Вінницький Національний Технічний Університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка мобільного застосунку аналізу грошових витрат у системі Google Pay»

Виконав: студент IV курсу, групи 3ПІ-20б

спеціальності

121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Трайтека Нікіта Олексійович
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Рейда О. М.
(прізвище та ініціали)

Рецензент: д.т.н., професор Мартинюк Т.Б.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри _____ О. Н. Романюк

«10» 06 2024 р.

ВНТУ – 2024

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
О.Н. Романюк _____
«12» березня 2024 р.

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

Трайтеці Нікіті Олексійовичу

1. Тема роботи – розробка мобільного застосунку аналізу грошових витрат у системі Google Pay.

Керівник роботи: Рейда Олександр Миколайович д.т.н., доцент, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи Моделі управління фінансовими даними включають алгоритми обробки доходів та витрат, що дозволяють користувачам додавати, редагувати та видаляти фінансові записи. Для цього використовуються структури даних для зберігання інформації про транзакції, такі як дата, назва, сума, категорія та тип (доходи чи витрати).

4. Зміст текстової частини: вступ; аналіз стану питань та обґрунтування питань розробки; розробка моделей та алгоритмів програмного продукту, розробка програмних модулів; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина

5. Перелік ілюстративного матеріалу:. титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; новизна і практична цінність одержаних результатів; порівняльний аналіз аналогів; методи та засоби розробки, модель роботи системи, структурна схема, таблиці баз даних, алгоритми роботи системи, інтерфейс додатка, вікна додатку, таблиці зі списками витрат, апробації та публікації, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Рейда О. М., к.т.н., доцент каф ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р

КАЛЕНДАРНИЙ ПЛАН

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку застосунків для моніторингу фінансових витрат	13.03.24 – 22.03.24	Виконав
2	Розробка методів аналізу грошових витрат	13.03.24 – 22.03.24	Виконав
3	Розробка алгоритмів роботи програмного забезпечення	25.03.24 – 19.04.24	Виконав
4	Розробка програмного забезпечення	22.04.24 – 10.05.24	Виконав
5	Тестування програмного забезпечення	13.05.24 – 24.05.24	Виконав
6	Оформлення матеріалів до захисту БДР	27.05.24 – 03.06.24	Виконав

Студент _____ **Н. О. Трайтека**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **О. М. Рейда**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 55 сторінок формату А4, на яких є 19 рисунків, 3 таблиці, список використаних джерел містить 22 найменування.

У бакалаврській дипломній роботі розроблено мобільний застосунок аналізу грошових витрат у системі Google Pay. Застосунок призначений для ефективного управління фінансами користувача та надає можливість моніторингу особистих фінансових операцій. Розроблені програмні модулі дозволяють користувачам вести облік витрат, доходів, виконувати аналіз фінансової діяльності та планувати бюджет.

Застосунок розроблено з використанням середовища програмування Visual Studio Code, що забезпечує легкість та швидкість розробки. Для реалізації інтерфейсу користувача використовується JavaScript та додаткові бібліотеки до неї.

Основні функціональні можливості додатку включають управління рахунками, категоріями витрат та доходів, аналітику фінансової активності та створення фінансових цілей. Застосунок включає в себе графічний інтерфейс, що допомагає ефективно вести контроль фінансів та досягати поставлених цілей.

Ключові слова: мобільний додаток, фінанси, продуктивність, моніторинг фінансів.

ABSTRACT

The bachelor thesis consists of 55 pages of A4 format, on which there are 19 pictures, 3 tables, the list of used sources contains 22 names.

In the bachelor's thesis, a mobile application for the analysis of money expenses in the Google Pay system was developed. The application is designed to effectively manage the user's finances and provides the ability to monitor personal financial transactions. The developed software modules allow users to keep track of expenses, income, analyze financial activities and plan a budget.

The application is developed using the Visual Studio Code programming environment, which ensures ease and speed of development. JavaScript and additional libraries are used to implement the user interface.

The main functionality of the application includes management of accounts, categories of expenses and income, analytics of financial activity and creation of financial goals. The application includes a graphical interface, which helps to effectively manage finances and achieve set goals.

Keywords: mobile application, finance, productivity, financial monitoring.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасні методи управління особистими фінансами для забезпечення фінансової стабільності враховують високу швидкість життя та зайнятість людей, що значно впливає на можливість ефективного відстеження витрат і обмежує можливості планування бюджету.

Для забезпечення фінансової стабільності використовують мобільні додатки для управління та моніторингу фінансів, що дозволяє відстежувати фінансові операції та аналізувати фінансовий стан. Як результат, це ускладнює контроль за витратами та призводить до зниження продуктивності управління фінансами.

Таким чином, існуючі методи управління особистими фінансами характеризуються суттєвою обчислювальною складністю, що в значній мірі впливає на ефективність досягнення фінансової стабільності. Тому актуальними є питання підвищення продуктивності методів і засобів фінансового управління.

Для існуючих методів управління фінансами характерні складність і різноманітність фінансових операцій, що обмежує можливості користувачів у досягненні фінансової стабільності.

Підвищення продуктивності методів управління особистими фінансами дасть можливість забезпечити краще відстеження витрат і планування бюджету, тому важливими є питання розробки відповідних методів управління фінансами.

При формуванні мобільних додатків для управління фінансами використовуються методи, що не автоматизовані системи розрахунків або автоматизовані системи витрат для контролю витрат коштів на рахунках. Тому важливими є питання розробки комбінованих методів управління фінансами.

Тому актуальними є питання підвищення продуктивності та ефективності (точності, швидкодії) мобільних додатків для управління фінансами, оскільки існуючі методи не задовольняють потреби користувачів в ефективному фінансовому управлінні, що передбачає розробку нових методів і засобів.

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

Мета та завдання дослідження. Метою роботи є оптимізація управління особистими фінансами за рахунок використання системи безконтактного способу оплати Google Pay для автоматичного управління витратами і надходженнями.

Основним завданням проекту є:

- провести аналіз існуючих мобільних застосунків і систем для аналізу грошових витрат;
- розробити метод контролю грошових витрат і надходжень користувача;
- розробити базу данних застосунка[1];
- розробити алгоритми аналізу грошових витрат;
- провести тестування та оцінку ефективності розробленої мобільної системи в реальних умовах.

Об’єкт дослідження – процес розробки мобільного додатку для аналізу грошових витрат.

Предмет дослідження – методи та засоби розробки мобільного додатку для аналізу грошових витрат.

Методи дослідження. У процесі досліджень використовувались:

- теорія чисел та чисельних методів
- комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень;
- теорія баз даних; UML-діаграми; прототипування; методи оптимізації даних;
- методи тестування та оцінки ефективності системи в реальних умовах для аналізу зручності використання та надійності управління фінансовими даними.

Новизна отриманих результатів.

Подальшого розвитку набув метод контролю грошових витрат і надходжень, який надає можливість автоматично визначати рух коштів на рахунках за допомогою системи безконтактного способу оплати Google Pay, що дозволило оптимізувати процес витрат, стабілізувати рівень залишків на рахунках, і як наслідок покращити фінансовий стан користувача.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в можливості використання розробленої мобільної системи для моніторингу фінансових витрат та надходжень.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованій роботі[2], автору належать такі результати: порівняльний аналіз аналогів й алгоритм роботи мобільної системи аналізу грошових витрат, розробка моделі та інтерфейсу мобільного додатку .

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської дипломної роботи доповідалися на конференції: «Молодь в науці: дослідження, проблеми, перспективи (МН-2024) підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024.

Публікації. Результати дослідження були опубліковані в матеріалах конференції [2].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 24 найменувань, 4 додатків. Робота містить 19 ілюстрації та 3 таблиці.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ОБГРУНТУВАННЯ ЗАДАЧ РОЗРОБКИ

1.1 Аналіз стану систем для аналізу грошових витрат

У сучасному світі фінансова грамотність[3] та ефективне управління особистими фінансами стають все більш важливими. Перед кожною людиною постає завдання не лише заробити гроші, але й ефективно ними керувати, планувати бюджет, вести облік витрат та доходів. У цьому контексті мобільні додатки для управління та моніторингу особистих фінансів стають невід'ємною частиною повсякденного життя.

На сьогоднішній день існує велика кількість мобільних додатків для фінансового управління, які надають різноманітні функції: від ведення бюджету та витрат до аналізу інвестицій та планування майбутніх витрат. Проте, багато з цих додатків мають обмежений функціонал, не завжди ідеально підходять для індивідуальних потреб користувача та можуть бути складними у використанні.

Потреба в розробці нового мобільного додатку для управління особистими фінансами зумовлена необхідністю в створенні зручного, ефективного та високофункціонального інструменту, який задовольняв би потреби різних категорій користувачів. Такий додаток має забезпечувати можливість швидкого та зручного ведення обліку фінансів, аналізу витрат, планування бюджету, контролю над доходами та витратами, а також надавати корисні фінансові поради та рекомендації.

Однією з ключових переваг, яку надає мобільний додаток для управління фінансами, є його зручність та висока доступність. Сучасний ритм життя ставить під сумнів ефективність традиційних методів ведення фінансового обліку через їх обмежену мобільність та універсальність. У таких умовах мобільний додаток стає невід'ємним інструментом для керування фінансами, оскільки він дозволяє користувачам вносити та відстежувати свої витрати навіть у русі, пристосовуючись до їхнього активного та мобільного способу життя.

Гнучкість та мобільність додатка для управління фінансами стає особливо важливою в контексті сучасних реалій, коли люди постійно перебувають в русі та мають обмежений доступ до комп'ютерів чи інших традиційних засобів управління фінансами. Завдяки мобільному додатку, користувачі можуть легко та зручно вносити дані про свої фінансові транзакції, контролювати витрати та аналізувати свою фінансову ситуацію у будь-який час та в будь-якому місці. Такий підхід дозволяє забезпечити ефективне управління фінансами навіть в умовах постійної зайнятості та рухливого образу життя.

Крім того, однією з ключових переваг мобільних додатків є їхня можливість автоматизувати і спрощувати багато фінансових процесів. Наприклад, такий додаток може автоматично відстежувати витрати за різними категоріями, проводити аналіз цих даних та надавати користувачеві зрозумілі звіти і рекомендації щодо оптимізації особистих фінансів. Це значно зменшує необхідність вручну вводити дані і дозволяє користувачам ефективніше контролювати свої фінанси, покращуючи їх фінансову грамотність та знижуючи ризики фінансових проблем.

