

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СІПЕНКО

(підпис)

“ _ ” _____ 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою “Інженерія програмного забезпечення
комп’ютерних систем”**

спеціальності 121 “Інженерія програмного забезпечення”

на тему: Додаток з планування сімейного бюджету на платформі Android

Виконала: студентка 4 курсу, групи ІІ-73
(шифр групи)

Дубина Марія Сергіївна

(прізвище, ім’я, по батькові)

(підпис)

Керівник асистент Ружевський М. С.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) професор, д.т.н. Сімоненко В. П.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент доцент, к.т.н. Мелкумян К. Ю.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТРЕНКО

(підпис)

“__” _____ 2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Дубини Марії Сергіївни

1. Тема проєкту «Додаток з планування сімейного бюджету на платформі Android» керівник проєкту Ружевський Микита Сергійович, асистент, затверджені наказом по університету від «11» травня 2021 року №1139-с
2. Термін здачі студентом закінченого проєкту «31» травня 2021 р.
3. Вихідні дані до проєкту технічна документація, теоретичні дані
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) опис задання, опис та порівняння додатків-аналогів, огляд сервісів та технологій для розробки програмного продукту, опис процесу розробки та інструкція користувача
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) схема алгоритму аутентифікації, загальна діаграма моделей баз даних, діаграма взаємодії

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання «30» серпня 2020р.

Календарний план

№	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>10.12.2020-15.12.2020</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2020-15.03.2021</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2021-25.03.2021</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2021-5.04.2021</i>	
5.	<i>Програмна реалізація системи</i>	<i>5.04.2021-15.04.2021</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2021-26.05.2021</i>	
7.	<i>Захист програмного продукту</i>	<i>25.04.2021</i>	
8.	<i>Передзахист</i>	<i>23.05.2021</i>	
9.	<i>Захист</i>	<i>18.06.2021</i>	

Студент-дипломник _____
(підпис)

Марія ДУБИНА

Керівник проєкту _____
(підпис)

Микита РУЖЕВСЬКИЙ

Анотація

В бакалаврському дипломному проєкті було розроблено додаток для дослідження доходів та витрат. Додаток являє собою клієнта, який має локальну базу даних та зв'язаний з віддаленою базою даних Firebase.

Програма дозволяє досліджувати доходи та витрати за категоріями та періодами. Програмний продукт був створений на мові Dart у візуальному середовищі Android Studio зі встановленими плагінами Flutter та Dart.

Annotation

In this project for a Bachelor's Degree, the app to monitor incomes and expenses is realized. The application is a client that has a local database and is connected to a remote Firebase database.

The program allows you to research incomes and expenses by categories and periods. The software product was created in Dart language in the visual environment of Android Studio with Flutter and Dart plug-ins installed.

Відомість технічного проєкту
до дипломного проєкту
на тему: «Додаток з планування сімейного бюджету
на платформі Android»

Київ – 2021 р.

ВІДОМІСТЬ ТЕХНІЧНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проєкт	2	
2	A4	ІАЛЦ.467200.001 ВП	Відомість технічного проєкту	1	
3	A4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	4	
4	A4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	74	
5	A4	ІАЛЦ.467200.004 Д1	Алгоритм аутентифікації	1	
6	A4	ІАЛЦ.467200.005 Д2	Загальна діаграма моделей баз даних	1	
7	A4	ІАЛЦ.467200.006 Д3	Діаграма взаємодії	1	
8	A4	ІАЛЦ.467200.007 Д4	Текст програми	26	

					ІАЛЦ.467200.001 ВП				
Зм.	Арк.	№ документа	Підпис	Дата	Додаток з планування сімейного бюджету на платформі Android	Літ.	Аркуш	Аркушів	
Розробив		Дубина М. С.							
Перевірив		Ружевський М. С.					1	1	
						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІІ-73			
Н. контр.		Сімоненко В. П.							
Затверд.					Відомість технічного проєкту				

Технічне завдання
до дипломного проєкту
на тему: «Додаток з планування сімейного бюджету на
платформі Android»

Київ – 2021 р.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ.....	2
2. ПРИЧИНИ ДЛЯ РОЗРОБКИ	2
3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ	2
4. ДЖЕРЕЛА РОЗРОБКИ	2
5. ТЕХНІЧНІ ВИМОГИ	3
5.1. Вимоги до розробленого продукту	3
5.2. Вимоги до програмного забезпечення.....	3
5.3. Вимоги до апаратної частини	3
6. ЕТАПИ РОЗРОБКИ.....	4

					ІАЛЦ.467200.002 ТЗ				
Зм.	Арк.	№ документа	Підпис	Дата	Додаток з планування сімейного бюджету на платформі Android	Літ.	Аркуш	Аркушів	
Розробив		Дубина М. С.					1	4	
Перевірів		Ружевський М. С.							
						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІІ-73			
Н. контр.		Сімоненко В. П.							
Затверд.					Технічне завдання				

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Поточне технічне завдання поширюється на розробку програмного додатку “Додаток з планування сімейного бюджету на платформі Android”.
Область застосування: аналог додатку для дослідження доходів та витрат.

2. ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту по освітньо-професійної програми “Інженерія програмного забезпечення комп’ютерних систем” спеціальності 121 “Інженерія програмного забезпечення”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка додатку для планування бюджету під операційну систему Android з використанням кросплатформеного фреймворку Flutter.

4. ДЖЕРЕЛА РОЗРОБКИ

В сучасному світі доволі велика кількість аналогів даного додатку, серед яких Spendee, Monefy та інші, що стали натхненням і забезпечили новими ідеями щодо створення додатку. Додатковими джерелами розробки є науково-технічна література з теорії і практики програмування, алгоритмів, шаблонів і публікації в Інтернеті з даних питань.

					ІАЛЦ.467200.002 ТЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		2

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Клієнтська частина із комфортним і зрозумілим користувацьким інтерфейсом, що потребує авторизації користувача.
- Можливість збереження даних у віддалену базу даних.
- Можливість роботи без наявного підключення до мережі Internet через синхронізацію даних з віддаленої бази даних та реляційною базою на пристрої.
- Незалежність використання додатку користувачем, маючи лише доступ до Інтернету та незалежність розробки інженером, маючи лише мінімальний набір кросплатформних інструментів.
- Актуальність методів розробки програмного забезпечення та дотримування тону чистої архітектури та чистого коду.

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 7 SP1 або вище.
- Windows PowerShell 5.0 або вище.
- Android Studio або VSCode.
- Flutter SDK.

5.3. Вимоги до апаратної частини

- Операційна система Android 4.1 або вище.
- Доступ до Інтернету.
- Оперативна пам'ять не менше 1 Гбайт.
- Вільне місце на жорсткому диску не менше 200 Мбайт.

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ документа	Підпис	Дата		

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	05.01.2021
Аналіз існуючих рішень	01.02.2021
Вибір технологій і архітектури	15.03.2021
Побудова графічних матеріалів і діаграм	01.04.2021
Написання практичної частини диплома	05.04.2021
Написання теоретичної частини диплома	15.04.2021

					ІАЛЦ.467200.002 ТЗ	Арк.
						4
Зм.	Арк.	№ документа	Підпис	Дата		

Пояснювальна записка
до дипломного проєкту
на тему: «Додаток з планування сімейного бюджету на
платформі Android»

Київ – 2021 р.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	4
ВСТУП	5
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ДЛЯ ПЛАНУВАННЯ СІМЕЙНОГО БЮДЖЕТУ	6
1.1. Опис завдання	6
1.2. Поняття бюджету	6
1.3. Можливості банківських додатків для дослідження своїх доходів та витрат	8
1.4. Огляд аналогів	9
1.4.1. Додаток «Spendee»	9
1.4.2. Додаток «Money Lover»	12
1.4.3. Додаток «AndroMoney»	14
1.4.4. Додаток «OneCoin»	16
1.4.5. Додаток «Monefy»	18
1.5. Порівняння аналогів із розроблюваним додатком	21
Висновок до розділу 1	22
РОЗДІЛ 2. ОГЛЯД СЕРВІСІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ДОДАТКУ	23
2.1. Сервіси для створення дизайн-проекту	23
2.1.1. NinjaMock	24

					ІАЛЦ.467200.003 ПЗ			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Дубина М. С.			Додаток з планування сімейного бюджету на платформі Android	Літ.	Аркуш	Аркушів
Перевірив		Ружевський М. С.					1	74
Н. контр.		Сімоненко В. П.			Пояснювальна записка	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІ-73		
Затверд.								

2.1.2. Moqups.....	24
2.1.3. Creately	25
2.1.4. Figma.....	26
2.2. Технології для розробки Android-застосунків	27
2.2.1. Інструменти для нативної розробки	27
2.2.1.1. Android SDK	28
2.2.1.2. Android NDK	29
2.2.2. Фреймворки для кросплатформеної розробки	29
2.2.2.1. Flutter	30
2.2.2.1.1. Dart.....	33
2.2.2.2. React Native.....	34
2.2.2.2.1. JavaScript.....	35
Продовження таблиці 2.2	36
2.2.2.3. Xamarin	36
2.2.2.3.1. C#	38
2.2.2.4. Порівняння корсплатформених фреймворків.....	39
2.3. База даних мобільного додатку.....	40
2.3.1. Реляційна СУБД SQLite.....	41
2.3.2. Нереляційна база даних Firebase.....	43
Висновок до розділу 2	44
РОЗДІЛ 3. ДЕТАЛІ РЕАЛІЗАЦІЇ ДОДАТКУ	45
3.1. База даних	45
3.1.1. Налаштування Firebase-проекту.....	45
3.1.2. Локальна база даних SQLite	47

3.2. Розробка клієнтської частини	48
Висновок до розділу 3	55
РОЗДІЛ 4. ІНСТРУКЦІЯ КОРИСТУВАЧА.....	56
4.1. Реєстрація та авторизація	56
4.2. Робота з транзакціями	59
4.3. Перегляд статистики.....	63
4.4. Налаштування категорій	66
Висновок до розділу 4	69
ВИСНОВОК.....	70
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	71

ПЕРЕЛІК СКОРОЧЕНЬ

СУБД	Система управління базами даних
AoT	Ahead-of-time
API	Application programming interface
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
JiT	Just-in-time
JNI	Java native interface
JSON	JavaScript object notation
JSX	JavaScript XML
JVM	Java virtual machine
MacOS	Macintosh operating system
NDK	Native development kit
NoSQL	Not only SQL
SDK	Software development kit
SQL	Structured query language
UWP	Universal windows platform
UX	User experience
WPF	Windows presentation foundation
XML	Extensible markup language

ВСТУП

Головою метою даної дипломної бакалаврської роботи була розробка Android-додатку для ведення особистої або ж сімейної бухгалтерії.

Дехто ще досі перекручено розуміє сенс ведення бюджету, асоціюючи його суто з економією, яка сприймається дуже негативно, як ніби це щось ганебне. Або думає, що бюджет потрібен тільки тим, у кого мало грошей. Проте, чим більше у тебе грошей, тим більше потрібен бюджет.

Справа в тому, що бюджет потрібен не для того, щоб економити, а для того щоб все контролювати. Це означає, що не тільки зменшення витрат, але і їх збільшення. У фінансах повинен бути порядок, щоб все було чітко розкладено по полицях. Тоді буде простіше ставити собі цілі і досягати їх.

Ведення бюджету дозволяє більш ефективно і головне свідомо розподіляти кошти. Нікого ж не дивує наявність в фірмі бухгалтера, який зводить дебіт з кредитом і вирішує, куди повинні піти гроші. У сімейному бюджеті так само, це фінансовий план сім'ї.

Дорога в тисячу миль починається з першого кроку, а подорож до фінансової свободи з простого записування щомісячних витрат.

Для обліку своїх витрат, а також доходів, можна використовувати спеціальні блокноти або ж зошити, проте їх не завжди носять з собою, а інформація про витрачені кошти може забутися. Тут на допомогу приходять смартфони із додатками для ведення бюджету, які майже завжди під рукою.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		5

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ДЛЯ ПЛАНУВАННЯ СІМЕЙНОГО БЮДЖЕТУ

1.1. Опис завдання

Завданням цієї роботи є розробка програмного додатка для платформи Android для планування сімейного бюджету за допомогою кроссплатформеного мобільного фреймворку Flutter.

Головна ідея додатку полягає у можливості дослідження доходів та витрат шляхом додавання грошових потоків у додатку та перегляді інфографіки.

Користувач створює аккаунт в якому він керує своїми грошовими потоками. Грошові потоки поділяються на доходи та витрати. Доходи та витрати у свою чергу поділяються на категорії, які можна редагувати, додавати та видаляти за власним бажанням. Також категорії можна не видалити, а просто відключити і включити, коли вона знову стане потрібна. Кожна створена транзакція додається до шаблонів, які можна буде в майбутньому використовувати для полегшення створення транзакцій.

В додатку присутня функція регулярних платежів, які можуть повторюватись за певний проміжок часу. Кількість таких платежів можна ввести вручну або ж зазначити дату закінчення цих операцій.

1.2. Поняття бюджету

Прообраз бюджетних відносин почав зароджуватись з розвитком перших цивілізацій, коли вони почали оформлятися у держави. Але пройшло багато століть, перш ніж бюджет став невід'ємним атрибутом держави, а необхідність його формування була усвідомлена людьми.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		6

Сьогодні під терміном «бюджет» розуміють кілька понять. Сучасна економіка вкладає в нього подвійний зміст. З одного боку – це розподіл доходів та витрат людини, родини, міста, країни за певний термін. По-друге, це сума грошових коштів у розрахунку на деякий період часу [1].

Сімейний бюджет – це план регулювання грошових доходів і витрат сім'ї, що складається зазвичай на місячний строк у вигляді таблиці, баланс сімейних доходів і витрат, це фінансовий план, підсумовані доходи і витрати родини за певний період часу [2].

Структура сімейного бюджету дуже різноманітна. Вона включає в себе безліч пунктів. Для початку розглянемо найбільш великі розділи, а саме доходи та витрати [3]. Баланс саме цих грошових потоків потрібно дотримуватися кожному, хто ставить за мету створення збалансованого або ж профіцитного бюджету. Тоді вдасться жити за коштами і навіть робити великі покупки.

Далі кожен великий розділ бюджету розділяється. Структуризація може бути абсолютно різною, на розсуд того, хто планує витрати в сім'ї.

Доходи, зазвичай, складаються із трьох, найбільше – п'яти пунктів, проте може бути і більше в залежності від складу сім'ї або ж наявних типів заробітку. До доходів можуть відноситись заробітна платня, стипендія, пенсія, соціальна допомога, подарунки, грошові компенсації тощо.

Структура витрат сімейного бюджету є більш складною, ніж доходів, оскільки в кожній сім'ї досить різні статті потреб [4]. Для кращого контролю витрат можна певним чином їх класифікувати. Наприклад, за значимістю – виділити обов'язкові витрати, без яких неможливо прожити, бажані, без яких за потреби можна обійтись, та розкішні. Кожна сім'я може обрати власну класифікацію витрат, яка краще підходить під її фінансову ситуації або ж яка більш легка для структуризації.

Для планування сімейного бюджету необхідно провести детальний аудит всіх витрат і доходів членів сім'ї [5]. Якщо це важко зробити вручну, варто взяти собі за правило користуватись спеціальними сервісами, які дозволяють аналізувати домашню бухгалтерію. До таких сервісів відносяться і мобільні додатки, які дозволять ще ефективніше стежити за власними та сімейними грошовими потоками, оскільки в сучасному світі телефон поповнив список речей, наявність яких перевіряють перед виходом з домівки.

1.3. Можливості банківських додатків для дослідження своїх доходів та витрат

Незмінним помічником для обліку особистих грошових потоків у більшості користувачів були й досі залишаються банківські додатки.

Всі функції додатків відкриті відразу, що є їх безперечною перевагою. Такі додатки дозволяють слідкувати за всіма транзакціям по всім картам в межах одного банку. Категорія витрат визначається автоматично, в залежності від типу контрагенту, якому ці кошти надходять. Доходи, зазвичай, відмічаються двома категоріями:

- цільові зарахування – це усі надходження, що поступили на карту з конкретними цільовим призначенням як то заробітна плата, пенсія, стипендія, соціальна допомога тощо;
- нецільові зарахування – усі надходження без цільового призначення. До них відносяться перекази, зарахування коштів із вкладів, надходження від поповнення в терміналах самообслуговування чи касі відділення [6].

До недоліків можна віднести відсутність можливості фіксування готівки, оскільки такі додатки фіксують грошові потоки по рахунках, що

відкриті саме в цьому банку. З цим зв'язаний ще один недолік – при відкритті рахунку в новому банку потрібно встановлювати новий додаток. При встановленні великої кількості банківських додатків виникають такі проблеми, як:

- нестача пам'яті на девайсі;
- необхідність запам'ятовувати великої кількості паролів, оскільки для безпеки особистих даних потрібно для кожного додатку становити унікальний пароль;
- необхідність ознайомлюватись з новим інтерфейсом.

1.4. Огляд аналогів

Наразі існує багато подібних додатків для планування особистого бюджету, оскільки людина не вперше стикається з проблемою контролю бюджету. Всі додатки мають різну функціональність, однак головне залишається незмінним – додаток створює гаманець або ж рахунок, до якого користувач вписує свої доходи, витрати та перекази.

Додатки-аналоги, в основному, розроблені під обидві мобільні платформи (Android і iOS) та деякі з них підтримують версію для комп'ютера. Додатки, що розглядаються в цій роботі безкоштовні для користувачів Android. Деякі представлені додатки мають ще й повну версію, яка коштує 4-5\$.

1.4.1. Додаток «Spendee»

Spendee [7] представлений як додаток для економії. Він дозволяє керувати власними коштами за допомогою гаманця до якого вписуються всі доходи та витрати.

В додаток для Android можна ввійти за допомогою акаунту Google або Facebook або ж зареєструватись з використанням електронної пошти. При останньому варіанті потрібно вибрати пароль, який має складатись мінімум із восьми символів. Також у додатку є версія для браузера.

В безкоштовній версії доступний один гаманець та операції додаються вручну. В повній версії додатку можна створювати додаткові гаманці та ділитися ними з іншими, що дозволяє переглядати та керувати не тільки своїм бюджетом, а загальним по всім гаманцям. Також в повній версії додатку можна синхронізувати гаманець з онлайн-банком, електронним гаманцем або крипто гаманцем, що дозволяє полегшити ввід даних.

Всі доходи та витрати розділені по категоріях. Для витрат у програмі передбачено сімнадцять іменованих категорій, а для доходів – вісім.

Для всіх категорій встановлено певну іконку та колір, яким категорія відмічається в інфографіці. В налаштуваннях можна змінити іконку та колір відображення для кожної категорії. Якщо певна категорія не потрібна – її можна видалити, або навпаки, якщо не вистачає – створити нову.

Категорії одного грошового потоку можуть об'єднуватись між собою. Ця функція схожа з видаленням, оскільки після об'єднання залишається тільки одна із категорій з її початковою назвою.

Для контролю витрат за певними категоріями можна створити спеціальний бюджет. В ньому визначається від одної до сімнадцяти категорій, встановлюється певна сума на період часу (день, тиждень, два тижні, місяць, рік або власноруч обраний період). Якщо витрати по обраних категоріях перевищують суму бюджету, то буде висвітлено

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		10

повідомлення із зазначенням суми, яка не ввійшла у виставлені грошові рамки. В безкоштовній версії можливо створити лише один такий бюджет.

Окрім доходів та витрат додаток передбачає можливість грошового переказу. В повній версії додатку перекази можуть відбуватися між двома гаманцями, а в безкоштовній версії вони відмічаються просто як вхідні та вихідні потоки коштів із гаманця.

Робота з додатком починається з головної сторінки, де надалі будуть відображатися всі доходи та витрати за обраним гаманцем. Поки на цій сторінці є кнопка для введення витрат вручну та кнопка для підключення до банківського рахунку. При спробі підключити банківський рахунок користувача перенаправляє на сторінку вибору банку, після чого пропонує обрати план «Spendee Premium».

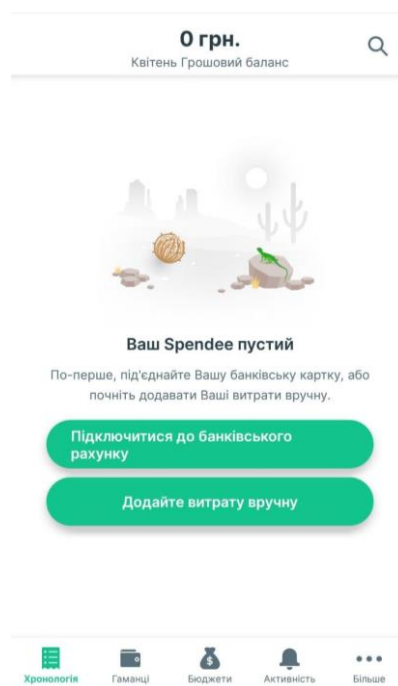


Рисунок 1.1 – Головна сторінка

Для створення відмітки про доходи або витрати потрібно обрати відповідний вид грошового потоку, категорію, з або до якого гаманця

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		11

відбулася транзакція, дату та суму. Також можна додати нотатку, фото або встановити повтор операції.

