

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Мобільний застосунок для персонального планування
бюджету та часу»**

Виконав студент групи ПЗ-20-1
КРАВЧУК Володимир Олександрович

Керівник роботи:
МІНТІЙ Ірина Сергіївна

Консультант:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ФАНТ Микола Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2024р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КРАВЧУК Володимир Олександрович**

Керівник роботи: **к.п.н., доц., доц. каф. ІІЗ МІНТІЙ Ірина Сергіївна**

Тема роботи: **«Мобільний застосунок для персонального планування бюджету та часу»**, затверджена Наказом закладу вищої освіти від **«26» лютого 2024 р., №74/с**

Вихідні дані для роботи: розробка мобільного додатку для персонального планування бюджету та часу, який використовується для відслідковування витраченого користувачами часу та коштів за відповідними категоріями.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	02.03.2024	02.03.2024
2	Вакалюк Т.А.	04.04.2024	04.04.2024
3	Вакалюк Т.А.	02.05.2024	02.05.2024

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з мобільного застосунку для персонального планування бюджету та часу і пояснювальної записки. Пояснювальна записка до випускної роботи містить 84 сторінки, 37 ілюстрацій та 23 таблиці.

Дипломний проєкт присвячений мобільному додатку для персонального планування бюджету та часу.

Метою дипломного проєкту є підвищення ефективності в роботі і персональному житті на основі оптимізації часу та коштів витрачених на задачі.

Об'єктом дослідження є програмне забезпечення для персонального планування бюджету та часу.

Предметом дослідження є підвищення ефективності в роботі і персональному житті на основі оптимізації часу та коштів витраченого на завдання.

В першому розділі роботи було розглянуто актуальність теми розробки мобільного додатку для персонального планування бюджету та часу.

В другому розділі було розглянуто проєктування мобільного застосунку були проаналізовані усі варіанти використання програмного продукту відповідно до вимог програмного забезпечення. На основі отриманого результату була спроектована структура платформи.

В третьому розділі була наведена детальна схема розгортання платформи, було надано опис інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку. Разом з цим, було описано процес мануального тестування, де програмне забезпечення піддавалося випробуванням без використання автоматизованих інструментів.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, ANDROID, БАЗА ДАНИХ, СТАТИСТИКА.

					ІПЗ.КР.Б-121-24-ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кравчук В.О.			Мобільний застосунок для персонального планування бюджету та часу	Літ.	Арк.	Аркушів
Керівник		Мінтій І.С.					3	84
						Житомирська політехніка, група ІПЗ-20-1		
Н. контр.								
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The explanatory note of the diploma project consists of three sections and includes 23 tables, 37 figures, totaling 84 pages.

The diploma project is dedicated to a mobile application for personal budget and time planning.

The aim of the diploma project is to increase efficiency in work and personal life through the optimization of time and expenses spent on tasks.

The object of the research is software for personal budget and time planning.

The subject of the research is to improve efficiency in work and personal life based on the optimization of time and expenses spent on tasks.

The first section of the work discusses the relevance of developing a mobile application for personal budget and time planning.

The second section covers the design of the mobile application, analyzing all possible uses of the software product according to software requirements. Based on the results, the structure of the platform was designed.

The third section provides a detailed deployment scheme of the platform and describes the interface of the developed software product. All functional capabilities of the application were examined. Additionally, the process of manual testing was described, where the software underwent testing without using automated tools.

KEYWORDS: MOBILE APPLICATION, ANDROID, DATABASE, STATISTICS.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	18
Висновки до першого розділу.	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	21
2.2 Вимоги користувачів:	24
2.3 Функціональні вимоги:	25
2.4 Не функціональні вимоги:	25
2.5 Системні вимоги:	26
2.6 Розробка бази даних системи.....	26
2.7 Проектування та реалізація алгоритмів роботи системи	29
2.8 Реалізація функціоналу мобільного застосунку для планування бюджету та часу.....	30
Висновки до другого розділу.....	38
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З МОБІЛЬНИМ ДОДАТКОМ	39
3.1 Порядок встановлення та налаштування параметрів платформи	39
3.2 Структура інтерфейсу мобільного додатку	40
3.3 Загальна статистика мобільного додатку.....	50
3.4 Тестування платформи.....	51
Висновки до третього розділу	58
ВИСНОВКИ	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТКИ.....	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

CRUD – основні операції, що використовуються для управління даними (Create, Read, Update, Delete)

HTML – HyperText Markup Language

JS – JavaScript

TS – TypeScript

SDK – набір для розробки програмного забезпечення

БД – база даних

ПЗ – програмне забезпечення

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми обумовлена зростаючою цифровізацією, яка робить смартфони основним інструментом для вирішення повсякденних завдань, включаючи фінанси та управління часом.

У сучасних умовах нестабільної економіки та зростання фінансових обов'язків здатність ефективно керувати своїми фінансами стає все більш важливою, а мобільні додатки допомагають відстежувати доходи та витрати, складати бюджети та планувати фінансові цілі. Ефективне управління часом за допомогою мобільних додатків дозволяє підвищити продуктивність, знизити рівень стресу та досягати особистих і професійних цілей.

Зростаючий попит на самовдосконалення також підкреслює значимість таких додатків, оскільки вони допомагають користувачам досягати фінансових і часових цілей, підвищуючи загальну якість життя.

Метою випускної кваліфікаційної роботи бакалавра є розробка мобільного застосунку для персонального планування бюджету та часу, забезпечення користувачів ефективним інструментом для управління фінансами і часом.

Для реалізації поставленої мети потрібно виконати наступні **завдання**:

1. Проведення аналізу вимог до мобільного застосунку та оцінка сучасних підходів до розробки мобільних додатків.
2. Вивчити архітектуру, інструменти та особливості розробки застосунку.
3. Обґрунтування та вибір засобів реалізації системи, таких як JavaScript, React Native, Expo go, Firebase.
4. Реалізація програмного комплексу, зокрема розробка мобільного додатку.
5. Протестування та оптимізація системи для забезпечення високої продуктивності та масштабованості.

Об'єктом дослідження в даному випадку є мова програмування JavaScript та її розширення TypeScript. JavaScript [1] є однією з найпоширеніших мов програмування для розробки веб-додатків, зокрема фронтенд-частини.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

React Native є ще одним важливим розширенням JavaScript. Фреймворк дозволяє створювати мобільні додатки для iOS та Android з використанням лише однієї кодової бази. React Native [2] забезпечує високу продуктивність додатків, дозволяючи використовувати нативні компоненти платформи, а також скорочує час розробки та полегшує підтримку коду. Використання TypeScript [3] з React Native дає додаткові переваги у вигляді покращеної типізації та інструментів для виявлення помилок, що робить процес розробки ще більш ефективним і надійним.

Expo Go — це мобільний застосунок, який дозволяє розробникам швидко запускати та тестувати свої React Native проекти без необхідності збирати додаток нативно для iOS або Android [4]. Використовуючи Expo Go, розробники можуть миттєво переглядаючи зміни в реальному часі на своїх пристроях, що значно прискорює процес розробки та тестування.

Firebase, платформа розробки мобільних і веб-додатків від Google, також відіграє значну роль у сучасному веб-розробництві [5]. Вона пропонує широкий спектр інструментів та сервісів, таких як реальна база даних, аутентифікація, хостинг, функції безпеки, аналітика тощо. Використання Firebase з React Native дозволяє легко додавати потужні серверні функціональності без необхідності керувати власними серверами. Поєднання JavaScript, React Native та Firebase створює потужну екосистему для розробки сучасних, надійних і масштабованих мобільних застосунків.

Предметом дослідження є ефективність та взаємодія JavaScript, Expo go, React Native та Firebase у розробці сучасних мобільних застосунків. Також досліджуються сервіси та інструменти Firebase, включаючи аутентифікацію, бази даних в реальному часі, хостинг та аналітику, і аналізується взаємодія між цими технологіями, їх інтеграція та вплив на ефективність розробки, масштабованість та продуктивність додатків.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ДОДАТКУ

1.1 Постановка задачі

Робота присвячена створенню мобільного застосунку, який має на меті полегшити та оптимізувати процес планування персонального бюджету та часу для користувачів.

Сучасні інформаційні технології швидко розвиваються, і з кожним роком з'являється все більше додатків, які можуть покращити різні аспекти нашого життя. Постійно розробляються нові програмні продукти, що охоплюють все більше областей людської діяльності. Перші програмні продукти були призначені для автоматизації системи з метою підвищення ефективності певних завдань. З часом розробка програмного забезпечення стала не лише необхідністю для великих компаній та наукових установ, а й хобі для ентузіастів. Програмісти починають оптимізувати та автоматизувати процеси у своєму середовищі існування. Як результат, існують програми, орієнтовані на підвищення продуктивності людей, включаючи ті, які допомагають відстежувати витрачений час та фінанси. Застосунки для персонального планування бюджету та часу корисні з різних причин:

- Користувачеві не потрібно запам'ятовувати всі заплановані завдання.
- Користувачі можуть легко переглядати історію своїх записів.
- Користувачі можуть взаємодіяти з додатком у будь-який час на мобільному пристрої.
- Перенесення обліку часу і бюджету з паперової версії на електронну дозволяє користувачам легко редагувати вже створені записи.

Такі додатки не повинні бути вимогливими до системи, щоб максимізувати охоплення потенційних користувачів. Детальний опис вимог застосунку, його функціональності, безпеки даних і формування вимог до простоти використання наведено в наступних розділах дипломної роботи.

Мобільні застосунки для персонального планування бюджету та часу виникли внаслідок поєднання двох ключових факторів: зростання

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

використання мобільних пристроїв і популярності концепцій управління часом та фінансами. Початкові версії таких додатків з'явилися в середині 2000-х років, коли мобільні телефони стали більш поширеними та функціональними.

У цей період в основному використовувалися прості додатки, які дозволяли користувачам вести список своїх витрат і розподілювати їх за категоріями. Однак із зростанням потужності мобільних пристроїв та розвитком інтернет-технологій, з'явилися більш функціональні додатки, які надавали можливості не лише ведення бюджету, а й планування фінансових цілей, отримання статистики витрат, інтеграцію з банківськими рахунками тощо.

У той же час, поняття управління часом стало все більш актуальним у сучасному світі, де швидкість життя зростає, а потреба у ефективному використанні часу стає важливішою. Мобільні додатки для планування часу дозволяють користувачам створювати розклади, відстежувати виконання завдань, отримувати нагадування про події тощо.

Сьогодні мобільні додатки [6] для персонального планування бюджету та часу стали не лише зручними інструментами для контролю фінансів та організації часу, а й важливими складовими елементами особистого розвитку та продуктивності. Вони постійно розвиваються і вдосконалюються, відповідаючи на зростаючі потреби сучасних користувачів.

Сформульовані основні вимоги до програмного продукту. Однією з основних вимог є доступність. Користувач має мати можливість швидкого доступу до додатку, та програмний продукт повинен бути простим у користуванні для різних категорій користувачів [7].

1.2 Аналіз аналогів програмного продукту

На момент написання цієї роботи було знайдено багато додатків, які поєднують планування бюджету та часу. Проте ці додатки часто є платними. Існують також програми, орієнтовані на управління проектами, які призначені для корпоративного використання, де планування витрат і часу здійснюється в

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

рамках проектів, а не для особистого застосування.

Отже, порівняння буде проводитися з найпопулярнішими додатками у відповідних категоріях: для планування часу та для управління бюджетом.

Monefy – це популярний мобільний додаток для управління особистими фінансами, який дозволяє користувачам легко відстежувати свої витрати і доходи. Інтерфейс програми інтуїтивно зрозумілий, що робить його доступним для широкого кола користувачів.

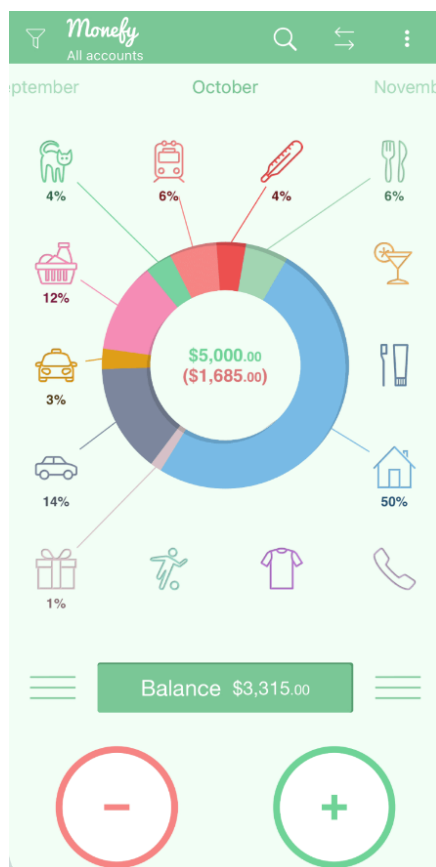


Рис. 1.1. Мобільний додаток Monefy [8]

Плюси:

- Інтуїтивний інтерфейс: Monefy має простий і зрозумілий інтерфейс, що дозволяє швидко додавати нові транзакції і переглядати фінансові дані.

- Візуалізація витрат: Програма пропонує діаграми та графіки, які допомагають користувачам бачити розподіл витрат по категоріях, що полегшує аналіз фінансових звичок.

- Підтримка різних валют: Monefy підтримує безліч валют, що є корисним для користувачів, які здійснюють транзакції в різних країнах.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

- Синхронізація з іншими пристроями: Програма дозволяє синхронізувати дані між кількома пристроями через Google Drive або Dropbox, що забезпечує зручний доступ до фінансової інформації з будь-якого місця.

- Гнучкі налаштування: Користувачі можуть налаштовувати категорії витрат, звітні періоди та інші параметри відповідно до своїх потреб.

- Безпека даних: Monefy забезпечує безпеку даних за допомогою пароля та шифрування даних.

Недоліки:

- Обмежені функції у безкоштовній версії: Безкоштовна версія Monefy має обмежений функціонал, що може спонукати користувачів придбати преміум-версію для отримання повного набору функцій.

- Відсутність інтеграції з банками: На відміну від деяких інших фінансових додатків, Monefy не підтримує автоматичну інтеграцію з банківськими рахунками, що означає необхідність вручну вводити всі транзакції.

- Обмежена категоризація: Хоча користувачі можуть створювати власні категорії, кількість можливих категорій все ще обмежена, що може бути незручним для тих, хто хоче більш детально структурувати свої витрати.

Відсутність функцій для інвестицій: Monefy орієнтований в основному на управління витратами та доходами і не пропонує функцій для відстеження інвестицій або планування заощаджень.

Money Lover – це потужний мобільний додаток для управління особистими фінансами, який допомагає користувачам відстежувати доходи, витрати та бюджет. Програма пропонує широкий набір інструментів для фінансового планування та аналізу.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

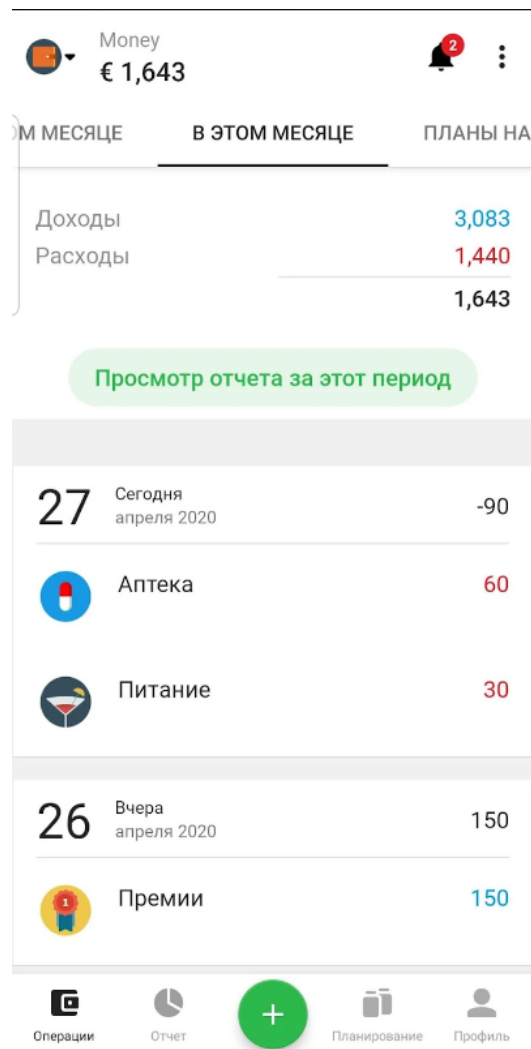


Рис. 1.2. Мобільний додаток Money Lover [9]

Плюси:

- Широкий функціонал: Money Lover пропонує безліч функцій, включаючи відстеження доходів і витрат, складання бюджету, управління заощадженнями та боргами, а також планування фінансових цілей.
- Інтеграція з банками: Програма підтримує автоматичну інтеграцію з банківськими рахунками, що дозволяє автоматично імпортувати транзакції та знижує необхідність вручну вводити дані.
- Зручний інтерфейс: Інтерфейс Money Lover є інтуїтивно зрозумілим і легко налаштовуваним, що робить його зручним для користувачів різного рівня досвіду.
- Аналітика та звіти: Додаток надає детальні звіти та аналітику, які допомагають користувачам аналізувати свої фінансові звички та приймати

обґрунтовані рішення.

