Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мобільного інтелектуального застосунку для аналізу особистих витрат з використанням технологій Kotlin та Compose»

Виконав: студент 4 курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Педосенко А.Ю.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д.І.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ЗІ Маліновський В.І.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«10» червня 2024 р.

ВНТУ – 2024

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
Романюк О. Н.
12 березня 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Педосенко Андрію Юрійовичу

1. Тема роботи – Розробка мобільного інтелектуального застосунку для аналізу особистих витрат з використанням технологій Kotlin та Compose.

Керівник роботи: Катєльніков Денис Іванович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 року №80.

2. Строк подання студентом роботи 3 червня 2024 року.

3. Вихідні дані до роботи: середовище розробки Android Studio, мова розробки – Kotlin, операційна система – Android.

4. Зміст розрахунково-пояснювальної записки: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка структури та алгоритмів мобільного застосунку; програмна розробка модулів мобільного застосунку; тестування мобільного застосунку; висновки; список використаних джерел.

5. Перелік графічного матеріалу: актуальність теми дослідження; мета, об'єкт та предмет дослідження; новизна та практична цінність отриманих результатів; основні задачі дослідження; аналоги; порівняння аналогів; засоби

для реалізації програмного забезпечення; архітектура даних застосунк загальний алгоритм роботи мобільного застосунку; алгоритм збереження дани алгоритм отримання порад; графічна частина; тестування мобільно інтелектуального застосунку; висновки; апробація та публікація результат робот застосунку; програмна реалізація мобільного застосунку; тестуванн висновки, апробація, публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийня
1-4	Катєльніков Д. І., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз проблеми, обґрунтування актуальності розробки системи та постановка задачі	14.03.24 – 20.03.24	вик.
2	Розробка архітектури та алгоритмів роботи мобільного застосунку	13.03.24 – 27.03.24	вик.
3	Вибір середовища та мови розробки	27.03.24 – 03.04.24	вик.
4	Розробка мобільного інтелектуального застосунку	03.04.24 – 12.05.24	вик.
5	Тестування мобільного застосунку	12.05.24 – 21.05.24	вик.
6	Оформлення матеріалів до захисту БДР	21.05.24 – 30.05.24	вик.

Студент _____ (підпис) Педосенко А.Ю (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) Катєльніков Д. (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 94 сторінок формату А4, містить 35 рисунків, 3 таблиці, список використаних джерел містить 22 найменувань.

У бакалаврській дипломній роботі було проаналізовано наявні мобільні інтелектуальні застосунки для аналізу особистих фінансів, також сформовано завдання та мету, предмет і об’єкт дослідження. Було запропоновано власну розробку, яка задовольняє потреби людей, які бажають збільшити контроль власних фінансів та оптимізувати витрати.

Розроблено архітектура, алгоритми мобільного застосунку для аналізу та генерування порад штучним інтелектом.

Мобільний застосунок був розроблений за допомогою мови програмування Kotlin у середовищі розробки Android Studio. Для створення графічного інтерфейсу був використаний фреймворк Compose. Кінцевий програмний продукт відповідає завданням та меті. У підсумку було отримано мобільний додаток, швидкодія та коректність роботи, якого була перевірена.

Ключові слова: мобільний інтелектуальний застосунок, фінанси, прибуток, витрати, аналіз, генерація порад, штучний інтелект.

ANNOTATION

The bachelor thesis consists of 94 pages of A4 format, contains 35 figures, 3 tables, the list of used sources contains 22 items.

In the bachelor's thesis, the existing mobile intelligent applications for the analysis of personal finances were analyzed, as well as the task and purpose, subject and object of the research were formed. A proprietary development was proposed that meets the needs of people who wish to increase control over their own finances and optimize costs.

The architecture and algorithms of a mobile application for analysis and generation of advice by artificial intelligence have been developed.

The mobile application was developed using the Kotlin programming language in the Android Studio development environment. The Compose framework was used to create the graphical interface. The final software product meets the tasks and purpose. As a result, a mobile application was obtained, the speed and correctness of work of which was checked.

Keywords: mobile intelligent application, finance, profit, costs, analysis, generation of advice, artificial intelligence.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі технології розвиваються занадто швидко, щоб людина була в змозі контролювати все одночасно, а здатність ефективно аналізувати та планувати власні фінанси є неймовірно необхідним вмінням для досягнення фінансової стабільності та подальшої незалежності у житті кожної людини. Тож проблема аналізу особистих витрат завжди залишається актуальною.

З стрімким розвитком мобільних пристроїв та зростаючим попитом на динамічні додатки з мінімальним розміром, різко набула популярності тенденція з впровадженням штучного інтелекту у мобільні додатки. Завдяки цій тенденції у мобільній розробці настала нова ера інновацій та персоналізації [1]. Штучний інтелект революціонізував підхід до розробки мобільних додатків: дизайну, розробки, підтримки, надаючи змогу розробникам створювати більш інтуїтивно зрозумілий і більш дружелюбний до користувача досвід [2].

Наявні аналогічні програмні забезпечення зі штучним інтелектом не надають достатнього корисного функціоналу. Аналіз фінансів користувача є не точним і вимагає великої кількості даних, а якщо прийняти до уваги той факт, що більшість подібних рішень є платними.

Чим більше функціональних можливостей містить мобільний інтелектуальний застосунок, тим ефективніше його аналіз та подальше генерування порад користувачу. Основна функція мобільного застосунку це аналіз витрат користувача та отримання порад від штучного інтелекту для оптимізації фінансів. Також основна вимога подібних мобільних застосунків це можливість додавати транзакції користувача для більш точного аналізу та генерації порад. Проте взаємодія зі штучним інтелектом вимагає інтернет з’єднання.

Отже, розробка мобільного інтелектуального застосунку для аналізу особистих витрат є актуальною, завдяки функціональності та доступності.

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є підвищення якості аналізу фінансів користувача та покращення фінансового стану в цілому за рахунок використання штучного інтелекту, що дозволяє та сприяє глибшому аналізу та більш якісним порадам. Відповідно до вищеописаної мети потрібно виконати наступні завдання:

- проаналізувати стан галузі мобільних додатків для аналізу особистих витрат;
- провести порівняльний аналіз аналогів;
- проаналізувати вибір технологій та середовища розробки;
- провести постановку задач мобільного інтелектуального застосунку для аналізу особистих витрат;
- розробити структури та алгоритми програмного продукту;
- провести варіантний аналіз і обґрунтувати вибір засобів для реалізації програмного засобу;
- розробити мобільний інтелектуальний застосунок для аналізу особистих витрат;
- провести тестування мобільного застосунку.

Об’єктом дослідження є процес розробки мобільного інтелектуального застосунку для аналізу особистих витрат.

Предметом дослідження є алгоритми і засоби реалізації мобільного інтелектуального застосунку для аналізу особистих витрат.

Методи дослідження. Протягом дослідження було використано: теорія людино-машинної взаємодії, теорія баз даних при організації зберігання інформації, теорія розподілених систем для побудови мобільного інтелектуального застосунку.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод аналізу особистих витрат, у якому на відміну від існуючих методів аналізу, застосовується модель

породжувального штучного інтелекту, що дозволило підвищити якість фінансового аналізу.

2. Подальшого розвитку отримав метод управління даними користувача, у якому на відміну від існуючих методів управління даними, застосовується двошарова організація потоку інформації: перший шар "data" відповідає за роботу з базою даних та штучним інтелектом, а другий шар "domain" відповідає за семантику та відображення інформації, що дозволяє швидко оброблювати велику кількість даних та успішно проводити необхідні маніпуляції для отримання бажаного результату.

Практична цінність отриманих результатів. Розроблено алгоритми і засоби реалізації мобільного інтелектуального застосунку для аналізу особистих витрат.

Апробація результатів роботи. Основні положення бакалаврської роботи доповідалися та обговорювалися на міжнародній конференції: «Сучасні тенденції розвитку фінансових та інноваційно-інвестиційних процесів в Україні (2024)» [3].

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану мобільних застосунків з елементами штучного інтелекту для аналізу особистих фінансів

Контроль над особистими фінансами для більшості людей – головний біль. Проте, такі застарілі методи, як використання електронних таблиць для відстеження витрат і бюджету більше не мають бути частиною списку справ користувача – існують інструменти штучного інтелекту, які можуть це робити за людину. Моделі штучного інтелекту можуть уніфікувати онлайн-банківські рахунки, відстежувати витрати, а потім на основі цих даних надавати персоналізовані рекомендації щодо оптимізації заощаджень, накопичення капіталу [4].

Однією з основних переваг використання штучного інтелекту є перекладання рутинних занять на більш досвідченого «помічника», який буде аналізувати дані на величезній кількості даних, оскільки його досвід не обмежується одним єдиним, як у людини. Штучний інтелект може бути використаним у великій кількості сфер: фінансів, освіти, медицини, розваг, торгівлі, армії тощо. Однак розробка програмного забезпечення для інтеграції штучного інтелекту в мобільні застосунки є доволі складним завданням, що вимагає спеціалізованих знань та навичок.

Для вирішення цієї проблеми, люди з потенційно-інноваційними ідеями звертаються до спеціалізованих компаній, для найму професіональних спеціалістів для розробки власних продуктів з інтеграцією штучного інтелекту. Саме ці спеціалісти спрощують та гарантують якісний продукт для подальшого користування

Також доволі важливою перевагою є сумісність з іншими сервісами. Наприклад, інтеграція штучного інтелекту у сервіси навігації, на основі даних він може в реальному часі будувати найкращий маршрут та економити величезну

кількість часу. Також гарним прикладом є інтеграція у додатки з новинами, штучним інтелект може аналізувати коментарі або цілі статті на наявність неправдивої інформації та маніпуляцій.

Отже, розробка мобільного інтелектуального застосунку для аналізу особистих витрат є надзвичайно важливим та корисним для суспільством завданням. За допомогою спеціалізованих компаній, люди отримують змогу створити революційне рішення для покращення життя людей та їх комфорту.

1.2 Порівняльний аналіз аналогів

Використання штучного інтелекту неймовірно швидко стало популярним та інтегрується на всіх платформах: веб-сайти, комп'ютерні програми, годинники, телевізори, побутова техніка, мобільні платформи не стали виключенням. Розробка мобільного інтелектуального застосунку для аналізу особистих витрат може спростити та підвищити якість життя безліч людей. Найбільш популярні рішення:

- wally;
- Emma;
- AI Expense Tracker;
- Expense Tracker.

Першим із прикладів використання штучного інтелекту для аналізу особистих фінансів є "wally" [5]. Безкоштовний мобільний додаток для керування витратами. Додаток дуже зручний для користувачів, завдяки вдалому та приємному користувачу інтерфейсу (рис. 1.1). Проте унеможливлює користування наявність тільки однієї мови – португальської. Тобто, наявність тільки однієї мови ускладнює користування додатком та перекриває усі плюси для не людей, які не розмовляють португальською.

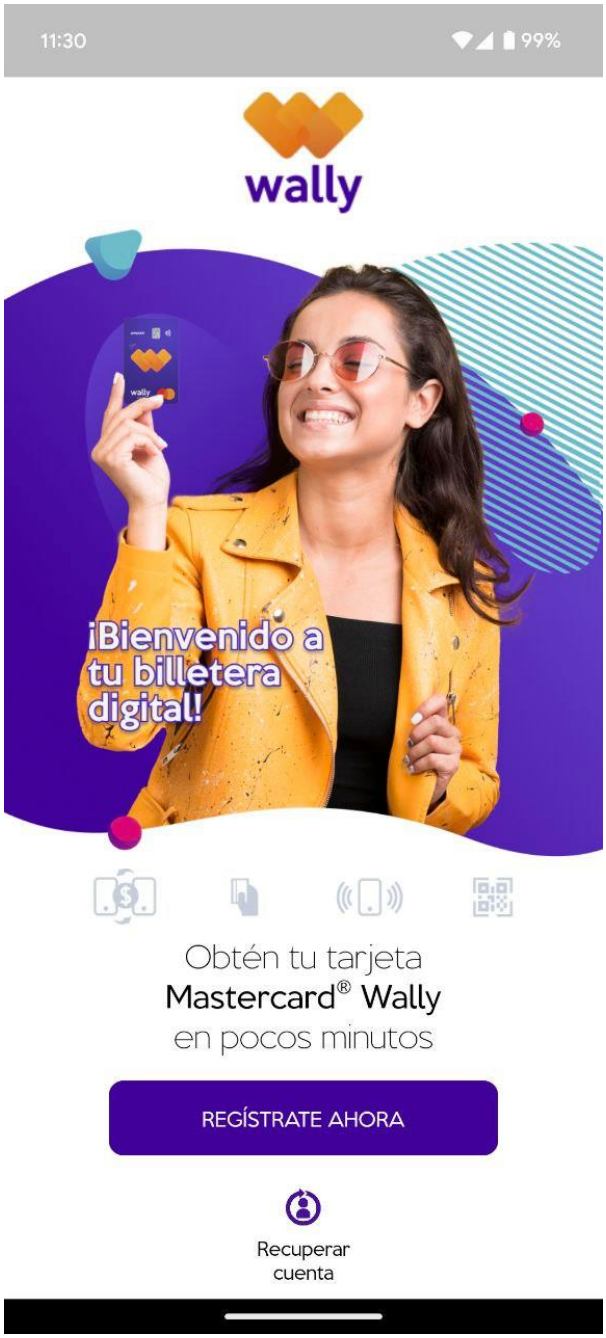

Рисунок 1.1 – Приклад роботи додатку “wally”

Наступний приклад – додаток Emma [6], який в свій час став революційним, завдяки вдалому графічному інтерфейсу, інтеграцією з банками та корисним вбудованим асистентом "Emma Assistant”. Не дивлячись на переваги вище описаного додатку, користування ним великій кількості людей унеможливлює лімітований вибір банків, серед яких немає українських продуктів (рис. 1.2).

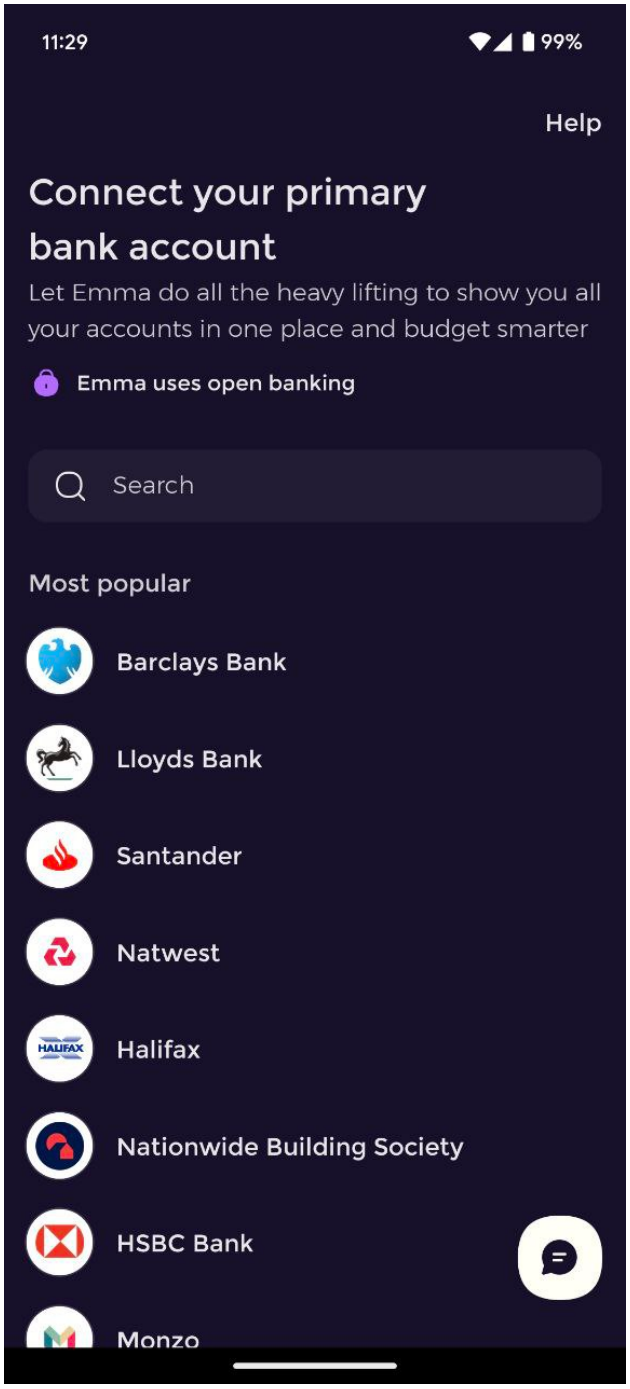

Рисунок 1.2 – Приклад роботи додатку “Emma”

Ще одним умовно безкоштовним додаток для керування фінансів є – “AI Expense Tracker” [7]. Менш привабливий графічний інтерфейс, але це не заважає наявності широкого функціоналу. Проте, користувач отримає доступ до всього функціоналу тільки у випадку платної підписки, а без неї йому прийдеться терпіти рекламні банери (рис. 1.3) та вставки на кожну дію, починаючи з переходу на нові екрани, закінчуючи перезапуском додатку.

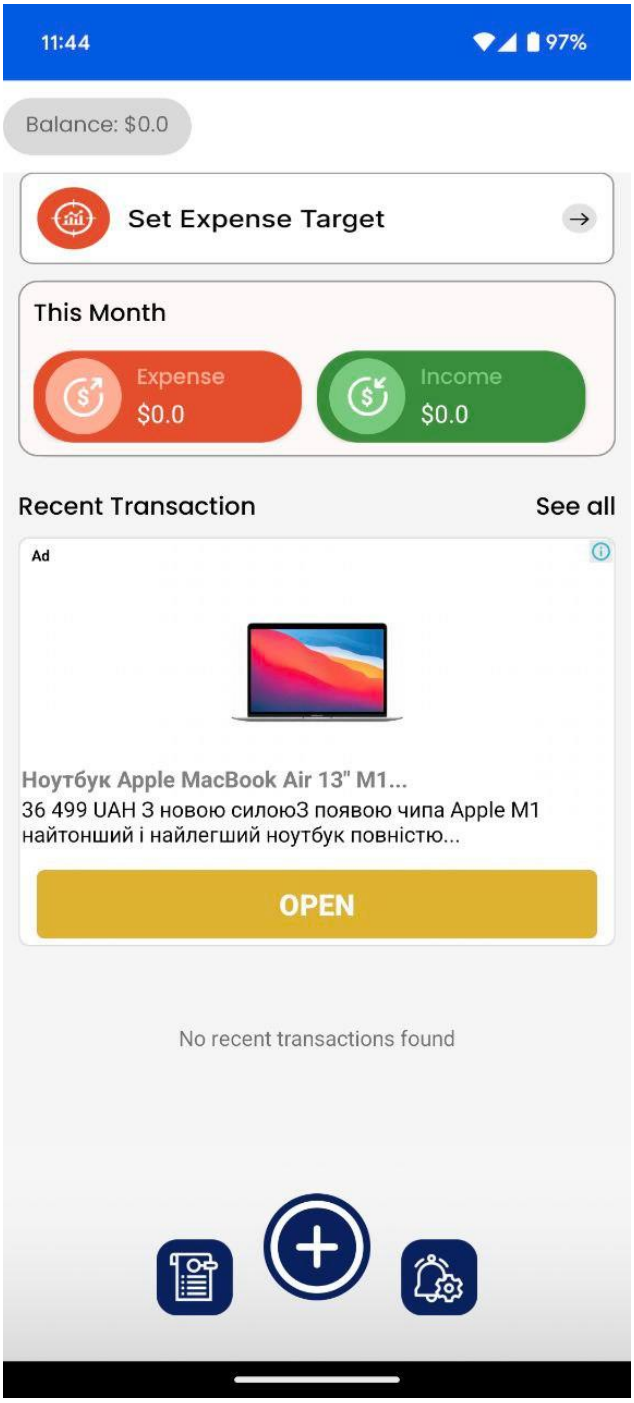

Рисунок 1.3 – Приклад роботи додатку “AI Expense Tracker”

Останнім додатком є “Expense Tracker” [8]. Мобільний застосунок має надзвичайно приємний інтерфейс. З існуючого функціоналу не тільки додавання фінансових операцій, їх аналізу та порад, а й додавання транзакції за текстом та максимального спрощення користування додатком. Проте, більшість з існуючих функцій користувач отримає тільки після купівлі повної версії, а без неї буде страждати від постійної реклами (рис. 1.4), що не уможливлює користування.

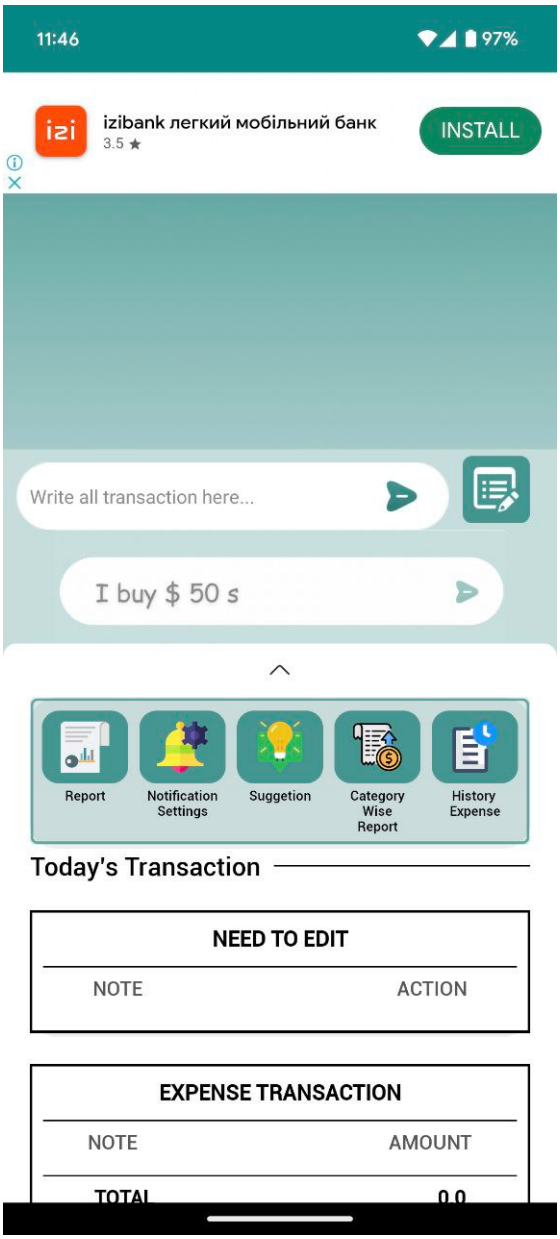

Рисунок 1.4 – Приклад роботи додатку “Expense Tracker”

Розробка мобільного інтелектуального застосунку для аналізу особистих витрат потребує немалого досвіду в мобільній розробці та роботі з штучним інтелектом. Це передбачає створення модулів:

- управління даними, для зберігання даних користувача;
- аналізу та отримання порад від моделі штучного інтелекту, для безперешкодного отримання коректних даних;
- графічного модуля, для створення приємного та інтуїтивно зрозумілого графічного інтерфейсу.

Результати порівняння аналогів продемонстровано в таблиці 1.1

Таблиця 1.1 – Порівняльні характеристики рішень

Критерії	wally	Emma	AI Expense Tracker	Expense Tracker	Власне програмне забезпечення
Наявність штучного інтелекту	+	+	+	+	+
Встановлення фінансових цілей	+	+	-	-	+
Підтримка багатомовності	-	+	-	-	+
Функції статистики	-	+	-	+	+
Експорт даних	+	-	+	-	+
Загальна оцінка	60%	80%	40%	40%	100%

Відповідно до таблиці порівняльних характеристик розробка мобільного інтелектуального застосунку для аналізу особистих витрат є доцільною. Кінцевий продукт покриє недоліки існуючих рішень та забезпечить позитивний досвід та можливості застосування штучного інтелекту для аналізу особистих фінансів.

Отже, розробка мобільного інтелектуального застосунку для аналізу особистих витрат дає змогу як і звичайним користувачам покращити своє життя, такі і бізнесу – підвищити рівень залученості. Проаналізувавши потенційні можливості застосування штучного інтелекту, бізнес може створювати та

інтегрувати інноваційні програмні забезпечення для подальшого зростання та збільшення прибутку.

1.3 Аналіз методів розв’язання задачі

Розробка мобільного інтелектуального застосунку для аналізу особистих витрат вимагає детального огляду до вирішення проблеми. У цьому розділі проаналізовано найефективніші методи розробки програмних модулів.

Найбільш швидким в плані імплементації є використання існуючих бібліотек та фреймворків [9]. З одного це надзвичайно зручно, завантажити існуюче рішення і використовувати у своєму програмному забезпечення, але цей підхід має свої проблеми, такі як:

- потенційні проблеми з ліцензією;
- неможливість редагування коду та залежність від розробників модуля;
- потенційні проблеми з підтримкою у майбутньому.

Наступним методом є розробка власного модуля. Це найкращий підхід, у випадку якщо програмне забезпечення планують розширювати у майбутньому. Переваги майже у всьому:

- можливість редагувати та адаптовувати функціонал під всі необхідні потреби;
- безпроблемна підтримка у майбутньому;
- можливість розповсюдження з монетизацією;
- підтримка мультиплатформеності.

У цього метода є тільки один недолік – час розробки. Оскільки це власне рішення, розробники можуть витратити велику кількість часу на створення та тестування модуля, який задовольняє усі потреби. Проте, якщо час розробки це не проблема для замовника, то розробка власного модуля це найкращий варіант, який на виході легко та стабільно масштабується.

Розглянувши переваги та недоліки кожного методу, було вирішено обрати метод розробки власного модуля для реалізації мобільного додатку з використанням штучного інтелекту для аналізу особистих фінансів.

1.4 Постановка задач бакалаврської дипломної роботи

Проаналізувавши переваги та недоліки існуючих рішень з використанням штучного інтелекту, було визначено наступні завдання, які потрібно виконати для розробки успішного мобільного додатку:

- визначити найкращу модель штучного інтелекту для аналізу та отримання даних;
- розробити модуль управління даними користувача;
- розробити модуль для аналізу та отримання даних від моделі штучного інтелекту;
- розробити графічний модуль;
- провести тестування мобільного інтелектуального застосунку згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан мобільних додатків з елементами штучного інтелекту для аналізу особистих фінансів. Було проведено порівняння існуючих рішень, таких як: “wally”, “Emma”, “AI Expense Tracker”, “Expense Tracker”.

