МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Катерина НЕСТЕРЕНКО
"____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: "Застосунок для фінансового планування, аналізу та оптимізації витрат"

Виконавець: Бойчук Олександр Анатолійович

Керівник: к.т.н. Гордієвський Олексій Тихонович

Нормоконтролер: ст.в Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Бойчука Олександра Анатолійовича

1. Тема кваліфікаційної роботи: «Застосунок для фінансового планування, аналізу та оптимізації витрат» затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2023 р. по 31.12.2023 р.

3. Вихідні данні до проекту: програмний продукт у вигляді мобільного додатку, розроблений за допомогою використання платформи Xamarin.

4. Зміст пояснювальної записки:
   1. Дослідження теми фінансового планування, аналізу та оптимізації витрат.
   2. Визначення вимог до програмного засобу.
   3. Структура мобільного застосунку для фінансового планування, аналізу та оптимізації витрат.
   4. Прототип мобільного застосунку для фінансового планування, аналізу та оптимізації витрат.

5. Перелік обов'язкових слайдів презентації:
   1. Проведення дослідження та порівняння програм-аналогів.
   2. Актуальність розробки застосунку та його цілі.
   3. Функціональні можливості застосунку.
   4. Архітектурне проектування мобільного застосунку.
   5. Структура проекту.
   6. Інтерфейс прототипу мобільного застосунку.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка та затвердження графіку роботи дипломного проектування.	02.10.23 – 08.10.23	
2.	Підготовка та написання 1 розділу, представлення керівнику.	09.10.23 – 22.10.23	
3.	Підготовка та написання 2 розділу, представлення керівнику.	23.10.23 – 05.11.23	
4.	Підготовка та написання 3 розділу, представлення керівнику.	06.11.23 – 19.11.23	
5.	Підготовка та написання 4 розділу, представлення керівнику.	20.11.23 – 03.12.23	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу.	04.12.23 – 10.12.23	
7.	Відсилка ПЗ для перевірки на плагіат.	11.12.23	
8.	Проходження нормо-контролю, перепліт пояснювальної записки. Отримання відгуку керівника. Підготовка презентації та тексту доповіді.	12.12.23 – 16.12.23	
9.	Отримання рецензії.	17.12.23 – 19.12.23	
10.	Передзахист роботи (наявність віддрукованої ПЗ, презентації, позитивного відгуку керівника).	20.12.23	
11.	Підготовка документів до захисту та здача їх секретарю ДЕК (ПЗ, ГМ, CD-R з електронними копіями ПЗ та ГМ, презентація, відгук керівника, рецензія, довідка про успішність, 2 папки, 2 конверта)	21.12.23 – 22.12.23	
12.	Захист дипломної роботи перед ЕК.	27.12.23	

7. Дата видачі завдання 02.10.2023 р.

Керівник: к.т.н. Олексій ГОРДІЄВСЬКИЙ

Завдання прийняв до виконання: Олександр БОЙЧУК

Дата

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Застосунок для фінансового планування, аналізу та оптимізації витрат»: 84 сторінок, 25 рисунків, 4 таблиці, 26 використаних джерел, 8 додатків.

МОБІЛЬНИЙ ЗАСТОСУНОК, ПРОГРАМНИЙ ЗАСІБ, ФІНАНСОВЕ ПЛАНУВАННЯ, ТРАНЗАКЦІЇ, СИНХРОНІЗАЦІЯ, ДОХОДИ ТА ВИТРАТИ, ЛОКАЛІЗАЦІЯ

Об’єкт розробки – мобільний застосунок для фінансового планування, аналізу та оптимізації витрат.

Мета дипломної роботи – допомога користувачам в аналізі доходів та витрат та ефективному використанні коштів, створення фінансового плану для полегшення розпорядження грошовими збереженнями, надання зручного та ефективного інструменту для керування їхніми фінансами та забезпечення більшої фінансової стабільності.

Метод дослідження – проведення опитувань серед потенційних користувачів, що надасть важливий інсайт щодо їхніх потреб, проблем та очікувань від застосунку, та вивчення та аналіз існуючих застосунків для фінансового планування для глибшого розуміння їхніх сильних та слабких сторін, а також виявлення прогалин, які можна заповнити в розроблюваному продукті. Комбінація цих методів дозволить зібрати різноманітні дані та оцінити потреби користувачів, щоб створити більш ефективний та користувацько-орієнтований застосунок для фінансового планування та аналізу витрат.

Результати роботи можуть бути використані для різноманітних цілей, починаючи від удосконалення власного створюваного продукту до створення ефективних маркетингових стратегій.

Розробка та дослідження проводилися під управлінням ОС Windows 10. Розробка програми проводилася в інтегрованому середовищі розробки Visual Studio на мові програмування C#.

ABSTRACT

Explanatory note to the thesis "Application for financial planning, analysis and cost optimisation": 84 p., 25 fig., 4 tables, 26 information sources, 8 appendices.

MOBILE APPLICATION, SOFTWARE, FINANCIAL PLANNING, TRANSACTIONS, SYNCHRONIZATION, INCOME AND EXPENSES, LOCALISATION

The **object of development** is a mobile application for financial planning, analysis and cost optimisation.

The **purpose of the thesis** is to help users analyse income and expenses and use funds efficiently, create a financial plan to facilitate the management of their savings, provide a convenient and effective tool for managing their finances and ensure greater financial stability.

The **research method** will include conducting surveys among potential users, which will provide important insights into their needs, concerns and expectations of the application, and studying and analysing existing financial planning applications to gain a deeper understanding of their strengths and weaknesses, as well as identifying gaps that can be filled in the product under development. The combination of these methods will allow us to collect a variety of data and assess user needs to create a more efficient and user-centric financial planning and cost analysis application.

The **results of the work** can be used for a variety of purposes, ranging from improving your own product to creating effective marketing strategies.

Development and research were carried out under the control of Windows 10. The program was developed in the Visual Studio integrated development environment in the C# programming language.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 14
ВСТУП 15
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТЕМИ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ 19
1.1 Анкетування та опитування 19
1.2 Порівняння програм-аналогів 25
1.2.1 Money Lover 30
1.2.2 Monefy 30
1.2.3 Expense Manage 31
1.3 Опис проблеми, що підлягає рішенню 33
1.4 Перелік вимог на основі аналізу програм-аналогів 33
Висновки 36
РОЗДІЛ 2. ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАСОБУ 37
2.1 Методологія розробки застосунку 37
2.2 Дослідження поширених практик інженерії вимог для розробки програмних засобів 38
2.2.1 Функціональні вимоги 39
2.2.2 Нефункціональні вимоги 43
2.2.3 Бізнес-вимоги 44
2.2.4 Вимоги користувачів 45
2.2.5 Системні вимоги 48
Висновки 49
РОЗДІЛ 3. СТРУКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ 50
3.1 Вибір платформи та інструментів для розробки програмного застосунку 50
3.2 Програмна архітектура застосунку 55
3.3 Класи та взаємодія між ними 58

3.4 База даних та зв’язки між сутностями 61
3.5 Безпека даних 69
Висновки 69
РОЗДІЛ 4. ПРОТОТИП МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ 71
4.1 Реєстрація 71
4.2 Створення гаманця 73
4.3 Керування транзакціями 76
4.4 Аналіз доходів та витрат 77
4.4 Формування фінансового плану 80
Висновки 83
ВИСНОВКИ 84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87
ДОДАТКИ **ОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.**

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

CSV – comma-separated values

БД – база даних.

СУБД – система керування базами даних.

API – application programming interface.

IDEF – integrated definition.

LTE – long term evolution.

SDK – software development kit.

ОС – операційна система.

HTML – hypertext markup language.

CSS – cascading style sheets.

UI – user interface.

ISO - international organization for standardization.

IDE – integrated development environment.

CRUD – create, read, update, delete.

MVP – minimal viable product.

ВСТУП

Фінансове планування та аналіз витрат – ключові аспекти успішного управління фінансами, які набувають особливого значення в сучасному світі, що переживає швидкі та складні зміни у економічній сфері. Поєднання високих технологій та постійної потреби в ефективному управлінні фінансами створює нові виклики, які вимагають новаторських рішень.

В умовах постійних змін у суспільстві, зростання економічної нестабільності та збільшення усвідомленості про важливість фінансової грамотності, потреба у зручних та ефективних інструментах для управління особистими фінансами стає критичною. Зі зростанням обсягу доступної інформації про особисті фінанси та їхнє управління, з'являється необхідність розробки нових методів обробки та аналізу цих даних для максимальної користі для кінцевого користувача.

На сьогоднішній день існує безліч різноманітних програм та засобів для відстеження особистих фінансів. Однак, багато з них мають обмежені можливості або не відповідають потребам користувачів у повному обсязі. Існує потреба у більшій гнучкості, зручності та ефективності управління фінансами, що стає каталізатором для розвитку нових рішень. Тому доцільність та актуальність даної роботи полягають у розробці та впровадженні застосунку, який надасть користувачам можливість не лише відстежувати свої витрати, а й ефективно планувати їхнє використання відповідно до поставлених цілей.

Застосунок - це користувацька програма, що дає змогу вирішувати конкретні прикладні задачі користувача. Поняття введене, щоб підкреслити відмінність від операційної системи, драйверів, бібліотек тощо та засобів і середовищ розробки [1].

Проведений аналіз існуючих рішень в області фінансового планування та аналізу витрат показує необхідність створення комплексного застосунку, що поєднує у собі зручний інтерфейс, функціональні можливості для детального аналізу та можливість гнучкого налаштування для потреб різноманітних категорій користувачів.

Ця магістерська робота присвячена розробці високоефективного застосунку, що сприятиме вдосконаленню фінансового планування та аналізу витрат. Її

значущість полягає в тому, що створення інструменту, який допоможе людям свідомо та ефективно керувати своїми фінансами, має потенціал позитивно вплинути на їх фінансовий стан та сприяти розвитку суспільства в цілому.

Отже, основна мета роботи – розробити та реалізувати застосунок для фінансового планування, аналізу та оптимізації витрат, що буде забезпечувати користувачів зручним та ефективним інструментом для керування своїми фінансами, допоможе краще управляти своїми фінансами, надаючи можливості для планування, аналізу та оптимізації витрат.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Проаналізувати потреби користувачів. Провести дослідження для визначення потреб та очікувань користувачів щодо функціоналу програми та її можливостей.
- Сформувати вимоги до застосунку. Обрати методологію розробки продукту та визначити вимоги до різних аспектів застосунку.
- Визначити структуру застосунку. Дослідити техніки та інструменти розробки мобільних застосунків, визначити архітектуру програми, ключові класи та структуру бази даних.
- Розробити функціонал застосунку. Створити програму з основними функціями відстеження витрат, планування бюджету, аналізу статистики витрат та зручним й ергономічний інтерфейсом.

Об'єкт дослідження роботи включає в себе широкий спектр аспектів фінансового управління, зокрема, процеси фінансового планування, контролю витрат, управління особистими фінансами та їх аналіз. Предметом дослідження є конкретна розробка програмного забезпечення, спрямованого на полегшення цих процесів для користувачів. Це включає в себе створення такого інструменту, який допоможе користувачам відстежувати свої витрати, створювати плани витрат та отримувати детальний аналіз своєї фінансової діяльності. Предмет дослідження також включає вивчення ефективних методів управління фінансами та аналізу їх стану за допомогою розробленого програмного рішення. Такий підхід дозволяє не лише зробити фінансове планування більш доступним та зручним, а й сприятиме

підвищенню фінансової грамотності та усвідомленості управління витратами серед користувачів.

При дослідженні заданої теми планується використовувати різноманітні методи для отримання інформації, аналізу даних та вибору оптимальних рішень, а саме:

- Анкетування та опитування. Збір інформації від користувачів щодо їхніх потреб у фінансовому плануванні та вимог до програмного забезпечення.
- Порівняльний аналіз з вже існуючими рішеннями. Оцінка конкурентної переваги розробленого застосунку порівняно з існуючими програмами для фінансового планування.

Кожен з цих методів зробить цінний внесок у дослідження з метою покращення програмного забезпечення для фінансового планування та аналізу витрат.

Отримані результати дослідження мають наукову новизну через комплексний підхід до фінансового управління, який поєднує в собі різні аспекти планування, витрат та аналізу, надаючи користувачам можливість ефективно керувати своїми фінансами. Цей застосунок вирізняється можливістю персоналізації та гнучкого налаштування під індивідуальні потреби, а також здатністю аналізувати фінансові дані в реальному часі, що дає змогу швидко реагувати на зміни та приймати обґрунтовані рішення. Крім того, застосунок може надавати рекомендації на основі аналізу даних, що сприяє покращенню фінансової грамотності та допомагає у прийнятті кращих фінансових рішень, що робить його цінним інструментом для ефективного фінансового управління у сучасному світі.

Отримані результати дослідження також мають низку практичних застосувань, спрямованих на поліпшення фінансового управління та планування особистих фінансів:

- Фінансове планування. Розроблений застосунок дозволить користувачам створювати персоналізовані плани бюджету на підставі аналізу власних фінансів, що допоможе краще контролювати витрати та досягати фінансових цілей.

- Аналіз витрат. Програма забезпечить можливість детального аналізу витрат за різними категоріями, що надасть користувачам усвідомлення своєї фінансової поведінки та можливості оптимізувати розподіл коштів.
- Ефективне управління фінансами. Відстеження реального стану фінансів дозволить швидко реагувати на зміни та приймати обґрунтовані рішення.
- Рекомендації та аналітика. На основі аналізу даних програма може надавати користувачам рекомендації щодо оптимізації витрат та здійснювати аналітику, яка допоможе зрозуміти, як краще розподіляти кошти для досягнення поставлених цілей.
- Підвищення фінансової грамотності. Використання цього застосунку сприятиме підвищенню фінансової грамотності користувачів, оскільки він надасть можливість краще розуміти свої фінансові потреби та планувати майбутні витрати.

Тож, ця магістерська робота спрямована на створення програмного забезпечення для ефективного фінансового планування та аналізу витрат. Вона покликана вирішити проблему доступності та зручності управління фінансами, надаючи користувачам інструмент, що допоможе відстежувати, планувати та аналізувати свої витрати. Результати цього дослідження пропонують новий підхід до фінансового управління, забезпечуючи можливості адаптації до різноманітних фінансових потреб користувачів. Це дослідження відкриває перспективи для покращення фінансової грамотності та сприяє ефективному управлінню особистими фінансами в умовах сучасного світу.

РОЗДІЛ 1.
ДОСЛІДЖЕННЯ ТЕМИ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ

Під час дослідження теми планується використовувати різні методи для отримання інформації, аналізу даних та вибору оптимальних рішень. Ці методи включають анкетування та опитування для збору від користувачів інформації про їхні потреби у фінансовому плануванні та вимоги до програмного забезпечення [2, 6]. Також передбачено порівняльний аналіз із вже існуючими рішеннями для оцінки конкурентної переваги розробленого застосунку порівняно з іншими програмами для фінансового планування. Кожен з цих методів має сприяти вдосконаленню програмного забезпечення для фінансового планування та аналізу витрат.

1.1 Анкетування та опитування

Для здійснення дослідження з теми дипломної роботи "Застосунок для фінансового планування, аналізу та оптимізації витрат" було вибрано метод анкетування та опитування учасників. Цей метод визначається як ефективний інструмент для отримання кількісних та якісних даних від різних груп людей.

Однією з переваг використання анкет та опитувань є можливість отримання широкого обсягу інформації від різних учасників, представляючи різноманітні погляди та досвід. Це сприяє формуванню комплексного уявлення про вимоги та очікування цільової аудиторії від запланованого застосунку. Анкетування також дозволяє збирати кількісні дані, що сприяє об'єктивності та статистичному аналізу отриманих результатів.

Окрім того, використання анкет дозволяє легко стандартизувати процес збору інформації, зменшуючи можливі помилки та сприяючи однаковості підходу до усіх учасників. Завдяки цьому методові, можна отримати відповіді на конкретні питання, а також відкриті коментарі, що дозволяє отримати глибші уявлення про вимоги користувачів. Такий підхід був обраний, оскільки він забезпечує комплексність та репрезентативність отриманих даних, необхідних для подальшого успішного впровадження та розвитку застосунку.

Для проведення анкетування з розробки застосунку було обрано Google Forms [3, 4]. Безкоштовність платформи робить її доступною для широкого кола користувачів, забезпечуючи максимальну участь учасників незалежно від їхнього фінансового стану. Легкий у використанні інтерфейс Google Forms робить анкетування зручним та ефективним для різних категорій учасників, включаючи тих, хто може мати обмежену технічну підготовку.

Інтеграція Google Forms з Google Sheets є ще однією перевагою, яка спрощує процес збору та аналізу даних. Отримані відповіді автоматично переносяться до таблиць Google Sheets, де їх легко обробити та проаналізувати. Такий підхід сприяє організованому та систематичному аналізу результатів.

Крім того, Google Forms дозволяє створювати адаптивні анкети, що може бути важливо для врахування різних потреб та умов користувачів. Загальною метою вибору цієї платформи є забезпечення найбільшого зручного та ефективного збору інформації в рамках конкретного дослідження.

У результаті проведення дослідження було створено Google Forms (рисунок 1.1), яке служило інструментом для збору відповідей та даних в рамках дипломної роботи [5].

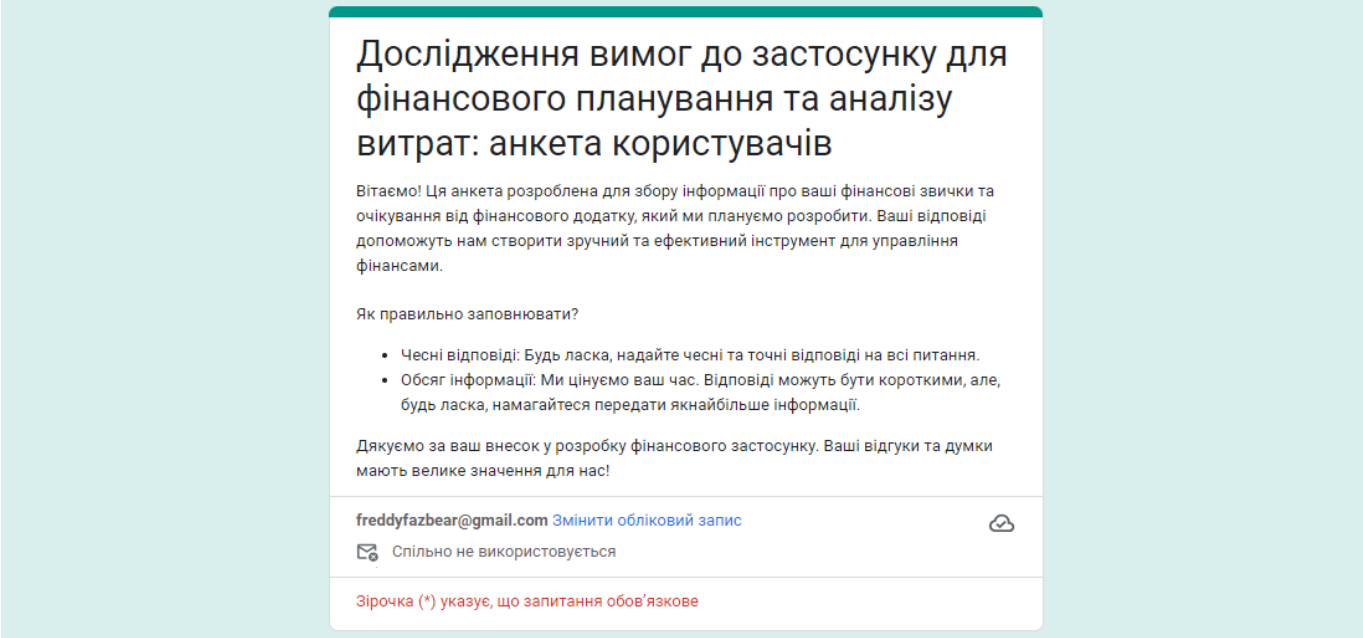

Рис. 1.1 – Анкета користувачів в Google Forms

Участь у опитуванні взяло 134 людей, що надало значний обсяг вхідної інформації. Анкета була поширена серед користувачів на популярних соціальних мережах, що забезпечило широкий охоплення та різноманітність відповідей. Цей метод розповсюдження анкети сприяв отриманню репрезентативних та різноманітних відгуків від різних груп користувачів, що забезпечить об'єктивність та різносторонність результатів.