- Підтримка мультивалютності: Money Lover підтримує роботу з різними валютами, що є корисним для мандрівників та тих, хто має доходи або витрати в іноземній валюті.

- Хмарна синхронізація: Програма дозволяє синхронізувати дані через хмарні сервіси, такі як Google Drive та Dropbox, що забезпечує доступ до фінансової інформації з будь-якого пристрою.

- Нагадування та планування: Money Lover дозволяє встановлювати нагадування про платежі та інші фінансові події, що допомагає уникнути прострочень.

Недоліки:

- Обмеження безкоштовної версії: Як і багато інших фінансових додатків, Money Lover у безкоштовній версії має обмежений функціонал, що може спонукати користувачів перейти на платну версію для отримання доступу до всіх можливостей.

- Інколи проблеми з синхронізацією: Деякі користувачі повідомляють про проблеми з синхронізацією даних між різними пристроями, що може бути незручним.

- Рекламні оголошення: У безкоштовній версії програми можуть з'являтися рекламні оголошення, що може відволікати та дратувати користувачів.

- Складність налаштування: Через широкий функціонал нові користувачі можуть відчувати певну складність у початковому налаштуванні додатка та освоєнні всіх його можливостей.

Asana – це потужна платформа для управління проєктами та завданнями, яка допомагає командам організовувати робочі процеси, відстежувати прогрес і співпрацювати ефективніше. Asana пропонує інструменти для планування, організації та моніторингу завдань, що робить її корисною для компаній будь-якого розміру.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

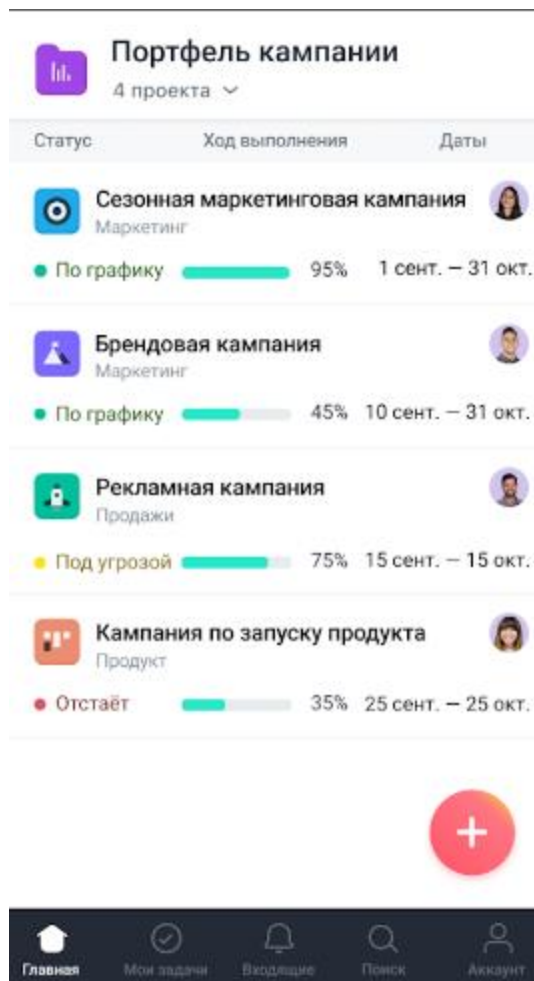


Рис. 1.3. Мобільний додаток Asana [10]

Плюси

- Інтуїтивний інтерфейс: Asana має зручний і зрозумілий інтерфейс, який легко освоїти навіть новачкам. Завдання та проєкти організовані у вигляді дощок, списків або календарів, що дозволяє вибрати найбільш зручний спосіб роботи.
- Гнучкі можливості налаштування: Користувачі можуть налаштовувати проєкти та завдання під свої потреби, включаючи встановлення пріоритетів, дедлайнів, етапів та відповідальних осіб.
- Сильна співпраця та комунікація: Asana дозволяє легко спілкуватися з членами команди через коментарі, обговорення завдань та проєктів, а також через інтеграцію з іншими інструментами, такими як Slack і Microsoft Teams.
- Інтеграції з іншими сервісами: Asana інтегрується з багатьма популярними сервісами, такими як Google Drive, Dropbox, Zoom, Slack, Microsoft Office 365, що дозволяє підвищити ефективність роботи.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

- Докладні звіти та аналітика: Програма пропонує інструменти для створення звітів та відстеження прогресу, що допомагає командам оцінювати свою продуктивність і вносити необхідні корективи.

- Безкоштовна версія: Asana пропонує безкоштовну версію з основними функціями, що дозволяє малим командам і окремим користувачам випробувати програму без фінансових витрат.

Недоліки

- Обмеження безкоштовної версії: Безкоштовна версія Asana має обмежений набір функцій і підходить в основному для невеликих команд. Доступ до більш просунутих можливостей потребує переходу на платні плани.

- Перевантаженість функціоналом: Через широкий спектр функцій нові користувачі можуть відчувати складність у освоєнні всіх можливостей програми, що потребує додаткового часу для навчання.

- Обмежена функціональність для управління ресурсами: Asana добре підходить для управління завданнями, але може не вистачати деяких інструментів для управління ресурсами, таких як детальний облік часу та ресурсів.

- Залежність від інтернету: Як хмарне рішення, Asana вимагає постійного доступу до інтернету для роботи, що може бути незручно в умовах поганого інтернет-з'єднання.

Clockify – це безкоштовний інструмент для відстеження робочого часу, який дозволяє користувачам реєструвати та аналізувати свій робочий час, організовувати завдання і проєкти, а також генерувати звіти. Clockify популярний серед фрілансерів, малих та середніх підприємств завдяки своїй простоті та широкому функціоналу.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

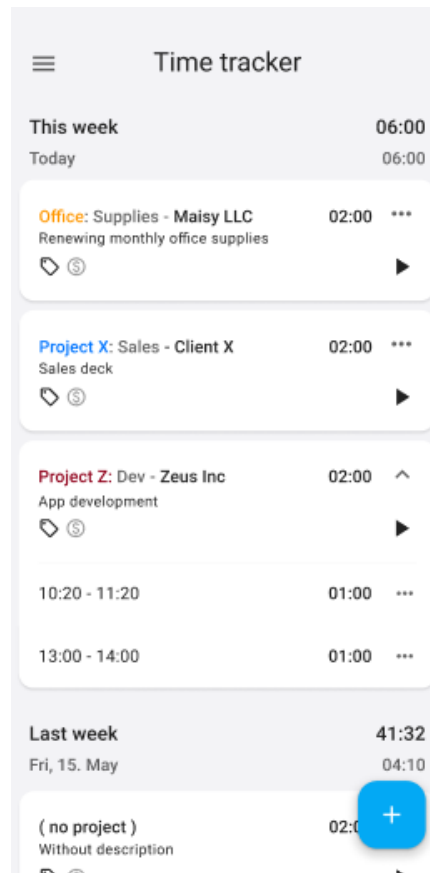


Рис. 1.4. Мобільний додаток Clockify [11]

Плюси

-Безкоштовність: Clockify є безкоштовним інструментом з багатим набором функцій, що робить його привабливим для широкого кола користувачів.

-Легкість використання: Інтерфейс програми інтуїтивно зрозумілий і легко освоюваний, що дозволяє швидко почати роботу без тривалого навчання.

-Мультиплатформність: Clockify доступний на різних платформах, включаючи веб-версію, мобільні додатки для iOS і Android, а також десктопні додатки для Windows, Mac і Linux.

-Гнучкість у налаштуваннях: Користувачі можуть налаштовувати проєкти, завдання та теги відповідно до своїх потреб, що дозволяє точно відстежувати час, витрачений на різні види діяльності.

-Звіти та аналітика: Clockify пропонує детальні звіти, які можна експортувати в різних форматах (PDF, CSV, Excel), що дозволяє аналізувати продуктивність і ефективно планувати робочий час.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

-Інтеграції: Clockify інтегрується з багатьма популярними інструментами для управління проєктами та продуктивності, такими як Asana, Trello, Jira, Slack та інші, що дозволяє легко включити його у вже існуючі робочі процеси.

Недоліки

-Обмеженість безкоштовної версії для великих команд: Хоча безкоштовна версія має багато корисних функцій, великі команди або організації можуть потребувати платної підписки для доступу до розширених можливостей, таких як адміністрування користувачів, контроль доступу та інші корпоративні функції.

-Ручний ввід даних: Деякі користувачі можуть вважати незручним необхідність вручну вводити час для кожного завдання, що може призвести до неточностей, якщо користувач забуде це зробити вчасно.

-Обмежена аналітика: Хоча Clockify пропонує базові звіти, для глибшого аналізу та детальнішої аналітики може знадобитися інтеграція з іншими інструментами або використання платних планів.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки сучасного, кросплатформенного та продуктивного мобільного додатку було обрано кілька потужних інструментальних засобів: JavaScript, TypeScript, React Native, Expo go та Firebase. Кожен з цих засобів має свої унікальні переваги, які роблять їх оптимальними для реалізації даного проєкту.

JavaScript обрано для розробки фронтенду додатку через його популярність [12], широке використання в веб-розробці та потужну екосистему бібліотек і фреймворків. Його здатність працювати безпосередньо в браузері користувача забезпечує швидку та інтерактивну взаємодію з інтерфейсом. JavaScript також підтримує розробку кросплатформених додатків через фреймворки, такі як React Native, що дозволяє створювати мобільні додатки для різних операційних систем з використанням одного коду.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

TypeScript, як надбудова над JavaScript, додає статичну типізацію до мови [13], що значно покращує процес розробки за рахунок виявлення помилок на етапі компіляції. Це особливо корисно для великих проєктів, де підтримка коду та його розширюваність є критичними. Використання TypeScript сприяє більшій структурованості коду, полегшує налагодження та поліпшує співпрацю в команді.

Firebase обрано для управління базою даних, аутентифікацією користувачів, зберіганням файлів та хостингом. Ця платформа надає широкий спектр сервісів, які значно спрощують розробку та обслуговування додатку [14]. Реальне часу синхронізація даних, що пропонує Firebase, є особливо корисною для додатків, де користувачі потребують миттєвого оновлення інформації. Крім того, Firebase забезпечує високу безпеку даних та масштабованість, що дозволяє легко адаптувати додаток під зростаючу кількість користувачів.

React Native дозволяє розробляти мобільні додатки для обох основних платформ - iOS та Android, використовуючи велику частину одного коду. Це дозволяє зменшити час розробки та зусилля, що потрібні для підтримки обох платформ [15]. React Native дозволяє використовувати нативний код для реалізації специфічних функцій для кожної платформи, які не підтримуються стандартним фреймворком. Крім того, велика кількість сторонніх бібліотек та плагінів дозволяє розширити можливості фреймворку.

Ехро Go надає швидкий спосіб розробки мобільних додатків без необхідності налаштовувати та налагоджувати середовище розробки [16]. Всі необхідні інструменти (наприклад, засоби для збірки, оновлення коду в реальному часі тощо) вже вбудовані в Ехро, що дозволяє зосередитися на функціональності додатка.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Висновки до першого розділу.

У першому розділі роботи було розглянуто актуальність теми розробки мобільного додатку для персонального планування бюджету та часу. Розглянуто еволюцію подібних додатків з початку їх появи до сучасних рішень. Були висвітлені переваги та причини популярності мобільних додатків для планування фінансів та часу.

Проведений аналіз показав, що в сучасному світі додатки для планування бюджету та часу стали необхідними інструментами для керування фінансами та оптимізації часу. Вони дозволяють користувачам ефективніше використовувати свої ресурси та досягати поставлених цілей.

На основі проведеного аналізу було сформульовано основні вимоги до програмного продукту для персонального планування бюджету та часу, які стануть основою подальшої розробки додатку.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Мобільний застосунок для персонального планування бюджету та часу надає користувачам можливість авторизуватися в системі, змінювати дані свого акаунту, переглядати графіки на основі часових та грошових витрат, виходити з раніше авторизованого акаунту, створювати, редагувати, видаляти відмітки та категорії для них, що сприяє ефективному плануванню бюджету та управлінню часом. Для наглядної взаємодії між елементами застосунку було створено діаграму варіантів використання (рис 2.1).

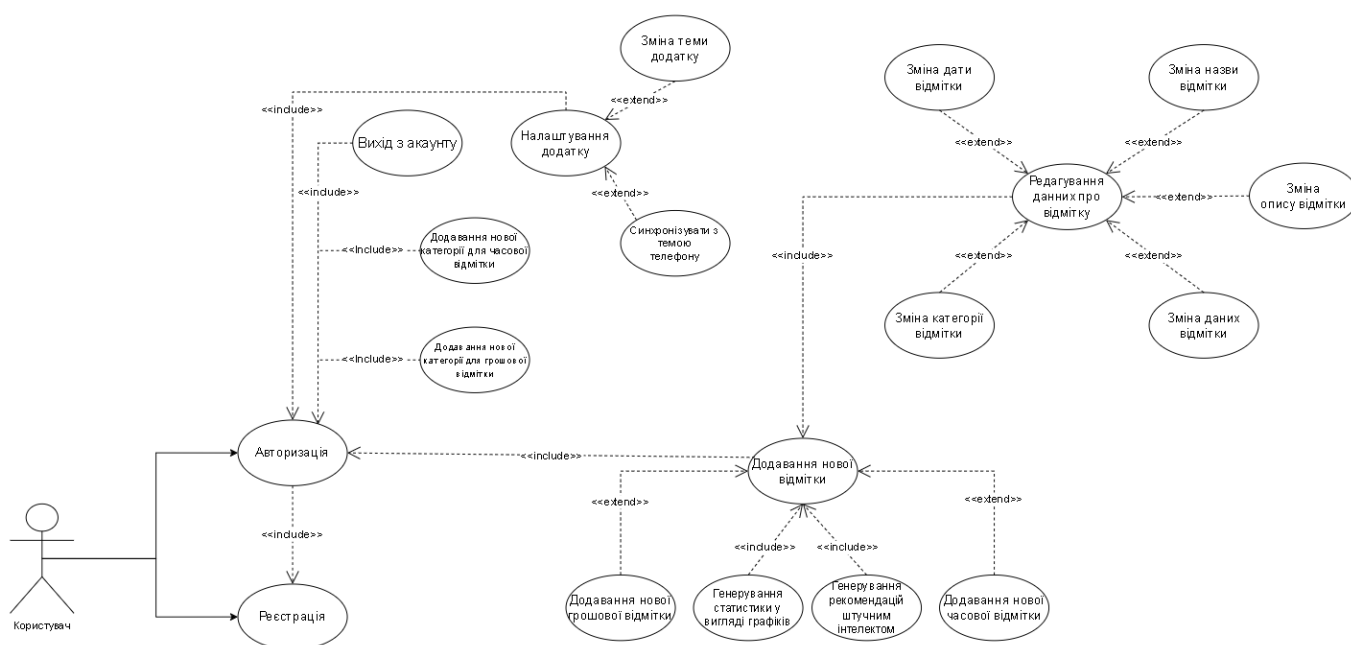


Рис. 2.1 Діаграма варіантів використання

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.11

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»	
Назва варіанту використання	Вхід/реєстрація
Основні актори	Користувач
Короткий опис	Користувачі можуть увійти в застосунок за допомогою гугл чи анонімно

Таблиця 2.2

Опис прецеденту «Додавання нової часової відмітки»	
Назва варіанту використання	Додавання нової часової відмітки
Основні актори	Користувач
Короткий опис	Користувачі можуть додати нову часову відмітку

Таблиця 2.3

Опис прецеденту «Додавання нової грошової відмітки»	
Назва варіанту використання	Додавання нової грошової відмітки
Основні актори	Користувач
Короткий опис	Користувачі можуть додати нову грошову відмітку

Таблиця 2.4

Опис прецеденту «Налаштування додатку»	
Назва варіанту використання	Зміна теми
Основні актори	Користувач
Короткий опис	Користувачі можуть змінити тему додатку (світла/темна) або синхронізувати тему додатку з темою телефону

Таблиця 2.5

Опис прецеденту «Генерування статистики»	
Назва варіанту використання	Аналіз та планування
Основні актори	Користувач
Короткий опис	Користувачі можуть ознайомитись з згенерованою статистикою у вигляді діаграм та планувати свій бюджет/час відповідно до даних

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Таблиця 2.6

Опис прецеденту «Вихід»	
Назва варіанту використання	Вихід з акаунту користувача
Основні актори	Користувач
Короткий опис	Користувач натискає кнопку виходу з акаунту, після чого відбувається його перенаправлення на сторінку авторизації у систему

Таблиця 2.7

Опис прецеденту «Редагування часової відмітки»	
Назва варіанту використання	Редагування часової відмітки
Основні актори	Користувач
Короткий опис	Користувач має можливість редагувати дані раніше створеної відмітки про витрату часу, включаючи назву, категорію, дату початку і кінця, що дозволяє внести потрібні зміни у вже існуючі записи.