Після порівняння було виявлено причину популярності серед учасників ринку штучного інтелекту, а саме:

- приємний інтерфейс;
- функціоналу;
- доступності.

Проаналізувавши переваги та недоліки існуючих мобільних додатків, було виявлено що вони не мають підтримки мультимовності, встановлення

фінансових цілей та повноцінного користування без преміум версій. Таким чином, було доведено доцільність розробки власного мобільного застосунку. На основі отриманої інформації був створений перелік задач, які необхідно виконати для розробки власного мобільного додатку.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Розробка методу управління даними користувача

Для розробки мобільного інтелектуального застосунку для аналізу особистих витрат будуть створені 2 шари даних: “data” та “domain” [10]. Це необхідно для спрощення розробки та більш гнучкого управління даними. Для взаємодії цих шарів будуть створенні спеціальні класи – “Mapper” [11], вони відповідають за перетворення даних з одного шару у інший без втрат даних.

Перший шар даних, а саме “data” буде відповідати за роботу з базою даних та штучним інтелектом. В ньому програма буде оперувати більш простими типами даними, такими як “String”, “Float”, “Long”, “Int”. Оскільки тільки такі типи даних можна зберігати в базах даних або передавати штучному інтелекту. Моделі цього шару мають в назві “Entity”, позначку про належність до відповідного шару.

Другим шаром даних є “domain”. Ці дані більш комплексні та вони використовуються для опису, наприклад категорій, оскільки вони містять назву, зображення, колір. Моделі цього шару не мають жодної приставки, тому що вони вже є фінальними та готовими даними для роботи з графічного показу.

Для кращого розуміння можна навести приклад з створенням транзакції користувача. Першим кроком користувач вводить необхідні дані на екрані створення, на цьому етапі використовуються дані domain шару. Коли користувач ввів необхідні дані і вирішив зберегти, то використовується клас “Mapper”, який перетворює комплексні дані “domain” шару у дані “data” шару, які у свою чергу зберігаються локально. Коли користувач захоче побачити існуючі транзакції, програма почне «виймати» дані з бази даних у вигляді “data” та перетворювати у “domain” для графічного показу за допомогою створеного класу “Mapper”. Візуальне зображення структури даних зображено на рисунку 2.1.

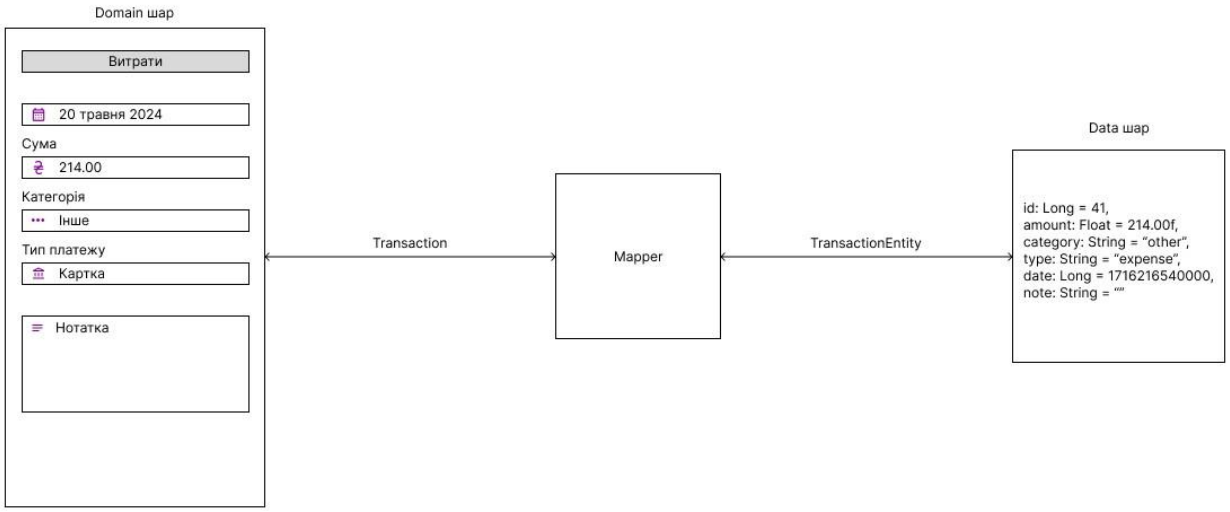

Рисунок 2.1 – Візуальне зображення структури даних

Подальша розробка програмних модулів для реалізації мобільного інтелектуального застосунку для аналізу особистих витрат передбачає декілька етапів. По-перше, необхідно створити модуль для управління даними користувача. Цей модуль буде відповідати за збір та безпечне зберігання даних користувача, що підвищить рівень захисту застосунку та покращить подальше використання для відображення користувачу та генерації порад штучним інтелектом.

Після отримання усіх необхідних даних від користувача про його витрати та доходи, буде розроблений модуль для аналізу та отримання даних від моделі штучного інтелекту. Наступним буде модуль для графічного відображення даних. Оскільки дані отримані від моделі штучного інтелекту можуть виглядати незрозумілими для звичайного користувача, необхідно відобразити їх у максимально просто та візуально приємному вигляді.

Отже, розробка мобільного інтелектуального застосунку для аналізу особистих витрат є комплексним завданням, яке включає декілька етапів. Маючи в наявності необхідний досвід та навички можна створити мобільний застосунок, який змінить уявлення про керування фінансами. Для досягненні цієї мети модулі повинні правильно спроектовані, розроблені та протестовані.

2.2 Розробка моделі системи

Для кращого розуміння роботи мобільного інтелектуального застосунку для аналізу особистих витрат було вирішено створити модель роботи системи на прикладі UML-діаграми [12] діяльності (рис. 2.2).

Відповідно до діаграми, після входу у застосунок користувач повинен додати усі його транзакції, на основі яких він хоче отримати поради від моделі штучного інтелекту. Наступним кроком є введення даних для самого аналізу, які допоможуть штучному інтелекту коректно згенерувати поради. Після натиснення відповідної кнопки для генерації, застосунок перевірить чи користувач авторизований, у випадку наявності акаунта, застосунок почне створення порад, у інакшому випадку, користувач буде переправлений на екран авторизації. Після успішної генерації даних користувачу залишається тільки переглянути створені поради.

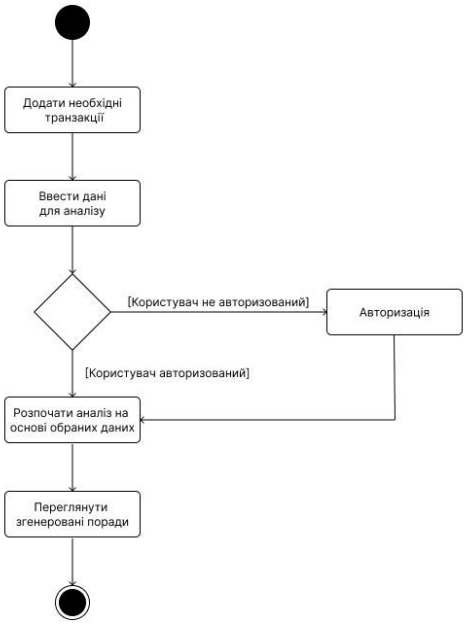

Рисунок 2.2 – Діаграма діяльності мобільного застосунку

2.3 Розробка методу аналізу особистих витрат

Для розробки мобільного інтелектуального застосунку для аналізу особистих витрат, необхідно розробити загальний алгоритм, згідно якого буде працювати застосунок (рис. 2.3).

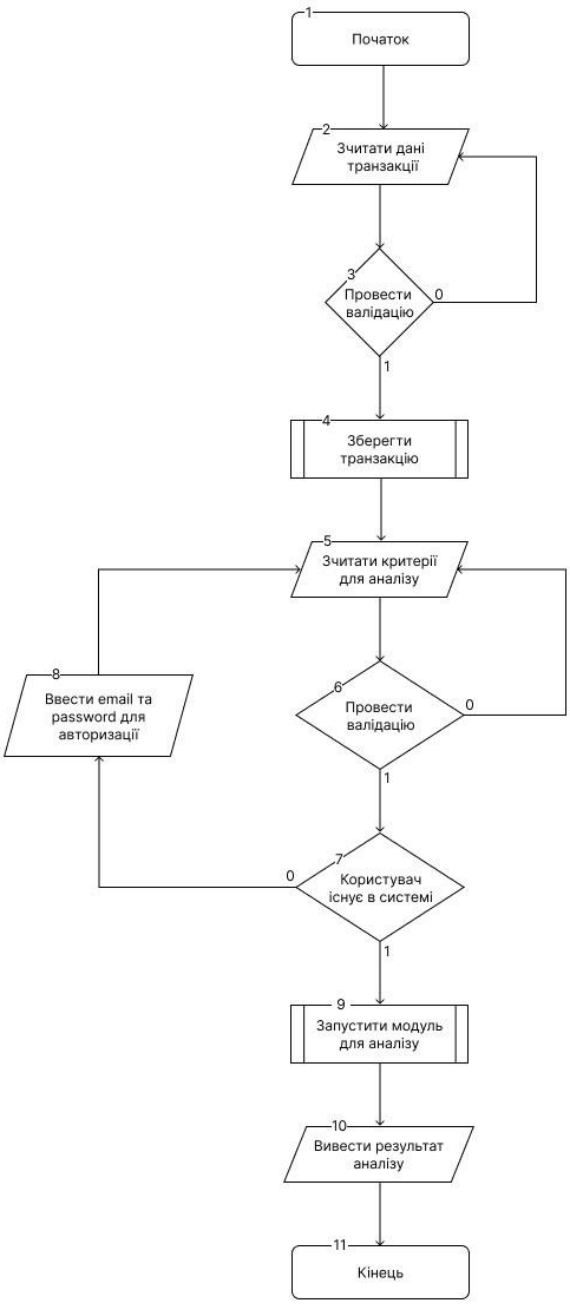

Рисунок 2.3 – Загальний алгоритм мобільного застосунку

Згідно створеного алгоритму перш за все користувач додає свої існуючі транзакції, перед додаванням введенні критерії проходять валідацію, після успішної валідації дані зберігаються локально для подальшої взаємодії зі штучним інтелектом. Наступним кроком є вибір користувачем критеріїв за якими він хоче отримати поради від штучного інтелекту. Після вибору необхідних критеріїв, які також проходять валідацію, застосунок перевіряє чи

існує користувач в системі, при відсутності акаунта, застосунок переправляє користувача на екран авторизації. При наявності користувача в системі, програма отримує раніше збережені дані та надсилає до моделі штучного інтелекту для аналізу та подальшого отримання і відображення. Отже, під час створень транзакцій запускається та відпрацьовує модуль для управління даними; для звертання до штучного інтелекту запускається модуль для аналізу та отримання даних від моделі штучного інтелекту.

Для кращого розуміння роботи модуля управління даними користувача було розроблено додатковий алгоритм (рис. 2.4).

Рисунок 2.4 – Алгоритм збереження даних

Згідно створеного алгоритму першим кроком є зчитування даних з введених полів. Після успішного проходження валідації дані трансформуються у відповідний тип для бази даних та зберігаються в локальній базі даних. У

подальшому дані можуть використовуватися не тільки для аналізу штучним інтелектом, а й для відображення списком, побудовою графіків, деталей. Також підтримується зміна вже існуючих транзакцій, це відбувається шляхом отримання їх з локальної бази даних, трансформації та зміни користувачем. Змінені дані після завершення редагування знову проходять валідацію, трансформування та зберігаються способом наведеним у алгоритмі.

Для кращого розуміння роботи модуля для аналізу та отримання даних від моделі штучного інтелекту було створено додатковий алгоритм (рис. 2.5).

Рисунок 2.5 – Алгоритм отримання порад

Після отримання даних з локальної бази даних, вони трансформуються у спеціальні моделі для відправки на аналіз штучному інтелекту. Через деякий проміжок часу модель повертає дані і вони знову трансформуються, але на зрозумілий для людини вигляд, після якого поради готові до відображення.

2.4 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмного застосунку. Було розроблено модель роботи програмного забезпечення у вигляді UML діаграм. Також було розроблено основний алгоритм мобільного застосунку, алгоритм для управління даними користувача та алгоритм для аналізу та отримання даних від моделі штучного інтелекту.

3 ПРОГРАМНА РОЗРОБКА МОДУЛІВ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробляючи мобільний інтелектуальний застосунок для аналізу особистих витрат потрібно ретельно вибирати технологію, яку будуть використовуватися під час розробки.

Kotlin – з недавніх пір стала офіційною та головною мовою для розробки застосунків для платформи Android. Вона змістила доволі стару та попередню мову, яка використовувалася для розробки під Android – Java. Порівнюючи з Java, Kotlin швидше, лаконічне виглядає, має краще безпеку, завдяки чому кількість вилетів Android додатків зменшила більше ніж вдвічі, підтримує технології, які спрощують розробку [13].

Jetpack Compose — це сучасний декларативний фреймворк графічного інтерфейсу для Android. Compose спрощує написання та підтримку інтерфейсу користувача програми, який дозволяє відтворювати інтерфейс користувача програми без обов’язкової зміни подання зовнішнього інтерфейсу. Порівняно з минулим підходом – XML версткою, Compose надає інструменти для створення інтерфейсу в декілька разів швидше та легше [14].

ChatGPT – це штучний інтелект, розроблений OpenAI на основі архітектури GPT (Generative Pre-trained Transformer). Основна функція полягає в обробці та генерації тексту відповідно до введених запитів.

Переваги, які ChatGPT має порівняно з іншими моделями штучного інтелекту, включають [15]:

- Масштабність: побудований на величезному корпусі тексту та параметрах, що дозволяє розуміти широкий спектр тем і концепцій;
- універсальність: може виконувати різноманітні завдання, включаючи відповіді на питання, генерацію тексту, переклад та багато іншого;

- навчання на контексті: архітектура базується на трансформерах, які дозволяють аналізувати та узагальнювати контекст в тексті, що дозволяє робити більш точні та змістовні відповіді.

Бібліотека локального сховища Room [16] забезпечує рівень абстракції над SQLite, щоб забезпечити вільний доступ до бази даних, одночасно використовуючи всю потужність SQLite [17]. Зокрема, Room надає такі переваги:

- Перевірка SQL-запитів під час компіляції;
- зручні анотації, які зводять до мінімуму повторюваний і схильний до помилок шаблонний код;
- оптимізовані шляхи міграції бази даних.

Для авторизації користувача буде використовуватися Firebase. Саме він був обраний через безкоштовність та легку інтеграцію з розроблюваним додатком.

В якості середовища розробки буде використовуватися Android Studio, оскільки саме вона є рекомендованою Google та через відсутність інших аналогів.

Поєднання вищеописаних технологій дозволяє розробити мобільний застосунок, який забезпечить новий та приємний досвід у контролі особистих фінансів. Kotlin та Compose забезпечать якісну кодову базу з привабливим та інтуїтивно зрозумілим інтерфейсом; ChatGPT надасть інструменти для аналізу фінансів користувача; Firebase надасть змогу легко та надійно керувати користувачами; Room забезпечить надійне сховище для зберігання інформації користувача.

Отже, вибір інструментів для розробки мобільного застосунку з елементами штучного інтелекту для аналізу особистих фінансів є критично важливим аспектом для створення якісного продукту, для цього були обрані Kotlin, Compose, ChatGPT, Firebase, Room.

3.2 Розробка модуля для управління даними користувача

Важливою задачею мобільного застосунку є безпека та коректна робота з даними користувача. Для цього був створений модуль управліннями даними. Він відповідає за створення та редагування транзакцій користувача.

Алгоритм створення транзакції описаний на рисунку 3.1.

Розглянемо детальніше алгоритм створення транзакції:

1. Початок роботи.
2. Зчитування введених користувачем даних.
3. Відловлювання натиску кнопки збереження.
4. Відбувається перевірка даних на коректність, а саме правильний формат введеної суми, дати тощо.
5. У разі не успішного проходження перевірки програма показує помилку користувачу та повертається на початок.
6. У разі успішної перевірки наступним кроком є перетворення даних у відповідний тип для бази даних.
7. Перетворенні дані зберігаються в локальній базі даних.
8. Кінець роботи.

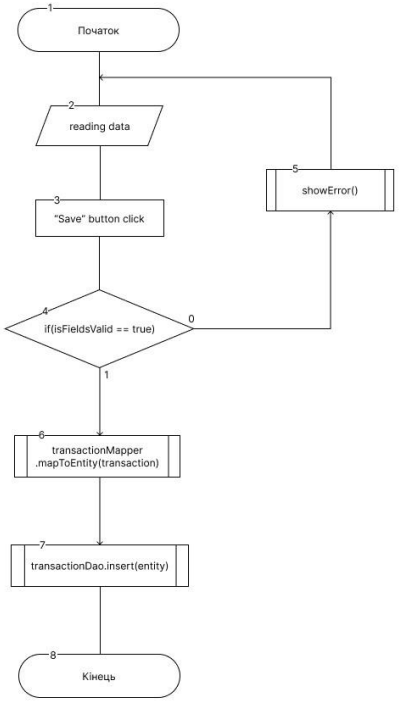

Рисунок 3.1 – Алгоритм створення транзакції

Наступний алгоритм – редагування транзакції, зображений на рисунку 3.2.

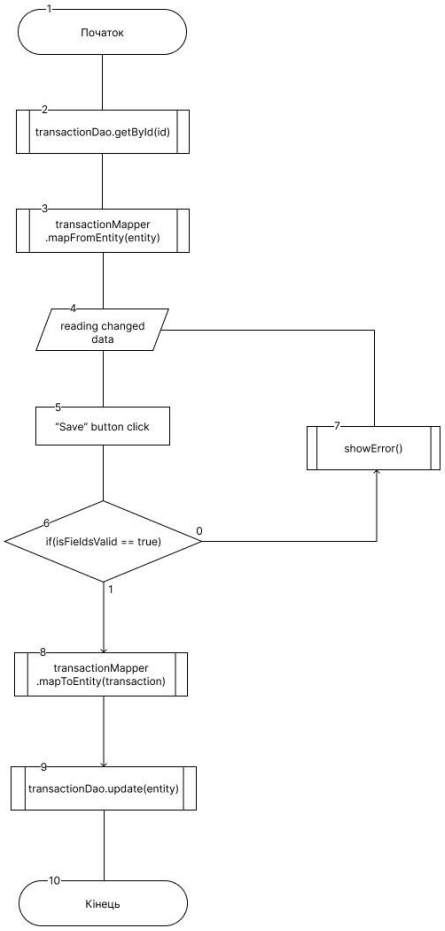

Рисунок 3.2 – Алгоритм редагування транзакції

Опис кожного етапу алгоритму редагування транзакції:

1. Початок роботи.
2. Отримання транзакції з бази даних.
3. Перетворення даних для відображення користувачу.
4. Редагування даних користувачем.
5. Натиск на кнопку «Збереження».
6. Відбувається перевірка введених полів на коректність, а саме правильний формат введеної суми, дати тощо.
7. У разі невдалого проходження перевірки програма показує помилку користувачу.

8. У разі успішної перевірки наступним кроком є перетворення даних у відповідний тип для бази даних.

9. Перетворенні дані зберігаються в локальній базі даних.

10. Кінець роботи.

3.3 Розробка модуля для аналізу та отримання даних від моделі штучного інтелекту

Основною задачею модуля є отримання порад від штучного інтелекту після аналізу даних. Для цього був створений алгоритм аналізу та отримання порад від моделі штучного інтелекту (рис. 3.3).

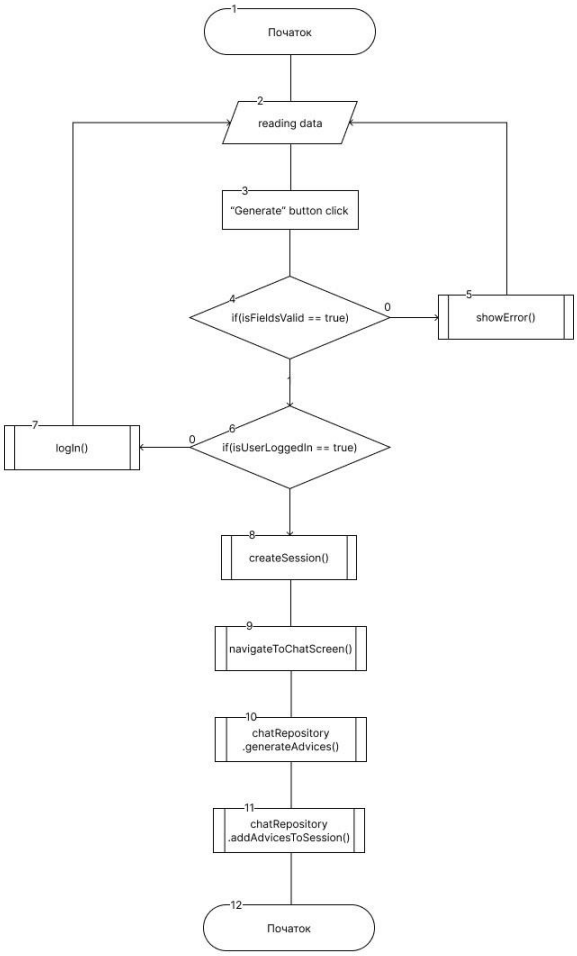

Рисунок 3.3 – Алгоритм аналізу та отримання порад від моделі штучного інтелекту

Опис етапів створеного алгоритму:

1. Початок роботи.
2. Введення та зчитування даних користувача.
3. Натиск кнопки «Згенерувати» користувачем.
4. Перевірка на коректність даних у введених полях.
5. При невдалій перевірці програма показує помилку та продовжує своє виконання з 2 етапу.
6. При вдалій перевірці даних програма перевіряє чи користувач авторизований.
7. При невдалій перевірці чи користувач авторизований програма перенаправляє його на екран авторизації, після якого користувач продовжить з 2 етапу.
8. Створення сесії на основі введених даних.
9. Перенаправлення на екран чату.
10. Генерація порад штучним інтелектом.
11. Додавання порад до існуючої сесії.
12. Кінець програми.

3.4 Розробка графічного модуля

Було вирішено розробити інтерфейс максимально простим та дружелюбним для користувача.

Основними кольорами були обрані чорний та білий, для уникнення конфлікту кольорів та легкого виділення важливих елементів.

При запуску додатку, користувач бачить екран завантаження з логотипом мобільного застосунку (рис 3.4). Під час відображення екрану завантаження з логотипом мобільного застосунку відбувається підготовка даних для початкового відображення та перевіряється чи користувач авторизований у системі.

Рисунок 3.4 – Екран завантаження

Мобільний застосунок передбачає авторизацію користувачів, для досягнення цієї цілі було спроектовано та розроблено екран з реєстрацією користувача у системи (рис. 3.5). Для створення акаунта, користувачу необхідно ввести ім’я, пошту та пароль.

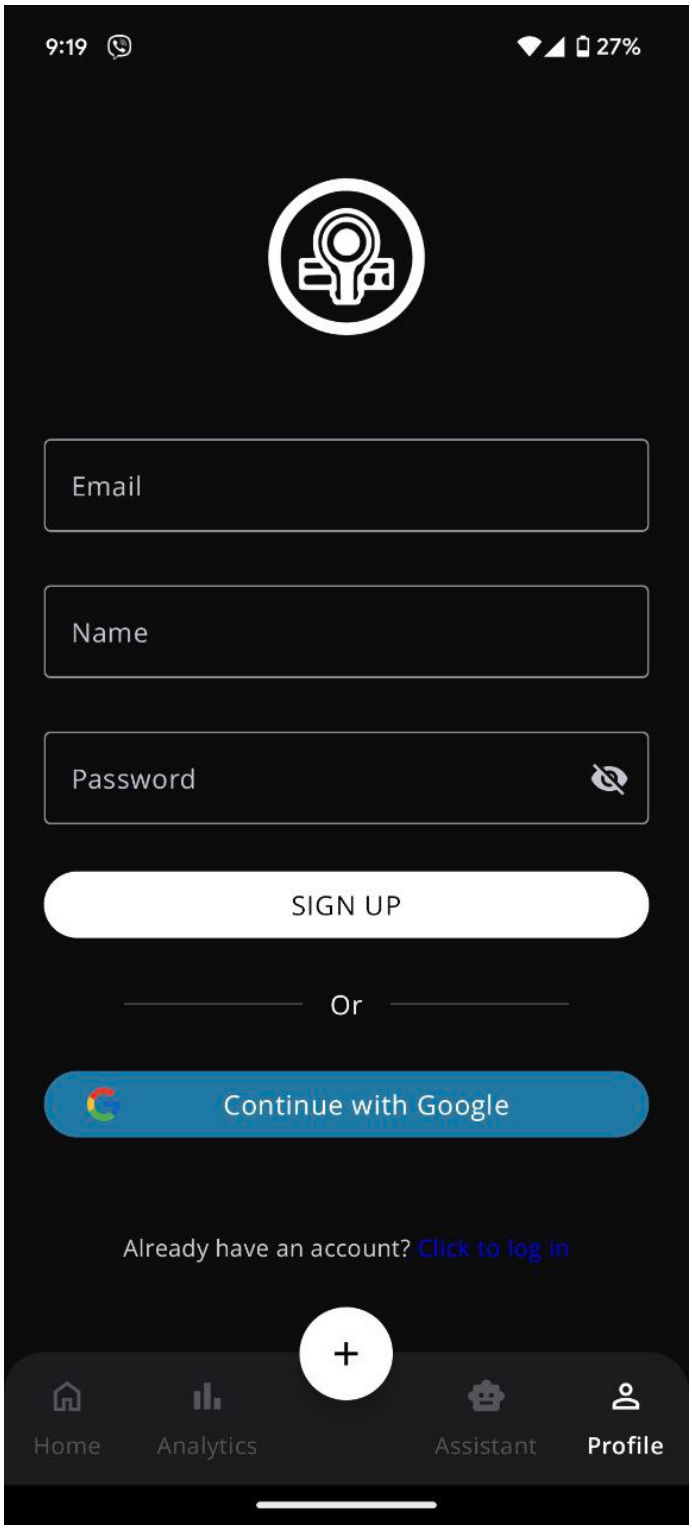

Рисунок 3.5 – Екран реєстрації

Після успішної реєстрації користувач потрапить на екран авторизації до системи (рис 3.6). Для входу у систему, необхідно ввести пошту та пароль або використати Google-акаунт.