Для анкетування було сформовано 2 групи питань. Перша група питань спрямована на збір даних про потенційних користувачів застосунку для фінансового планування, аналізу та оптимізації витрат (рис. 1.2), що є важливим аспектом у розвитку та адаптації програми до їхніх потреб. Ця інформація допоможе краще зрозуміти аудиторію застосунку, включаючи вік, статус.

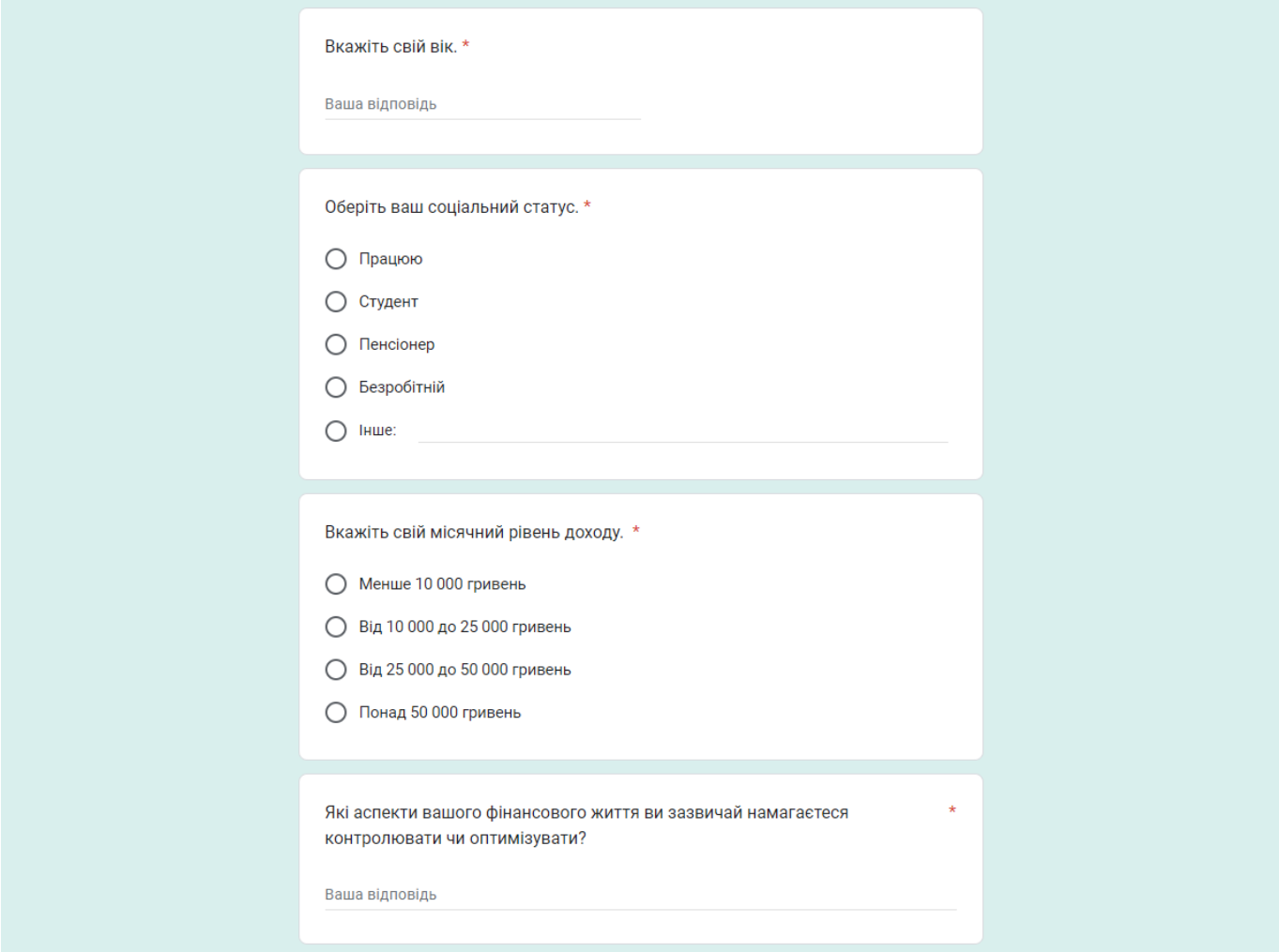

Рис. 1.2 – Перша частина питань з анкети користувачів

Надані особисті дані дадуть можливість персоналізувати сервіс, створюючи індивідуальні пропозиції та функції для кожного користувача, допоможуть у підтримці та розвитку продукту, аналізуючи та вдосконалюючи функціонал застосунку відповідно до потреб користувачів. Така інформація може також бути використана в майбутньому для здійснення маркетингу та реклами, дозволяючи краще спрямовувати рекламні кампанії на відповідну аудиторію.

Друга група питань була спрямована на збір даних користувачів про досвід користування подібними застосунками (рис. 1.3). що є важливим для поліпшення власного продукту. Ця інформація надає можливість зрозуміти, що працює або не працює у конкурентів, дозволяючи покращити функціонал власного застосунку.

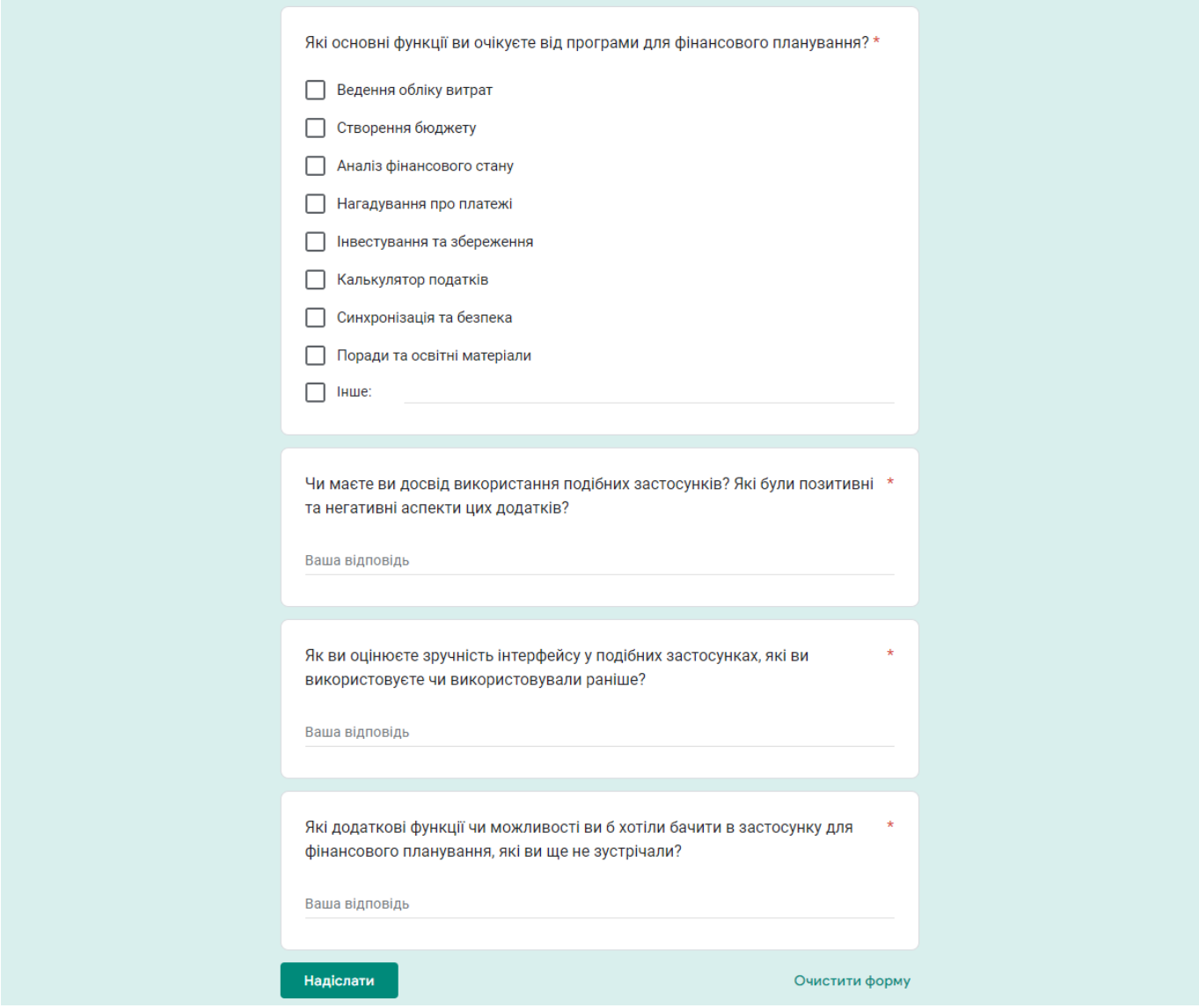

Рис. 1.3 – Друга частина питань з анкети користувачів

Аналіз досвіду користування іншими застосунками допомагає визначити сильні та слабкі сторони конкурентів, що дозволяє вдосконалити свій продукт. Отримані дані про досвід користування подібними застосунками також надають уявлення про реальні потреби користувачів та можуть стати основою для створення більш привабливого та корисного застосунку. Враховуючи досвід користування іншими програмами, можна створити інноваційні рішення та покращення, що зроблять продукт більш конкурентоспроможним та привабливим для потенційних користувачів.

Аналіз отриманих результатів анкетування представляє собою важливий та складний процес, що вимагає вдумливого та систематичного підходу. Цей етап дослідження є ключовим для отримання значущих висновків та розуміння основних тенденцій, виявлених відповідями учасників. Аналіз забезпечує можливість глибше розкрити контекст фінансових поглядів та практик, що визначаються досліджуваною темою [6, 7].

Для здійснення аналізу використовуються різноманітні методи, серед яких:

- Статистичний аналіз: Використання статистичних методів, такі як середнє значення, медіана, мода та стандартне відхилення, для оцінки центральних та розподільчих характеристик даних. Це дозволяє здійснити загальний аналіз розподілу відповідей та виявити загальні тенденції.
- Сегментація груп: Розподілення учасників на групи за певними критеріями, такими як вік, дохід, освіта тощо. Порівняння відповідей між різними сегментами, щоб виявити варіації у фінансових підходах різних груп.
- Кореляційний аналіз: Визначення взаємозв'язків між різними показниками, такими як рівень доходу та розподіл витрат. Кореляційний аналіз дозволяє визначити, чи існують статистично значущі зв'язки між різними фінансовими аспектами.
- Кластерний аналіз: Використання алгоритмів кластеризації для групування учасників за подібними характеристиками. Це може допомогти виділити спільні моделі поведінки та фінансові стратегії в різних групах.

- Аналіз відкритих відповідей: Ретельне дослідження відкритих коментарі та відповіді учасників для отримання контексту та глибини розуміння їхніх фінансових поглядів та витрат.
- Порівняння з результатами інших досліджень: Порівняння отриманих результати з результатами схожих досліджень та наукових публікацій для отримання більш широкого контексту та порівняльного аналізу.

Для аналізу результатів моєї анкети було використано методи "сегментація груп", "кореляційний аналіз" та "аналіз відкритих відповідей", оскільки ці методи відповідають характеру та типу питань, включених в анкету.

Після аналізу відповідей у анкетах були сформовані стандартні ліміти для кожної категорії витрат, враховуючи вік та рівень доходу. Ці дані передбачено використовувати в застосунку: коли користувач обирає свій вік та дохід під час реєстрації, система пропонуватиме стандартні ліміти витрат для різних категорій. Зазначимо, що користувач матиме можливість коригувати ці ліміти згідно власних уподобань та потреб. Цей підхід призначений для надання користувачам зручного початкового планування витрат, яке може бути далі персоналізовано відповідно до їхньої фінансової ситуації.

Виявлено, що користувачі вважають три основні функції найбільш важливими для фінансового застосунку: відстеження витрат, створення бюджетів та попередження про ліміти. Ці висновки свідчать про те, що основний підхід до фінансового планування полягає в систематичному контролі за витратами, раціональному розподілі коштів та уникненні перебільшених витрат.

Тому розробка нашого фінансового застосунку буде обов’язково включати реалізацію цих важливих функцій. Відстеження витрат дозволить користувачам детально аналізувати свої фінансові витрати, щоб зрозуміти, куди йдуть їхні гроші. Створення бюджетів дозволить встановлювати ліміти витрат на різні категорії, тим самим допомагаючи ефективно розподіляти фінансові ресурси. А функція попередження про ліміти стане надійним помічником у відборі відповідального підходу до фінансового планування, сприяючи уникненню перевищення встановлених граничних значень. Такий інтегрований підхід дозволить нашому

застосунку найкращим чином відповісти на потреби та очікування користувачів у сфері фінансового управління.

В даному застосунку буде приділено особливу увагу виправленню зазначених недоліків, які були виявлені в результатах анкетах. Зокрема:

- Спрощений інтерфейс: застосунок з простим та зрозумілим інтерфейсом, що дозволить користувачам легко орієнтуватися та використовувати всі функції без зайвих ускладнень.
- Робота в офлайн-режимі: застосунок буде максимально адаптованим до використання в офлайн-режимі, що зменшить залежність від постійного інтернет-з'єднання та забезпечить неперервний доступ до фінансових даних.
- Повна синхронізація з українськими банками: застосунок буде підтримувати повну синхронізацію транзакцій із найбільш популярними українськими банками, що дозволить користувачам з легкістю відстежувати історію транзакцій без необхідності вводити дані вручну.

Використання методу анкетування стало важливим етапом у даному дослідженні, дозволяючи опрацювати обширний обсяг даних та глибше зрозуміти потреби та очікування потенційних користувачів. Завдяки цьому методу було отримано вимоги до застосунку для фінансового планування, а також конкретні аспекти, які вони вважають важливими у таких інструментах. Анкетування стало ефективним інструментом для формування стратегії розробки застосунку, який буде максимально відповідати потребам та очікуванням користувачів.

1.2 Порівняння програм-аналогів

Метод порівняльного аналізу з існуючими рішеннями полягає в систематичному огляді та порівнянні різних рішень, наявних на ринку, з метою ідентифікації їхніх переваг, недоліків та особливостей. Застосування порівняльного аналізу дозволяє краще розуміти сильні та слабкі сторони існуючих рішень, а також визначити унікальні аспекти, які можна вдосконалити чи впровадити в застосунок, що розроблюється. Цей підхід сприяє створенню більш конкурентоспроможного та

ефективного продукту, спрямованого на задоволення потреб користувачів у сфері фінансового планування.

Серед великої кількості аналогічних або схожих застосунків для фінансового планування та оптимізації витрат було обрано наступні популярні рішення для детального аналізу:

- Money Lover
- Monefy
- Expense Manager

Ці застосунки визначаються своєю популярністю та широким спектром функціоналу, що зробило їх ідеальними об'єктами для порівняльного аналізу та виокремлення ключових характеристик, які можуть бути враховані при розробці нового застосунку.

Для подальшого розгляду та виявлення недоліків аналогів, необхідно розглянути основні поняття, які вони використовують:

- Гаманець – є обліковим рахунком, що визначає для користувача конкретне джерело доходу чи витрат. Зазвичай гаманці можуть бути класифіковані на два типи: звичайні та пов'язані з банківською карткою.
- Транзакція (операція) – це зміна поточного балансу гаманця. Ця подія може бути пов'язана із заробітком або витратою коштів.
- Синхронізація транзакцій – це отримання та збереження в гаманці інформації про транзакції з банківської картки за певний період часу.
- Категорія – це групування подібних джерел доходу або схожих типів витрат для легшого управління та аналізу.
- Звіт – представляє собою статистичну інформацію про витрати та доходи за певний проміжок часу.
- Фінансовий план – визначає місячне планування доходів за різними категоріями.

Порівняння властивостей програм-аналогів наведено у вигляді таблиці 1.1.

Таблиця 1.1

Порівняння існуючих програм-аналогів

Властивість програмного засобу	**Money Lover**	**Monefy**	**Expense Manager**
Адаптивність інтерфейсу	Інтерфейс автоматично пристосовується до розмірів екранів смартфонів та планшетів	Інтерфейс автоматично пристосовується до розмірів екранів смартфонів та планшетів	Погана підтримка планшетів
Чистота інтерфейсу	Значна кількість спам-реклам	Відсутність реклам	Відсутність реклам
Налаштування гаманців	Присутнє, але лише один гаманець безкоштовний	Повністю безкоштовна функція	Повністю безкоштовна функція
Налаштування категорій	Створення власних категорій наявне лише в платній версії застосунка	Надається можливість створювати категорії та редагувати їх відображення	Надається можливість створювати категорії та редагувати їх відображення
Експорт даних	-	Наявна підтримка вивантаження даних у файл з форматом CSV	Наявна підтримка вивантаження даних у файл з форматом Excel
Імпорт даних	-	-	-

Продовження табл. 1.1

Властивість програмного засобу	**Money Lover**	**Monefy**	**Expense Manager**
Прив'язка до кредитних карток	Прив'язувати кредитну картку можна лише в платній версії. Відсутня підтримка українських банків	-	-
Фінансове планування	Дана функція несе лише інформативний характер та ніяк не сповіщає чи контролює користувача про досягнення ліміту	-	-
Безпека	Для отримання транзакцій з банківської картки використовується зовнішнє API банку, доступ до якого забезпечується за допомогою користувацького ключа.	Не зберігає дані про банківські картки, оскільки відсутня функція прив'язки кредитної картки до гаманця	Не зберігає дані про банківські картки, оскільки відсутня функція прив'язки кредитної картки до гаманця

Продовження табл. 1.1

Властивість програмного засобу	**Monefy**	**Money Lover**	**Expense Manager**
Звітність	Застосунок підтримує різні типи звітності	Функція звітності присутня, але її можливості обмежені, і вона не надає деталізованої інформації для аналізу транзакцій за визначений період часу.	-
Синхронізація даних із Cloud	-	Синхронізація через файловий обмінник Dropbox доступний лише в платній версії	-
Пароль	-	-	Наявна функція встановлення паролю для входу у вигляді графічного ключа або PIN коду
Вартість	Деякі функції є безкоштовними, проте більшість функціоналу потребує підписки	Велика частина функцій є безкоштовними, але наяна підписка для розширених можливостей	Безкоштовний

1.2.1 Money Lover

Money Lover – застосунок, що дозволяє з легкістю керувати та слідкувати за особистими фінансами [8].

Компанія, що розробила: Finsify

Підтримуванні платформи: iOS та Android.

Додаток відзначається наступними перевагами:

- Графіки та діаграми: Money Lover пропонує багатий функціонал для створення звітності по доходам та витратам за обраний період.
- Прив’язка до банків: користувач може прив'язати свій гаманець до банківської картки та синхронізувати транзакції в додаток, що полегшує відстеження фінансових операцій.
- Планування бюджету: застосунок дозволяє користувачам планувати свої місячні витрати, сприяючи більшій контрольні над фінансами.