Таблиця 2.8

Опис прецеденту «Редагування грошової відмітки»	
Назва варіанту використання	Редагування грошової відмітки
Основні актори	Користувач
Короткий опис	Користувач має можливість редагувати дані раніше створеної відмітки про витрату коштів.

Таблиця 2.9

Опис прецеденту «Створення категорії про грошову витрату»	
Назва варіанту використання	Створення нової категорії
Основні актори	Користувач
Короткий опис	Користувач може створити нову категорію для відміток про грошові витрати, надаючи їй назву та, за необхідності, додатковий опис.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Таблиця 2.10

Опис прецеденту «Видалення категорії про грошову витрату»	
Назва варіанту використання	Видалення категорії
Основні актори	Користувач
Короткий опис	Користувач може видалити категорію для відміток про грошові витрати.

Таблиця 2.11

Опис прецеденту «Генерування за допомогою AI»	
Назва варіанту використання	Генерування нової відмітки за допомогою AI
Основні актори	Користувач
Короткий опис	Користувач отримує може додати нову відмітку за допомогою скріншоту на основі AI.

Основною ціллю створення мобільного застосунку для персонального планування бюджету та часу є підвищення особистої продуктивності користувачів шляхом ефективного управління їхніми фінансовими та часовими ресурсами.

Основна мета полягає у наданні зручного інструменту, який допоможе відстежувати витрати, планувати фінансові операції, організовувати особистий час, встановлювати пріоритети та досягати особистих цілей.

2.2 Вимоги користувачів:

Внутрішній користувач – клієнт:

1. Робота з акаунтом користувача

1.1 Реєстрація акаунту користувача

1.2 Авторизація користувача

1.3 Зміна персональних даних акаунту користувача

1.4 Налаштування теми додатку

2. Робота з відмітками

2.1 Графік відміток про часові витрати

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

2.2 Графік відміток про грошові витрати

2.3 Створення нової відмітки

2.4 Редагування відмітки

2.5 Створення категорії відмітки

2.6 Видалення категорії відмітки

3. Генерування нової відмітки за допомогою AI

4. Вихід з акаунту користувача

2.3 Функціональні вимоги:

1. Система повинна надавати можливість користувачеві авторизуватись в системі за допомогою пошти чи анонімно.

2. Система повинна надавати можливість користувачеві змінювати дані свого акаунту.

3. Система повинна надавати графіки на основі часових витрат.

4. Система повинна надавати графіки на основі грошових витрат.

5. Система повинна надавати можливість користувачеві виходити з раніше авторизованого акаунту.

6. Система повинна надавати можливість користувачеві створювати нову відмітку.

7. Система повинна надавати можливість користувачеві редагувати дані раніше створеної відмітки.

8. Система повинна надавати можливість користувачеві створювати та видаляти категорію для відмітки.

9. Система повинна надавати можливість користувачеві генерувати нову відмітку за допомогою AI.

2.4 Не функціональні вимоги:

1. Інтуїтивність інтерфейсу: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та легким у використанні. Він повинен мати чітку структуру та бути дружнім до користувача, відповідаючи сучасним стандартам дизайну.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

2. Портативність: Система має бути сумісною з Android та IOS девайсами.

3. Стабільність: Надійність та стабільність системи є невід'ємною частиною її функціонування. Мета полягає в тому, щоб забезпечити неперервну роботу без аварій та помилок, що забезпечить користувачам безперебійний та стабільний досвід використання програмного забезпечення.

2.5 Системні вимоги:

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність мобільного пристрою з операційною системою на базі Android з версією 6.0 чи IOS 13.2 або вище;
- наявність доступу до Інтернету;
- для встановлення додатку на мобільному пристрої повинно бути не менше 50 МБ вільної пам'яті.

2.6 Розробка бази даних системи

Для збереження файлів та інформації у додатку використовуються Firebase (Firestore). Cloud Firestore — це гнучка масштабована база даних для розробки мобільних, веб-сайтів та серверів від Firebase та Google Cloud. Як і база даних Firebase Realtime [17], вона синхронізує дані між клієнтськими програмами через прослуховувачі в реальному часі та пропонує автономну підтримку для мобільних пристроїв та Інтернету, тому можна створювати адаптивні програми, які працюють незалежно від затримки мережі або підключення до Інтернету [18].

Основні принципи роботи Cloud Firestore зображені на рисунку 2.2:

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

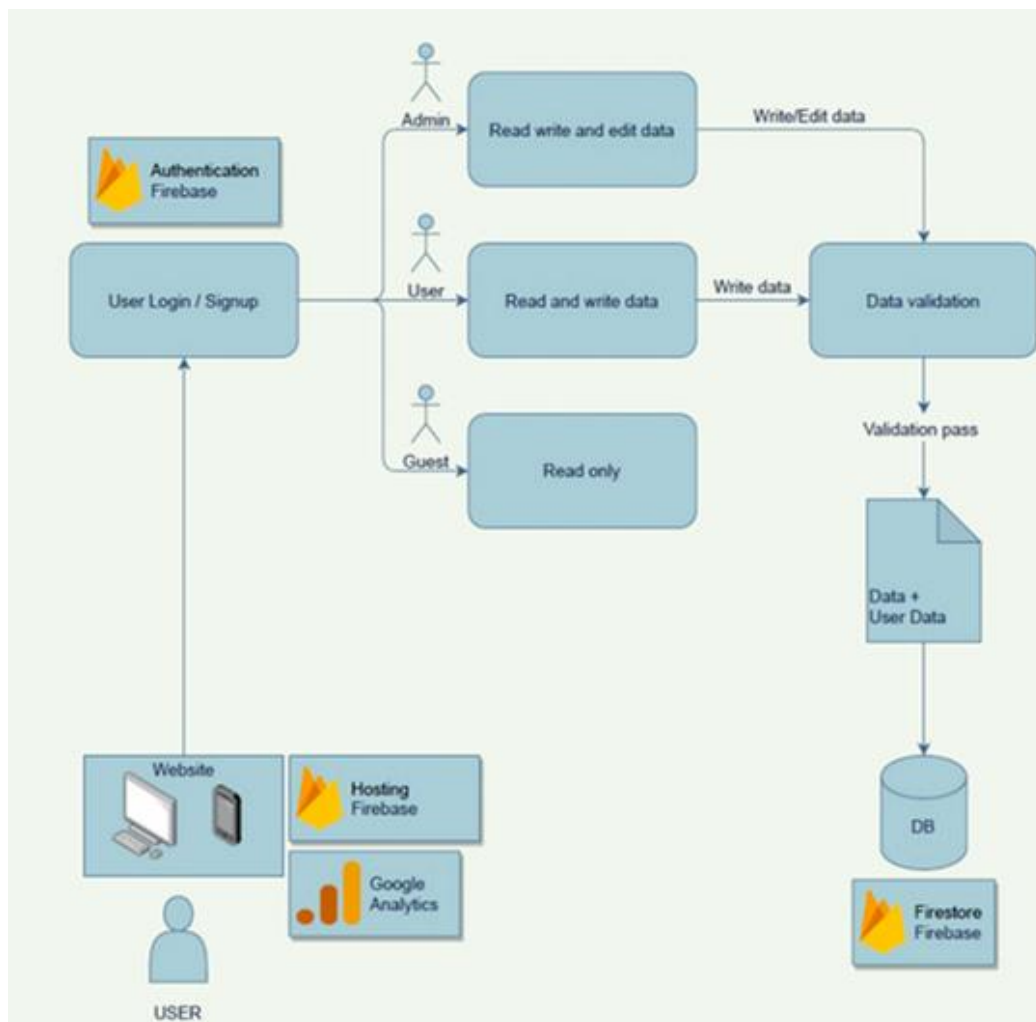


Рис. 2.2 – Діаграма принципів роботи FireStore

Cloud Firestore є сучасною базою даних NoSQL від Google, яка входить до складу платформи Firebase [19]. Завантажена діаграма ілюструє основні принципи роботи цієї системи. Спершу користувачі проходять процес входу або реєстрації через Firebase Authentication [20]. В залежності від рівня доступу, користувач може бути адміністратором, звичайним користувачем або гостем. Адміністратор має права на читання, запис і редагування даних, звичайний користувач має права на читання і запис даних, а гість має тільки права на читання даних.

Після успішного входу користувач взаємодіє з базою даних: адміністратори можуть читати, записувати і редагувати дані, звичайні користувачі можуть читати і записувати дані, а гості мають доступ тільки до читання даних. Записані дані проходять валідацію перед збереженням у базу

даних. Всі записані або змінені дані перевіряються на валідність, і якщо дані успішно проходять валідацію, вони зберігаються у Firestore Firebase.

Користувачі можуть взаємодіяти з даними через вебсайт або мобільний додаток, який хоститься за допомогою Firebase Hosting [21]. Для аналізу даних користувачів та їх взаємодії з вебсайтом або додатком використовується Google Analytics [22]. Діаграма показує, як різні ролі користувачів мають різні рівні доступу до бази даних, і як дані обробляються та зберігаються у Firestore Firebase після успішної аутентифікації і валідації.

Функціональні можливості Firestore:

Структура даних:

- Колекції та документи: У Firestore дані зберігаються у вигляді колекцій, що містять документи. Документи є основними одиницями зберігання і можуть містити структуровані дані у вигляді пар "ключ-значення". Кожен документ зберігається у певній колекції.

- Підколекції: Документи можуть містити підколекції, що дозволяє створювати складні структури даних з ієрархічними зв'язками.

Реальний час:

- Синхронізація: Firestore підтримує синхронізацію даних у режимі реального часу між клієнтськими додатками і базою даних. Зміни, внесені в одному місці, миттєво відображаються у всіх підключених клієнтах.

- Слухачі: Ви можете встановити слухачів для документів або колекцій, які автоматично оновлюють дані в додатку при будь-яких змінах у базі даних.

Запити:

- Гнучкі запити: Firestore підтримує потужні запити, які можуть включати фільтри, сортування та обмеження кількості результатів. Запити можуть бути виконані як на колекціях, так і на підколекціях.

- Індексация: Firestore автоматично індексує всі поля у документах, що забезпечує швидке виконання запитів.

Безпека:

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

- **Правила безпеки:** Firestore використовує правила безпеки на основі виразів для контролю доступу до даних. Ви можете визначати, хто і що може читати або записувати в базу даних.

- **Аутентифікація:** Firestore інтегрується з Firebase Authentication, що дозволяє легко керувати доступом користувачів на основі їхньої автентифікації.

Офлайн-підтримка:

- **Кешування:** Firestore підтримує роботу в офлайн-режимі, кешуючи дані на пристрої користувача. Зміни, внесені під час відсутності інтернет-з'єднання, синхронізуються з сервером, коли з'єднання відновлюється.

Масштабованість:

- **Автоматичне масштабування:** Firestore автоматично масштабується, обробляючи зростаючі обсяги даних і збільшуючи кількість запитів без необхідності ручного втручання.

2.7 Проектування та реалізація алгоритмів роботи системи

Застосунок є простим у використанні, має всі основні функції, що дозволяє користувачам легко взаємодіяти з ним. Алгоритм роботи з застосунком зображено на рисунку 2.3, також описано детально його пункти.

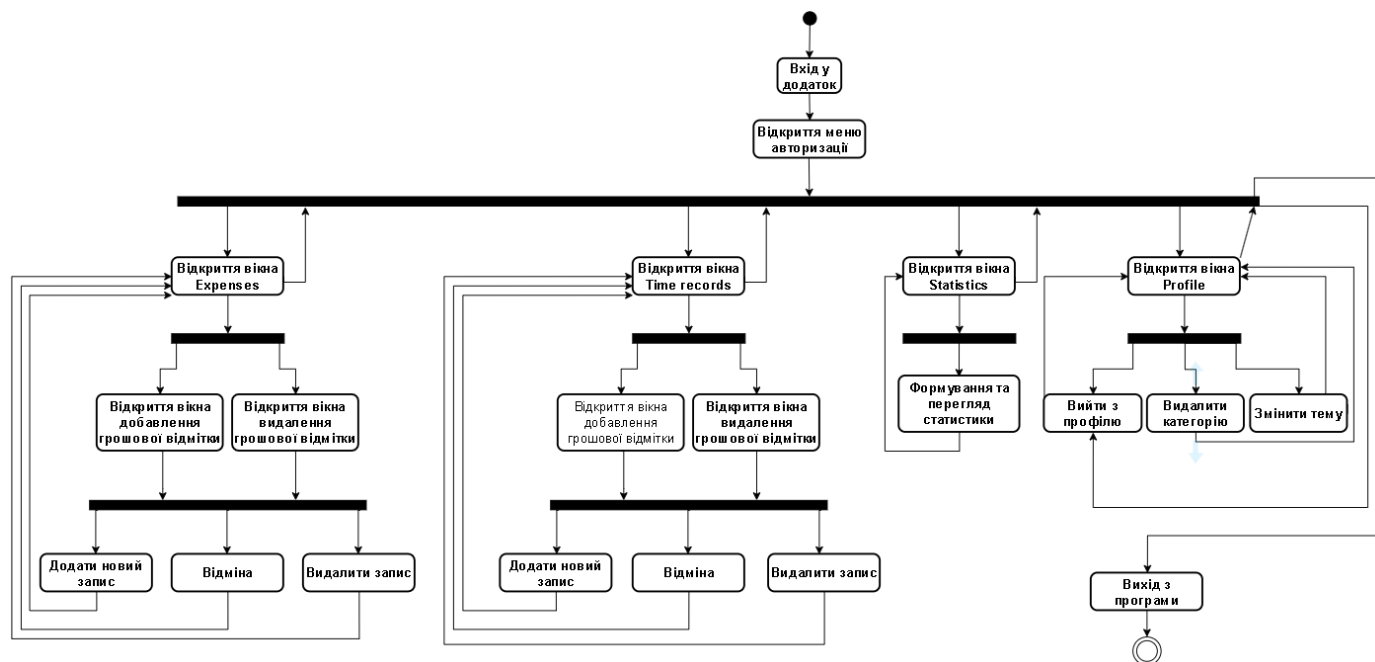


Рис. 2.3 Діаграма активності

Основні функції застосунку:

Користувач потрапляє на сторінку авторизації.

Користувач може зайти за допомогою пошти, або анонімно.

Якщо користувач хоче змінити дані – переходить на сторінку налаштування.

Якщо користувач хоче зробити грошову відмітку – переходить на сторінку створення грошову відмітки.

Якщо користувач хоче зробити часову відмітку – переходить на сторінку створення часову відмітки.

Якщо користувач хоче вийти з акаунту – переходить на сторінку налаштування та виходить з акаунту.

Якщо користувач хоче створити нову категорію – переходить на сторінку створення нової категорії.

Якщо користувач хоче видалити категорію – переходить на сторінку видалення категорії.

Якщо користувач хоче редагувати раніше створені відмітки – переходить на сторінку редагування відміток.

Якщо користувач хоче створити грошову відмітку за допомогою скриншоту – переходить на сторінку створення відмітки за допомогою AI.

2.8 Реалізація функціоналу мобільного застосунку для планування бюджету та часу.

Розглянемо основні етапи роботи з застосунком. Для цього користувач повинен здійснити реєстрацію нового аккаунту за допомогою Google чи авторизацію анонімно. Функція `signInWithGoogle` [23], яка дозволяє користувачеві увійти в систему за допомогою Google на рисунку 2.4.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

```

signInWithGoogle: async () : Promise<boolean> => {
  try {
    // Check if your device supports Google Play
    await GoogleSignin.hasPlayServices({ showPlayServicesUpdateDialog: true });
    // Get the users ID token
    const { idToken : string|null } = await GoogleSignin.signIn();

    // Create a Google credential with the token
    const googleCredential : FirebaseAuthTypes.AuthCredenti... = auth.GoogleAuthProvider.credential(idToken);

    const res : FirebaseAuthTypes.UserCredenti... = await auth().signInWithCredential(googleCredential);

    console.log("successfully signed in with google", res.user, res.additionalUserInfo);
    setSession(JSON.stringify(res.user));
    // Sign-in the user with the credential
  } catch (e) {
    console.error("ERROR while sign-in with google", e);
    return false;
  }
}

```

Рис. 2.4 Авторизація за допомогою Google

Функція `signInWithGoogle` перевіряє підтримку Google Play Services на пристрої, ініціює вхід через Google, створює облікові дані Google і входить у систему за допомогою цих облікових даних. У разі успіху, вона оновлює сесію користувача та повертає `true`. У разі помилки, вона логує помилку і повертає `false`.