Також важливою функцією мобільного додатку є забезпечення надійності та безпеки фінансових даних користувача. При розробці програмного забезпечення необхідно надавати особливу увагу захисту особистої інформації та фінансових даних від несанкціонованого доступу. Для цього використовуються сучасні методи шифрування, що забезпечують захист інформації під час її передачі та зберігання[4]. Крім того, впровадження двофакторної аутентифікації та інших передових заходів безпеки[5] дозволяє знижувати ризики втрати конфіденційності даних і небажаних фінансових операцій. Завдяки цим заходам користувачі можуть бути впевнені у захищеності своїх фінансових даних під час використання мобільного додатку.

У майбутньому розвиток мобільних додатків для управління фінансами обіцяє включати широке застосування штучного інтелекту та машинного навчання. Ці технології дозволять проводити глибокий аналіз фінансових даних користувачів і надавати їм персоналізовані поради та рекомендації. Наприклад,

алгоритми штучного інтелекту зможуть автоматично виявляти тенденції в особистих витратах і доходах, аналізувати історичні дані та прогнозувати майбутні фінансові показники. Такий підхід дозволить користувачам не лише краще розуміти свій фінансовий стан, але й зробити більш обдумані рішення щодо збереження, інвестування та планування бюджету. Покращення якості обслуговування і зручності користування стануть важливими аспектами, що сприятимуть популяризації таких інноваційних додатків серед широкого кола користувачів.

Підводячи підсумок, можна зазначити, що на сьогоднішній день існує настійна потреба у створенні нового мобільного додатку для ефективного управління та моніторингу особистих фінансів. Такий додаток повинен поєднувати в собі не лише зручний інтерфейс, але й високий рівень функціональності та надійність у роботі. Розробка такого програмного забезпечення має значний потенціал у вирішенні потреб користувачів і може стати важливим інструментом для досягнення фінансової стабільності, ефективного управління фінансами та особистого успіху. Правильно спроектований інструмент забезпечить не лише зручність в керуванні фінансами, але й підвищить фінансову грамотність користувачів, що сприятиме їхньому особистому і професійному розвитку.

1.2 Порівняльний аналіз аналогів

Порівняльний аналіз існуючих аналогів у сфері розробки мобільних додатків для управління та моніторингу особистих фінансів відіграє ключову роль у визначенні конкурентних переваг та унікальних особливостей запропонованого продукту. Цей аналіз дозволяє з'ясувати сильні та слабкі сторони існуючих рішень на ринку, виявити недоліки, які можна виправити у новому додатку, а також знайти можливості для вдосконалення. Враховуючи відгуки користувачів та аналітичні дані, можна зрозуміти, які саме функції та можливості є наразі недостатніми чи найбільш важливими для цільової

аудиторії. Такий підхід дозволяє розробникам не лише створити конкурентоздатний продукт, а й забезпечити його відповідність сучасним вимогам і очікуванням користувачів.

Деякі існуючі додатки, такі як Mint, Personal Capital, YNAB (You Need a Budget) PocketGuard та Monefy, вже визнані на ринку і користуються популярністю серед користувачів. Вони пропонують широкий спектр функцій, включаючи відслідковування витрат, створення бюджету, аналіз фінансового стану та навіть інвестиційні поради. Однак, вони можуть мати обмеження, такі як складність використання, обмежені можливості аналізу, або відсутність інтеграції з певними фінансовими установами.

Під час розробки нового додатку варто уникати простого копіювання існуючих рішень, а замість цього зосередитися на вдосконаленні та інноваціях. Наприклад, можливість використання штучного інтелекту для аналізу фінансових звітів користувача та надання персоналізованих порад може стати конкурентною перевагою нового додатку. Також, розробка зручного та графічного інтерфейсу користувача, який дозволить легко взаємодіяти з додатком, буде ключовим аспектом успіху.

Додатки для управління фінансами відрізняються різноманітними підходами до візуалізації фінансових даних та надання користувачам інформації. Кожен з цих додатків може мати свої унікальні функціональні можливості, які спрямовані на полегшення сприйняття та аналізу фінансової інформації. Від відображення кругових діаграм до графіків інтерактивних та аналітичних звітів, кожен інструмент старається забезпечити максимальну зрозумілість і зручність для користувача в управлінні своїми фінансами. Такий різноманітний підхід дозволяє користувачам обирати додаток, який найкращим чином відповідає їхнім потребам і стильовим уподобанням, забезпечуючи при цьому необхідний рівень функціональності і якості обслуговування.

Деякі додатки відображають фінансову інформацію у вигляді графіків, діаграм або таблиць, що дозволяє користувачам швидко оцінити свою фінансову ситуацію та зрозуміти основні тенденції. Це допомагає користувачам легше

виявити свої фінансові потреби, визначити пріоритетні напрямки витрат та зробити більш обґрунтовані фінансові рішення.

Інші додатки можуть пропонувати інструменти для встановлення та відстеження фінансових цілей. Наприклад, користувачі можуть встановлювати цілі на заощадження для покупки автомобіля, оплату власного житла чи інших важливих витрат. Ці функції дозволяють користувачам створювати конкретні плани дій, визначати кроки для досягнення своїх фінансових мет та відслідковувати їх виконання в реальному часі. Такий підхід стимулює користувачів до більш осмисленого та ефективного управління своїми фінансами, сприяючи досягненню фінансової стабільності та благополуччя.

Окрім цього, важливо розглянути і оцінити рівень доступності та якість підтримки користувачів у різних додатках для управління фінансами. Швидкість і ефективність вирішення користувальницьких запитів та проблем можуть значно впливати на задоволення користувачів від використання додатку, а також на їхню готовність залишатися з ним на довгостроковій основі. Добре організована і оперативна підтримка може забезпечити користувачам впевненість у тому, що їхні питання будуть вирішені швидко і професійно, що підвищує загальний рівень взаємодії з продуктом і сприяє збереженню лояльності клієнтів. Такий підхід до підтримки дозволяє покращувати якість обслуговування та підвищує конкурентоспроможність додатка на ринку управління фінансами.

Також варто враховувати важливість інтеграції з іншими фінансовими інструментами та сервісами у мобільних додатках для управління фінансами. Наприклад, деякі програмні засоби можуть автоматично синхронізуватися з банківськими акаунтами користувачів, іншими фінансовими сервісами, або навіть з електронними гаманцями та інвестиційними платформами. Це розширює можливості користувачів у моніторингу та управлінні своїми фінансами, забезпечуючи їм зручність і ефективність в щоденному фінансовому управлінні. Інтеграція з іншими сервісами дозволяє автоматизувати процеси

обліку витрат, отримання фінансових звітів і аналізу, що є важливим для більш глибокого розуміння та керування особистими фінансами.

Усі ці аспекти важливі для розробки конкурентоздатного мобільного додатку для управління та моніторингу особистих фінансів. Ретельний аналіз аналогів у цій сфері дозволить виявити недоліки існуючих рішень та розробити продукт, який задовольнить потреби користувачів та відповідає сучасним вимогам ринку.

Окрім того, важливо врахувати питання безпеки та конфіденційності даних. Користувачі мають довіру в додаток, який гарантує захист їх фінансових даних та особистої інформації. Тому важливо використовувати сучасні методи шифрування та захисту даних, а також надавати користувачам повний контроль над своєю приватністю.

Mint[6] є одним із передових мобільних додатків для управління фінансами, що надає користувачам широкі можливості відстеження та керування їхніми фінансовими потоками. В основі Mint лежить низка функціональних можливостей, спрямованих на максимальне зручне та ефективне управління фінансовими ресурсами.

Серед ключових функцій Mint варто відзначити можливість створення бюджетів, що дозволяє користувачам планувати свої витрати та доходи, а також відстежувати їх виконання. Додаток надає можливість відслідковувати транзакції на банківських рахунках і кредитних картках, а також проводити детальний аналіз фінансового стану. Крім того, він забезпечує користувачів нагадуваннями про платежі та витрати, допомагаючи таким чином підтримувати дисципліну в управлінні фінансами.

Однією з ключових переваг Mint (див. рисунок 1.1) є його здатність автоматично синхронізуватися з різними фінансовими установами. Це дозволяє користувачам легко відстежувати всі свої фінансові операції в одному місці, що значно спрощує процес управління фінансами та робить його більш прозорим та ефективним.

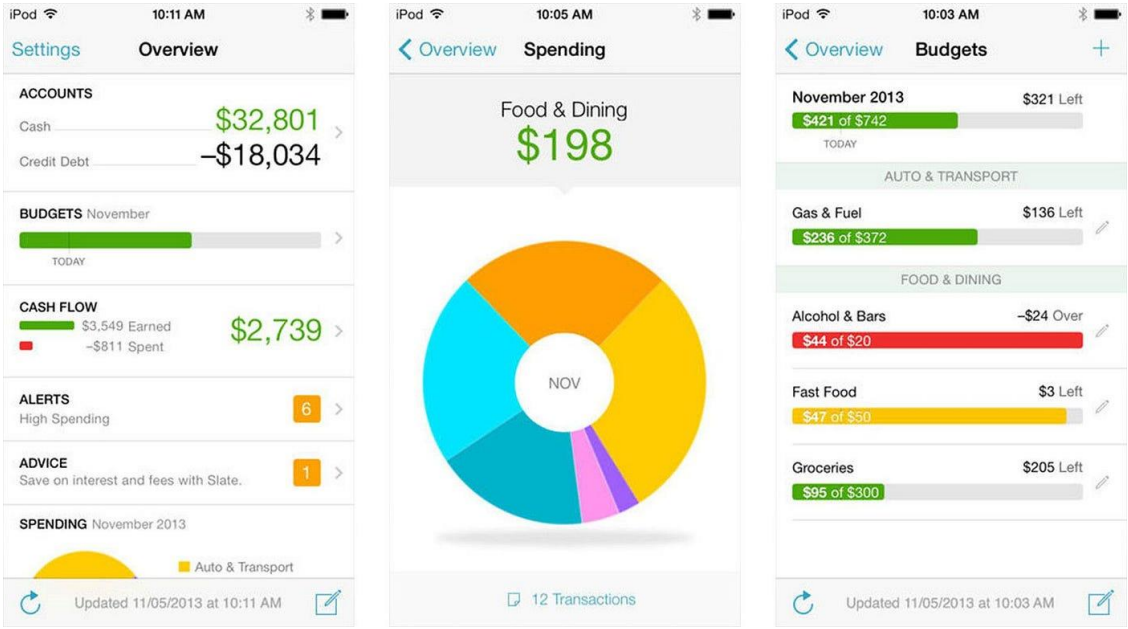


Рисунок 1.1 – Інтерфейс застосунку Mint

Personal Capital[7], як відомий мобільний додаток, спеціалізується на інвестиційному управлінні та фінансовому плануванні. Його основна функціональність спрямована на надання користувачам інструментів для ефективного керування їхніми інвестиційними портфелями та планування фінансового майбутнього.