Незважаючи на ці переваги, Money Lover також має свої недоліки:

- Синхронізація: відсутність можливості синхронізації даних через Cloud може обмежити зручність в роботі з додатком.
- Інтеграція з банками: підтримка інтеграції обмежується лише певними банками, не включаючи українські фінансові установи.
- Планування бюджету: інформативний характер функціоналу з планування бюджету може ускладнити ефективний контроль над витратами.
- Імпорт та експорт даних: дані функції відсутні.
- Налаштування категорій та гаманців: отримання доступу до розширених можливостей, таких як створення власних категорій та керування кількома гаманцями вимагає купівлі підписки.

1.2.2 Monefy

Monefy - фінансовий менеджер для відстеження власних витрат [9].

Компанія, що розробила: Reflectly.

Підтримуванні платформи: iOS, Android.

Переваги:

- Інтуїтивний інтерфейс: для додавання транзакцій користувачам достатньо лише натиснути "+" для доходів або "-" для витрат, вибрати категорію та ввести суму.
- Dropbox синхронізація: ця функція гарантує збереження даних користувача, навіть у випадку втрати або крадіжки телефону.
- Керування категоріями: доступна функція створення власних категорій витрат та доходів, редагування зображення тощо.
- Експорт файлів у CSV форматі: забезпечує можливість вивантаження даних про доходи та витрат у файли, які можуть бути використані сторонніми застосунками.
- Безпека: додаток не вимагає введення банківської інформації, оскільки всі транзакції вводяться вручну.

Серед недоліків можна відзначити наступне:

- Платні функції: безкоштовна версія застосунку пропонує лише стандартний та неповний набір функціоналу, а такі функції як синхронізація, встановлення паролю та створення власних категорій витрат вимагає купівлі підписки на продукт
- Синхронізація транзакцій з банками: дана функція відсутня, тому всі транзакції створюються вручну
- Різниця в функціоналі та інтерфейсі: iOS версія програми має скорочений функціонал на противагу Android
- Робота зі звітністю: даний застосунок не дає змогу переглянути та проаналізувати витрати й доходи
- Фінансове планування: відсутнє
- Імпорт даних: застосунок не передбачає імпорт даних, що може спричинити труднощі при міграції з іншого схожого застосунку

1.2.3 Expense Manage

Expense Manager – особистий фінансовий менеджер та трекер бюджету [10].

Компанія, що розробила: Markus Hintersteiner

Підтримуванні платформи: Android.

Переваги:

- Ліміт: користувач може встановити прийнятну для себе місячну суму витрат, щоб відслідковувати та контролювати свій бюджет
- Пароль: з метою забезпечення безпеки використання додатку, доступ до нього може бути захищений паролем, що забезпечить конфіденційність фінансової інформації.
- Нагадування: для зручності користувача передбачено можливість встановлення щоденних, щомісячних або інших нагадувань, які сприятимуть своєчасному отриманню сповіщень щодо стану фінансів.
- Налаштування категорій: користувачеві можна створювати власні категорії витрат та доходів, дозволяючи більш гнучко налаштовувати відслідковування фінансів
- Експорт: додаток надає можливість вивантажувати дані про доходи та витрати у форматі Excel, що сприяє зручності обробки і аналізу фінансової інформації сторонніми програмами
- Безпека: не вимагає введення банківської інформації, оскільки всі транзакції можуть бути внесені користувачем вручну, що гарантує додатковий рівень конфіденційності

Недоліки:

- Синхронізація: дана функція відсутня, що може ускладнювати обмін інформацією між різними пристроями користувача.
- Імпорт даних: немає можливості імпортувати дані із сторонніх застосунків
- Кросплатформеність: додаток працює лише на Android
- Синхронізація з банківськими картками: дана функція відсутня
- Звітність: функція відсутня
- Фінансове планування: не підтримується

1.3 Опис проблеми, що підлягає рішенню

Проведене анкетування та аналіз програм-аналогів у сфері фінансового планування, аналізу та оптимізації витрат надали інформацію щодо очікувань користувачів від подібного продукту. Отримані відповіді дозволили визначити ключові функції та особливості, які вважаються важливими для ефективного планування та аналізу фінансами.

Згідно з результатами дослідження, було виявлено, що існуючі програми-аналоги хоча і мають схожий функціонал, проте вони обмежені рядом недоліків, які можна врахувати та усунути під час розробки нового застосунку. До цих недоліків належать:

- Неповна реалізація системи контролю витрат, що може ускладнити користувачам ефективне ведення обліку своїх фінансів.
- Відсутність системи сповіщень про відхилення від фінансового плану, що позбавляє користувачів можливості оперативно реагувати на перевищення лімітів на категорії.
- Відсутність рекомендацій щодо розподілу фінансів може ускладнити оптимальне управління бюджетом та раціональне розподілення коштів між різними категоріями витрат.
- Майже відсутня підтримка синхронізації транзакцій з українськими банками стає значущим обмеженням для користувачів з України.

Отже, враховуючи ці аспекти, розробка нового застосунку буде націлена на усунення зазначених недоліків та впровадження високоефективної системи фінансового управління.

1.4 Перелік вимог на основі аналізу програм-аналогів

На основі аналізу отриманих результатів досліджень можна скласти перелік вимог для застосунку фінансового планування, аналізу та оптимізації витрат, враховуючи всі виявлені недоліки і слабкі сторони існуючих програм. Один із ефективних методів формулювання вимог - це використання "user stories" або історій користувача.

User stories - це короткі описи функціональностей додатку, представлені з точки зору користувача. Кожна історія користувача визначає конкретну можливість або функцію, яку користувач хоче отримати від продукту [11].

Переваги використання user stories включають:

- Орієнтація на користувача: user stories допомагають зосередитися на реальних потребах користувачів та їх очікуваннях від продукту.
- Прозорість: цей метод робить вимоги більш зрозумілими для всіх учасників проекту, включаючи розробників, тестувальників та менеджерів.
- Легкість змін: user stories дозволяють легко вносити зміни або доповнювати вимоги в процесі розробки, що особливо важливо в гнучких методологіях розробки, таких як Scrum або Kanban.
- Позначення пріоритетів: вимоги можна легко встановити пріоритет, визначаючи, які функції є найбільш важливими для користувачів та бізнесу.
- Створення єдиної мови: user stories сприяють створенню єдиної мови між командами розробників та іншими учасниками проекту.

З використанням цього методу можна ефективно формалізувати та структурувати вимоги до додатку, що розроблюється, забезпечуючи його високу якість та відповідність потребам користувачів.

Отже, до застосунку було сформульовано наступні вимоги:

- Як користувач, я хочу мати можливість синхронізації транзакцій з моєю банківською карткою, щоб уникнути ручного введення та забезпечити точність фінансового обліку.
- Як користувач, я хочу мати можливість створювати фінансовий план та встановлювати ліміти для кожної категорії витрат, щоб ефективно планувати свої фінанси та уникати перевищення бюджету.
- Як користувач, я хочу отримувати регулярні сповіщення про відхилення або успішність дотримання фінансового плану, щоб завжди бути в курсі своєї фінансової ситуації.

- Як користувач, я хочу мати можливість генерувати звітність за певний період часу для аналізу своїх витрат та доходів, щоб краще розуміти своє фінансове становище.
- Як користувач, я хочу мати можливість імпорту даних з інших фінансових менеджерів, щоб легко перейти до нового додатку без втрати історії своїх фінансів.
- Як користувач, я хочу мати можливість зберігати резервну копію своїх фінансових даних в хмарному сховищі, щоб уникнути втрати інформації в разі втрати або зміни пристрою.
- Як користувач, я хочу мати функцію експорту даних у форматі CSV, щоб легко аналізувати свої доходи та витрати за допомогою зовнішніх інструментів.
- Як користувач, я хочу мати функцію керування регулярними платіжками, такими як рахунки за комунальні послуги чи інтернет, для автоматизації цих операцій та запобігання прострочення платежів.
- Як користувач, я хочу мати можливість встановлювати пароль, біометрію та двох факторну автентифікацію для безпеки входу в додаток, щоб забезпечити захист своїх даних.
- Як користувач, я хочу мати можливість реєстрації через обліковий запис Google, Twitter або Microsoft, щоб швидко та безпечно отримати доступ до функціоналу застосунку
- Як користувач, я хочу отримувати рекомендації та поради щодо оптимізації моїх витрат, щоб ефективніше управляти своїми фінансами.
- Як користувач, я хочу мати можливість класифікувати транзакції за різними категоріями, щоб мати змогу зручно аналізувати доходи та витрати по категоріями.
- Як користувач, я хочу бачити історію своїх транзакцій.
- Як користувач, я хочу мати можливість створювати гаманці з різними типами валют.

Висновки

У першому розділі дипломної роботи було проаналізовано потреби користувачів. Для цього проведено комплексне анкетування з використанням Google Forms, яке дозволило отримати значний обсяг відповідей від потенційних користувачів фінансового додатку. Результати анкетування були аналізовані та оброблені, надаючи унікальну можливість здобути важливі відомості про вимоги та очікування користувачів у сфері фінансового планування та управління витратами.

У процесі аналізу були визначені та порівняні популярні програми-аналоги, такі як Monefy, Money Lover та Expense Manager. Оцінка їхніх функцій та особливостей дала можливість виокремити ключові аспекти, які визначають ефективність та зручність використання фінансових додатків.

В процесі порівняльного аналізу були виявлені недоліки та обмеження кожного з розглянутих програм-аналогів, такі як відсутність певних функцій, обмежена синхронізація транзакцій, проблеми з інтеграцією з українськими банками та інші. Ці недоліки було використано для формування конкретних вимог до застосунку, що розроблюється.

На основі аналізу результатів дослідження та ідентифікації потреб користувачів було сформовано перелік вимог у вигляді user stories. Ці вимоги стануть основою для подальшого процесу проектування та розробки, спрямованого на створення продукту, що повністю відповідає очікуванням та потребам користувачів.

РОЗДІЛ 2.
ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАСОБУ

2.1 Методологія розробки застосунку

При розробці програмного продукту важливо усвідомлювати, що він проходить через свій життєвий цикл, який включає детальний план розробки, тестування та поліпшення продукту. Цей план можна розглядати як послідовні фази, кожна з яких стає вихідними даними для подальших етапів [12, 13]. Життєвий цикл складається з різних фаз (рис. 2.1).

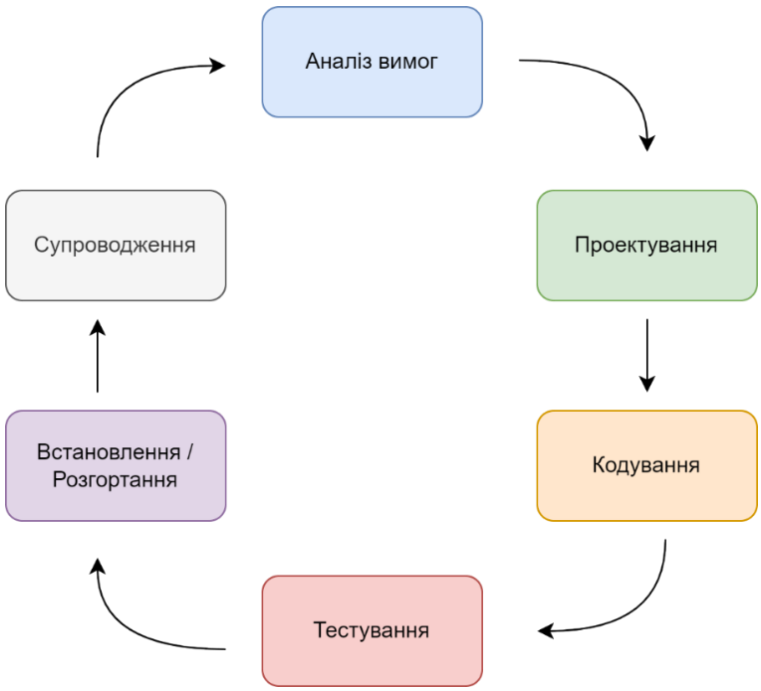

Рис. 2.1 – Фази життєвого циклу ПЗ

Вибір моделі життєвого циклу напряму впливає на методологію розробки програмного забезпечення. Серед популярних моделей життєвого циклу виділяють наступні:

- Каскадна модель. Ця модель передбачає лінійний підхід до розробки, де кожен етап послідовно виконується, починаючи з аналізу та завершуючи тестуванням.
- Спіральна модель. Заснована на поєднанні елементів каскадної моделі та елементів ітераційного планування. Розробка відбувається поступово, з кожним циклом розширюючи функціональність.

- Інкрементальна модель. Розробка програми розбивається на невеликі частини, а кожна нова частина додається поступово, розвиваючи продукт.
- Ітераційна модель. Робота проводиться через короткі ітерації або цикли, де кожен цикл може включати етапи розробки від аналізу до тестування.

Для розробки застосунку для фінансового планування, аналізу та оптимізації витрат найбільш відповідною є ітераційна модель. Основна ідея цієї моделі полягає у розділенні проекту на умовні частини, що називаються ітераціями. Кожна ітерація включає всі етапи життєвого циклу, що дає змогу отримувати функціональний продукт на кожному етапі розробки. Таким чином, в кінці кожного етапу отримуємо працюючий продукт із певними обмеженнями, які піддаються оптимізації та розвитку в наступних ітераціях.

Порівняно з іншими даний підхід є вигідним, оскільки забезпечує захист від невизначеностей та помилок на початкових етапах. Кожна ітерація надає функціональний продукт, який можна випробувати та перевірити на правильність виконання. Таким чином, ітераційна модель дозволяє зменшити ризики втрати бюджету та уникнути непорозумінь між замовником та командою розробників.

Виходячи з обраного життєвого циклу програмного забезпечення, Scrum методологія виявляється відповідною для розробки. Ця методологія ґрунтується на понятті спринту, який має часові рамки (від тижня до місяця), протягом яких проводиться розробка продукту. Завдання на кожний спринт утворюють його беклог (sprint backlog), який формується з переліку завдань від замовника (product backlog) під час планування (sprint planning) [14].

2.2 Дослідження поширених практик інженерії вимог для розробки програмних засобів

Розділ інженерії програмного забезпечення, який охоплює збір та аналіз вимог до програмного продукту, визначення основних функцій, алгоритмів та властивостей, що стосуються розробки програмного продукту, має назву інженерія вимог. Крім того, цей розділ також займається перевіркою та документуванням встановлених вимог. Вимоги, у свою чергу, представляють собою висловлення про

характеристики та властивості програмного продукту, які визначаються під час аналізу вимог. Зазвичай вони надходять від замовника продукту, але іноді виникають під час аналізу в самій команді розробників [15, 16].

2.2.1 Функціональні вимоги

Функціональні вимоги визначають функції програмного продукту, які розробники мають реалізувати, щоб кінцеві користувачі змогли досягати своїх цілей за допомогою цього застосунку. Важливо, щоб такі вимоги були зрозумілі як самій команді розробників, так і сторонам, зацікавленим в розроблюваному застосунку. Основна мета функціональних вимог полягає в описі поведінки програмного продукту в конкретних умовах [17].

Функціональні вимоги для застосунку, що розробляється:

- Користувач повинен мати можливість використовувати свій обліковий запис Google, Twitter або Microsoft для безпечного та зручного входу в застосунок. Система повинна надавати можливість обрати між цими двома опціями та використовувати їх для автентифікації користувача..
- Користувач повинен мати можливість додавати транзакції (витрати та доходи) зручним та інтуїтивно зрозумілим способом.
- Застосунок повинен надавати зручні інструменти для генерації звітів щодо витрат та доходів за вказаний період часу
- Застосунок повинен надавати можливість користувачеві створювати та налаштовувати фінансовий план з урахуванням місячних витрат та доходів
- Застосунок повинен підтримувати синхронізацію транзакцій з банківськими картками, забезпечуючи автоматизований запис фінансових операцій до гаманця в застосунку
- Застосунок повинен автоматично класифікувати транзакції за категоріями витрат для полегшення аналізу та створення звітів
- Користувач повинен мати можливість імпортувати та експортувати дані для легкого переходу з інших фінансових додатків або для збереження копій інформації.

- Користувач повинен мати можливість встановлювати ліміти на категорії витрат та отримувати сповіщення при їх перевищенні
- Застосунок повинен автоматично або за вказаною користувачем частотою здійснювати резервне копіювання фінансових даних в хмарних сховищах.
- Користувач повинен мати можливість встановлювати регулярні нагадування про платежі, або інші фінансові події

Нотація графічного моделювання IDEF0 використовується для створення функціональної моделі, що відображає структуру та функції програмного продукту, що розроблюється, включаючи потоки даних та матеріальних об’єктів, що зв’язують ці функції [18].

Початково контекстна діаграма IDEF0 представляє систему одним блоком як цілісну одиницю (рис. 2.2), а потім, у більш деталізованому форматі, показує взаємодію функцій програми між собою (рис. 2.3). Одним з ключових процесів є взаємодія користувача з програмним забезпеченням.

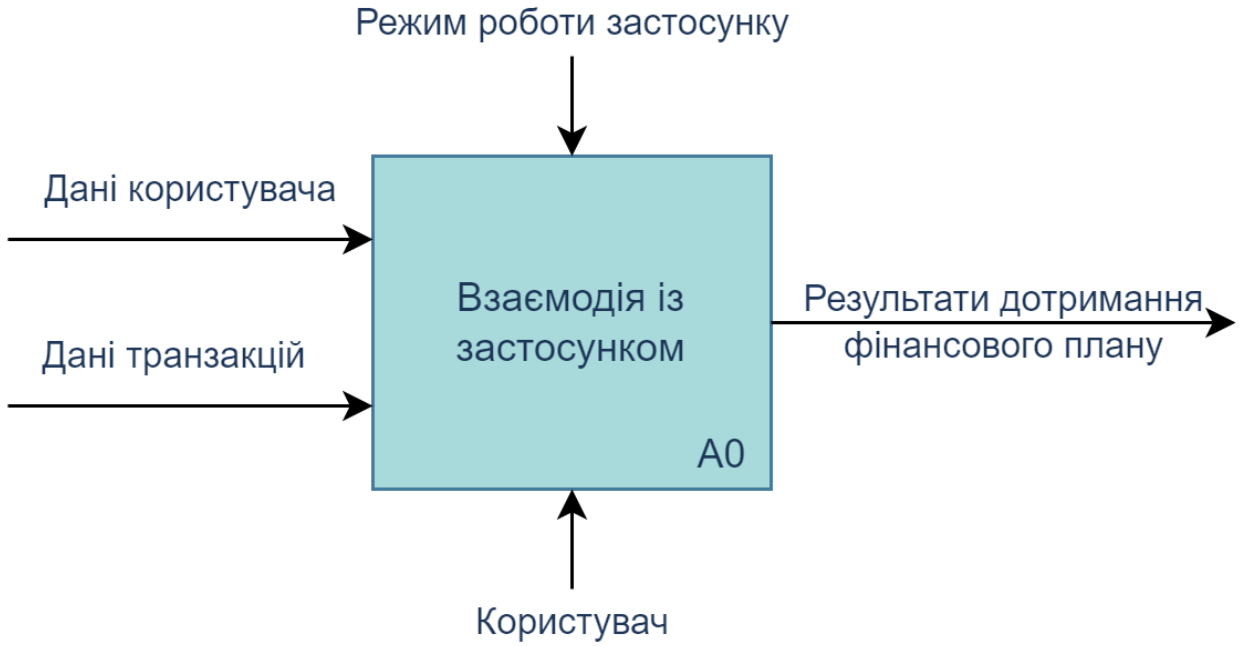

Рис. 2.2 – Контекстна діаграма IDEF0

Ліворуч відображені стрілки відповідають за дані користувачів та транзакцій, що є вхідними даними і важливими елементами для правильного функціонування програмного засобу. Верхній рядок містить обмеження, які впливають на режим роботи застосунка. У нижній частині розташовані люди або обладнання, що

використовуються у даному процесі. Результати виконання функцій програмного продукту, наприклад, досягнення фінансового плану, відображені у вигляді стрілки праворуч.