Для введення переказу потрібно обрати з та до якого гаманця відбуватиметься переказ, дату та суму. Як із витратами та доходами можна додати нотатку та встановити повтор.

1.4.2. Додаток «Money Lover»

Money Lover – це додаток для ведення обліку готівки [8]. В додатку можна під'єднати банківські рахунки, але як і в Spendee для цього потрібна повна версія. Вхід відбувається за допомогою облікових записів Facebook або Google або після стандартної реєстрації через електронну пошту. Під час першої авторизації створюється перший гаманець – встановлюється назва та валюта. Окрім мобільної версії доступні також версії для браузера та комп'ютера.

Для створення категорії спочатку потрібно обрати гаманець в якому вона буде присутня, а лише потім назву та іконку.

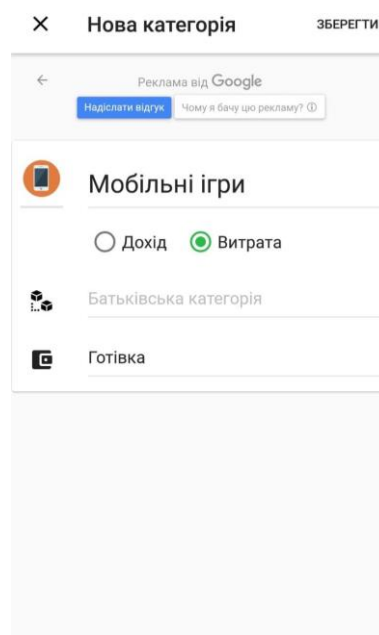


Рисунок 1.2 – Створення нової категорії

У вкладці створення категорії можна також створити підкатегорію. Для цього потрібно вибрати батьківську категорію серед наявних. Створення підкатегорій дозволяє більш детально структурувати всі позиції витрат.

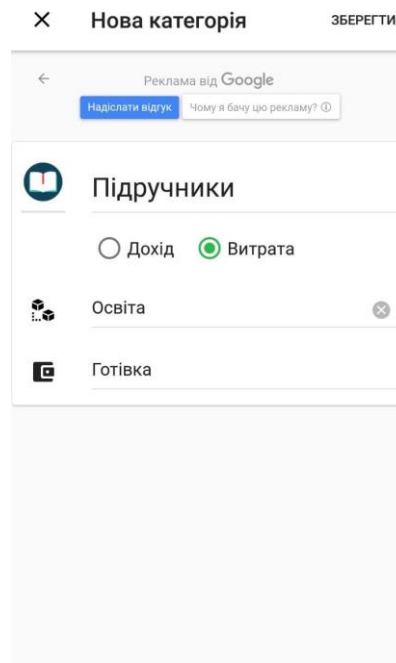


Рисунок 1.3 – Створення підкатегорії

Грошові потоки в додатку поділяють на доходи, витрати та борги або кредити. Вони у свою чергу поділяються на різні категорії з під категоріями, як можна додавати під час введення даних, а не тільки в налаштуваннях.

В базовій версії створюється лише один стандартний гаманець. В повній версії можна додавати стандартні гаманці, для кредитів та зв'язані з банківською картою.

З повною версією відкривається можливість експорту даних гаманця в Excel таблицю.

В додатку передбачена функція нагадування про виплату введених заборгованостей у визначений день. Також програма має щоденні

нагадування у визначений час (зазвичай вечірній, але можна встановити будь-який зручний час), щоб користувач вводив свої доходи та витрати кожного дня. Такі нагадування зменшують можливість того, що якась операція готівкою не була зафіксована.

Серед основних функцій є можливість створення події. Початком події стає день її створення, а день закінчення обирається користувачем. Всі введені доходи та витрати, яким додатково встановлена певна подія, можна переглянути в її фінальній аналітиці. Для всіх подій підсумовується баланс і в залежності від того чи він додатний, чи від’ємний показує, що під час події було зароблено або ж витрачено певну суму. Дана функція гарно підходить для відстеження грошових потоків під час подорожі або ж інших схожих випадків.

Для контролю витрат в певній категорії можна створити бюджет та встановити максимальну суму і період за який сума може бути витрачена. За кожним бюджетом будується інфографіка, показується скільки днів та яка сума залишились до завершення бюджету та наскільки бюджет перевищений, якщо так уже сталося.

Загалом, Money Lover навіть у базовій версії дозволяє працювати з більшістю своїх функцій, однак вагомим недоліком цієї версії є велика кількість реклами.

1.4.3. Додаток «AndroMoney»

AndroMoney – це продвинутий інструмент для обліку особистих фінансів [9]. Додаток дозволяє записувати доходи, витрати та перекази в різних валютах, вести облік заборгованостей і доступного ліміту по кредитним карткам, створювати бюджети та багато-багато іншого.

Великим плюсом даного додатку є можливість створювати необмежену кількість гаманців, які в AndroMoney називаються акаунтами. Вони можуть бути трьох типів: готівкова, кредитна карта та банк. Однак функції підключення до гаманців своїх банківських рахунків немає.

Серед корисних функцій базової версії додатку є збереження бази даних додатку на Google диску. У випадку втрати даних або зміні девайсу дані легко відновлюються.

Створення бюджету відбувається одразу для всіх категорій, тобто вводиться певна сума і для кожної категорії вона стає максимальною на місяць. Після введення суми бюджету на місяць одразу відображається бюджет на рік, тиждень та день. Присутня функція повідомлення про перевищення бюджету та можна встановити певний поріг в межах бюджету (певний відсоток) при досягненні якого також з'являється спливаюче вікно з повідомленням.

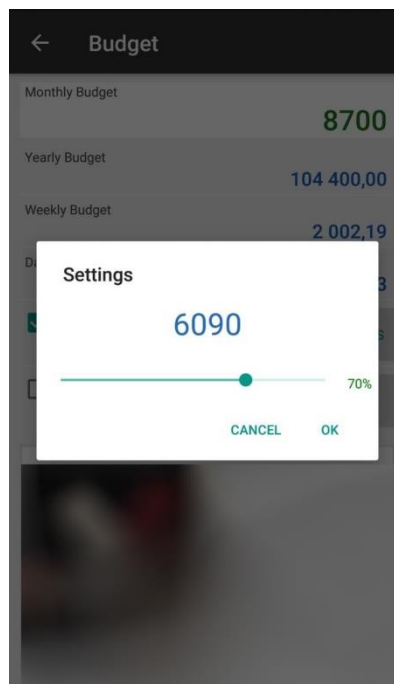


Рисунок 1.4 – Створення бюджету та встановлення відсотку

Додаток передбачає підвищення рівню безпеки обліку фінансів за допомогою паролю, який можна включити в налаштуваннях. Також у налаштуваннях можна змінити тему калькулятора, колір фону додатку, встановити початок місяця на бажане число, формат дати та числа та інше.

Мінусом даного додатку можна визнати відсутність функції розділу гаманців у будь-якій версії. Також до мінусів можна віднести інтерфейс, який досить тяжкий для розуміння. Авторизація в додатку відбулась за допомогою синхронізованого з телефоном облікового запису Google ще до безпосереднього входу, однак як змінити обліковий запис – не зовсім зрозуміло.

1.4.4. Додаток «OneCoin»

OneCoin – повністю безкоштовний додаток для контролю доходів та витрат [10].

При першому вході в додаток потрібно вказати своє ім'я або ж певний логін та вказати цифровий пароль, який зазвичай буває у банківських картках, довжиною в чотири цифри. Після цього відкривається головна сторінка основного рахунку (гаманця).

В даному додатку можна створювати довільну кількість рахунків. Для кожного рахунку можна змінити назву, видаляти ж можна лише ті, що були створені користувачем – перший основний рахунок видаляти не можна.

Доходи та витрати додаються через спеціальну кнопку внизу екрану. Після натискання на дану кнопку користувача перенаправляє на екран створення нової грошової транзакції. На цій сторінці мають бути дві кнопки для зміни грошового потоку.

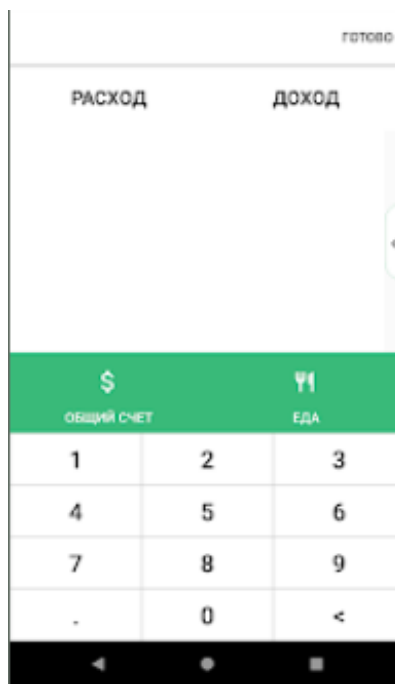


Рисунок 1.5 – Скріншот з Google Play Store вкладки для створення транзакції [10]

Спочатку було не зрозуміло яким чином обирати дохід це чи витрата, оскільки на моєму девайсі дані кнопки відсутні.

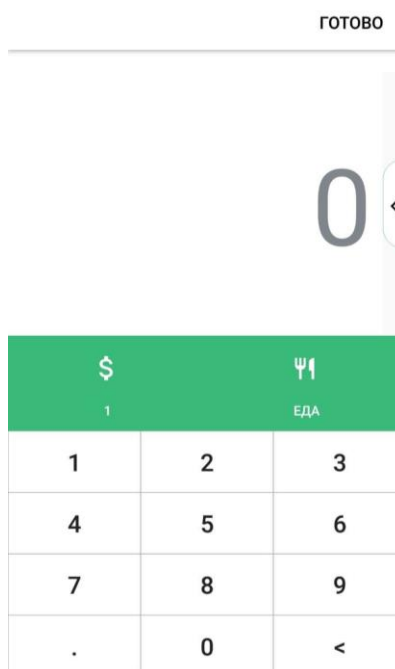


Рисунок 1.6 – Скріншот вкладки для створення транзакції з власного девайсу

Було припущення, що дану функцію деактивували в останній версії додатку. Проте вона все ж працює. Для зміни грошового потоку потрібно натискати там, де мали б бути кнопки.

Також не зрозумілим моментом стало те, що при переході на сторінку створення грошової транзакції з певного рахунку, він не вибирається автоматично як такий з якого чи до якого надходять гроші, а встановлюється основний. Можливо це зумовлено тим, що основний рахунок є своєрідною функцією контролю над усіма грошовими потоками, оскільки в ньому відображаються всі доходи та витрати з усіх наявних рахунків.

Створену транзакцію не можна скоригувати, лише видалити.

Всі категорії доходів та витрат є незмінними.

При повторному вході в додаток потрібно ввести пароль або ж відсканувати відбиток пальця, за наявності відповідного сканеру на девайсі. При введенні паролю додаток без проблем продовжує працювати, однак при спробі ввійти за допомогою відбитку програма припиняє свою роботу.

1.4.5. Додаток «Monefy»

Monefy – додаток для планування бюджету та обліку витрат [11].

Робота в додатку починається з основної сторінки усіх рахунків. На цій сторінці присутні дві кнопки «Дохід» та «Витрата» для введення однойменних грошових потоків. Надалі дані кнопки мають позначки «+» та «-». Також для додання витрат можна натиснути на одну із представлених категорій. У різних рахунків можуть бути представлені різні категорії, однак з часом будуть представлені найчастіше обрані.

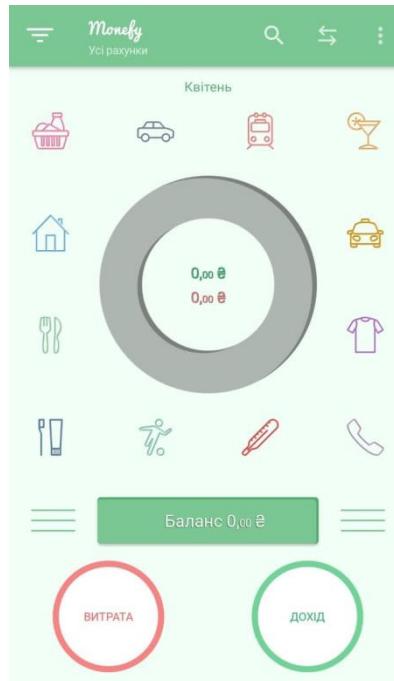


Рисунок 1.7 – Головна сторінка

В додатку одразу представлені два рахунки: готівковий та «Платіжна карта». В обох рахунків функція синхронізації з банківським акаунтом відсутня. При створенні нового рахунку для нього обирається спеціальна іконка. Спочатку було припущення, що ця іконка якось впливає на тип рахунку, однак цього не відбувається. Можливо це відноситься до розширених функцій у повній версії додатку.

Перекази відбуваються лише між рахунками в додатку. Доходи та витрати поділяються на певні статичні категорії, які, в безкоштовній версії додатку, можна лише видаляти.

Присутня функція створення бюджету. Він створюється один на всі рахунки та категорії. Бюджет відображається на спеціальній вкладці, яка об'єднує всі транзакції по рахункам. Суть полягає в тому, що для бюджету встановлюється сума, яка має бути витрачена за місяць, тому у вкладці «Усі рахунки» при перевищенні зазначеного бюджету баланс буде від'ємним навіть, якщо у решти баланс додатній.

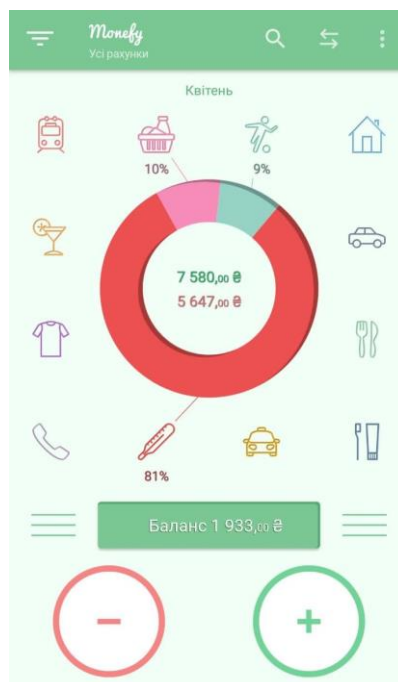


Рисунок 1.8 – Вкладка «Усі рахунки» із ввімкненою функцією бюджету

Якщо переглядати дану вкладку з відключеною функцією бюджету, то баланс буде рахуватись як сума балансів усіх рахунків.



Рисунок 1.9 – Вкладка «Усі рахунки» із вимкненою функцією бюджету

У повній версії додатку присутня функція створення декількох облікових записів, що дозволяє використовувати рахунки спільно. Також ця версія надає можливість синхронізувати дані додатку через Dropbox або Google Drive, створювати резервні копії даних, встановити пароль, змінювати наявні та додавати нові категорії доходів та витрат.

1.5. Порівняння аналогів із розроблюваним додатком

Загалом розроблюваний додаток має функціональність, яка відсутня в тих чи інших додатках-аналогах або ж доступна лише у повних версіях, проте яка дозволяє більш зручно керувати власними грошовими потоками, а також сімейними або ж груповими (таблиця 1.1).

Таблиця 1.1 – Порівняння безкоштовних версій додатків

Функціональність	Spendee	Money Lover	Andro-Money	One-Coin	Monefy	Мій додаток
Перегляд даних за різними періодами	+	+	+	-	+	+
Регулярні платежі	-	+	+	-	-	+
Редагування категорій	+	+	+	-	-	+
Шаблони транзакцій	-	-	+	-	-	+
Зміна користувача	+	+	-	-	-	+

Висновок до розділу 1

Можна відзначити, що існує досить велика кількість різних додатків, які допомагають контролювати доходи та витрати. Усі вони надають можливість відмічати свої грошові надходження та трати, деякі з них окремо виділяють перекази.

Для більш детального обліку грошових потоків, всі додатки розподіляють їх на категорії, а Money Lover ще й на підкатегорії. Більшість програм дозволяють видаляти, коригувати уже наявні категорії та створювати нові, проте Monefy надає таку можливість лише у повній версії додатку, а в OneCoin взагалі дані функції не передбачені.

При додаванні нової транзакції у всіх додатків загалом схожий функціонал. Для цього, в основному, потрібно обрати категорію, гаманець, та ввести суму. Всі додатки, окрім OneCoin, при введенні суми транзакції мають спеціальний калькулятор. Дипломний проєкт має доволі скромний функціонал введення суми. Наразі, при введенні доходу або витрати, потрібно вводити загальну суму одразу. Відсутність даної функції не можна однозначно назвати мінусом, оскільки при покупках видають чек, який має остаточну суму, а доходи мають спеціальну виписку з результуючою сумою.

Для відображення інфоргафіки майже у всіх додатків транзакції можна відсортувати по періоду та наявним гаманцям. Це досить корисна функція, яка дозволяє більш детально відстежити трати та доходи.

Загалом, переглянуті додатки мають як недоліки так і переваги, в порівнянні з даною дипломною бакалаврською роботою. Проте, варто зазначити, що більшість переваг в плані інструментарію відносяться до платних функцій додатків.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		22

РОЗДІЛ 2. ОГЛЯД СЕРВІСІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ДОДАТКУ

2.1. Сервіси для створення дизайн-проєкту

Важливими етапами в створенні мобільних додатків є розробка прототипів і створення дизайн-макетів. Прототип необхідний для чіткого визначення структури додатку, кількості сторінок, обов'язкових модулів і окремих елементів. На його основі створюється дизайн. При наявності готових прототипів скоротиться час на розробку технічного завдання та створення дизайну проєкту [12].

Прототип – схема взаємодії всіх сторінок додатку. Його можна описати на папері у вигляді тексту, або намалювати у вигляді схем і блоків [13].

Ручка і папір – старі та перевірені інструменти для зберігання ідей, але спроба внести навіть мінімальні правки в намальований макет перетворюється на справжнісіньке випробування. Сервіси для прототипування дозволяють перенести ідею на робоче «полотно», створивши макет майбутнього проєкту.

Задачею даного етапу є створення інтерактивного прототипу майбутньої програми або сервісу. Прототип сайту або програми являє собою набір екранів, який виглядає майже як готовий додаток. На ньому видно розташування елементів, можна переходити по посиланням, на кнопки можна натискати. Прототип дає чітке розуміння того, як буде влаштований майбутній продукт, які можливості варто додати, а від яких відмовитися.

Після створення прототипу додатку створюють дизайн-макети сторінок.

Дизайн-макет – це візуальний образ майбутнього продукту. У дизайн-макеті будуються всі елементи: текст, заголовки, картинки, кнопки та ін. Ця інформація додається відповідно до розробленого прототипу [14].

2.1.1. NinjaMock

NinjaMock [15] – інструмент для розробки прототипів сайтів. У візуальному редакторі можна створити макет сторінки, що дозволяє наочно побачити, як вона буде виглядати. Прототип містить текстові блоки, зображення та інші складові елементи додатку. При необхідності в макет легко можна внести зміни.

Система працює тільки в браузері. Версії для комп'ютерів і додатків для мобільних пристроїв не існує. Інтерфейс доступний тільки англійською мовою. Це не викликає великих складнощів у користувачів, оскільки вся основна робота ведеться в візуальному редакторі.

У безкоштовній версії можна створювати до трьох проєктів. Створені об'єкти будуть знаходитися в публічному доступі. Для розробки більшої кількості макетів і роботи з приватними об'єктами потрібно оформити платну підписку на сервіс. Основний функціонал у платній і безкоштовній версіях не відрізняється.

2.1.2. Moqups

Moqups – це онлайн-редактор для створення прототипів сайтів і додатків [16]. Проєктування відбувається за простим сценарієм.

Бібліотека Moqups містить сотні елементів, які можна використовувати для «конструювання» сайтів, сервісів, настільних і

мобільних додатків. За допомогою панелі з елементами можна легко перетягувати їх в робочу область, потім міняти розміри, відкривати меню властивостей, змінювати форму, кольори та інші параметри.

Безкоштовний тариф дозволяє працювати з одним проєктом, і використовувати до трьохсот елементів, для невеликих завдань цього цілком достатньо.

2.1.3. Creately

Creately – редактор для побудови більш ніж п'ятдесяти видів діаграм і схем [17]. Вбудовані функції сервісу дозволяють створювати блок-схеми, макети сайтів, організаційні діаграми, моделі бізнес-процесів. Є підтримка колективної роботи в режимі реального часу. Сервіс можна використовувати онлайн або встановити на комп'ютер.

Редактор підходить для спільної віддаленої роботи, в якій можуть брати участь до п'ятисот учасників. Сервіс може використовуватися як інструмент для планування, проектування, структурування та презентації інформації.

У сервісі представлено понад тисячу графічних фрагментів і понад п'ятдесят шаблонів діаграм, які можна редагувати за допомогою вбудованих інструментів. Можливості сервісу дозволяють змінювати шрифт текстів, масштабувати об'єкти, застосовувати різні кольори. Додавання і переміщення об'єктів здійснюється за допомогою миші.

