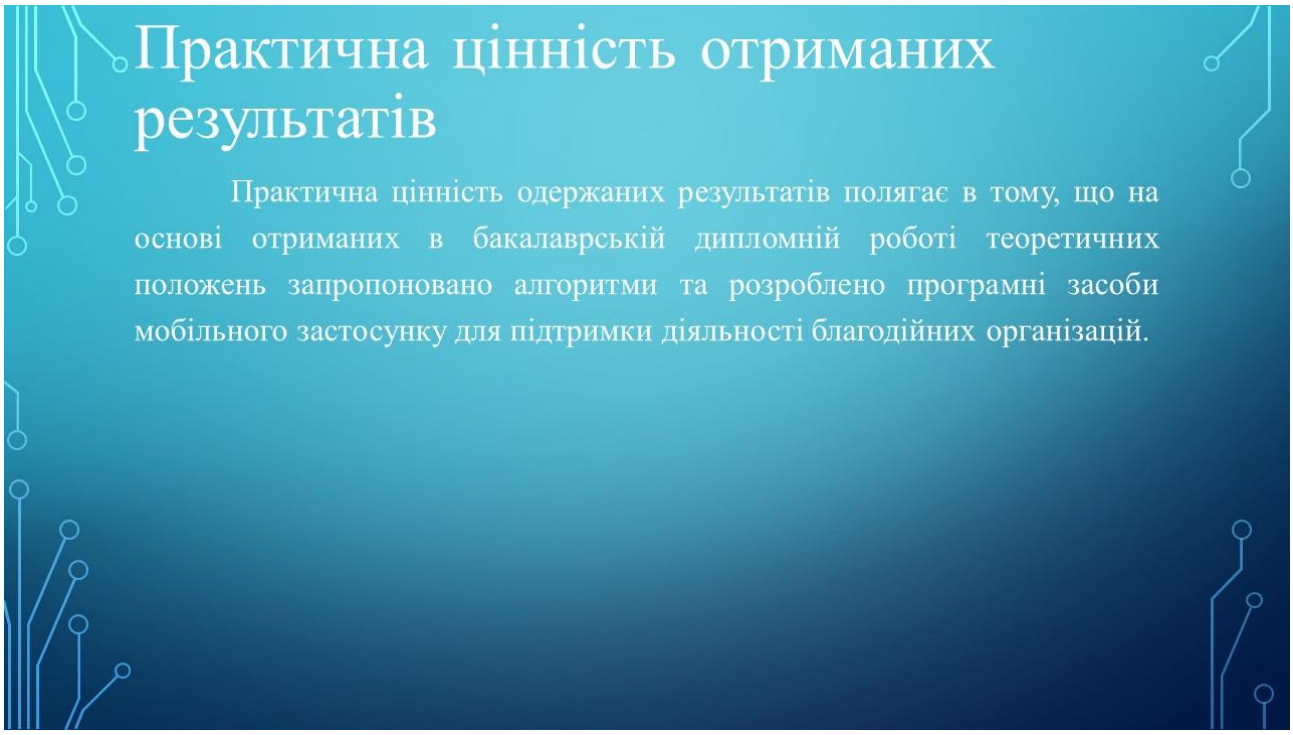

Рисунок Г.6 – Практична цінність отриманих результатів

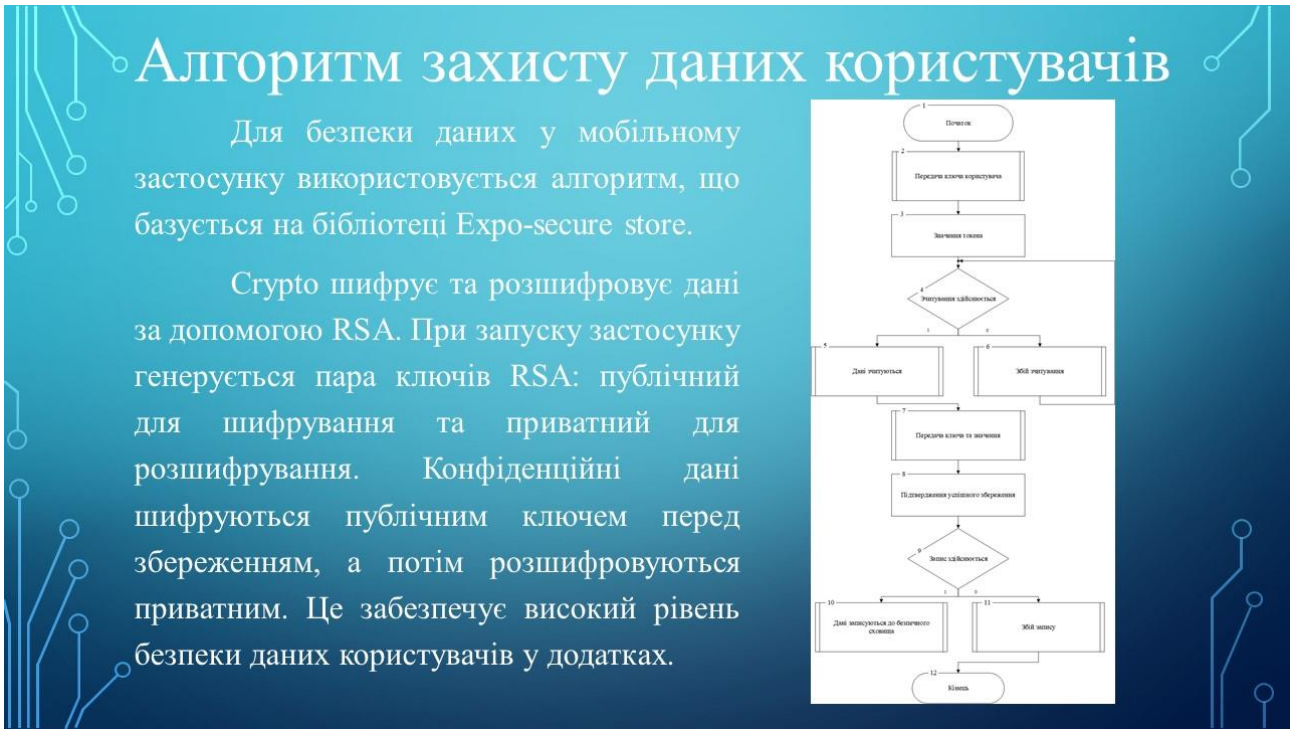