Діаграма першого рівня декомпозиції IDEF0 (рис. 2.3) використовується для подальшого розчленування великих та складних процесів на більші елементарні та керовані етапи. Це дозволяє отримати більш детальний огляд внутрішніх функцій та взаємодій системи, розкриваючи конкретні завдання та процеси, які здійснюються на кожному етапі роботи.

Розглянемо детальніше кожен етап на даній діаграмі:

- Блок A1. Створення фінансового плану: цей етап включає введення у застосунок даних щодо джерел доходів та витрат та їх ліміти. Користувач має можливість визначити свій фінансовий план на основі цих даних. Застосунок обробляє цю інформацію і генерує фінансовий план, який включатиме розподіл бюджету, мету збереження, та інші важливі аспекти фінансового планування.
- Блок A2. Створення гаманця: на цьому етапі користувач може ввести дані про свою банківську картку для подальшої прив’язки до гаманця. Застосунок використовує цю інформацію для створення гаманця, який відображає слугує контейнером операцій доходу та витрат, містить інформацію про доступні кошти та інші фінансові аспекти.
- Блок A3. Додання транзакції: на цьому етапі користувач водить дані щодо нових транзакцій. Це включати операції витрат або доходів. Застосунок зберігає цю інформацію та враховує її при аналізі фінансового стану користувача.
- Блок A4. Аналіз транзакцій: на цьому етапі застосунок аналізує фінансовий план та транзакції, щоб оцінити, наскільки користувач дотримується свого плану. Результати аналізу можуть включати порівняння фактичних транзакцій із запланованими витратами та доходами, а також вказівки щодо ефективного фінансового управління чи нагадування.

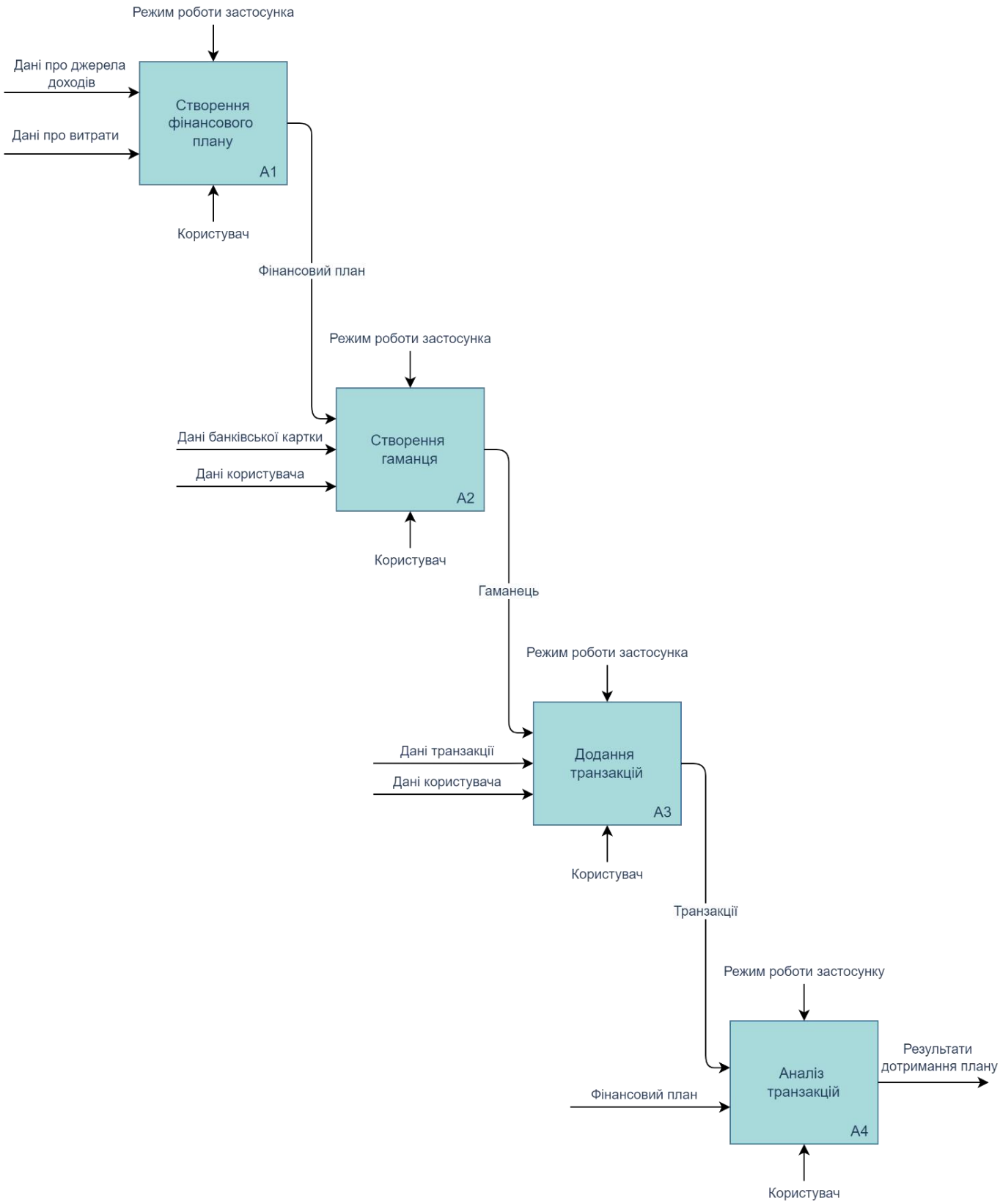

Рис. 2.3 – Діаграма першого рівня декомпозиції IDEF0

2.2.2 Нефункціональні вимоги

Нефункціональні вимоги - це характеристики та показники, що визначають вигляд та функціональність програмної системи. Цей тип вимог відображає якість та ефективність програмного продукту [19].

Для розробки програмного застосунку критично враховувати нефункціональні вимоги, які наведені у таблиці 2.1.

Таблиця 2.1

Характеристика нефункціональних вимог до програмного забезпечення, що розроблюється

№ з/п	Характеристика вимоги
NF 1	Застосунок повинен працювати як по мережі LTE, так і по з'єднанню Wi-Fi, забезпечуючи стабільний доступ до функцій навіть при обмежених мережевих умовах
NF 2	Застосунок повинен забезпечувати швидку генерацію звітності, забезпечуючи миттєвий доступ до фінансових даних користувачів
NF 3	Користувачам повинна бути доступна можливість використання деяких функцій застосунку без постійного з'єднання з Інтернетом
NF 4	Застосунок повинен забезпечувати високий рівень захисту особистих та фінансових даних користувачів від несанкціонованого доступу та зловживань
NF 5	Застосунок повинен бути сумісний з операційними системами Android версії 8.0 і вище та iOS версії 10 і вище
NF 6	Застосунок повинен надавати користувачам оперативний доступ до фінансової інформації, забезпечуючи швидку обробку запитів та відображення даних
NF 7	Застосунок повинен мати інтуїтивний інтерфейс, який забезпечить легке використання для користувачів на будь-якому рівні навичок
NF 8	Застосунок повинен надавати можливості розширення функціоналу для індивідуальних потреб користувачів

Продовження табл. 2.1

№ з/п	Характеристика вимоги
NF 9	Протягом місяця застосунок повинен бути доступним користувачам принаймні 98% часу
NF 10	Застосунок повинен підтримувати автоматичне резервне копіювання даних користувачів та відновлення інформації в разі втрати або пошкодження даних
NF 11	Застосунок повинен ефективно працювати на пристроях з різними технічними характеристиками, відповідаючи мінімальним вимогам до обладнання
NF 12	Застосунок повинен підтримувати двох факторну автентифікацію для підвищення рівня безпеки входу
NF 13	Інформація про облікові записи для входу через OAuth не повинна зберігатись вразливим чином, забезпечуючи конфіденційність користувачів
NF 14	Застосунок повинен підтримувати українську та англійську для зручності різних категорій користувачів
NF 15	Застосунок повинен працювати з різними видами валют
NF 16	Застосунок повинен правильно функціонувати та виявляти специфічні для культур особливості

2.2.3 Бізнес-вимоги

Бізнес-вимоги – це високорівневі вимоги проєкту, які узагальнено визначають бізнес-цілі та завдання з погляду компанії або замовника. Це вид стратегічної мапи проєкту. Залежно від областей застосування, бізнес-вимоги можуть бути або простими, описаними у кількох рядках, або складним набором цілей з різних сфер діяльності [20, 21].

При створенні застосунку для фінансового планування, аналізу та оптимізації витрат, замовник прагне досягнути наступних цілей:

- Досягти понад 40 000 активних користувачів протягом першого півроку після впровадження програмного забезпечення, а до кінця року – перевищити показник у 200 000 користувачів.
- Отримати 35% користувачів, що вибрали платну підписку за функціональні можливості або підтримку розробників протягом шести місяців з моменту запуску проєкту.
- Упродовж першого року функціонування застосунку, вдосконалити алгоритм складання фінансового плану на підставі користувацьких відгуків та статистики.
- Отримати 25% користувачів, які готові оплачувати можливість синхронізації даних у файлообміннику протягом перших шести місяців від впровадження програми, з подальшим зростанням цього показника на 10% до кінця року.
- Завоювати аудиторію як серед користувачів iOS, так і серед користувачів Android, зокрема 20% користувачів планшетів, протягом першого року функціонування.
- Протягом року після впровадження продукту, розширити перелік банків для синхронізації даних про транзакції користувачів до п'яти.
- Планування доходу у суму 200 000 грн. за півроку від монетизації реклами, платних підписок та функції синхронізації з файловим обмінником після запуску продукту. Метою є перевищення цієї суми до 600 000 грн. до завершення року.

2.2.4 Вимоги користувачів

Вимоги користувачів є важливим етапом у розробці програмного забезпечення, оскільки вони визначають функціональні та нефункціональні вимоги, які має задовольняти система. Вимоги користувачів визначають очікування та потреби кінцевих користувачів щодо функцій та можливостей програми. Для ефективного документування та розуміння цих вимог використовується діаграма варіантів використання.

Діаграма варіантів використання є інструментом моделювання, який дозволяє візуалізувати різні сценарії взаємодії між користувачами та системою. Цей вид діаграми створює зручну і зрозумілу картину того, як користувачі будуть взаємодіяти з програмним забезпеченням у різних ситуаціях.

На діаграмі варіантів використання (рис. 2.4) зображені ключові сценарії взаємодії користувачів із розроблювальним застосунком. Розглянемо деякі з них:

- Реєстрація: цей варіант використання передбачає можливість користувачів реєструватися в застосунку шляхом авторизації через облікові записи Google, Twitter або Microsoft.
- Робота з гаманцями: користувачі можуть створювати, редагувати та видаляти гаманці – ізольовані фінансові контейнери для відстеження витрат та доходів. Гаманці можуть бути прив'язані до банківських карток для автоматичної синхронізації транзакцій.
- Робота з транзакціями: - користувачі можуть створювати, редагувати та синхронізувати транзакції, що відносяться до конкретного гаманця. Це дозволяє точно відстежувати витрати та доходи, а також автоматизує процес фінансового аналізу.
- Робота з категоріями доходів та витрат - функціонал дозволяє користувачам створювати та редагувати категорії для класифікації своїх доходів та витрат. Це спрощує аналіз фінансів та дозволяє генерувати звіти за різними категоріями.
- Робота з фінансовим планом: користувачі можуть створювати та відстежувати фінансові плани, визначаючи мети та обмеження для доходів та витрат. Цей варіант використання допомагає в плануванні та оптимізації фінансових ресурсів.
- Робота зі звітами: застосунок дозволяє користувачам генерувати різноманітні звіти, такі як звіти про витрати, доходи, бюджет та інші. Це забезпечує зрозумілу статистику та допомагає в аналізі фінансового стану.

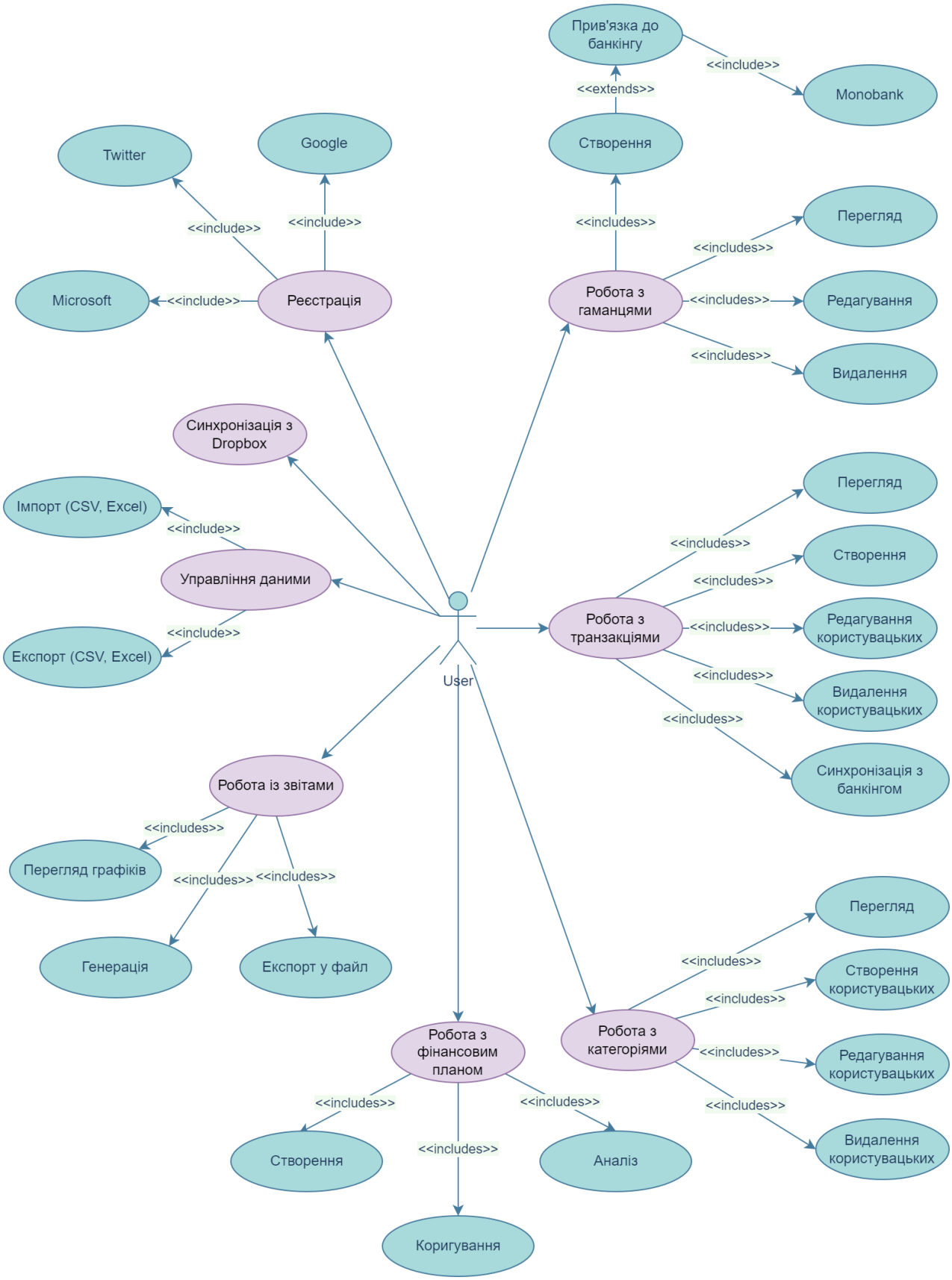

Рис. 2.4 – Діаграма варіантів використання

Ці варіанти використання допомагають конкретизувати та визначити очікування користувачів, а також створюють відмінну основу для подальшого проектування та розробки функціоналу застосунку для фінансового планування, аналізу та оптимізації витрат.

2.2.5 Системні вимоги

Системні вимоги визначають необхідні параметри та умови програмного забезпечення, що повинні бути виконані пристроєм для успішного і безперебійного функціонування програми. Наприклад, несумісність версій операційних систем або апаратного забезпечення, а також відсутність доступу до мережі Інтернет можуть призвести до неправильної роботи програми, її затримок або взагалі неможливості коректного встановлення [22].

Застосунок, що розробляється для фінансового планування, аналізу та оптимізації витрат, проектується як мобільне кросплатформне програмне забезпечення. Це означає, що його можна встановити на різноманітні пристрої з різними операційними системами, що розширює його доступність для більшої аудиторії користувачів.

Мобільна операційна система

Нижче описано мінімальні версії операційних систем, необхідних для нормальної роботи програмного забезпечення. Рекомендується оновити операційну систему до останньої версії, якщо поточна не відповідає вказаним вимогам:

- Android: версія 7.0 і вище
- iOS: версія 10 і вище

Апаратне забезпечення

Нижче наведено опис мінімальних вимог до апаратного забезпечення, необхідних для комфортної роботи із застосунком:

- Роздільна здатність дисплея 480x800 пікселів або вище
- Діагональ екрану 4 дюйми або більше
- Сенсорний дисплей
- Підтримка Wi-Fi або мобільного передавання даних LTE

- Оперативна пам’ять об’ємом не менше, ніж 500 МБ
- Процесор із двома ядрами
- Вільна пам’ять на пристрої обсягом 100 МБ
- Наявність задньої камери, якщо планується використання функції штрих-кодування для підключення до банкінгу.

Підключення до інтернету

- Мобільний інтернет: 3G, 4G, або LTE для завантаження даних та синхронізації з сервером
- Wi-Fi: Рекомендовано для високошвидкісного інтернет-з'єднання та оновлення додатку

Висновки

У другому розділі магістерської роботи були сформовані вимоги до мобільного застосунку, призначеного для фінансового планування, аналізу та оптимізації витрат. Початково було визначено життєвий цикл розробки програмного продукту, що мало прямий вплив на обрану методологію.

Специфікація вимог охоплювала визначення різних категорій вимог: функціональних та нефункціональних вимог, бізнес-вимог, користувацьких та системних вимог. Функціональні вимоги, які були описані, регулюють поведінку та можливості програмного забезпечення, тоді як нефункціональні вимоги фокусуються на продуктивності та встановлюють показники для майбутньої системи. Визначені бізнес-вимоги описують загальні бізнес-цілі проекту, користувацькі вимоги визначають необхідність впровадження певних можливостей для користувачів, а системні вимоги характеризують необхідні характеристики для правильної роботи програмного засобу.

Таким чином, були визначені очікувані потреби та умови роботи мобільного застосунку, розробленого для фінансового планування, аналізу та оптимізації витрат. Проведений аналіз вимог є важливим етапом у процесі розробки програмного продукту та є ключовим для успішного завершення проекту.

РОЗДІЛ 3.
СТРУКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ

3.1 Вибір платформи та інструментів для розробки програмного застосунку

Одним з ключових етапів в процесі розробки програмного забезпечення є вибір оптимальної платформи, для розробки мобільного застосунку, оскільки від цього напряму буде залежати стратегія розробки. Цей вибір також визначає інструменти, які будуть використані у процесі створення застосунку.

Під час планування, також важливо визначити техніку розробки застосунку: нативну, кросплатформну або гібридну [23].

Нативна розробка застосунку - це метод, коли програмне забезпечення розробляється спеціально для певної операційної системи, використовуючи SDK, мови програмування та інструменти, які зазвичай надаються виробником цієї платформи (табл. 3.1).

Таблиця 3.1

Порівняння нативної розробки для різних ОС

Операційна система	Мова програмування	Інструменти розробки
Android	Kotlin або Java	Android Studio (або Eclipse)
iOS	Objective-C	xCode
Windows Phone	C#	Visual Studio

У контексті розробки застосунку нативна розробка може включати низку обмежень. По-перше, цей підхід призводить до затримок у розробці, оскільки потрібно створювати окремі версії для різних платформ (Android та iOS). Це може подовжити терміни розробки та впровадження застосунку.