Серед основних можливостей Personal Capital слід відзначити можливість відстеження інвестиційних портфелів, що дозволяє користувачам аналізувати ризики та доходність своїх інвестиційних активів. Крім того, додаток надає персоналізовані поради з оптимізації інвестиційного портфеля, що сприяє максимізації прибутків та мінімізації ризиків.

Окрім інвестиційного управління, Personal Capital (див. рисунок 1.2) також надає інструменти для планування пенсійних накопичень, податкового планування та фінансового аналізу. Ці можливості роблять додаток незамінним інструментом для тих користувачів, які цікавляться довгостроковим фінансовим плануванням та бажають забезпечити стабільне фінансове майбутнє для себе та своєї сім'ї.

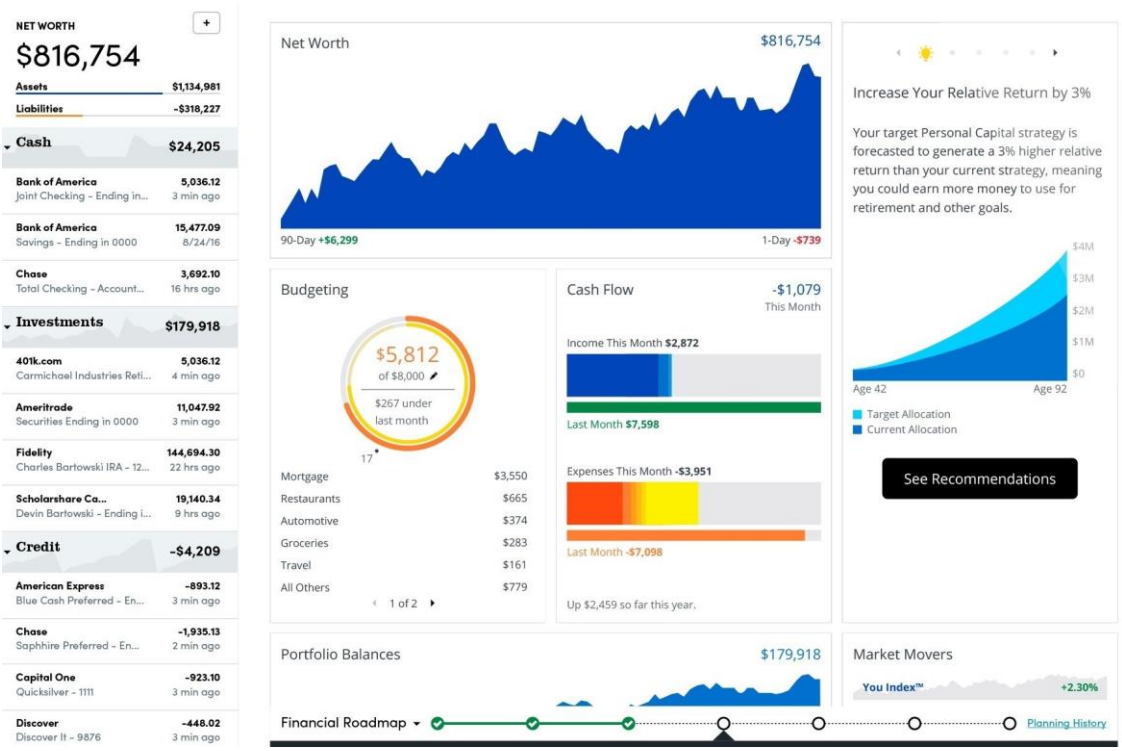


Рисунок 1.2 – Інтерфейс застосунку Personal Capital

YNAB (You Need a Budget), як мобільний додаток (див. рисунок 1.3), спрямований на те, щоб допомогти користувачам ефективно створювати та дотримуватися особистого бюджету. Його основний принцип полягає в тому, щоб призначити кожному долару певну роль в бюджеті, що допомагає користувачам керувати своїми фінансами більш ефективно та систематично.

Однією з ключових функцій YNAB є можливість відстежувати доходи та розходи, дозволяючи користувачам точно аналізувати свої фінансові потоки. Крім того, додаток дозволяє створювати категорії для різних видів витрат, що сприяє більшій структуризації та організації бюджету. Користувачі можуть визначати фінансові цілі та планувати свої витрати в майбутньому, що допомагає забезпечити фінансову стабільність та досягти поставлених цілей.

Поміж інших переваг YNAB [8] є доступ до різноманітних навчальних матеріалів та ресурсів, які допомагають користувачам вивчити навички управління бюджетом та стати фінансово незалежними. Ці ресурси можуть включати інтерактивні курси, практичні поради та інструменти для ефективного використання фінансів. Такий підхід дозволяє користувачам не лише

контролювати свої витрати, але й активно працювати над покращенням своєї фінансової грамотності та здатності управляти грошима.

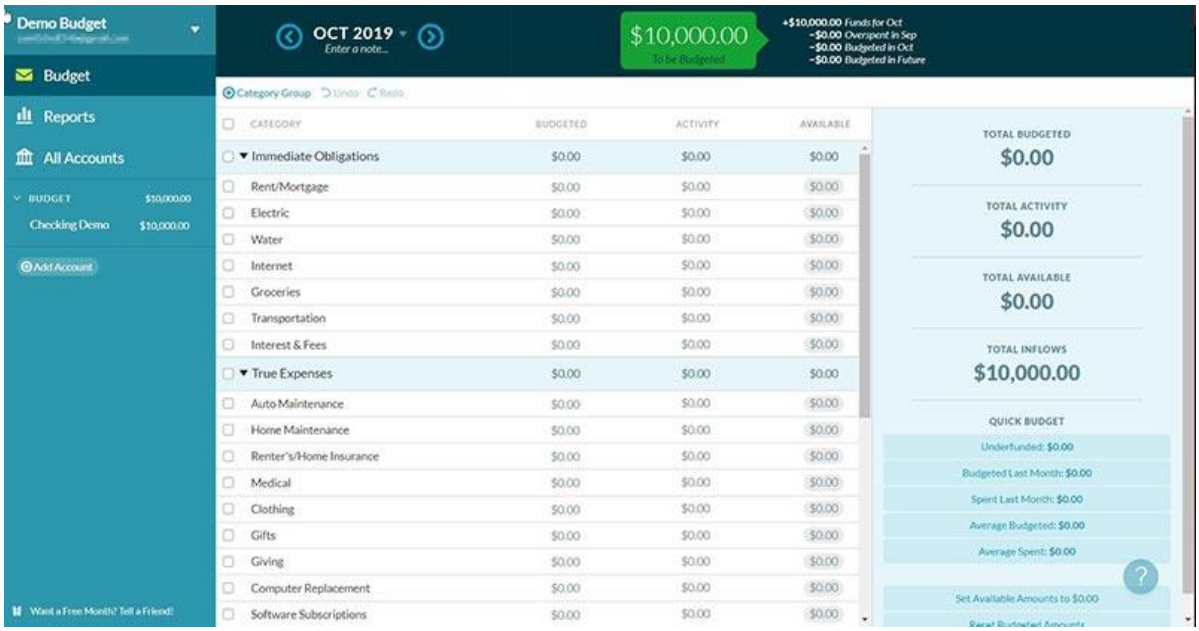


Рисунок 1.3 – Інтерфейс застосунку YNAB

PocketGuard (див.рисунок 1.4) - це додаток для управління особистими фінансами, що відкриває перед користувачами широкі можливості відстеження витрат, планування бюджету та ефективного управління їхніми фінансами. Однією з його ключових особливостей є автоматизована система категоризації транзакцій, яка дозволяє користувачам отримати чітке уявлення про свій фінансовий стан та основні напрямки витрат.

Додаток PocketGuard[9] також надає широкий спектр звітів та аналізів фінансової активності, що допомагає користувачам зрозуміти, куди саме йдуть їхні гроші та як можна оптимізувати свої витрати для досягнення більшої фінансової стабільності.

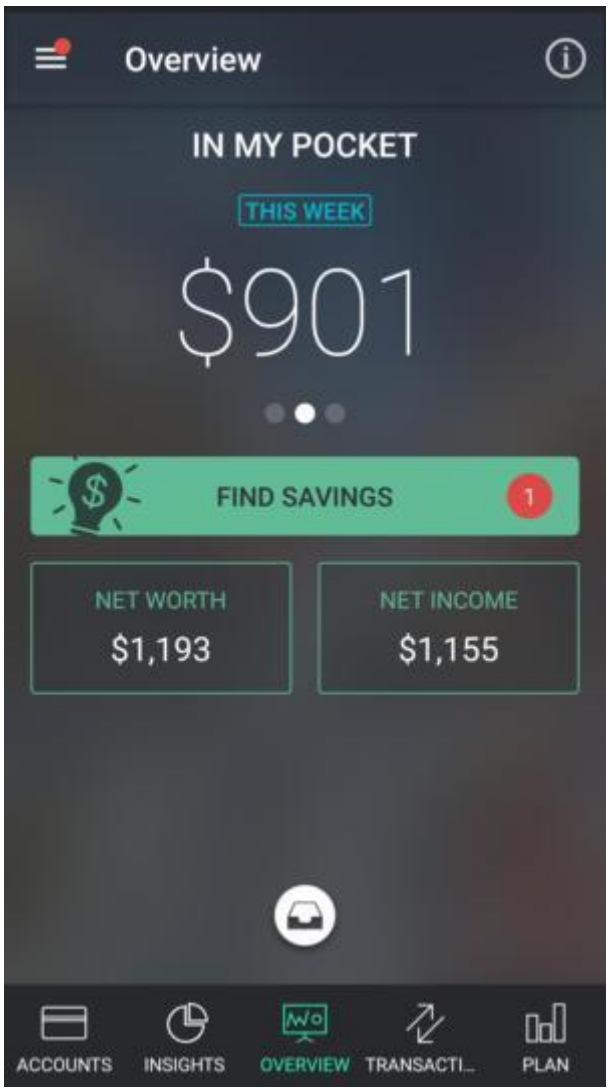


Рисунок 1.4 – Інтерфейс застосунку PocketGuard

Monefy (див. рисунок 1.5) - це додаток для ведення особистих фінансів, що дозволяє користувачам ефективно контролювати свої витрати та доходи. Він пропонує широкі можливості управління фінансами, спрощуючи процес ведення обліку та аналізу фінансових операцій.