Цей код налаштовує бібліотеку Google Sign-In для реалізації аутентифікації через обліковий запис Google в мобільному додатку (рис. 2.5).

```

GoogleSignin.configure({
  scopes: [], // what API you want to access on behalf of the user, default is email and pr
  webClientId: "523130753114-tp977vij4qpsrs0mihc79fvh52aepcch.apps.googleusercontent.com",
  offlineAccess: true, // if you want to access Google API on behalf of the user FROM YOUR
  hostedDomain: "", // specifies a hosted domain restriction
  forceCodeForRefreshToken: true, // [Android] related to `serverAuthCode`, read the docs 1
  accountName: "", // [Android] specifies an account name on the device that should be usec
  iosClientId: "523130753114-8n9hba1ldi7v5uaf18hspeu6cfinl60q.apps.googleusercontent.com",
  googleServicePlistPath: "", // [iOS] if you renamed your GoogleService-Info file, new nan
  openIdRealm: "", // [iOS] The OpenID2 realm of the home web server. This allows Google to
  profileImageSize: 120, // [iOS] The desired height (and width) of the profile image. Defc
});

```

Рис. 2.5 Налаштування бібліотеки Google Sign-In

Ці налаштування визначають поведінку бібліотеки Google Sign-In у додатку.

Цей фрагмент коду визначає функцію `signInAnonymously`, яка дозволяє користувачеві увійти в систему анонімно за допомогою Firebase Authentication (рис. 2.6).

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

```
signInAnonymously: async () : Promise<boolean> => {
  const res : FirebaseAuthTypes.UserCredenti... = await auth().signInAnonymously();
  console.log("successfully signed in anonymously", res.user, res.additionalUserInfo);
  setSession(JSON.stringify(res.user));
  return true;
},
```

Рис. 2.6 Анонімна авторизація

Функція `signInAnonymously` дозволяє користувачеві увійти в систему анонімно за допомогою Firebase Authentication [24]. Після успішного входу вона логує інформацію про користувача в консоль, зберігає дані користувача в сесію і повертає `true`, вказуючи на успішний вхід.

Метод `onSnapshot` [25] – оновлення в реальному часі (рис. 2.7).

```
.onSnapshot(documentSnapshot : FirebaseFirestoreTypes.QuerySn... => {
  setTimeRecords(documentSnapshot.docs.map((doc : FirebaseFirestoreTypes.QueryDo... ) => ({
    ...doc.data(),
    id: doc.id,
  }) as TimeRecords));
}, (error : Error ) : void => {
  console.log("useTimeRecords", error);
  setError(error);
}, () : void => {
  setIsLoading(false);
});
```

Рис. 2.7 Метод `onSnapshot`

Цей код підписується на зміни у певній колекції Firestore, обробляє отримані документи, оновлюючи стан компоненту з категоріями, обробляє можливі помилки, логуючи їх і оновлюючи стан помилки, а також встановлює стан завантаження як завершений після початкового запиту.

Після успішної авторизації користувач може створити, редагувати чи видаляти грошові відмітки.

Функції `createAction` та `updateAction` є функціями для створення та оновлення витрат в додатку відповідно (рис. 2.8).

```

const createAction : {create: (function(Expenses): ... = useCreateExpense();
const updateAction : {update: (function(Expenses): ... = useUpdateExpense(expenseId as string);
const [processingAction : boolean , setProcessingAction : React.Dispatch<React.SetStateA... ] = useState(false);
const handleActionButton = async (submitValues: Expenses) : Promise<void> => { Show usages
  setProcessingAction(true);

  try {
    if (mode === "create") {
      await createAction.create(submitValues);
    } else {
      await updateAction.update(submitValues);
    }

    navigation.goBack();
  } catch (e) {

  } finally {
    setProcessingAction(false);
  }
};

```

Рис. 2.8 Створення або оновлення грошових відміток

Даний код використовується для керування діями створення або оновлення витрат. Він використовує хуки стану для відстеження процесу дії, викликає відповідні методи для створення або оновлення витрати в залежності від режиму, а також здійснює перехід на попередню сторінку після успішної операції.

Хук useCreateExpense (рис. 2.9).

```

export const useCreateExpense = () : {create: (expense: Expenses) => Promise...} => {
  const create = useCallback(async (expense: Expenses) : Promise<void> => {
    await firestore() Module
      .collection("Expenses") CollectionReference<DocumentData>
      .add(expense);
  }, []);

  return {
    create,
  };
};

```

Рис. 2.9 Хук створення грошових відміток

Цей хук визначає та повертає функцію create, яка додає новий запис витрат до колекції "Expenses" у Firestore.

Хук [26] useUpdateExpense (рис. 2.10).

```
export const useUpdateExpense = (id: string) : {update: (expense: Expenses) => Promise...} => {
  const update = useCallback(async (expense: Expenses) : Promise<void> => {
    await firestore() Module
      .collection("Expenses") CollectionReference<DocumentData>
      .doc(id) DocumentReference<DocumentData>
      .update(expense);
  }, [id]);

  return {
    update,
  };
};
```

Рис. 2.10 Хук оновлення грошових відміток

Цей хук визначає та повертає функцію update, яка оновлює існуючий запис витрат у колекції "Expenses" у Firestore за заданим ідентифікатором.

Хук useRemoveExpense (рис. 2.11).

```
export const useRemoveExpense = () : {remove: (id: string) => Promise<void>} => {
  const remove = useCallback(async (id: string) : Promise<void> => {
    await firestore() Module
      .collection("Expenses") CollectionReference<DocumentData>
      .doc(id) DocumentReference<DocumentData>
      .delete();
  }, []);

  return {
    remove,
  };
};
```

Рис. 2.11 Хук видалення грошових відміток

Цей хук визначає та повертає функцію remove, яка видаляє існуючий запис витрат з колекції "Expenses" у Firestore за заданим ідентифікатором.

Хук useCreateCategory (рис. 2.12).

```
export const useCreateCategory = () => { Show usages
  const create = useCallback(async (category: Category) => {
    return firestore() Module
      .collection(COLLECTION) CollectionReference<DocumentData>
      .add(category);
  }, []);

  return {
    create,
  };
};
```

Рис. 2.12 Хук створення категорії

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей хук визначає та повертає функцію create для додавання нової категорії до колекції Firestore. Функція create приймає об'єкт category, додає його до вказаної колекції за допомогою методу add.

Хук useRemoveCategory (рис. 2.13).

```
export const useRemoveCategory = () => { Show usages
  const remove = useCallback(async (id: string) => {
    await firestore() Module
      .collection(COLLECTION) CollectionReference<DocumentData>
      .doc(id) DocumentReference<DocumentData>
      .delete();
  }, []);

  return {
    remove,
  };
};
```

Рис. 2.13 Хук видалення категорії

Цей хук визначає та повертає функцію remove для видалення категорії з колекції Firestore, використовуючи ідентифікатор категорії (id). Ця функція звертається до вказаної колекції, знаходить документ за ідентифікатором і видаляє його.

Метод SecureStorage (рис. 2.14).

```
SecureStore.getItemAsync(key).then(value : string | null => {
  setState(value);
});
```

Рис. 2.14 Метод SecureStorage

Цей код отримує значення з SecureStore і, коли значення отримане, оновлює стан компонента у React цим значенням. Це забезпечує безпечне зберігання та використання конфіденційних даних в додатку.

Цей код реалізує функцію для вибору зображення з бібліотеки зображень на пристрої користувача [27], розпізнавання даних на зображенні за допомогою штучного інтелекту, створення нової категорії (якщо вона не існує) та обробки отриманих даних. (рис. 2.15)

```

const createCategoryAction : {create: (function(Category): ... = useCreateCategory();

const pickImage = async () : Promise<void> => { Show usages
// No permissions request is necessary for launching the image library
let result : ImagePicker.ImagePickerResult = await ImagePicker.launchImageLibraryAsync({
  mediaTypes: ImagePicker.MediaTypeOptions.All,
  allowsEditing: false,
  quality: 0.7,
  base64: true,
});

if (!result.canceled) {
  try {
    setProcessingAI(true);
    const recognition : string | null = await AI.getInstance().getMoneyRecordRecognition(`data:image/jpeg;base64,${result.assets[0].base64}`);

    console.log("recognition", recognition);

    if (typeof recognition !== "string") {
      throw new Error("Recognition is not a string");
    }
    const parsed = JSON.parse(recognition);

    let category : Category | undefined = categories.find((c : Category) : boolean => c.name === parsed.category);

    if (category === undefined) {
      const res : FirebaseFirestoreTypes.DocumentSnapshot = await createCategoryAction.create({
        name: parsed.category,
        userId: session.uid,
        order: 0,
        description: "",
      });
      const data : FirebaseFirestoreTypes.DocumentSnapshot = await res.get();
      category = {
        ...data.data(),
        id: data.id,
      } as Category;
    }
  }
}

```

Рис. 2.15 Створення грошової відмітки за допомогою AI

Цей код обробляє результат розпізнавання витрат за допомогою AI [28], перевіряє, чи результат не відмінено, встановлює стан обробки, отримує результат розпізнавання витрат у форматі base64, перевіряє, чи результат є рядком, парсить його як JSON, знаходить відповідну категорію з наявних або створює нову категорію, якщо така не знайдена, і нарешті оновлює категорії з розпізнаною інформацією.

Функція useThemeColor визначає колір [29], який буде використовуватися в залежності від поточної теми (світлої або темної) та наданих кольорів (рис. 2.16)

```

export function useThemeColor( Show usages
  props: { light?: string; dark?: string },
  colorName: keyof typeof Colors.light & keyof typeof Colors.dark
) : string {
  const theme : "light" | "dark" = useColorScheme() ?? 'light';
  const colorFromProps : string | undefined = props[theme];

  if (colorFromProps) {
    return colorFromProps;
  } else {
    return Colors[theme][colorName];
  }
}

```

Рис. 2.16 Зміна теми застосунку

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Висновки до другого розділу

В процесі проектування мобільного застосунку були проаналізовані усі варіанти використання програмного продукту відповідно до вимог програмного забезпечення. На основі отриманого результату була спроектована структура платформи. Проектування включало в себе моделювання структури проекту та взаємодії між їх складовими частинами.

Також було описано базу даних для даних які будуть зберігатися на серверній частині застосунку, таких як користувачі, категорії, грошові та часові відмітки.

Було описано алгоритм роботи з додатком та описані послідовні пункти варіантів його використання.

Також був наведений опис основного функціоналу додатку – це створення користувачем акаунту, створення грошових та часових відміток, створення грошових відміток за допомогою AI та налаштування профілю.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З МОБІЛЬНИМ ДОДАТКОМ

3.1 Порядок встановлення та налаштування параметрів мобільного застосунку

Встановлення та налаштування мобільного застосунку вимагає розгортання всіх його компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому.

Для опису архітектури мобільного застосунку було створено діаграму розгортання (рис. 3.1).

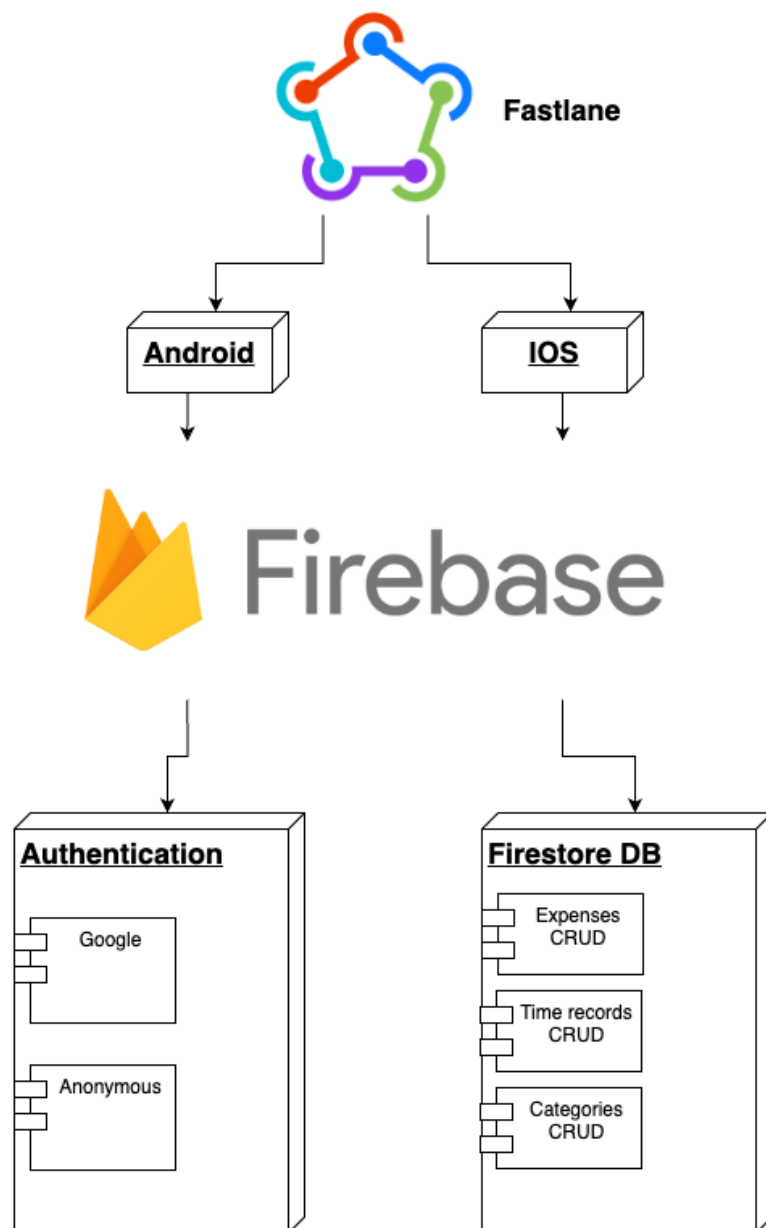


Рис. 3.1 Діаграма розгортання мобільного додатку

Опис процесу розгортання.

Метою є розгортання мобільного застосунку для Android та iOS, з використанням Fastlane [30] для автоматизації процесів CI/CD. Застосунок інтегрується з Firebase для забезпечення автентифікації користувачів та бази даних Firestore для управління CRUD операціями для витрат, записів часу та категорій.

3.1.1 Етапи розгортання

1. Налаштування Fastlane автоматизує процеси розробки, тестування та розгортання мобільних додатків. Основні кроки:

2. Розробка мобільного додатку (Android / iOS)

3. Інтеграція Firebase

- Додавання Firebase до Android/iOS проекту
- Налаштування Firebase у проекті

4. Автентифікація через Firebase

- Google автентифікація:
Використання Firebase Auth SDK для Google Sign-In.
- Анонімна автентифікація

5. Firestore DB для CRUD операцій

- Створюється колекції для витрат, записів часу та категорій у Firestore.

З вищенаведеними кроками ви зможете розгорнути мобільний застосунок для Android та iOS, інтегрований з Firebase для автентифікації та управління даними через Firestore. Fastlane значно полегшить процес розгортання додатку на обох платформах.

3.2 Структура інтерфейсу мобільного додатку

На сьогоднішній день додаток можна інсталювати, завантаживши його за допомогою Expo go.

Після завершення інсталяції на головному екрані мобільного пристрою має з'явитися іконка цього додатку. Якщо іконка не з'явилася, це свідчить про

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

неуспішну інсталяцію. У випадку успішної інсталяції користувач зможе відкрити додаток, натиснувши на його іконку. Після цього має відкритися стартова сторінка додатку.

Користування додатком розпочинається з сторінки авторизації.