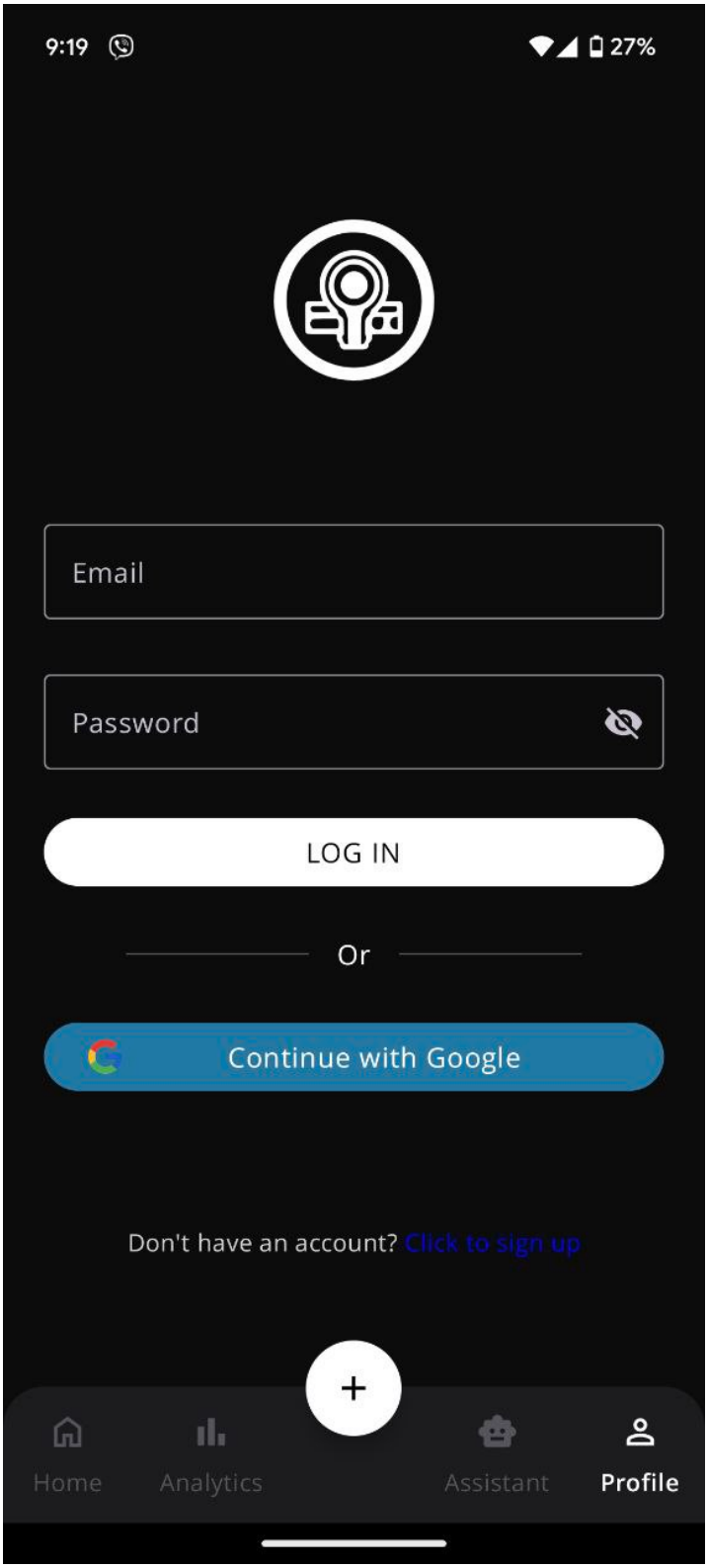

Рисунок 3.6 – Екран авторизації

На головному екрані (рис. 3.7), користувач може побачити поточний баланс, витрати та доходи за поточний місяць та недавні транзакції.

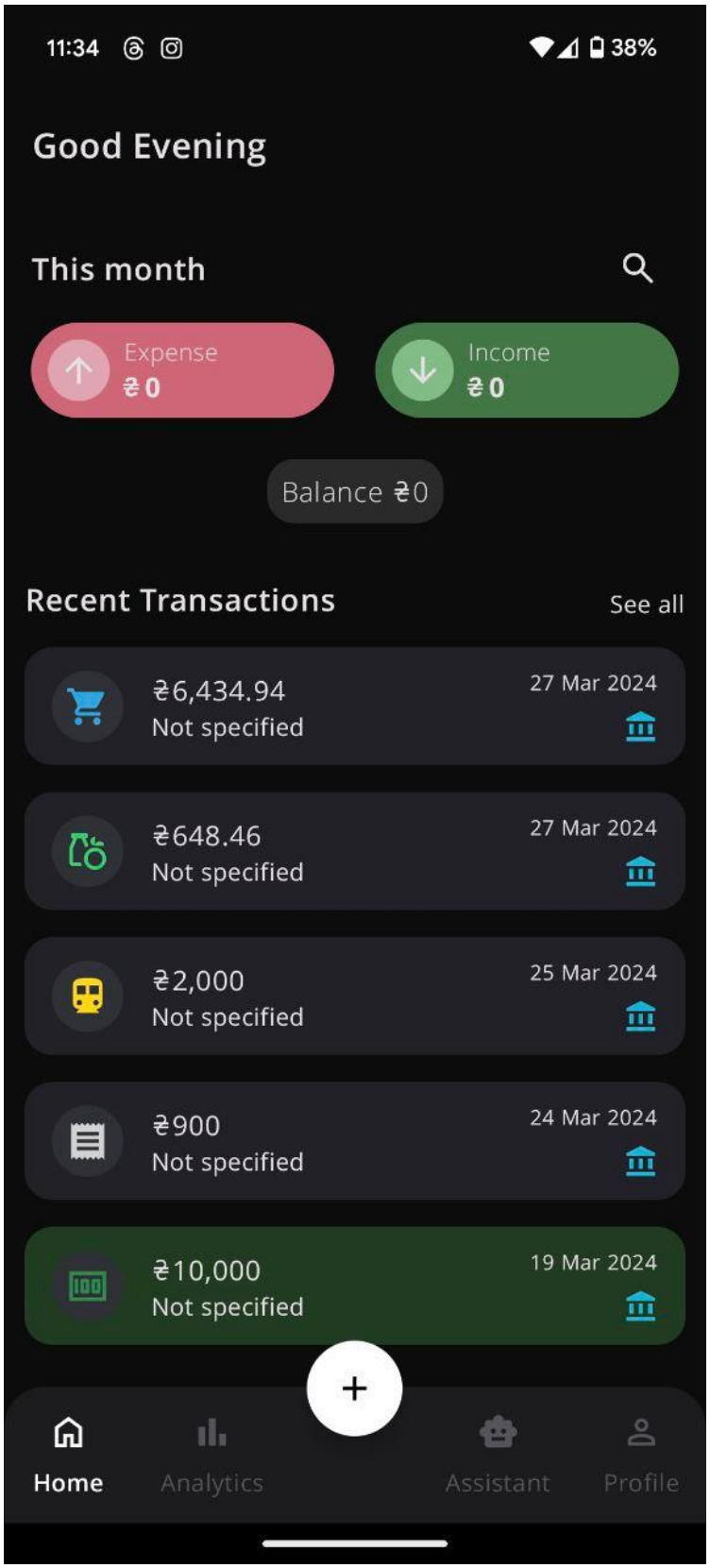

Рисунок 3.7 – Головний екран

Натиснувши на іконку лупи, відкриється екран пошуку транзакцій за назвою та фільтрами (рис. 3.8).

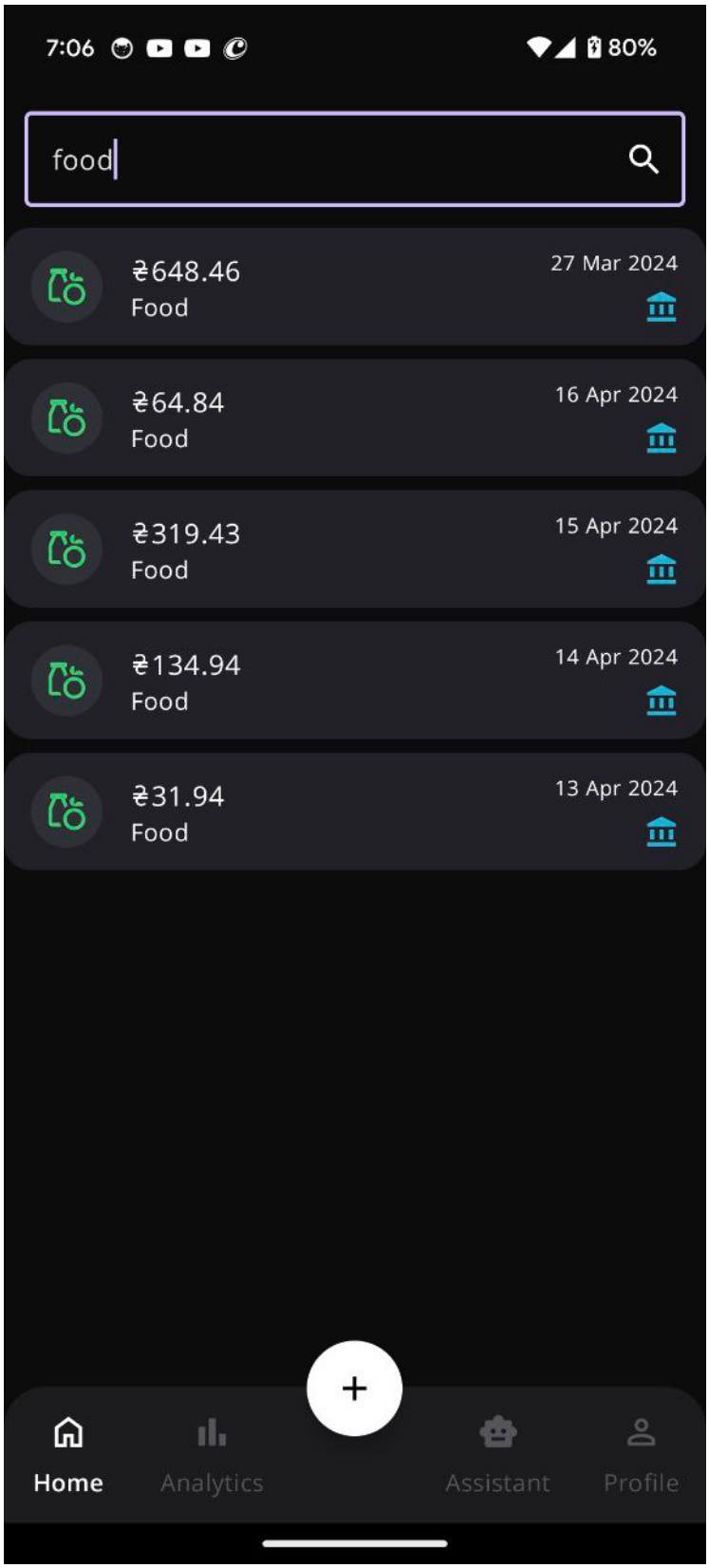

Рисунок 3.8 – Екран пошуку

Також наявний екран для перегляду усіх існуючих транзакцій користувача (рис. 3.9), на нього можна потрапити натиснувши на текст “See all”.

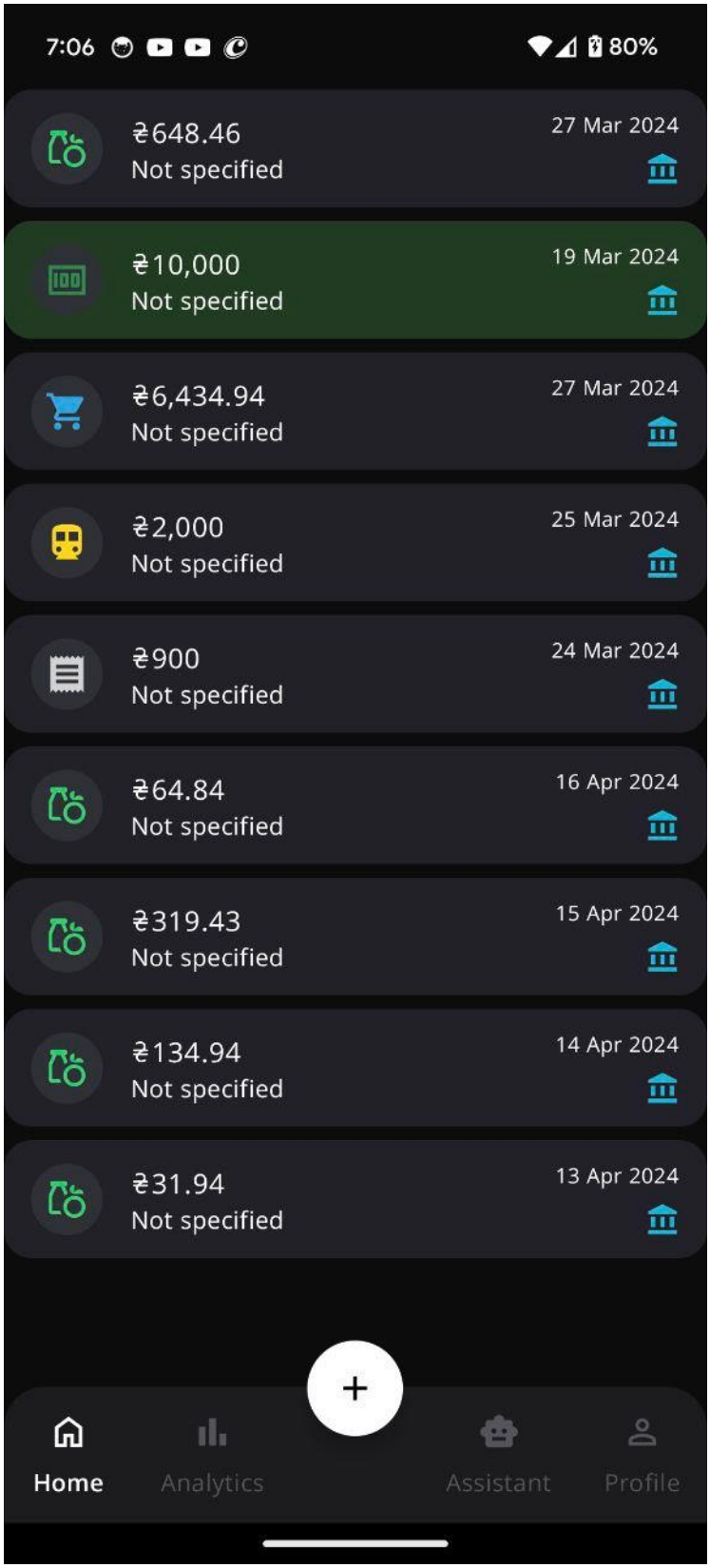

Рисунок 3.9 – Екран з усіма транзакціями

У другій вкладці у навігаційній панелі розташований екран з аналітикою витрат та доходів. Відкривши його, користувач отримає змогу вибирати бажаний

інтервал для аналітики витрат та доходів. Натиснувши на стрілочку біля діаграми, користувач потрапить на екран з аналізом фінансів за обраний проміжок часу штучним інтелектом.

Для отримання аналітики фінансів за поточний тиждень – користувач повинен натиснути кнопку “Week” та вибрати бажаний інтервал (рис. 3.10).

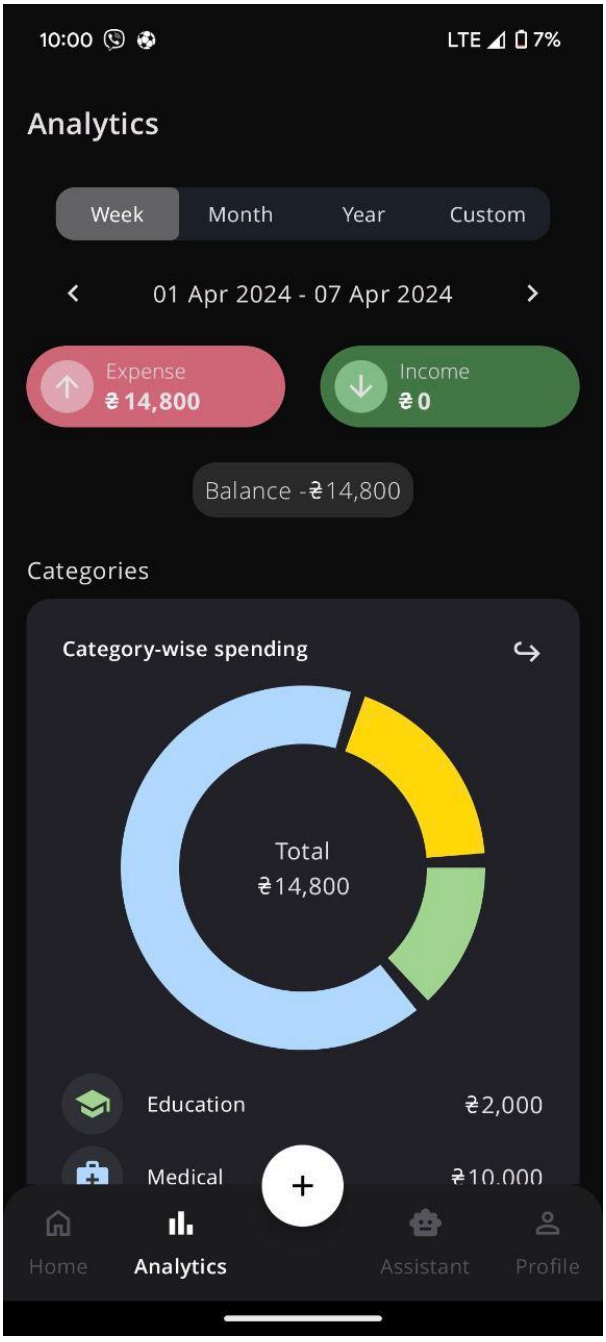

Рисунок 3.10 – Екран з аналітикою фінансів за тиждень

На натиск опції “Month” користувач отримає аналітику фінансів за місяць (рис. 3.11).

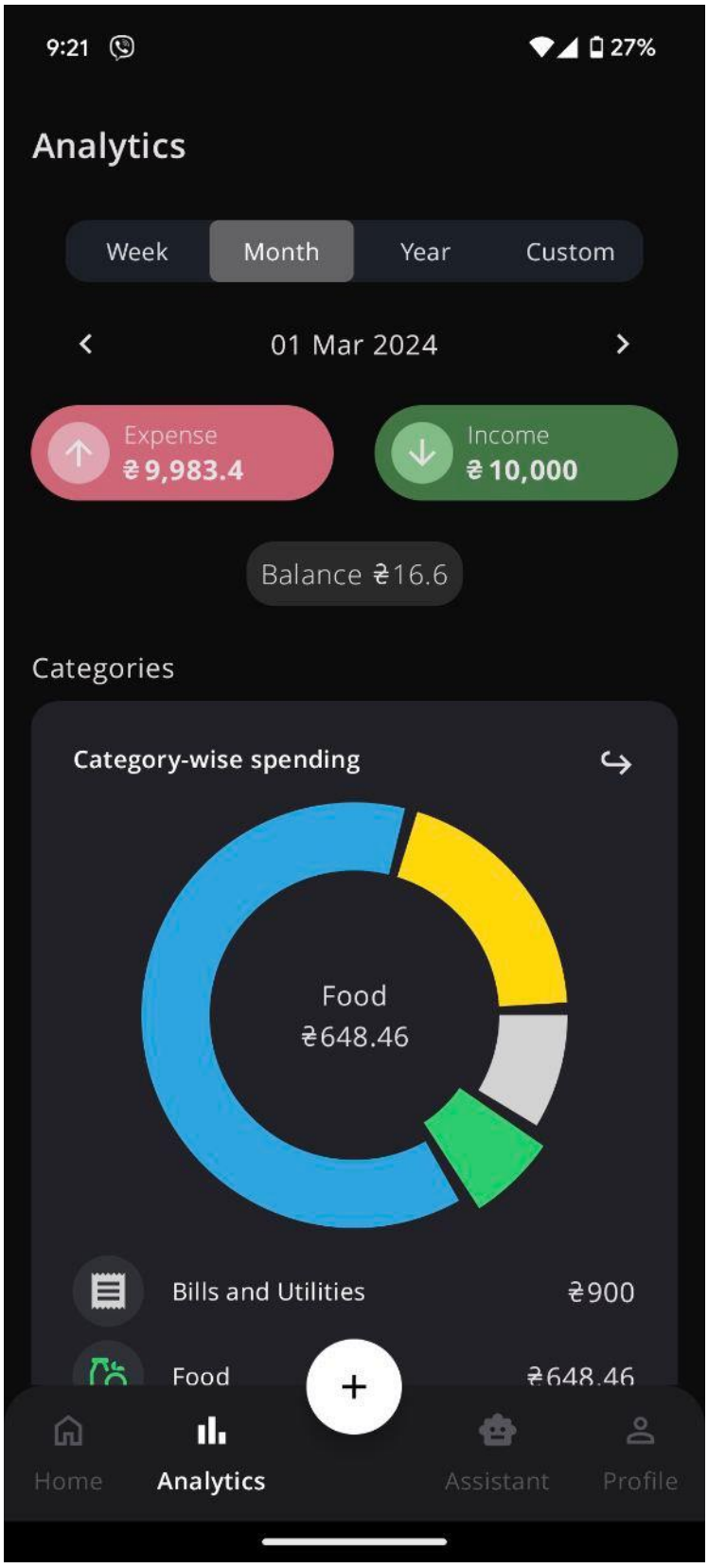

Рисунок 3.11 – Екран з аналітикою фінансів за місяць

При бажанні отримати аналітику за весь рік, користувач повинен обрати опцію “Year” та обрати бажаний рік (рис. 3.12).

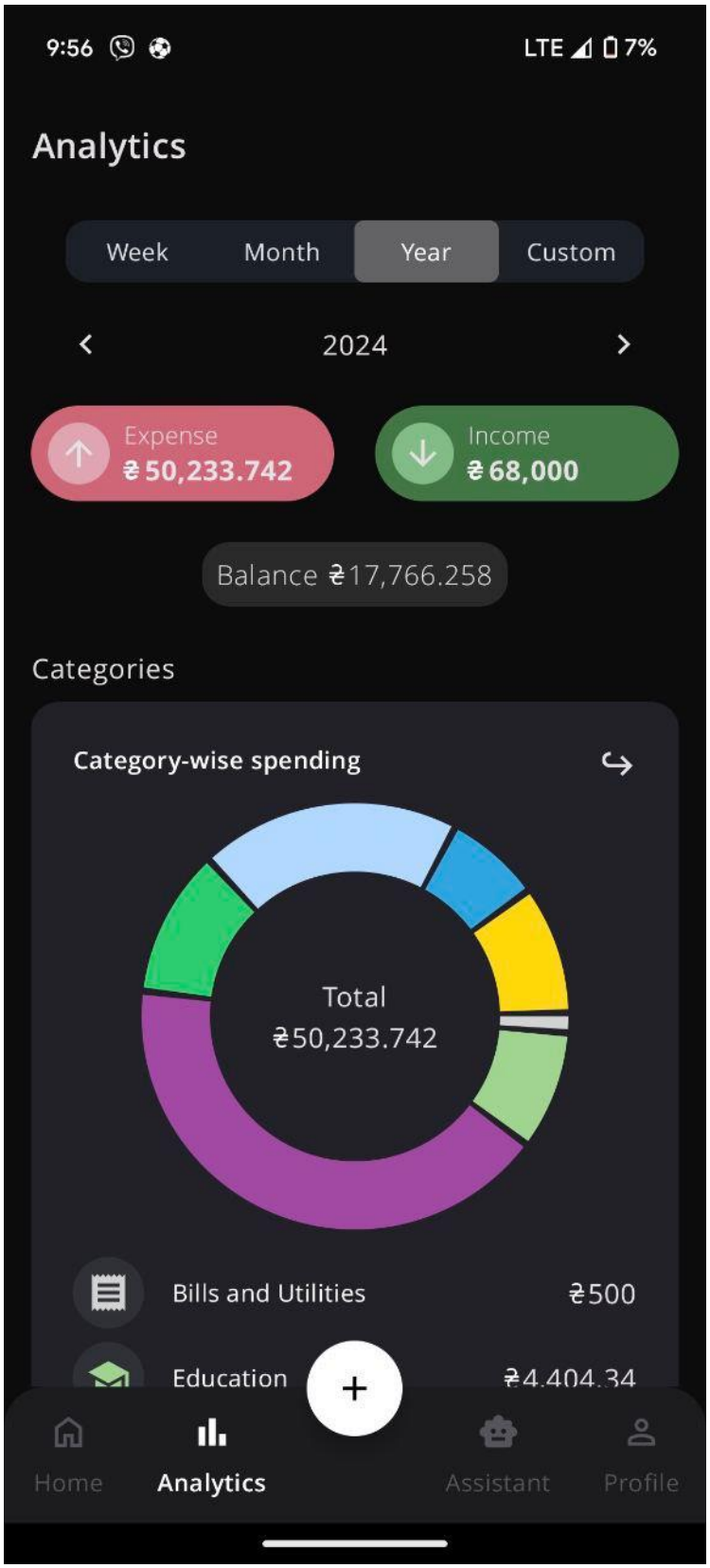

Рисунок 3.12 – Екран з аналітикою фінансів за рік

У третій вкладці розташований екран з вибором даних для аналізу та подальшого генерування порад штучним інтелектом (рис. 3.13). Наявні поля для введення назви поради, вибору проміжку часу для аналізу транзакцій користувача для аналізу, типу фінансової цілі (короткочасна, довгочасна або просто оптимізація бюджету), фінансова ціль у гривнях, бажана дата досягнення цілі, нотатка яка буде враховуватися при аналізі.