По-друге, нативна розробка вимагає великих витрат ресурсів, оскільки потрібно стежити за двома окремими кодовими базами для різних платформ. Це може бути фінансово та трудовитратно, особливо для невеликого проекту.

По-третє, управління оновленнями та виправленнями помилок у нативному застосунку вимагає впровадження змін окремо для Android і iOS. Це може ускладнити та подовжити процес підтримки.

Гібридні застосунки, подібно до нативних, створюються для конкретної операційної системи. Однак вони розробляються з використанням веб-технологій, таких як HTML, CSS та JavaScript, що дозволяє використовувати код повторно. Гібридні рішення функціонують у межах нативного контейнера та використовують механізм браузера пристрою (але не сам браузер) для відтворення HTML, CSS та виконання коду JavaScript. Для отримання доступу до можливостей пристрою, які не доступні в звичайних веб-застосунках (наприклад, камера чи локальне сховище), створено рівень абстракції web-to-native.

Техніка кросплатформного розроблення дозволяє створювати застосунки, які можуть працювати на кількох операційних системах одночасно. Це означає, що велика частина кодової бази проєкту залишається спільною для різних платформ, часто від 90% до 98%. Це сприяє швидшому розвитку програмного забезпечення.

Порівняння технік розробки наведено в таблиці 3.2.

Таблиця 3.2

Порівняння технік розробки

Фактор	**Нативна**	**Гібридна**	**Кросплатформна**
Механізм відображення	Нативний	Браузер	Нативний
Швидкодія застосунків	Нативна	Середня	Майже нативна
Інструменти налагодження	Використовуються нативні інструменти	Нативні та веб-інструменти	Залежить від фреймворку

Кодова база	Різна для кожної платформи	Одна для всіх платформ	Одна для всіх платформ

Продовження табл. 3.2

Фактор	**Нативна**	**Гібридна**	**Кросплатформна**
Вартість розробки	Висока	Дешевша за нативну	Дешевша за нативну
Доступ до функціоналу платформи	Має повний доступ до усіх функцій та API конкретної платформи	Обмежений доступ, але можливості зростають завдяки розвитку гібридних фреймворків	Залежно від фреймворку, може мати обмеження, але намагається надати максимальний доступ

Для створення програмного забезпечення для аналізу фінансового стану було прийнято рішення скористатися кросплатформною розробкою. Це обумовлено тим, що цей підхід є більш вигідним у порівнянні з нативною розробкою за рахунок можливості повторного використання коду для різних платформ. Крім того, важливим аспектом вибору є здатність застосунку працювати швидко, що відповідає вимогам, поставленим раніше.

Кожна з платформ для розробки мобільних застосунків має свої переваги та обмеження. Важливо ретельно проаналізувати ці аспекти, виявити сильні та слабкі сторони кожної платформи та вибрати варіант, що найбільш підходить для подальшої розробки.

Наразі у сфері розробки мобільних додатків виділяються два основних лідери: Flutter та Xamarin.

Flutter – це комбінація набору інструментів та фреймворку для розробки за допомогою мови програмування Dart від компанії Google. Його початки сягають 2017 року, а вже через рік була представлена перша стабільна версія платформи. За даними офіційної статистики, на сьогоднішній день існує понад 400 000 додатків, розроблених на Flutter [24]. Цей фреймворк має декілька переваг:

- Кросплатформність

- Швидкодія за рахуок вбудованого двигуна Skia
- «Hot reload», що полегшує налагодження
- Елементи UI сумісні між різними ОС
- Активна спільнота

Оскільки Flutter перебуває в стадії активної розробки, він має свої недоліки та обмеження:

- Великий розмір файлу застосунку
- Не повністю нативні віджети
- Менша екосистема порівняно з іншими фреймворками
- Вивчення мови Dart
- Проблеми з оптимізацією під iOS

Xamarin – це відкритий фреймворк для створення кросплатформних застосунків з використанням мови програмування C# [25]. На відміну від Flutter, Xamarin з'явився на ринку раніше, у 2011 році, і був представлений одним із засновників платформи Mono. Починаючи з 2016 року, після того як була придбана компанією Microsoft, платформа була інтегрована в .NET. За роки свого існування Xamarin накопичив ряд переваг, завдяки чому визнається піонером у розробці кросплатформних мобільних застосунків. Перелік переваг:

- Великий розмір файлу застосунку
- Не повіністю нативні віджети
- Менша екосистема порівняно з іншими фреймворками
- Вивчення мови Dart
- Проблеми з оптимізацією під iOS

Проте Xamarin має також слабкі сторони:

- Великий розмір вихідного файлу
- Затримки із оновленням при виході нової версії Android
- Все ще доволі високі системні вимоги
- Прив’язка до стеку .NET

Після ретельного порівняльного аналізу Flutter та Xamarin для розробки нашого застосунку було обрано Xamarin. Однією з важливих переваг Xamarin є

можливість використання мови програмування C#. Це враховує зручність роботи для розробників, які вже мають досвід з C#.

Крім того, Xamarin забезпечує більше можливостей для взаємодії з нативними API та віджетами на обох платформах (Android та iOS), що гарантує більший контроль над функціональністю та дозволяє створити більш нативний вигляд застосунка.

Інтеграція Xamarin з Visual Studio, яка є потужним інструментом для розробників, робить розробку зручною та дозволяє ефективно відлагоджувати код [26].

Visual Studio також надає можливість швидко виконувати та тестувати додаток на різних платформах, забезпечуючи високий рівень ефективності та швидкості розробки. Така інтеграція дозволить максимально використовувати переваги розробки на базі Xamarin у зручному та продуктивному середовищі Visual Studio.

Також важливим аспектом було те, що Xamarin підтримує не лише Android та iOS, а й інші платформи, що стає перевагою для можливого майбутнього розширення функціоналу застосунка.

Нарешті, екосистема Xamarin та якісна документація сприяють швидкому розвитку проекту, а також полегшують вирішення будь-яких технічних питань. З урахуванням цих факторів, Xamarin – інструмент, що в нашому випадку найбільш підходить для успішної розробки застосунка.

3.2 Програмна архітектура застосунку

Архітектура програмного продукту є ключовим аспектом в розробці зручних, надійних та ефективних застосунків. Вибір відповідної архітектури має визначальне значення для досягнення мети проекту та забезпечення його подальшого розвитку та супроводу.

Однією з найбільш поширених архітектур є трирівнева архітектура, що визначається своєю здатністю розділення функціональності застосунку на три основні рівні (рис 3.1) [27].

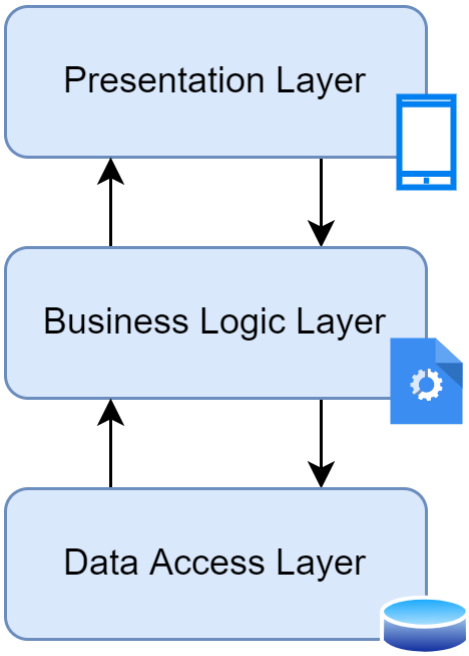

Рис. 3.1 – Шари трирівневої архітектури

Рівень представлення (presentation layer) - це верхній рівень застосунку, який відповідає за взаємодію з користувачем та представлення інформації. Тут знаходиться все, що пов’язано із графічним інтерфейсом користувача (UI). Даний рівень обробляє введення користувача, відображає дані та результати операцій а також керує подіями і взаємодіє з іншими рівнями.

Рівень бізнес-логіки (business logic layer) - це середній рівень, де виконується бізнес-логіка застосунку та розташовані набір бізнес-правил, алгоритми та логіка додатку. Він відповідальний за здійснення бізнес-операцій предметної області.

Рівень доступу до даних (data access layer) - це нижній рівень, відповідальний за роботу з базою даних або іншими джерелами даних такі як публічні API банків [28]. Основними функціями даного рівня є виконання операцій з базою даних (читання, запис, оновлення, видалення) та забезпечення інших рівнів доступом до необхідних даних.

Важливо зазначити що існує визначена взаємодія між презентаційним та бізнес-логічним рівнями, коли презентаційний рівень викликає відповідні методи та передає дані для подальшої обробки. Бізнес-логіка, у свою чергу, взаємодіє з презентаційним рівнем, використовуючи дані для координації обчислень та логічних операцій. Крім того, бізнес-логіка передає ці дані рівню доступу до даних

для зберігання чи отримання інформації з бази даних. Такий взаємозв'язок дозволяє системі ефективно координувати функції кожного рівня, забезпечуючи гнучкість та оптимальну взаємодію між ними.

У проекті, окрім основних рівнів, є й додаткові проекти– AppSetting і MigrationsManager (рис. 3.2). Перший відповідає за зберігання та надання доступу до налаштувань застосунку. А MigrationsManager використовується для запуску процесів міграції на локальній базі даних, що є методом для редагування структури бази даних.

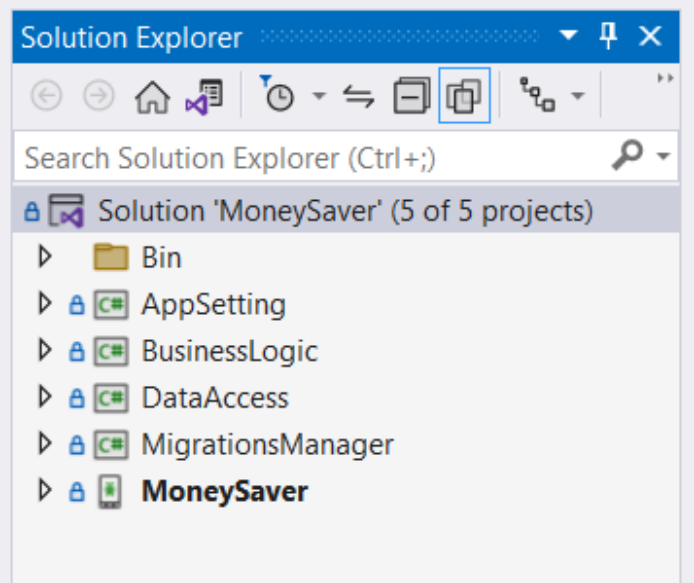

Рис. 3.2 – Структура рішення застосунку

Вибір трирівневої архітектуру для розрозблювального застосунку обґрунтовано наступними перевагами:

- Масштабованість та підтримка – така архітектура полегшує масштабування та розвиток додатку. Кожен рівень може розвиватися незалежно, що полегшує підтримку та розширення функціоналу. Також можливість заміни або оновлення одного рівня без впливу на інші робить систему більш гнучкою.
- Розділення відповідальностей – архітектура дозволяє чітко розділити функції та відповідальності між різними рівнями.

- Тестування та підтримка- розділена структура сприяє ефективному тестуванню. Кожен рівень може бути протестований окремо, що полегшує виявлення та усунення помилок. Підтримка стає менш проблематичною, оскільки можливо впроваджувати зміни в окремих рівнях без значного впливу на інші.
- Незалежність від платформи – Xamarin надає можливість розробки мобільних додатків для різних платформ (iOS, Android) на C#. Трирівнева архітектура допомагає вирішити завдання множинності платформ, забезпечуючи легкість портабельності та багатоплатформності.
- Бізнес-логіка та взаємодія з даними – для планування та оптимізації витрат ключовими є точність та надійність обчислень. Це може бути краще виражено в окремому рівні бізнес-логіки, де можна обробляти складні бізнес-правила та аналіз.

3.3 Класи та взаємодія між ними

В ході розробки застосунку, яка здійснюється за допомогою Xamarin та мови програмування C#, створюється система, що ґрунтується на класах, взаємодія між якими визначає ключові аспекти функціональності продукту. Класи в програмі не лише виконують конкретні завдання, а й взаємодіють між собою, обмінюючи даними та змінюючи свій внутрішній стан.

Кожен клас у програмі відповідає за конкретну функціональність та долучається до вищезгаданої трирівневої архітектури. Діаграма класів використовується для візуального представлення структури, методів та зв'язків між цими класами (рис 3.3).

Використання діаграм класів стає ключовим елементом для глибшого розуміння структури нашого програмного продукту. Це інструмент, що сприяє чіткому візуальному відображенню внутрішньої організації та взаємодії класів, що є критичним для подальшої розробки, тестування та розвитку застосунку [29, 30, 31].

На цій діаграмі можна бачити класи, які визначають основні компоненти нашого застосунку, і їхні взаємозв'язки. Кожен клас несе в собі свої власні

відповідальності, в той час як взаємодія між ними забезпечує високоорганізовану та ефективну роботу програми. Розглянемо наступні види класів:

- Activities and Fragments виступають в ролі будівельних блоків у Xamarin-застосунку, об'єднуючи графічний інтерфейс і код, який реагує на події, що виникають в процесі взаємодії користувача з програмою. Activities відповідають за весь екран та його візуальне представлення, тоді як Fragments розділяють екран на частини для більшої гнучкості та повторного використання.
- Adapters використовуються для зв'язування джерела даних із елементами графічного інтерфейсу, щоб коректно відображати дані користувачеві. Вони дозволяють ефективно управляти відображенням списків чи інших структур даних на екрані, забезпечуючи плавну та оптимізовану роботу інтерфейсу.
- Services виступають як компоненти, що реалізують бізнес-логіку застосунку. Вони виокремлюють певний функціонал від графічного інтерфейсу та служать абстракцією від джерела даних, що спрощує підтримку та розширення функціональності.
- Класи Database Access є ключовими для роботи з базою даних та іншими джерелами даних в застосунку. Вони реалізують операції CRUD над базою даних і моделюють її структуру. Ці класи відповідають за ефективне та безпечне взаємодію з даними, забезпечуючи стабільність та надійність додатку.

Також діаграма класів ілюструє взаємозв'язки між класами. Наприклад, ReportsFragment взаємодіє з ReportsService, запитуючи дані для відображення звітності, тоді як ReportsService взаємодіє з DatabaseContext для отримання необхідних для генерації звіту даних.

Ця взаємодія забезпечує ефективний обмін інформацією між різними рівнями та дозволяє системі працювати координовано та гнучко. Детальне розглядання внутрішніх механізмів кожного класу та їхньої взаємодії визначить ключові аспекти функціонування програмного продукту і стане основою для подальших аналізів та вдосконалень.


Database Access
EAccountType
<<enumeration>>
NotDefined
Black
White
Platinum
Iron
Fop
Yellow
EWalletType
<<enumeration>>
NotDefined
Default
Shared
User
+ UserId: int
+ Guid: Guid
+ Name: string
+ Email: string
Currency
+ CurrencyId: int
+ Name: string
+ Code: int
+ Symbol: string
Operation
+ OperationId: int
+ WalletId: int
+ CategoryId: int
+ Amount: long
+ Date: DateTime
+ MCC: int
+ Description: string
+ Comment: string
FinancialPlan
+ FinancialPlanId: int
+ From: DateTime
+ To: DateTime
Wallet
+ WalletId: int
+ UserId: int
+ CurrencyId: int
+ Guid: Guid
+ Name: string
+ Balance: long
+ Type: EWalletType
+ Picture: string
DatabaseContext
+ Accounts: DbSet<Account>
+ Categories: DbSet<Category>
+ Currencies: DbSet<Currency>
+ Operations: DbSet<Operation>
+ Users: DbSet<User>
+ Wallets: DbSet<Wallet>
+ FinancialPlans: DbSet<FinancialPlan>
+ OnModelCreating(): void
+ OnConfiguring(): void
Account
+ AccountId: int
+ UserId: int
+ OriginalId: int
+ WalletId: int
+ CurrencyId: int
+ Type: EAccountType
+ MaskedPan: string
FinancialPlanItem
+ FinancialPlanItemId: int
+ CategoryId: int
+ FinancialPlanId: int
+ AmountLimit: long
+ AmountCurent: long
Category
+ CategoryId: int
+ ParentId: int?
+ Name: string
+ Picture: string
+ IsIncome: bool
+ IsSystem: bool
Services
FinancialPlanService
+ GetFinancialPlanInfo(): FinancialPlanInfoDto
+ CreateFinancialPlanAsync(CreateFinancialPlanDto): void
+ GetCurrentFinPlanItems(): FinancialPlanItemInfoDto[]
+ GetFinPlanAsync(date: DateTime): FinancialPlanDto
WalletService
+ HasAnyWalletAync(): bool
+ CreateWalletAsync(CreateWalletDto): void
+ GetAllWallets(): WalletItemDto[]
+ GetStringsAsync(): string[]
CategoryService
+ GetStringsAsync(): string[]
+ GetIncomeCategoriesAsync(): CategoryDto[]
+ GetExpenseCategoriesAsync(): CategoryDto[
ReportService
+ GetCartesianChartData(): CartesianChartDataDto
+ GetPieChartData(): PieChartDataDto
OperationService
+ CreateAsync(CreateOperationDto): void
+ GetAllOperations(): OperationItemDto[]
UserService
+ IsUserExists() : bool
+ CreateUsesAsync (CreateUserDto)
Activities and Fragments
ReportsFragment
+ from: DateTime
+ to: DateTime
+ NewInstance(): ReportsFragment
+ OnCreateView():View
- GoToDetailedPieChart(): void
- OnClickDateEditText(): void
- DatePickedHandler(): void
- RefreshCharts(): void
- InitCartesianChart(): void
- InitPieChart(): void
MainActivity
+ OnCreate(): void
+ OnNavigationItemSelected():void
- LoadFragment(): void
SplashActivity
+ OnCreate(): void
+ OnResume(): void
+ Startup(): void
CreateOperationActivity
+ OnCreate(): void
+ CreateOperation():void
- ShowDatePicker(): void
- OnOptionsItemSelected(): void
OperationsFragment
- operationAdapter: OperationsAdapter
- waletsAdapter: WalletsAdapter
+ NewInstance(): OperationsFragment
+ Fab_Click():void
- OnWalletClick(): void
FinancialPlanFragment
- planAdapter: FinancialPlanAdapter
- categoryAdapter: CategoryAmountAdapter
+ NewInstance(): FinancialPlanFragment
+ OnCreateView():View
- GoToCreate(): void
CreateWalletActivity
+ OnCreate(): void
- SaveBtn_Click():void
- OnOptionsItemSelected(): void
Adapters
WalletsAdapter
+ ItemCount: int
+ ItemClick: event
- package: string
- resources: Resources
+ OnCreateViewHolder(): ViewHolder
+ OnBindViewHolder():void
- OnClick(): void
OperationsAdapter
+ ItemCount: int
- package: string
- resources: Resources
+ OnCreateViewHolder(): ViewHolder
+ OnBindViewHolder():void
- OnClick(): void
FinancialPlanAdapter
+ ItemCount: int
- package: string
- resources: Resources
- activityContext: Context
+ OnCreateViewHolder(): ViewHolder
+ OnBindViewHolder():void
- OnClick(): void
CategoryAmountAdapter
+ ItemCount: int
- package: string
- resources: Resources
- activityContext: Context
+ OnCreateViewHolder(): ViewHolder
+ OnBindViewHolder():void
- OnClick(): void

Рис. 3.3 – Діаграма класів

3.4 База даних та зв’язки між сутностями

Обрання системи управління базами даних (СУБД) є стратегічним рішенням, яке визначає ефективність та успішність розробки програмного продукту. В контексті нашого дослідження та розробки мобільного застосунку на платформі Xamarin, вибір СУБД набуває ключового значення, оскільки він прямо впливає на ряд критичних аспектів.