Користувач може зайти в додаток за допомогою пошти чи анонімно (рис. 3.2)

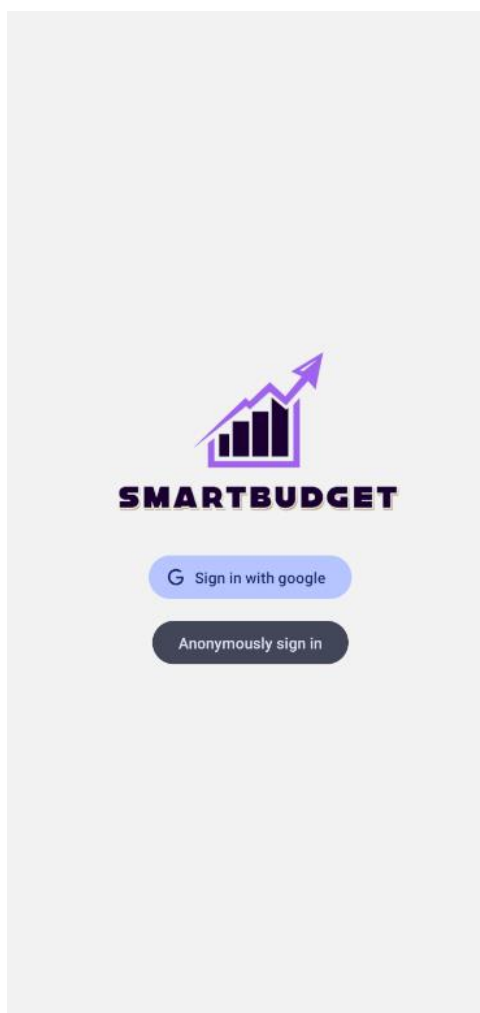


Рисунок 3.2 Сторінка авторизації

На цій сторінці користувач може авторизуватись через обліковий запис Google чи анонімно. Після успішної авторизації користувач буде перенаправлений на свій обліковий запис і автоматично направлений на сторінку перегляду фінансових відміток, як показано на рисунку 3.3.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

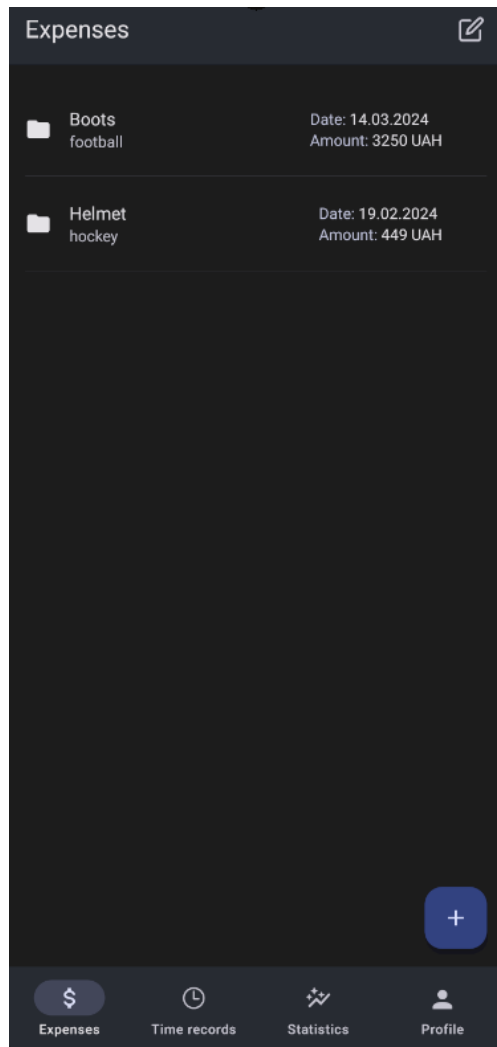


Рисунок 3.3 Сторінка перегляду грошових відміток

На цій сторінці відображаються всі грошові записи, створені користувачем. Для нового користувача, який щойно встановив додаток, ця сторінка буде порожньою. Для створення нового грошового запису користувач повинен натиснути кнопку "Додати новий запис", яка розташована у нижній частині сторінки. Після цього його буде перенаправлено на додаткову сторінку для проходження процесу створення нового грошового запису. Таку сторінку можна побачити на рисунку 3.4.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

Рисунок 3.4 Сторінка створення грошових відміток

Для створення нового запису користувачу потрібно вказати інформацію про назву, опис, категорію, дату та суму витрачених коштів у відповідні поля форми, а потім натиснути кнопку "Create". Користувач може обрати категорію з переліку існуючих або створити нову категорію, натиснувши на кнопку "New" поряд з полем "Select category". Після натискання цієї кнопки користувач буде направлений на сторінку створення нової категорії. Її можна побачити на рисунку 3.4.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

← New Expense

Name

Description

Select an item... New

Amount

Create new category.

Category name

Close Create

Create

Рисунок 3.5 Сторінка створення нової категорії

Для створення нової категорії користувач повинен вказати назву та натиснути кнопку для збереження запису. Після цього він буде повернутий до попередньої процедури, де він може використати нову категорію. Також користувач може створити грошову відмітку за допомогою скріншоту та штучного інтелекту, детальніше на рисунку 3.5.1

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

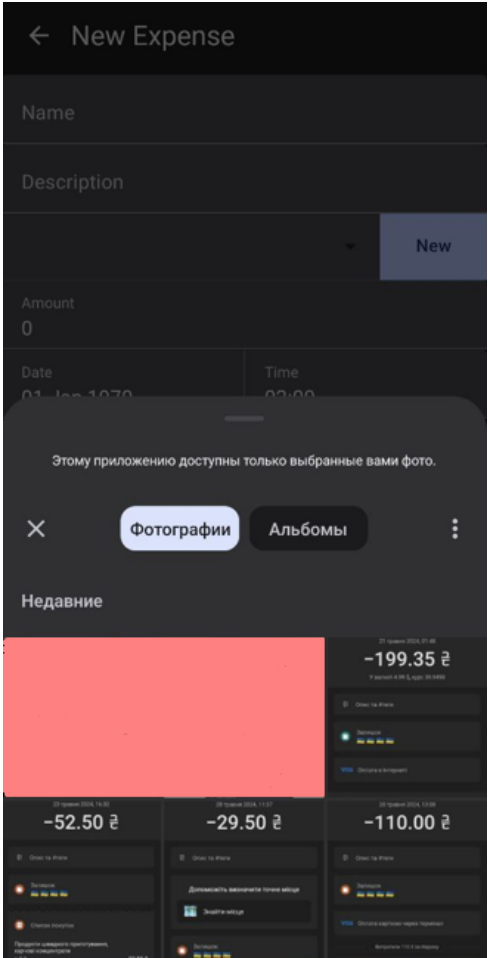


Рисунок 3.5.1 Сторінка створення нової категорії за допомогою AI

Коли користувач завершує створення запису, його повертає на головну сторінку з відмітками. Після цього він може повторити цей процес стільки разів, скільки необхідно, або перейти до створення часових відміток. Для цього він може обрати потрібний пункт у меню навігації на рисунку 3.6.

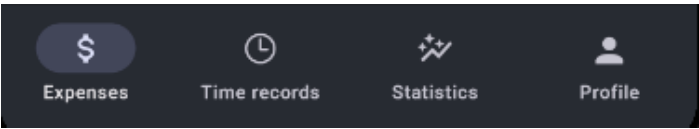


Рисунок 3.6 Панель навігації [7][8]

Для створення нової часової відмітки користувач повинен перейти на сторінку зображену на рисунку 3.7.

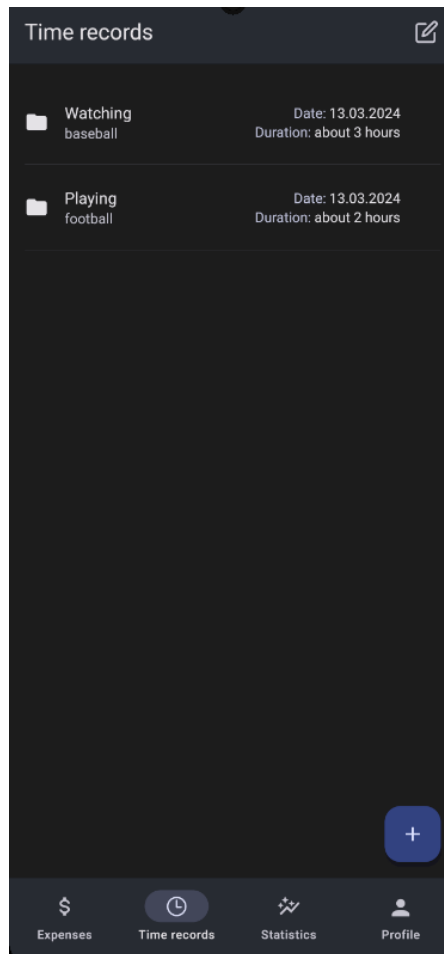


Рисунок 3.7 Сторінка перегляду часових відміток

На цій сторінці відображаються всі часові відмітки, створені користувачем. Для нового користувача, який щойно встановив додаток, ця сторінка буде порожньою. Для створення нової часової відмітки користувач повинен натиснути кнопку "Додати новий запис", яка розташована у нижній частині сторінки. Після цього його буде перенаправлено на додаткову сторінку для проходження процесу створення нового грошового запису. Таку сторінку можна побачити на рисунку 3.8.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

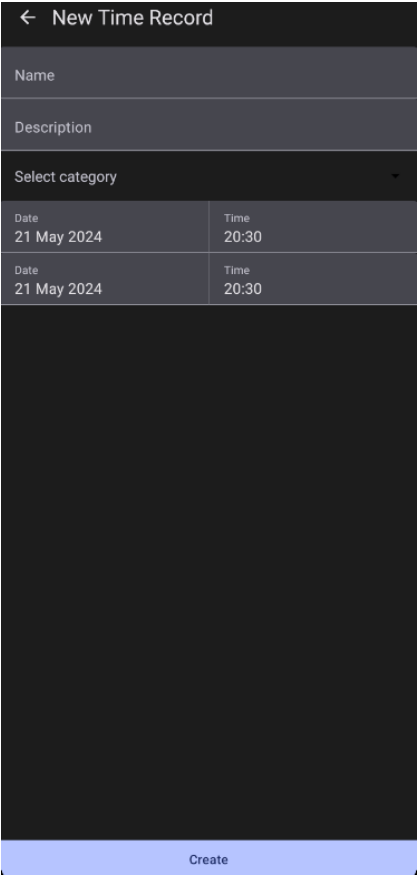
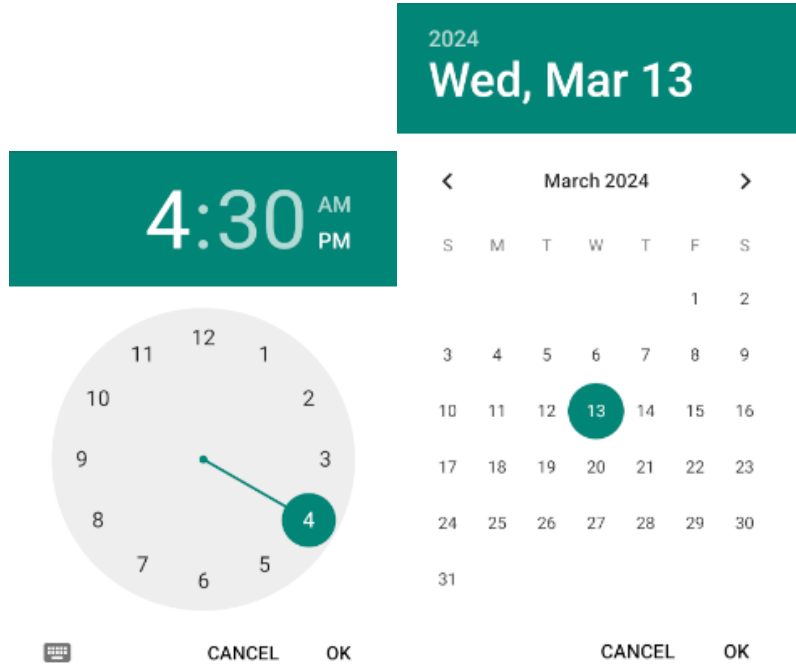


Рисунок 3.8 Сторінка створення часових відміток

Тут користувач може заповнити відповідні поля обравши дату та визначивши часовий проміжок за допомогою віджетів зображених на рисунках 3.9 та 3.10.



Рисунки 3.9 та 3.10 Віджет вибору часу та дати

Коли користувач завершує створення запису, його повертає на головну сторінку з відмітками. Після цього він може повторити цей процес стільки разів, скільки необхідно, або перейти на сторінку статистики. Для цього він може обрати потрібний пункт у меню навігації на рисунку 3.11.

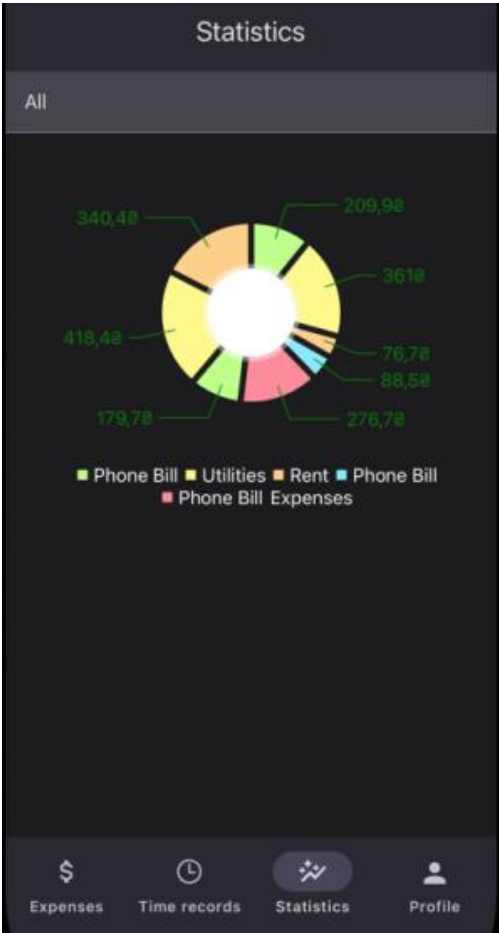


Рисунок 3.11 Сторінка статистики

Тут користувач може подивитися статистику свого профілю, що основана на відмітках створених ним. Користувач також може перейти до сторінки профілю, де може відредагувати свої дані та змінити тему додатку. Перехід на сторінку зображену на рисунку 3.12 можна здійснити з одного з меню додатку.

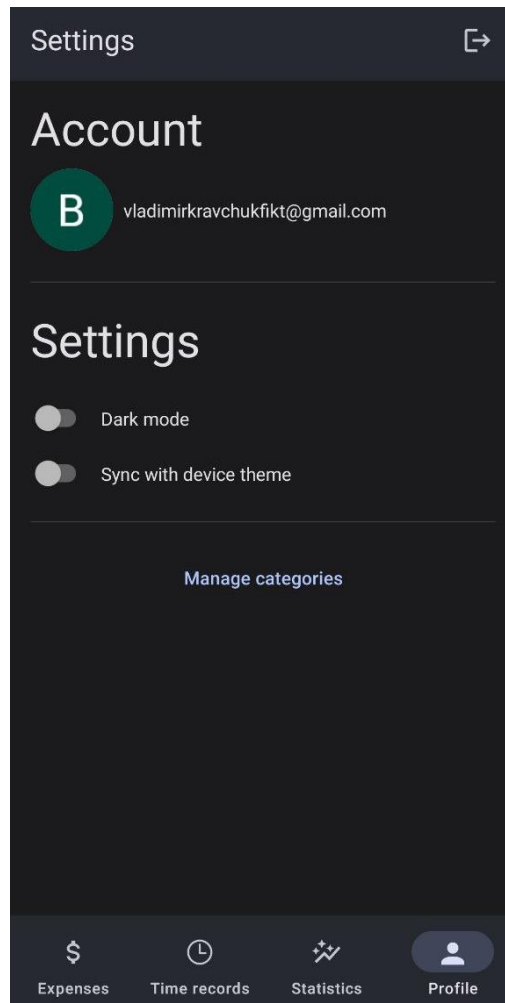


Рисунок 3.12 Сторінка профілю

Тут користувач може змінити тему застосунку. Користувач також може перейти до сторінки видалення категорії, де може видалити категорію. Перехід на сторінку зображену на рисунку 3.13 можна здійснити з меню профілю додатку.

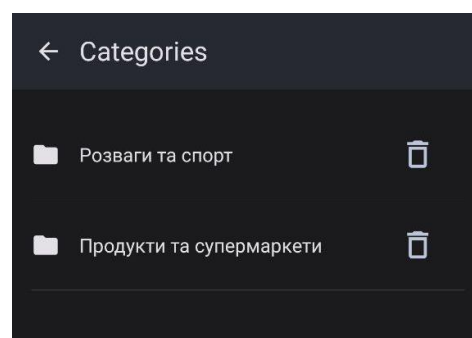


Рисунок 3.13 Сторінка видалення категорії

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

3.3 Загальна статистика мобільного додатку

У адміністрації є можливість переглянути статистику користування додатком, це дозволяє відслідковувати динаміку створення нових грошових чи часових відміток. Адміністрація має доступ до всіх відміток, тому в них відображається загальна статистика, та скільки відміток було створено та видалено рис. 3.14.

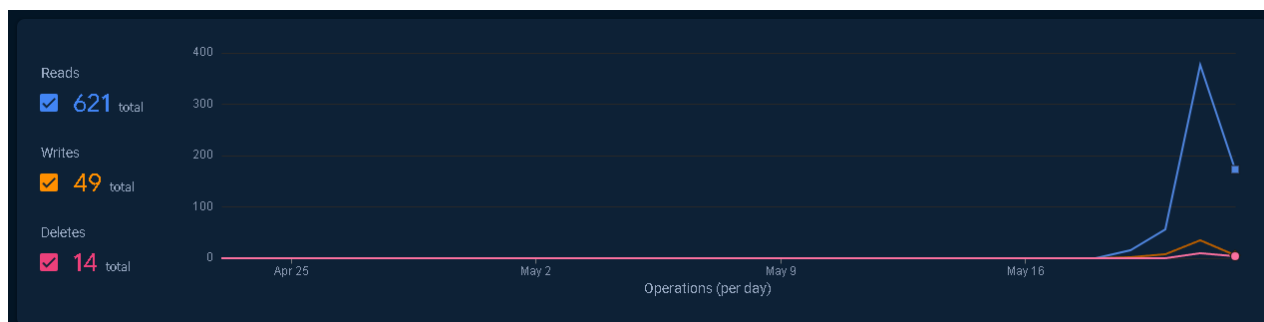


Рисунок 3.14 Статистика взаємодії з відмітками

Адміністратор має можливість відслідкувати кількість активних користувачів, які авторизовувались протягом дня чи місяця. Також можна виключити зі статистики анонімних користувачів рис. 3.15 – 3.16.