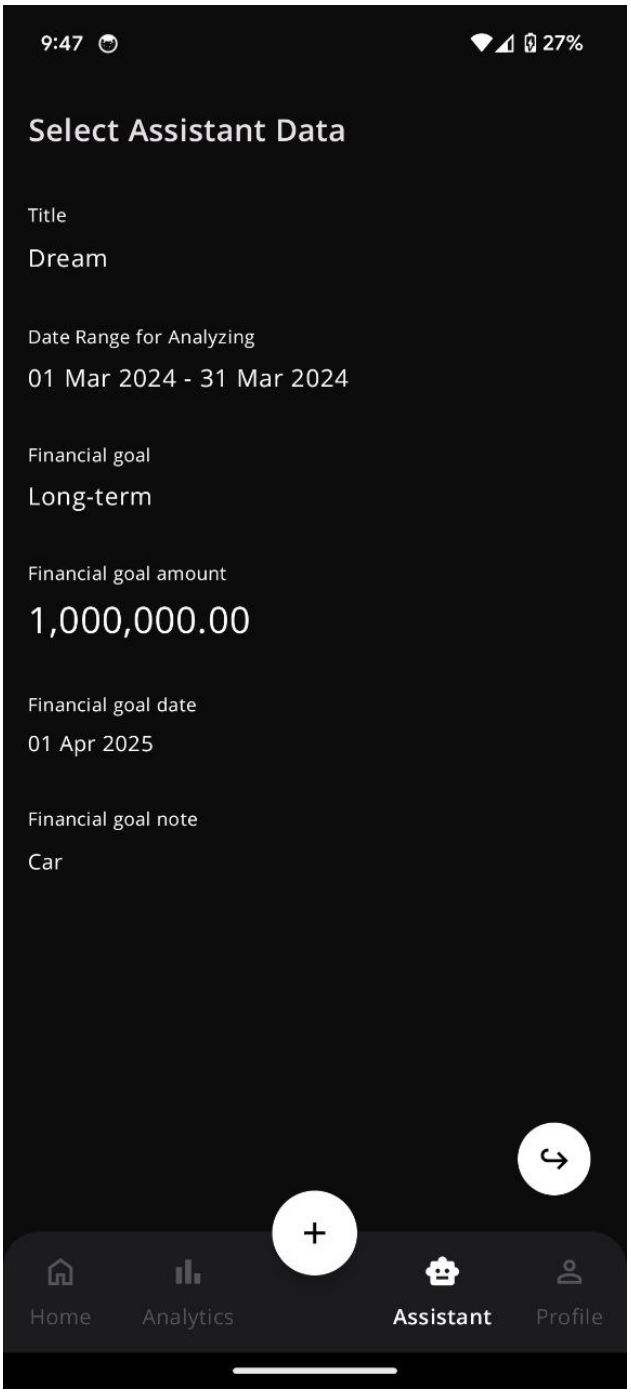

Рисунок 3.13 – Екран для введення даних для аналізу

Коли обрана оптимізація (рис. 3.14) бюджету як ціль, зникають деякі поля для введення: фінансова ціль у гривнях, дата досягнення цілі, також ці дані не використовуються для подальшого аналізу.

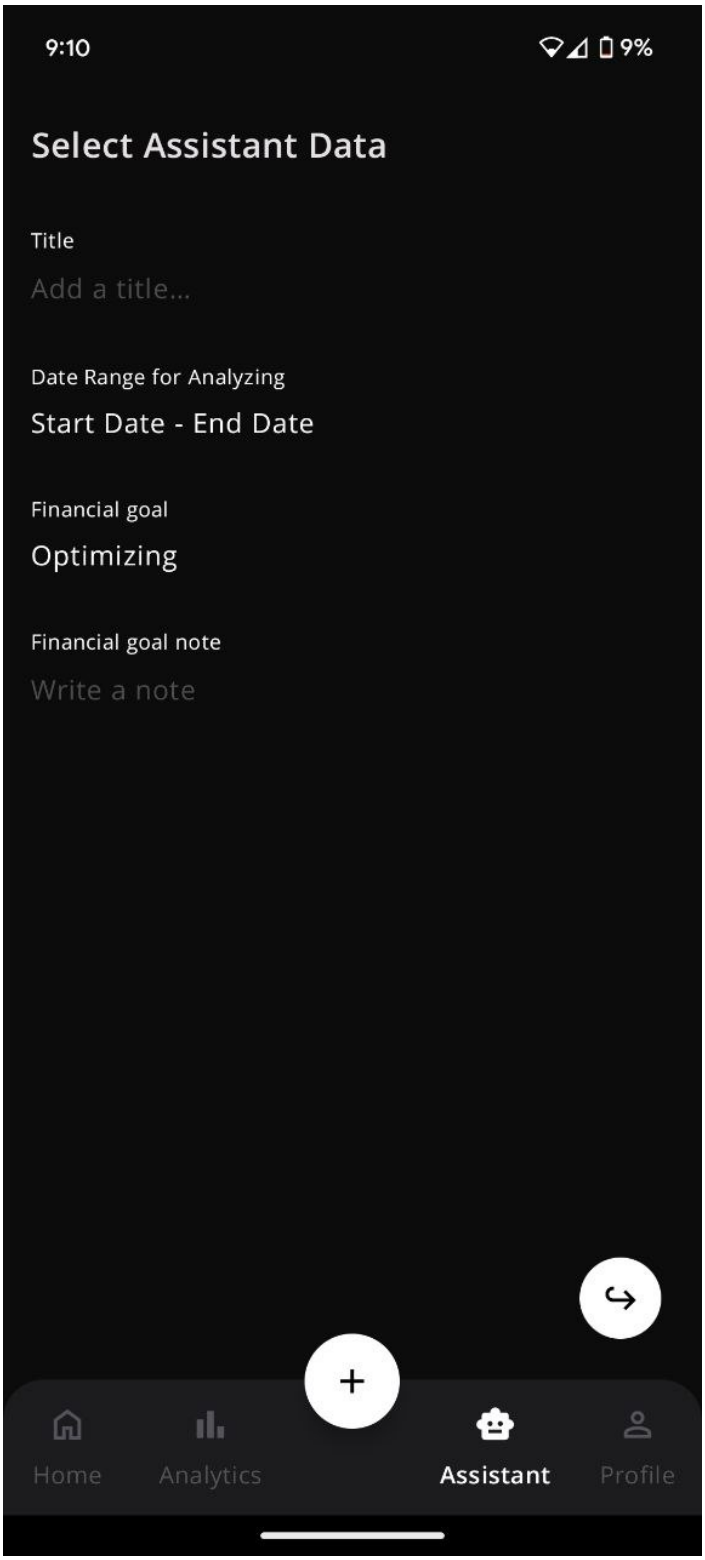

Рисунок 3.14 – Ціль «Оптимізація» на екрані введення даних для аналізу

Після вибору всіх даних, натиснувши на білу іконку зі стрілочкою, користувач потрапить на екран з згенерованими штучним інтелектом порадами (рис. 3.15 – рис. 3.16).

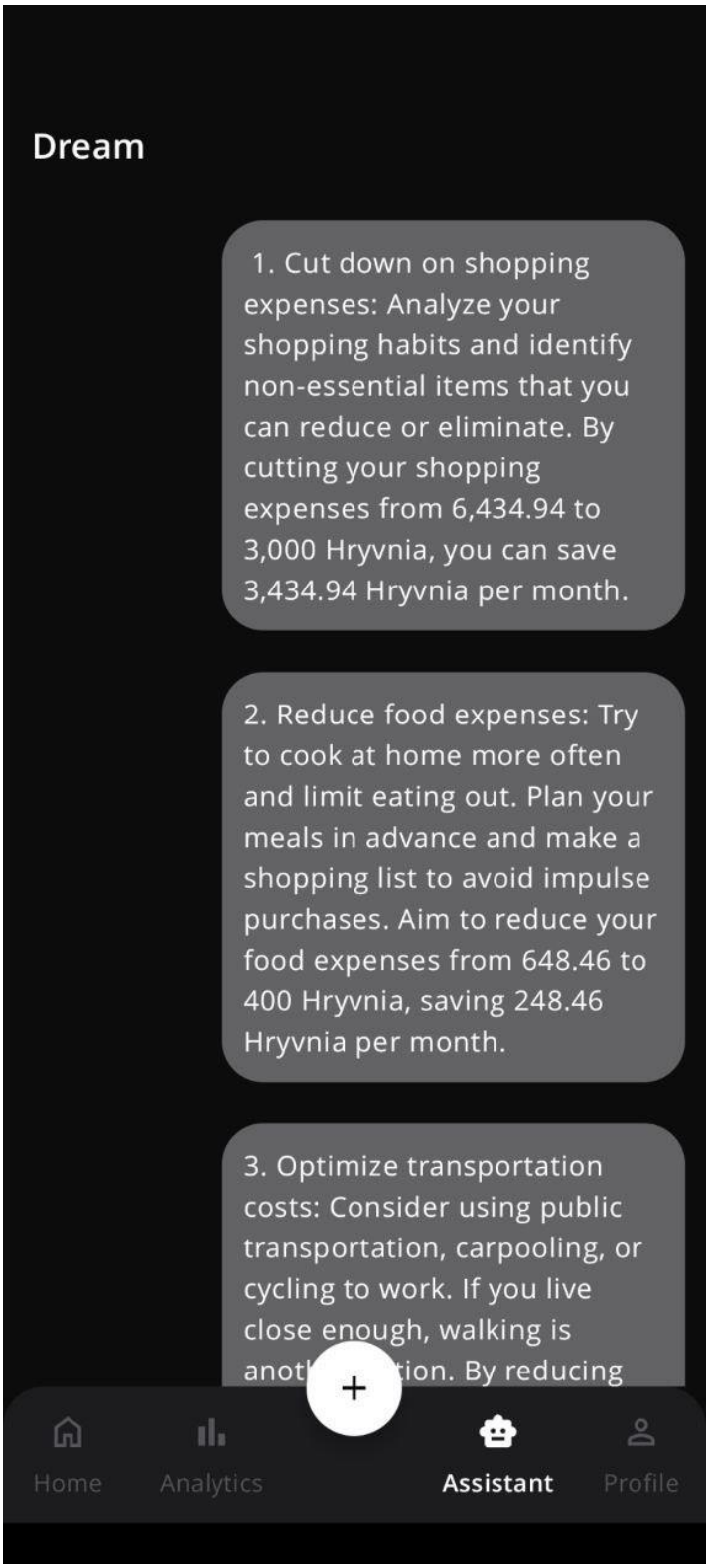

Рисунок 3.15 – Екран зі згенерованими порадами

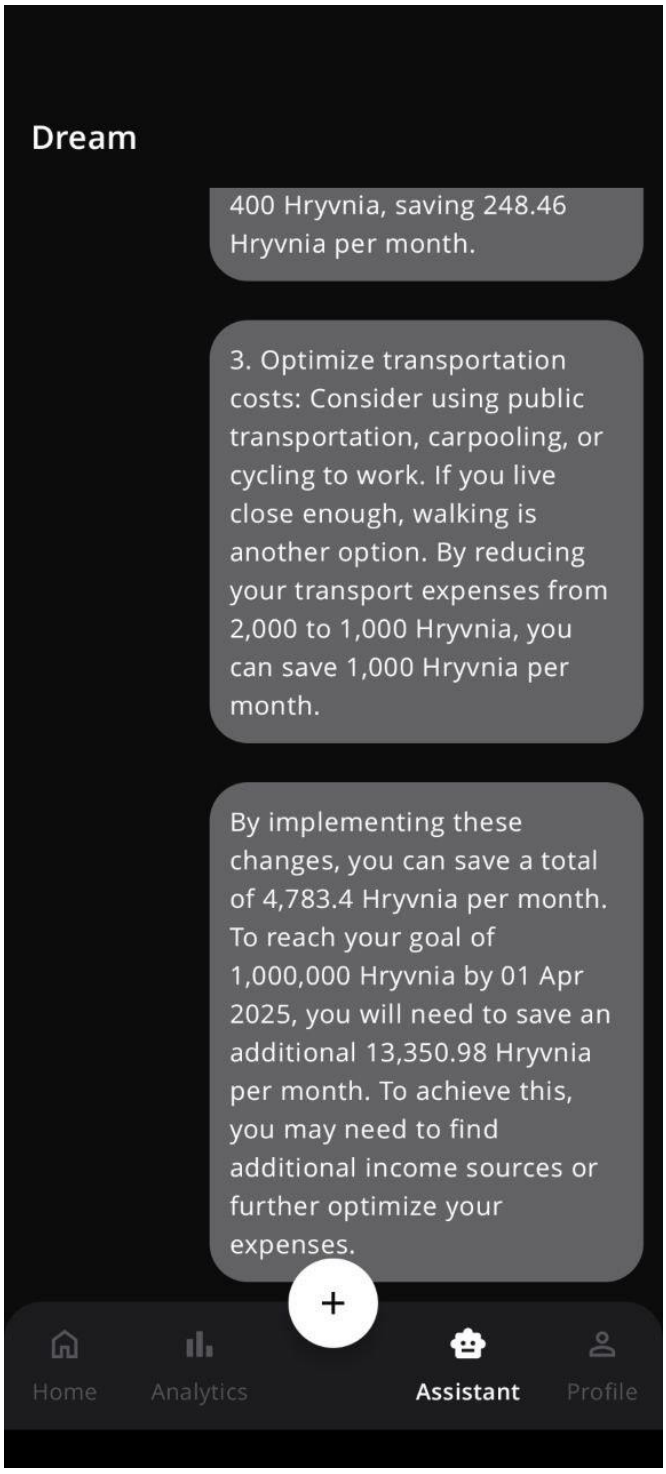

Рисунок 3.16 – Екран зі згенерованими порадами

За замовчуванням, модель штучного інтелекту згенерує 3 поради. На цьому екрані користувач може знайти не тільки згенеровані поради, а й загальну назву, та дані, які він обрав на минулому екрані.

Для додавання транзакцій користувач може натиснути на білу іконку з плюсом, яка доступна з будь-якого екрану. У вікні наявні дві опції: додавання

витрат (рис 3.17) та доходів (рис. 3.18). У обох режимах користувач може обрати наступні дані: дату, суму, категорію, тип оплати та коментар.

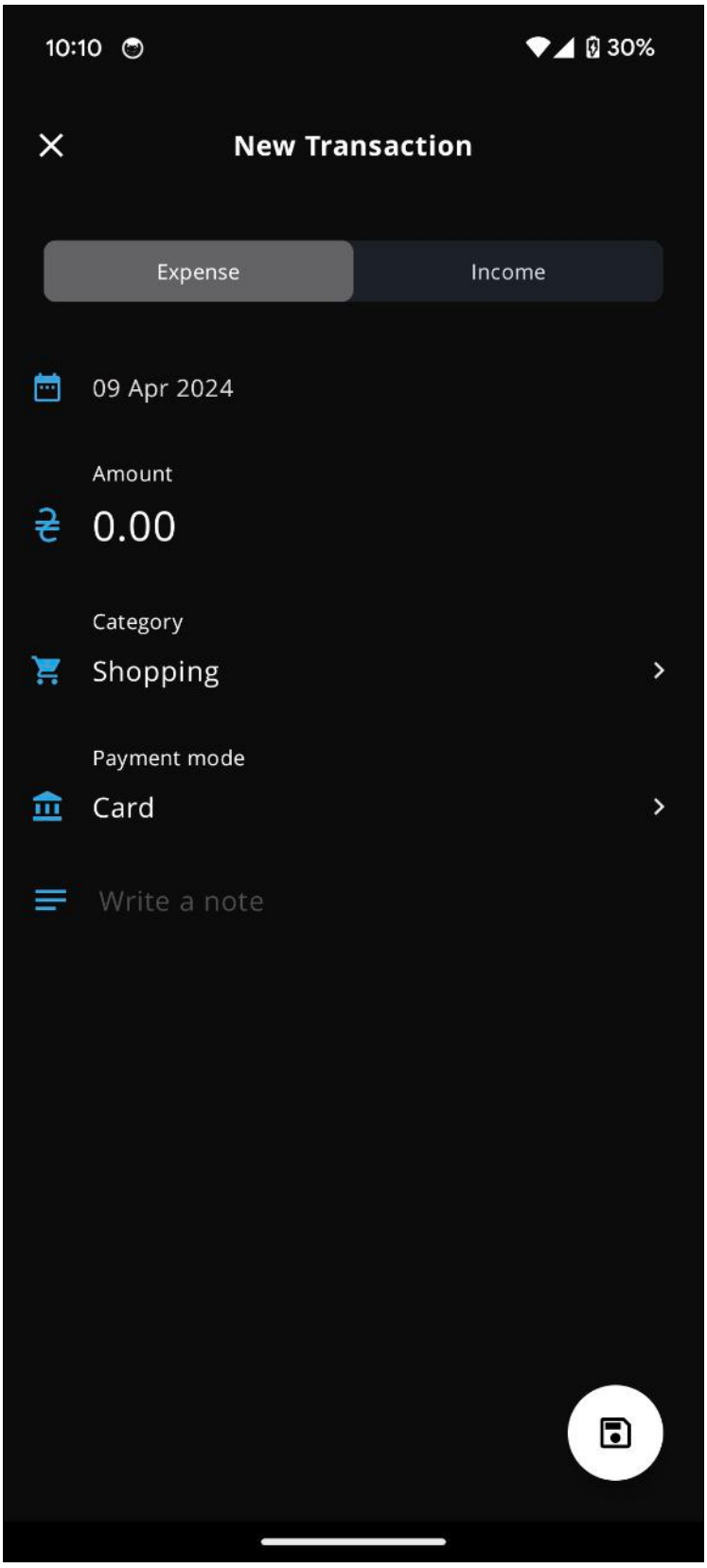

Рисунок 3.17 – Екран додавання витрат

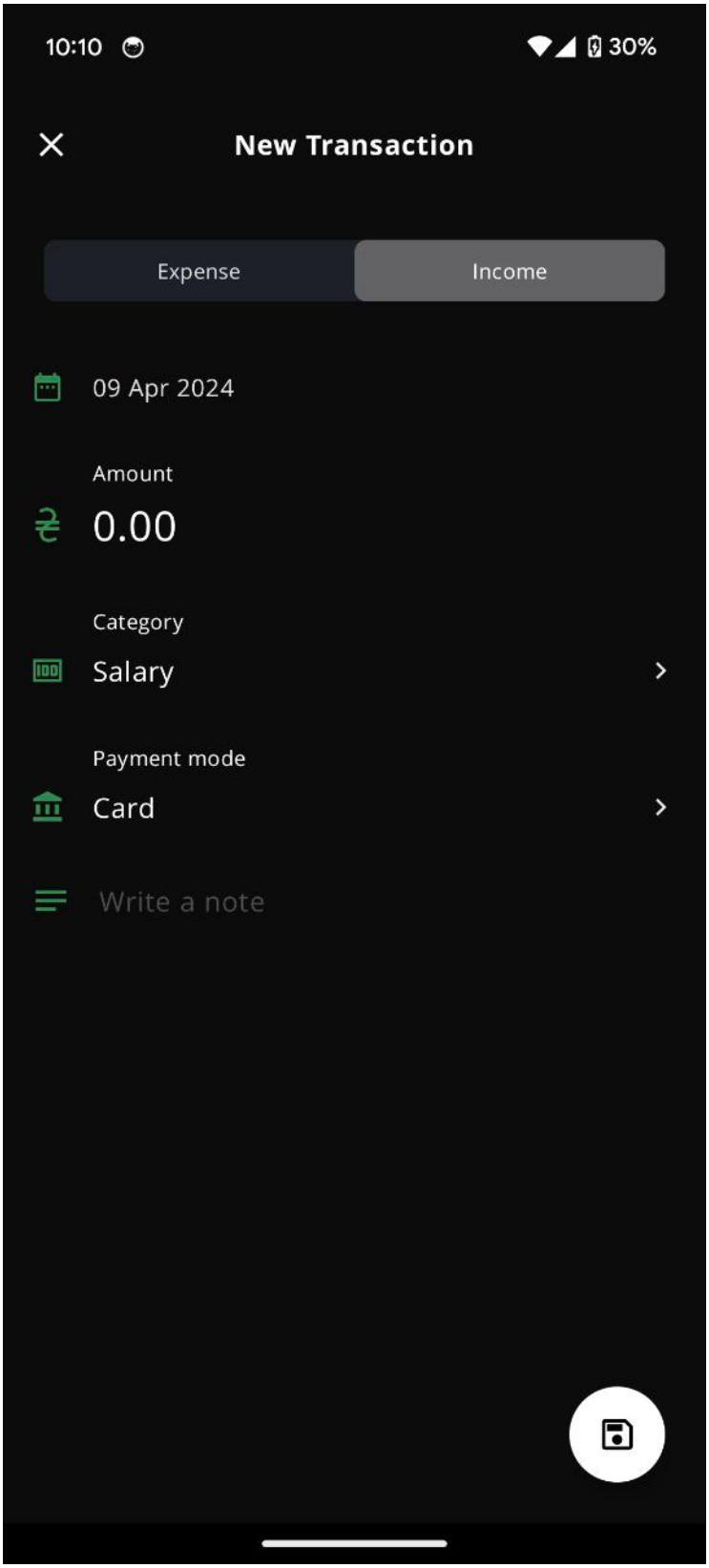

Рисунок 3.18 – Екран додавання доходу

Розробка графічного інтерфейсу мобільного застосунку була проведена у середовищі Android Studio з використанням технології Compose [18].

3.5 Висновки

У третьому розділі було продемонстровано вибір мови програмування та технології для розробки системи. Мова програмування Kotlin була обрана в результаті аналізу потреб програмного продукту. Для зручності розробки графічного інтерфейсу було обрано Compose, який дозволяє писати графічні інтерфейси за допомогою мови Kotlin. В якості моделі штучного інтелекту було обрано ChatGPT. Для локального зберігання файлів було обрано бібліотеку локального зберігання даних Room.

Було описано розробку наступних модулів:

- управління даними користувача;
- аналізу та отримання даних від моделі штучного інтелекту;
- графічного.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування мобільного застосунку на Android [19] є надзвичайно важливим етапом під час розробки. Оскільки, саме тестування є вирішальним показником якості, відповідності очікуванням користувачів та безперебійної роботи на різних пристроях програмного забезпечення.

Для початку тестування слід встановити тестову версію мобільного додатку на смартфон з операційною системою Android. Безпосередньо тестування починається з функціональних тестів для перевірки правильності виконання програми, а саме введення ти виведення даних, на непередбачені дані та бізнес-логіку. Результат тестування зображено на рисунку 4.1.

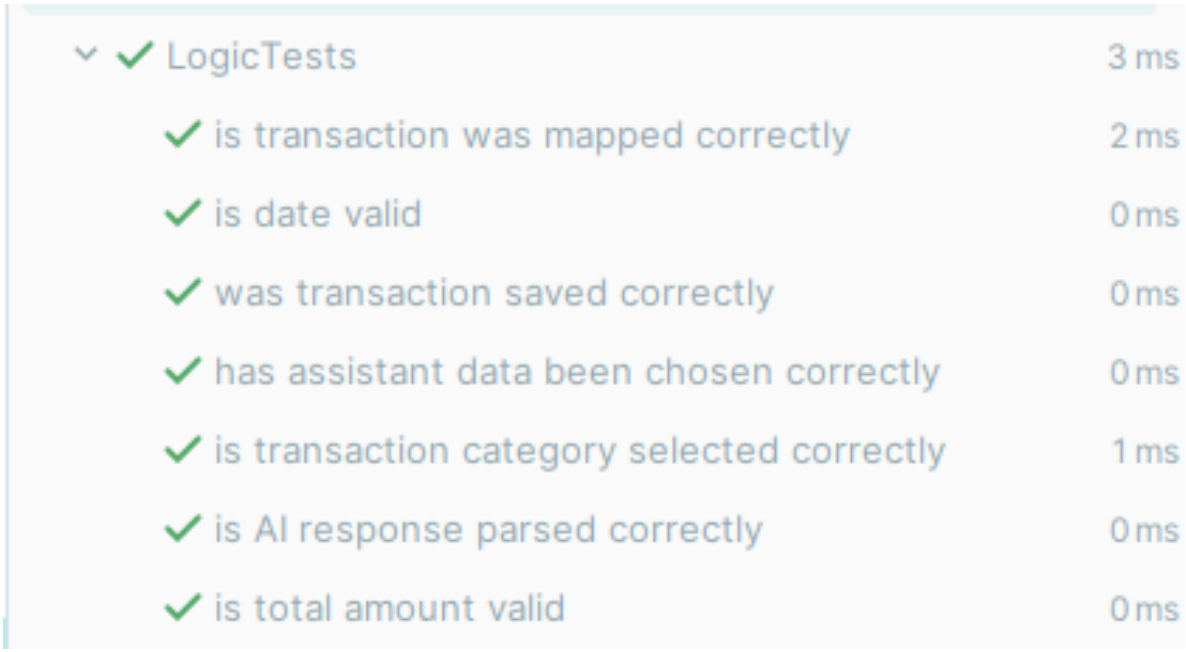

Рисунок 4.1 – Результат виконання функціональних тестів

Наступним кроком є тестування безпеки програми, зокрема перевірки на цілісність даних під час створення, оновлення, видалення, а також міграцій бази даних. Результат тестування зображено на рисунку 4.2.