Однією з ключових особливостей Monefy[10] є його інтуїтивно зрозумілий інтерфейс, який забезпечує зручність у використанні. Користувачі можуть легко додавати нові транзакції та відстежувати свою фінансову активність, не витрачаючи багато часу на навчання роботі з додатком.

Завдяки зручній категоризації транзакцій, користувачі можуть швидко та зрозуміло визначити, на що саме витрачають свої кошти. Крім того, додаток

дозволяє створювати власні категорії витрат, що дозволяє персоналізувати облік фінансів з урахуванням індивідуальних потреб та пріоритетів користувачів.

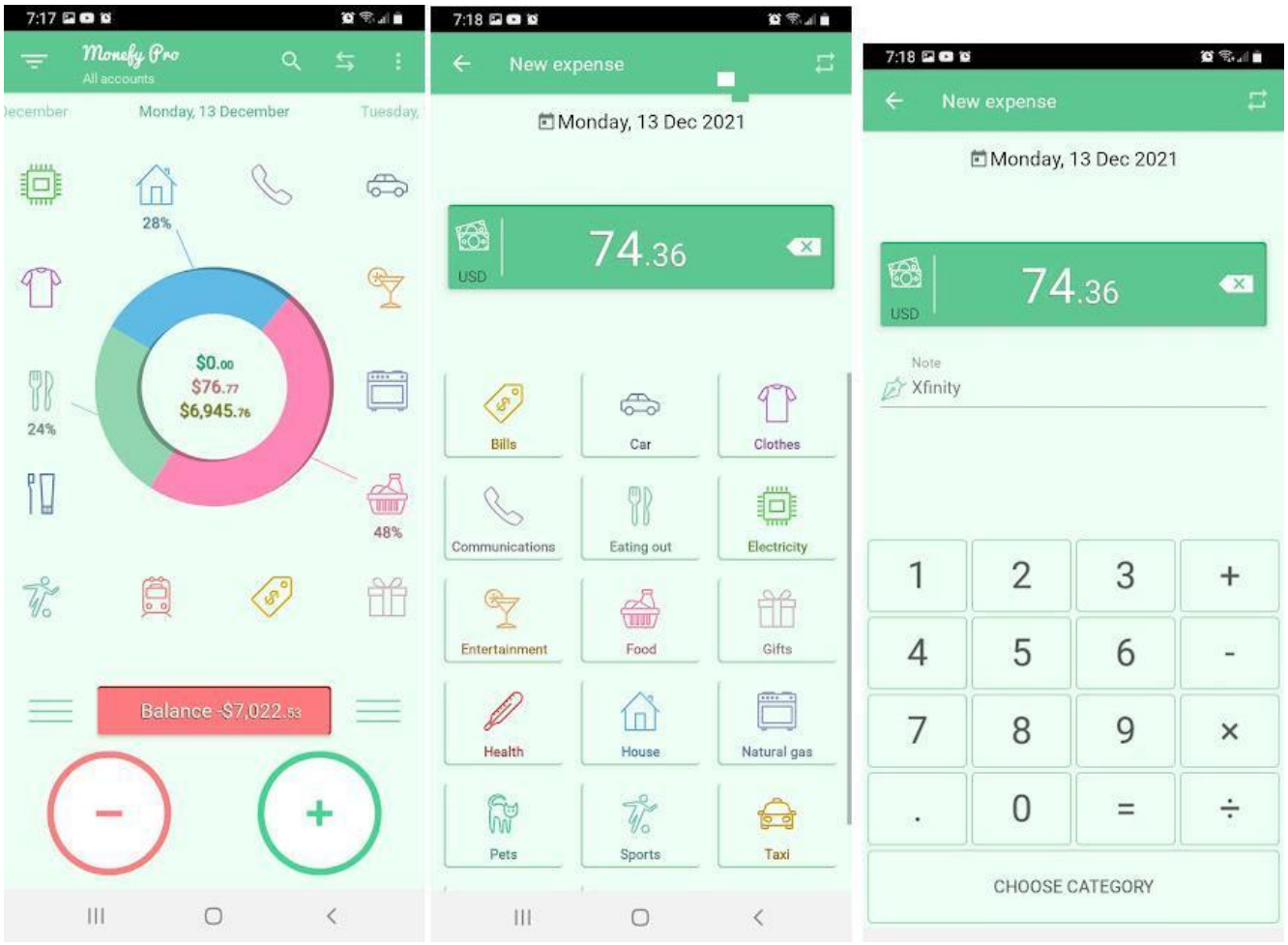


Рисунок 1.5 – Інтерфейс застосунку Monefy

Розробка програмних засобів для мобільної системи управління та моніторингу особистих фінансів передбачає впровадження функціоналу для ефективного аналізу та контролю фінансових потоків користувача. Це включає розробку модулів для збору, обробки та візуалізації фінансових даних, створення інтерфейсів для взаємодії з користувачем, а також забезпечення безпеки та конфіденційності інформації. Для успішної реалізації проекту необхідно провести аналіз потреб цільової аудиторії, розробити ефективну архітектуру програмної системи, інтегрувати необхідні технології для забезпечення функціональності та продуктивності додатку.

Результати порівняння аналогів зведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Mint	Personal Capital	YNAB	PocketGuard	Monefy	Budget
Автоматична синхронізація з різними банками та кредитними картками	1	1	0	0	0	1
Велика кількість доступних функцій	1	1	1	1	1	1
Аналітика та звіти про фінансовий стан	1	1	1	1	0	1
Можливість встановлення додаткових цілей	0	1	0	1	1	1
Безкоштовний доступ	1	0	0	0	0	1
Сумарний коефіцієнт	4	4	2	3	2	5

Порівняльний аналіз аналогів у сфері розробки мобільних додатків для управління фінансами відкриває широкі можливості для ідентифікації як сильних, так і слабких сторін існуючих рішень. Цей аналіз дозволяє виявити ключові переваги та недоліки кожного додатку, а також визначити можливості для вдосконалення та інновацій.

Результати такого порівняльного аналізу важливі для того, щоб створити продукт, який відповідає потребам користувачів і має конкурентні переваги на ринку фінансових додатків. Інсайти, отримані з аналізу, можуть служити

основою для розробки стратегій підвищення якості та функціональності власного додатку, а також для пошуку нових можливостей для вдосконалення та інновацій. Такий підхід допомагає побудувати успішну та конкурентоздатну продуктову стратегію, спрямовану на задоволення потреб і очікувань користувачів.

1.3 Аналіз методів розв'язання задачі

Розробка мобільного додатку для управління та моніторингу особистих фінансів є актуальною темою в сучасному світі, де цифрові технології стають все більш важливими для керування фінансами. Такий додаток може значно спростити життя користувачів, допомагаючи їм краще контролювати свої витрати, планувати бюджет, вести облік доходів і витрат, а також ефективно використовувати свої фінансові ресурси.

Одним з ключових функціональних можливостей мобільного додатку для управління фінансами є можливість вести детальний облік доходів і витрат. Користувачам можуть бути доступні різні категорії витрат, такі як їжа, транспорт, житло, розваги тощо, щоб вони могли легко класифікувати свої витрати та аналізувати, куди йдуть їхні кошти. Крім того, додаток може автоматично відстежувати витрати, за допомогою імпорту операцій з системи Google Pay.

Ще однією важливою функцією може бути можливість планування бюджету. Додаток може надавати користувачам зручні інструменти для встановлення місячного бюджету на різні категорії витрат, а потім відстежувати, наскільки вони дотримуються своїх фінансових цілей. Також можливість отримання попереджень про перевищення бюджету може допомогти уникнути зайвих витрат та керувати фінансами більш раціонально.

Крім того, мобільний додаток для управління фінансами може включати інструменти для створення і відстеження фінансових цілей. Користувачі можуть встановлювати цілі на накопичення грошей на покупку автомобіля, оплату

подорожі або купівлю житла, а потім відстежувати прогрес у їх досягненні та отримувати поради щодо ефективного управління своїми фінансами.

Нарешті, зручність і простота використання мобільного додатку також є важливими аспектами. Зрозумілий графічний інтерфейс, а також можливість синхронізації даних між різними пристроями користувача можуть значно полегшити процес ведення обліку фінансів та зробити його більш ефективним і зручним для користувача.

Порівняно з іншими аналогами, розроблений додаток виділяється своїм унікальним функціоналом, що дозволяє користувачам здійснювати широкий спектр фінансових операцій з легкістю та ефективністю. Його простота використання та зручний інтерфейс зроблять його ідеальним вибором для будь-якого, хто прагне ефективно керувати своїми фінансами, незалежно від їхнього досвіду у використанні технологій. Наш додаток ставить перед собою завдання не лише спростити фінансовий управління, але й зробити його цікавим та стимулюючим процесом для користувачів, допомагаючи їм досягати фінансової стабільності та досягати своїх цілей.

1.4 Постановка задач

Проаналізувавши переваги та недоліки існуючих програмних засобів для управління та моніторингу особистих фінансів, було визначено наступні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

– спочатку потрібно визначити основний функціонал додатку, такий як ведення обліку витрат та доходів, створення бюджету, аналіз фінансових потоків тощо. Це допоможе визначити основні можливості, якими повинен наділений додаток;

– важливо створити інтерфейс, який буде зручним у використанні для користувачів будь-якого рівня технічної підготовки. Це включає в себе розробку

зручних екранів для введення даних, просте навігаційне меню та зрозумілі підказки;

– при розробці додатку необхідно враховувати важливість захисту фінансових даних користувачів. Додаток повинен мати надійний механізм шифрування даних та використовувати безпечні методи збереження та обробки інформації;

– після розробки додатку необхідно провести комплексне тестування для виявлення можливих помилок та недоліків. Після цього додаток може бути вдосконалений з урахуванням отриманих результатів;

– після успішного завершення розробки та тестування додатку, його можна випустити на ринок. Потрібно також забезпечити подальшу підтримку додатку, включаючи виправлення помилок та випуск оновлень з новим функціоналом.