Безкоштовний тариф наявний лише в онлайн-версії. В рамках даного тарифу можна створити до п'яти проєктів.

2.1.4. Figma

Figma – це графічний редактор для веб-дизайну [18]. У Figma можна створювати:

- інтерактивні прототипи сайтів і мобільних додатків;
- елементи інтерфейсу – іконки, кнопки, меню, вікна, форми зворотного зв'язку;
- векторні ілюстрації.

У Figma всі документи зберігаються в хмарі. Завдяки цьому в редакторі можна колективно працювати над макетами та відкривати їх за посиланням, без скачування.

Можна заходити в Figma через браузер або ж завантажити програму на комп'ютер. Вона підходить як для Windows, так і для Mac. У десктопній версії можна працювати офлайн, а коли з'являється доступ до інтернету, зміни синхронізуються.

Як і в Google Docs, у Figma можна працювати над документом разом: давати доступ на перегляд і редагування, паралельно працювати над макетом – на екрані буде видно курсори різного кольору. Завдяки цьому всі учасники проєкту краще розуміють контекст.

Інструмент безкоштовний для індивідуальних користувачів. Просто реєструєтесь і починаєте працювати. Можна давати доступ на перегляд документа – це теж безкоштовно.

У Figma кожен проєкт – це папка. У ній команда зберігає всі файли, які відносяться до одного проєкту. Це зручно, якщо в кожному проєкті десятки файлів і ви не хочете в них плутатися. На безкоштовному тарифі ви зможете створити максимум три проєкти [19].

2.2. Технології для розробки Android-застосунків

Мобільні додатки давно є мейнстрімом для віддаленого обслуговування клієнтів і надання сервісів. Вони активно витісняють повнофункціональні веб-додатки в багатьох категоріях. Це обумовлено різними причинами, серед яких головною є динамічна природа нашого часу, коли все потрібно робити швидко і на бігу – мобільний завжди під рукою, а ось повноформатний комп'ютер доступний набагато рідше.

У поточний момент мобільна розробка відбувається під дві основні платформи Android і iOS. Кожна з платформ має свою специфіку, але в загальному має схожий функціоналом. Написання додатків під ці платформи дозволяє забезпечити найкращий UX, відповідний актуальним гайдлайнам кожної з платформ, але писати код програми доведеться двічі – під Android і під iOS. Для того, щоб не писати код двічі з'явилися кросплатформені рішення [20].

2.2.1. Інструменти для нативної розробки

Нативна розробка – це класичне рішення, яке вимагає написання програми під кожен платформу окремо, використовуючи різні мови і з огляду на особливості кожної платформи. При створенні декількох версій над проєктом одночасно працюють кілька команд.

Нативна розробка для Android означає, що додаток призначений тільки для Android. Для розробки в екосистемі, розробленої спеціально для операційної системи Android, можна використовувати Android Studio або Visual Studio. Продуктивність буде оптимізована для пристроїв Android, зовнішній вигляд інтерфейсу користувача відповідатиме іншим нативним додаткам на пристрої, а всі функції або можливості пристрою користувача будуть перенаправлені на доступ і використання. Розробка програми у

нативному форматі відповідає всім шаблонами взаємодії і стандартам користувача, встановленим спеціально для пристроїв Android [21].

2.2.1.1. Android SDK

Android SDK [22] – це набір для розробки програмного забезпечення, розроблений Google для платформи Android. Щоразу, коли Google випускає нову версію Android або оновлення, також випускається відповідний SDK.

Android SDK оптимізований для Android Studio, а отже, щоб ефективно скористатися його перевагами, потрібно буде встановити Android Studio. Керувати Android SDK з Android Studio простіше, оскільки підтримка таких мов, як Java та Kotlin обробляється автоматично.

Самим простим способом створення нативного додатка Android є використання Android Studio з Java або Котлін. Засоби Android Studio SDK компілюють файли коду, даних і ресурсів в архівний пакет Android з розширенням apk [23].

Довгий час основною мовою для розробки додатків під Android була Java. Kotlin – це нова мова програмування, яку 2019 року компанія Google оголосила кращою мовою для Android-розробки.

Що робить Kotlin ще більш привабливим для розробки власних додатків для Android, це простіший синтаксис та його взаємодія з Java. Kotlin дозволяє написати те ж саме, що і Java, використовуючи меншу кількість рядків.

Фактично, Kotlin використовує віртуальну машину Java, і її код працюватиме так само, як і Java. У багатьох сценаріях він розглядається як заміна Java, оскільки суттєво впорядковує процес розробки. Багато програмістів прийняли Kotlin як основну мову для Android.

Додаток, розроблений за допомогою Kotlin, також працює так само швидко, як і додаток, створений за допомогою Java, оскільки має спільну структуру байт-коду [24].

2.2.1.2. Android NDK

Android NDK – це набір інструментів, які дозволяють реалізувати частину програми використовуючи такі мови як C/C++ [25].

Google рекомендує вдаватися до використання NDK тільки в рідкісних випадках. Найчастіше це такі випадки:

- потрібно збільшити продуктивність;
- потрібно використати сторонню бібліотеку. Наприклад, багато вже чого написано на C/C++ і потрібно використати вже існуючий матеріал;
- програмування на низькому рівні.

Android NDK – це допоміжний інструмент, який використовується тільки в поєднанні з Android SDK. NDK не може безпосередньо звертатися до безлічі API-інтерфейсів, доступних в Android SDK, а розробка в NDK завжди буде вводити додаткову складність в програму.

JNI – стандартний механізм для запуску коду, під керуванням віртуальної машини Java, який написаний на мовах C/C++ або Assembler, і скомпонований у вигляді динамічних бібліотек, дозволяє не використовувати статичні зв'язування. Це дає можливість виклику функції C/C++ з програми на Java, і навпаки [26].

2.2.2. Фреймворки для кросплатформеної розробки

Кросплатформені фреймворки надають єдину базу коду, що спільно використовується пристроями з різними операційними системами.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		29

Використання кросплатформеного фреймворку дозволяє додатку підтримувати однаковий вигляд і роботу на різних пристроях, а також використовувати переваги автоматичного розгортання оновлень і виправлень. Замість того, щоб розбиратися в різних мовах коду для конкретних пристроїв, додаток розробляється в загальній базі коду, як правило, написаній одною мовою.

Суть фреймворку – збільшити продуктивність, зменшивши витрати зусиль на розробку. В ідеалі вони економлять розробникам час на те, щоб зайнятися реально важливими при розробці програми питаннями.

Різні фреймворки мають багато однакових атрибутів, а також унікальні функції, які роблять їх більш придатними для певних типів проєктів [27].

2.2.2.1. Flutter

Flutter – один із нових членів сімейства Google, який уже встиг завоювати непогану репутацію в кросплатформеній розробці. Принцип фреймворка – єдина кодова база і для мобільних платформ, і для веб-застосунків, і для десктопних додатків.

Flutter використовує об'єктно-орієнтовану мову програмування Dart. При цьому підтримується Android Studio та IntelliJ, а також Visual Studio Code [28].

Flutter-додатки можна запустити на iOS і Android, при цьому інструмент нібито відходить від традиційних уявлень в UI-фреймворків обох операційних систем. Flutter не компілюється безпосередньо в iOS- і Android-додатки. Вони запускаються на змішаному інструменті з движка для рендерингу на C++ і фреймворку Flutter на Dart. Отримані таким

чином файли приєднуються до кожного додатку і SDK збирає з цього готову програму під конкретну платформу.

Такий підхід називається розробкою додатків методом «відеоігор». Так само як гра ігнорує майже весь програмний фреймворк і основну роботу перекладає на свій движок, додатки на фреймворку від Google замінюють flutter-елементами всіх можливі частини нативних фреймворків. Від цього страждає компактність додатку, однак такий додаток працює швидко.

Кросплатформена розробка з Flutter не означає погіршення якості програмних продуктів. Фреймворк надає всі нативні віджети інтерфейсів Android (Material Design) і iOS (Cupertino) і сам вміє змінювати поведінку окремих елементів, щоб програми максимально відповідали звичним користувачам.

Крім кросплатформених можливостей Flutter надає функцію роздільної компіляції файлів в режимі розробки. Замість AoT-компілятора для готового продукту JiT-компілятор дозволяє збільшити швидкість створення і налагодження програм. Більш того, фреймворк від Google підтримує близько тисячі плагінів, включаючи SQLite, Firebase, Facebook Connect і GraphQL. А ще він дозволяє поєднувати код Flutter з елементами стандартних мов розробки додатків під конкретну платформу.

Центральним елементом програми на Flutter є віджети. Фактично це ті візуальні компоненти, з яких складається графічний інтерфейс.

Виконання програми Flutter починається з функції `main()`. Для створення графічного інтерфейсу в цій функції викликається інша вбудована функція – `runApp(Widget app)`. Вона прикріплює певний віджет до екрану. Тобто по суті це, що ми бачимо під час запуску програми на

екрані пристрою. Прикріплюваний віджет передається в функцію `runApp()` в якості параметра.

Всі віджети успадковується від класу `Widget`. Одні віджети можуть включати інші, утворюючи тим самим ієрархічну систему у вигляді дерева.

Віджети описують, як повинні виглядати елементи інтерфейсу, враховуючи їх поточну конфігурацію і стан. Коли стан віджета змінюється, він перебудовує свій опис, який фреймворк надалі відрізняє від попереднього опису.

При написанні програми, зазвичай створюються нові віджети, які є підкласами `StatelessWidget` або `StatefulWidget`, в залежності від того, чи управляє віджет будь-яким станом. Основне завдання віджета – це реалізувати функцію `build()`. `StatelessWidget` рекомендується для незмінних віджетів. Це такі віджети які не мають внутрішнього стану, залежать тільки від конфігураційних параметрів і від батьківських віджетів. `StatefulWidget` потрібен, коли внутрішніх станів більше один і вони можуть змінювати один одного [29].

Flutter поставляється з набором потужних базових віджетів, з яких дуже часто використовуються наступні:

- `Text` – дозволяє вам створювати стилізований текст в вашому додатку;
- `Row`, `Column`;
- `Stack` – дозволяє розміщувати віджети один над одним в порядку малювання;
- `Container` – дозволяє створювати прямокутний візуальний елемент. Контейнер може бути прикрашений за допомогою `BoxDecoration`, наприклад фоном, рамкою чи тінню. Контейнер також може мати поля, відступи і обмеження, що застосовуються до його розміру.

2.2.2.1.1. Dart

Dart – об'єктно-орієнтована мова. Всі значення, які використовуються в програмі на Dart, представляють об'єкти. У своєму розвитку Dart зазнав впливу більш ранніх мов, таких як Smalltalk, Java, JavaScript. Його синтаксис схожий на синтаксис інших С-подібних мов.

Для зберігання даних в мові Dart, як і в інших мовах програмування, використовуються змінні. Для іменування змінних застосовується стиль `lowerCamelCase`, тобто назва змінної починається з маленької літери. Якщо ім'я змінної складається з двох і більше слів, то перше слово починається з маленької, а всі наступні слова з великої літери.

Тип даних визначає, які саме значення може зберігати об'єкт. Dart має ряд вбудованих типів (таблиця 2.1).

Таблиця 2.1 – Типи даних в Dart

Тип	Опис
int	цілочислове значення
double	число з плаваючою комою
String	послідовність символів
Runes	послідовність символів в кодуванні UTF-32
bool	логічне значення, має два значення: true і false
Symbol	символ
List	список, який представляє собою набір значень або ж просто масив
Set	невпорядкована колекція унікальних елементів
Map	колекція, яка зберігає інформацію ключ-значення

Окрім перерахованих типів даних змінну можна оголосити за допомогою операторів `var` та `dynamic`. У випадку з `var`, Dart сам визначає тип змінної, виходячи з присвоєного їй значення. Оператор `dynamic`, як і `var`, дозволяє вивести тип змінної виходячи з присвоєного їй значення. Але на відміну від `var`, `dynamic` дозволяє змінювати тип змінної.

2.2.2.2. React Native

React Native – це фреймворк з відкритим кодом, випущений в 2015 році за підтримки Facebook. Побудований на React, популярній бібліотеці JavaScript для веб-розробки, React Native використовує вбудовані компоненти власного інтерфейсу та API, що забезпечують високу продуктивність та швидкий розвиток.

Завдяки власним компонентам інтерфейсу, програми, побудовані на React Native, мають природний вигляд та стиль. Крім того, розробники можуть швидко оновити сторінку, яку ви створюєте, не чекаючи закінчення власних збірок.

Додатки, які створюються за допомогою React Native, не є мобільними веб-додатками, оскільки React Native використовує ті ж компоненти, що і звичайні програми для iOS і Android. Замість того щоб використовувати мову Swift, Kotlin або Java, ці компоненти збираються за допомогою JavaScript і React.

React – це інструмент для створення користувацьких інтерфейсів. Його основне завдання – забезпечити відображення на екрані того, що можна побачити на веб-сторінках. React дозволяє легко створювати інтерфейси, розділяючи кожен сторінку на невеликі фрагменти і компоненти. React популярний з кількох причин. Він компактний і відрізняється високою продуктивністю, особливо при роботі з мінливими даними.

Всі додатки React будуються на основі компонентів. Це можуть бути прості функції і класи JavaScript, але їх можна використовувати як персоналізовані елементи HTML. Кнопки, меню і будь-які інші елементи інтерфейсу можна створювати як компоненти. Компоненти також можуть містити інформацію про стан і розмітку екрану.

Зазвичай компоненти будуються за допомогою JSX. Він є корисним посібником при роботі з користувацьким інтерфейсом всередині коду JavaScript. JSX являє собою суміш синтаксису HTML і JavaScript.

Компоненти можуть посилатися на інші компоненти у дереві компонентів. Це дозволяє використовувати одну і ту ж абстракцію – компоненти – на будь-якому рівні. У додатках, написаних на React з нуля, як правило, є один компонент App, який знаходиться на самому верху [30].

2.2.2.2.1. JavaScript

JavaScript – мультипарадигмена мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний стилі.

JavaScript є слабо типізованою мовою. Це означає, що тип змінної не потрібно визначати заздалегідь. Тип визначиться автоматично під час виконання програми. Також це означає, що одну змінну можна використовувати для зберігання даних різних типів. Змінні оголошуються за допомогою ключового слова `var` або `let`. Наразі `var` вже не використовується. Для іменування змінних застосовується стиль як і в Dart.

Таблиця 2.2 – Типи даних в JavaScript

Тип	Опис
undefined	вказує, що значення не встановлено

Тип	Опис
Boolean	логічне значення, має два значення: true і false
String	послідовність символів
Number	числа
BigInt	вбудований об'єкт, який надає спосіб представлення цілих чисел, які більші за максимальне число типу Number
Symbol	символ
null	вказує на невизначений тип
Object	структура, яка використовується для зберігання даних і для створення інших структур
Function	спеціальний підвид Object, який спрощує визначення типу для функцій

2.2.2.3. Xamarin

Xamarin – один з найбільш популярних інструментів кросплатформеної розробки додатків для iOS, Android, tvOS, watchOS, macOS і Microsoft з використанням C# і .NET.

Алгоритми, розроблені на C# для однієї платформи, програмісти використовують і для всіх інших платформ. Наприклад, при роботі на Xamarin.Android і Xamarin.iOS, до 75% коду можна перевикористати: додаток працює однаково і на iPhone, і на Android-смартфоні.

Якщо в додатку головне – функціональність, а з дизайну всього пара кнопок і красивий графік – Xamarin чудово підходить. На проєктах зі

складним інтерфейсом вартість допрацювання іноді порівнянна з розробкою двох-трьох нативних рішень. Але бізнес-логіка одна та із кожною новою версією програми розробляти ще буде простіше.

На відміну від деяких інших кросплатформених рішень, Xamarin.Forms не відображається в веб-поданні. Замість цього воно використовує пакети SDK для Android або iOS для створення власних елементів управління, що забезпечує високу продуктивність.

Інтерфейс додатків Xamarin.Forms складається з об'єктів, які зіставляються з власними органами управління кожної цільової платформи. Це дозволяє додаткам для конкретних платформ для iOS, Android і універсальна платформа Windows використовувати Xamarin.Forms код, що міститься в бібліотеці .NET Standard.

Xamarin має чотири основні групи управління, які використовуються для створення інтерфейсу програми Xamarin.Forms:

- сторінки – кросплатформені екрани;
- макети – контейнери для інших елементів;
- представлення – об'єкти користувальницького інтерфейсу, такі як мітки, кнопки і повзунки;
- комірка – спеціалізований елемент, який використовується для елементів в таблиці і описує, як повинен візуалізуватися кожен елемент в списку.

Крім елементів управління, що надаються з Xamarin.Forms, доступні сторонні елементи управління [31].

2.2.2.3.1. C#

C# – об'єктно-орієнтована мова програмування, розроблена в 1998-2001 роках групою інженерів компанії Microsoft як мову розробки додатків для платформи Microsoft .NET Framework.

Мова програмування C# використовує синтаксис, який сильно нагадує C і наближений до C++ та Java.

Якщо мова заходить про C#, часто мається на увазі технологія, пов'язані з платформою .NET. Те ж саме справедливо в зворотному напрямку, кажучи про .NET, думають про C#. Безумовно, поняття мають жорсткий взаємозв'язок, але не є синонімами. C# – це мова, створена для взаємодії з фреймворком .NET.

Типи даних мають особливе значення в C#, оскільки це строго типізована мова. Це означає, що всі операції піддаються постійному контролю з боку компілятора на відповідність типів, причому неприпустимі операції не компілюються. Отже, строгий контроль типів дозволяє виключити помилки і підвищити надійність програм. Для забезпечення контролю типів всі змінні, вирази і значення повинні належати до певного типу (таблиця 2.3).

Таблиця 2.3 – Типи даних в C#

Тип	Опис
byte	цілі додатні числа (1 байт)
sbyte	цілі числа (1 байт)
short	цілі числа (2 байта)
ushort	цілі додатні числа (2 байта)

Тип	Опис
int	цілі числа (4 байта)
uint	цілі додатні числа (4 байта)
long	цілі числа (8 байт)
ulong	цілі додатні числа (8 байт)
float	числа з плаваючою комою (4 байта)
double	числа з плаваючою комою (8 байт)
char	символи
bool	логічне значення
decimal	десяткові числа
string	послідовність символів
object	може зберігати значення будь-якого типу даних

2.2.2.4. Порівняння корсплатформених фреймворків

Кросплатформені фреймворки для розробки володіють багатьма унікальними перевагами, які подобаються розробникам. Кросплатформеність в проєкті потрібна не заради призначеного для користувача зручності, а для оптимізації вартості розробки і підтримки програми.

Кожен фреймворк має ряд переваг та недоліків, проте виграшним варіантом, незважаючи на його «юність», є фреймворк Flutter.

Таблиця 2.4 – Порівняння кросплатформених фреймворків

Характеристика	Flutter	React Native	Xamarin
Мова розробки	Dart	JavaScript	C#
Open source	Так	Так	Має платні пакети
Спільнота	Невелика, проте активно розвивається	Дуже велика	Велика
Продуктивність	Дуже висока	Висока, близька до нативної	Висока, близька до нативної
Підтримувані платформи	iOS, Android, Linux, Mac, Windows, Google Fuchsia	Android, Android TV, iOS, macOS, tvOS, Web, Windows, UWP	Android, UWP, WPF, Android TV, iOS, macOS, tvOS, watch OS, Web, Windows
Функція hot reload	Так	Так	Ні

2.3. База даних мобільного додатку

Коли необхідно вибрати СУБД, головне питання зазвичай полягає у виборі реляційної (SQL) або нереляційної (NoSQL) структури. У обох варіантів є свої переваги, а також кілька ключових особливостей, які варто мати на увазі при виборі.

Реляційні бази даних використовують структуровану мову запитів для визначення і обробки даних. З одного боку, це відкриває великі можливості для розробки: SQL одна з найбільш гнучких і поширених мов запитів, тому її вибір дозволяє мінімізувати ряд ризиків, і буде особливо

вигідною, якщо має бути робота з комплексними запитами. З іншого боку, в SQL є ряд обмежень. Побудова запитів на цій мові зобов'язує визначати структуру даних і подальша зміна структури даних може бути згубною для всієї системи.

Нереляційні бази даних, в свою чергу, пропонують динамічну структуру даних, які можуть зберігатися кількома способами: орієнтовано по колонках, документо-орієнтовано, у вигляді графів або на основі пар «ключ-значення». Така гнучкість означає наступне:

- можливість створювати документи, не створюючи їх структуру заздалегідь;
- кожен документ може мати власну структуру;
- у кожній базі даних може бути власний синтаксис;
- можливість додавати поля прямо під час роботи з даними.

У реляційних СУБД дані представлені в вигляді таблиць, в той час як в нереляційних – у вигляді документів, пар «ключ-значення», графів або wide-column сховищ. Це робить SQL бази даних кращим вибором для додатків, які передбачають транзакції з декількома записами – як, наприклад, система облікових записів – або для застарілих систем, які були побудовані для реляційних структур [32].