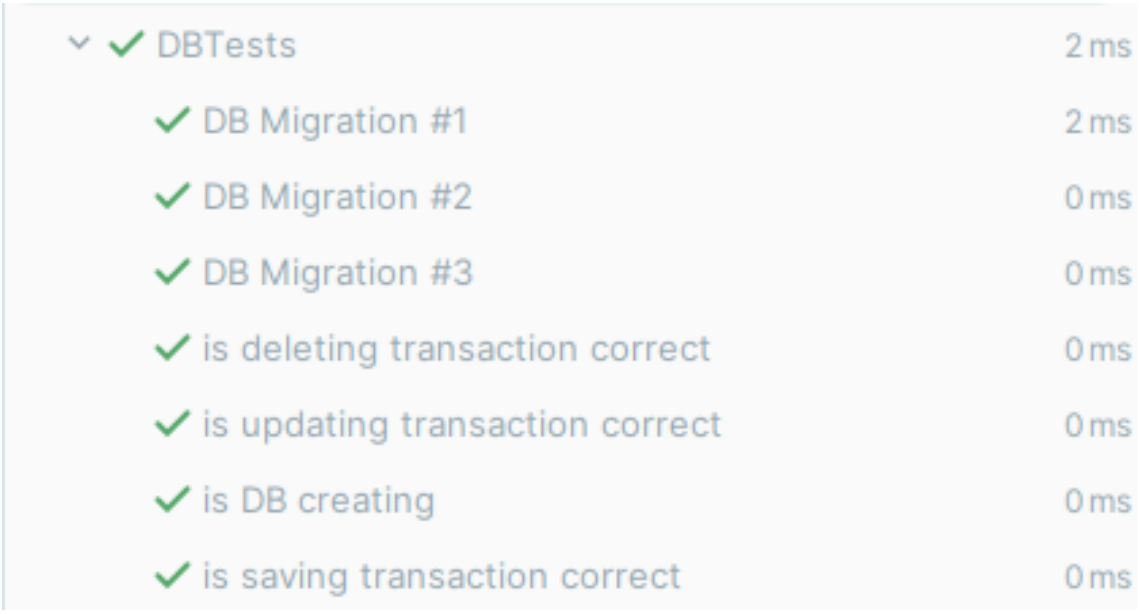

Рисунок 4.2 – Результат тестування безпеки програми

Далі тестуються графічні елементи, для перевірки елементів інтерфейсу на корекетне відображення, чи мають відповідні розміри та розташування, чи реагують на дії користувача належним чином. Також UI-тести [20] допомагають виявити проблеми з зручністю користування: незручне розташування кнопок, не чіткі інструкції. Результат виконання зображено на рисунку 4.3.

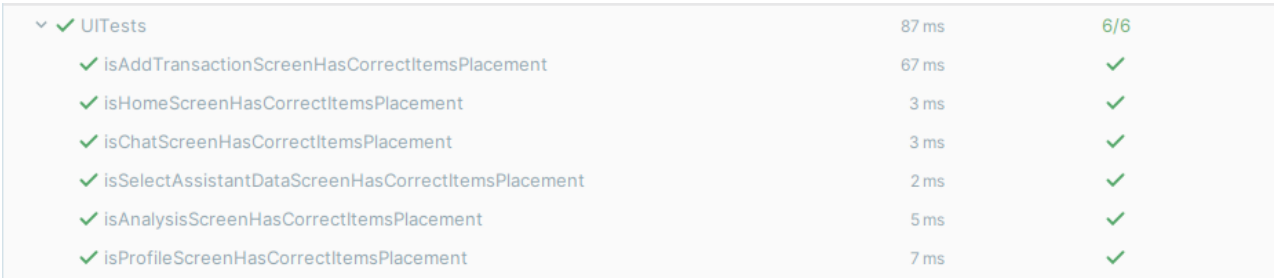

Рисунок 4.3 – Результат графічного тестування

Після успішного проходження всім тестів, проводиться загальна оцінка продуктивності застосунку за допомогою вбудованого інструмента у Android Studio під назвою Profiler [21]. Результат зображений на рисунку 4.4.

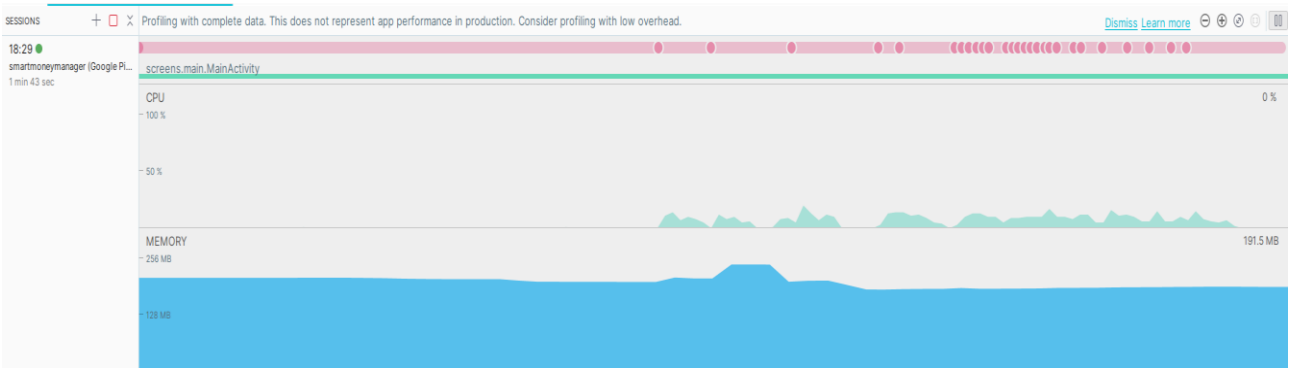

Рисунок 4.4 – Результат тестування продуктивності системи

Наступним кроком буде перевірка екрану вибору даних для аналізу штучним інтелектом. У разі спроби генерації порад без введення назви, додаток покаже помилку і запропонує ввести назву, оскільки це є необхідною умовою для подальшої роботи додатку (рис. 4.5).

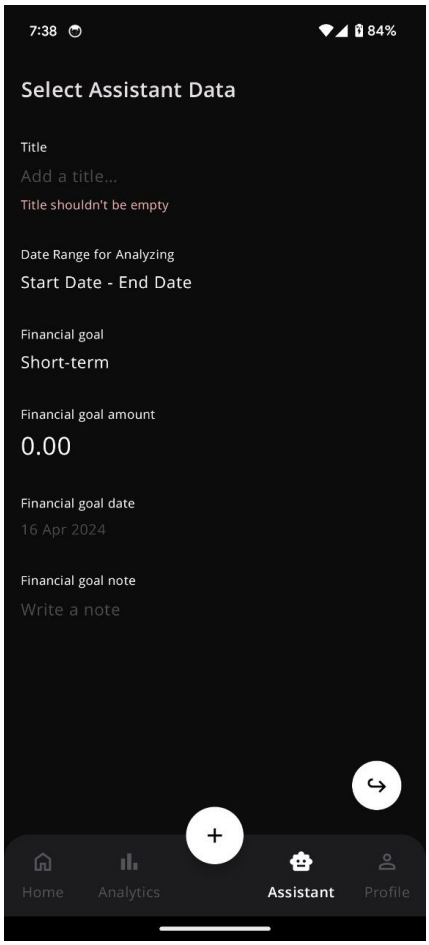

Рисунок 4.5 – Створення порад без назви

Також подібна валідація є для поля вибору проміжку часу (рис. 4.6), сумі фінансової цілі (рис. 4.7) та бажаної дати (рис. 4.8).

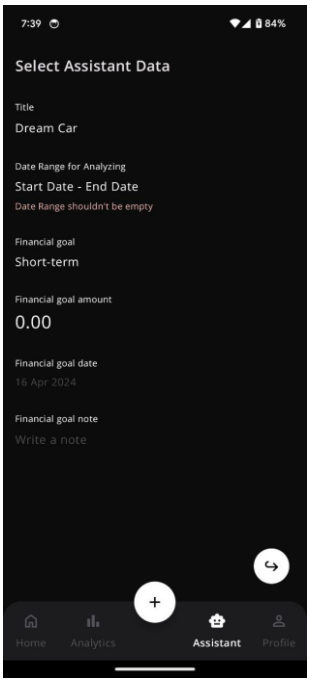

Рисунок 4.6 – Створення порад проміжку часу

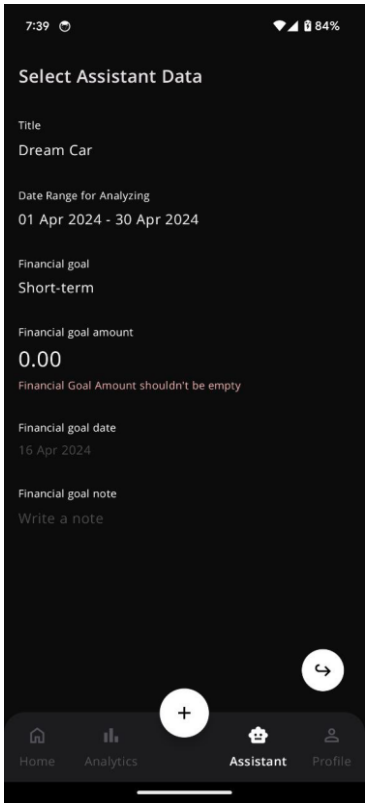

Рисунок 4.7 – Створення порад без суми фінансової цілі

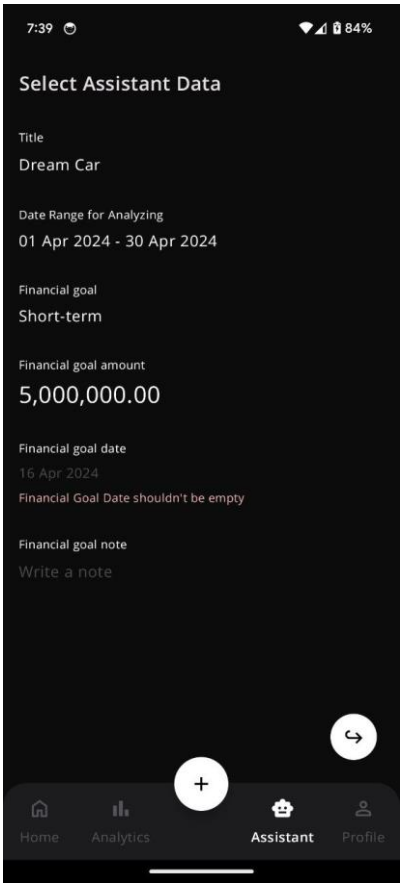

Рисунок 4.8 – Створення порад без бажаної дати

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску мобільного додатку. Деталі щодо мінімальної та рекомендованої конфігурації смартфона наведені в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація

Процесор	Snapdragon 660 або еквівалентний для Android
Оперативна пам’ять	3 GB
Пропускна здатність мережи	4G з'єднання
Вільне місце	770 мб
Версія ОС	8.0

Таблиця 4.2 – Рекомендована конфігурація

Процесор	Snapdragon 845 або еквівалентний для Android
Оперативна пам’ять	4 GB або більше
Пропускна здатність мережи	5G з'єднання
Вільне місце	770 мб
Версія ОС	13

Після запуску мобільного додатку користувач повинен додати певну кількість транзакцій, на основі яких будуть згенеровані поради. Після створення порад, користувач отримує змогу переглядати детальну статистику, графіки про його фінанси. Для створення порад штучним інтелектом, користувачу потрібно ввести необхідні дані, які необхідні для аналізу та увійти у свій акаунт. Без авторизація у нього не буде змоги взаємодіяти зі штучним інтелектом.

4.3 Висновки

У четвертому розділі було проведено тестування програмного забезпечення мобільного застосунку з елементами штучного інтелекту для аналізу особистих фінансів. Було перевірено валідацію полів введення інформації, правильність зберігання інформації, взаємодії зі штучним інтелектом. Також було розроблено інструкцію користувача по встановленню та початковій експлуатації програмного забезпечення.

ВИСНОВКИ

У бакалаврській дипломній роботі розроблено мобільний інтелектуальний застосунок для аналізу особистих витрат. Для розробки було використано середовище програмування Android Studio. Бакалаврська дипломна робота розроблена згідно методичних вказівок [22].

Під час виконання бакалаврської дипломної роботи було проаналізовано сьогоднішній стан проблеми. Були розглянуті існуючі аналоги мобільного додатку. Також було визначено їх недоліки та порівняно з розробленим програним забезпеченням. Базуючись на порівнянні були створенні основні задачі бакалаврської дипломної роботи.

У бакалаврській дипломній роботі зроблено наступне:

- визначено найкращу модель штучного інтелекту для аналізу та отримання даних;
- розроблено модуль управління даними користувача;
- розроблено модуль для аналізу та отримання даних від моделі штучного інтелекту;
- розроблено графічний модуль;
- проведено тестування мобільного інтелектуального застосунку згідно поставлених задач.

Було розроблено загальний алгоритм роботи мобільного застосунку, модуля управління даних користувача та модуля для аналізу та отримання даних від моделі штучного інтелекту. Також було розроблено модель системи з використанням UML діаграм.

Тестування мобільного інтелектуального застосунку підтвердило повну функціональність мобільного застосунку та його придатність для виконання технічних завдань. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Artificial Intelligence (AI) and ChatGPT: detailed history and timelines [Електронний ресурс] – Режим доступу до ресурсу: https://www.officetimeline.com/blog/artificial-intelligence-ai-and-chatgpt-history-and-timelines

2. The Implications of AI in the Mobile App Industry [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@sandeepsrivastav_38514/the-ai-revolution-in-mobile-app-development-trends-and-predictions-for-2024-3e3e26f8e788

3. Педосенко А.Ю. Розробка програмного рішення для аналізу та оптимізації особистих фінансів на основі моделей штучного інтелекту / А.Ю. Педосенко, Катєльніков Д.І. // Матеріали VII Міжнародної науково-практичної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/fiip/fiip2024/paper/view/20163

4. How AI is Supercharging Money Management [Електронний ресурс] – Режим доступу до ресурсу: https://intive.com/insights/how-ai-is supercharging-money-management

5. wally [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.kinpos.wallytech&hl=en&gl=US

6. Emma [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.emmaprod&hl=en&gl=US

7. AI Money Manager [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.expense.tracker.gpt

8. Expense Tracker [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.AI.expense.spending.tracker&hl=en_US

9. The pros and cons of using a framework [Електронний ресурс] – Режим доступу до ресурсу: https://devm.io/programming/pros-cons-using-framework-135901-001

10. Domain layer in Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/architecture/domain-layer

11. Mapping of Data Objects? [Електронний ресурс] – Режим доступу до ресурсу: https://www.baeldung.com/kotlin/data-objects

12. Все про UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html

13. Atomic Kotlin навч. посіб. / Bruce Eckel. 2021 – 636 ст.

14. Android UI Development with Jetpack Compose: Bring declarative and native UIs to life quickly and easily on Android using Jetpack Compose навч. посіб. / Thomas Künneth. 2022 – 248 ст.

15. Advantages of Chat GPT [Електронний ресурс] – Режим доступу до ресурсу: https://www.appmatics.com/en/blog/vorteile-nachteile-chat-gpt

16. Save data in a local database using Room [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room

17. About SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/about.html

18. Jetpack Compose UI App Development Toolkit [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/develop/ui/compose

19. Fundamentals of testing Android apps [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/testing/fundamentals

20. Automate UI tests [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/testing/instrumented-tests/ui-tests

21. Profile your app performance [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio/profile

22. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / Уклад. Романюк О. Н., Черноволик Г. О. – Вінниця: ВНТУ, 2022. – 52 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. Романюк О.Н
12 березня 2024р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка мобільного інтелектуального застосунку для аналізу особистих витрат з використанням технологій Kotlin та Compose» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц., Д.І. Катєльніков
«12» березня 2024 р.
Виконав:
_____ студент гр. 1ПІ-20б, А.Ю. Педосенко
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільного інтелектуального застосунку для аналізу особистих витрат з використанням технологій Kotlin та Compose».

Галузь застосування – управління фінансами.

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 якості аналізу фінансів користувача та покращення фінансового стану в цілому за рахунок штучного інтелекту, що дозволяє та сприяє глибшому аналізу та більш якісним порадам.

Призначення роботи – розробка мобільного застосунку для аналізу фінансів.

4. Технічні вимоги

Вихідні дані до роботи: середовище розробки Android Studio, мова розробки – Kotlin, операційна система – Android.

5. Конструктивні вимоги.

Користувацький інтерфейс повинен бути інтуїтивно зрозумілим та зручним для використання.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до БДР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії і етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки системи та постановка задачі	14.03.24 – 20.03.24	вик.
2	Розробка архітектури та алгоритмів роботи мобільного застосунку	13.03.24 – 27.03.24	вик.
3	Вибір середовища та мови розробки	27.03.24 – 03.04.24	вик.
4	Розробка мобільного інтелектуального застосунку	03.04.24 – 12.05.24	вик.
5	Тестування мобільного застосунку	12.05.24 – 21.05.24	вик.
6	Оформлення матеріалів до захисту БДР	21.05.24 – 30.05.24	вик.

9. Порядок контролю і приймання

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

Додаток Б. Перевірка на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка мобільного інтелектуального застосунку для аналізу особистих витрат з використанням технологій Kotlin та Compose»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. Катєльніков Д.І.

Оригінальність	**96,2%**
Схожість	3,8%

Аналіз звіту подібності

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

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

□ 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 _____ Педосенко А.Ю.

Керівник роботи _____ Катєльніков Д.І.

Додаток В. Лістинг програми

Лістинг файлу TransactionRepositoryImpl.kt

```
package com.smartmoneymanager.transaction.repository

import androidx.paging.Pager
import androidx.paging.PagingConfig
import androidx.paging.PagingData
import androidx.paging.map
import com.smartmoneymanager.transaction.dao.TransactionDao
import com.smartmoneymanager.transaction.entity.transaction.Transaction
import com.smartmoneymanager.transaction.entity.transaction.TransactionType
import
com.smartmoneymanager.transaction.entity.transactionByCategory.TransactionTypedByCategory
import com.smartmoneymanager.transaction.mapper.TransactionByCategoryMapper
import com.smartmoneymanager.transaction.mapper.TransactionMapper
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.map

class TransactionRepositoryImpl(
    private val transactionDao: TransactionDao,
    private val transactionMapper: TransactionMapper,
    private val transactionByCategoryMapper: TransactionByCategoryMapper
) : TransactionRepository {

    private val pagingSourceFactory = { transactionDao.getTransactions() }
    override suspend fun insert(transaction: Transaction) {
        val entity = transactionMapper.mapToEntity(transaction)
        transactionDao.insert(entity)
    }

    override suspend fun update(transaction: Transaction) {
        val entity = transactionMapper.mapToEntity(transaction)
        transactionDao.update(entity)
    }

    override suspend fun delete(transaction: Transaction) {
        val entity = transactionMapper.mapToEntity(transaction)
        transactionDao.delete(entity)
    }

    override suspend fun getById(id: Long): Transaction {
        val entity = transactionDao.getById(id)
        return transactionMapper.mapFromEntity(entity)
    }

    override suspend fun getLastTransactions(limit: Int): List<Transaction> {
        val entityList = transactionDao.getLastTransactions(limit)
        return transactionMapper.mapFromEntityList(entityList)
    }
```

```
    override suspend fun getTransactionSumByCategory(
        transactionType: TransactionType,
        startDateMillis: Long,
        endDateMillis: Long
    ): List<TransactionTypedByCategory> {
        val entityList = transactionDao.getTransactionSumByCategory(
            transactionType.toString(),
            startDateMillis,
            endDateMillis
        )

        return transactionByCategoryMapper.fromEntityList(entityList)
    }

    override fun getTransactions(): Flow<PagingData<Transaction>> {
        return Pager(
            config = PagingConfig(
                pageSize = 10,
                enablePlaceholders = false,
                initialLoadSize = 15
            ),
            pagingSourceFactory = pagingSourceFactory
        ).flow
            .map { pagingData ->
                pagingData.map { entity ->
                    transactionMapper.mapFromEntity(entity)
                }
            }
    }
}
```

Лістинг файлу BalanceRepositoryImpl.kt

```
package com.smartmoneymanager.transaction.repository

import com.smartmoneymanager.common.extensions.convertToString
import com.smartmoneymanager.transaction.dao.TransactionDao
import com.smartmoneymanager.transaction.entity.transaction.TransactionType

class BalanceRepositoryImpl(
    private val transactionDao: TransactionDao
) : BalanceRepository {
    override suspend fun getCurrentBalance(): String {
        val income = transactionDao.getAllTypedAmountSum(TransactionType.Income.toString())
        val expense = transactionDao.getAllTypedAmountSum(TransactionType.Expense.toString())

        return (income - expense).convertToString()
    }

    override suspend fun getIncomeBalance(): String {
        val income = transactionDao.getAllTypedAmountSum(TransactionType.Income.toString())

        return income.convertToString()
```

```
    }

    override suspend fun getExpenseBalance(): String {
        val expense = transactionDao.getAllTypedAmountSum(TransactionType.Expense.toString())

        return expense.convertToString()
    }

    override suspend fun getIncomeBalanceByDateRange(
        startDateMillis: Long,
        endDateMillis: Long
    ): Float {
        return transactionDao.getTypedAmountSumByDateRange(
            startDateMillis,
            endDateMillis,
            TransactionType.Income.toString()
        )
    }

    override suspend fun getExpenseBalanceByDateRange(
        startDateMillis: Long,
        endDateMillis: Long
    ): Float {
        return transactionDao.getTypedAmountSumByDateRange(
            startDateMillis,
            endDateMillis,
            TransactionType.Expense.toString()
        )
    }
}
```

Лістинг файлу ChatRepositoryImpl.kt

```
package com.smartmoneymanager.advice.repository

import com.smartmoneymanager.advice.entity.Advice
import com.smartmoneymanager.advice.entity.Session
import com.smartmoneymanager.advice.entity.financialGoal.FinancialGoal
import com.smartmoneymanager.advice.local.dao.AdviceDao
import com.smartmoneymanager.advice.local.dao.SessionAndAdviceDao
import com.smartmoneymanager.advice.local.dao.SessionDao
import com.smartmoneymanager.advice.local.entity.SessionEntity
import com.smartmoneymanager.advice.local.mapper.AdviceMapper
import com.smartmoneymanager.advice.local.mapper.SessionMapper
import com.smartmoneymanager.advice.remote.api.AdviceApi
import com.smartmoneymanager.advice.remote.entity.ChatMessageDto
import com.smartmoneymanager.advice.remote.entity.Role
import com.smartmoneymanager.advice.remote.entity.chatRequest.ChatRequestBodyDto
import com.smartmoneymanager.advice.remote.entity.chatResponse.ChatResponseDto
import com.smartmoneymanager.advice.remote.mapper.AdviceDtoMapper
import com.smartmoneymanager.common.base.BaseRepository
import com.smartmoneymanager.common.extensions.convertToFloat
import com.smartmoneymanager.common.extensions.toMillisByDate
```

```
import com.smartmoneymanager.common.utils.DataState

private const val DEEPSEEK_MODEL = "deepseek-chat"
private const val DEEPSEEK_SYSTEM_CONTENT = "You are a helpful finance assistant"
private const val DEEPSEEK_USER_CONTENT =
    "Give me 3 advices for optimizing my expenses, currency is Hryvnia, spaced by \\n symbol based
on the following information:"

class ChatRepositoryImpl(
    private val adviceApi: AdviceApi,
    private val adviceDtoMapper: AdviceDtoMapper,
    private val sessionDao: SessionDao,
    private val adviceDao: AdviceDao,
    private val sessionAndAdviceDao: SessionAndAdviceDao,
    private val sessionMapper: SessionMapper,
    private val adviceMapper: AdviceMapper
) : BaseRepository(), ChatRepository {
    override suspend fun generateAdvices(text: String): DataState<List<Advice>> {
        val messages = listOf(
            ChatMessageDto(
                content = DEEPSEEK_SYSTEM_CONTENT,
                role = Role.System.value
            ),
            ChatMessageDto(
                content = "$DEEPSEEK_USER_CONTENT\n$text",
                role = Role.User.value
            )
        )

        val body = ChatRequestBodyDto(
            model = DEEPSEEK_MODEL,
            temperature = 0.5f,
            messages = messages
        )

        return obtain<ChatResponseDto, List<Advice>>(
            request = adviceApi.getAdvice(body),
            mapper = { adviceDtoMapper.mapChatResponse(it) }
        )
    }

    override suspend fun createSession(
        title: String,
        startDateAnalyze: Long,
        endDateAnalyze: Long,
        financialGoal: FinancialGoal
    ): Long {
        val sessionEntity = SessionEntity(
            title = title,
            creationDate = System.currentTimeMillis(),
            startDateAnalyze = startDateAnalyze,
            endDateAnalyze = endDateAnalyze,
```

```
            amountFinancialGoal = financialGoal.amount.convertToFloat(),
            financialGoalType = financialGoal.type.toString(),
            dateFinancialGoal = financialGoal.date.toMillisByDate(),
            noteFinancialGoal = financialGoal.note
        )

        return sessionDao.insertSession(sessionEntity)
    }

    override suspend fun addAdvicesToSession(sessionId: Long, advices: List<Advice>): Session {
        advices.forEach {
            val adviceEntity = adviceMapper.mapToEntity(it, sessionId)
            adviceDao.insertAdvice(adviceEntity)
        }

        val sessionAndAdvice = sessionAndAdviceDao.getSessionAndAdvice(sessionId)

        return sessionMapper.map(sessionAndAdvice)
    }

    override suspend fun getAllSessions(): List<Session> {
        return sessionAndAdviceDao.getAllSessions().map {
            sessionMapper.map(it)
        }

    }

    override suspend fun getSession(sessionId: Long): Session {
        return sessionMapper.map(sessionAndAdviceDao.getSessionAndAdvice(sessionId))
    }
}
```

Лістинг файлу AdviceApi.kt

```
package com.smartmoneymanager.advice.remote.api

import com.smartmoneymanager.advice.BuildConfig.BASE_URL_DEEPSEEK
import com.smartmoneymanager.advice.remote.entity.chatRequest.ChatRequestBodyDto
import io.ktor.client.HttpClient
import io.ktor.client.request.post
import io.ktor.client.request.setBody
import io.ktor.client.statement.HttpResponse

class AdviceApi(
    private val client: HttpClient
) {
    suspend fun getAdvice(requestBody: ChatRequestBodyDto): HttpResponse {
        return client.post(Endpoints.ChatCompletions.url) {
            setBody(requestBody)
        }
    }

    sealed class Endpoints(val url: String) {
```

```
        data object ChatCompletions : Endpoints("$BASE_URL_DEEPSEEK/chat/completions")
    }
}
```