1.5 Висновки

У першому розділі було проведено детальний та ґрунтовний аналіз різноманітних програмних рішень, призначених для управління особистими фінансами. Серед цих рішень були розглянуті такі популярні додатки, як Mint, Personal Capital, YNAB (You Need a Budget), PocketGuard та Monefy. Кожен з цих додатків має свої унікальні особливості, переваги та недоліки, які були ретельно досліджені та проаналізовані у рамках даного розділу.

Аналіз показав, що на сучасному ринку вже існують рішення, які в цілому відповідають основним потребам користувачів у керуванні особистими фінансами. Однак, були виявлені певні прогалини і можливості для подальшого покращення існуючих продуктів. Наприклад, деякі додатки можуть бути надмірно складними для новачків, що лише починають свій шлях в управлінні фінансами. Інші ж додатки не забезпечують достатньої глибини аналітики, яка б допомогла користувачам краще зрозуміти свій фінансовий стан і приймати більш обґрунтовані фінансові рішення.

Враховуючи результати аналізу, було визначено, що розробка власного мобільного додатку для управління та моніторингу особистих фінансів є цілком доцільною та перспективною. Новий додаток може поєднувати найкращі риси існуючих рішень, а також впроваджувати власні інновації та вдосконалення для кращого задоволення потреб користувачів. Зокрема, новий додаток може мати більш інтуїтивний та простий у використанні інтерфейс, що дозволить залучити ширшу аудиторію, включаючи тих, хто тільки починає користуватися фінансовими інструментами.

Основні напрямки подальшої роботи над проектом включають розробку інтуїтивного та зручного інтерфейсу користувача, призначеного для максимальної зрозумілості навіть для новачків у області фінансів. Важливим аспектом є розробка функціоналу для автоматизації та оптимізації процесів управління фінансами, що дозволить користувачам ефективно керувати своїми фінансовими ресурсами. Не менш важливою частиною додатку буде створення потужної системи звітності та аналітики, яка надасть користувачам детальний облік їхніх фінансів. Це допоможе користувачам глибше розуміти свої витрати та доходи, а також знаходити можливості для заощадження та інвестування в майбутнє. Підходяще оформлення та логічна структура інтерфейсу сприятимуть зручності користування додатком, що в свою чергу забезпечить задоволення від використання та позитивний досвід користувачів.

Розробка мобільного додатку для управління та моніторингу особистих фінансів відкриває широкі можливості для створення зручного та ефективного інструменту, який допоможе користувачам не лише керувати своїми фінансовими ресурсами, але й зростати у впевненості та грамотності у фінансових питаннях. Новий додаток має потенціал стати надійним помічником у фінансовому плануванні та управлінні бюджетом, сприяючи досягненню фінансової стабільності та добробуту користувачів. Його інтуїтивний і простий у використанні інтерфейс дозволить легко організувати та аналізувати фінансові дані, що стане основою для ефективних рішень у фінансовій сфері. Такий додаток буде не лише інструментом для ведення обліку доходів та витрат, але й

освітнім ресурсом, що допомагатиме користувачам розуміти й оптимізувати їх фінансове становище на шляху до фінансової незалежності.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз даних системи

Для успішної розробки мобільного застосунку для аналізу грошових витрат необхідно провести детальне вивчення та аналіз вхідних даних. Цей етап є ключовим у процесі розробки, оскільки від правильної інтерпретації та обробки даних залежить якість і ефективність додатку. Аналіз вхідних даних включає в себе збір та організацію інформації про фінансові транзакції користувачів, їх категоризацію за видами витрат, часом та місцем здійснення, а також можливість автоматизованого збору даних з підключених банківських рахунків або інших джерел. Правильно налаштована система обробки вхідних даних забезпечить точність аналізу і надійність отриманих результатів, що є важливим для забезпечення коректної роботи всього фінансового аналітичного інструменту.

Вхідні дані:

- дані про доходи користувача;
- дані про витрати користувача за день, місяць, рік;
- додаткові бібліотеки які будуть використані для покращення функціоналу застосунку;
- опитування та фідбек користувачів для визначення потреб та очікувань від системи.

Потрібно провести ознайомлення з вхідними даними. Інформація про витрати та доходи користувача можуть бути введені вручну корисутвачами або отриманні за допомогою завнішніх додатків та баз даних.

Після ознайомлення з усіма отриманими даними необхідно провести попередню обробку, що включає кілька важливих етапів. По-перше, необхідно перевірити дані на наявність помилок, таких як неправильні значення чи некоректні формати. Важливо також переконатися, що інформація повна і ніякі дані не втрачені або пошкоджені під час збору або передачі. Валідація форматів

є не менш важливою, оскільки вона дозволяє переконатися, що дані про доходи та витрати користувача введені вірно та відповідають встановленим стандартам. Цей етап допомагає забезпечити точність та достовірність аналізу фінансових даних, що є критичним для подальшої роботи з ними в мобільному додатку.

Додатково, важливим етапом аналізу даних є визначення потреб користувачів і встановлення вимог до функціональності системи. Цей процес включає проведення опитувань та досліджень цільової аудиторії, щоб зрозуміти їхні очікування та потреби від системи. Важливо визначити, які конкретні функції користувачі вважають найбільш важливими та корисними для їхнього повсякденного використання. Встановлення пріоритетів функцій дозволяє зосередитися на ключових аспектах, які необхідно впровадити у системі для забезпечення максимальної користі для кінцевих користувачів. Такий підхід допомагає підвищити задоволеність користувачів і зробити систему більш конкурентоспроможною на ринку програмного забезпечення для управління фінансами.

Крім того потрібно провести аналіз конкурентних рішень, таких як Mint, Personal Capital, YNAB, PocketGuard та Monefy. Це дозволяє визначити сильні та слабкі сторони існуючих додатків і врахувати їх при розробці власного продукту.

Після закінчення аналізу даних можна почати розробку концептуальної моделі застосунку для аналізу грошових витрат. Модель повинна відображати компоненти системи та показувати, як вони взаємодіють між собою. Для наочності структуру та організацію системи можна представити у вигляді структурної схеми системи (див. рисунок 2.1). Це дозволить чітко зрозуміти, як функціонують окремі частини системи та їх взаємозв'язки.

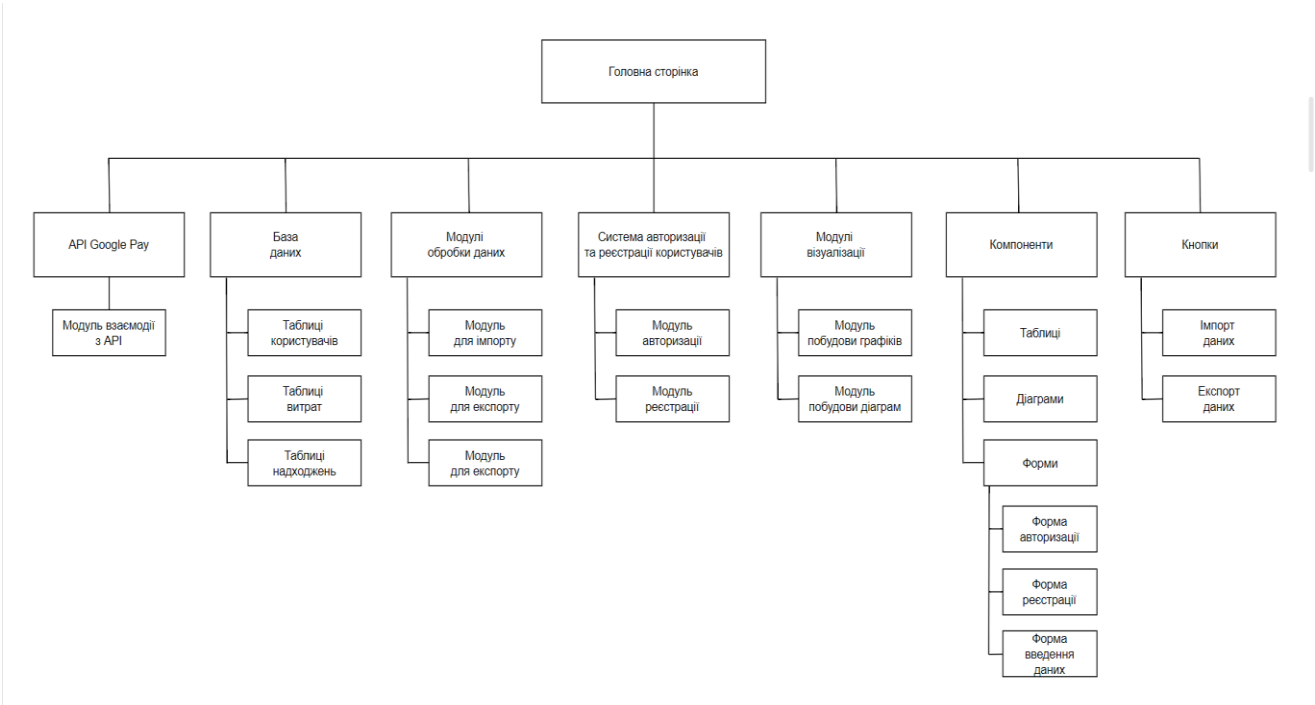


Рисунок 2.1 – Структурна схема системи

2.2 Розробка моделі системи

З метою забезпечення більшого розуміння принципів функціонування мобільного додатку для управління та моніторингу особистих фінансів, ми прийняли рішення розробити модель системи, використовуючи мову UML[11], зокрема діаграму діяльності (див. рисунок 2.2). Ця діаграма буде служити інструментом для ілюстрації послідовності операцій та взаємодій користувача з додатком. Вона дозволить краще зрозуміти, як саме додаток працює на рівні користувача, які кроки він виконує та які операції відбуваються під час цих взаємодій. Це надасть можливість розробникам, тестувальникам та іншим учасникам проекту отримати чітке уявлення про функціональність та логіку додатку, що сприятиме якості та ефективності розробки.

Процес розробки діаграми діяльності виконувався з використанням програмного забезпечення DrawIO[12]. Цей інструмент забезпечує широкі можливості для створення діаграм, а також дозволяє легко та зручно взаємодіяти з ними.

Розроблена система є веб-додатком для управління особистими фінансами, який забезпечує користувачам можливість точного відслідковування їхніх доходів та витрат. Крім того, вона дозволяє зберігати і імпортувати фінансові дані у форматі Excel, що спрощує обмін інформацією між різними пристроями та програмами. Одним із ключових функціональних компонентів системи є інтуїтивний інтерфейс користувача, який спрощує навігацію та використання додатку. Також важливими елементами є таблиці для зручного відображення фінансових даних, кнопки для швидкого експорту і імпорту даних, а також графіки, які дозволяють візуалізувати розподіл доходів та витрат у зручному для сприйняття форматі.