2.3.1. Реляційна СУБД SQLite

SQLite – це автономна база даних без сервера SQL. Це вбудована кросплатформена база даних, яка підтримує досить повний набір команд SQL і доступна у вихідних кодах. Движок бази даних являє собою бібліотеку, з якої програма компонується і SQLite стає складовою частиною програми. Вся база зберігається в єдиному стандартному файлі на машині, на якій виконується програма.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

SQLite підтримує типи TEXT (аналог String в Java), INTEGER (аналог long в Java) і REAL (аналог double в Java). Решта типів слід конвертувати, перш ніж зберігати в базі даних. SQLite не перевіряє типи даних, тому можна записати ціле число в колонку, призначену для рядків, і навпаки.

Таблиця 2.5 – Типи даних в SQLite

Тип	Опис
NULL	порожнє значення
INTEGER	цілочислове значення
REAL	число з плаваючою комою
TEXT	рядок або символ
BLOB	масив бінарних даних

SQLite ретельно дотримується цілісність даних, реалізуючи механізм зміни даних через транзакції. Коротко про транзакції: транзакція або повністю виконується, або не виконується взагалі. Проміжних станів бути не може. Якщо ви не використовуєте транзакції явно, то завжди створюється неявна транзакція. Вона стартує перед виконанням команди і зберігає дані відразу після її завершення.

SQLite реалізує більшу частину стандарту SQL, але не підтримує клієнт-серверну модель [33].

2.3.2. Нереляційна база даних Firebase

База даних Firebase Realtime – це хмарна NoSQL база даних компанії Firebase. Дані зберігаються в форматі JSON і синхронізуються в режимі реального часу з кожним підключеним клієнтом.

Замість типових HTTP-запитів база даних Firebase Realtime використовує синхронізацію даних – кожен раз, коли дані змінюються, будь підключений пристрій отримує це оновлення протягом мілісекунд.

Доступ до бази даних Firebase Realtime можна отримати безпосередньо з мобільного пристрою або веб-браузера, а тому немає необхідності в серверній частині додатку.

Безпека і перевірка даних доступні через правила безпеки баз даних Firebase Realtime. Ці правила засновані на виразах, які виконуються при читанні або запису даних. Правила мають синтаксис, подібний до JavaScript, що полегшує його розуміння, і вони бувають чотирьох типів:

- read – описує, чи дозволено і коли користувачеві читати дані;
- write – описує, чи дозволено і коли можливо записувати дані;
- validate – визначає, як буде виглядати правильно відформатоване значення, чи має воно дочірні атрибути, і тип даних;
- indexOn – визначає дочірню частину, яку потрібно індексувати для підтримки впорядкування та запитів.

За допомогою бази даних Firebase Realtime можна масштабувати потреби додатків в даних, розбиваючи дані на кілька примірників бази даних в одному проєкті Firebase [34].

Висновок до розділу 2

Загалом створено багато допоміжних сервісів для розробки дизайнів проєктів для мобільних додатків та веб-застосунків. Для створення прототипу та дизайн-макету в дипломній бакалаврській роботі було використано онлайн-сервіс Figma, оскільки вона надає найбільш повний перелік функцій та можливостей для створення дизайн-проєкту.

Для програмної частини також є великий вибір технологій. Вони діляться на дві категорії – технологій для розробки нативних додатків та технології для кросплатформених рішень. Звичайно, нативний додаток виходить найбільш продуктивним, проте у подальшому може з'явитися потреба використовувати додаток не тільки на Android-платформі, тому при виборі технології для розробки клієнта брали участь лише кросплатформені фреймворки. Серед них було переглянуто три найпопулярніші та в результаті обрано Flutter.

Для розробки віддаленої бази даних буде використовуватись Firebase Realtime, оскільки вона дозволяє працювати з базою даних одразу декільком клієнтам без розробки серверної частини додатку, тоді як SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якою програма компонується і движок стає складовою частиною програми. Окрім цього, Firebase просто і легко інтегрується в платформу Flutter. Проте, SQLite буде використовуватись як локальна база даних для забезпечення роботи додатку без підключення до мережі.

РОЗДІЛ 3. ДЕТАЛІ РЕАЛІЗАЦІЇ ДОДАТКУ

3.1. База даних

3.1.1. Налаштування Firebase-проєкту

Для підключення Firebase до додатку потрібно спочатку створити пустий Firebase Console Project. Потрібно продумати його назву, якщо потрібно ввімкнути Google Analytics та почати реєструвати цей проєкт.

Для реєстрації проєкту потрібно ввести назву пакету вже створеного проєкту для клієнтської частини додатку. У випадку даної дипломної роботи це «com.gmail.dubyna.mari.thesis_app». Потім завантажити файл «google-services.json» та додати у клієнтський проєкт потрібні залежності. Після цього клієнтський додаток може взаємодіяти з проєктом Firebase.

Користувачі будуть реєструватись за допомогою Firebase Authentication, то у вкладці Firebase-проєкту потрібно додати залежності до клієнтської частини та вибрати спосіб входу. У нашому випадку це буде вхід за допомогою електронної пошти та паролю.

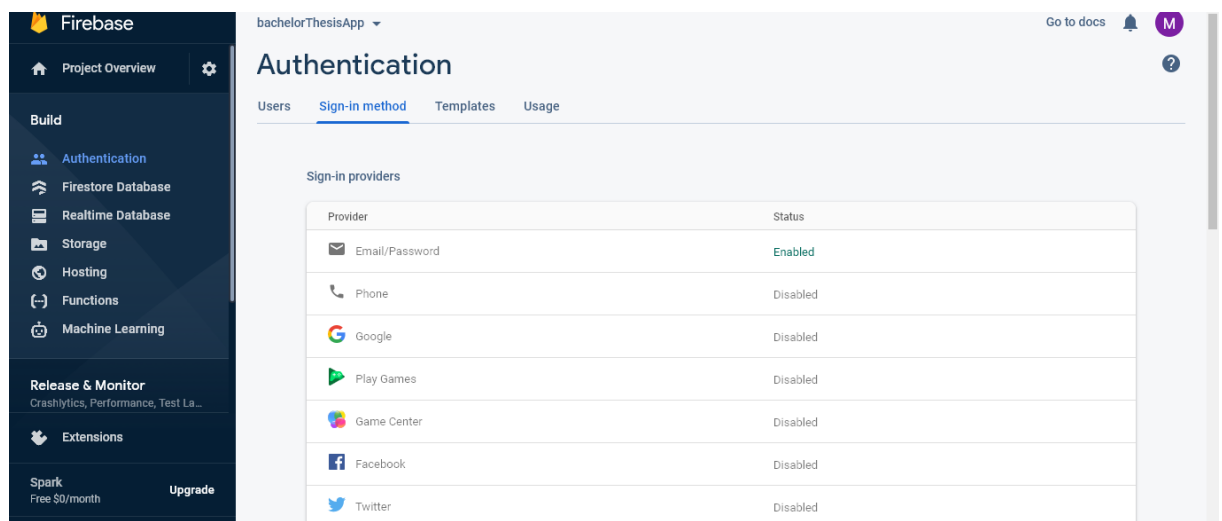


Рисунок 3.1 – Вибір способу входу

Оскільки користувачам буде потрібно буде підтвердити свою пошту, потрібно налаштувати відповідний темплейт листа, який буде відправлятися за запитом клієнтської частини.

Sender name	From	
Fundex	fundex@bachelorthesisapp.firebaseio.com	
Reply to		
noreply		
Subject		
Verify your email for Fundex		
Message		
Hello %DISPLAY_NAME%,		
Follow this link to verify your email address.		
https://bachelorthesisapp.firebaseio.com/_/auth/action?mode=action&oobCode=code		
If you didn't ask to verify this address, you can ignore this email.		
Thanks,		
Your %APP_NAME% team		

Рисунок 3.2 – Шаблон листа для підтвердження пошти

Також потрібен буде шаблон листа для відновлення паролю. Цей шаблон містить посилання на сторінку з полем, де вводиться новий пароль.

Sender name	From	
Fundex	fundex@bachelorthesisapp.firebaseio.com	
Reply to		
noreply		
Subject		
Reset your password for Fundex		
Message		
Hello,		
Follow this link to reset your Fundex password for your %EMAIL% account.		
https://bachelorthesisapp.firebaseio.com/_/auth/action?mode=action&oobCode=code		
If you didn't ask to reset your password, you can ignore this email.		
Thanks,		
Your Fundex team		

Рисунок 3.3 – Шаблон листа для відновлення паролю

Для того, щоб можна було зберігати певні дані в цьому проєкті потрібно у вкладці «Cloud Firestore» створити базу даних та обрати місце знаходження цієї бази. Автоматично встановлюється «nam5 (us-central)», проте для дипломного проєкту було обрано «eur3 (europe-west)».

Після створення бази даних клієнтська частина може зберігати в ній певні дані.

3.1.2. Локальна база даних SQLite

Для того, щоб використати базу даних SQLite в клієнтському проєкті потрібно, як і з віддаленою базою даних, додати пакет «sqflite» у файлі із залежностями.

Для збереження даних потрібна певна модель. В дипломному проєкті таких моделей шість. Детальніше моделі можна переглянути в додатку 2. В кожній моделі повинен бути унікальний ідентифікатор та функції toMap() та fromMap(), для конвертації об'єкту відповідного класу у JSON-формат та навпаки.

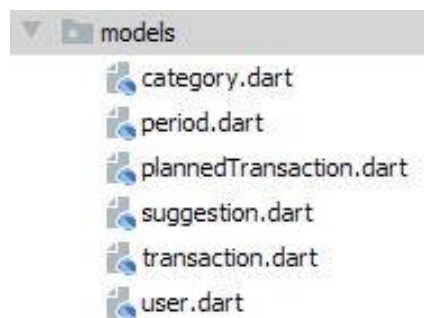


Рисунок 3.4 – Папка з моделями

Для роботи з базою потрібно отримати посилання на неї. Посилання отримується з двох частин:

- назва директорії, яка отримується через функцію з пакету «sqflite», а саме getDatabasesPath();

- назва бази даних.

Якщо база за заданим шляхом не існує, то вона створюється. В методі, де створюється база, генеруються таблиці для всіх наявних моделей.

3.2. Розробка клієнтської частини

Основу частину клієнтської частини складає інтерфейс. Весь інтерфейс розбитий на сторінки. Вони знаходяться в директорії pages.

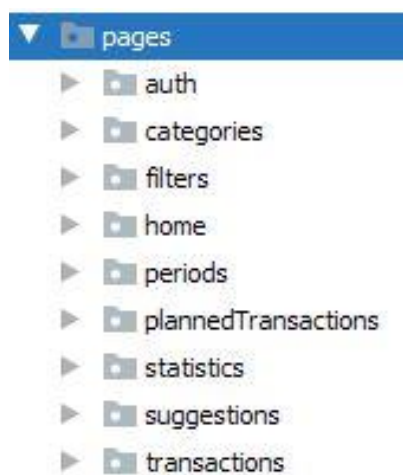


Рисунок 3.5 – Сторінки додатку

Початок роботи додатку починається з файлу main, який містить однойменну функцію. В цьому класі визначаються головні стилі та будується тіло головної сторінки через виклик класу Wrapper. У останньому класі перевіряється наявність авторизованого користувача. Це зроблено для того, щоб вже авторизовані користувачі могли продовжувати свою роботу в додатку без постійних повторних авторизацій після згортання або ж закриття додатку. Якщо користувач авторизований, відривається домашня сторінка додатку, в іншому випадку – сторінка авторизації.



Рисунок 3.6 – Стартові класи проєкту

Папка auth містить наступні файли:

- authWrapper – в залежності від того, буде користувач авторизовуватись чи реєструватись по-різному промальовує дочірній елемент authForm;
- authForm – сторінка, з усіма полями для реєстрації та авторизації;
- emailVerification – сторінка, для підтвердження пошти та кнопкою для повторної відправки листа;
- forgotPassword – сторінка з полем, для введення електронної адреси, на яку відправиться лист з посиланням для відновлення паролю;
- validators – файл з методами для перевірки коректності введених даних.

Після успішної авторизації користувач перенаправляється на сторінку home. Вона знаходиться у відповідній папці та містить в собі дві сторінки, які змінюються за допомогою натискання на кнопки в нижній панелі екрану або ж свайпом вправо. Перша сторінка містить список транзакцій, а друга – статистику.

Також в цій директорії знаходиться файл mainDrawer, в якому знаходиться сторінка з навігацією по всіх сторінках додатку та інформацією про користувача.

Папка transactions містить наступні файли:

- transactionList – містить віджет ListView.builder, який відображає список довільної кількості елементів типу transactionTile;
- transactionTile – віджет, який відображає інформацію про кожну окрему транзакцію;
- transactionForm – віджет, що в залежності від двох параметрів – isPlannedTxMode та isEditMode – будує екрани створення або редагування звичайних транзакцій або повторюваних.

Останній клас, тобто клас TransactionForm, використовується також у класі PlannedTransactionsList, що знаходиться у папці plannedTransactions. Сам клас PlannedTransactionsList, як і TransactionList, являє собою ListView.builder, який відображає список елементів внутрішнього віджету plannedTransactionCard. Останній описує вигляд повторюваної транзакції в списку.

У папці suggestions знаходиться клас SuggestionsList – список всіх шаблонів транзакцій. Шаблони відображаються за макетом внутрішнього віджета suggestionCard.

Для перегляду статистичних даних використовується класи із папки statistics:

- statistics – основна сторінка статистики, яка містить всі основні її компоненти та перемикач для зміни періодизації статистики;
- balance – віджет для відображення загальних статистичних даних за обраний період, а саме: баланс за даний період, кількість днів до його закінчення, суму грошей з розрахунком на кожен день до закінчення періоду та гістограма доходів та витрат за період;
- barTile – віджет, що відображає елемент гістограми із підписом;

- `categories` – віджет для відображення витрат у вигляді кругової діаграми;
- `indicator` – віджет для відображення категорій витрат по списку до кругової діаграми. Відображує кольором категорії, якщо натиснуто на дану категорію, та сірим в іншому випадку;
- `topExpenses` – віджет для відображення списку найбільших витрат за обраний період. Стандартно показує до п'яти елементів включно, якщо потрібно переглянути більше елементів, можна підвантажити весь список, та згорнути його назад, якщо весь список не потрібен.

Для роботи з періодами для статистики використовуються файли із папки `periods`:

- `periodList` – віджет, що відображає список періодів у вигляді внутрішнього віджету `periodCard`;
- `periodForm` – викликається з віджету `periodList`, після натискання на `floatingActionButton`, та являє собою сторінку для створення нових періодів.

Категорії відображаються та вся основна взаємодія з ними відбуваються через класи із папки `categories`:

- `categoriesList` – віджет зі списком всіх наявних категорій, які відображаються у вигляді `categoryTile`, та кнопкою для створення нових категорій;
- `categoryForm` – віджет, який відображає вікно для створення та редагування категорій;
- `categoryTile` – віджет, що відображає інформацію про категорію;
- `categoryRegistry` – містить список стандартних категорій, що додаються користувачу під час реєстрації;

- `iconsList` – віджет для відображення списку всіх іконок, які можна вибрати для категорій;
- `iconsRegisrty` – список іконок, які можна використовувати для категорій.

Основна робота по взаємодії з базами даних відбувається у файлах папки `services`.

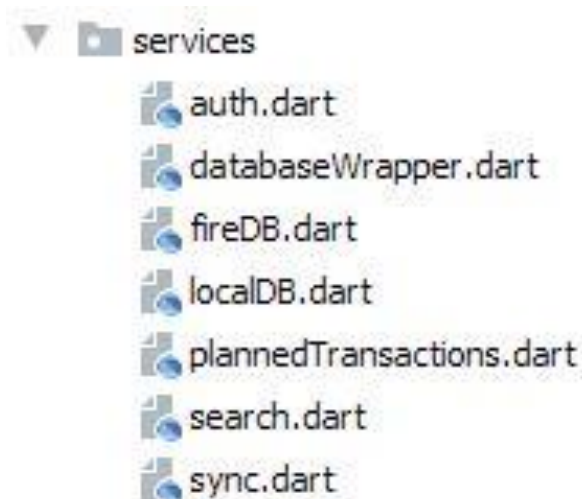


Рисунок 3.7 – Сервіси

`AuthService` відповідає за авторизацію користувача. Цей сервіс має функції, які через `FirebaseAuth` дозволяють авторизацію, реєстрацію, вихід з облікового запису та відправку листів для підтвердження електронної пошти та відновлення паролю. Саме через те, що цей сервіс працює з `FirebaseAuth`, всі перераховані вище функції відбуваються лише з підключенням до мережі Internet. Для забезпечення безпеки пароля користувача, він не зберігається ні в одній із баз даних, а лише у `FirebaseAuth` зі встановленим шифруванням.

Надалі вся взаємодія відбуватиметься через `DatabaseWrapper`, який в залежності від наявного підключення до мережі Internet обирає локальну або віддалену базу даних для взаємодії. Даний клас містить функції

додавання до бази, видалення, оновлення даних та отримання даних за ідентифікатором для всіх моделей.

Класи FireDBService та LocalDBService містять перелік тих самих функцій, що й DatabaseWrapper, для віддаленої та локальної бази відповідно.

Клас PlannedTransactionsService містить лише одну функцію, яка проходиться по всім повторюваним транзакціям та створює на їх базі звичайні транзакції та записує їх до бази даних.

Клас SyncService відповідає за синхронізацію даних між локальною базою даних та віддаленою. Окрім окремих функцій синхронізації кожної моделі, клас містить дві функції syncToCloud() та syncToLocal(), які підгружають дані з локальної бази до віддаленої та навпаки.

Окремо виділені в папку shared стилі, певні структурні компоненти, функції з підрахунками для інфографіки, дані для конфігурації локальної бази даних та програмні константи.

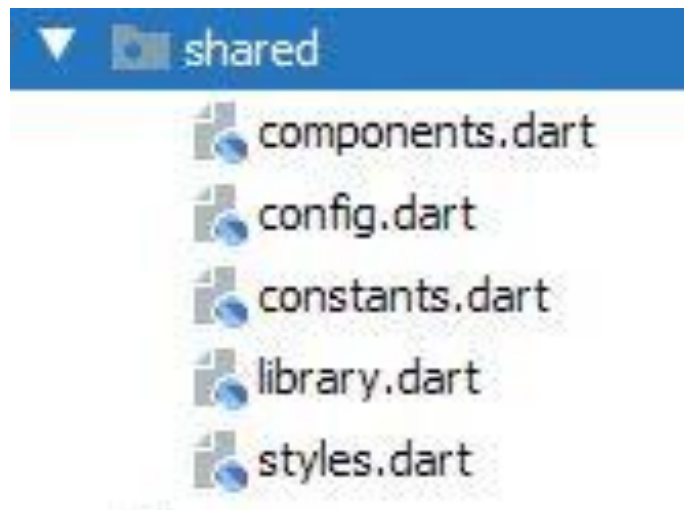


Рисунок 3.8 – Папка для спільного користування сторінками

До компонентів відносяться такі віджети як:

- StatTitle – для відображення заголовку;
- Alert – для відображення попередження перед видаленням об'єкту моделі або ж перед виходом із облікового запису;
- Loader – віджет, який відображається під час загрузки інформації;
- DeleteIcon – кнопка з іконкою смітника, яка видаляє певний елемент;
- DatePicker – відображає віджет для обрання дати;
- TimePicker – відображає віджет для обрання часу;
- FloatingButton – кнопка, яка на різних сторінках використовується для додавання елементів;
- TabSelector – віджет, що відображає кнопки для обрання доходу чи витрати або ж періоду в статистиці.

Висновок до розділу 3

У даному розділі було розглянуто деталі розробки клієнтської частини додатку, налаштування Firebase Console Project та підключення локальної бази даних SQLite.

Firebase Console Project містить базу Firestore Database для збереження даних віддалено у форматі JSON та FirebaseAuth налаштована таким чином, щоб користувачі реєструвались та заходили в систему за допомогою електронної адреси та паролю. Також у FirebaseAuth містяться шаблони листів, що використовуються для підтвердження електронної пошти та відновлення паролю.

Локальна база даних SQLite створюється та генерує таблиці для всіх наявних в проєкті моделей під час першого запуску програми. Всі таблиці містять унікальні ідентифікатори та зовнішні ключі для зв'язку з іншими таблицями.

Клієнтська частина розбита по сторінкам, які несуть певне структурне та функціональне навантаження – кожна сторінка відповідає за інтерфейс взаємодії користувача з системою.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		55

РОЗДІЛ 4. ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1. Реєстрація та авторизація

Робота неавторизованого користувача в додатку починається зі сторінки входу. На цій сторінці користувач вводить логін та пароль вже зареєстрованого акаунту або ж переходить на вкладку з реєстрацією. Також з даної сторінки можна відновити пароль після натиску на кнопку «Forgot password?».