Лістинг файлу AdviceDao.kt

```
package com.smartmoneymanager.advice.local.dao

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import com.smartmoneymanager.advice.local.entity.AdviceEntity

@Dao
interface AdviceDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAdvice(adviceEntity: AdviceEntity)

    @Delete
    suspend fun deleteAdvice(adviceEntity: AdviceEntity)
}
```

Лістинг файлу SessionAndAdviceDao.kt

```
package com.smartmoneymanager.advice.local.dao

import androidx.room.Dao
import androidx.room.Query
import androidx.room.Transaction
import com.smartmoneymanager.advice.local.entity.SessionAndAdvice

@Dao
interface SessionAndAdviceDao {

    @Transaction
    @Query("SELECT * FROM sessions WHERE sessionId =:sessionId")
    suspend fun getSessionAndAdvice(sessionId: Long): SessionAndAdvice

    @Transaction
    @Query("SELECT * FROM sessions")
    suspend fun getAllSessions(): List<SessionAndAdvice>
}
```

Лістинг файлу SessionDao.kt

```
package com.smartmoneymanager.advice.local.dao

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import com.smartmoneymanager.advice.local.entity.SessionEntity

@Dao
```

```
interface SessionDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertSession(sessionEntity: SessionEntity): Long

    @Delete
    suspend fun deleteSession(sessionEntity: SessionEntity)
}
```

Лістинг файлу AuthRepositoryImpl.kt

```
package com.smartmoneymanager.profile.repository

import android.content.Context
import androidx.credentials.ClearCredentialStateRequest
import androidx.credentials.CredentialManager
import androidx.credentials.CustomCredential
import androidx.credentials.GetCredentialRequest
import com.google.android.libraries.identity.googleid.GetGoogleIdOption
import com.google.android.libraries.identity.googleid.GoogleIdTokenCredential
import com.google.firebase.Firebase
import com.google.firebase.auth.GoogleAuthProvider
import com.google.firebase.auth.auth
import com.smartmoneymanager.common.utils.DataState
import com.smartmoneymanager.profile.ProfileErrorType
import com.smartmoneymanager.profile.entity.User
import kotlinx.coroutines.tasks.await

class AuthRepositoryImpl(private val context: Context) : AuthRepository {

    private val auth = Firebase.auth
    private val credentialManager: CredentialManager = CredentialManager.create(context)

    override suspend fun signIn(): DataState<User> {
        val result =
            credentialManager.getCredential(context = context, request = buildSignInRequest())

        return try {
            val credentials = result.credential as CustomCredential

            when (credentials.type) {
                GoogleIdTokenCredential.TYPE_GOOGLE_ID_TOKEN_CREDENTIAL -> {
                    val googleIdTokenCredential =
                        GoogleIdTokenCredential.createFrom(credentials.data)

                    val credential =
                        GoogleAuthProvider.getCredential(googleIdTokenCredential.idToken, null)

                    auth.signInWithCredential(credential).await().user?.let { user ->
                        val mappedUser = User(
                            uid = user.uid,
                            displayName = user.displayName,
                            email = user.email
                        )
```

```
                DataState.successes(mappedUser)
            } ?: run {
                DataState.error(errorType = ProfileErrorType.UserNotFound)
            }

        }

        else -> {
            DataState.error(errorType = ProfileErrorType.UserNotFound)
        }
      }
    } catch (exception: Exception) {
      exception.printStackTrace()
      DataState.error(errorType = ProfileErrorType.UserNotFound)
    }
  }

  override suspend fun signOut() {
    auth.signOut()
    credentialManager.clearCredentialState(ClearCredentialStateRequest())
  }

  private fun buildSignInRequest(): GetCredentialRequest {
    val googleIdOption: GetGoogleIdOption = GetGoogleIdOption.Builder()
      .setFilterByAuthorizedAccounts(true)
      .setServerClientId("850061527962-
s523n59nt8cuqifb1fqc6cbekn4oln2q.apps.googleusercontent.com")
      .build()

    return GetCredentialRequest.Builder()
      .addCredentialOption(googleIdOption)
      .build()

  }
}
```

Лістинг файлу DatabaseModule.kt

```
package com.smartmoneymanager.di

import androidx.room.Room
import com.smartmoneymanager.advice.local.dao.AdviceDao
import com.smartmoneymanager.advice.local.dao.SessionAndAdviceDao
import com.smartmoneymanager.advice.local.dao.SessionDao
import com.smartmoneymanager.data.MainDatabase
import com.smartmoneymanager.transaction.dao.TransactionDao
import org.koin.android.ext.koin.androidApplication
import org.koin.dsl.module

val databaseModule = module {
  single {
    Room.databaseBuilder(
      androidApplication(),
```

```
            MainDatabase::class.java,
            "MAIN_DATABASE"
        ).build()
    }

    single<TransactionDao> {
        val database = get<MainDatabase>()
        database.transactionDao
    }

    single<AdviceDao> {
        val database = get<MainDatabase>()
        database.adviceDao
    }

    single<SessionDao> {
        val database = get<MainDatabase>()
        database.sessionDao
    }

    single<SessionAndAdviceDao> {
        val database = get<MainDatabase>()
        database.sessionAndAdviceDao
    }
}
```

Лістинг файлу DataStoreModule.kt

```
package com.smartmoneymanager.di

import androidx.datastore.core.DataStore
import androidx.datastore.core.handlers.ReplaceFileCorruptionHandler
import androidx.datastore.preferences.SharedPreferencesMigration
import androidx.datastore.preferences.core.PreferenceDataStoreFactory
import androidx.datastore.preferences.core.Preferences
import androidx.datastore.preferences.core.emptyPreferences
import androidx.datastore.preferences.preferencesDataStoreFile
import com.smartmoneymanager.data.DataStoreManager
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.SupervisorJob
import org.koin.android.ext.koin.androidContext
import org.koin.dsl.module

private const val USER_PREFERENCES = "user_preferences"

val dataStoreModule = module {
    single<DataStore<Preferences>> {
        PreferenceDataStoreFactory.create(
            corruptionHandler = ReplaceFileCorruptionHandler(
                produceNewData = { emptyPreferences() }
            ),
            migrations = listOf(SharedPreferencesMigration(androidContext(),
```

```
USER_PREFERENCES)),
        scope = CoroutineScope(Dispatchers.IO + SupervisorJob()),
        produceFile = { androidContext().preferencesDataStoreFile(USER_PREFERENCES) }
    )
  }

  single<DataStoreManager> { DataStoreManager(get()) }
}
```

Лістинг файлу ManagerModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.common.dateManager.DateManager
import com.smartmoneymanager.data.LanguageManager
import org.koin.dsl.module

val managerModule = module {
  factory { DateManager() }
  single { LanguageManager(get()) }
}
```

Лістинг файлу MapperModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.advice.local.mapper.AdviceMapper
import com.smartmoneymanager.advice.local.mapper.SessionMapper
import com.smartmoneymanager.advice.remote.mapper.AdviceDtoMapper
import com.smartmoneymanager.transaction.mapper.TransactionByCategoryMapper
import com.smartmoneymanager.transaction.mapper.TransactionMapper
import org.koin.dsl.module

val mapperModule = module {
  single { TransactionMapper() }
  single { TransactionByCategoryMapper() }
  single { AdviceDtoMapper() }
  single { AdviceMapper() }
  single { SessionMapper(get()) }
}
```

Лістинг файлу NetworkModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.BuildConfig.DEEPSEEK_API_KEY
import com.smartmoneymanager.advice.remote.api.AdviceApi
import io.ktor.client.HttpClient
import io.ktor.client.engine.okhttp.OkHttp
import io.ktor.client.plugins.DefaultRequest
import io.ktor.client.plugins.HttpTimeout
import io.ktor.client.plugins.contentnegotiation.ContentNegotiation
import io.ktor.client.plugins.logging.DEFAULT
import io.ktor.client.plugins.logging.LogLevel
import io.ktor.client.plugins.logging.Logger
```

```
import io.ktor.client.plugins.logging.Logging
import io.ktor.client.request.header
import io.ktor.http.ContentType
import io.ktor.http.HttpHeaders
import io.ktor.http.contentType
import io.ktor.serialization.kotlinx.json.json
import kotlinx.serialization.json.Json
import org.koin.dsl.module

val networkModule = module {
    single<HttpClient> {
        HttpClient(OkHttp) {

            install(Logging) {
                logger = Logger.DEFAULT
                level = LogLevel.BODY
            }

            install(ContentNegotiation) {
                json(
                    Json {
                        prettyPrint = true
                        ignoreUnknownKeys = true
                    }
                )
            }

            install(HttpTimeout) {
                requestTimeoutMillis = 45_000
                connectTimeoutMillis = 45_000
                socketTimeoutMillis = 45_000
            }

            install(DefaultRequest) {
                header(HttpHeaders.Authorization, "Bearer $DEEPSEEK_API_KEY")
                contentType(ContentType.Application.Json)
            }
        }
    }

    single<AdviceApi> { AdviceApi(get()) }
}
```

Лістинг файлу RepositoryModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.BuildConfig.DEEPSEEK_API_KEY
import com.smartmoneymanager.advice.remote.api.AdviceApi
import io.ktor.client.HttpClient
import io.ktor.client.engine.okhttp.OkHttp
import io.ktor.client.plugins.DefaultRequest
import io.ktor.client.plugins.HttpTimeout
```

```
import io.ktor.client.plugins.contentnegotiation.ContentNegotiation
import io.ktor.client.plugins.logging.DEFAULT
import io.ktor.client.plugins.logging.LogLevel
import io.ktor.client.plugins.logging.Logger
import io.ktor.client.plugins.logging.Logging
import io.ktor.client.request.header
import io.ktor.http.ContentType
import io.ktor.http.HttpHeaders
import io.ktor.http.contentType
import io.ktor.serialization.kotlinx.json.json
import kotlinx.serialization.json.Json
import org.koin.dsl.module

val networkModule = module {
    single<HttpClient> {
        HttpClient(OkHttp) {

            install(Logging) {
                logger = Logger.DEFAULT
                level = LogLevel.BODY
            }

            install(ContentNegotiation) {
                json(
                    Json {
                        prettyPrint = true
                        ignoreUnknownKeys = true
                    }
                )
            }

            install(HttpTimeout) {
                requestTimeoutMillis = 45_000
                connectTimeoutMillis = 45_000
                socketTimeoutMillis = 45_000
            }

            install(DefaultRequest) {
                header(HttpHeaders.Authorization, "Bearer $DEEPSEEK_API_KEY")
                contentType(ContentType.Application.Json)
            }
        }
    }

    single<AdviceApi> { AdviceApi(get()) }
}
```

Лістинг файлу UseCaseModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.advice.usecase.CreateAdviceBasedOnSelectedInfoUseCase
import com.smartmoneymanager.advice.usecase.CreateSessionUseCase
```

```
import com.smartmoneymanager.advice.usecase.GetAllSessionsUseCase
import com.smartmoneymanager.advice.usecase.GetSessionUseCase
import com.smartmoneymanager.advice.usecase.HandleNewAdviceUseCase
import com.smartmoneymanager.transaction.usecase.GetBalanceByRangeUseCase
import com.smartmoneymanager.transaction.usecase.GetLastTransactionsUseCase
import com.smartmoneymanager.transaction.usecase.GetTransactionsSumByCategoryUseCase
import org.koin.dsl.module

val useCaseModule = module {
    factory { GetBalanceByRangeUseCase(get()) }
    factory { GetLastTransactionsUseCase(get()) }
    factory { GetTransactionsSumByCategoryUseCase(get()) }
    factory { CreateAdviceBasedOnSelectedInfoUseCase(get(), get()) }
    factory { CreateSessionUseCase(get()) }
    factory { GetAllSessionsUseCase(get()) }
    factory { GetSessionUseCase(get()) }
    factory { HandleNewAdviceUseCase(get(), get(), get()) }
}
```

Лістинг файлу ViewModelModule.kt

```
package com.smartmoneymanager.di

import com.smartmoneymanager.advice.usecase.CreateAdviceBasedOnSelectedInfoUseCase
import com.smartmoneymanager.advice.usecase.CreateSessionUseCase
import com.smartmoneymanager.advice.usecase.GetAllSessionsUseCase
import com.smartmoneymanager.advice.usecase.GetSessionUseCase
import com.smartmoneymanager.advice.usecase.HandleNewAdviceUseCase
import com.smartmoneymanager.transaction.usecase.GetBalanceByRangeUseCase
import com.smartmoneymanager.transaction.usecase.GetLastTransactionsUseCase
import com.smartmoneymanager.transaction.usecase.GetTransactionsSumByCategoryUseCase
import org.koin.dsl.module

val useCaseModule = module {
    factory { GetBalanceByRangeUseCase(get()) }
    factory { GetLastTransactionsUseCase(get()) }
    factory { GetTransactionsSumByCategoryUseCase(get()) }
    factory { CreateAdviceBasedOnSelectedInfoUseCase(get(), get()) }
    factory { CreateSessionUseCase(get()) }
    factory { GetAllSessionsUseCase(get()) }
    factory { GetSessionUseCase(get()) }
    factory { HandleNewAdviceUseCase(get(), get(), get()) }
}
```

Лістинг файлу AnalyticsNavGraph.kt

```
package com.smartmoneymanager.navigation

import androidx.compose.foundation.layout.PaddingValues
import androidx.navigation.NavController
import androidx.navigation.NavGraphBuilder
import androidx.navigation.compose.composable
import androidx.navigation.navigation
import com.smartmoneymanager.screens.analytics.AnalyticsScreen
```

```
import org.koin.androidx.compose.koinViewModel

fun NavGraphBuilder.analyticsNavGraph(navController: NavController, paddingValues:
PaddingValues) {
    navigation(
        startDestination = AnalyticsNavRoute.Analytics.screenRoute,
        route = RootRoute.Analytics.route
    ) {
        composable(route = AnalyticsNavRoute.Analytics.screenRoute) {
            AnalyticsScreen(
                viewModel = koinViewModel(),
                paddingValues = paddingValues
            ) {
                navController.navigate(AssistantNavRoute.AssistantChat.screenRoute)
            }
        }
    }
}

sealed class AnalyticsNavRoute(val screenRoute: String) {
    data object Analytics : AnalyticsNavRoute("analytics_screen")
}
```

Лістинг файлу AssistantNavGraph.kt

```
package com.smartmoneymanager.navigation

import androidx.compose.foundation.layout.PaddingValues
import androidx.navigation.NavController
import androidx.navigation.NavGraphBuilder
import androidx.navigation.NavType
import androidx.navigation.compose.composable
import androidx.navigation.navArgument
import androidx.navigation.navigation
import com.smartmoneymanager.screens.assistantChat.AssistantChatScreen
import com.smartmoneymanager.screens.assistantChat.AssistantChatViewModel
import com.smartmoneymanager.screens.selectAssistantData.SelectAssistantDataScreen
import com.smartmoneymanager.screens.selectAssistantData.SelectAssistantDataViewModel
import org.koin.androidx.compose.koinViewModel

fun NavGraphBuilder.assistantNavGraph(navController: NavController, paddingValues:
PaddingValues) {
    navigation(
        startDestination = AssistantNavRoute.SelectData.screenRoute,
        route = RootRoute.Assistant.route
    ) {
        composable(route = AssistantNavRoute.SelectData.screenRoute) {
            val viewModel = koinViewModel<SelectAssistantDataViewModel>()
            SelectAssistantDataScreen(
                viewModel = viewModel,
                paddingValues = paddingValues
            ) { sessionId ->
                navController.navigate("${AssistantNavRoute.AssistantChat.screenRoute}/$sessionId")
```

```
            }
        }

        composable(
            route = "${AssistantNavRoute.AssistantChat.screenRoute}/{sessionId}",
            arguments = listOf(
                navArgument("sessionId") { type = NavType.LongType },
            )
        ) {
            val arguments = it.arguments!!

            val sessionId = arguments.getLong("sessionId")

            AssistantChatScreen(
                viewModel = koinViewModel<AssistantChatViewModel>(),
                sessionId = sessionId,
                paddingValues = paddingValues
            )
        }
    }
}

sealed class AssistantNavRoute(val screenRoute: String) {
    data object SelectData : AssistantNavRoute("select_data_screen")
    data object AssistantChat : AssistantNavRoute("assistant_chat_screen")
}
```

Лістинг файлу HomeNavGraph.kt

```
package com.smartmoneymanager.navigation

import androidx.compose.foundation.layout.PaddingValues
import androidx.navigation.NavController
import androidx.navigation.NavGraphBuilder
import androidx.navigation.compose.composable
import androidx.navigation.navigation
import com.smartmoneymanager.screens.allTransactions.AllTransactionsScreen
import com.smartmoneymanager.screens.home.HomeScreen
import org.koin.androidx.compose.koinViewModel

fun NavGraphBuilder.homeNavGraph(navController: NavController, paddingValues:
PaddingValues) {
    navigation(startDestination = HomeNavRoute.Home.screenRoute, route =
RootRoute.Home.route) {
        composable(route = HomeNavRoute.Home.screenRoute) {
            HomeScreen(
                viewModel = koinViewModel(),
                paddingValues = paddingValues,
                navigateTo = {
                    navController.navigate(it.screenRoute)
                }
            )
        }
```

```
        composable(route = HomeNavRoute.AllTransactions.screenRoute) {
            AllTransactionsScreen(
                viewModel = koinViewModel(),
                paddingValues = paddingValues,
                navigateBack = {
                    navController.popBackStack()
                }
            )
        }
    }
}

sealed class HomeNavRoute(val screenRoute: String) {
    data object Home : HomeNavRoute("home_screen")

    data object AllTransactions : HomeNavRoute("all_transactions")
}
```

Лістинг файлу ProfileNavGraph.kt

```
package com.smartmoneymanager.navigation

import androidx.compose.foundation.layout.PaddingValues
import androidx.navigation.NavController
import androidx.navigation.NavGraphBuilder
import androidx.navigation.compose.composable
import androidx.navigation.navigation
import com.smartmoneymanager.screens.main.MainUiEvent
import com.smartmoneymanager.screens.profile.ProfileScreen
import com.smartmoneymanager.screens.signIn.SignInScreen
import org.koin.androidx.compose.koinViewModel

fun NavGraphBuilder.profileNavGraph(
    navController: NavController,
    onMainEvent: (MainUiEvent) -> Unit,
    paddingValues: PaddingValues
) {
    navigation(
        startDestination = ProfileNavRoute.Profile.screenRoute,
        route = RootRoute.Profile.route
    ) {
        composable(route = ProfileNavRoute.Profile.screenRoute) {
            ProfileScreen(
                viewModel = koinViewModel(),
                onMainEvent = onMainEvent,
                paddingValues = paddingValues,
                navigateToSignIn = {
                    navController.navigate(ProfileNavRoute.SignIn.screenRoute)
                }
            )
        }
```

```
        composable(route = ProfileNavRoute.SignIn.screenRoute) {
            SignInScreen(
                viewModel = koinViewModel(),
                paddingValues = paddingValues,
                navigateToProfile = {
                    navController.navigate(ProfileNavRoute.Profile.screenRoute) {
                        popUpTo(ProfileNavRoute.Profile.screenRoute)
                    }
                }
            )
        }
    }
}

sealed class ProfileNavRoute(val screenRoute: String) {
    data object Profile : ProfileNavRoute("profile_screen")
    data object SignIn : ProfileNavRoute("sign_in_screen")
}
```

Лістинг файлу RootNavHost.kt

```
package com.smartmoneymanager.navigation

import androidx.compose.animation.core.tween
import androidx.compose.animation.slideInVertically
import androidx.compose.animation.slideOutVertically
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.smartmoneymanager.screens.main.MainUiEvent
import com.smartmoneymanager.screens.newTransaction.NewTransactionScreen
import org.koin.androidx.compose.koinViewModel

@Composable
fun RootNavHost(
    navHostController: NavHostController,
    onMainEvent: (MainUiEvent) -> Unit,
    paddingValues: PaddingValues
) {
    NavHost(
        navController = navHostController,
        startDestination = RootRoute.Home.route
    ) {
        homeNavGraph(navHostController, paddingValues)

        assistantNavGraph(navHostController, paddingValues)

        analyticsNavGraph(navHostController, paddingValues)

        profileNavGraph(navHostController, onMainEvent, paddingValues)
```

```
        composable(
            route = RootRoute.NewTransaction.route,
            enterTransition = {
                slideInVertically(
                    initialOffsetY = { it },
                    animationSpec = tween(500)
                )
            },
            exitTransition = {
                slideOutVertically(
                    targetOffsetY = { it },
                    animationSpec = tween(500)
                )
            }
        ) {

            NewTransactionScreen(
                viewModel = koinViewModel()
            ) {
                navHostController.popBackStack()
            }
        }
    }
}