Зважаючи на вимоги застосунку для нього добре підійде СУБД SQLite, що має наступні характеристики:

- Легкість використання – SQLite це вбудована СУБД, яка не вимагає окремого сервера, і її можна використовувати як локальну базу даних без необхідності налаштування складних серверних конфігурацій.
- Мобільна сумісність – підходить для мобільних додатків, таких як ті, що розробляються на платформі Xamarin. Вона легко інтегрується з такими проектами та забезпечує ефективне управління даними на мобільних пристроях.
- Низькі вимоги до ресурсів – SQLite славиться своєю високою продуктивністю та низьким використанням ресурсів. Це особливо важливо для мобільних додатків, де обмежені ресурси пристроїв можуть впливати на продуктивність.
- Розгортання – SQLite не вимагає значних зусиль для розгортання. Вона працює в межах одного файлу бази даних.
- Широке застосування – SQLite використовується в різних проектах та додатках, включаючи мобільні додатки, десктопні застосунки та веб-сайти. Ця універсальність свідчить про його успішність у різних сценаріях використання.

В обраній СУБД для нашого застосунку використовується реляційна модель, що відображається в структурі бази даних. Цей вибір зумовлений потребами фінансового додатку в плануванні, аналізі та оптимізації витрат.

У реляційній базі даних інформацію організовано у вигляді таблиць, забезпечуючи структуроване зберігання даних, що є важливим аспектом для фінансових додатків, де дані часто мають складні взаємозв'язки. Реляційна модель також дозволяє ефективно виконувати різноманітні запити та аналіз фінансових даних, забезпечуючи точність та швидкість обробки інформації [32, 33].

Методи нормалізації в реляційній моделі сприяють уникненню дублювання даних та забезпечують їх консистентність. Це особливо важливо в нашому випадку, де точність та цілісність інформації є пріоритетами.

Для належного моделювання бази даних необхідно створити структуру інформаційної моделі, яка відображатиме основні сутності та їх зв'язки. Опис цієї моделі включає такі параметри:

- Сутність – об'єкт, що визначається як окрема таблиця в базі даних. Кожна сутність відображає конкретний об'єкт або концепцію, яка має значення для додатку
- Атрибути – властивості та характеристики сутності, які описують її основні характеристики. Атрибути можуть бути обов'язковими (з обов'язковим заповненням) або факультативними (необов'язковими)
- Зв’язки – відображають взаємодію та відношення між різними сутностями
- Бізнес-правила – обмеження та умови, які визначаються для атрибутів сутності або самої сутності. Ці правила допомагають забезпечити консистентність та точність даних.

Відповідно до вимог до застосунку було ідентифіковано ряд ключових сутностей, які визначатимуть структуру бази даних. Для кожної з цих сутностей також визначено атрибути, які будуть використовуватися як стовпці в таблицях. Нижче подано опис даних сутностей:

1. Сутність «Категорія». Визначає види витрат або доходів для транзакцій (операції).

Атрибути сутності:

- Головний ідентифікатор категорії
- Ідентифікатор пов’язаної (головної) категорії

- Назва категорії
- Файл зображення, що буде показано на UI
- Індикатор, що дана категорія є доходом
- Індикатор, що категорія є незмінною (системно)

Зв’язки з іншими сутностями:

- Сутність «Категорія» може будувати ієрархію у вигляді “головна категорія” - “підпорядкована категорія”

Бізнес-правила сутності:

- Головний ідентифікатор категорії є первинним ключем
- Ідентифікатор пов’язаної категорії є вторинним ключем
- Майже всі атрибути повинні містити значення (окрім ідентифікатора пов’язаної категорії)

2. Сутність «Користувач». Описує користувача, що зареєструвався в системі. Є унікальним об’єктом в системі.

Атрибути сутності:

- Головний ідентифікатор
- Нікнейм
- Електронна адреса

Бізнес-правила сутності:

- Головний ідентифікатор є первинним ключем
- Всі атрибути повинні містити значення

3. Сутність «Валюта». Визначає різні грошові валюти, які використовуються в системі.

Атрибути сутності:

- Головний ідентифікатор валюти
- Назва валюти
- Код

Бізнес-правила сутності:

- Головний ідентифікатор є первинним ключем
- Код повинен відповідати формату ISO 4217 [34]

- Всі атрибути повинні містити значення

4. Сутність «Гаманець». Описує рахунок, що представляє для користувача певне джерело, що генерує дохід чи витрату.

Атрибути сутності:

- Головний ідентифікатор гаманця
- Назва гаманця
- Баланс гаманця
- Тип гаманця
- Файл зображення, що буде показано на UI
- Ідентифікатор валюти
- Ідентифікатор користувача

Зв’язки з іншими сутностями:

- Сутність «Гаманець» відноситься лише до одного користувача
- Сутність «Гаманець» підтримує лише одну валюту

Бізнес-правила сутності:

- Головний ідентифікатор є первинним ключем
- Ідентифікатор валюти є вторинним ключем
- Ідентифікатор користувача є вторинним ключем
- Всі атрибути сутності повинні містити значення

5. Сутність «Транзакція». Представляє операцію доходу чи витрати в рамках пов’язаного гаманця.

Атрибути сутності:

- Головний ідентифікатор транзакції
- Сума
- Дата створення
- Код MCC
- Детальний опис
- Короткий коментар
- Ідентифікатор категорії
- Ідентифікатор гаманця

Зв’язки з іншими сутностями:

- Сутність «Транзакція» повинна бути додана в рамках одного гаманця
- Сутність «Транзакція» повинна мати лише одну категорію

Бізнес-правила сутності:

- Головний ідентифікатор транзакції є первинним ключем
- Ідентифікатор категорії є вторинним ключем
- Ідентифікатор гаманця є вторинним ключем
- Код MCC повинен відповідати ISO 18245 [35]
- Всі атрибути окрім коментаря та опису повинні мати значення

6. Сутність «Банківський рахунок». Представляє банківську картку, що прив’язується до гаманця для подальшої синхронізації даних з банку та збереження їх у вигляді транзакцій.

Атрибути сутності:

- Головний ідентифікатор рахунку
- Номер банківського рахунку
- Ідентифікатор користувача
- Ідентифікатор валюти
- Ідентифікатор гаманця

Зв’язки з іншими сутностями:

- Сутність «Банківський рахунок» може бути прив’язаною лише до одного гаманця
- Сутність «Банківський рахунок» може відноситись лише до одного користувача системи
- Сутність «Банківський рахунок» може відноситись лише до однієї валюти

Бізнес-правила сутності:

- Головний ідентифікатор рахунку є первинним ключем
- Ідентифікатор користувача є вторинним ключем
- Ідентифікатор валюти є вторинним ключем
- Ідентифікатор гаманця є вторинним ключем

- Всі атрибути повинні мати значення

7. Сутність «Фінансовий план». Представляє фінансовий план доходів та витрат користувача на визначений проміжок часу.

Атрибути сутності:

- Головний ідентифікатор плану
- Початкова дата
- Кінцева дата
- Ідентифікатор користувача

Зв’язки з іншими сутностями:

- Сутність «Фінансовий план» може мати відношення лише до одного користувача

Бізнес-правила сутності:

- Головний ідентифікатор є первинним ключем
- Ідентифікатор користувача є вторинним ключем
- Всі атрибути повинні мати значення

8. Сутність «Елемент фінансового плану». Ця сутність описує елемент фінансового плану, який місить ліміт та веде облік доходів та витрат по зв’язній категорії.

Атрибути сутності:

- Головний ідентифікатор елементу
- Початково встановлений ліміт
- Поточний ліміт
- Ідентифікатор фінансового плану

Зв’язки з іншими сутностями:

- Сутність «Елемент фінансового плану» відноситься лише до одного фінансового плану

Бізнес-правила сутності:

- Головний ідентифікатор елементу є первинним ключем
- Ідентифікатор фінансового плану є вторинним ключем
- Всі атрибути є обов’язковими

Визначивши атрибути ключових сутностей та їх взаємозв'язки, ми тепер готові підсумувати структуру даних у вигляді концептуальної моделі бази даних. Це абстрактне представлення дозволяє зосередитися на сутностях, їхніх атрибутах та взаємодіях, залишаючи осторонь технічні деталі зберігання та обробки даних.

На рис. 3.4 представлено концептуальну модель бази даних, яка відображає ключові сутності та їхні взаємозв'язки. Дана модель є основою для подальшого проектування та розробки бази даних, надаючи чітке розуміння її структури та логіки взаємодії.

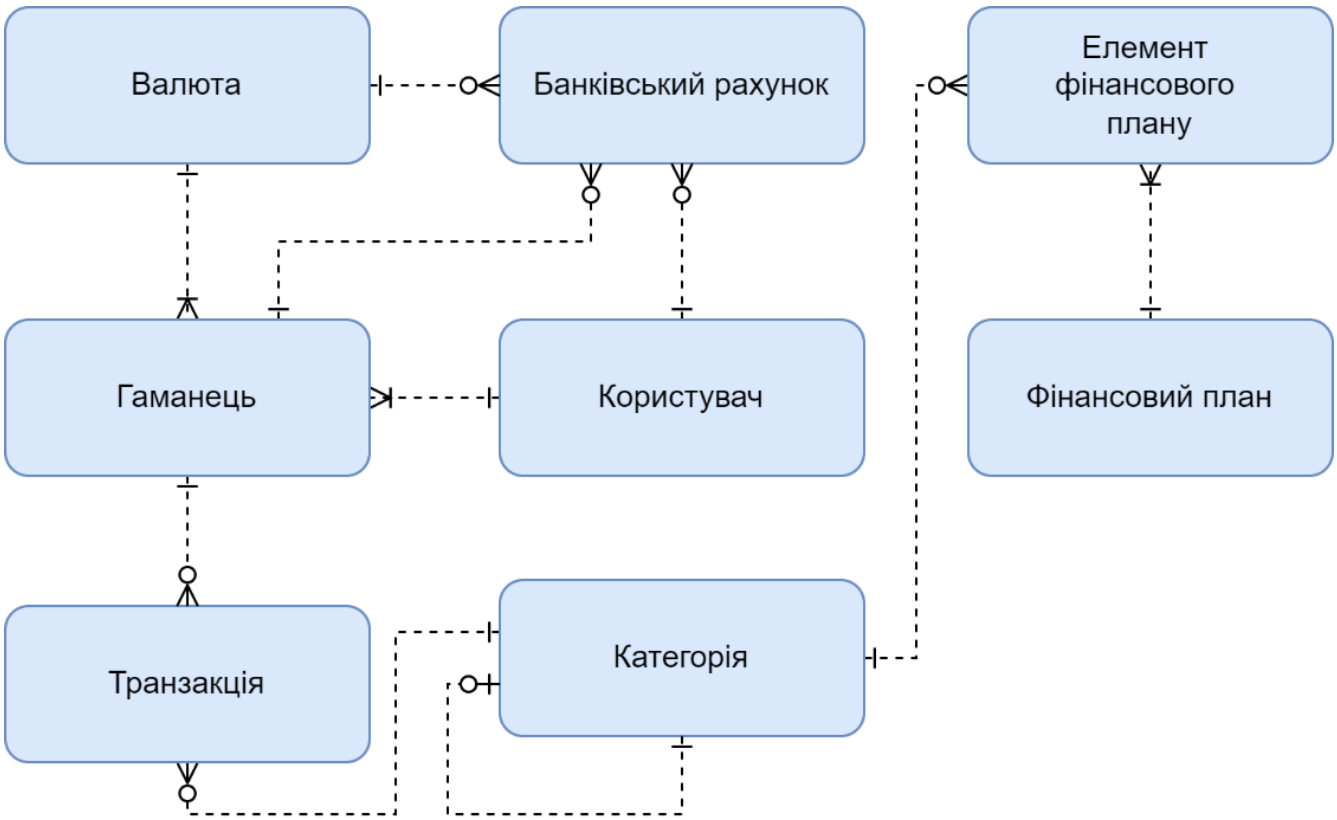

Рис. 3.4 – Концептуальна модель бази даних

На основі розробленої концептуальної моделі бази даних була реалізована фізична модель, яка визначає конкретні технічні аспекти зберігання та обробки даних в системі. Така модель деталізує концепції, визначені в концептуальній моделі, та переводить їх у конкретні об'єкти та структури для певної системи СУБД.

У випадку нашого застосунку, в якості СУБД було обрано SQLite. Тому фізична модель включає в себе визначення концепцій, які є притаманними для даної системи управління баз даних.

На рис. 3.5 зображено фізичну модель бази даних. Це графічне представлення відображає структуру та взаємодії між таблицями, а також включає конкретні деталі, такі як назви полів, їхні типи даних, ключі та інші характеристики.

Реалізація фізичної моделі є кроком у напрямку створення функціональної та ефективної бази даних для розроблювального застосунку.

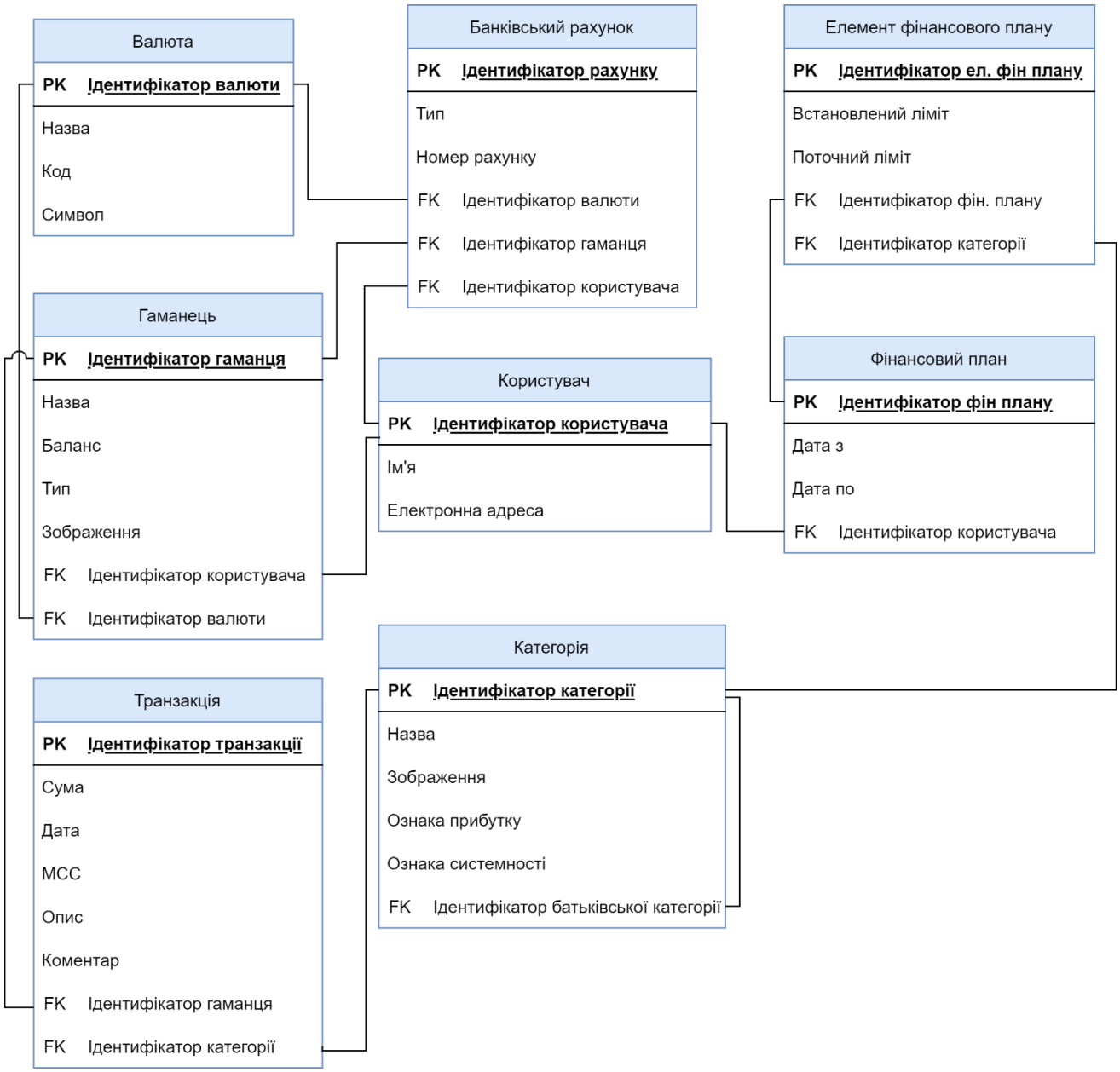

Рис. 3.5 – Фізична модель бази даних

3.5 Безпека даних

Для розроблювального застосунку фінансового планування, аналізу та оптимізації витрат відданий серйозний аналіз питань безпеки даних з метою забезпечення повного захисту конфіденційності та цілісності користувацької інформації. Зокрема були впроваджені наступні механізми безпеки:

- Локальність даних – однією з ключових переваг є використання локальної бази даних SQLite, що дозволяє зберігати дані безпосередньо на пристрої користувача. Це зменшує ризик віддаленого несанкціонованого доступу та гарантує, що дані залишаються під контролем користувача.
- Шифрування БД – всі дані, які зберігаються в базі даних SQLite, будуть піддані механізмам шифрування. Це створює додатковий захист від несанкціонованого доступу, забезпечуючи конфіденційність інформації навіть у випадку фізичного доступу до пристрою.
- Аутентифікація користувача – для забезпечення безпеки доступу до застосунку передбачено дві опції: аутентифікація біометричними даними та введенням паролю. Це дозволяє користувачам вибрати найбільш зручний та безпечний спосіб входу.
- Механізм резервного копіювання БД – планується впровадити механізм регулярного резервного копіювання бази даних. Це забезпечить надійність та можливість відновлення даних у випадку втрати чи випадкового пошкодження інформації, що є важливим аспектом безпеки даних.

Висновки

В третьому розділі дипломної роботи було визначено структуру застосунку.

Проведено глибокий аналіз різних технік розробки мобільних застосунків, зокрема нативної, гібридної та кросплатформної. Порівняльна оцінка їхніх переваг та недоліків була проведена з урахуванням конкретних вимог застосунку

В результаті цього порівняння було визначено, що кросплатформний підхід виявився найоптимальнішим для розробки застосунку для фінансового планування, аналізу та оптимізації витрат. Вибір такої стратегії розробки виправдовується не

лише необхідністю ефективної підтримки обох основних мобільних платформ (iOS та Android), але й зниженням витрат на розробку та підтримку кодової бази.

Окрім того, визначено та обґрунтовано вибір платформи Xamarin та інтегрованої середовища розробки Visual Studio. Xamarin надає можливість розробки на мові програмування C#, що спрощує командну роботу та забезпечує високий рівень переносу коду між платформами. Використання IDE Visual Studio підсилює цю можливість, надаючи широкий функціонал для розробки, тестування та відлагодження коду.

Було також обґрунтовано використання трирівневої архітектури. Цей підхід сприяє структуризації програмного коду, полегшує його розширення та підтримку, що є ключовим для успішного функціонування, розробки та підтримки додатка.

Суттєвий внесок у розробку становить створена діаграма класів, яка чітко визначає головні класи та їхні взаємозв'язки. Це надає команді програмістів відмінний інструмент для розуміння структури програми та полегшує подальше розширення функціоналу.

Вибір реляційної СУБД SQLite був обґрунтований конкретними потребами застосунку, забезпечуючи чітку структуру даних та їх ефективну й швидку обробку. Формування концептуальної та фізичної моделей даних додатково підтримало організованість та послідовність взаємодій між ключовими сутностями.