Імпорт даних в систему здійснюється за допомогою кнопки "Import". При активації цієї кнопки відкривається діалогове вікно, де користувач може вибрати файл у форматі Excel зі свого пристрою. Обрані дані зчитуються з вибраного файлу за допомогою бібліотеки xlsx, конвертуються в формат JSON і автоматично додаються до відповідних таблиць витрат та доходів в системі. Ця функція значно спрощує процес імпорту фінансових даних, що дозволяє користувачам легко і швидко використовувати попередньо створені або збережені файли Excel для оновлення своєї фінансової інформації.

Експорт даних реалізований через кнопку "Export". При натисканні цієї кнопки поточні дані з таблиць витрат та доходів зберігаються у вигляді Excel файлу. Структура файлу забезпечує сумісність з функцією імпорту, що дозволяє користувачу легко зберігати та переносити свої дані. Ця функція використовує бібліотеку xlsx для генерації та завантаження файлу.

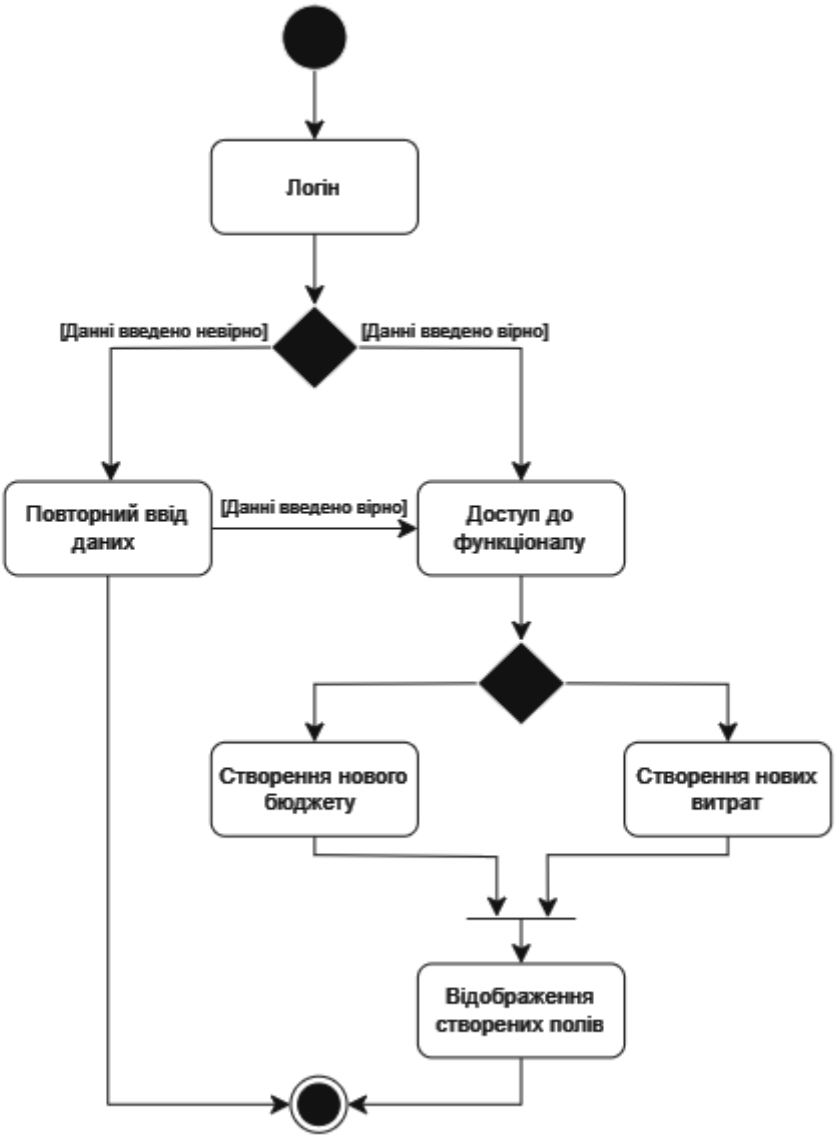


Рисунок 2.2 – Діаграма діяльності програмних засобів

Користувачі можуть додавати нові записи витрат та доходів через відповідні форми у таблицях. Кожен запис включає дату, назву, вартість, категорію (для витрат) та тип (для доходів). Записи можуть бути відредаговані або видалені за допомогою кнопок редагування та видалення, розташованих у кожному рядку таблиці. Це забезпечує гнучкість та точність у управлінні фінансовими даними.

Графіки, розташовані під таблицями, використовуються для візуалізації даних витрат та доходів. Графік витрат (expenditureChart) показує суму витрат за категоріями, а графік доходів (incomeChart) показує суму доходів за категоріями. Графіки автоматично оновлюються при додаванні, редагуванні або видаленні записів у таблицях[13] (див. рисунок 2.3). Відсоткові значення на графіках розраховуються відносно загальної суми витрат або доходів, що забезпечує точне відображення фінансової інформації. Для створення графіків використовується бібліотека Chart.js.

Система розроблена з використанням сучасних веб-технологій. Інтерфейс користувача побудований за допомогою HTML та CSS, зокрема з використанням бібліотеки Bootstrap CSS для забезпечення адаптивного та сучасного дизайну. Функціональність додатку реалізована на JavaScript, основний код міститься у файлі app.js. Для роботи з Excel файлами використовується бібліотека xlsx, а для створення графіків - бібліотека Chart.js.

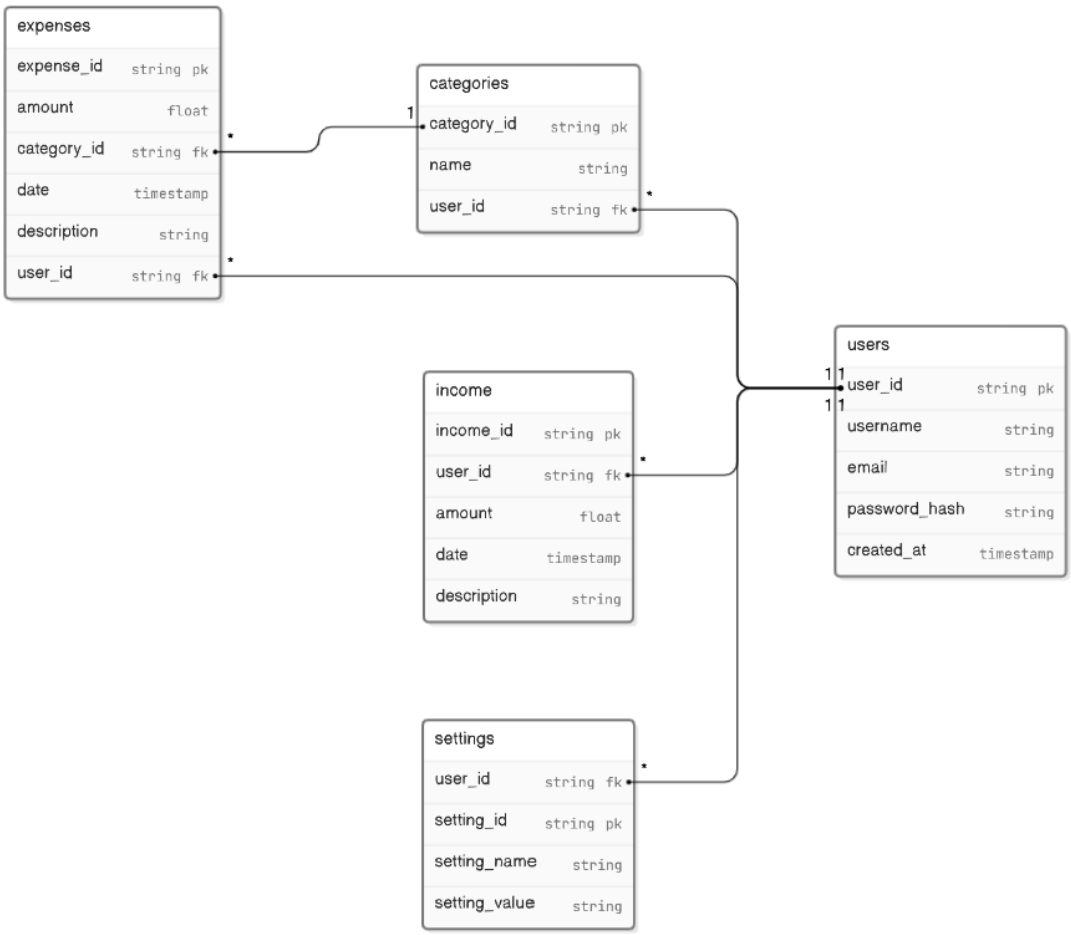


Рисунок 2.3 – Таблиці бази даних системи

Користувач взаємодіє з системою через графічний інтерфейс. Вибір місяця, імпорт та експорт даних, додавання та редагування записів здійснюються за допомогою простих та зручних елементів інтерфейсу. Графіки забезпечують візуальне представлення даних, що полегшує аналіз фінансової інформації. Завдяки використанню сучасних веб-технологій, система забезпечує високу продуктивність та надійність роботи.

У результаті розробки цієї системи користувачі отримують зручний інструмент для управління своїми фінансами, який дозволяє легко імпортувати, експортувати та аналізувати фінансові дані.

2.3 Розробка загального алгоритму роботи системи

Розробка програмних засобів для мобільної системи управління та моніторингу особистих фінансів передбачає створення ефективних алгоритмів, які дозволять користувачам зручно та ефективно керувати своїми фінансами.

Нижче описано загальний алгоритм роботи системи (див. рисунок 2.4).

– авторизація та налаштування облікового запису: користувач першим кроком авторизується в системі. Після цього він має можливість налаштувати особисті дані, встановити фінансові цілі та вказати персоналізовані параметри;

– додавання фінансових надходжень: користувач має можливість вказати свої доходи для візуального відображення у таблицях та файлах;

– додавання витрат: користувач може додавати окремі витратию При додавані нових витрат користувач може вказати суму, назву та категорію витрат, такі як їжа, житло, розваги, транспорт тощо.;

– моніторинг та аналіз фінансів: система надає користувачу зручні інструменти для моніторингу й аналізу його фінансів. Це включає відображення поточного стану бюджетів за допомогою цифрових значень, таблиці з данними та візуальних діаграм.