Рисунок Г.7 – Алгоритм захисту даних користувачів

Рисунок Г.8 – Структура мобільної системи

Рисунок Г.9 – Розробка графічного інтерфейсу

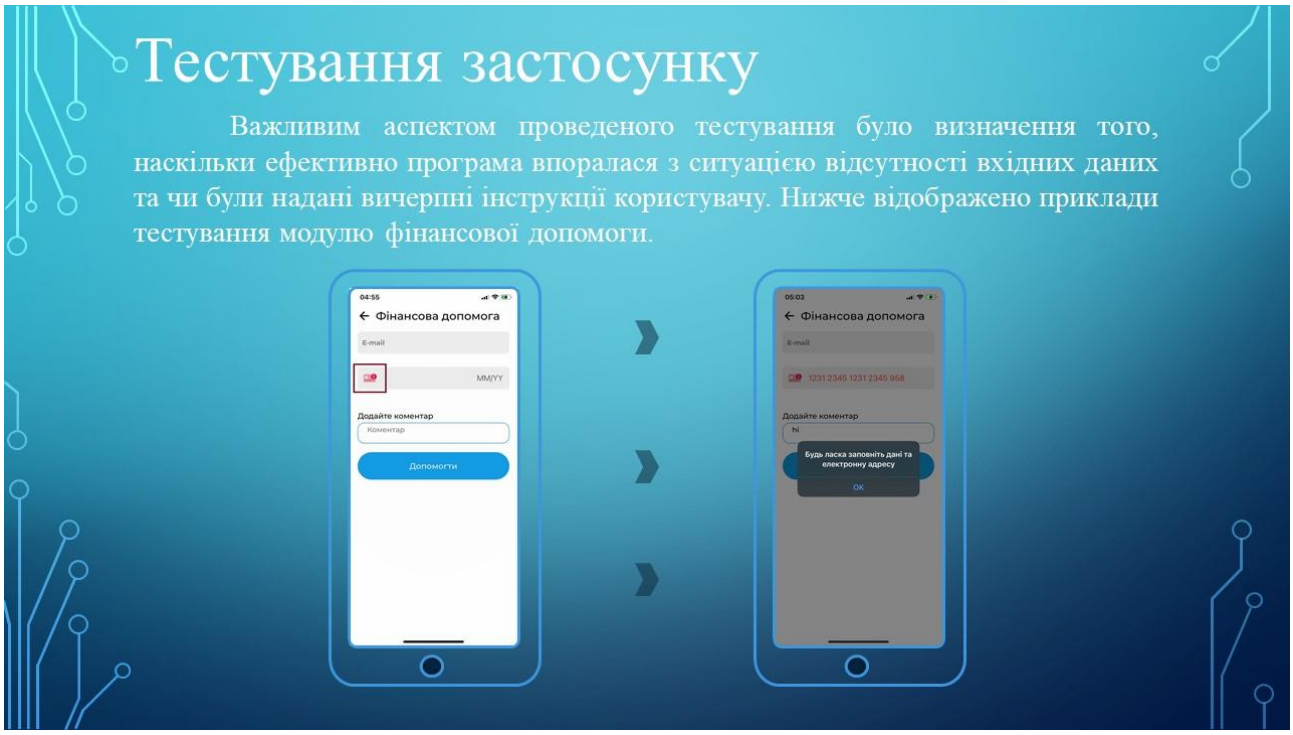

Рисунок Г.10 – Тестування застосунку

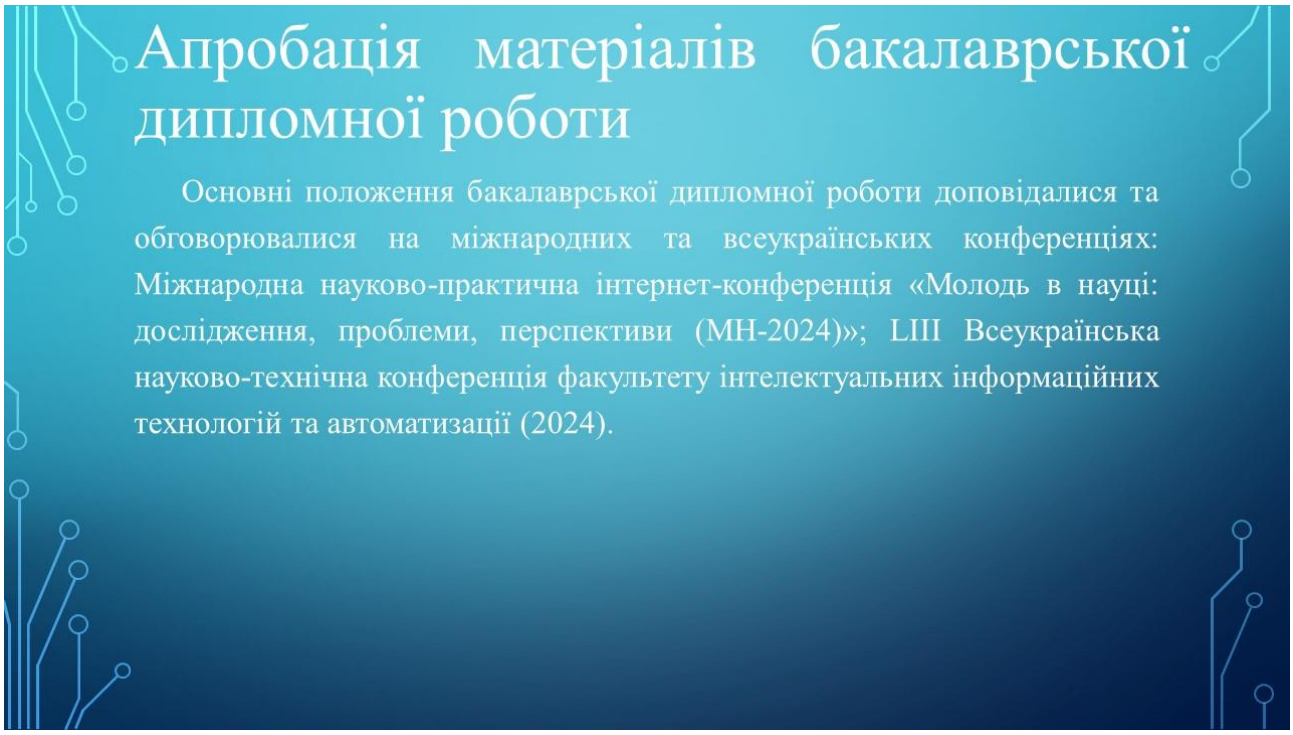