sealed class RootRoute(val route: String) {
    data object Home : RootRoute("home")
    data object Assistant : RootRoute("assistant")
    data object Analytics : RootRoute("analytics")
    data object Profile : RootRoute("profile")
    data object NewTransaction : RootRoute("new_transaction")
}
```

Додаток Г. Графічна частина

ГРАФІЧНА ЧАСТИНА
РОЗРОБКА МОБІЛЬНОГО ІНТЕЛЕКТУАЛЬНОГО ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ОСОБИСТИХ ВИТРАТ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ KOTLIN ТА COMPOSE

Рисунок Г.1 – Титульний слайд

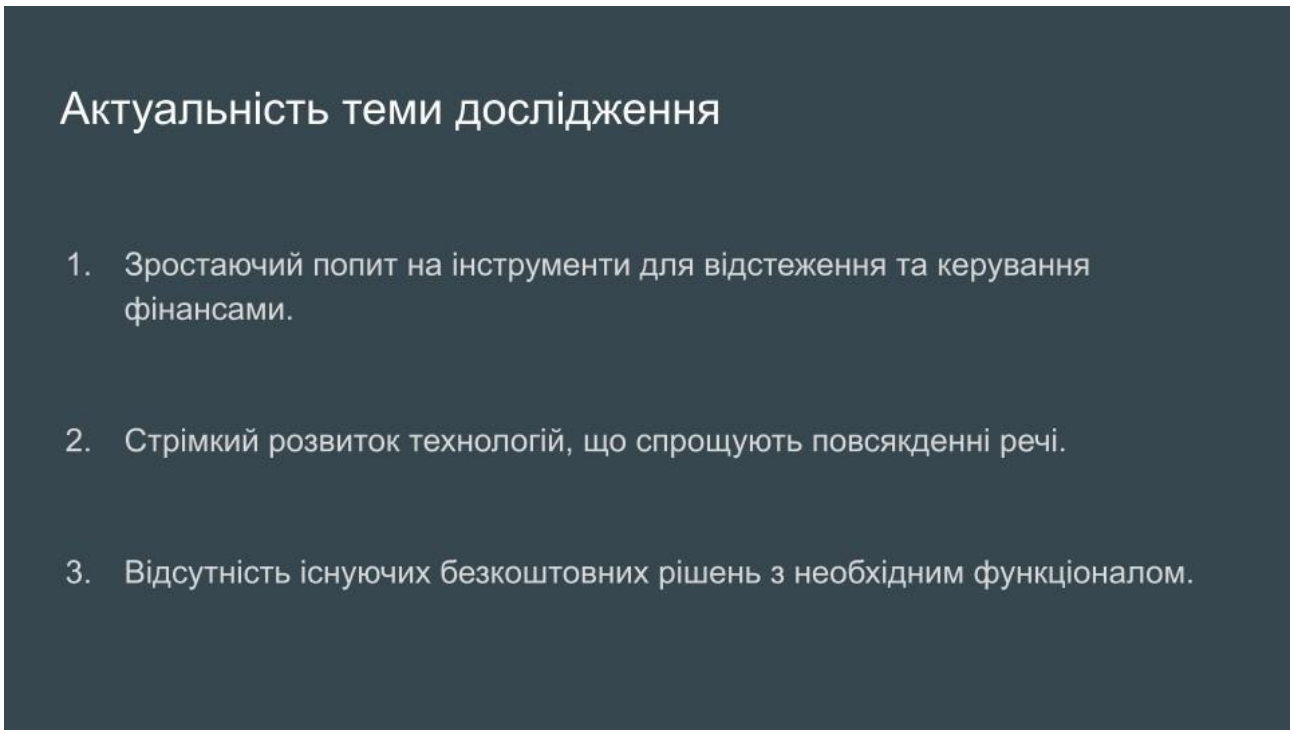

Рисунок Г.2 – Актуальність теми дослідження

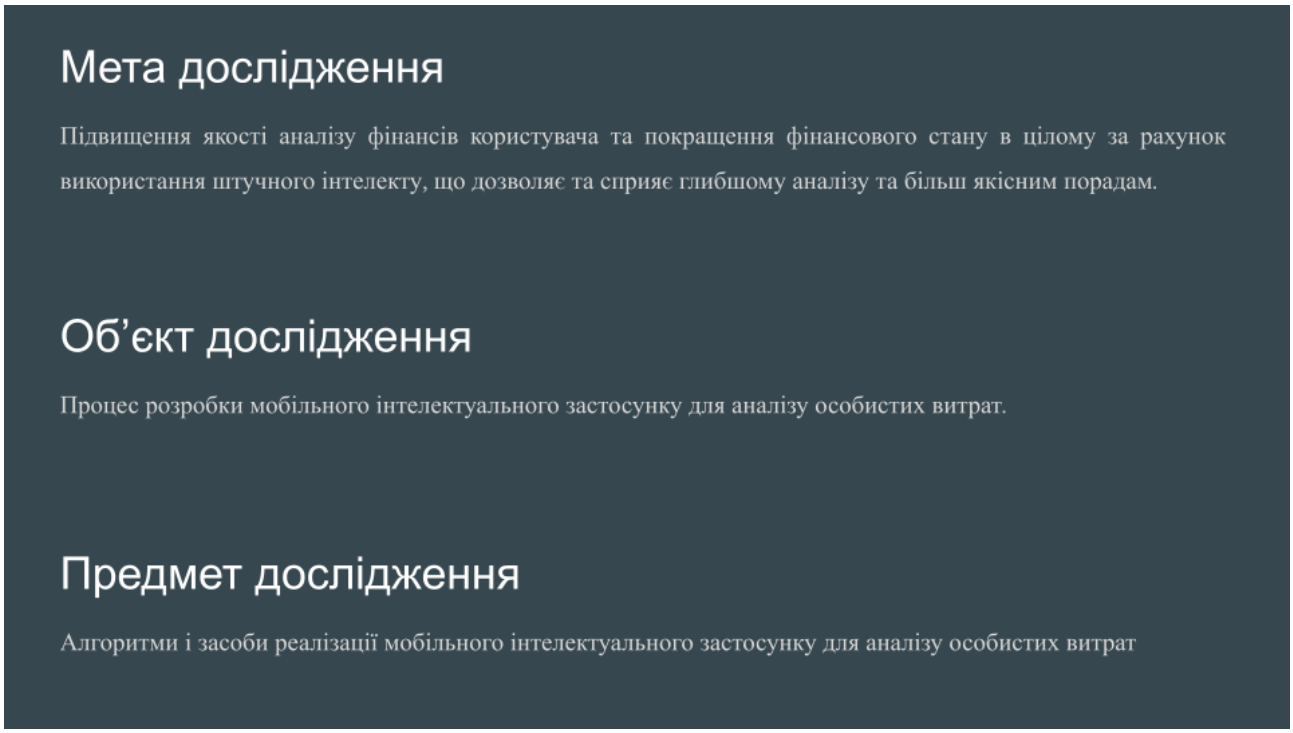

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

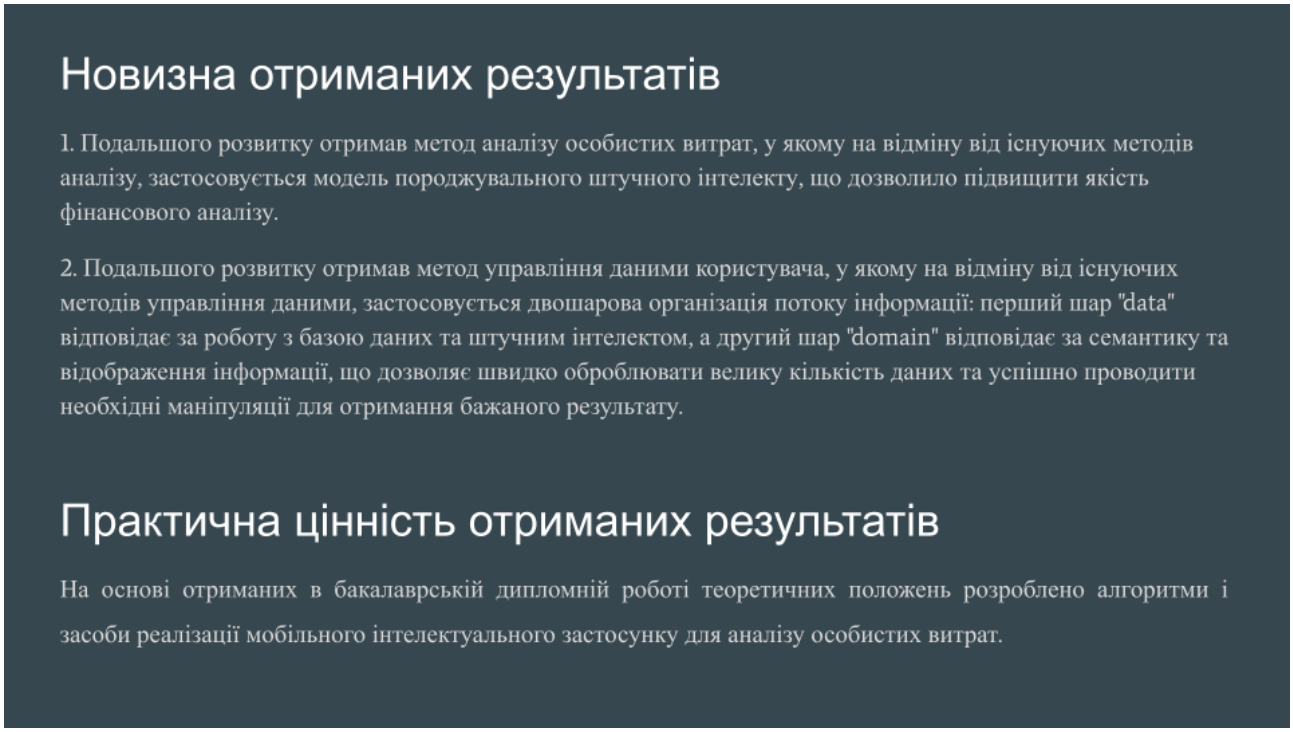

Рисунок Г.4 – Новизна та практична цінність отриманих результатів

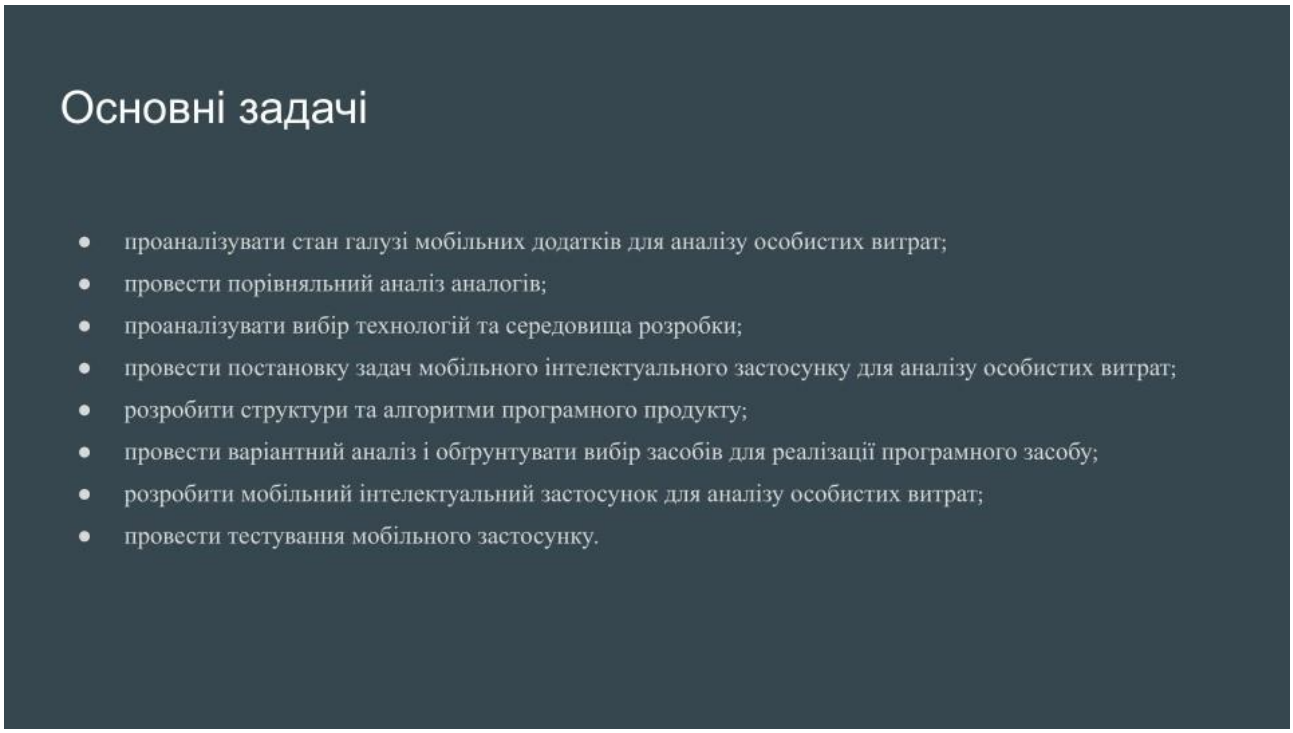

Рисунок Г.5 – Основні задачі дослідження

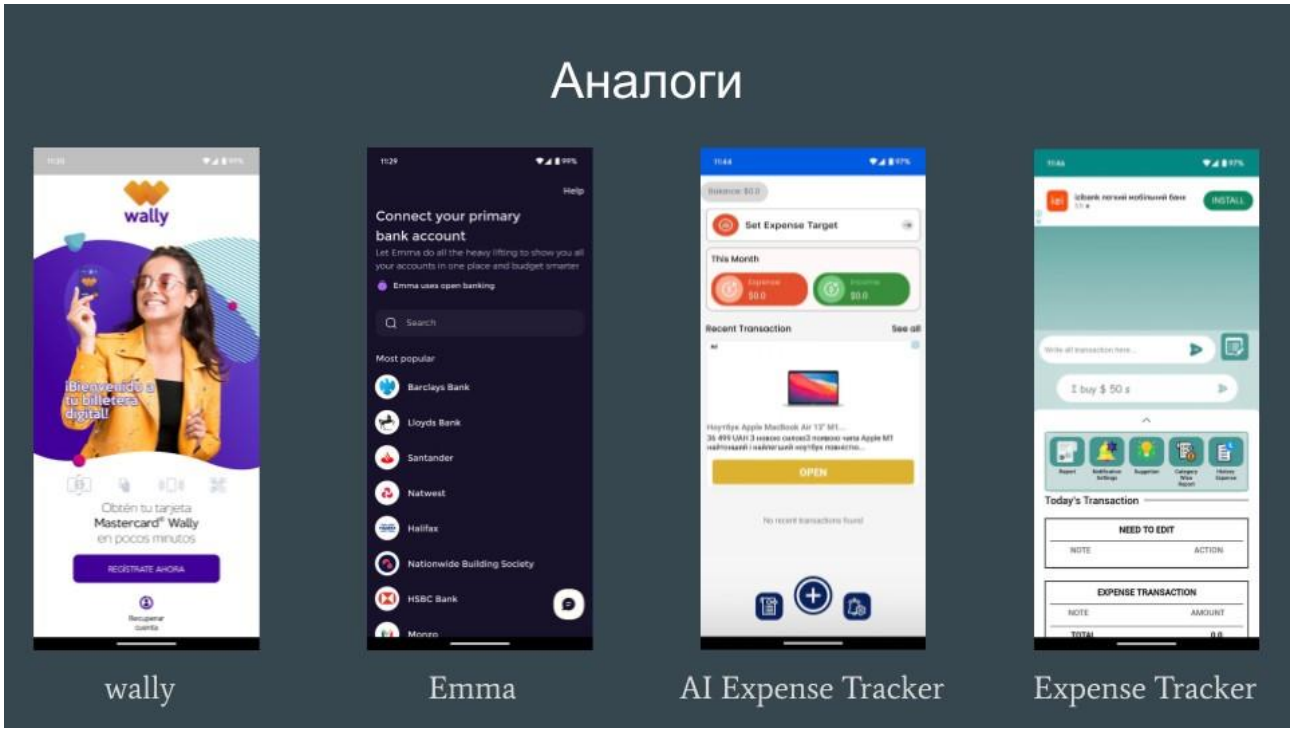

Рисунок Г.6 - Аналоги

Порівняння аналогів

Критерії	wally	Emma	AI Expense Tracker	Expense Tracker	Власне програмне забезпечення
Наявність штучного інтелекту	+	+	+	+	+
Встановлення фінансових цілей	+	+	-	-	+
Підтримка багатомовності	-	+	-	-	+
Функції статистики	-	+	-	+	+
Експорт даних	+	-	+	-	+
Загальна оцінка	60%	80%	40%	40%	100%

Рисунок Г.7 – Порівняння аналогів

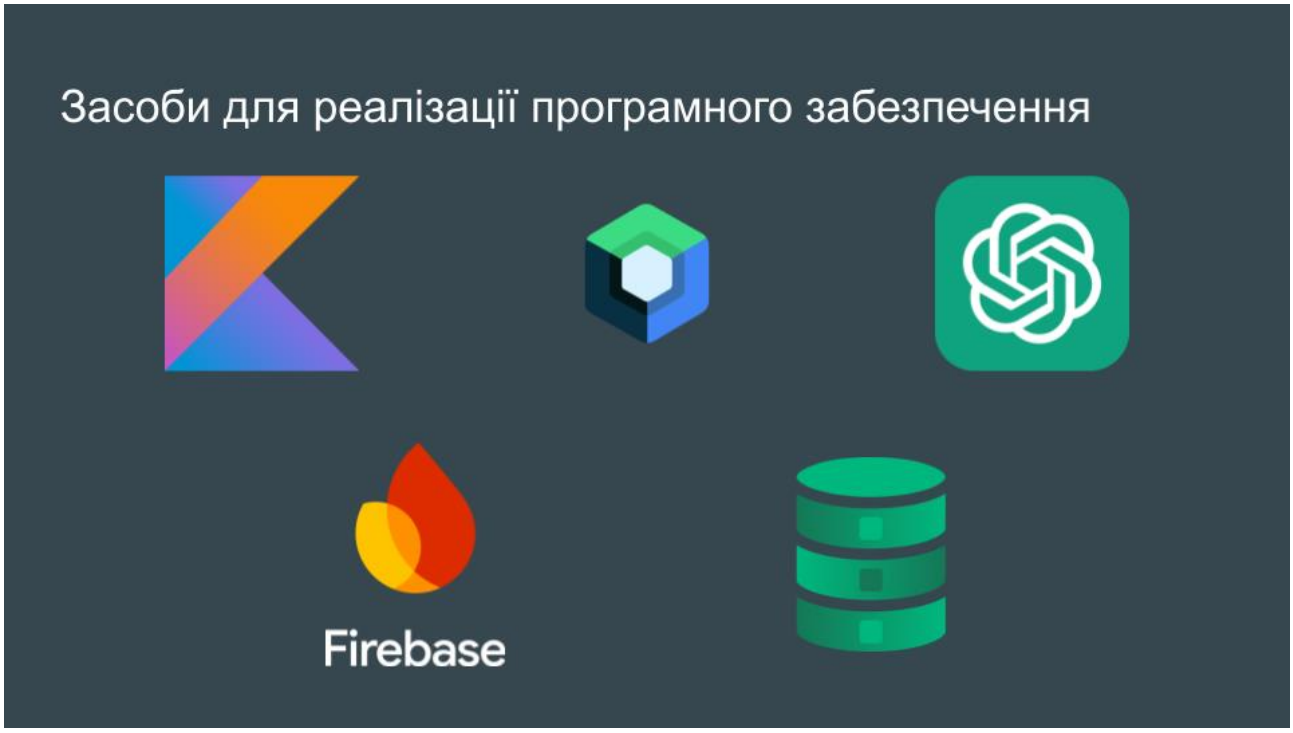