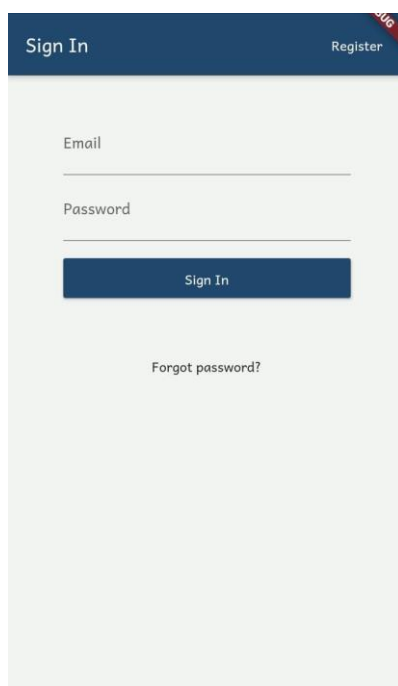


Рисунок 4.1 – Вкладка авторизації

Після натиску на кнопку «Forgot password?» користувачу відобразиться сторінка з полем для вводу електронної пошти та кнопкою для відправлення листа на пошту (рисунок 4.2). Лист містить посилання на форму відновлення паролю. Після того, як користувач введе пароль на даній формі та натисне на кнопку «Save», він може авторизуватись на відповідній сторінці з використанням цього паролю. Варто зазначити, що пароль має бути не менше шести знаків.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		56

Рисунок 4.2 – Вкладка відновлення паролю

На вкладці реєстрації користувач має ввести адресу електронної пошти, вигадати пароль та продублювати його, ввести своє ім'я або ж нікнейм.

Рисунок 4.3 – Вкладка реєстрації

Після реєстрації свого акаунту, користувач повинен підтвердити свою електронну адресу, тому його перенаправляє на відповідну вкладку. Якщо лист не було отримано, потрібно повернутись в додаток та натиснути на кнопку для повторного відправлення листа.

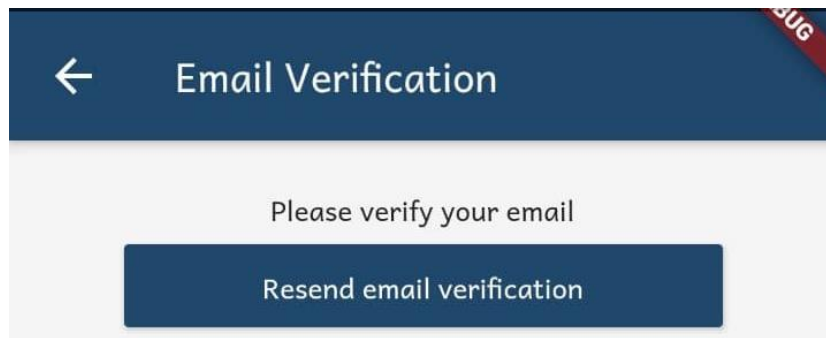


Рисунок 4.4 – Вкладка підтвердження електронної пошти

Всі перераховані вище операції проводяться лише з наявним підключенням до мережі Internet.

Після авторизації або ж реєстрації користувач отримує доступ до системи та знаходиться на сторінці з усіма наявними транзакціями. Значок меню на цій вкладці відкриває меню з інформацією про користувача та навігація по вкладкам додатку. Перейти можна на головну сторінку з транзакціями, вкладку з категоріями, вкладку з періодами для фільтрування даних для інфографіки, регулярних транзакцій, вкладку зі збереженими шаблонами операцій та натиснувши на «Sign Out» вийти з облікового запису.

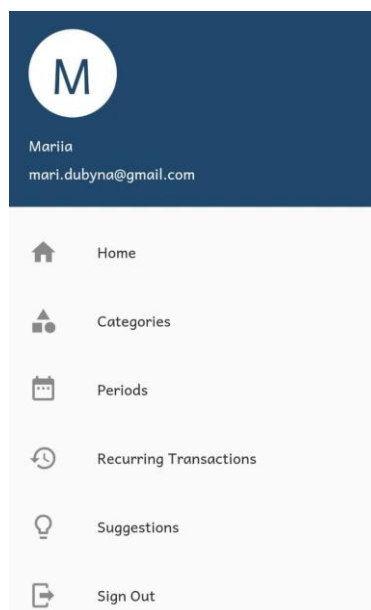


Рисунок 4.5 – Меню навігації

4.2. Робота з транзакціями

З головної сторінки додатку створюються грошові потоки. Для цього потрібно натиснути на кнопку внизу екрану.

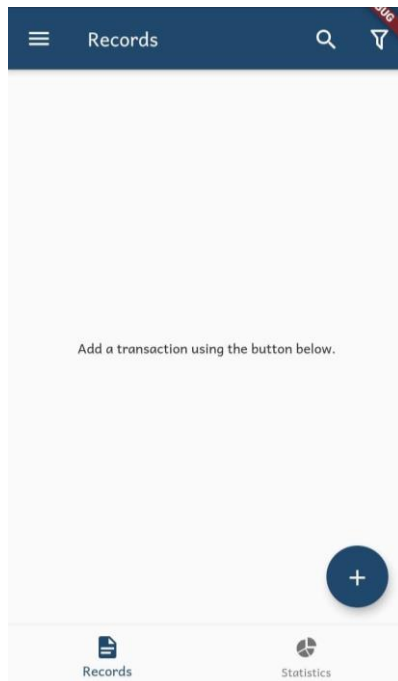


Рисунок 4.6 – Головна сторінка

Після натиску на кнопку користувачу відкривається вікно із вкладкою для створення нової транзакції (рисунок 4.7). Тут обирається тип транзакції – витрата чи дохід. Дата і час транзакції автоматично встановлюється той, який на даний момент в системі. За потреби дату і час можна змінити. Після цього потрібно ввести засіб платежу, наприклад карта, готівка тощо, проте можна обирати довільні назви. Вводиться сума транзакції, обирається із уже створених категорій потрібна. Всі операції додаються в шаблони. На даній сторінці можна обрати чи буде цей шаблон використовуватись надалі чи він буде просто збережений. Після натиску кнопки «Add» транзакція додається до списку транзакцій, а користувача переносить на головну сторінку. Якщо на головній сторінці натиснути на

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		59

певну транзакцію, відкриється вкладка, яка схожа на вкладку створення транзакції, де можна буде її змінити або ж видалити.

Рисунок 4.7 – Сторінка створення транзакції

У вкладці «Suggestion» або ж по-простому у вкладці з шаблонами знаходяться всі наявні шаблони транзакцій. Тут можна відмічати які з них будуть використовуватись, а які ні.

Рисунок 4.8 – Вкладка зі збереженими шаблонами

При створенні наступних транзакцій під час введення засобу оплати користувачу буде показано вікно з вибором. Якщо платіжний засіб один, але категорії у транзакцій були різні, то буде запропоновано стільки активних шаблонів, скільки різних категорій було по цьому засобу платежу.

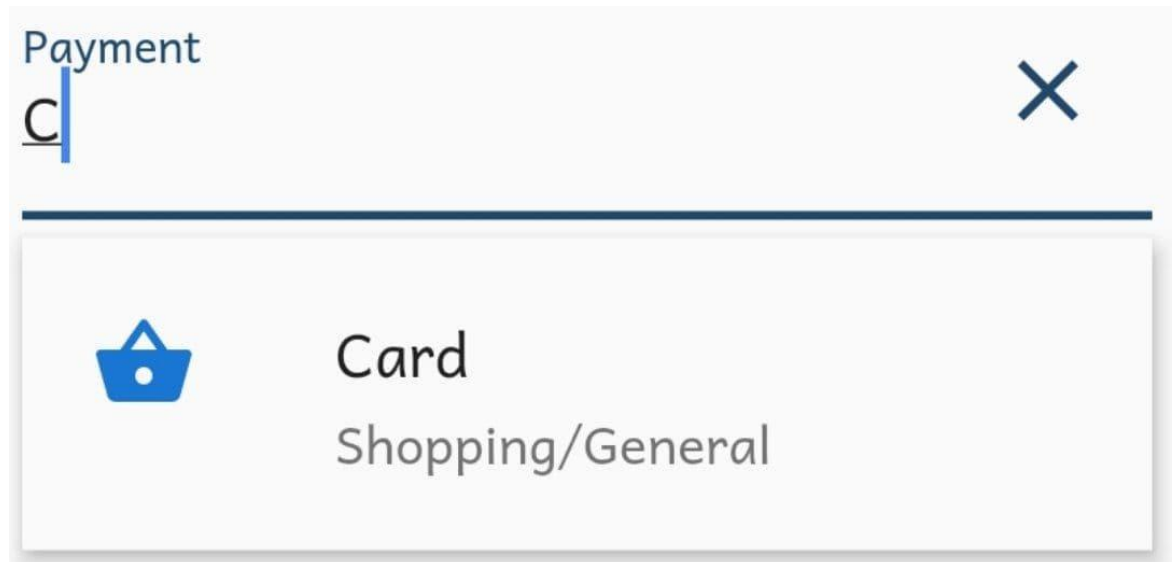


Рисунок 4.9 – Поле введення засобу платежу з підказкою

Якщо транзакції будуть певний час повторюватись, для того, щоб не створювати кожного разу нову транзакцію можна створити повторювану транзакцію. Для цього потрібно з меню навігації потрібно перейти у вкладку з запланованими транзакціями. Така транзакція створюється по схожому шаблону як і звичайні транзакції, проте дата обирається для наступного такого платежу або ж доходу і потрібно обрати наскільки часто ця операція проводитиметься. Наприклад: якщо операція відбувається раз на два дні, то потрібно в поле «Frequency» потрібно поставити 2, а «Weeks» змінити на «Days» у випадаючому списку. Якщо ця операція виконуватиметься постійно, то більше нічого не потрібно робити і можна закінчувати створення повторюваної транзакції. Якщо ж транзакція має відбуватись певну кількість разів або ж закінчитись в якийсь день, то потрібно встановити відповідні кінцеві умови.

Рисунок 4.10 – Вкладка створення повторюваних транзакцій

Кожна повторювана транзакція буде відображатись у відповідній вкладці з відзначенням наступної дати транзакції. Якщо було встановлено кінцеві умови, це буде відображено у відповідній транзакції. Якщо потрібно таку транзакцію змінити або ж видалити потрібно натиснути на неї та змінити потрібні поля або ж умови. Кожна повторювана транзакція буде автоматично створювати звичайну транзакцію опівночі у відповідний день.

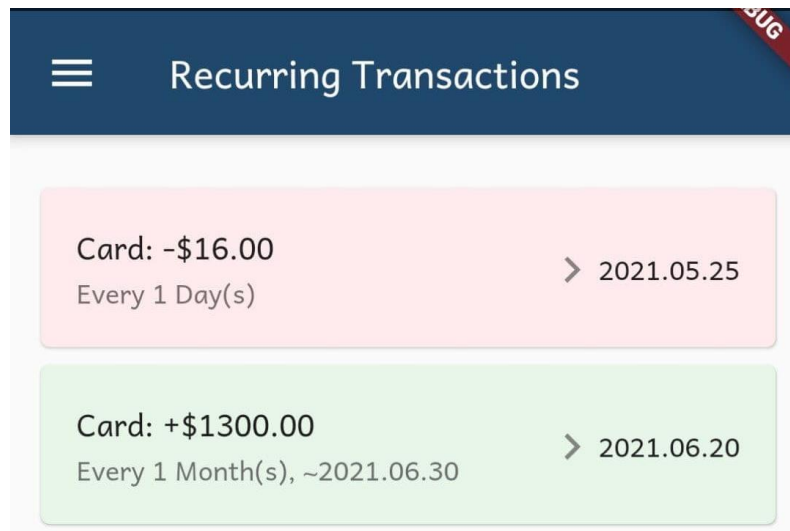


Рисунок 4.11 – Вклад з повторюваними транзакціями

4.3. Перегляд статистики

Статистику можна переглянути, якщо натиснути відповідну кнопку на головному екрані або ж проліснути головну сторінку вправо. На вкладці статистики можна переглянути статистику за весь період, за певний встановлений період часу або ж за кожен день.

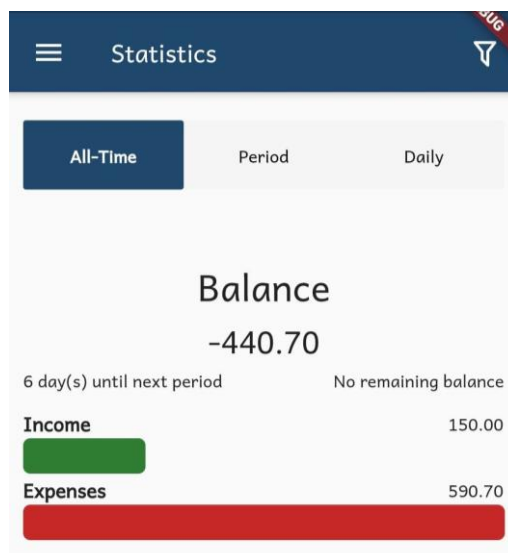


Рисунок 4.12 – Вкладка статистики

Тут показується баланс за обраний період, сума, яка залишилась до закінчення періоду та гістограма доходів та витрат. Після цього

показується кругова діаграма витрат з підписами і відсотковим відношенням суми витрати до загальної суми всіх витрат.

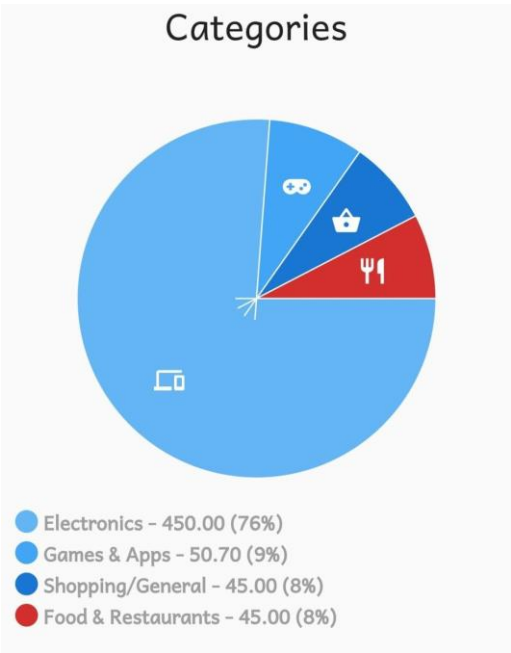


Рисунок 4.13 – Кругова діаграма витрат

І з самого низу сторінки показуються п'ять найбільших витрат. Якщо потрібно показати повний список, потрібно натиснути на кнопку «Show more», та «Show less» якщо потрібно зменшити цей список назад.

Top Expenses		
Card		450.00
Electronics		
Card		50.70
Games & Apps		
Card		45.00
Shopping/General		
Card		45.00
Food & Restaurants		
Show less		Show more

Рисунок 4.14 – Список найбільших витрат

Для відображення інформації за певними періодами їх потрібно створити. Для цього потрібно перейти через меню навігації у вкладку з періодами. Щоб створити період потрібно вибрати початковий день, назву, вибрати скільки цей період триватиме і, якщо потрібно, встановити його періодом за замовченням, який буде використовуватись до того часу, поки не обереться інший.

Рисунок 4.15 – Створення періоду

Після створення періоду у вкладці зі статистикою можна буде обирати проміжки часу, довжиною у вибраний період.

Рисунок 4.16 – Вибір проміжку за періодом

Після встановлення періоду всі транзакції будуть відображатись з розділами по проміжках часу, які визначені періодом за замовченням.

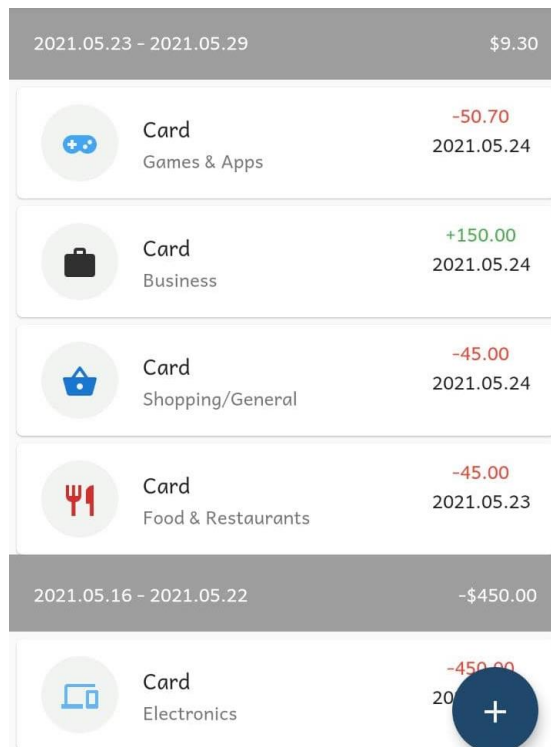


Рисунок 4.17 – Транзакції розділені по періодам

4.4. Налаштування категорій

На вкладці з категоріями можна переглянути всі наявні категорії в даному гаманці. Тут можна змінювати їх чергу для відображення а також «вимкнути» категорію, якщо вона не потрібна, а видаляти не хочеться. Якщо категорія «вимкнена», вона зникає зі списку можливих категорій для нових транзакцій, проте уже створені транзакції з даною категорією з нею ж і залишаються.

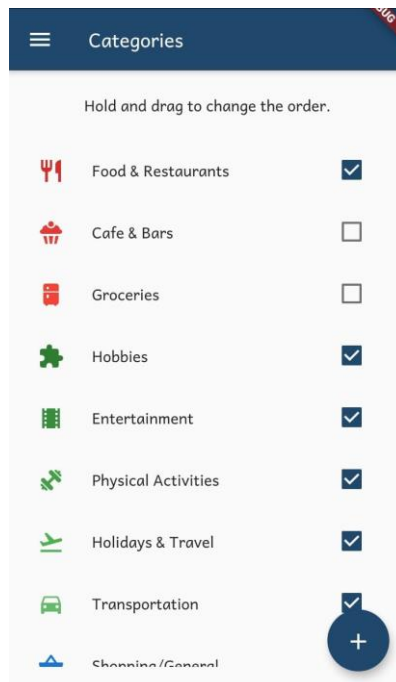


Рисунок 4.18 – Вкладка зі списком категорій

При натисканні на категорію відкривається вкладка для її редагування. Тут можна змінити назву, зображення та колір категорії. Якщо вона вже не потрібна, то через дану вкладку категорію можна видалити натиснувши на кнопку на верхній панелі додатку.

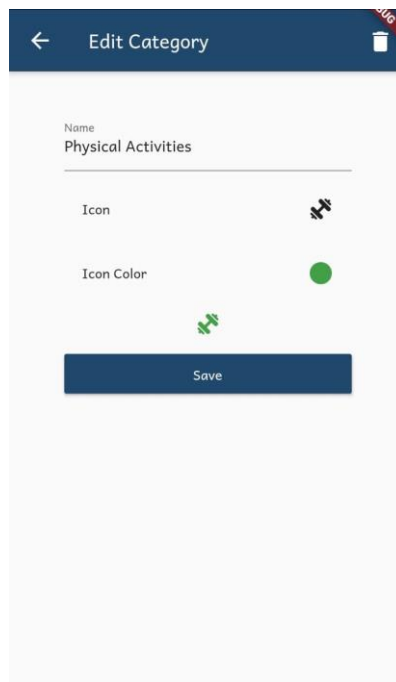


Рисунок 4.19 – Редагування категорії

Для створення нової категорії потрібно у вкладці з їх списком натиснути на кнопку з «+». Користувача перенаправляє на відповідну сторінку. Тут вводиться назва категорії, обирається іконка та колір, які надалі буде відображатися в інфографіці.

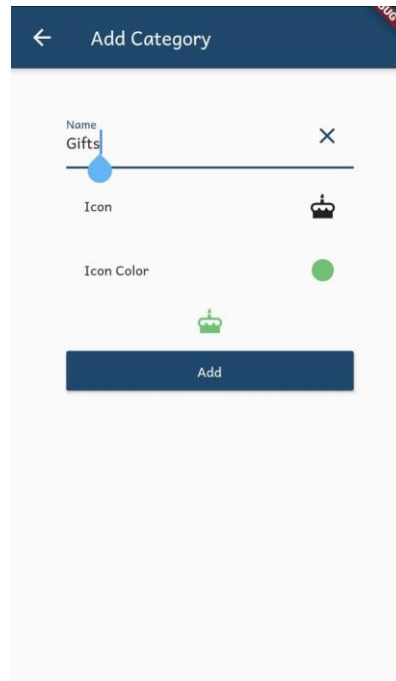


Рисунок 4.20 – Створення нової категорії

Висновок до розділу 4

В даному розділі був проведений аналіз розробленої системи.

Була розглянута детальна взаємодія користувача зі звичайними та повторюваними транзакціями, які відповідають за грошові потоки – їх створення, редагування та видалення. Було показано відображення інфорграфіки за весь час та можливість зміни періоду за рахунок власних створених періодів та за окремими днями.

В розділі наведені скріншоти, які відображають роботу додатку при зміні налаштувань періодів, роботі з категоріями, шаблонами транзакцій, створення різних типів транзакцій та відображення їх у списку наголовному екрані і побудова інфорграфіки на базі цих даних.