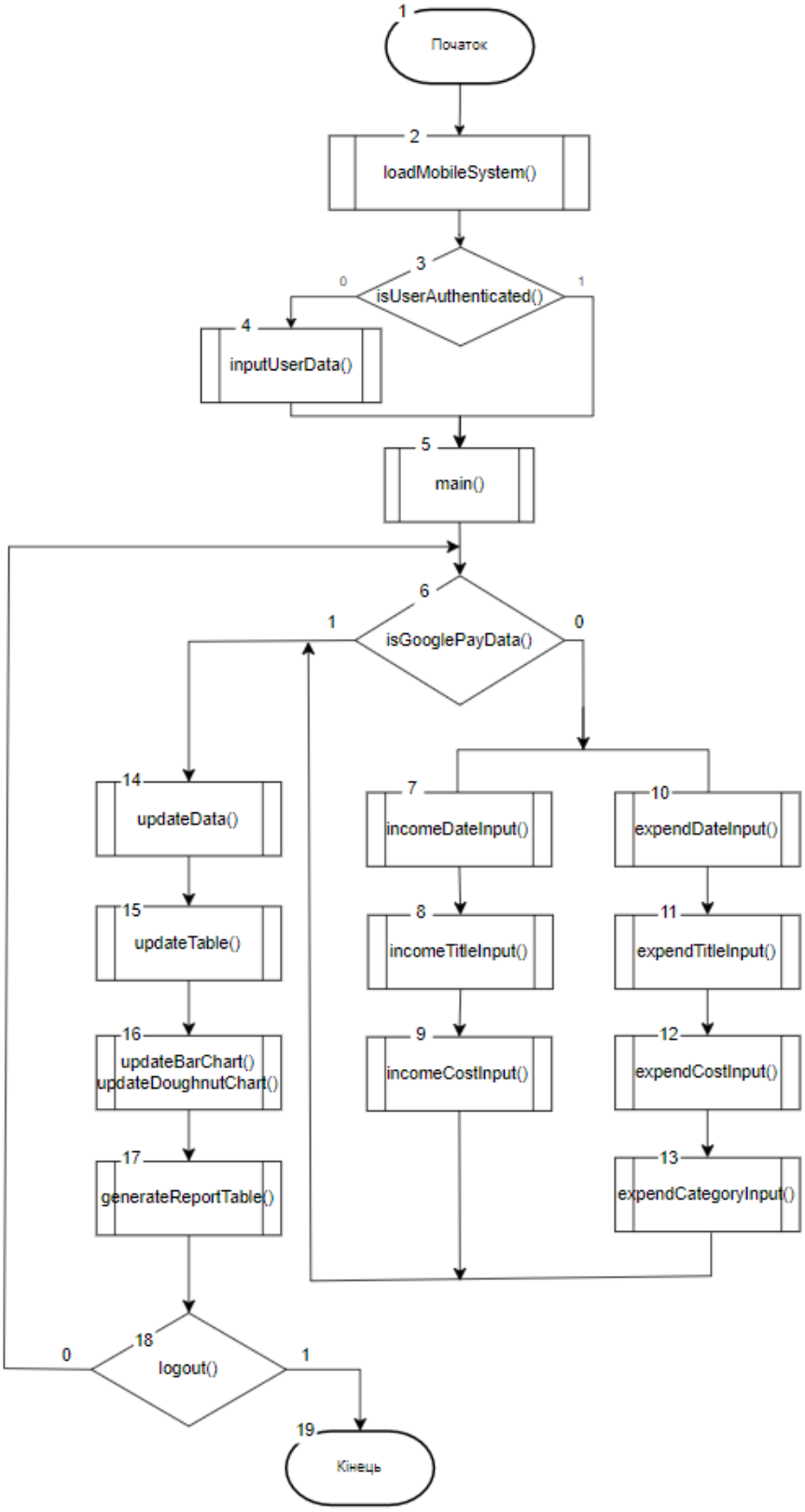


Рисунок 2.4 – Алгоритм роботи застосунку

Ці алгоритми, що були розроблені та впроваджені у мобільну програму, мають ключове значення для забезпечення ефективного управління та

моніторингу особистих фінансів користувачів. Вони створюють основу, на якій ґрунтується весь функціонал програми, дозволяючи зручно та ефективно керувати фінансами в їх різноманітних аспектах.

Ці алгоритми забезпечують можливість виконання різноманітних фінансових операцій, включаючи, але не обмежуючись, додавання та видалення витрат, управління рахунками та категоріями витрат, а також проведення аналізу фінансової ситуації. Вони дозволяють користувачам не лише вести облік своїх витрат та доходів, але й здійснювати розумні фінансові рішення на основі отриманих даних.

Завдяки алгоритмам, користувачі можуть отримувати актуальну інформацію про свої фінанси в реальному часі, а також ефективно планувати свої фінансові дії. Вони створюють можливість для глибокого аналізу та вивчення фінансових показників, що допомагає користувачам досягати своїх фінансових цілей та планів з більшою впевненістю і ефективністю.

2.4 Висновки

У другому розділі нашого дослідження було проведено глибокий аналіз вхідних та вихідних даних програмних засобів, що застосовуються в сфері управління та моніторингу особистих фінансів. Під час цього аналізу визначалися ключові типи даних, необхідні для оптимального функціонування мобільного додатку. Було враховано різноманітні аспекти, такі як дані про доходи, витрати, категорії транзакцій, кредитні картки, банківські рахунки тощо. Також було визначено, які саме дані можуть бути корисними для кінцевих користувачів у процесі управління фінансами.

Завдяки отриманому уявленню про типи даних, їх структуру та взаємозв'язки, була розроблена модель роботи програмного продукту з використанням UML діаграм. Ця модель ілюструє як система буде інтегруватися та функціонувати у реальному середовищі. Вона включає в себе основні

компоненти системи, взаємодію між ними та зовнішніми агентами, такими як користувачі та зовнішні сервіси, з якими система може взаємодіяти.

Детальна UML діаграма також дозволяє краще зрозуміти, як саме користувачі будуть взаємодіяти з програмним продуктом. Вона відображає всі можливі взаємодії з системою, включаючи введення даних, перегляд і аналіз фінансової інформації, а також налаштування особистих налаштувань і параметрів додатку. Такий підхід дозволяє забезпечити більш ефективне і зручне використання мобільного додатку для управління особистими фінансами кінцевими користувачами.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка мобільного застосунку для аналізу грошових витрат є складним завданням, яке потребує використання сучасних технологій та інструментів для забезпечення ефективності та зручності користувачів. Для реалізації такого застосунку було обрано середовище розробки VS Code[14] та використано ряд бібліотек, таких як Bootstrap CSS, Box Icons, Chart JS, Chart JS Plugin, CountUp JS, Sweet Alert JS, Sheet JS та File Save JS.

Bootstrap CSS[15] використовується для створення адаптивного та стильного інтерфейсу користувача. Ця бібліотека забезпечує швидке та легке налаштування стилів компонентів, що дозволяє зменшити час на розробку дизайну та забезпечити його єдиний вигляд на різних пристроях.

Box Icons[16] - це бібліотека векторних іконок, яка допомагає зробити інтерфейс додатку більш зрозумілим та привабливим. Іконки можуть бути використані для позначення різних функцій, таких як додавання нових витрат, перегляд звітів або налаштування облікових записів.

Chart JS[17] використовується для візуалізації даних про витрати у вигляді графіків та діаграм. Це допомагає користувачам краще зрозуміти їх фінансовий стан та виявляти тенденції у витратах. Chart JS Plugin додає додаткову функціональність для покращення взаємодії з графіками.

CountUp JS[18] застосовується для анімації числових даних. Наприклад, під час завантаження звіту про витрати, числа можуть плавно збільшуватись до своїх значень, що додає динамічності та привабливості інтерфейсу.

Sweet Alert JS[19] використовується для створення красивих та інтерактивних сповіщень. Це допомагає зробити повідомлення про помилки, успішні дії чи попередження більш приємними та зрозумілими для користувачів.

Sheet JS[20] дозволяє зчитувати та обробляти дані з електронних таблиць, що може бути корисним для імпорту або експорту даних про витрати. File Save JS використовується для збереження цих даних у файл на пристрої користувача.

При розробці програмного забезпечення для аналізу грошових витрат було враховано декілька ключових аспектів:

Додаток повинен дозволяти користувачам вводити свої витрати, категоризувати їх та надавати можливість додавання нотаток. Всі ці дані зберігаються локально.

Використання Chart JS дозволяє створювати графіки та діаграми, які відображають динаміку витрат за різні періоди, категорії витрат та інші параметри. Це допомагає користувачам візуально оцінити свої фінансові звички та приймати обґрунтовані рішення.

Інтерфейс додатку розроблено з використанням Bootstrap CSS та Box Icons, що забезпечує його адаптивність та зручність. Sweet Alert JS додає інтерактивності через зрозумілі сповіщення, а CountUp JS робить процес взаємодії з числовими даними більш приємним.

Використання Sheet JS та File Save JS дозволяє користувачам легко імпортувати дані з інших джерел та зберігати свої фінансові звіти у вигляді файлів для подальшого аналізу або архівування.

Поєднання цих технологій дозволяє створити мобільний застосунок для аналізу грошових витрат, який є не лише функціональним, але й привабливим та зручним у використанні. Користувачі можуть легко відстежувати свої витрати, отримувати зрозумілі звіти та приймати кращі фінансові рішення завдяки візуалізації та аналітиці.

Розробка мобільного застосунку для аналізу грошових витрат у середовищі VS Code з використанням зазначених бібліотек дозволяє забезпечити високий рівень якості та зручності для користувачів. Такий підхід гарантує, що додаток буде відповідати сучасним стандартам розробки програмного забезпечення та задовольняти потреби користувачів у зручному управлінні своїми фінансами.

3.2 Розробка модулів додавання витрат та доходів

Однією з ключових функцій мобільного застосунку для аналізу грошових витрат є модулі додавання витрат та доходів. Ці модулі дозволяють користувачам ефективно відстежувати свої фінансові потоки та зберігати інформацію про кожну транзакцію. Першим кроком у розробці модулів додавання витрат та доходів є створення інтерфейсу користувача за допомогою HTML і CSS, зокрема бібліотеки Bootstrap для забезпечення адаптивності та стилізації компонентів. Інтерфейс включає текстові поля для введення суми, вибору категорії, додаткових нотаток, а також кнопки для підтвердження введення даних.