Рисунок Г.8 – Засоби для реалізації програмного забезпечення

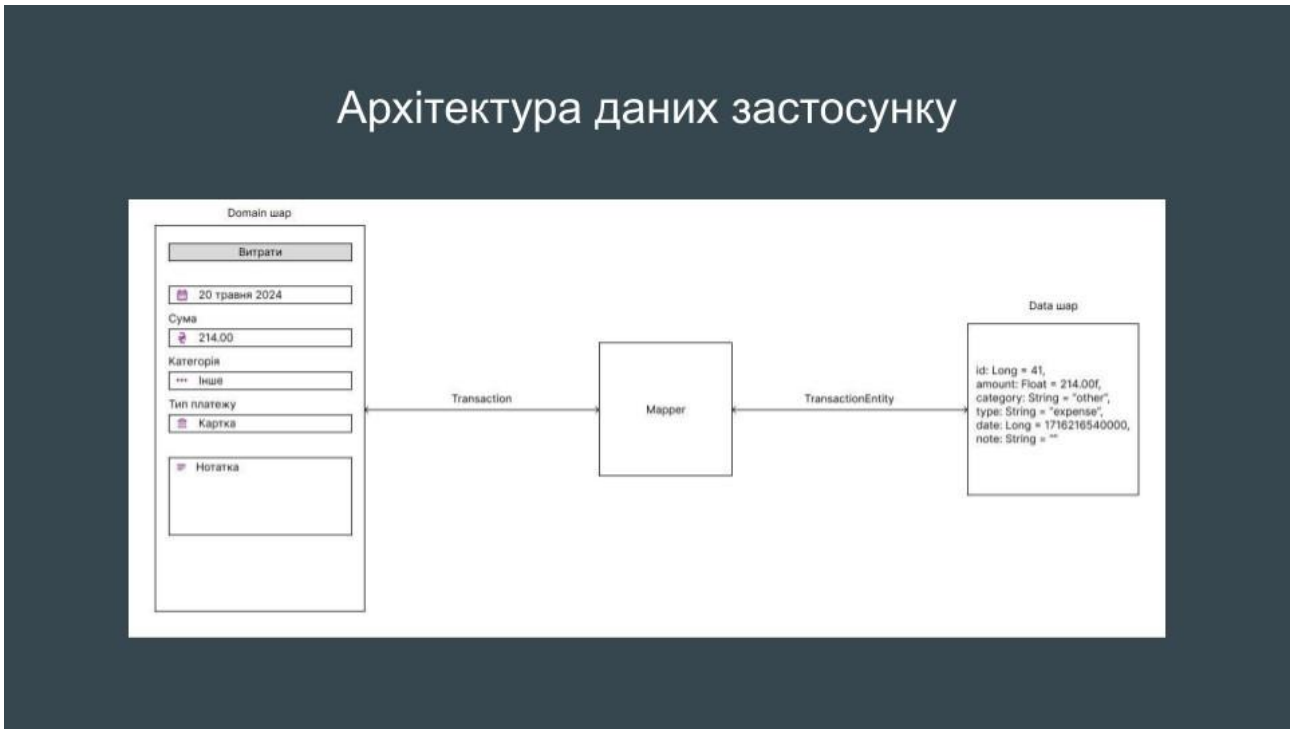

Рисунок Г.9 – Архітектура даних застосунку

Рисунок Г.10 – Загальний алгоритм роботи мобільного застосунку

Рисунок Г.11 – Алгоритм збереження даних

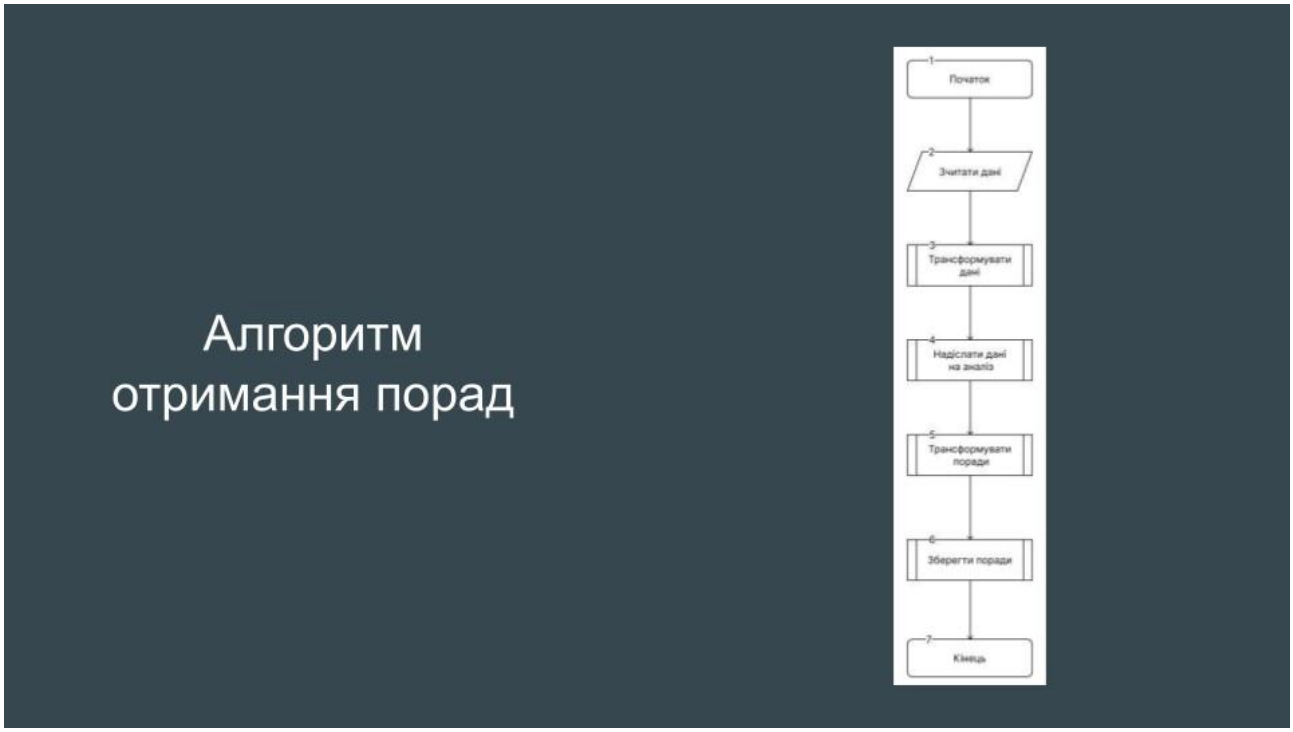

Рисунок Г.12 – Алгоритм отримання порад

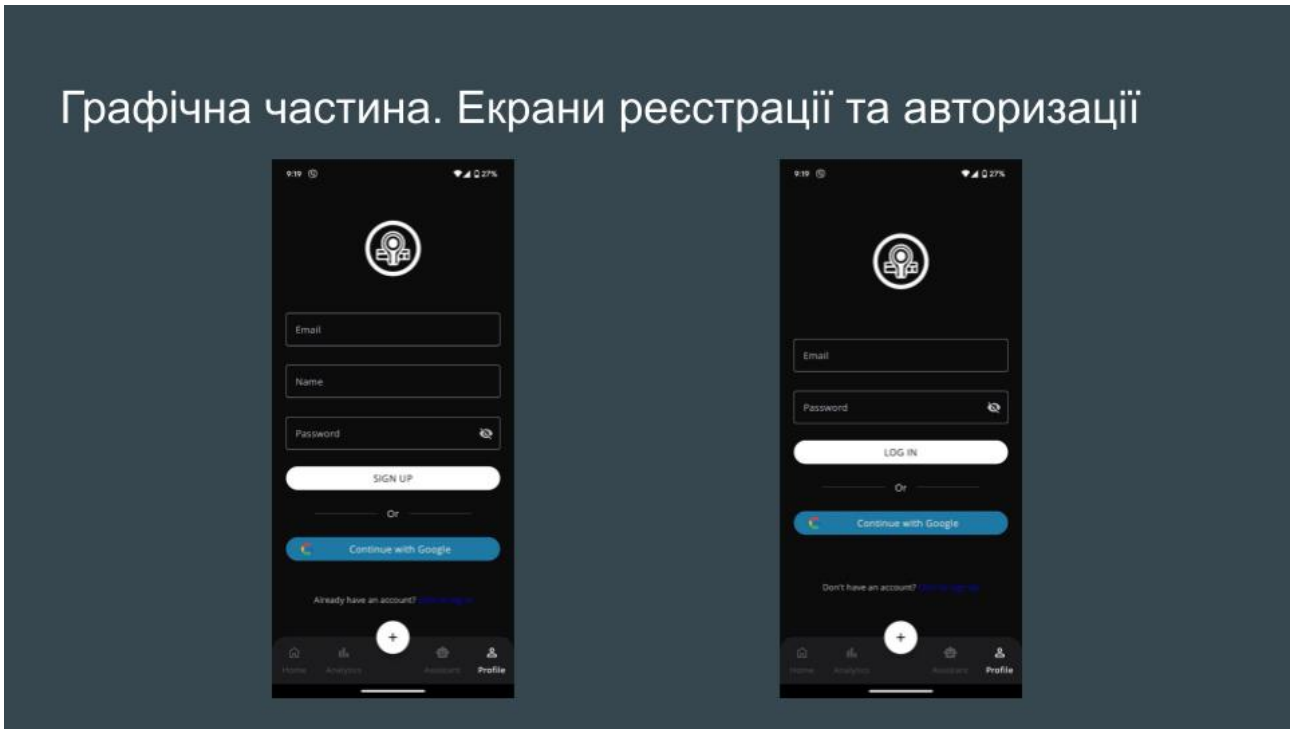

Рисунок Г.13 – Графічна частина. Екрани реєстрації та авторизації

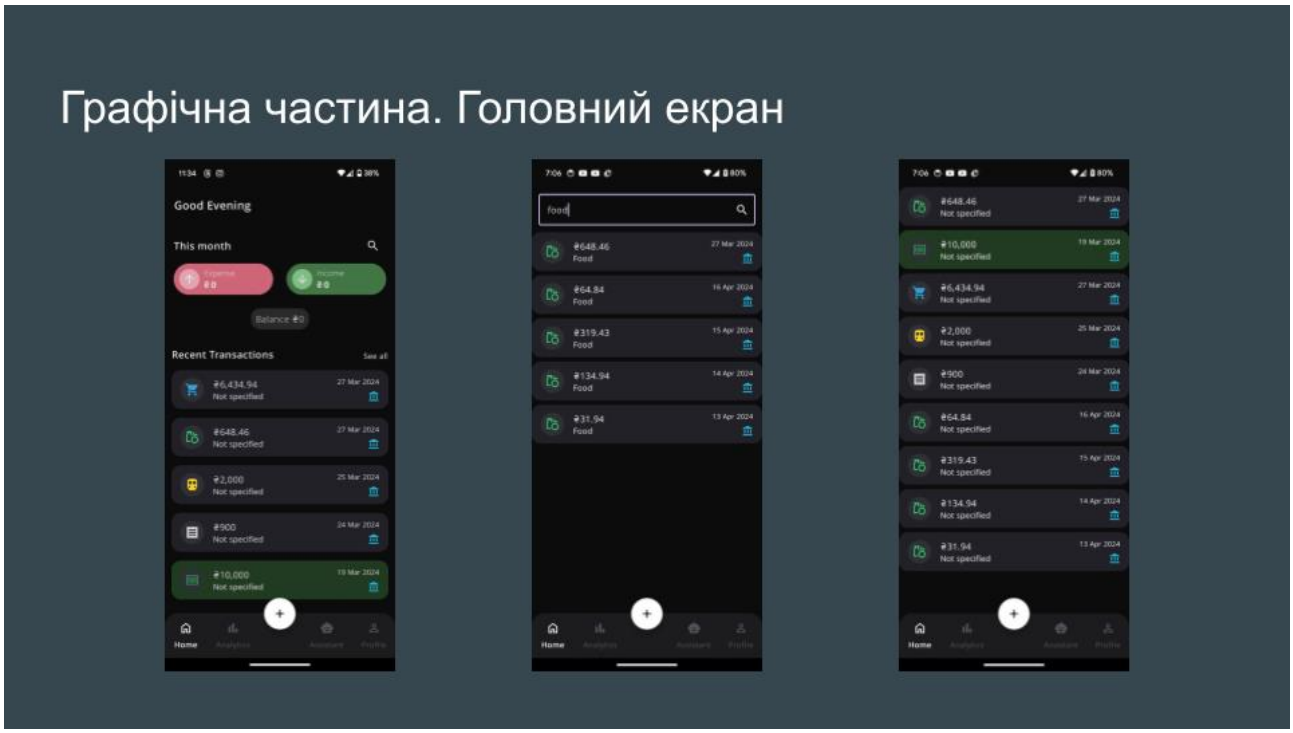

Рисунок Г.14 – Графічна частина. Головний екран

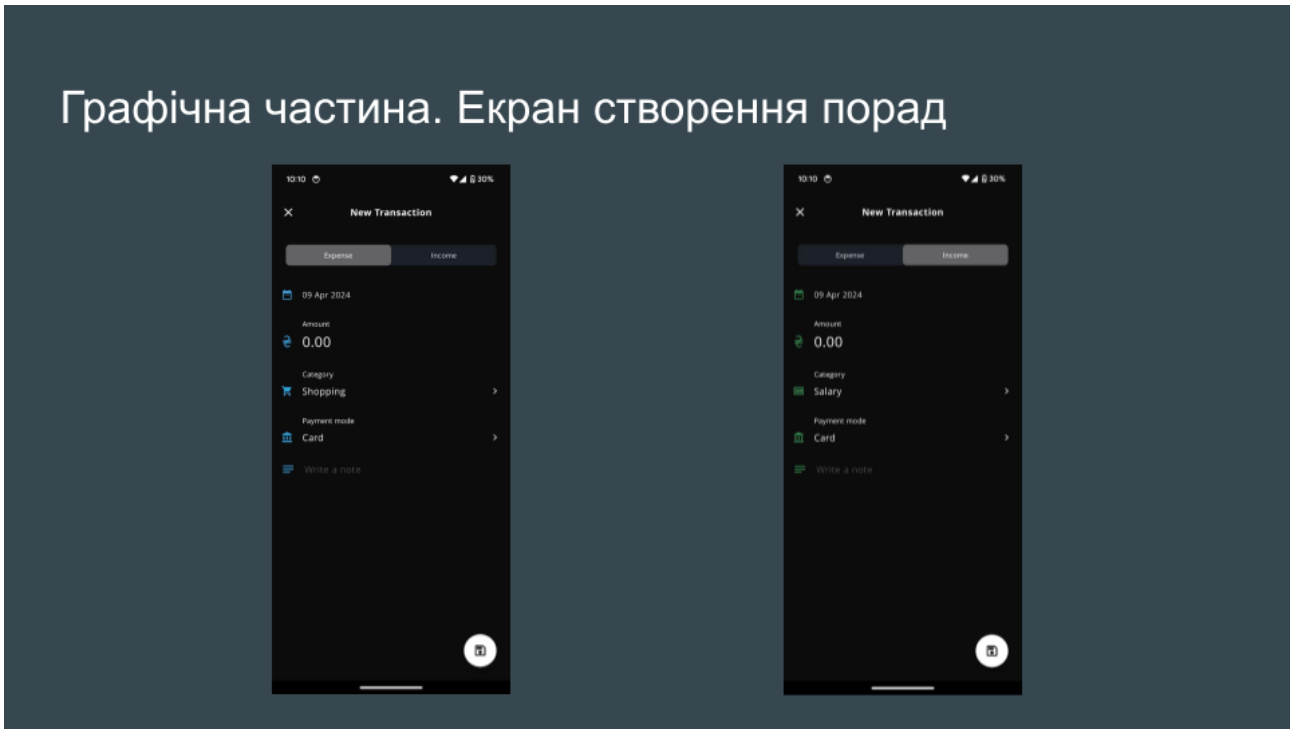

Рисунок Г.15 – Графічна частина. Екран створення порад

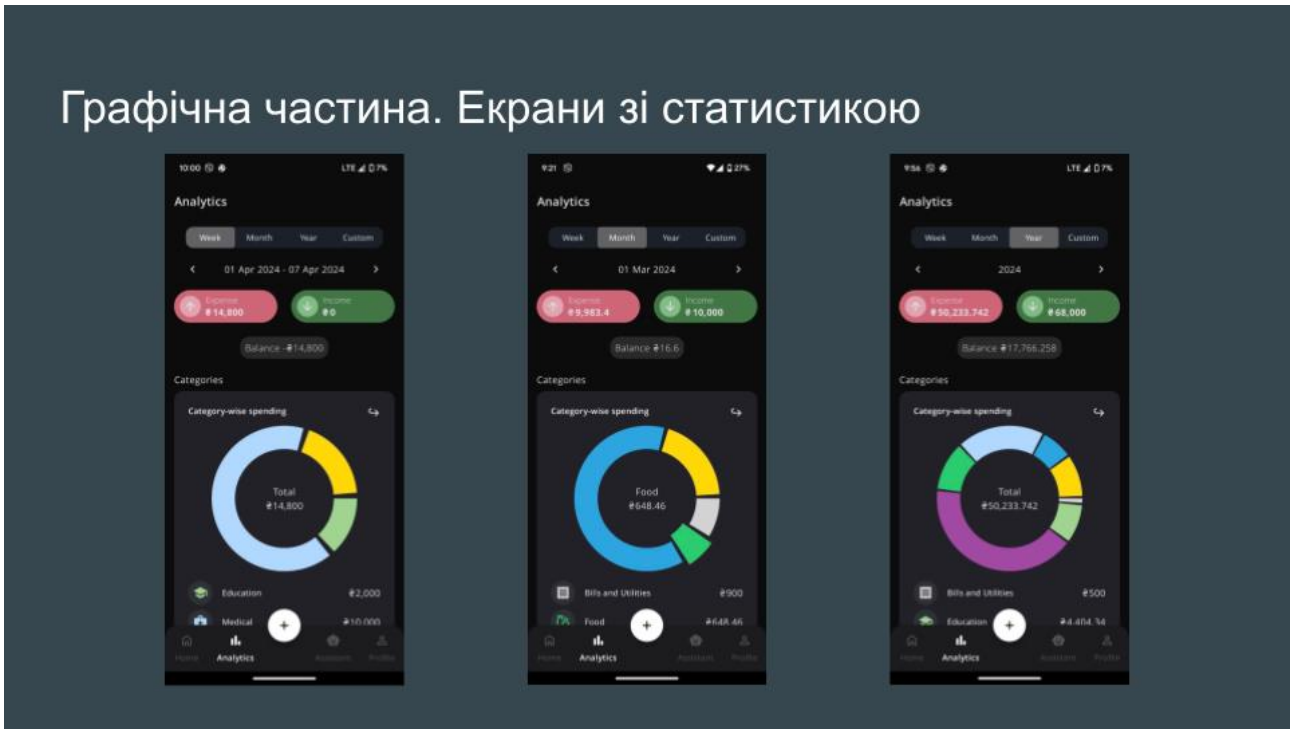

Рисунок Г.16 – Графічна частина. Екрани зі статистикою

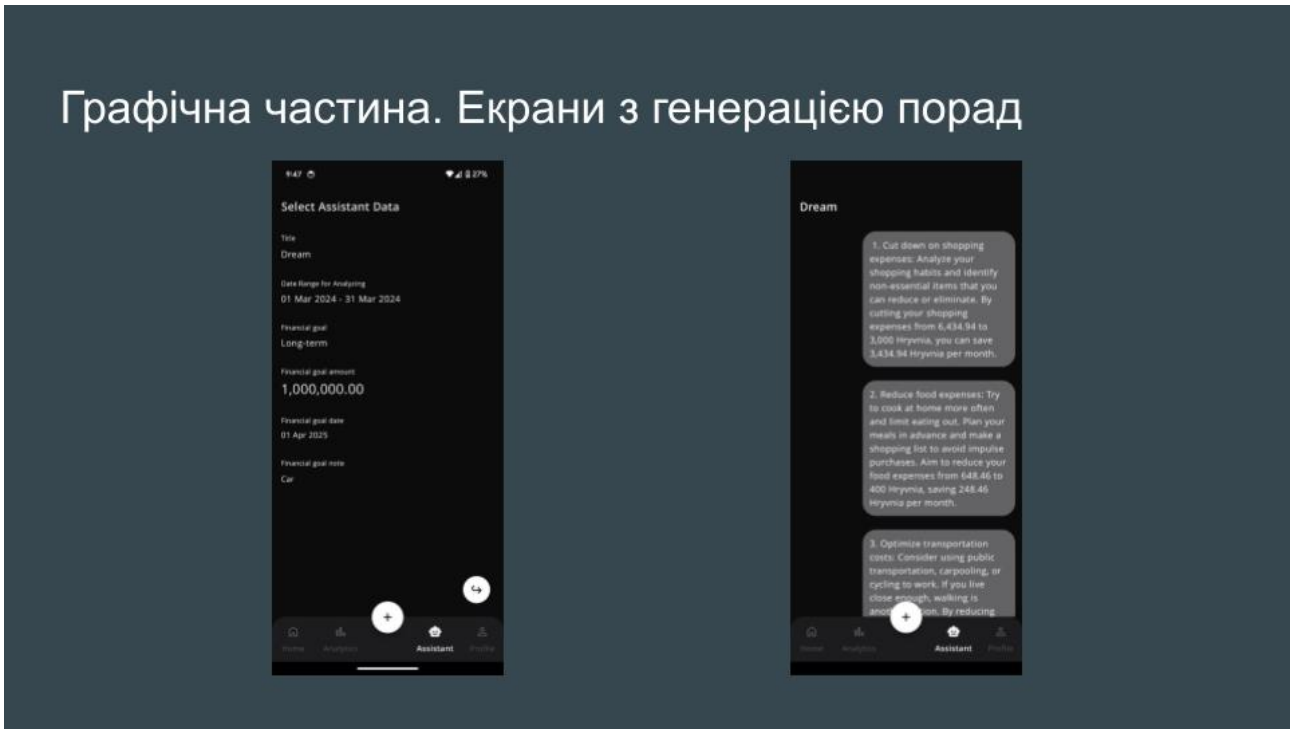

Рисунок Г.17 – Графічна частина. Екрани з генерацією порад

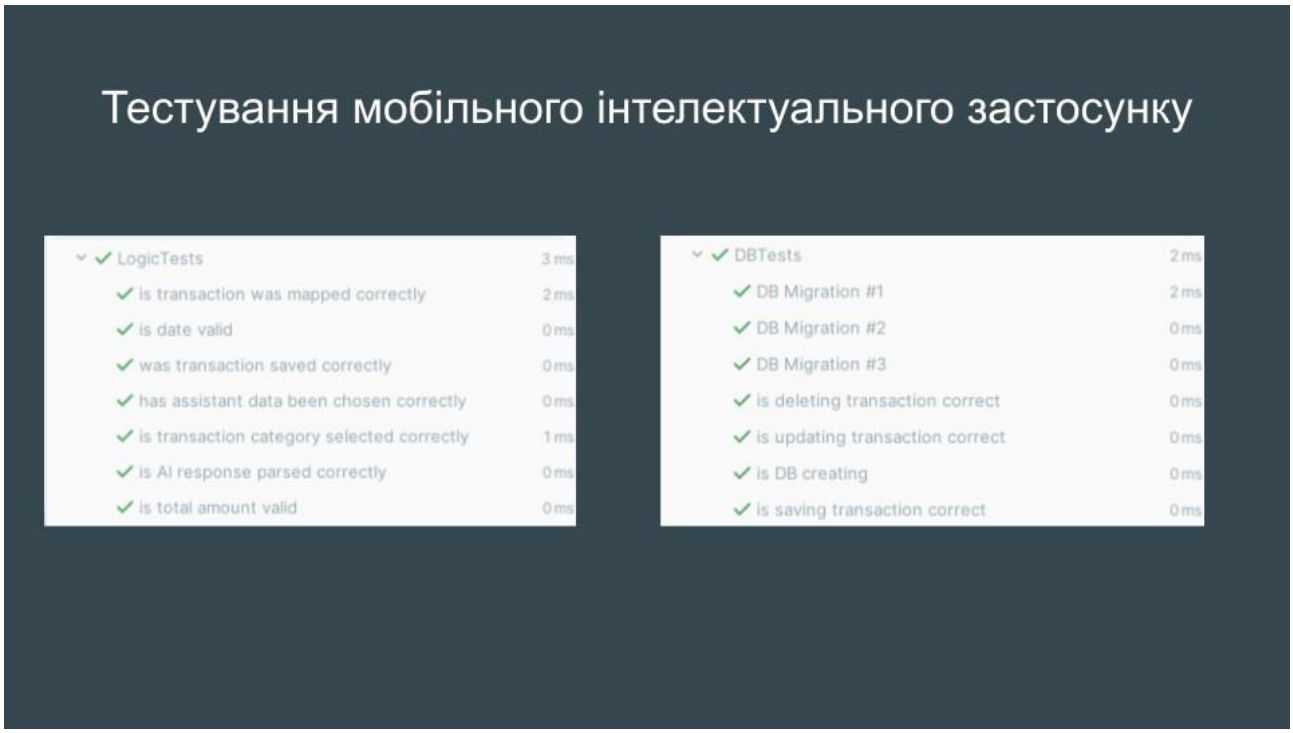

Рисунок Г.18 – Тестування мобільного інтелектуального застосунку (Unit тести)

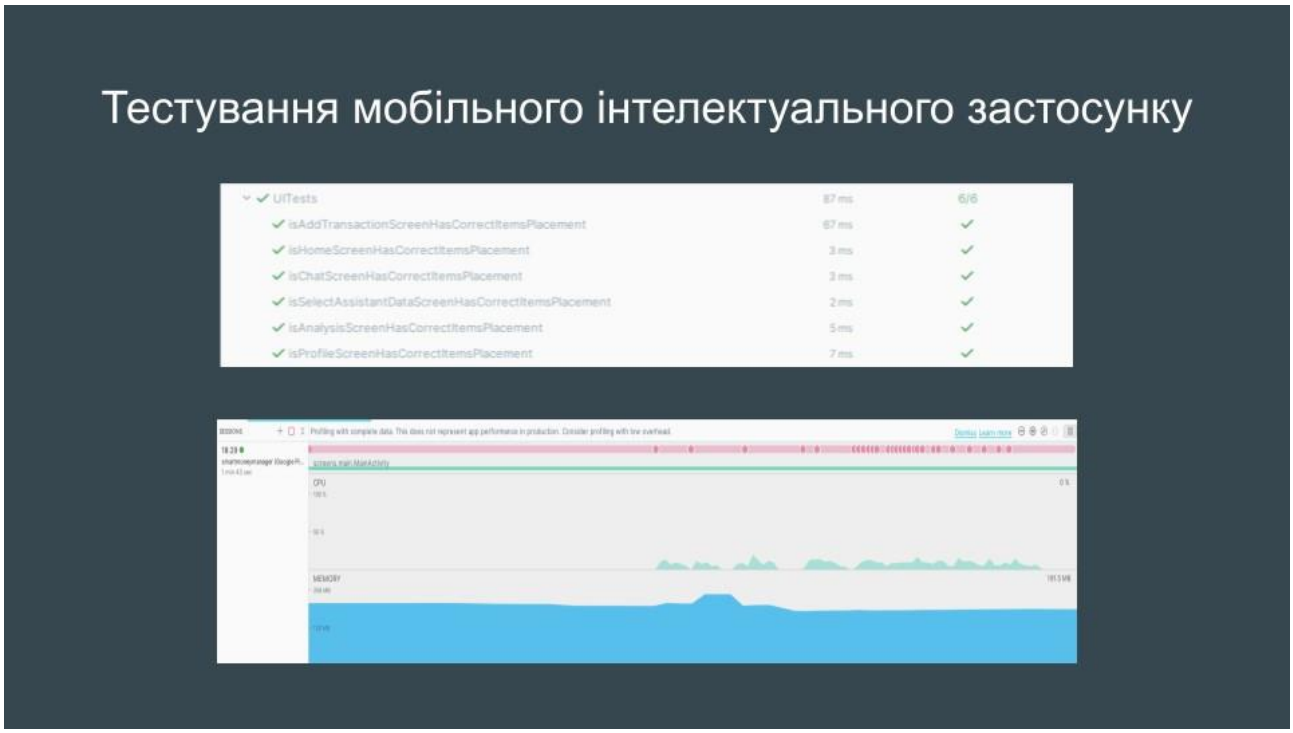

Рисунок Г.19 – Тестування мобільного інтелектуального застосунку (UI тести та Profiler)

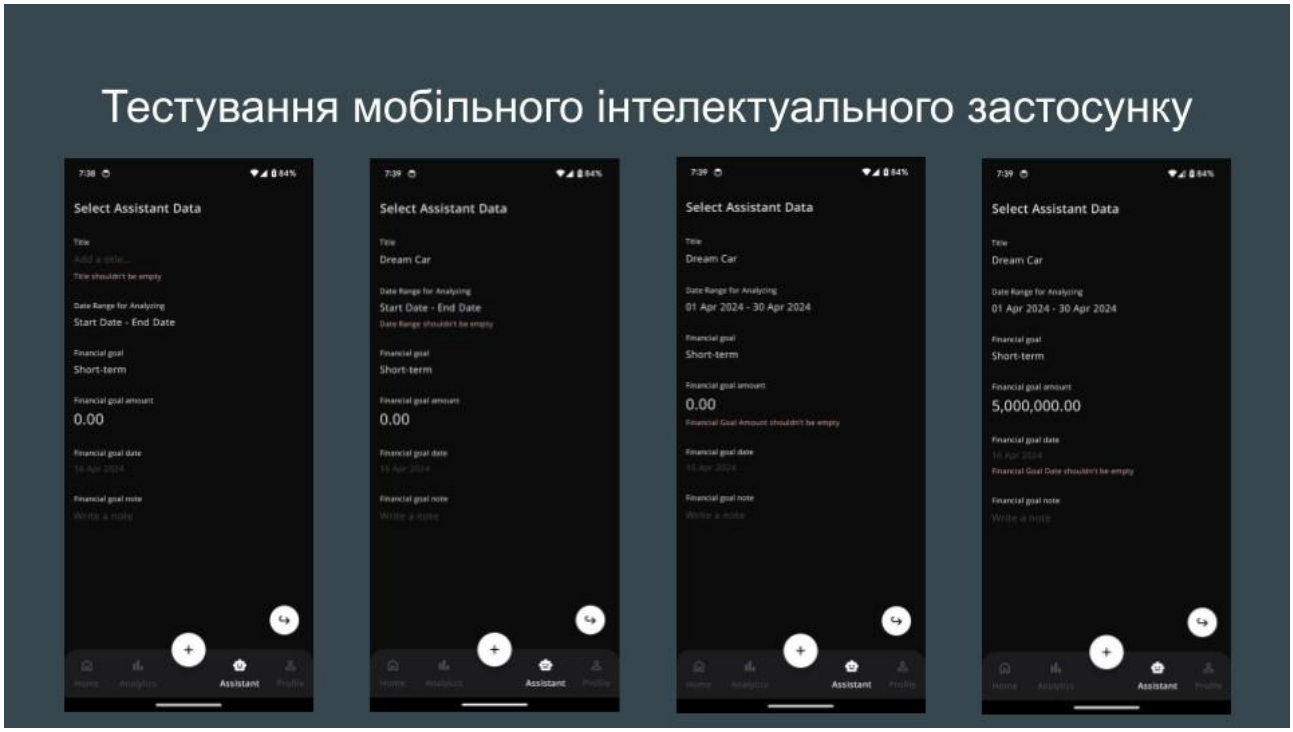

Рисунок Г.20 – Тестування мобільного інтелектуального застосунку (Екран створення транзакції)

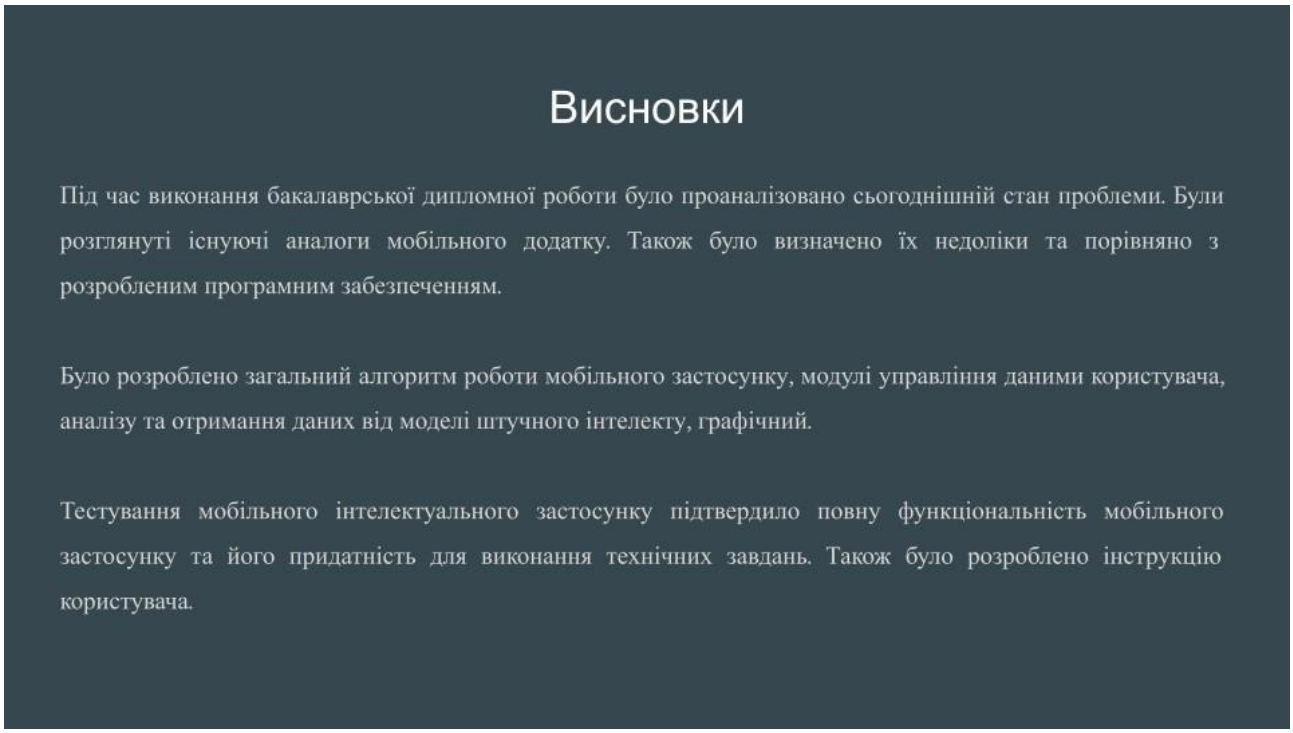

Рисунок Г.21 – Висновки

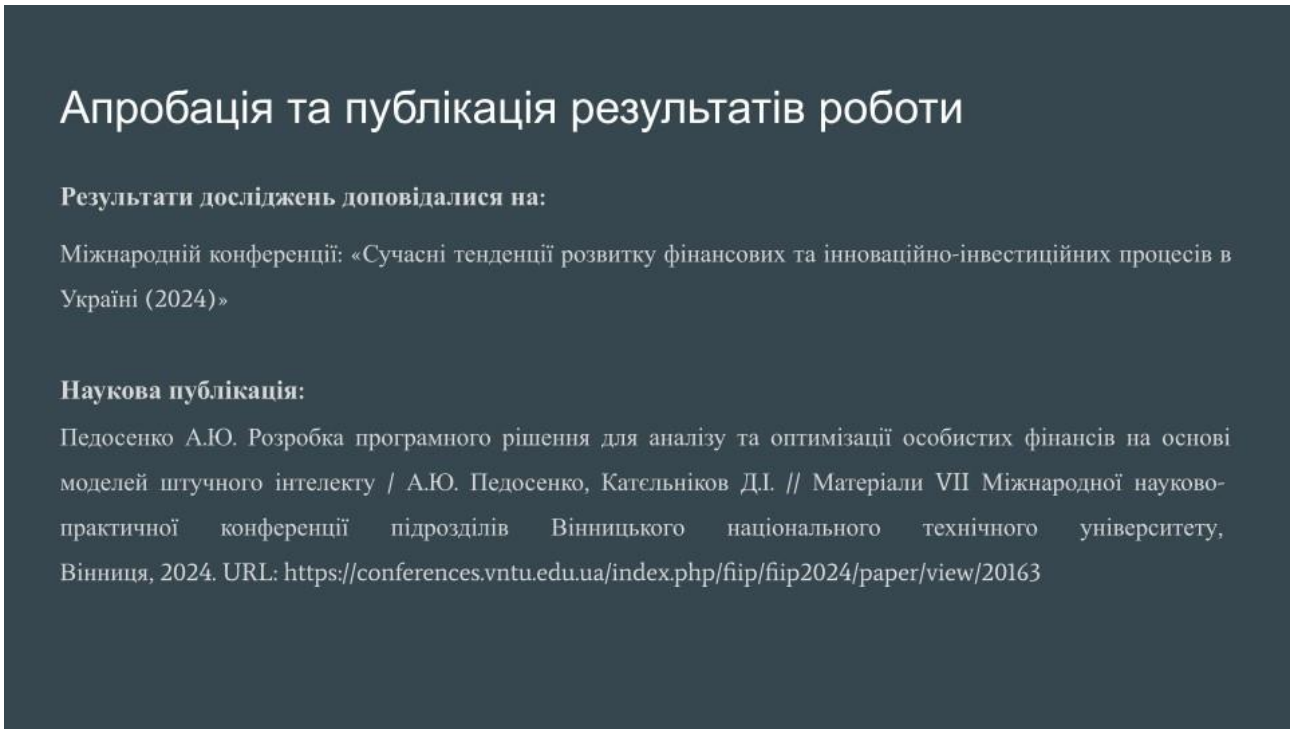

Рисунок Г.22 – Апробація та публікація результатів роботи

Рисунок Г.23 – Фінальний слайд