Рисунок Г.11 – Апробація матеріалів бакалаврської дипломної роботи

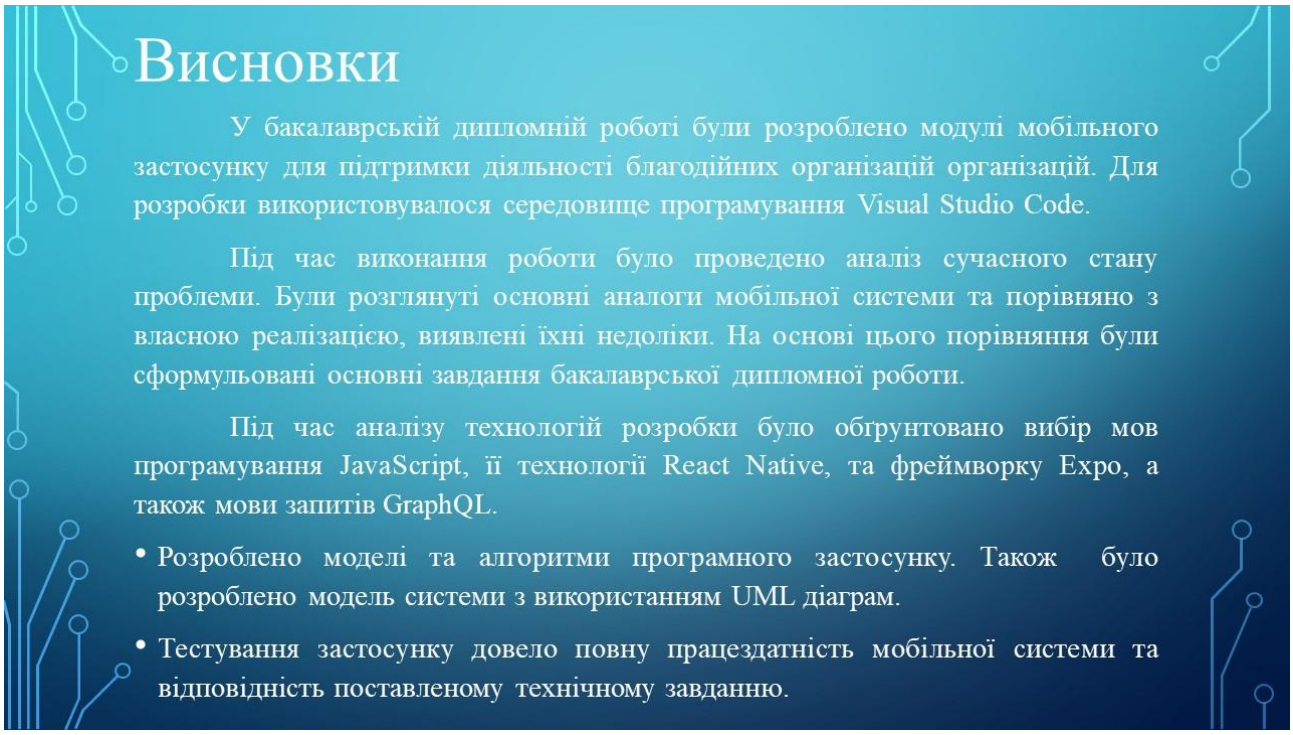

Рисунок Г.12 – Висновки