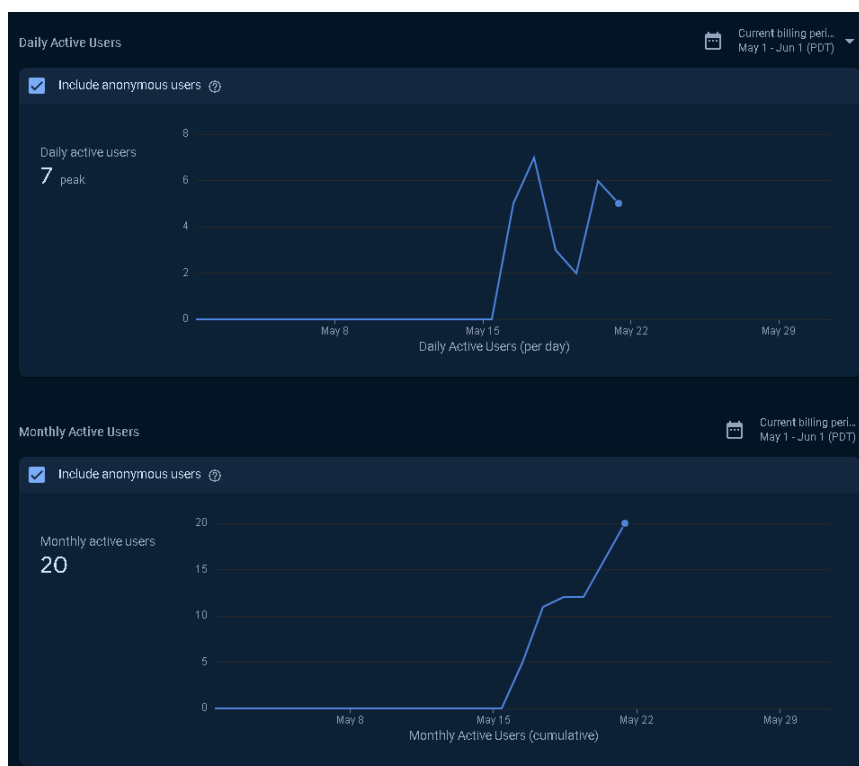


Рисунок 3.15 Статистика активних користувачів

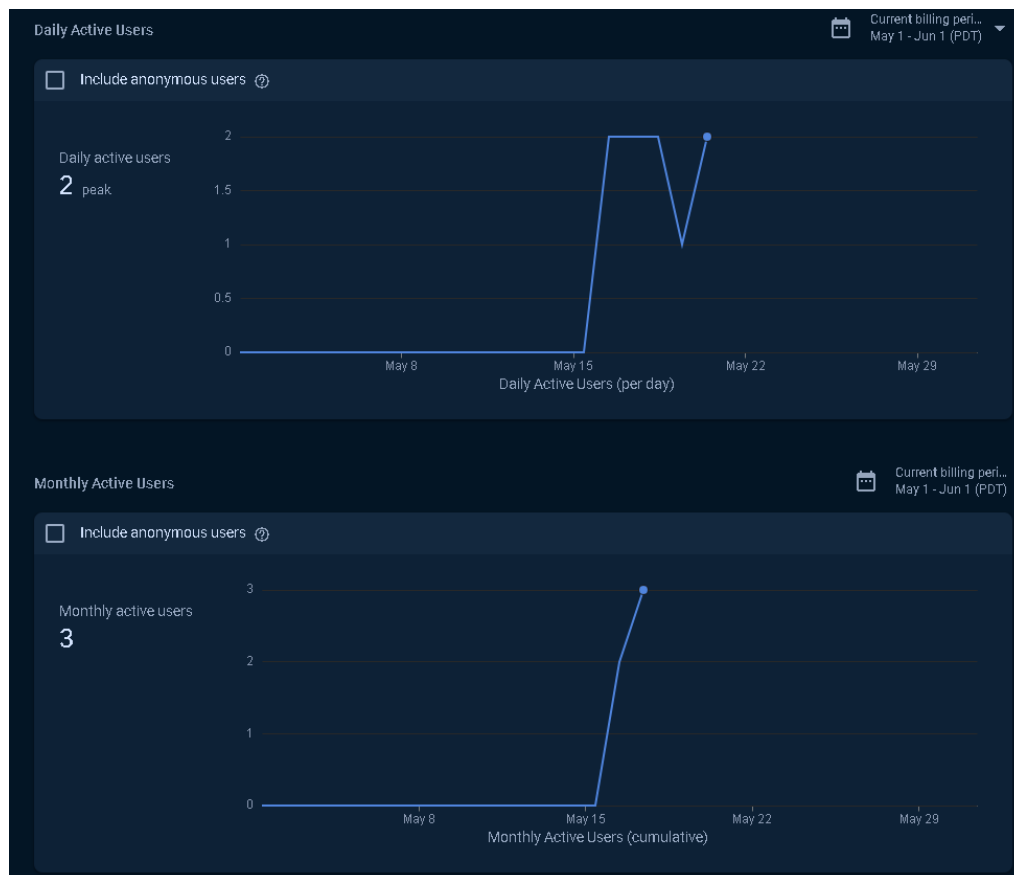


Рисунок 3.16 Статистика активних користувачів (без анонімних)

3.4 Тестування платформи

3.4.1 Мета тестування

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка сумісності застосунку з операційною системою Android, що націлена на мобільні пристрої;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.

3.4.2 Методи тестування

Для перевірки програмного забезпечення можуть бути застосовані різні методи [31]:

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

1. **Статичне тестування** – цей метод передбачає перевірку програми разом із супровідною документацією. Аналізується дотримання стандартів програмування без виконання коду.

2. **Динамічне тестування** – цей метод використовується під час виконання програми. Програмне забезпечення тестується на основі визначеного набору тестів, під час яких збираються та аналізуються дані щодо проблем та помилок у роботі програми.

3. **Функціональне тестування** – передбачає перевірку того, чи відповідає фактична поведінка програмного забезпечення очікуванням та вимогам специфікації.

4. **Системне тестування** – охоплює перевірку всього програмного забезпечення в цілому, що дозволяє оцінити його функціонування у комплексі.

5. **Мануальне тестування** – виконується без використання автоматизованих інструментів. Тест-кейси складає і виконує вручну тестувальник, що дозволяє виявити помилки, які можуть бути пропущені автоматизованими тестами.

6. **Тестування «білої скриньки»** – фокусується на внутрішній структурі програми. Перевіряється правильність реалізації всіх компонентів програми та їх взаємодії.

3.4.3 Засоби та порядок тестування

Мануальне тестування проводиться для виявлення помилок і недоліків у функціональних частинах програмного забезпечення, а також для підвищення зручності використання. Для перевірки працездатності і відмовостійкості додатки необхідно виконати наступні види тестів:

1. **Динамічне тестування** на відповідність функціональним вимогам.

2. **Тестування на мобільних пристроях** з різною роздільною здатністю екрана.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

3. **Тестування на мобільних пристроях з різними версіями операційної системи.**

4. **Тестування повідомлень про помилки, щоб переконатися у коректному виведенні відповідних повідомлень у разі необхідності.**

5. **Тестування працездатності програми при відсутності з'єднання з мережею.**

6. **Тестування інтерфейсу користувача, щоб перевірити зручність та логічність його використання.**

7. **Тестування зручності використання, щоб забезпечити, що програма є інтуїтивно зрозумілою та легкою у користуванні.**

3.4.4 Мануальне тестування

Мануальне тестування [32] - це процес, коли перевірка програмного продукту виконується вручну, без використання автоматизованих інструментів або скриптів. Програмне забезпечення пройшло ручне тестування, і подробиці цих перевірок можна знайти у відповідних таблицях в діапазоні від 3.1 до 3.12.

Таблиця 3.1 – Тест 1

Тест «Логін користувача за допомогою пошти»	
Модуль	Аунтетифікація
Початковий стан системи	Користувач відкрив застосунок
Вхідні данні	Обліковий запис Google
Опис проведення тесту	Користувач вибирає опцію входу за допомогою Google і вводить дані свого облікового запису
Очікуваний результат	Користувач перенаправляється на сторінку з грошовими витратами
Фактичний результат	Користувач перенаправляється на сторінку з грошовими витратами

Таблиця 3.2 – Тест 2

Тест «Створення грошової відмітки»	
Модуль	Грошові витрати
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Назва, опис, дата, категорія, сума витрати
Опис проведення тесту	Користувач вводить необхідні дані про витрати та натискає кнопку "Add new record"
Очікуваний результат	Користувач перенаправляється на сторінку з грошовими витратами
Фактичний результат	Користувач перенаправляється на сторінку з грошовими витратами

Таблиця 3.3 – Тест 3

Тест «Створення часової відмітки»	
Модуль	Часові витрати
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Назва, опис, дата, категорія, час з, час до
Опис проведення тесту	Користувач вводить необхідні дані про витрати та натискає кнопку "Add new record"
Очікуваний результат	Користувач перенаправляється на сторінку з часовими витратами
Фактичний результат	Користувач перенаправляється на сторінку з часовими витратами

Таблиця 3.4 – Тест 4

Тест «Створення часової відмітки»	
Модуль	Часові витрати
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Назва, опис, дата, категорія, час з, час до
Опис проведення тесту	Користувач вводить необхідні дані про витрати та натискає кнопку "Add new record"
Очікуваний результат	Користувач перенаправляється на сторінку з часовими витратами
Фактичний результат	Користувач перенаправляється на сторінку з часовими витратами

Таблиця 3.5 – Тест 5

Тест «Створення грошової відмітки за допомогою AI»	
Модуль	Грошові витрати
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Скріншот грошової витрати
Опис проведення тесту	Користувач вибирає скріншот з галереї
Очікуваний результат	Поля введення інформації про відмітку заповнюються даними
Фактичний результат	Поля введення інформації про відмітку заповнюються даними

Таблиця 3.6 – Тест 6

Тест «Редагування грошової відмітки»	
Модуль	Грошові витрати
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Назва, опис, дата, категорія, сума витрати
Опис проведення тесту	Користувач вводить необхідні дані про витрати та натискає кнопку "Save"
Очікуваний результат	Користувач перенаправляється на сторінку з грошовими витратами
Фактичний результат	Користувач перенаправляється на сторінку з грошовими витратами

Таблиця 3.7 – Тест 7

Тест «Зміна теми оформлення додатку»	
Модуль	Профіль
Початковий стан системи	Аунтетифікований користувач, що перейшов на сторінку профілю
Вхідні данні	Вхідних даних немає
Опис проведення тесту	Користувач натискає на кнопку перемикання теми
Очікуваний результат	Тема додатку змінюється на протилежну
Фактичний результат	Тема додатку змінюється на протилежну

Таблиця 3.8 – Тест 8

Тест «Генерація статистики»	
Модуль	Статистика
Початковий стан системи	Аунтетифікований користувач
Вхідні данні	Грошові та часові відмітки
Опис проведення тесту	Користувач заходить на сторінку зі статистикою
Очікуваний результат	Користувачу виводиться статистика по категоріям, часовим та грошом витратам у вигляді графіків
Фактичний результат	Користувачу виводиться статистика по категоріям, часовим та грошом витратам у вигляді графіків

Таблиця 3.9 – Тест 9

Тест «Створення нової категорії»	
Модуль	Категорії
Початковий стан системи	Меню створення категорій
Вхідні данні	Назва
Опис проведення тесту	Користувач вводить дані про категорію та натискає кнопку «Save»
Очікуваний результат	Користувачу перенаправляється на попередню сторінку
Фактичний результат	Користувачу перенаправляється на попередню сторінку

Таблиця 3.10 – Тест 10

Тест «Видалення часової відмітки»	
Модуль	Часові відмітки
Початковий стан системи	Меню редагування часової відмітки
Вхідні данні	Часова відмітка
Опис проведення тесту	Користувач натискає кнопку «Delete» в меню редагування відмітки
Очікуваний результат	Користувачу перенаправляється на сторінку з часовими відмітками
Фактичний результат	Користувачу перенаправляється на сторінку з часовими відмітками

Таблиця 3.11 – Тест 11

Тест «Редагування часової відмітки»	
Модуль	Часові відмітки
Початковий стан системи	Меню редагування грошової відмітки
Вхідні данні	Назва, опис, дата, категорія, час з, час до
Опис проведення тесту	Користувач вводить вхідні дані та натискає кнопку «Save» в меню редагування відмітки
Очікуваний результат	Користувачу перенаправляється на сторінку з часовими відмітками
Фактичний результат	Користувачу перенаправляється на сторінку з часовими відмітками

Таблиця 3.12 – Тест 12

Тест «Видалення категорій»	
Модуль	Категорії
Початковий стан системи	Меню профілю
Вхідні данні	Меню видалення категорій
Опис проведення тесту	Користувач натискає кнопку «Delete» в меню редагування категорій
Очікуваний результат	Користувачу перенаправляється на попередню сторінку
Фактичний результат	Користувачу перенаправляється на попередню сторінку

Висновки до третього розділу

В даному розділі була наведена детальна схема розгортання платформи. Кожен структурний елемент системи був описаний, а також було продемонстровано його взаємодію з іншими частинами платформи. Це дозволило зрозуміти архітектуру системи та взаємозв'язки між її компонентами.

Далі було надано опис інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку.

Окрім того, було продемонстровано, як працює статистика для адміністрації та звичайних користувачів.

Разом з цим, було описано процес мануального тестування, де програмне забезпечення піддавалося випробуванням без використання автоматизованих інструментів. Цей підхід дозволяв перевірити функціональність, зручність та відповідність інтуїтивним вимогам користувачів.

Мануальний процес тестування та аналізу якості забезпечує об'єктивну оцінку якості розробленого програмного продукту, виявлення потенційних проблем та недоліків, а також покращення ефективності процесу розробки та підтримки програмного забезпечення.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

ВИСНОВКИ

У результаті виконання дипломного проекту було розроблено мобільний додаток для персонального планування бюджету та часу. Додаток створено з використанням JavaScript та фреймворку React Native і використовує Firestore як базу даних для зберігання і керування інформацією. Це забезпечило зручний та ефективний інструмент для управління фінансами та часом користувачів.

Для розробки використовувалось IDE WebStorm [33], яке пропонує широкі можливості для програмістів JavaScript, включаючи зручне відлагодження та керування проектом.

Додаток надає функції створення бюджету, відстеження фінансових та часових витрат, а також генерування статистики у різних категоріях. Це дозволяє користувачам ефективно та організовано керувати своїми фінансами та часом.

Мобільний додаток для персонального планування бюджету та часу має великий потенціал для поліпшення управління фінансами та часом користувачів. Його впровадження може сприяти підвищенню фінансової грамотності та ефективності використання часу, що позитивно вплине на якість життя користувачів. Це підкреслює соціально-економічну важливість даного програмного продукту.

Результатом роботи є створення додатку, який може бути використаний для покращення фінансового планування та ефективного управління часом користувачами. Додаток може служити основою для подальшого розвитку та досліджень у галузі персонального планування та управління ресурсами, включаючи розширення функціональності, покращення інтерфейсу користувача та інтеграцію з іншими сервісами та платформами.