					ІАЛЦ.467200.003 ПЗ	Арк.
						69
Зм.	Арк.	№ документа	Підпис	Дата		

ВИСНОВОК

У даній дипломній бакалаврській роботі створено додакто для дослідження доходів та витрат. Додаток працює як і з наявним підключенням до мережі Internet, так і без нього. Дану функціональність забезпечує використання двох баз даних: віддаленої та локальної.

У першому розділі роботи було описано аналоги розроблюваного продукту. Після детального аналізу було проведено порівняння додатку дипломного проєкту з аналогами.

У другому розділі розглядалися сервіси для створення дизайн-проєкту додатку та технології і сервіси для розробки тезнічної частини. Серед технологій було описано наявні засоби розробки нативних додатків під Android-платформу та фреймворки для кросплатформених рішень. Серед можливих баз даних було обрано та описано дві, а саме Firebase Database, для віддаленого збереження даних, та SQLite, для локального збереження.

У третьому розділі було детально розглянуто розробку клієнтської частини. Також у цьому розділі деталі налаштування Firebase Cloud Project та підключення до додатку локальної бази даних SQLite.

У четвертому розділі було продемонстровано роботу розробленого програмного продукту та наведено скріншоти-приклади.

Під час розробки додатку було враховано всі вимоги, зазначені в плані даної дипломної бакалаврської роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bragg S. Budgeting: Fifth Edition: A Comprehensive Guide / Steven Bragg., 2020. – 269 с.
2. Brayfield A. Balancing the Family Budget: Differences in Child Care Expenditures by Race/Ethnicity, Economic Status, and Family Structure / A. Brayfield, S. Hofferth. // Social Science Quarterly. – 1995. – №76. – С. 158–177.
3. Сімейний бюджет: структура доходів та витрат [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.campwaltblog.com/4306165-family-budget-structure-of-income-and-expenses>.
4. Структура витрат сімейного бюджету [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/23971/politekonomiya/struktura_vitrat_simeynogo_byudzhetu.
5. Ramsey D. The Total Money Makeover: Classic Edition: A Proven Plan for Financial Fitness / Dave Ramsey., 2013.
6. Для клієнтів ПриватБанку довідка про стан рахунку стала ще простішою та зрозумілішою [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: <https://te.20minut.ua/novini-kompanij/dlya-klientiv-privatbanku-dovidka-pro-stan-rahunku-stala-sche-prostish-10585849.html>.
7. Off To A Great Start! First Steps With Spendee [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://medium.com/spendee/off-to-a-great-start-first-steps-with-spendee-567d3815e4ca>.
8. Василевский Э. Money Lover – приложение для тех, кто хочет учесть абсолютно все [Електронний ресурс] / Эрнест Василевский. – 2016. – Режим доступу до ресурсу: <https://androidinsider.ru/obzory->

prilozhenij/money-lover-prilozhenie-dlya-teh-kto-hochet-uchest-absolyutno-vse.html.

9. Software Reviews: Budgeting debt away. // Engineering & Technology. – 2013. – №7. – С. 92–93.

10. OneCoin — Учет финансов [Электронный ресурс] – Режим доступа до ресурсу:
<https://play.google.com/store/apps/details?id=com.ukolovapps.percival.wallet&hl=ru>.

11. Карпов А. С. Изучение предпочтений пользователей мобильных приложений для планирования личных финансов / А. С. Карпов, Н. С. Павлов. – 2020. – №2020. – С. 73–78.

12. Hsu Y. Mobile App Design for Teaching and Learning: Educators' Experiences in an Online Graduate Course / Y. Hsu, Y. Ching. // International Review of Research in Open and Distributed Learning. – 2013. – №14. – С. 117–139.

13. Прототип сайту – як він допомагає оцінювати і розробляти сайти [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:
<https://evergreens.com.ua/ua/articles/prototype-site.html>.

14. Момотов А. Как подготовить макет интерфейса мобильного приложения к передаче в разработку? [Электронный ресурс] / Александр Момотов. – 2018. – Режим доступа до ресурсу: <https://designpub.ru/как-подготовить-макет-интерфейса-мобильного-приложения-к-передачи-в-разработку-50ec94c2f039>.

15. NinjaMock — бесплатное приложение для прототипирования интерфейсов [Электронный ресурс]. – 2014. – Режим доступа до ресурсу:
<https://habr.com/ru/company/ninjamock/blog/208406/>.

16. Moqups — новый сервис для создания макетов UI [Электронный ресурс]. – 2012. – Режим доступа до ресурсу:
<https://habr.com/ru/post/147571/>.

17. Баданов А. Г. Creately [Електронний ресурс] / Александр Геннадьевич Баданов – Режим доступу до ресурсу: <https://sites.google.com/site/badanovweb2/home/creately>.
18. Степанов С. Что такое Figma: возможности и принципы работы [Електронний ресурс] / Степан Степанов. – 2019. – Режим доступу до ресурсу: https://skillbox.ru/media/design/что_такое_figma/.
19. Окунев А. Руководство по Figma / Александр Окунев., 2019. – 256 с.
20. Типи мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: <https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types>.
21. Нативні додатки [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://cheport.com.ua/blog/native-app>.
22. A Developer's First Look At Android. // LINUX FOR YOU. – 2008. – С. 48–50.
23. Get started with native Android development on Windows [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/windows/android/native-android>.
24. 14 Best Android Frameworks for App Development in 2020 [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://www.intellectsoft.net/blog/best-android-frameworks/>.
25. Son K. The method of android application speed up by using NDK / K. Son, J. Lee. // International Conference on Awareness Science and Technology. – 2011.
26. Шматко О. В. Аналіз методів і технологій розробки мобільних додатків для платформи android. Частина 2 / О. В. Шматко, А. О. Поляков, В. М. Федорченко. – Харків: Видавничий центр НТУ «ХПІ», 2018. – 286 с.
27. Smith R. Top 10 Best Cross Platform App Development Frameworks in 2021 [Електронний ресурс] / Ryan Smith. – 2021. – Режим

доступу до ресурсу: <https://www.codenameone.com/blog/top-10-best-cross-platform-app-development-frameworks-in-2021.html>.

28. Flutter is no longer a cross-platform framework — it is something more. [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://itnext.io/flutter-is-no-longer-a-cross-platform-framework-b53c87b14c39>.

29. Матросов К. Как работает Flutter [Електронний ресурс] / Кирилл Матросов. – 2019. – Режим доступу до ресурсу: <https://habr.com/ru/post/476018/>.

30. Кольцов Н. Как разработать своё первое приложение на React Native [Електронний ресурс] / Никита Кольцов. – 2020. – Режим доступу до ресурсу: <https://tproger.ru/articles/your-first-app-in-react-native/>.

31. Hermes D. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals / Dan Hermes., 2015. – 470 с.

32. Smallcombe M. SQL vs NoSQL: 5 Critical Differences [Електронний ресурс] / Mark Smallcombe. – 2020. – Режим доступу до ресурсу: <https://www.xplenty.com/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20relational%2C%20NoSQL%20are%20non%2Drelational.&text=NoSQL%20databases%20have%20dynamic%20schemas,graph%20or%20wide%2Dcolumn%20stores..>

33. Owens M. The Definitive Guide to SQLite / Mike Owens., 2006. – 440 с.

34. Kumar A. Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase / Ashok Kumar., 2018. – 394 с.

ДОДАТОК 1

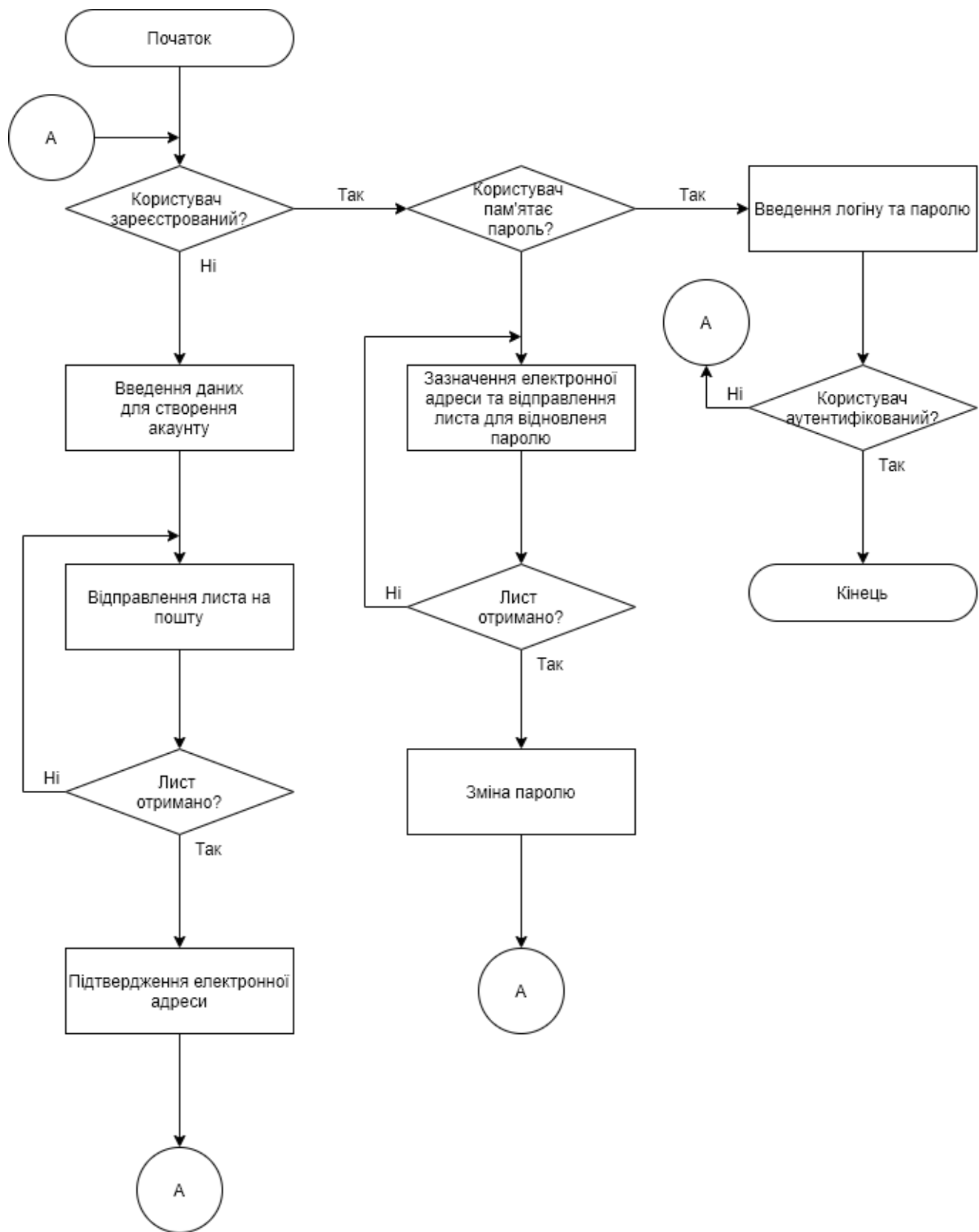
Додаток з планування сімейного бюджету на платформі
Android

Алгоритм аутентифікації

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ – 2021 р.



					ІАЛЦ.467200.004 Д1		
Зм.	Арк.	№ документа	Підпис	Дата	Додаток з планування сімейного бюджету на платформі Android Алгоритм аутентифікації		
Розробив		Дубина М. С.					
Перевірив		Ружевський М. С.					
Н. контр.		Сімоненко В. П.					
Затверд.							
					Літ.	Аркуш	Аркушів
						1	1
					НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІІ-73		

ДОДАТОК 2

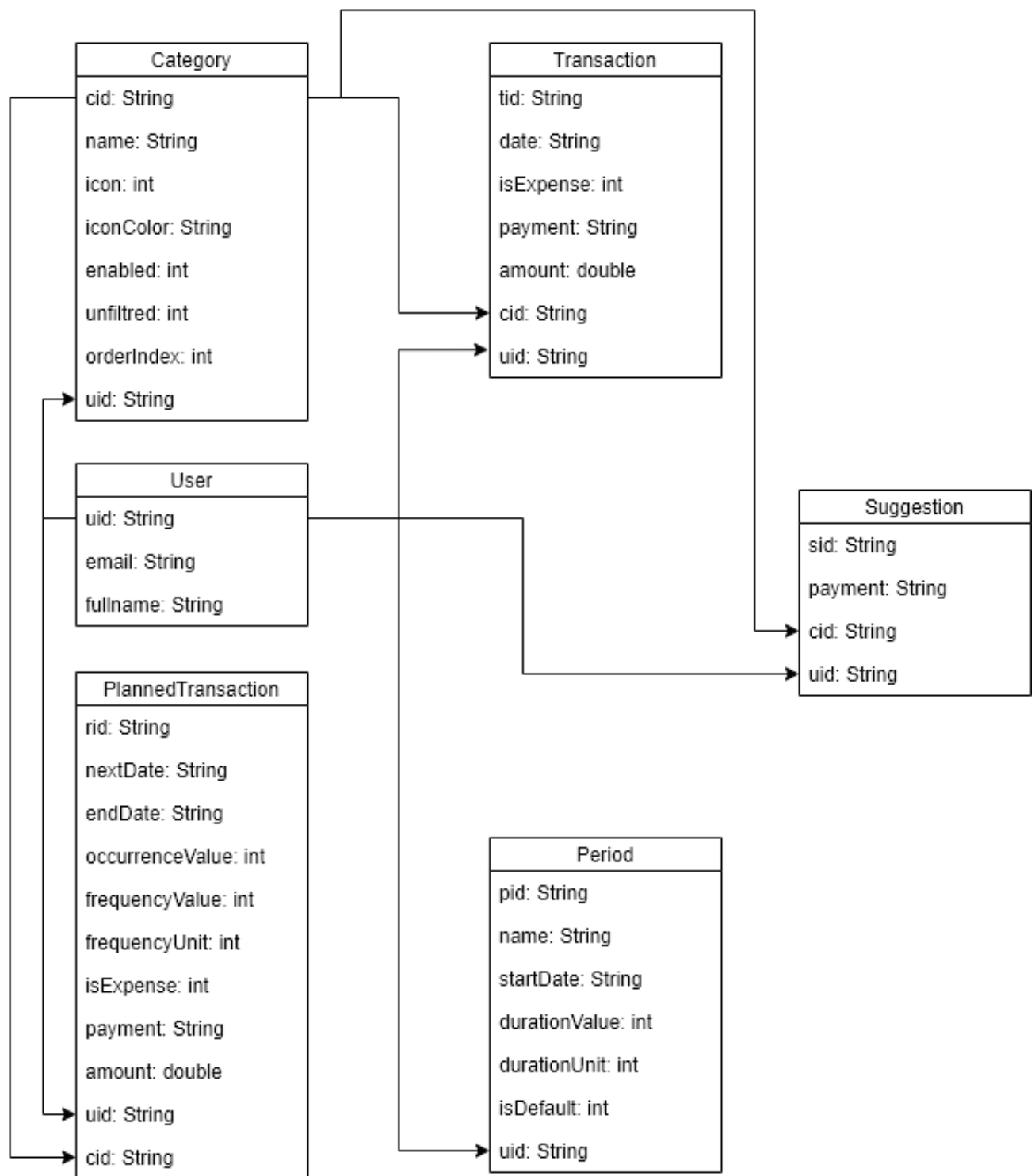
Додаток з планування сімейного бюджету на платформі
Android

Загальна діаграма моделей баз даних

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ – 2021 р.



					ІАЛЦ.467200.005 Д2			
Зм.	Арк.	№ документа	Підпис	Дата	Додаток з планування сімейного бюджету на платформі Android Загальна діаграма моделей баз даних	Літ.	Аркуш	Аркушів
Розробив	Дубина М. С.						1	1
Перевірив	Ружевський М. С.							
Н. контр.	Сімоненко В. П.					НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІІ-73		
Затверд.								

ДОДАТОК 3

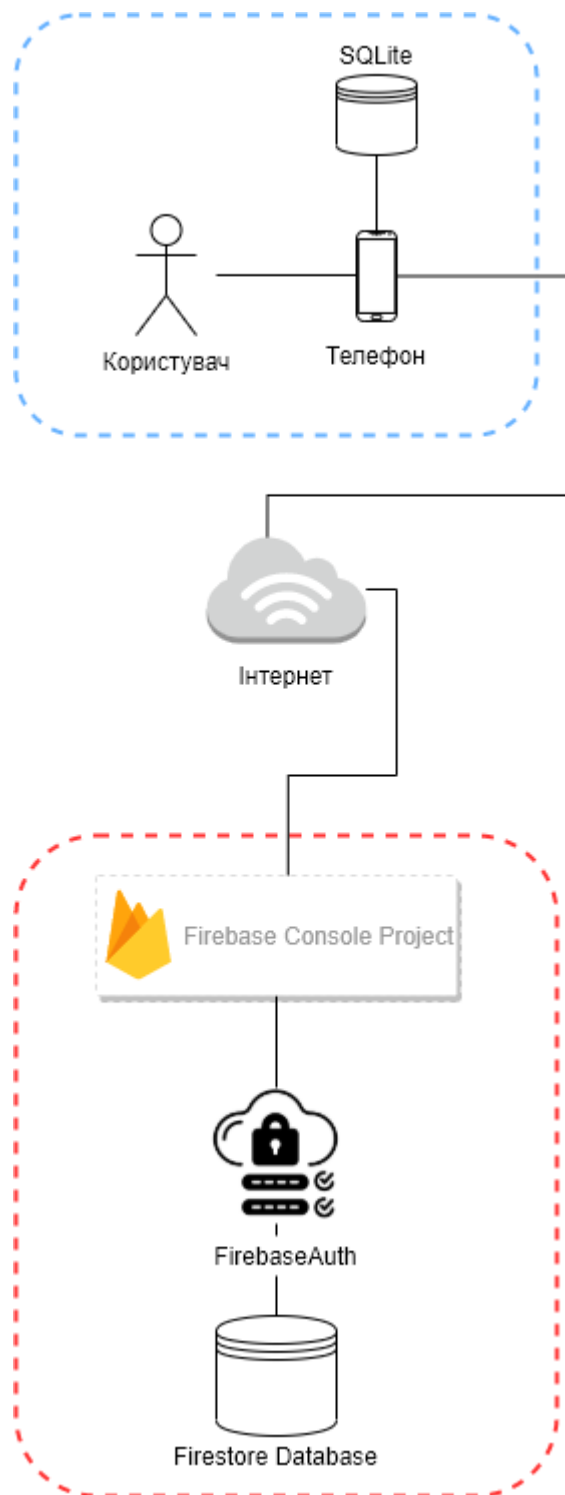
Додаток з планування сімейного бюджету на платформі
Android

Діаграма взаємодії

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ – 2021 р.



					ІАЛЦ.467200.006 ДЗ			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Дубина М. С.			Додаток з планування сімейного бюджету на платформі Android	Літ.	Аркуш	Аркушів
Перевірив		Ружевський М. С.					1	1
Н. контр.		Сімоненко В. П.			Діаграма взаємодії	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІ-73		
Затверд.								