Усі ці технічні рішення фактично визначають успішний шлях розробки та впровадження застосунку для фінансового планування, аналізу та оптимізації витрат, забезпечуючи його стійкість та оптимальну продуктивність.

РОЗДІЛ 4.
ПРОТОТИП МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ФІНАНСОВОГО ПЛАНУВАННЯ, АНАЛІЗУ ТА ОПТИМІЗАЦІЇ ВИТРАТ

Під час розробки мобільного додатку було отримано MVP (робочий мінімально життєздатний продукт). MVP – це продукт із базовим функціоналом, достатнім для використання. Це оптимальний варіант для демонстрації прототипу мобільного додатку, оскільки вимагає мінімум ресурсів та призначений лише для демонстрації та тестування ідеї.

Створена версія продукту включає основні ключові функції, необхідні для роботи додатку, такі як:

- Реєстрація.
- Створення гаманця.
- Управління транзакціями.
- Аналіз доходів та витрат.
- Формування фінансового плану.

Ці функції є основою для роботи додатку та забезпечують його життєздатність та практичність для користувача.

4.1 Реєстрація

Після завантаження та запуску мобільного додатку на сумісний з раніше визначеними системними вимогами пристрій, новому користувачу пропонується авторизуватися через такі мережі, як Google, Microsoft, Twitter (рис. 4.1).

При виборі однієї з опцій, відкривається вікно вибору доступних облікових записів користувача у відповідній мережі (рис. 4.2). Після вибору потрібного облікового запису в системі створюється новий користувач, який автоматично прив'язується до обраного акаунту. Інформацію облікового запису можна буде редагувати в розділі налаштувань акаунту для забезпечення більшої персоналізації та зручності користувача.

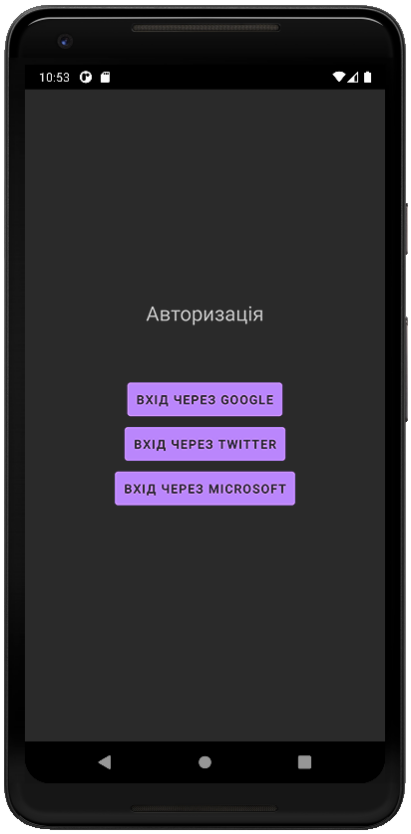

Рис. 4.1 – Авторизація через облікові записи сторонніх сервісів

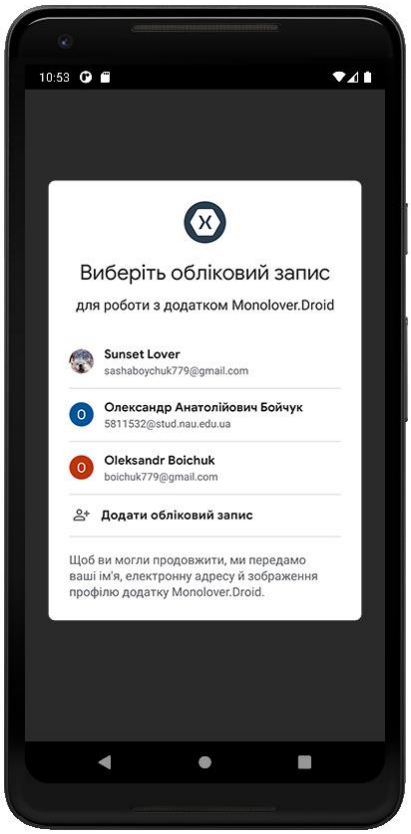

Рис. 4.2 – Вибір облікового запису для авторизації

4.2 Створення гаманця

Після авторизації в застосунку, новий користувач автоматично відкриває можливість створення гаманця (рис 4.3). Гаманець фактично є цифровим контейнером, до якого користувач може прив'язувати та зберігати усі грошові операції, що включають у себе як доходи, так і витрати. Ця функція створює основу для фінансового управління, дозволяючи користувачам моніторити та контролювати свої фінансові операції в зручному цифровому середовищі.

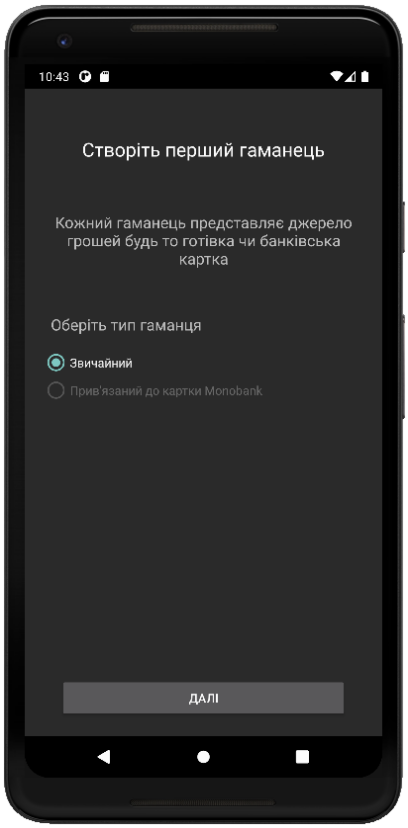

Рис. 4.3 – Створення гаманця

Користувач застосунку має можливість не лише створити гаманець, але й налаштувати його за своїми уподобаннями. Наприклад, він може вказати ім’я для гаманця та початкову кількість коштів на рахунку. Більше того, є опція створення гаманця, який буде прив’язаний до карти Monobank. Це означає, що кошти на гаманці автоматично синхронізуються з залишком на картці відповідного банку, що

дозволяє зручно керувати та контролювати фінансами в одному місці (зображено на рис. 4.4).

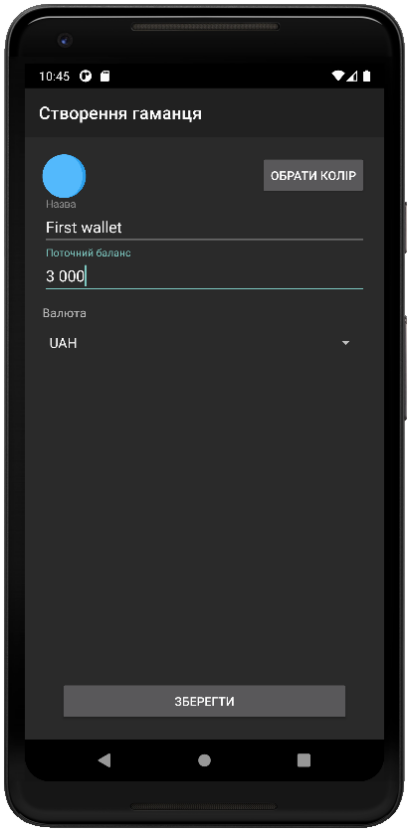

Рис. 4.4 – Характеристики гаманця

У майбутньому планується розширення можливостей застосунку через додання функцій синхронізації з іншими українськими банками, що забезпечить більш широкий спектр вибору для користувачів.

Серед інших можливостей кастомізації гаманця буде включено вибір валюти для обліку коштів, а також можливість вибору кольору для гаманця. У планах підтримка таких валют, як UAH, USD та EUR, щоб надати користувачам більш широкі фінансові можливості та зручність при веденні обліку коштів.

Після успішного створення гаманця користувач переходить на головну сторінку застосунку (рис. 4.5), де доступні наступні функції та елементи інтерфейсу (розташовані від верху до низу сторінки):

- Перелік створених гаманців та їх баланс: Тут користувач може переглянути список усіх своїх гаманців разом з їх поточним балансом. Є можливість

увімкнути або вимкнути гаманець, а також додати новий гаманець до переліку.

- Список транзакцій: Користувач може переглядати усі свої транзакції з можливістю редагування окремої транзакції при натисканні на неї.
- Додавання нової транзакції: Функція, що дозволяє користувачеві швидко та легко додати нову транзакцію до будь-якого з його гаманців.
- Навігаційна панель: Цей елемент інтерфейсу дозволяє користувачеві легко переміщатися між різними сторінками та функціями застосунку (елементи меню).

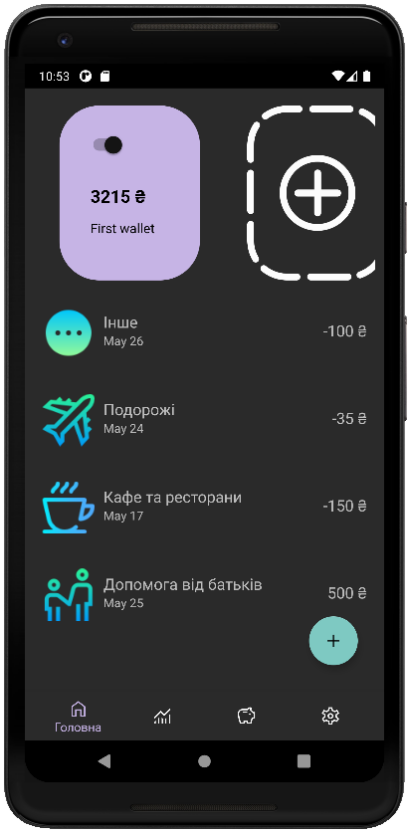

Рис. 4.5 – Головна сторінка застосунку

Зараз у застосунку реалізовано можливість ручного введення грошових операцій для стандартного типу гаманця. Однак заплановано додати функцію синхронізації транзакцій між застосунком та карткою банку, що прив’язана до гаманця (для прив’язаного типу гаманця). Це дозволить автоматично синхронізувати транзакції між обліковим записом у застосунку та банківською карткою, надаючи більшу зручність та точність у фінансовому обліку.

4.3 Керування транзакціями

Крім можливості створення гаманців та перегляду транзакцій, користувач може додавати нові проведені грошові операції до застосунку (зображено на рис. 4.6). Для цього він обирає потрібний гаманець, до якого прив'язується транзакція, обирає категорію витрат або доходів, вказує суму операції, дату її проведення та може додати коментар за бажанням.

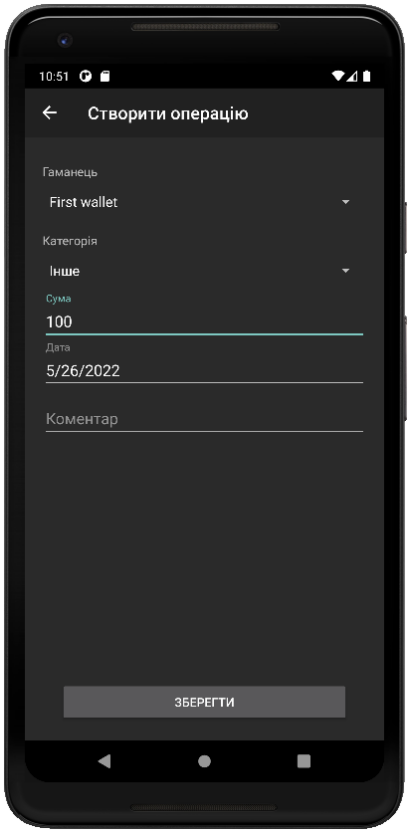

Рис. 4.6 – Створення грошової операції

Користувачеві доступні загальні категорії проведення транзакцій, що вже були введені в застосунок (рис. 4.7). Також заплановано додати функціонал, який дозволить користувачеві самостійно додавати, редагувати та видаляти категорії. Це дозволить кожному користувачеві налаштувати застосунок під свої потреби та вимоги, роблячи його більш персоналізованим та зручним для ведення обліку своїх фінансів.

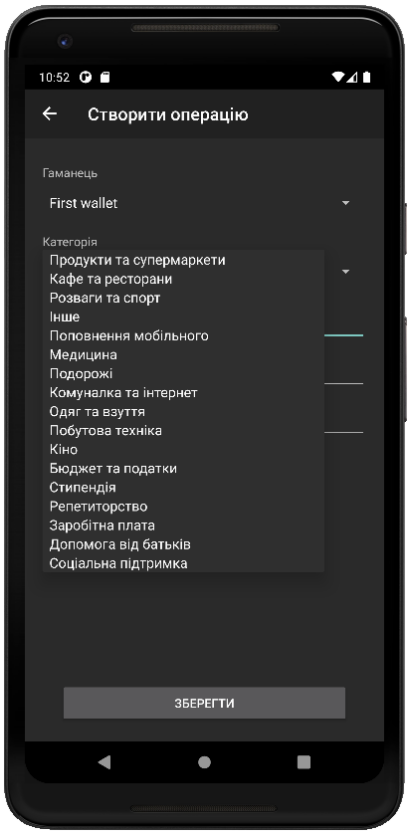

Рис. 4.7 – Загальні доступні категорії проведення транзакцій

Після створення нової транзакції вона автоматично додається до загального списку транзакцій, що доступний користувачеві. Цей список дозволяє переглянути усі проведені операції, а також переглянути деталі конкретної транзакції шляхом простого натискання на відповідне поле з транзакцією.

При бажанні користувач може редагувати деталі конкретної транзакції прямо зі списку транзакцій, щоб внести будь-які необхідні зміни або корекції до інформації щодо операції. Це забезпечує зручний доступ до інформації про транзакції та можливість швидко вносити будь-які необхідні зміни за потреби.

4.4 Аналіз доходів та витрат

Основною метою мобільного застосунку, що розроблюється є надання можливості користувачам аналізувати свої доходи та витрати. Для досягнення цієї мети було реалізовано функцію створення графіків на основі доданих користувачем транзакцій.

При переході на другий елемент в навігаційній панелі відкривається вкладка зі звітністю (проілюстрована на рис. 4.8), яка формується автоматично на основі

наявних у системі даних користувача. Один з параметрів, які тут можна налаштувати – це дата, яку можна вибрати для формування звітів. Ця функція надає користувачам можливість швидко та зручно оцінювати свою фінансову активність за певний період часу за допомогою графічних звітів та аналізу отриманих даних.

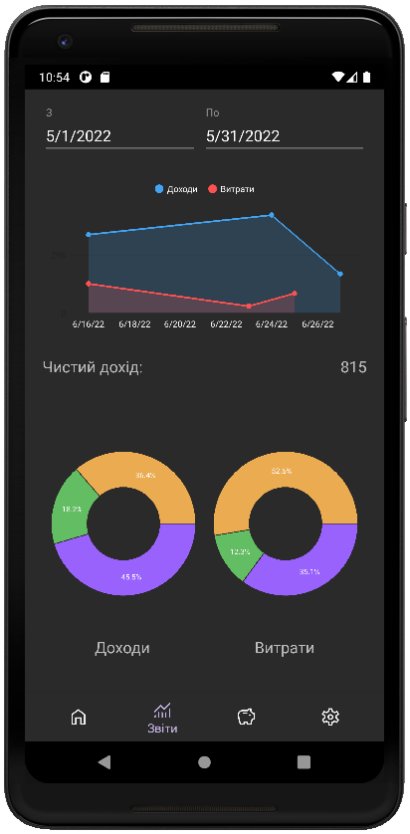

Рис. 4.8 – Сторінка звітності

На першому графіку ми спостерігаємо діаграму доходів та витрат користувача, яка відображена у формі декартової площини. Цей графік можна збільшувати або зменшувати, а також аналізувати точкові дані для отримання більш детальної інформації.

Результатом обчислень у цьому графіку є сума чистого доходу - це грошовий прибуток або збиток, який користувач отримав протягом обраного ним проміжку часу після врахування усіх доходів та витрат. Це дає змогу користувачам легко оцінювати свою фінансову ситуацію та формувати уявлення про те, як їхні витрати співвідносяться до їхніх доходів у певний період часу.

На другому графіку відображені окремі кругові діаграми, які віддзеркалюють дані щодо витрат та доходів користувача. Кожен сегмент цих діаграм представляє

окрему категорію витрат чи джерело доходу. Натиснувши на будь-який з сегментів, користувач може отримати детальнішу інформацію про його складові частини.

Більш того, кожна з цих кругових діаграм може бути розглянута у деталізованому вигляді. Для цього потрібно натиснути на назву діаграми (рис. 4.9). Кожна кругова діаграма відображає категорії доходів чи витрат та їхнє співвідношення одне до одного у відсотках за обраний період часу. Опис категорій надано знизу під діаграмою, а також при натисканні на кожну окрему секцію діаграми можна побачити назву цієї секції. Це дозволяє користувачам отримати розширену інформацію щодо конкретних категорій витрат або джерел їхніх доходів, допомагаючи краще зрозуміти, куди йде їхній бюджет та звідки отримуються кошти. Такий аналіз є важливим для ефективного контролю над фінансами та прийняття обґрунтованих фінансових рішень.

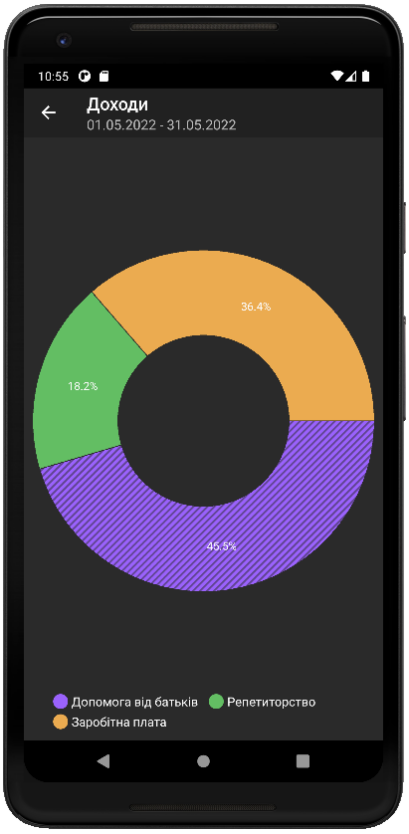

Рис. 4.9 – Кругова діаграма доходів та витрат

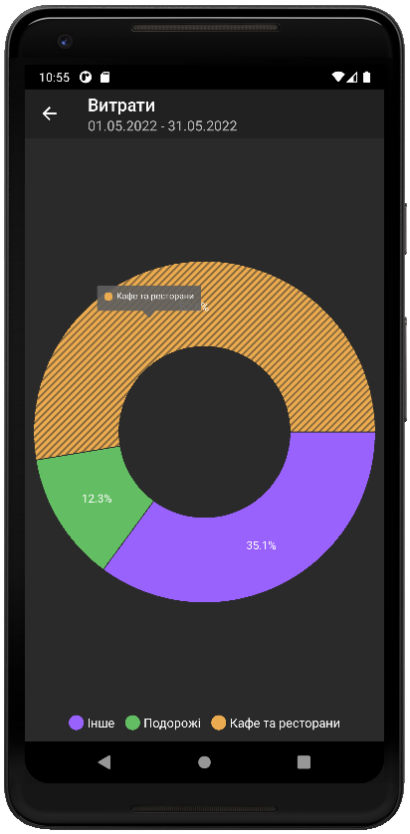

Рис. 4.10 – Кругова діаграма витрат

4.4 Формування фінансового плану

Третій пункт у навігаційній панелі відповідає за сторінку фінансового планування (рис. 4.11). Фінансове планування стає надзвичайно важливим інструментом для ефективного контролю та управління особистими фінансами. Воно допомагає чітко визначити пріоритети, щоб краще розуміти, куди спрямовувати фінансові ресурси.