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Introduction to JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://javascript.info/intro#languages-over-javascript>
2. Introduction to The Current Technology – React Native [Електронний ресурс] – Режим доступу до ресурсу: <https://aboutreact.com/intro-react-native/>
3. TypeScript is JavaScript with syntax for types [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/>
4. Expo Go Community Guidelines [Електронний ресурс] – Режим доступу до ресурсу: <https://expo.dev/go>
5. Плюси і мінуси Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.goodteam.dev/blog/plyusy-i-minusy-firebase>
6. What is a mobile app (mobile application) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/whatis/definition/mobile-app>
7. 16 Best Techniques for Creating a User-Friendly Interface [Електронний ресурс] – Режим доступу до ресурсу: <https://geniusee.com/single-blog/16-techniques-for-creating-a-user-friendly-interface>
8. Monefy [Електронний ресурс] – Режим доступу до ресурсу: <https://monefy.me/>
9. MoneyLover [Електронний ресурс] – Режим доступу до ресурсу: <https://moneylover.me/>
10. Asana [Електронний ресурс] – Режим доступу до ресурсу: <https://asana.com/>
11. Clockify [Електронний ресурс] – Режим доступу до ресурсу: <https://clockify.me/>
12. JavaScript — дослідження SlashData [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/35342/>

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

- 13.Різниця між TypeScript та JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/typescript/>
- 14.Firebase як бекенд для будь-яких застосунків, та як використовувати Firebase-сервіси [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/44058/>
- 15.React Native challenges and benefits in 2024 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.purrweb.com/blog/react-native/>
- 16.Expo Go - Configure with app config [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.expo.dev/workflow/configuration/>
- 17.Realtime Database [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/database/rtdb-vs-firestore>
- 18.React Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://firebaseopensource.com/projects/rakannimer/react-firebase/>
- 19.Cloud Firestore [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/firestore>
- 20.Firebase Authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/auth>
- 21.Firebase Hosting [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/hosting>
- 22.Start learning about Google Analytics [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.google.com/analytics>
- 23.Get realtime updates with Cloud Firestore [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/firestore/query-data/listen>
- 24.Authenticate with Firebase Anonymously Using JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/docs/auth/web/anonymous-auth>
- 25.Linking [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.expo.dev/guides/linking/>

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

- 26.Хуки в React [Електронний ресурс] – Режим доступу до ресурсу:
<https://foxminded.ua/huky-tse/>
- 27.Expo ImagePicker [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.expo.dev/versions/latest/sdk/imagepicker/>
- 28.Community libraries ChatGPT [Електронний ресурс] – Режим доступу до ресурсу: <https://platform.openai.com/docs/libraries/community-libraries>
- 29.Color themes [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.expo.dev/develop/user-interface/color-themes/#tips>
- 30.Fastlane - Automate your development and release process [Електронний ресурс] – Режим доступу до ресурсу: <https://fastlane.tools/>
- 31.Тестування програмного забезпечення: типи, види та застосування [Електронний ресурс] – Режим доступу до ресурсу:
<https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/>
- 32.Ручне або Manual Testing [Електронний ресурс] – Режим доступу до ресурсу: <https://university.sigma.software/manual-testing-vs-automation-testing/>
- 33.WebStorm Features [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.jetbrains.com/webstorm/features/>

ДОДАТКИ

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

sign-in.tsx

```
import {Redirect, router} from 'expo-router';
import {Image, StyleSheet, Text, View} from 'react-native';
import {useSession} from "@components/context/AuthContext";
import {Button} from "react-native-paper";

export default function SignIn() {
  const {session, signInAnonymously, signInWithGoogle} = useSession();
  if (session) {
    return <Redirect href={'/'}/>
  }
  return (
    <View style={{flex: 1, justifyContent: 'center', alignItems: 'center', gap:
20}}>
      <Text/>
      <Text onPress={() => {
        signIn();
        // Navigate after signing in. You may want to tweak this to ensure
        // successful before navigating.
        router.replace('/');
      }}>Sign In</Text>
      <Image source={require('@assets/images/testlogo.png')}
style={styles.reactLogo}></Image>
      <Button icon="google" mode="contained" onPress={async () => {
        if (await signInWithGoogle()) {
          router.replace('/');
        }
      }}>Sign in with google</Button>
      <Button mode="contained-tonal" onPress={async () => {
        if (await signInAnonymously()) {
          router.replace('/');
        }
      }}>Anonymously sign in</Button>
    </View>
  );
}

const styles = StyleSheet.create({
  reactLogo: {
    height: 178,
    width: 290,
  },
});
```

useCategories.ts

```
import { useCallback, useEffect, useState } from "react";
import firestore from "@react-native-firebase/firestore";
import { useSession } from "@components/context/AuthContext";
import { Category } from "@types/category";

const COLLECTION = "Categories";

export const useCategories = (withGeneral = true) => {
  const { session } = useSession();
  const [categories, setCategories] = useState<Category[]>([]);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);
  const [refetch, setRefetch] = useState(0);
```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

useEffect(() => {
  const subscriber = firestore()
    .collection(COLLECTION)
    .where("userId", "in", [session.uid, ...(withGeneral ? ["general"] : [])])
    .onSnapshot(documentSnapshot => {
      setCategories(documentSnapshot.docs.map((doc) => ({
        ...doc.data(),
        id: doc.id,
      }) as Category));
    }, (error) => {
      console.log("useCategories", error);
      setError(error);
    }, () => {
      setIsLoading(false);
    });

  // Stop listening for updates when no longer required
  return () => subscriber();
}, [refetch]);

return {
  categories,
  isLoading,
  error,
  refetch: () => setRefetch((prev) => prev + 1),
};
};

export const useCreateCategory = () => {
  const create = useCallback(async (category: Category) => {
    return firestore()
      .collection(COLLECTION)
      .add(category);
  }, []);

  return {
    create,
  };
};

export const useRemoveCategory = () => {
  const remove = useCallback(async (id: string) => {
    await firestore()
      .collection(COLLECTION)
      .doc(id)
      .delete();
  }, []);

  return {
    remove,
  };
};

```

useExpenses.ts

```

import { useCallback, useEffect, useState } from "react";
import firestore from "@react-native-firebase/firestore";
import { useSession } from "@components/context/AuthContext";
import { Expenses } from "@types/expenses";

export const useExpenses = () => {
  const { session } = useSession();
  const [expenses, setExpenses] = useState<Expenses[]>([]);
  const [isLoading, setIsLoading] = useState(true);

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

const [error, setError] = useState<Error | null>(null);
const [refetch, setRefetch] = useState(0);

useEffect(() => {
  const subscriber = firestore()
    .collection("Expenses")
    .where("userId", "==", session.uid)
    .orderBy("date", "desc")
    .onSnapshot(documentSnapshot => {
      setExpenses(documentSnapshot.docs.map((doc) => ({
        ...doc.data(),
        id: doc.id,
      }) as Expenses));
    }, (error) => {
      setError(error);
    }, () => {
      setIsLoading(false);
    });

  // Stop listening for updates when no longer required
  return () => subscriber();
}, [refetch]);

return {
  expenses,
  isLoading,
  error,
  refetch: () => setRefetch((prev) => prev + 1),
};
};

export const useExpense = (id: string) => {
  const { session } = useSession();
  const [expense, setExpense] = useState<Expenses | null>(null);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

  useEffect(() => {
    const subscriber = firestore()
      .collection("Expenses")
      .doc(id === "new" ? undefined : id)
      .onSnapshot(documentSnapshot => {
        setExpense(documentSnapshot.data() as Expenses);
      }, (error) => {
        setError(error);
      }, () => {
        setIsLoading(false);
      });

    // Stop listening for updates when no longer required
    return () => subscriber();
  }, [id]);

  return {
    expense,
    isLoading,
    error,
  };
};

export const useCreateExpense = () => {
  const create = useCallback(async (expense: Expenses) => {
    await firestore()
      .collection("Expenses")

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

        .add(expense);
    }, []);

    return {
        create,
    };
};

export const _testOnlyCreateExpenses = () => {
    const { session } = useSession();
    const create = useCallback(async () => {
        for (const e of _testOnlyExpenses) {
            await firestore()
                .collection("Expenses")
                .add({
                    ...e,
                    userId: session.uid,
                    date: e.date * 1000,
                });
        }
    }, []);

    return {
        create,
    };
};

export const useUpdateExpense = (id: string) => {
    const update = useCallback(async (expense: Expenses) => {
        await firestore()
            .collection("Expenses")
            .doc(id)
            .update(expense);
    }, [id]);

    return {
        update,
    };
};

export const useRemoveExpense = () => {
    const remove = useCallback(async (id: string) => {
        await firestore()
            .collection("Expenses")
            .doc(id)
            .delete();
    }, []);

    return {
        remove,
    };
};

```

useStorageState.ts

```

import * as SecureStore from 'expo-secure-store';
import * as React from 'react';
import { Platform } from 'react-native';

type UseStateHook<T> = [[boolean, T | null], (value: T | null) => void];

function useAsyncState<T>({
    initialValue: [boolean, T | null] = [true, null],
}): UseStateHook<T> {

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

    return React.useReducer(
      (state: [boolean, T | null], action: T | null = null): [boolean, T | null]
    => [false, action],
      initialValue
    ) as UseStateHook<T>;
  }

export async function setStorageItemAsync(key: string, value: string | null) {
  if (Platform.OS === 'web') {
    try {
      if (value === null) {
        localStorage.removeItem(key);
      } else {
        localStorage.setItem(key, value);
      }
    } catch (e) {
      console.error('Local storage is unavailable:', e);
    }
  } else {
    if (value == null) {
      await SecureStore.deleteItemAsync(key);
    } else {
      await SecureStore.setItemAsync(key, value);
    }
  }
}

export function useStorageState(key: string): UseStateHook<string> {
  // Public
  const [state, setState] = useAsyncState<string>();

  // Get
  React.useEffect(() => {
    if (Platform.OS === 'web') {
      try {
        if (typeof localStorage !== 'undefined') {
          setState(localStorage.getItem(key));
        }
      } catch (e) {
        console.error('Local storage is unavailable:', e);
      }
    } else {
      SecureStore.getItemAsync(key).then(value => {
        setState(value);
      });
    }
  }, [key]);

  // Set
  const setValue = React.useCallback(
    (value: string | null) => {
      setState(value);
      setStorageItemAsync(key, value);
    },
    [key]
  );

  return [state, setValue];
}

```

useThemeColor.ts

```

import { useColorScheme } from 'react-native';

import { Colors } from '@constants/Colors';

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

export function useThemeColor(
  props: { light?: string; dark?: string },
  colorName: keyof typeof Colors.light & keyof typeof Colors.dark
) {
  const theme = useColorScheme() ?? 'light';
  const colorFromProps = props[theme];

  if (colorFromProps) {
    return colorFromProps;
  } else {
    return Colors[theme][colorName];
  }
}

```

useTimeRecords.ts

```

import { useCallback, useEffect, useState } from "react";
import firestore from "@react-native-firebase/firestore";
import { useSession } from "@components/context/AuthContext";
import { TimeRecords } from "@types/time-records";

const COLLECTION = "TimeRecords";

export const useTimeRecords = () => {
  const { session } = useSession();
  const [timeRecords, setTimeRecords] = useState<TimeRecords[]>([]);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);
  const [refetch, setRefetch] = useState(0);

  useEffect(() => {
    const subscriber = firestore()
      .collection(COLLECTION)
      .where("userId", "==", session.uid)
      .orderBy("dateFrom", "desc")
      .onSnapshot(documentSnapshot => {
        setTimeRecords(documentSnapshot.docs.map((doc) => ({
          ...doc.data(),
          id: doc.id,
        }) as TimeRecords));
      }, (error) => {
        console.log("useTimeRecords", error);
        setError(error);
      }, () => {
        setIsLoading(false);
      });

    // Stop listening for updates when no longer required
    return () => subscriber();
  }, [refetch]);

  return {
    timeRecords,
    isLoading,
    error,
    refetch: () => setRefetch((prev) => prev + 1),
  };
};

export const useTimeRecord = (id: string) => {
  const { session } = useSession();
  const [timeRecord, setTimeRecord] = useState<TimeRecords | null>(null);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

useEffect(() => {
  const subscriber = firestore()
    .collection(COLLECTION)
    .doc(id === "new" ? undefined : id)
    .onSnapshot(documentSnapshot => {
      setTimeRecord(documentSnapshot.data() as TimeRecords);
    }, (error) => {
      setError(error);
    }, () => {
      setIsLoading(false);
    });

  // Stop listening for updates when no longer required
  return () => subscriber();
}, [id]);

return {
  timeRecord,
  isLoading,
  error,
};
};

export const useCreateTimeRecord = () => {
  const create = useCallback(async (timeRecords: TimeRecords) => {
    await firestore()
      .collection(COLLECTION)
      .add(timeRecords);
  }, []);

  return {
    create,
  };
};

export const useUpdateTimeRecord = (id: string) => {
  const update = useCallback(async (timeRecords: TimeRecords) => {
    await firestore()
      .collection(COLLECTION)
      .doc(id)
      .update(timeRecords);
  }, [id]);

  return {
    update,
  };
};

export const useRemoveTimeRecord = () => {
  const remove = useCallback(async (id: string) => {
    await firestore()
      .collection(COLLECTION)
      .doc(id)
      .delete();
  }, []);

  return {
    remove,
  };
};

```

[record].tsx

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

import { useSafeAreaInsets } from "react-native-safe-area-context";
import { useLocalSearchParams, useNavigation } from "expo-router";
import { Appbar, Button, TextInput, useTheme } from "react-native-paper";
import { getFlatInputColors } from "react-native-
paper/src/components/TextInput/helpers";
import { useSession } from "@components/context/AuthContext";
import { useFormik } from "formik";
import * as Yup from "yup";
import { useEffect, useState } from "react";
import { StyleSheet, Text, View } from "react-native";
import { DateTimePicker } from "@components/form/DateTimePicker";
import { useCreateTimeRecord, useTimeRecord, useUpdateTimeRecord } from
"@hooks/useTimeRecords";
import { TimeRecords } from "@types/time-records";
import { Select } from "@components/form/Select";

const defaultTimeRecord: TimeRecords = {
  id: "",
  category: "",
  description: "",
  name: "",
  userId: "",
  dateFrom: new Date().getTime(),
  dateTo: new Date().getTime(),
  timeConsumed: 0,
};

export default function TimeRecord() {
  const insets = useSafeAreaInsets();
  const navigation = useNavigation();
  const theme = useTheme();
  const colors = getFlatInputColors({
    theme,
  });

  const { record: recordId } = useLocalSearchParams<{ record: string }>();
  const mode = recordId === "new" ? "create" : "edit";
  const { session } = useSession();

  const { timeRecord, error } = useTimeRecord(recordId as string);

  const { submitForm, values, setFieldValue, setValues, errors } =
useFormik<TimeRecords>({
  initialValues: {
    ...defaultTimeRecord,
    userId: session.uid,
  },
  validateOnChange: false,
  validateOnBlur: false,
  validateOnMount: false,
  validationSchema: Yup.object().shape({
    name: Yup.string().max(55).required(),
    description: Yup.string().max(255),
    category: Yup.string().required(),
    dateFrom: Yup.number().required(),
    dateTo: Yup.number().required(),
  }),
  onSubmit: async (values) => {
    await handleActionButton({
      ...values,
      userId: session.uid,
      timeConsumed: values.dateTo - values.dateFrom,
    });
  },
});

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

useEffect(() => {
  setValues(timeRecord || defaultTimeRecord);
}, [timeRecord]);

const createAction = useCreateTimeRecord();
const updateAction = useUpdateTimeRecord(recordId as string);
const [processingAction, setProcessingAction] = useState(false);
const handleActionButton = async (submitValues: TimeRecords) => {
  setProcessingAction(true);

  try {
    if (mode === "create") {
      await createAction.create(submitValues);
    } else {
      await updateAction.update(submitValues);
    }

    navigation.goBack();
  } catch (e) {

  } finally {
    setProcessingAction(false);
  }
};

const handleDateChange = (date: Date, field: keyof TimeRecords) => {
  setFieldValue(field, date.getTime());
};

return (
  <View style={{
    ...styles.container,
    backgroundColor: theme.colors.background,
  }}>
    <View>
      <AppBar.Header>
        <AppBar.BackAction onPress={() => {
          navigation.goBack();
        }} />
        <AppBar.Content title={mode === "create" ? "New Time Record" : "Edit
Time Record"} />
      </AppBar.Header>
      <TextInput
        label="Name"
        error={!!errors?.name}
        value={values.name}
        onChangeText={(val) => {
          setFieldValue("name", val);
        }}
      />
      <TextInput
        label="Description"
        error={!!errors?.description}
        value={values.description}
        onChangeText={(val) => {
          setFieldValue("description", val);
        }}
      />
      <Select value={values.category} options={[]} onChange={(value) => {
        setFieldValue("category", value);
      }} />
    </View>
  </View>
);

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

<DateTimePicker value={values.dateFrom} onChange={(date) => {
  handleDateChange(date, "dateFrom");
}} error={errors.dateFrom} />

<DateTimePicker value={values.dateTo} onChange={(date) => {
  handleDateChange(date, "dateTo");
}} error={errors.dateTo} />

{Object.keys(errors).length > 0 && (
  <Text>
    {JSON.stringify(errors)}
  </Text>
)}
</View>

<Button mode="contained" loading={processingAction} style={{
  borderRadius: 0,
  marginBottom: insets.bottom,
}} onPress={() => {
  void submitForm();
}}>{mode === "create" ? "Create" : "Update"}</Button>
</View>
);
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "space-between",
  },
  halfSplitContainer: {
    flexDirection: "row",
    justifyContent: "space-between",
  },
});