ДОДАТОК 4

Додаток з планування сімейного бюджету на платформі
Android

Текст програми

ІАЛЦ.467200.007 Д4

Аркушів 26

Київ – 2021 р.

```

import 'package:firebase_auth/firebase_auth.dart' as FirebaseAuthentication
show User;
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:thesis_app/wrapper.dart';
import 'package:thesis_app/services/auth.dart';
import 'package:provider/provider.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(App());
}

class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        FocusScopeNode currentFocus = FocusScope.of(context);
        if (!currentFocus.hasPrimaryFocus &&
            currentFocus.focusedChild != null) {
          currentFocus.focusedChild.unfocus();
        }
      },
      child: StreamProvider<FirebaseAuthentication.User>(
        create: (_) => AuthService().user,
        child: MaterialApp(
          home: Wrapper(),
          theme: ThemeData(
            primaryColor: Color(0xff20476c),
            accentColor: Color(0xff4d4b6a),
            fontFamily: 'Andika',
            backgroundColor: Color(0xffff2f4f1),
          ),
        ),
      ),
    );
  }
}

```

```

import 'package:firebase_auth/firebase_auth.dart' as FirebaseAuthentication
show User;
import 'package:flutter/material.dart';
import 'package:thesis_app/pages/auth/authWrapper.dart';
import 'package:thesis_app/pages/home/home.dart';
import 'package:provider/provider.dart';

class Wrapper extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    dynamic _user = Provider.of<FirebaseAuthentication.User>(context);
  }
}

```

					ІАЛЦ.467200.007 Д4			
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Дубина М. С.			Додаток з планування сімейного бюджету на платформі Android	Літ.	Аркуш	Аркушів
Перевірів		Ружевський М. С.					1	26
Н. контр.		Сімоненко В. П.			Текст програми	НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІІІ-73		
Затверд.								

```

        if (_user != null && _user.emailVerified) {
          return Home(user: _user);
        } else {
          return AuthWrapper();
        }
      }
    }
  }

  import 'package:flutter/material.dart';
  import 'package:thesis_app/pages/auth/authForm.dart';
  import 'package:thesis_app/shared/constants.dart';

  class AuthWrapper extends StatefulWidget {
    @override
    _AuthWrapperState createState() => _AuthWrapperState();
  }

  class _AuthWrapperState extends State<AuthWrapper> {
    AuthMethod method = AuthMethod.SignIn;

    void toggleView() {
      setState(() {
        method = (method == AuthMethod.SignIn)
          ? AuthMethod.Register
          : AuthMethod.SignIn;
      });
    }

    @override
    Widget build(BuildContext context) {
      switch (method) {
        case AuthMethod.SignIn:
          return AuthForm(toggleView: toggleView, method: AuthMethod.SignIn);
          break;
        case AuthMethod.Register:
          return AuthForm(toggleView: toggleView, method: AuthMethod.Register);
          break;
        default:
          return Text('Error in the AuthWrapper.');
```

```

          break;
      }
    }
  }

  import 'package:flutter/material.dart';
  import 'package:thesis_app/models/user.dart';
  import 'package:thesis_app/pages/auth/emailVerification.dart';
  import 'package:thesis_app/pages/auth/forgotPassword.dart';
  import 'package:thesis_app/pages/auth/validators.dart';
  import 'package:thesis_app/services/auth.dart';
  import 'package:thesis_app/services/databaseWrapper.dart';
  import 'package:thesis_app/services/sync.dart';
  import 'package:thesis_app/shared/constants.dart';
  import 'package:thesis_app/shared/styles.dart';
  import 'package:thesis_app/shared/components.dart';

  class AuthForm extends StatefulWidget {
    final Function toggleView;
    final AuthMethod method;

    AuthForm({this.toggleView, this.method});
  }

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		2

```

@override
_AuthFormState createState() => _AuthFormState();
}

class _AuthFormState extends State<AuthForm> {
  final AuthService _auth = AuthService();
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final _passwordConfirmController = TextEditingController();
  final _fullnameController = TextEditingController();
  final FocusNode _emailFocus = new FocusNode();
  final FocusNode _passwordFocus = new FocusNode();
  final FocusNode _passwordConfirmFocus = new FocusNode();
  final FocusNode _fullnameFocus = new FocusNode();
  bool _isEmailInFocus = false;
  bool _isPasswordInFocus = false;
  bool _isPasswordConfirmInFocus = false;
  bool _isFullnameInFocus = false;
  String _email = '';
  String _password = '';
  String _passwordConfirm = '';
  String _fullname = '';
  String _error = '';
  bool _isLoading = false;
  bool _obscurePassword = true;
  bool _obscurePasswordConfirm = true;
  Widget title(bool isRegister) {
    return Text(
      isRegister ? 'Register' : 'Sign In',
      style: TextStyle(
        color: Colors.white,
      ),
    );
  }
  Widget authToggleText(bool isRegister) {
    return Text(isRegister ? 'Sign In' : 'Register');
  }

  @override
  void initState() {
    super.initState();
    _emailFocus.addListener(_checkFocus);
    _passwordFocus.addListener(_checkFocus);
    _passwordConfirmFocus.addListener(_checkFocus);
    _fullnameFocus.addListener(_checkFocus);
  }

  void _checkFocus() {
    setState(() {
      _isEmailInFocus = _emailFocus.hasFocus;
      _isPasswordInFocus = _passwordFocus.hasFocus;
      _isPasswordConfirmInFocus = _passwordConfirmFocus.hasFocus;
      _isFullnameInFocus = _fullnameFocus.hasFocus;
    });
  }

  @override
  Widget build(BuildContext context) {

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		3

```

bool isRegister = widget.method == AuthMethod.Register;
return Scaffold(
  backgroundColor: Colors.grey[100],
  appBar: AppBar(
    title: title(isRegister),
    actions: <Widget>[
      FlatButton(
        textColor: Colors.white,
        child: authToggleText(isRegister),
        onPressed: () {
          widget.toggleView();
        },
      ),
    ],
  ),
  body: _isLoading
    ? Loader()
    : Form(
      key: _formKey,
      child: ListView(
        padding: formPadding,
        children: <Widget>[
          SizedBox(height: 10.0),
          TextFormField(
            controller: _emailController,
            focusNode: _emailFocus,
            autovalidate: _email.isNotEmpty,
            validator: emailValidator,
            decoration: clearInput(
              labelText: 'Email',
              enabled: _email.isNotEmpty && _isEmailInFocus,
              onPressed: () {
                setState(() => _email = '');
                _emailController.safeClear();
              },
            ),
            keyboardType: TextInputType.emailAddress,
            onChanged: (val) {
              setState(() => _email = val);
            },
          ),
          TextFormField(
            controller: _passwordController,
            focusNode: _passwordFocus,
            autovalidate: _password.isNotEmpty,
            validator: passwordValidator,
            obscureText: _obscurePassword,
            decoration: clearInput(
              labelText: 'Password',
              enabled: _password.isNotEmpty && _isPasswordInFocus,
              onPressed: () {
                setState(() => _password = '');
                _passwordController.safeClear();
              },
            ),
            passwordToggle: true,
            onPasswordTogglePressed: () => setState(
              () => _obscurePassword = !_obscurePassword,
            ),
            passwordToggleVisible: _obscurePassword,
          ),
        ],
      ),
    ),
);

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		4

```

        onChanged: (val) => setState(() => _password = val),
      ),
    if (isRegister) ...[
      TextFormField(
        controller: _passwordConfirmController,
        focusNode: _passwordConfirmFocus,
        autovalidate: _passwordConfirm.isEmpty,
        validator: (val) =>
          passwordConfirmValidator(val, _password),
        obscureText: _obscurePasswordConfirm,
        decoration: clearInput(
          labelText: 'Confirm Password',
          enabled: _passwordConfirm.isEmpty &&
            !_isPasswordConfirmInFocus,
          onPressed: () {
            setState(() => _passwordConfirm = '');
            _passwordConfirmController.safeClear();
          },
          passwordToggle: true,
          onPasswordTogglePressed: () => setState(
            () => _obscurePasswordConfirm =
              !_obscurePasswordConfirm,
          ),
          passwordToggleVisible: _obscurePasswordConfirm,
        ),
        onChanged: (val) {
          setState(() => _passwordConfirm = val);
        },
      ),
      TextFormField(
        controller: _fullnameController,
        focusNode: _fullnameFocus,
        autovalidate: _fullname.isEmpty,
        validator: fullNameValidator,
        textCapitalization: TextCapitalization.words,
        decoration: clearInput(
          labelText: 'Full Name',
          enabled: _fullname.isEmpty && _isFullnameInFocus,
          onPressed: () {
            setState(() => _fullname = '');
            _fullnameController.safeClear();
          },
        ),
        onChanged: (val) => setState(() => _fullname = val),
      )
    ],
  ),
  SizedBox(height: 10.0),
  RaisedButton(
    color: Theme.of(context).primaryColor,
    child: title(isRegister),
    onPressed: () async {
      setState(() => _error = '');
      if (_formKey.currentState.validate()) {
        setState(() => _isLoading = true);
        dynamic _user = isRegister
          ? await _auth.register(_email, _password)
          : await _auth.signIn(_email, _password);
        if (_user is String) {
          setState(() {
            _error = _user;
          });
        }
      }
    },
  ),
);

```

```

        _isLoading = false;
    });
    } else {
    if (!isRegister) {
        if (_user == null) {
            await showEmailVerificationDialog(context, _auth);
        } else {
            await SyncService(_user.uid).syncToLocal();
        }
    } else {
        await Future.wait([
            DatabaseWrapper(_user.uid).addUser(
                User(
                    uid: _user.uid,
                    email: _email,
                    fullname: _fullname,
                ),
            ),
            DatabaseWrapper(_user.uid).addDefaultCategories(),
            DatabaseWrapper(_user.uid)
                .addDefaultPreferences(),
        ]);
        await showEmailVerificationDialog(context, _auth);
        widget.toggleView();
    }
    }
    },
    ),
    SizedBox(height: 12.0),
    Text(
        _error,
        style: TextStyle(
            color: Theme.of(context).primaryColor,
            fontSize: 14.0,
        ),
    ),
    ),
    (!isRegister)
        ? FlatButton(
            child: Text('Forgot password?'),
            onPressed: () async {
                await showForgotPasswordDialog(context, _auth);
            },
        )
        : SizedBox(
            height: 1,
        ),
    ],
    ),
    ),
);
}
Future showEmailVerificationDialog(
    BuildContext context, AuthService auth) async {
    await showDialog(
        context: context,
        builder: (context) {
            return EmailVerification(
                auth: auth,
            );
        },
    );
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		6

```

    },
  );
  setState(() => _isLoading = false);
}
Future showForgotPasswordDialog(
  BuildContext context, AuthService auth) async {
  await showDialog(
    context: context,
    builder: (context) {
      return ForgotPassword(
        auth: auth,
      );
    },
  );
  setState(() => _isLoading = false);
}
}

extension on TextEditingController {
  void safeClear() {
    WidgetsBinding.instance.addPostFrameCallback((_) {
      this.clear();
    });
  }
}

import 'package:flutter/material.dart';
import 'package:thesis_app/services/auth.dart';
import 'package:thesis_app/shared/components.dart';
import 'package:thesis_app/shared/styles.dart';

class EmailVerification extends StatefulWidget {
  final AuthService auth;

  EmailVerification({this.auth});

  @override
  _EmailVerificationState createState() => _EmailVerificationState();
}

class _EmailVerificationState extends State<EmailVerification> {
  bool _emailVerificationSent = false;
  bool _isLoading = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        title: Text('Email Verification'),
      ),
      body: Container(
        child: ListView(
          padding: formPadding,
          children: <Widget>[
            Center(child: Text('Please verify your email')),
            RaisedButton(
              color: Theme.of(context).primaryColor,
              child: Text(
                'Resend email verification',
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		7

```

        style: TextStyle(color: Colors.white),
      ),
      onPressed: () async {
        setState(() {
          _emailVerificationSent = false;
          _isLoading = true;
        });
        await widget.auth.sendEmailVerification();
        setState(() {
          _emailVerificationSent = true;
          _isLoading = false;
        });
      },
    ),
    if (_isLoading) ...[Loader()],
    if (_emailVerificationSent) ...[
      Center(child: Text('New email verification sent!'))
    ]
  ],
),
);}}

```

```

import 'package:flutter/material.dart';
import 'package:thesis_app/pages/auth/validators.dart';
import 'package:thesis_app/services/auth.dart';
import 'package:thesis_app/shared/components.dart';
import 'package:thesis_app/shared/styles.dart';

class ForgotPassword extends StatefulWidget {
  final AuthService auth;

  ForgotPassword({this.auth});

  @override
  _ForgotPasswordState createState() => _ForgotPasswordState();
}

class _ForgotPasswordState extends State<ForgotPassword> {
  bool _forgotPasswordEmailSent = false;
  bool _isLoading = false;
  final _emailController = TextEditingController();
  final FocusNode _emailFocus = new FocusNode();
  bool _isEmailInFocus = false;
  String _email = '';

  @override
  void initState() {
    super.initState();
    _emailFocus.addListener(_checkFocus);
  }

  void _checkFocus() {
    setState(() {
      _isEmailInFocus = _emailFocus.hasFocus;
    });
  }

  @override

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		8

```

Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[100],
    appBar: AppBar(
      title: Text('Forgot Password?'),
    ),
    body: Container(
      child: ListView(
        padding: formPadding,
        children: <Widget>[
          TextFormField(
            controller: _emailController,
            focusNode: _emailFocus,
            autovalidate: _email.isNotEmpty,
            validator: emailValidator,
            decoration: clearInput(
              labelText: 'Email',
              enabled: _email.isNotEmpty && _isEmailInFocus,
              onPressed: () {
                setState(() => _email = '');
                _emailController.safeClear();
              },
            ),
            keyboardType: TextInputType.emailAddress,
            onChanged: (val) {
              setState(() => _email = val);
            },
          ),
          SizedBox(height: 10.0),
          RaisedButton(
            color: Theme.of(context).primaryColor,
            child: Text(
              'Send password reset email',
              style: TextStyle(color: Colors.white),
            ),
          ),
          onPressed: () async {
            setState(() {
              _forgotPasswordEmailSent = false;
              _isLoading = true;
            });
            await widget.auth.sendPasswordResetEmail(_email);
            setState(() {
              _forgotPasswordEmailSent = true;
              _isLoading = false;
            });
          },
        ),
        if (_isLoading) ...[Loader()],
        if (_forgotPasswordEmailSent) ...[
          Center(child: Text('Password reset email sent!'))
        ]
      ],
    ),
  );
}
}

extension on TextEditingController {
  void safeClear() {

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		9

```

        WidgetsBinding.instance.addPostFrameCallback((_) {
            this.clear();
        });
    }
}

import 'package:community_material_icon/community_material_icon.dart';
import 'package:firebase_auth/firebase_auth.dart' as FirebaseAuth show
User;
import 'package:flutter/material.dart';
import 'package:thesis_app/models/category.dart';
import 'package:thesis_app/models/period.dart';
import 'package:thesis_app/models/preferences.dart';
import 'package:thesis_app/models/suggestion.dart';
import 'package:thesis_app/models/transaction.dart';
import 'package:thesis_app/services/databaseWrapper.dart';
import 'package:thesis_app/services/fireDB.dart';
import 'package:thesis_app/services/localDB.dart';
import 'package:thesis_app/services/plannedTransactions.dart';
import 'package:thesis_app/services/search.dart';
import 'package:thesis_app/services/sync.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/components.dart';

import 'package:thesis_app/pages/filters/filterList.dart';
import 'package:thesis_app/pages/statistics/statistics.dart';
import 'package:thesis_app/pages/transactions/transactionForm.dart';
import 'package:thesis_app/pages/transactions/transactionsList.dart';
import 'package:thesis_app/pages/home/mainDrawer.dart';

import 'package:provider/provider.dart';

class Home extends StatefulWidget {
    final FirebaseAuth.User user;

    Home({this.user});

    @override
    _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> with WidgetsBindingObserver {
    PageController _pageController = PageController();
    List<Transaction> _transactions;
    List<Category> _categories;
    Period _currentPeriod;
    Preferences _prefs;
    List<Suggestion> _hiddenSuggestions;
    int _selectedIndex = 0;
    List<String> categoriesFiltered = [];
    bool isAnyCategoryFiltered = false;

    @override
    void initState() {
        super.initState();
        retrieveNewData(widget.user.uid);
        WidgetsBinding.instance.addObserver(this);
    }

    @override

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		10

```

void didChangeAppLifecycleState(AppLifecycleState state) async {
  if (state == AppLifecycleState.resumed) {
    retrieveNewData(widget.user.uid);
  }
}

@override
void dispose() {
  WidgetsBinding.instance.removeObserver(this);
  super.dispose();
}

@override
Widget build(BuildContext context) {
  if (_categories != null) {
    categoriesFiltered = _categories
      .where((cat) => !cat.unfiltered)
      .map((cat) => cat.cid)
      .toList();
    isAnyCategoryFiltered = categoriesFiltered.length > 0;
    _transactions = _transactions
      .where((tx) => !categoriesFiltered.contains(tx.cid))
      .toList();
    if (!_prefs.incomeUnfiltered) {
      _transactions = _transactions.where((tx) => tx.isExpense).toList();
    }
    if (!_prefs.expensesUnfiltered) {
      _transactions = _transactions.where((tx) => !tx.isExpense).toList();
    }
  }
  final List<Map<String, dynamic>> _pages = [
    {
      'name': 'Records',
      'actions': <Widget>[
        IconButton(
          icon: Icon(CommunityMaterialIcons.magnify),
          onPressed: () => showTxSearch(_transactions),
        ),
        filterCategoriesButton(),
      ],
      'body': TransactionsList(
        transactions: _transactions,
        categories: _categories,
        currentPeriod: _currentPeriod,
        hiddenSuggestions: _hiddenSuggestions,
        refreshList: () => retrieveNewData(widget.user.uid),
      ),
      'addButton': FloatingButton(
        context,
        page: MultiProvider(
          providers: [
            FutureProvider<List<Transaction>>.value(
              value: DatabaseWrapper(widget.user.uid).getTransactions(),
            ),
            FutureProvider<List<Category>>.value(
              value: DatabaseWrapper(widget.user.uid).getCategories(),
            ),
          ],
          child: TransactionForm(
            hiddenSuggestions: _hiddenSuggestions,
            getTxOrRecTx: () => Transaction.empty(),
          ),
        ),
      ),
    },
  ];
}

```

```

        ),
        callback: () => retrieveNewData(widget.user.uid),
    ),
  },
  {
    'name': 'Statistics',
    'actions': <Widget>[
      filterCategoriesButton(),
    ],
    'body': Statistics(
      allTransactions: _transactions,
      categories: _categories,
      currentPeriod: _currentPeriod,
      prefs: _prefs,
    ),
  }
];
return Scaffold(
  drawer: MainDrawer(user: widget.user, openPage: openPage),
  appBar: AppBar(
    title: Text(_pages[_selectedIndex]['name']),
    actions: _pages[_selectedIndex]['actions'],
  ),
  body: PageView(
    controller: _pageController,
    onPageChanged: (index) {
      setState(() {
        _selectedIndex = index;
      });
    },
    children: <Widget>[_pages[0]['body'], _pages[1]['body']],
  ),
  floatingActionButton: _pages[_selectedIndex]['addButton'],
  bottomNavigationBar: BottomNavigationBar(
    items: <BottomNavigationBarItem>[
      BottomNavigationBarItem(
        icon: Icon(CommunityMaterialIcons.file_document),
        title: Text('Records'),
      ),
      BottomNavigationBarItem(
        icon: Icon(CommunityMaterialIcons.chart_pie),
        title: Text('Statistics'),
      ),
    ],
    currentIndex: _selectedIndex,
    onTap: (index) {
      if (_selectedIndex != index) {
        _pageController.jumpToPage(index);
      }
    },
  ),
);
}

Widget filterCategoriesButton() {
  return IconButton(
    icon: Icon(isAnyCategoryFiltered
      ? CommunityMaterialIcons.filter
      : CommunityMaterialIcons.filter_outline),
    onPressed: () async {

```

```

        await showDialog(
          context: context,
          builder: (context) {
            return FilterList(
              user: widget.user,
              openPage: openPage,
            );
          },
        );
        retrieveNewData(widget.user.uid);
      },
    );
  }

void showTxSearch(List<Transaction> txs, {String prevQuery}) async {
  await showSearch(
    context: context,
    query: prevQuery,
    delegate: SearchService(
      transactions: txs,
      categories: _categories,
      currentPeriod: _currentPeriod,
      prefs: _prefs,
      hiddenSuggestions: _hiddenSuggestions,
      refreshList: (prevQuery) async {
        Navigator.pop(context);
        List<Transaction> newTxs =
          await DatabaseWrapper(widget.user.uid).getTransactions();
        showTxSearch(newTxs, prevQuery: prevQuery);
      },
    ),
  );
  retrieveNewData(widget.user.uid);
}

void retrieveNewData(String uid) async {
  List<Future> dataFutures = [];

  if (await LocalDBService().getUser(uid) == null) {
    dataFutures.add(FireDBService(uid).getTransactions());
    dataFutures.add(FireDBService(uid).getCategories());
    dataFutures.add(FireDBService(uid).getDefaultPeriod());
    dataFutures.add(FireDBService(uid).getPreferences());
    dataFutures.add(FireDBService(uid).getHiddenSuggestions());
  } else {
    await PlannedTransactionsService.checkPlannedTransactions(uid);
    SyncService(uid).syncPlannedTransactions();
    dataFutures.add(DatabaseWrapper(uid).getTransactions());
    dataFutures.add(DatabaseWrapper(uid).getCategories());
    dataFutures.add(DatabaseWrapper(uid).getDefaultPeriod());
    dataFutures.add(DatabaseWrapper(uid).getPreferences());
    dataFutures.add(DatabaseWrapper(uid).getHiddenSuggestions());
  }

  List<dynamic> data = await Future.wait(dataFutures);

  setState(() {
    _transactions = data[0];
    _categories = data[1];
    _currentPeriod = data[2];
  });
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		13

```

        _prefs = data[3];
        _hiddenSuggestions = data[4];
    });
}

void openPage(Widget page) async {
    goHome(context);
    await showDialog(
        context: context,
        builder: (context) {
            return page;
        },
    );
    retrieveNewData(widget.user.uid);
}
}

import 'package:community_material_icon/community_material_icon.dart';
import 'package:firebase_auth/firebase_auth.dart' as FirebaseAuthentication show
User;
import 'package:flutter/material.dart';
import 'package:thesis_app/models/period.dart';
import 'package:thesis_app/models/user.dart';
import 'package:thesis_app/pages/categories/categoriesList.dart';
import 'package:thesis_app/pages/periods/periodsList.dart';
import 'package:thesis_app/pages/preferences/preferencesForm.dart';
import
'package:thesis_app/pages/plannedTransactions/plannedTransactionsList.dart';
import 'package:thesis_app/pages/suggestions/suggestionsList.dart';
import 'package:thesis_app/services/auth.dart';
import 'package:thesis_app/services/databaseWrapper.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/components.dart';
import 'package:provider/provider.dart';

class MainDrawer extends StatefulWidget {
    final FirebaseAuthentication.User user;
    final Function openPage;