На цій сторінці користувач має можливість створити фінансовий план, описавши свої місячні доходи відповідно до встановлених категорій. Також можна встановити бажані ліміти на місячні витрати для кожної з цих категорій (рис. 4.12). Це дозволяє контролювати та встановлювати межі для грошових витрат у кожній конкретній категорії, що сприяє більшій фінансовій дисципліні та ефективному управлінню бюджетом.

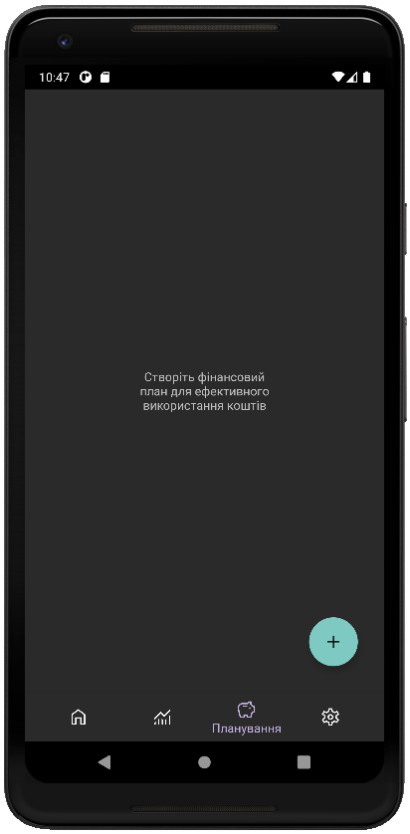

Рис. 4.11 – Сторінка фінансового планування

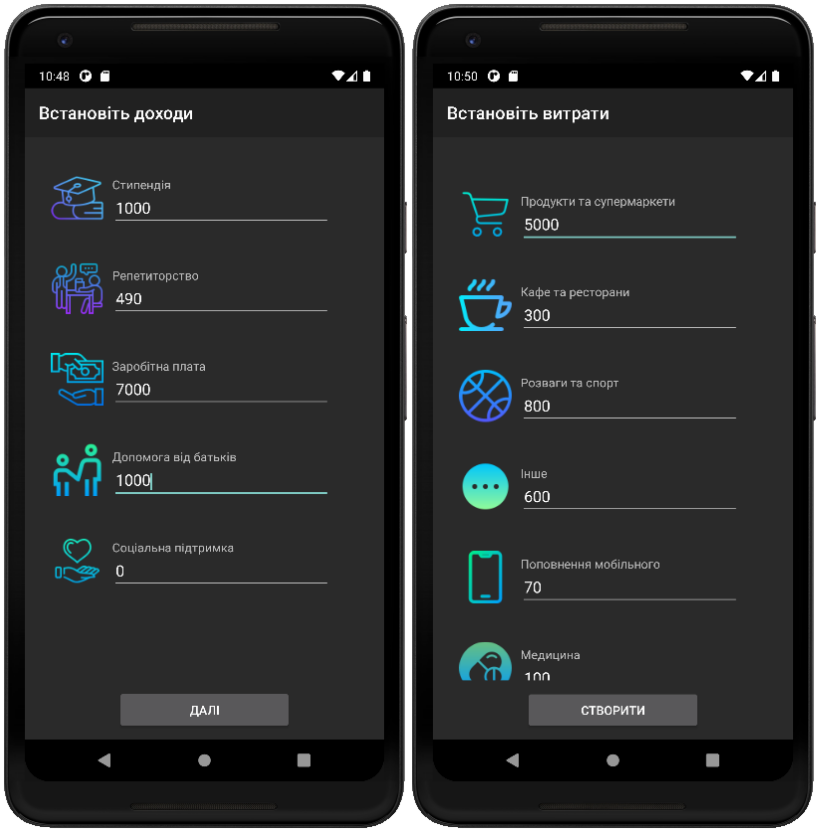

Рис. 4.12 – Встановлення місячних доходів та допустимих лімітів на витрати

Планування фінансів також дозволяє збирати дані про користувачів застосунку. Ці дані використовуються для пропозиції оптимальних лімітів витрат, які враховують фінансові можливості користувача та допомагають пристосувати план до конкретних потреб і цілей.

При подальшому внесенні нових транзакцій, їхня вартість автоматично буде врахована у відповідних категоріях у фінансовому плані, що дозволить користувачам моніторити заплановані доходи та витрати (показано на рис. 4.13). Це дає можливість у реальному часі переглядати, наскільки користувач підтримує заплановані фінансові цілі.

Крім того, коли сума витрат у певній категорії наблизиться до її ліміту чи досягне встановленого рівня доходу для цієї категорії, заплановано сповіщати користувача за допомогою повідомлень. Це допоможе своєчасно усвідомлювати, коли доходи чи витрати наблизяться до певних меж, що встановлені в фінансовому плані, і вчасно вживати відповідних заходів для ефективного управління бюджетом.

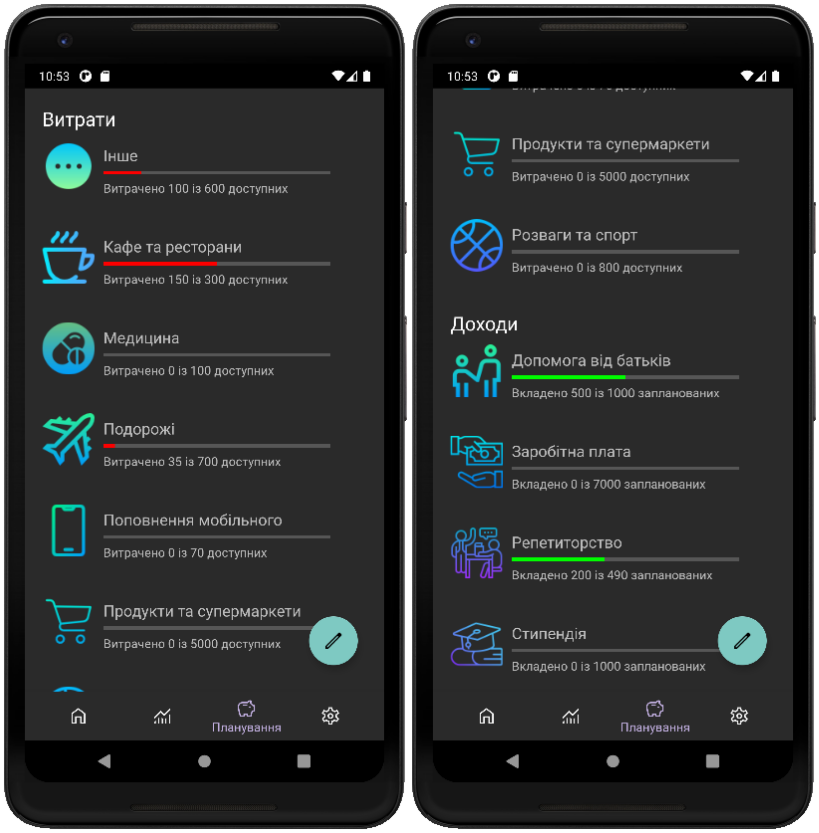

Рис. 4.13 – Стан фінансового плану

Висновки

У четвертому розділі дипломної роботи було розроблено функціонал застосунку. Результат розробки мобільного застосунку для фінансового планування, аналізу та оптимізації витрат, представлено у вигляді мінімально життєздатного продукту (MVP). Результат проектування було представлено через скріншоти роботи програми та опис можливих варіантів використання системи.

При розробці було розроблено простий та лаконічний за дизайном продукт, що визначає його високий рівень зручності та ефективності. Кожен елемент інтерфейсу розташований таким чином, щоб максимально спростити користування застосунком. Мінімалістичний дизайн дозволяє швидко орієнтуватися в інтерфейсі та зосереджуватися на головних функціях, що робить взаємодію з застосунком інтуїтивно зрозумілою для будь-якого користувача.

Було реалізовано основні функції застосунку, включаючи авторизацію через сторонні застосунки, управління гаманцями та транзакціями, формування фінансового плану та аналітику доходів та витрат користувача. Логічна організація функціоналу забезпечує зручність використання продукту. Інтуїтивно зрозумілі елементи керування, швидкий доступ до ключових опцій та послуг, а також зручна навігація роблять використання застосунку приємним та ефективним. Даний продукт є базовою основою для подальшого розвитку та розширення функціоналу для забезпечення більш широкого спектру можливостей користувачів у сфері управління власними фінансами.

Подальший розвиток застосунку може включати розширення функціоналу для більш деталізованого аналізу та звітності, автоматизації певних процесів, наприклад, регулярних платежів або рекомендацій щодо ефективного управління фінансами. Також, можливість персоналізації і налаштування відповідно до індивідуальних фінансових потреб користувача може стати одним з ключових напрямів подальшого вдосконалення програми.

ВИСНОВКИ

В ході виконання дипломного проекту було розроблено мінімально життєздатний продукт для фінансового планування, аналізу та оптимізації витрат на мобільній платформі розробки Xamarin. Даний продукт надає можливість керувати кількома гаманцями, вести облік транзакцій та аналізувати доходи та витрати користувача за вказаний період, використовуючи візуальні діаграми та графіки. Найсуттєвішою особливістю застосунку є його потужний та гнучкий механізм роботи з фінансовим плануванням, що дозволяє здійснювати раціональний контроль над особистим бюджетом з урахуванням усіх фінансових потреб та уподобань користувача.

Під час цього дослідження було проведено опитування серед користувачів для збору відгуків та визначення їхніх потреб у сфері фінансового планування та аналізу витрат через мобільні застосунки. Опитування охопило аудиторію різних вікових груп та соціальних категорій з метою отримання різноманітної та репрезентативної інформації. Результати опитування виявили чітку потребу у зручному, ефективному і простому у використанні інструменті для ведення особистих фінансів через мобільні додатки. Такий підхід дозволив врахувати пріоритети та очікування майбутніх користувачів у процесі розробки мобільного застосунку.

Перед розробкою даного рішення було також проведено ретельний аналіз поточного ринку мобільних застосунків для фінансового планування та аналізу витрат з аналогічним функціоналом. Метою цього етапу було визначити переваги та недоліки існуючих продуктів, їхню популярність серед користувачів, а також ідентифікувати необхідні функції та можливості для конкурентоспроможного продукту.

На основі отриманих відомостей був сформований перелік вимог до мобільного застосунку з урахуванням потреб та очікувань кінцевих користувачів. Ці вимоги враховують не лише технічні аспекти, а й зручність використання, інтуїтивність інтерфейсу, адаптованість до різних груп користувачів та можливості для розширення функціоналу в майбутньому.

Отриманий аналіз визначив стратегічні напрямки для подальшої розробки, гарантуючи, що розроблений мобільний застосунок буде відповідати вимогам та очікуванням користувачів, а також матиме конкурентоспроможні переваги на ринку мобільних додатків для фінансового управління.

Окрім визначення користувацьких вимог, в процесі аналізу було детально розглянуто функціональні та нефункціональні для розробки мобільного застосунку для фінансового планування. Функціональні вимоги визначали операції та можливості програмного продукту, такі як функції керування гаманцями, облік транзакцій, аналіз доходів та витрат за вказаний період, а також можливість візуального подання результатів у вигляді діаграм та графіків. У нефункціональних вимогах враховувалися аспекти продуктивності, безпеки, ергономіки інтерфейсу, а також складності та зручності використання застосунку користувачами.

Під час цього етапу розробки також були сформульовані бізнес-вимоги та системні вимоги. Бізнес-вимоги відображали бізнес-потреби та очікування від застосунку з точки зору користувачів і компанії-розробника. Визначені системні вимоги описували технічні характеристики, потрібні для нормального виконання функцій програми.

Крім того, було обрано методологію розробки, що визначило підхід до планування та організації роботи над проектом. Вибір методології є важливим етапом для успішної розробки програмного продукту. Обрана методологія допомогла визначити кроки, що потрібно виконати для успішного завершення проекту, сприяла управлінню ризиками та термінами виконання та слугувала запорукою забезпечення якісної розробки та відповідності програми вимогам користувачів у процесі її реалізації.

На наступному етапі розробки мобільного застосунку було обрано оптимальну платформу та інструменти для досягнення поставленої мети. Після уважного аналізу різних доступних варіантів було прийнято рішення про використання платформи розробки Xamarin, оскільки вона надавала широкі можливості для створення мобільних додатків у множині операційних систем, забезпечуючи оптимальну швидкість розробки та ефективність. Крім того, було ретельно продумано

архітектуру програми, розглянуто класи, їх взаємодію та визначено складові бази даних. Це стало основою для подальшої ефективної і структурованої роботи над розробкою мобільного застосунку для фінансового планування та аналізу витрат.

В ході виконання дипломного проекту вдалося створити мінімально життєздатний продукт, який відповідає основним вимогам та функціональності, необхідній для фінансового планування та управління витратами. Застосунок визначається своєю здатністю до ефективного керування кількома гаманцями, веденням обліку транзакцій із можливістю їх аналізу у вигляді діаграм та графіків. Головною перевагою стало потужне і гнучке планування фінансів, що дозволяє користувачам оптимально розпоряджатися своїм бюджетом та вносити відповідні корективи.

Впровадження цього мобільного застосунку в життя відкриє користувачам можливість не тільки ефективно використовувати власні фінансові ресурси, уникаючи ризиків їхнього неконтрольованого зменшення, але й дозволяє без зайвих зусиль та витрат часу створювати фінансовий запас на майбутнє. Застосунок створено з урахуванням потреб та вимог сучасного користувача, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс для щоденного користування.

Цей застосунок поєднує в собі потужний функціонал, який допомагає керувати фінансами у найефективніший спосіб. Завдяки його можливостям керування гаманцями, точному обліку транзакцій та візуалізації доходів та витрат у формі діаграм та графіків, користувач отримає зручний інструмент для ефективного управління бюджетом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Застосунок [Електронний ресурс] / Вікіпедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA.

2. What Is Data Collection: Types, Methods (+ Top 25 Tools) [Електронний ресурс] / Layer Blog – Режим доступу до ресурсу: https://blog.golayer.io/business/data-collection-methods.

3. Google Forms [Електронний ресурс] / Google – Режим доступу до ресурсу: https://www.google.com/forms/about/.

4. The 12 best free survey tools and form builders in 2023 [Електронний ресурс] / Zapier – Режим доступу до ресурсу: https://zapier.com/blog/best-free-survey-tool-form-app/.

5. Дослідження вимог до застосунку для фінансового планування та аналізу витрат: анкета користувачів [Електронний ресурс] / Google Docs – Режим доступу до ресурсу: https://forms.gle/ZyBAwNVf7Kvk9zCJ9.

6. What is data analysis? Methods, techniques, types & how-to [Електронний ресурс] / Bernardita Calzon – Режим доступу до ресурсу: https://www.datapine.com/blog/data-analysis-methods-and-techniques/#data-analysis-methods.

7. The 7 Most Useful Data Analysis Methods and Techniques [Електронний ресурс] / Emily Stevens – Режим доступу до ресурсу: https://careerfoundry.com/en/blog/data-analytics/data-analysis-techniques/.

8. No.1 Expense Manager Budget Planner - Money Lover [Електронний ресурс] / Finsify – Режим доступу до ресурсу: https://moneylover.me/#cta.

9. Monefy | Handy personal finance management tool for iOS and Android [Електронний ресурс] / Reflectly – Режим доступу до ресурсу: https://monefy.me/.

10. Expense Manager – Додатки в Google Play [Електронний ресурс] / Markus Hintersteiner – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=at.markushi.expensemanager.

11. User Stories and User Story Examples [Електронний ресурс] / Mike Cohn –

Режим доступу до ресурсу: https://www.mountaingoatsoftware.com/agile/user-stories.

12. What Is SDLC? Understand the Software Development Life Cycle [Електронний ресурс] / Stackify – Режим доступу до ресурсу: https://stackify.com/what-is-sdlc/.

13. The Software Development Life Cycle (SDLC): 7 Phases and 5 Models [Електронний ресурс] / Hannah Clark – Режим доступу до ресурсу: https://theproductmanager.com/topics/software-development-life-cycle/.

14. What Is Scrum: A Guide to the Most Popular Agile Framework [Електронний ресурс] / ScrumAlliance – Режим доступу до ресурсу: https://www.scrumalliance.org/about-scrum.

15. What Is Requirement Analysis: Best Practices and Examples [Електронний ресурс] / LambdaTest – Режим доступу до ресурсу: https://www.lambdatest.com/learning-hub/requirement-analysis.

16. Requirement Analysis: how to use this startup-friendly approach + a case study [Електронний ресурс] / Turgay Celik – Режим доступу до ресурсу: https://www.freecodecamp.org/news/how-to-analyze-the-requirements-of-a-new-product-a-startup-friendly-approach-and-a-case-study-833970e5c36c/.

17. What are Functional Requirements: Examples, Definition, Complete Guide: Examples, Types, Approaches [Електронний ресурс] / Visure Solutions – Режим доступу до ресурсу: https://visuresolutions.com/blog/functional-requirements/.

18. The Complete Guide To Understand IDEF Diagram [Електронний ресурс] / Wondershare EdrawMax – Режим доступу до ресурсу: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html.

19. What are Non Functional Requirements — With Examples [Електронний ресурс] / Paula Rome – Режим доступу до ресурсу: https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples.

20. How to Write a Business Requirements Document (BRD) [Електронний ресурс] / Brenna Schwartz – Режим доступу до ресурсу: https://www.projectmanager.com/blog/business-requirements-document.

21. Business requirements [Електронний ресурс] / Wikipedia – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Business_requirements.

22. System Requirements Definition - What are system requirements? [Електронний ресурс] / TechTerms – Режим доступу до ресурсу: https://techterms.com/definition/system_requirements.

23. Native, Hybrid, and Cross-Platform Apps Explained [Електронний ресурс] / Leobit – Режим доступу до ресурсу: https://leobit.com/blog/native-hybrid-cross-platform-application-development-explained/.

24. What is Flutter — Pros and Cons of Using Flutter [Електронний ресурс] / Claire D. Costa – Режим доступу до ресурсу: https://medium.com/flutter-community/pros-and-cons-of-using-flutter-1f5d1269a4b9.

25. Xamarin App Development: Advantages and Disadvantages [Електронний ресурс] / Softjourn – Режим доступу до ресурсу: https://softjourn.com/insights/xamarin-app-development-advantages-and-disadvantages.

26. Visual Studio 2022 IDE - Programming Tool for Software Developers [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: https://visualstudio.microsoft.com/vs/.

27. 3-Tier Architecture: Everything You Need to Know in a Nutshell [Електронний ресурс] / HowToDepoy – Режим доступу до ресурсу: https://www.howtodeploy.io/blog/3-tier-architecture-a-complete-guide.

28. 9 Open Banking API Examples (With Use Cases) [Електронний ресурс] / Digital Directions – Режим доступу до ресурсу: https://digitaldirections.com/open-banking-api-examples/.

29. Why use UML Class diagrams? [Електронний ресурс] / Paulina Kondratowicz – Режим доступу до ресурсу: https://synergycodes.com/blog/why-use-uml-class-diagrams/.

30. Class diagrams [Електронний ресурс] / IBM – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.

31. Introduction to class diagrams [Електронний ресурс] / Priya Pedamkar – Режим доступу до ресурсу: https://www.educba.com/class-diagram/.

32. Relational Database Benefits and Limitations (Advantages & Disadvantages) [Електронний ресурс] / DatabaseTown – Режим доступу до ресурсу: https://databasetown.com/relational-database-benefits-and-limitations/.

33. What Is a Relational Database? [Електронний ресурс] / Manuel Silverio – Режим доступу до ресурсу: https://builtin.com/data-science/relational-database.

34. Класифікація валют (ISO 4217) [Електронний ресурс] / Вікіпедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F_%D0%B2%D0%B0%D0%BB%D1%8E%D1%82_(ISO_4217).

35. ISO 18245 [Електронний ресурс] / Вікіпедія – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ISO_18245