```

[expense].tsx

```

import { AppBar, Button, TextInput, useTheme } from "react-native-paper";
import { useLocalSearchParams, useNavigation } from "expo-router";
import { StyleSheet, Text, View, Image } from "react-native";
import { useCreateExpense, useExpense, useUpdateExpense } from
"@/hooks/useExpenses";
import { useEffect, useRef, useState } from "react";
import { Expenses } from "@/types/expenses";
import { useSession } from "@/components/context/AuthContext";
import { useSafeAreaInsets } from "react-native-safe-area-context";
import { useFormik } from "formik";
import * as Yup from "yup";
import { DateTimePicker } from "@/components/form/DateTimePicker";
import { Select } from "@/components/form/Select";
import { CreateCategoryModal } from "@/components/modal/CreateCategoryModal";
import { useCategories, useCreateCategory } from "@/hooks/useCategories";
import { Category } from "@/types/category";
import * as ImagePicker from "expo-image-picker";
import { FontAwesome6 } from "@expo/vector-icons";
import { AI } from "@/classes/AI";

const defaultExpense: Expenses = {
  id: "",
  date: new Date().getTime(),
  amount: 0,
  category: "",
  currency: "",
  description: "",

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

    name: "",
    userId: "",
  };

export default function Expense() {
  const insets = useSafeAreaInsets();
  const navigation = useNavigation();
  const theme = useTheme();
  const { expense: expenseId } = useLocalSearchParams<{ expense: string }>();
  const mode = expenseId === "new" ? "create" : "edit";
  const { session } = useSession();

  const { expense, error } = useExpense(expenseId as string);

  const { categories } = useCategories();

  const [processingAI, setProcessingAI] = useState(false);

  const { submitForm, values, setFieldValue, setValues, errors } =
useFormik<Expenses>({
  initialValues: {
    ...defaultExpense,
    userId: session.uid,
  },
  validateOnChange: false,
  validateOnBlur: false,
  validateOnMount: false,
  validationSchema: Yup.object().shape({
    name: Yup.string().max(55).required(),
    description: Yup.string().max(255),
    category: Yup.string().required(),
    amount: Yup.number().required(),
    date: Yup.number().required(),
  }),
  onSubmit: async (values) => {
    await handleActionButton({
      ...values,
      category: categories.find((c) => c.id === values.category)?.name ||
values.category,
      userId: session.uid,
    });
  },
});

  useEffect(() => {
    if (expense) {
      setValues({
        ...expense,
        category: categories.find((c) => c.name === expense.category)?.id ||
expense.category,
      });
    } else {
      setValues(defaultExpense);
    }
  }, [expense, categories]);

  const createAction = useCreateExpense();
  const updateAction = useUpdateExpense(expenseId as string);
  const [processingAction, setProcessingAction] = useState(false);
  const handleActionButton = async (submitValues: Expenses) => {
    setProcessingAction(true);

    try {
      if (mode === "create") {

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        await createAction.create(submitValues);
    } else {
        await updateAction.update(submitValues);
    }

    navigation.goBack();
} catch (e) {

} finally {
    setProcessingAction(false);
}
};

const handleDateChange = (date: Date) => {
    setFieldValue("date", date.getTime());
};

const createCategoryRef = useRef<{
    showModal: () => void;
    hideModal: () => void;
    promise: Promise<Category> | null;
}>(null);

const createCategoryAction = useCreateCategory();

const pickImage = async () => {
    // No permissions request is necessary for launching the image library
    let result = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.All,
        allowsEditing: false,
        quality: 0.7,
        base64: true,
    });

    if (!result.canceled) {
        try {
            setProcessingAI(true);
            const recognition = await
AI.getInstance().getMoneyRecordRecognition(`data:image/jpeg;base64,${result.assets[0].base64}`);

            console.log("recognition", recognition);

            if (typeof recognition !== "string") {
                throw new Error("Recognition is not a string");
            }
            const parsed = JSON.parse(recognition);

            let category = categories.find((c) => c.name === parsed.category);

            if (category === undefined) {
                const res = await createCategoryAction.create({
                    name: parsed.category,
                    userId: session.uid,
                    order: 0,
                    description: "",
                });
                const data = await res.get();
                category = {
                    ...data.data(),
                    id: data.id,
                } as Category;
            }

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

        void delayedSetValues(parsed, category.id as string);

        console.log("parsed", parsed);
    } catch (e) {
        console.log(e);
    } finally {
        setProcessingAI(false);
    }
}
};

const delayedSetValues = async (values: Expenses, categoryId: string) => {
    // await new Promise((resolve) => {
    //     setTimeout(resolve, 1000);
    // });

    await setValues({
        ...values,
        category: categoryId || "",
        date: values.date * 1000,
        userId: session.uid,
    })
}

return (
    <View style={{
        ...styles.container,
        backgroundColor: theme.colors.background,
    }}>
        <View>
            <AppBar.Header>
                <AppBar.BackAction onPress={() => {
                    navigation.goBack();
                }} />
                <AppBar.Content title={mode === "create" ? "New Expense" : "Edit
expense"} />
            </AppBar.Header>
            <TextInput
                label="Name"
                error={!errors?.name}
                value={values.name}
                onChangeText={(val) => {
                    setFieldValue("name", val);
                }}
            />
            <TextInput
                label="Description"
                error={!errors?.description}
                value={values.description}
                onChangeText={(val) => {
                    setFieldValue("description", val);
                }}
            />

            <View style={{
                ...styles.halfSplitContainer,
            }}>
                <View style={{
                    flex: 1,
                }}>
                    <Select value={values.category} options={categories.map((c) => ({
                        label: c.name as string,
                        value: c.id as string,
                    }))} onChange={(value) => {

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        setFieldValue("category", value);
    }} />
</View>
<Button mode={"contained"} style={{
    borderRadius: 0,
    padding: 5,
}} onPress={async () => {
    createCategoryRef.current?.showModal();
    try {
        const category = await createCategoryRef.current?.promise;
        if (category) {
            setFieldValue("category", category.id);
        }
    } catch (e) {
        console.log(e);
    }
}}>New</Button>
</View>

<TextInput
    label="Amount"
    value={values.amount.toString()}
    keyboardType={"decimal-pad"}
    onChangeText={(val) => {
        setFieldValue("amount", val ? parseFloat(val) : 0);
    }}
/>

<DateTimePicker value={values.date} onChange={handleDateChange}
error={errors.date} />

<Button
    loading={processingAI}
    style={{
        marginTop: 8,
    }}
    icon={({ size, color }) => (
        <FontAwesome6 name="wand-magic-sparkles" size={size} color={color}
/>
    )}
    onPress={() => {
        void pickImage();
    }}
>Recognize with AI</Button>

{Object.keys(errors).length > 0 && (
    <Text>
        {JSON.stringify(errors)}
    </Text>
)}
</View>

<Button mode="contained" loading={processingAction} style={{
    borderRadius: 0,
    marginBottom: insets.bottom,
}} onPress={() => {
    void submitForm();
}}>{mode === "create" ? "Create" : "Update"}</Button>

<CreateCategoryModal ref={createCategoryRef} />
</View>
);
}

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "space-between",
  },
  halfSplitContainer: {
    flexDirection: "row",
    justifyContent: "space-between",
  },
});
```

settings.tsx

```
import { Appearance, ColorSchemeName, View } from "react-native";
import { AppBar, Avatar, Button, Divider, Switch, Text } from "react-native-paper";
import { MaterialIcons } from "@expo/vector-icons";
import { useSession } from "@components/context/AuthContext";
import { useStorageState } from "@hooks/useStorageState";
import { setTheme } = Appearance;
import { router } from "expo-router";

type SyncWithDevice = "sync" | "no-sync";

export default function Settings() {
  const { session, isLoading, signOut } = useSession();
  const [[, deviceTheme], setDeviceTheme] = useStorageState("theme") as unknown as [[boolean, ColorSchemeName], (value: string | null) => void];
  const [[, sync], setSync] = useStorageState("syncWithDevice") as unknown as [[boolean, SyncWithDevice], (value: SyncWithDevice) => void];

  return (
    <>
      <AppBar.Header elevated={true}>
        <AppBar.Content title="Settings" />
        <AppBar.Action icon={({ size, color }) => (
          <MaterialIcons name="logout" size={size} color={color} />
        )} onPress={() => {
          void signOut();
        }} />
      </AppBar.Header>

      <View style={{
        display: "flex",
        gap: 8,
        padding: 16,
      }}>
        <Text variant="displaySmall">Account</Text>
        <View style={{
          display: "flex",
          flexDirection: "row",
          alignItems: "center",
          gap: 8,
        }}>
          <Avatar.Image source={{ uri: session.photoURL }} />
          <Text>{session.email || "Anonymous"}</Text>
        </View>

        <Divider style={{
          marginVertical: 16,
        }} />

        <Text variant="displaySmall" style={{ marginBottom: 16 }}>Settings</Text>
        <View style={{
```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        flexDirection: "row",
        gap: 8,
        alignItems: "center",
        marginBottom: 8,
    }}>
    <Switch
      value={deviceTheme === "dark"}
      onChange={() => {
        const newMode = deviceTheme === "dark" ? "light" : "dark";
        setDeviceTheme(newMode);
        setColorScheme(newMode);
        setSync("no-sync");
      }} />
    <Text>Dark mode</Text>
  </View>

  <View style={{
    flexDirection: "row",
    gap: 8,
    alignItems: "center",
  }}><Switch value={sync === "sync"} onChange={() => {
    setSync(sync === "sync" ? "no-sync" : "sync");
    setDeviceTheme(null);
    setColorScheme(null);
  }} /><Text>Sync with device theme</Text></View>

  <Divider style={{
    marginVertical: 16,
  }} />

  <Button onPress={() => {
    router.navigate("/categories");
  }}>Manage categories</Button>

</View>
</>
);
}

```

statistics.tsx

```

import { AppBar, useTheme } from "react-native-paper";
import { MaterialIcons } from "@expo/vector-icons";
import { processColor, Text, View } from "react-native";
import { useExpenses } from "@/hooks/useExpenses";
import { StyleSheet } from "react-native";

import { LineChart, PieChart } from "react-native-charts-wrapper";
import { Select } from "@components/form/Select";
import { useState } from "react";
import { format } from "date-fns";

export function Statistics() {
  const { expenses } = useExpenses();
  const theme = useTheme();
  const [chartType, setChartType] = useState<"line" | "pie">("line");
  return (
    <>
      <AppBar.Header elevated={true}>
        <AppBar.Content title="Statistics" />
      </AppBar.Header>
      <View>
        <Select value={chartType} options={[

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

        { label: "Pie chart", value: "pie" },
        { label: "Line chart", value: "line" },
    ]] onChange={ (val) => {
        setChartType(val as "line" | "pie");
    }} />

</View>
<View style={styles.container}>
    {chartType === "pie" && (
        <PieChart style={styles.chart}
            data={{
                dataSets: [
                    {
                        label: "Expenses",
                        values: expenses.map((expense) => ({
                            value: expense.amount,
                            label: expense.name,
                        })),
                        config: {
                            colors: [processColor("#C0FF8C"),
processColor("#FFF78C"), processColor("#FFD08C"), processColor("#8CEAFF"),
processColor("#FF8C9D")],
                            valueTextSize: 15,
                            valueTextColor: processColor("green"),
                            sliceSpace: 5,
                            selectionShift: 13,
                            xValuePosition: "OUTSIDE_SLICE",
                            yValuePosition: "OUTSIDE_SLICE",
                            valueFormatter: "#.#'@"',
                            valueLineColor: processColor("green"),
                            valueLinePart1Length: 0.7,
                            valueLinePart2Length: 0.8,
                        },
                    },
                ],
            }}
            legend={{
                enabled: true,
                textSize: 15,
                horizontalAlignment: "CENTER",
                verticalAlignment: "BOTTOM",
                orientation: "HORIZONTAL",
                wordWrapEnabled: true,
            }}
            extraOffsets={{ left: 5, top: 5, right: 5, bottom: 5 }}
            holeRadius={50}
            drawEntryLabels={false}
        />
    )}

    {chartType === "line" && (
        <LineChart style={{
            ...styles.chart,
            flex: 0.8,
        }}
            pinchZoom={true}
            doubleTapToZoomEnabled={true}
            data={{
                dataSets: [
                    {
                        label: "Expenses",
                        values: expenses.map((expenses, i) => ({
                            y: expenses.amount,

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        x: i,
        marker: expenses.name,
      )),
      config: {
        lineWidth: 2,
        valueTextSize: 10,
        valueFormatter: expenses.map((expense) =>
`${expense.amount}€\n(${expense.name}) \n`),
        drawValues: true,
        drawCircles: true,
        // highlightColor: processColor('red'),
        // color: processColor('red'),
        drawFilled: true,
        // fillColor: processColor('blue'),
        // fillAlpha: 60,
        highlightEnabled: true,
        dashedLine: {
          lineLength: 20,
          spaceLength: 5,
        },
      },
    },
  ],
})
legend={{
  enabled: true,
  textSize: 15,
  horizontalAlignment: "LEFT",
  verticalAlignment: "BOTTOM",
  orientation: "HORIZONTAL",
  wordWrapEnabled: true,
}}
extraOffsets={{ bottom: 10}}
xAxis={{
  valueFormatter: expenses.map((expense) =>
`${format(expense.date, "d MMM")} \n ${format(expense.date, "HH:mm")}`),

  // granularityEnabled: true,
  // granularity: 1,
  position: "BOTTOM",
  // drawGridLines: false,
  drawAxisLine: true,
  // drawLabels: true,
  // avoidFirstLastClipping: true,
  // axisMaximum: expenses.length,
  // axisMinimum: 0,
}}

  />
  )}

  </View>
</>
);
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  chart: {
    flex: 0.5,
    marginTop: 50,
  },
});

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

AI.ts

```
import OpenAI from "openai";
import { ChatCompletionContentPartImage } from
"openai/src/resources/chat/completions";

const openai = new OpenAI({
  apiKey: process.env.EXPO_PUBLIC_OPEN_AI_API_KEY,
});

export class AI {
  public static instance: AI;

  protected constructor() {}

  public static getInstance(): AI {
    if (!AI.instance) {
      AI.instance = new AI();
    }
    return AI.instance;
  }

  public async getMoneyRecordRecognition(imageUrl: string): Promise<string |
null> {
    const image: ChatCompletionContentPartImage = {
      image_url: {
        url: imageUrl,
      },
      type: "image_url",
    };

    const response = await openai.chat.completions.create({
      model: "gpt-4o",
      messages: [
        {
          "role": "system",
          "content": "Imagine you are a system for recognizing money expenses.
Dates are always in the UNIX format, date timezone in always Europe/Kyiv. People
going to send an image to you and you should always respond with only valid JSON
and nothing else!\nNote provide currency in the format like: \"UAH\", \"USD\",
\"EUR\", etc.\nJSON structure to create:\nexport interface Expenses { name:
string; description: string; category: string; date: number; amount: number;
currency: string; }"
        },
        {
          "role": "user",
          "content": [
            image,
          ]
        }
      ],
      temperature: 1,
      max_tokens: 256,
      top_p: 1,
      frequency_penalty: 0,
      presence_penalty: 0,
    });

    if (!response.choices || !response.choices[0].message ||
```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ПЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```
!response.choices[0].message.content) {
  throw new Error("Invalid response from OpenAI");
}

return response.choices[0].message.content.replace(/``json\n|\n``/g, "");
}
}
```

AuthContext.tsx

```
import React from "react";
import { useStorageState } from "@hooks/useStorageState";
import auth from "@react-native-firebase/auth";
import { GoogleSignin } from "@react-native-google-signin/google-signin";
import { Session } from "@types/session";

const AuthContext = React.createContext<{
  signIn: () => void;
  signInAnonymously: () => Promise<boolean>;
  signInWithGoogle: () => Promise<boolean>;
  signOut: () => Promise<boolean>;
  session: Session;
  isLoading: boolean;
}>({
  signIn: () => null,
  signInAnonymously: () => Promise.resolve(false),
  signInWithGoogle: () => Promise.resolve(false),
  signOut: () => Promise.resolve(false),
  session: {} as Session,
  isLoading: false,
});

// This hook can be used to access the user info.
export function useSession() {
  const value = React.useContext(AuthContext);
  if (process.env.NODE_ENV !== "production") {
    if (!value) {
      throw new Error("useSession must be wrapped in a <SessionProvider />");
    }
  }

  return value;
}

export function SessionProvider(props: React.PropsWithChildren) {
  const [[isLoading, session], setSession] = useStorageState("session");

  return (
    <AuthContext.Provider
      value={{
        signIn: () => {
          // Perform sign-in logic here
          setSession("xxx");
        },
        signInWithGoogle: async () => {
          try {
            // Check if your device supports Google Play
            await GoogleSignin.hasPlayServices({ showPlayServicesUpdateDialog:
true });
            // Get the users ID token
            const { idToken } = await GoogleSignin.signIn();

            // Create a Google credential with the token
            const googleCredential =
```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

    auth.GoogleAuthProvider.credential(idToken);

    const res = await auth().signInWithCredential(googleCredential);

    console.log("successfully signed in with google", res.user,
res.additionalUserInfo);
    setSession(JSON.stringify(res.user));
    // Sign-in the user with the credential
  } catch (e) {
    console.error("ERROR while sign-in with google", e);
    return false;
  }

  return true;
},
signInAnonymously: async () => {
  const res = await auth().signInAnonymously();
  console.log("successfully signed in anonymously", res.user,
res.additionalUserInfo);
  setSession(JSON.stringify(res.user));
  return true;
},
signOut: async () => {
  await auth().signOut();
  setSession(null);
  return true;
},
session: session ? JSON.parse(session) : null,
isLoading,
}}>
{props.children}
</AuthContext.Provider>
);
}

```

		Кравчук В.О.			ІПЗ.КР.Б-121-24-ІЗ	Арк.
		Мінтій І.С.				84
Змн.	Арк.	№ докум.	Підпис	Дата		