    MainDrawer({this.user, this.openPage});

    @override
    _MainDrawerState createState() => _MainDrawerState();
}

class _MainDrawerState extends State<MainDrawer> {
    final AuthService _auth = AuthService();
    User userInfo;

    @override
    void initState() {
        super.initState();
        DatabaseWrapper(widget.user.uid).getUser().then((user) {
            setState(() => userInfo = user);
        });
    }

    @override
    Widget build(BuildContext context) {
        return Drawer(

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		14

```

child: ListView(
  children: <Widget>[
    UserAccountsDrawerHeader(
      accountName: Text(userInfo != null ? userInfo.fullname : ''),
      accountEmail: Text(userInfo != null ? userInfo.email : ''),
      currentAccountPicture: CircleAvatar(
        backgroundColor: Colors.white,
        child: Text(
          userInfo != null && userInfo.fullname.length > 0 ?
userInfo.fullname[0] : '',
          style: TextStyle(
            fontSize: 40.0,
            color: Theme.of(context).primaryColor,
          ),
        ),
      ),
    ),
    ListTile(
      title: Text('Home'),
      leading: Icon(CommunityMaterialIcons.home),
      onTap: () => goHome(context),
    ),
    ListTile(
      title: Text('Categories'),
      leading: Icon(CommunityMaterialIcons.shape),
      onTap: () => widget.openPage(
        CategoriesList(
          user: widget.user,
          openPage: widget.openPage,
        ),
      ),
    ),
    ListTile(
      title: Text('Periods'),
      leading: Icon(CommunityMaterialIcons.calendar_range),
      onTap: () => widget.openPage(
        FutureProvider<List<Period>>(
          create: (_) => DatabaseWrapper(widget.user.uid).getPeriods(),
          child: PeriodsList(
            user: widget.user,
            openPage: widget.openPage,
          ),
        ),
      ),
    ),
    ListTile(
      title: Text('Recurring Transactions'),
      leading: Icon(CommunityMaterialIcons.history),
      onTap: () => widget.openPage(
        PlannedTransactionsList(
          user: widget.user,
          openPage: widget.openPage,
        ),
      ),
    ),
    ListTile(
      title: Text('Suggestions'),
      leading: Icon(CommunityMaterialIcons.lightbulb_outline),
      onTap: () => widget.openPage(
        SuggestionsList(

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		15

```

        user: widget.user,
        openPage: widget.openPage,
      ),
    ),
  ),
  ListTile(
    title: Text('Sign Out'),
    leading: Icon(CommunityMaterialIcons.logout),
    onTap: () async {
      bool hasBeenConfirmed = await showDialog(
        context: context,
        builder: (BuildContext context) {
          return Alert('You will be signed out.');
```

```

        },
      ) ??
      false;
      if (hasBeenConfirmed) {
        goHome(context);
        _auth.signOut();
      }
    },
  ),
],
),
);
}
}

import 'package:flutter/material.dart';
import 'package:thesis_app/models/category.dart';
import 'package:thesis_app/models/period.dart';
import 'package:thesis_app/models/preferences.dart';
import 'package:thesis_app/models/transaction.dart';
import 'package:thesis_app/pages/statistics/balance.dart';
import 'package:thesis_app/pages/statistics/categories.dart';
import 'package:thesis_app/pages/statistics/periodic.dart';
import 'package:thesis_app/pages/statistics/topExpenses.dart';
import 'package:thesis_app/shared/constants.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/styles.dart';
import 'package:thesis_app/shared/components.dart';
```

```

class Statistics extends StatefulWidget {
  final List<Transaction> allTransactions;
  final List<Category> categories;
  final Period currentPeriod;
  final Preferences prefs;

  Statistics({this.allTransactions, this.categories, this.currentPeriod,
    this.prefs,});

  @override
  _StatisticsState createState() => _StatisticsState();
}

class _StatisticsState extends State<Statistics> {
  bool _showAllTimeStats = false;
  bool _showPeriodStats = false;
  bool _showCustomStats = false;
  Widget _body = Center(
```

```

        child: Text(
          'No statistics available. Requires at least one transaction.',
          textAlign: TextAlign.center,
        ),
      ),
    );
    Widget _limitCustomizer;
    List<Transaction> _transactions;
    List<Map<String, dynamic>> _dividedTransactions = [];
    int _daysLeft;
    DateTime _customLimitByDate;
    Map<String, dynamic> _customPeriod;
    bool _showStatistics = true;
    ScrollController _scrollController = ScrollController();

    @override
    Widget build(BuildContext context) {
      if (widget.allTransactions != null &&
        widget.currentPeriod != null &&
        widget.prefs != null &&
        widget.allTransactions.length > 0) {
        _showStatistics = true;
        if (!_showAllTimeStats && !_showPeriodStats && !_showCustomStats) {
          if (widget.prefs.defaultCustomLimitTab == LimitTab.AllTime) {
            _showAllTimeStats = true;
          } else if (widget.prefs.defaultCustomLimitTab == LimitTab.Period) {
            _showPeriodStats = true;
          } else if (widget.prefs.defaultCustomLimitTab == LimitTab.Custom) {
            _showCustomStats = true;
          } else {
            _showPeriodStats = true;
          }
        }
        _dividedTransactions = divideTransactionsIntoPeriods(
          widget.allTransactions, widget.currentPeriod);
        Map<String, dynamic> _currentPeriodTransactions =
          findCurrentPeriod(_dividedTransactions);
        if (_currentPeriodTransactions.containsKey('transactions')) {
          _daysLeft = _currentPeriodTransactions['endDate']
            .difference(DateTime.now())
            .inDays +
            1;
        } else {
          _daysLeft = 0;
        }

        if (_showAllTimeStats) {
          _transactions = widget.allTransactions;
          _limitCustomizer = SizedBox(height: 48.0);
        }

        if (_showPeriodStats) {
          if (_customPeriod != null) {
            _transactions = _customPeriod['transactions'];
            if (_customPeriod['startDate'] ==
              _currentPeriodTransactions['startDate']) {
              _customPeriod = null;
            }
          }
          if (_customPeriod == null) {
            if (_currentPeriodTransactions.containsKey('transactions')) {

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		17

```

        _transactions = _currentPeriodTransactions['transactions'];
    } else {
        _transactions = [];
    }
}
_limitCustomizer = DropdownButton<Map<String, dynamic>>(
    items: _dividedTransactions.map((map) {
        return DropdownMenuItem<Map<String, dynamic>>(
            value: map,
            child: Center(
                child: Text(
                    '${getDateStr(map['startDate'])} -
${getDateStr(map['endDate'])}',
                ),
            ),
        ),
    ).toList(),
    onChanged: (val) {
        setState(() => _customPeriod = val);
    },
    hint: Center(
        child: Text(
            _customPeriod != null
                ? '${getDateStr(_customPeriod['startDate'])} -
${getDateStr(_customPeriod['endDate'])}'
                : '${getDateStr(_currentPeriodTransactions['startDate'])} -
${getDateStr(_currentPeriodTransactions['endDate'])}',
        ),
    ),
    isExpanded: true,
);
}
List<Map<String, dynamic>> _customDividedTransactions;
if (_showCustomStats) {
    if (widget.allTransactions.length > 0 &&
        widget.prefs.isLimitDaysEnabled ||
        widget.prefs.isLimitByDateEnabled) {
        Preferences customPrefs = Preferences(
            pid: widget.prefs.pid,
            limitDays: 1,
            isLimitDaysEnabled: widget.prefs.isLimitDaysEnabled,
            limitPeriods: widget.prefs.limitPeriods,
            isLimitPeriodsEnabled: widget.prefs.isLimitPeriodsEnabled,
            limitByDate: widget.prefs.limitByDate,
            isLimitByDateEnabled: widget.prefs.isLimitByDateEnabled,
            defaultCustomLimitTab: widget.prefs.defaultCustomLimitTab,
            incomeUnfiltered: widget.prefs.incomeUnfiltered,
            expensesUnfiltered: widget.prefs.expensesUnfiltered,
        );
        if (_customLimitByDate != null) {
            customPrefs =
                customPrefs.setPreference('limitByDate', _customLimitByDate);
        }
        _transactions =
            filterTransactionsByLimit(widget.allTransactions, customPrefs);
        _customDividedTransactions = divideTransactionsIntoPeriods(
            _transactions, widget.currentPeriod);
    } else {
        _customDividedTransactions = filterPeriodsWithLimit(
            _dividedTransactions, widget.prefs.limitPeriods);
    }
}

```

```

        _transactions = _customDividedTransactions
            .map<List<Transaction>>((map) => map['transactions'])
            .expand((x) => x)
            .toList();
    }
    DateTime limitFirstDate;
    if (widget.prefs.isLimitByDateEnabled) {
        limitFirstDate = widget.prefs.limitByDate;
    }
    else if (widget.prefs.isLimitDaysEnabled) {
        limitFirstDate = getDateNotTime(DateTime.now().add(Duration(days: 1)))
            .subtract(Duration(days: widget.prefs.limitDays));
    }
    else if (widget.prefs.isLimitPeriodsEnabled) {
        limitFirstDate = findStartDateOfGivenNumPeriodsAgo(
            widget.prefs.limitPeriods, widget.currentPeriod);
    }
    _limitCustomizer = DatePicker(
        context,
        leading: getDateStr(_customLimitByDate ?? limitFirstDate),
        updateDateState: (date) =>
            setState(() => _customLimitByDate = getDateNotTime(date)),
        openDate: limitFirstDate,
    );
    if (_customLimitByDate != null &&
        _customLimitByDate.isAfter(widget.allTransactions.first.date)) {
        _showStatistics = false;
    } else {
        _showStatistics = true;
    }
}
_body = ListView(
    controller: _scrollController,
    padding: bodyPadding,
    children: <Widget>[
        TabSelector(
            context,
            tabs: [
                {
                    'enabled': _showAllTimeStats,
                    'title': 'All-Time',
                    'onPressed': () => setState(() {
                        _showAllTimeStats = true;
                        _showPeriodStats = false;
                        _showCustomStats = false;
                    }),
                },
                {
                    'enabled': _showPeriodStats,
                    'title': 'Period',
                    'onPressed': () => setState(() {
                        _showAllTimeStats = false;
                        _showPeriodStats = true;
                        _showCustomStats = false;
                    }),
                },
                {
                    'enabled': _showCustomStats,
                    'title': 'Daily',
                    'onPressed': () => setState(() {

```

```

        _showAllTimeStats = false;
        _showPeriodStats = false;
        _showCustomStats = true;
    })),
    ],
  ),
  _limitCustomizer,
  if (_showStatistics) ...[
    Balance(
      transactions: _transactions,
      showPeriodStats: _customPeriod == null ? _showPeriodStats : false,
      daysLeft: _daysLeft,
    ),
    SizedBox(height: 20.0),
    Divider(height: 1.0),
    Categories(
      transactions: _transactions,
      categories: widget.categories,
      prefs: widget.prefs,
    ),
    if (_showAllTimeStats || _showCustomStats) ...[
      SizedBox(height: 20.0),
      Periodic(
        dividedTransactions: _showAllTimeStats
          ? _dividedTransactions
          : _customDividedTransactions,
      ),
    ],
    SizedBox(height: 20.0),
    Divider(height: 1.0),
    TopExpenses(
      transactions: _transactions.where((tx) => tx.isExpense).toList(),
      categories: widget.categories,
      scrollController: _scrollController,
    ),
  ] else ...[
    Center(
      child: Text('No transactions available after this date.'),
    ),
  ],
],
);
}

return _body;
}
}

```

```

import 'package:flutter/material.dart';
import 'package:thesis_app/models/transaction.dart';
import 'package:thesis_app/pages/statistics/barTile.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/components.dart';

```

```

class Balance extends StatelessWidget {
  final List<Transaction> transactions;
  final bool showPeriodStats;
  final int daysLeft;

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		20

```

Balance({this.transactions, this.showPeriodStats, this.daysLeft});

@override
Widget build(BuildContext context) {
  final Map<String, double> balancesList = {
    'income': filterAndGetTotalAmounts(
      transactions,
      filterOnlyExpenses: false,
    ),
    'expenses': filterAndGetTotalAmounts(
      transactions,
      filterOnlyExpenses: true,
    ),
  };
  final double balance = balancesList['income'] - balancesList['expenses'];
  final Map<String, dynamic> relativePercentages =
    getRelativePercentages(balancesList);
  final String remainingPerDay = balance <= 0
    ? 'No remaining balance'
    : '${(balance / daysLeft).toStringAsFixed(2)} / day';
  return Column(
    children: <Widget>[
      StatTitle(title: 'Balance'),
      Center(
        child: Text(getAmountStr(balance), style: TextStyle(fontSize: 25.0)),
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: <Widget>[
          Text('$daysLeft day(s) until next period'),
          Text(remainingPerDay),
        ],
      ),
      SizedBox(height: 10.0),
      BarTile(
        title: 'Income',
        amount: balancesList['income'],
        percentage: relativePercentages['income'],
        color: Colors.green[800],
      ),
      BarTile(
        title: 'Expenses',
        amount: balancesList['expenses'],
        percentage: relativePercentages['expenses'],
        color: Colors.red[800],
      ),
    ],
  );
}

String getPrevStr(double current, double previous) {
  double percentage = 100 * current / previous;
  if (percentage.isNaN) {
    return 'No available data this period';
  }
  String percentageStr = percentage < 0
    ? percentage.toStringAsFixed(2)
    : '+${percentage.toStringAsFixed(2)}';
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		21

```

    return '$percentageStr% (vs prev. ${getAmountStr(previous)})';
}

import 'package:flutter/material.dart';

class BarTile extends StatelessWidget {
  final String title;
  final String subtitle;
  final double amount;
  final double percentage;
  final Color color;

  BarTile({this.title, this.subtitle, this.amount, this.percentage, this.color,});

  @override
  Widget build(BuildContext context) {
    return Column(
      children: <Widget>[
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Text(
              title,
              style: TextStyle(
                fontSize: 16.0,
                fontWeight: FontWeight.bold,
              ),
            ),
            Text('${amount.toStringAsFixed(2)}'),
          ],
        ),
        subtitle != null
          ? Row(
              mainAxisAlignment: MainAxisAlignment.start,
              children: <Widget>[
                Text(subtitle),
              ],
            )
          : Container(),
        percentage != null
          ? Row(
              mainAxisAlignment: MainAxisAlignment.start,
              children: <Widget>[
                Container(
                  height: 25.0,
                  width:
                    percentage * (MediaQuery.of(context).size.width - 20),
                  decoration: BoxDecoration(
                    color: color,
                    borderRadius: new BorderRadius.all(Radius.circular(5.0)),
                  ),
                ),
              ],
            )
          : Container(),
      ],
    );
  }
}

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ документа	Підпис	Дата		22

```

import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import 'package:thesis_app/models/category.dart';
import 'package:thesis_app/models/preferences.dart';
import 'package:thesis_app/models/transaction.dart';
import 'package:thesis_app/pages/statistics/indicator.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/components.dart';

class Categories extends StatefulWidget {
  final List<Transaction> transactions;
  final List<Category> categories;
  final Preferences prefs;

  Categories({this.transactions, this.categories, this.prefs});

  @override
  _CategoriesState createState() => _CategoriesState();
}

class _CategoriesState extends State<Categories> {
  int touchedIndex = -1;
  bool onlyExpenses;

  @override
  Widget build(BuildContext context) {
    List<Map<String, dynamic>> _categoricalData;
    List<PieChartSectionData> sectionData = [];
    onlyExpenses = onlyExpenses ?? widget.prefs.isOnlyExpenses;
    final List<Transaction> txExpenses =
      widget.transactions.where((tx) => tx.isExpense).toList();
    if (widget.transactions.length > 0) {
      final List<Map<String, dynamic>> _transactionsInCategories = onlyExpenses
        ? divideTransactionsIntoCategories(txExpenses, widget.categories)
        : divideTransactionsIntoCategories(
            widget.transactions, widget.categories);
      final List<Map<String, dynamic>> _categoriesWithTotalAmounts =
        appendTotalCategoricalAmounts(_transactionsInCategories);
      final List<Map<String, dynamic>> _categoriesWithPercentages =
        appendIndividualPercentages(_categoriesWithTotalAmounts);
      _categoricalData = combineSmallPercentages(_categoriesWithPercentages);
      _categoricalData
        .sort((a, b) => b['percentage'].compareTo(a['percentage']));
      sectionData = _categoricalData
        .asMap()
        .map((index, category) {
          return MapEntry(
            index,
            PieChartSectionData(
              value: category['percentage'] * 100,
              color: category['iconColor'],
              radius: touchedIndex == index ? 145 : 140,
              title: category['percentage'] > 0.04
                ? String.fromCharCode(category['icon'])
                : '',
              titleStyle: TextStyle(
                color: Colors.white,
                fontSize: 25.0,
                fontFamily: 'MaterialDesignIconFont',
                package: 'community_material_icon',

```

```

        ),
        titlePositionPercentageOffset: 0.7,
      ),
    );
  })
  .values
  .toList();
}
return Column(children: <Widget>[
  StatTitle(title: 'Categories'),
  if (sectionData.length > 0) ...[
    SizedBox(height: 10.0),
    SizedBox(height: 35.0),
    PieChart(
      PieChartData(
        sections: sectionData,
        sectionsSpace: 1,
        borderData: FlBorderData(
          show: false,
        ),
        pieTouchData: PieTouchData(
          touchCallback: (pieTouchResponse) => setState(() {
            touchedIndex = (pieTouchResponse.touchInput is FlLongPressEnd ||
              pieTouchResponse.touchInput is FlPanEnd ||
              pieTouchResponse.touchedSectionIndex == null)
              ? touchedIndex
              : pieTouchResponse.touchedSectionIndex;
          })),
        ),
      ),
    ),
    SizedBox(height: 20.0),
    Column(
      mainAxisAlignment: MainAxisAlignment.max,
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: _categoricalData
        .asMap()
        .map(
          (index, category) => MapEntry(
            index,
            Indicator(
              color: category['iconColor'],
              text:
                '${category['name']} -
                ${category['amount'].toStringAsFixed(2)} (${category['percentage'] *
                100).toStringAsFixed(0)}%',
              isSquare: false,
              size: touchedIndex == index ? 18 : 16,
              textColor:
                touchedIndex == index ? Colors.black : Colors.grey,
              onTap: () => setState(() => touchedIndex = index),
            ),
          ),
        ),
      .values
      .toList(),
    ),
  ] else ...[
    SizedBox(height: 35.0),
    Center(

```

```

        child: Text(
          'No ${onlyExpenses ? 'expenses' : 'negative balance'} in current
period.'),
      ),
    ],
  ));
}
}

```

```

import 'package:flutter/material.dart';
import 'package:thesis_app/models/category.dart';
import 'package:thesis_app/models/transaction.dart';
import 'package:thesis_app/pages/statistics/barTile.dart';
import 'package:thesis_app/shared/library.dart';
import 'package:thesis_app/shared/components.dart';

class TopExpenses extends StatefulWidget {
  final List<Transaction> transactions;
  final List<Category> categories;
  final ScrollController scrollController;

  TopExpenses({this.transactions, this.categories, this.scrollController,});

  @override
  _TopExpensesState createState() => _TopExpensesState();
}

```

```

class _TopExpensesState extends State<TopExpenses> {
  int _showCount = 5;
  List<Map<String, dynamic>> _sortedTransactions;
  List<Widget> _columnContent;

  @override
  Widget build(BuildContext context) {
    if (widget.transactions.length > 0) {
      _sortedTransactions = sortByAmountDescending(widget.transactions)
        .map((tx) => {
          'payee': tx.payee,
          'cid': tx.cid,
          'amount': tx.amount,
        })
        .toList();
      _columnContent = <Widget>[
        ...sublist(_sortedTransactions, 0, _showCount)
          .map((tx) => <Widget>[
            SizedBox(height: 10.0),
            BarTile(
              title: tx['payee'],
              subtitle: getCategory(widget.categories, tx['cid']).name,
              amount: tx['amount'],
            ),
          ])
          .expand((x) => x)
          .toList(),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: <Widget>[
            FlatButton(
              child: Text('Show less'),
              onPressed: () => setState(() => _showCount = 5),
            ),
          ],
        ),
      ];
    }
  }
}

```

```

    ),
    FlatButton(
      child: Text('Show more'),
      onPressed: () {
        setState(() => _showCount += 5);
        widget.scrollController.animateTo(
          99999,
          duration: Duration(seconds: 3),
          curve: Curves.easeInOutQuint,
        );
      },
    ),
  ],
)
];
} else {
  _columnContent = <Widget>[
    SizedBox(height: 35.0),
    Center(
      child: Text('No expenses found in current period.'),
    ),
  ];
}

return Column(
  children: <Widget>[
    StatTitle(title: 'Top Expenses'),
    ..._columnContent,
  ],
);
}
}

List<Transaction> sortByAmountDescending(List<Transaction> transactions) {
  transactions.sort((a, b) => b.amount.compareTo(a.amount));
  return transactions;
}

```