Далі необхідно приєднати цей інтерфейс до відповідного JavaScript-коду для додавання необхідного функціоналу. Це включає обробку подій кнопок, контроль за текстовими полями та управління списком транзакцій. При натисканні на кнопку “Додати витрату” або “Додати дохід”, програма зчитує дані з текстових полів та зберігає їх у відповідний список. При натисканні кнопки ”Відмінити”, вікно для додавання витрат або доходів закривається, а введенні дані не зберігаються. Списки витрат та доходів можуть бути представлені у вигляді таблиць, які автоматично оновлюються після кожного додавання нової транзакції.

Для реалізації цього функціоналу використовується кілька бібліотек. Chart JS використовується для візуалізації даних про витрати та доходи. Графіки та діаграми допомагають користувачам аналізувати свої фінансові потоки, виявляти тенденції та приймати обґрунтовані рішення. Sweet Alert JS використовується для створення інтерактивних сповіщень, таких як підтвердження додавання нової транзакції або попередження про невірно введені дані. CountUp JS застосовується для анімації числових даних, що робить процес взаємодії з сумами витрат та доходів більш динамічним та приємним для користувачів. Sheet JS дозволяє імпортувати та експортувати дані про транзакції

з електронних таблиць, а File Save JS забезпечує можливість збереження даних у файл на пристрої користувача.

Алгоритм роботи модулів починається із завантаження екрану, де користувач бачить текстові поля для введення даних про витрати та доходи, а також кнопки для підтвердження. Користувач вводить дані про суму, категорію та нотатки, після чого натискає кнопку для додавання витрат або доходів. Дані зберігаються у локальну базу даних. Нові транзакції додаються до списку на екрані, а графіки оновлюються для відображення нових даних. Користувачі можуть переглядати деталі транзакцій, редагувати або видаляти їх за необхідності. Крім того, користувачі можуть імпортувати дані з електронних таблиць або експортувати свої фінансові звіти у вигляді файлів.

Цей підхід до розробки модулів додавання витрат та доходів забезпечує зручний графічний інтерфейс для користувачів. Використання сучасних бібліотек дозволяє створити функціональний та привабливий додаток, який допомагає ефективно управляти фінансами. Таким чином, розробка модулів додавання витрат та доходів з використанням VS Code та зазначених бібліотек дозволяє створити потужний інструмент для аналізу грошових витрат, який відповідає сучасним вимогам до мобільних застосунків та забезпечує зручність користувачів у повсякденному житті.

3.3 Розробка модуля перетворення данних в Excel таблицю

Головною метою модуля є створення Excel таблиці зі всіма даними про витрати та надходження користувача.

Функції writeToSheet та exportData є ключовими компонентами мобільного застосунку для аналізу грошових витрат, що дозволяють користувачам експортувати дані про свої витрати та доходи у файл Excel.

Функція writeToSheet відповідає за створення та збереження файлу Excel, що містить дані про витрати та доходи за вибраний місяць. Її робота починається з отримання значення вибраного місяця з елемента інтерфейсу. Далі вона

використовує бібліотеку SheetJS для перетворення списків витрат та доходів у формати аркушів Excel. Потім створюється новий робочий зошит (workbook), куди додаються створені аркуші з відповідними назвами, що включають вибраний місяць. Після цього робочий зошит записується у формат Excel, і за допомогою бібліотеки FileSaver.js створений файл зберігається на комп'ютері користувача під ім'ям, що включає назву файлу та вибраний місяць.

Функція exportData виконує збір, фільтрацію та сортування даних про витрати та доходи користувача перед їх експортом до Excel. Спочатку вона отримує витрати з локального сховища та фільтрує їх за вибраний місяць і поточний рік. Потім відсортовані витрати за датою. Аналогічні дії виконуються для доходів: їх отримують з локального сховища, фільтрують за місяцем та роком, і сортують за датою. Після підготовки даних про витрати та доходи, функція передає ці дані до функції writeToSheet, яка вже безпосередньо створює та зберігає файл Excel.

Ці функції забезпечують зручний спосіб для користувачів експортувати свої фінансові дані, що дозволяє їм аналізувати витрати та доходи за обраний період у форматі Excel, який легко використовувати для подальшого аналізу та зберігання.

3.4 Розробка дизайну інтерфейсу мобільної системи

Інтерфейс додатку має простий і інтуїтивно зрозумілий дизайн, який складається з декількох основних елементів, розташованих у логічному порядку для зручного користування. Основні елементи включають випадаючий список для вибору місяця, кнопки для імпорту та експорту даних, а також таблиці для відображення витрат і доходів. Додатково, є два графіки, які візуалізують витрати і доходи користувача. На рисунку 3.1 показано графічну схему інтерфейсу, а на рисунку 3.2 розроблений інтерфейс.

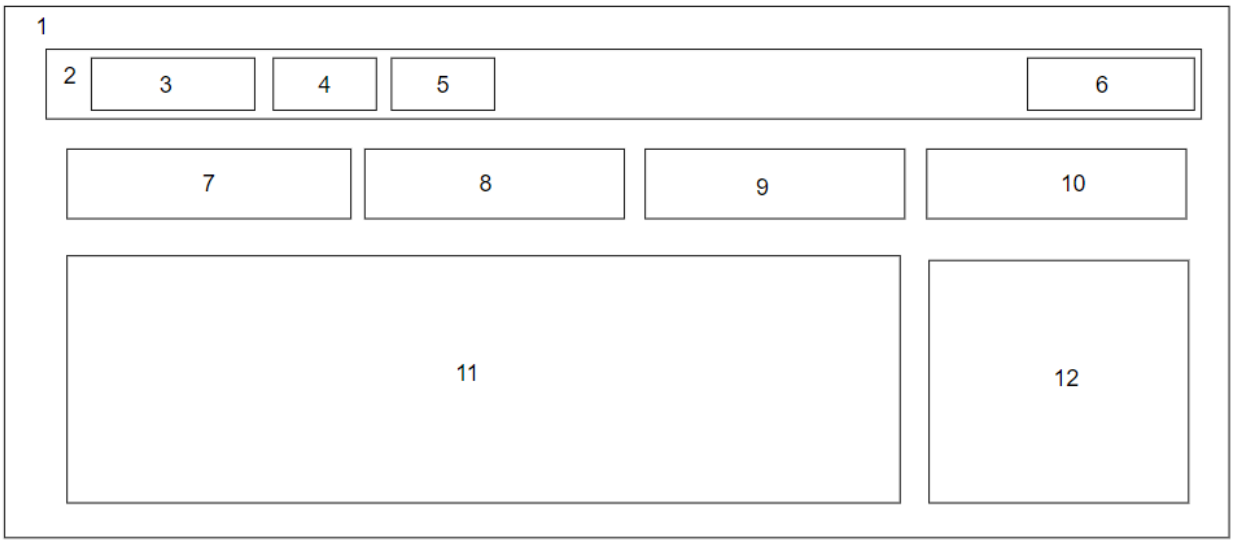


Рисунок 3.1 – Графічна схема інтерфейсу додатка

Елементи форми інтерфейсу додатка

1. Робоча область
2. Header
3. Назва додатка
4. Кнопка додавання витрат
5. Кнопка додавання доходів
6. Кнопка налаштування акаунта
7. Поле відображення денних витрат
8. Поле відображення місячних витрат
9. Поле відображення річних витрат
10. Поле відображення доходів
11. Графік витрат за місяць
12. Кругова діаграма витрат

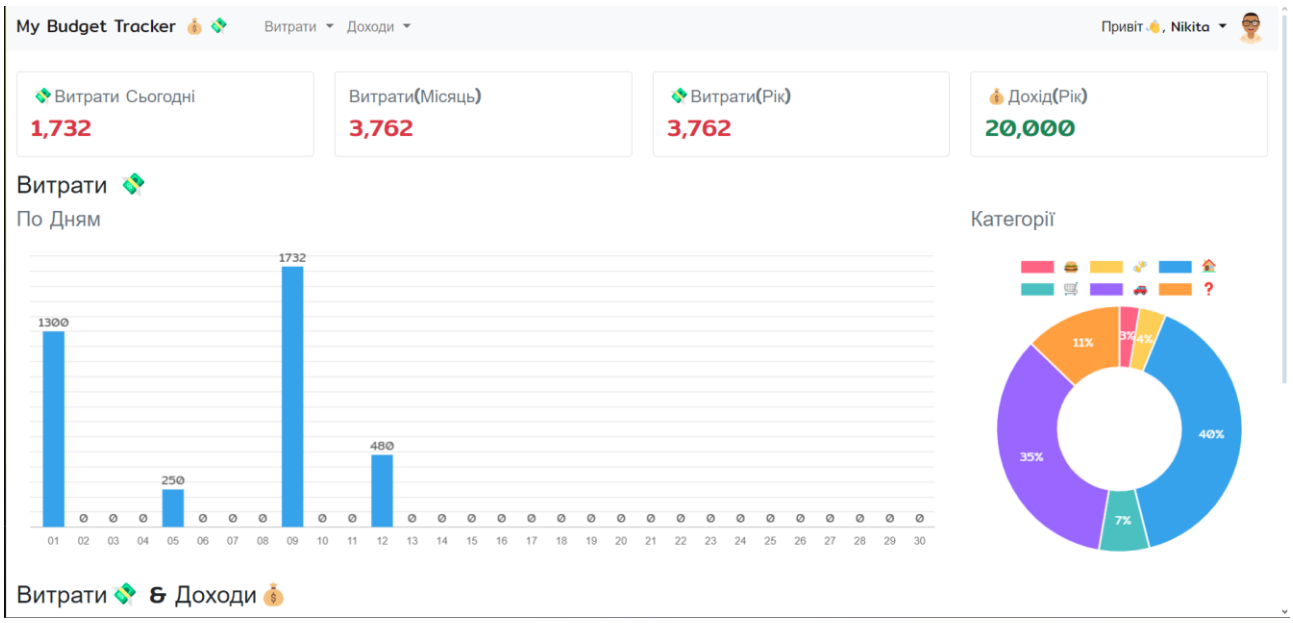


Рисунок 3.2 – Інтерфейс додатка

У верхній частині інтерфейсу є дві кнопки для додавання витрат та надходжень користувача. При натискані кнопки “Витрати” відкривається вікно в якому користувач вводить данні по витратам такі як дата, опис, сума та вибір категорії. При повному заповнені всіх полів можна зберегти витрату і вона відобразиться на діаграмах та внизу в таблиці, або натиснути кнопку “Відмінити”. На рисунку 3.3 показано графічну схему вікна додавання витрат, а на рисунку 3.4 розроблений інтерфейс вікна.

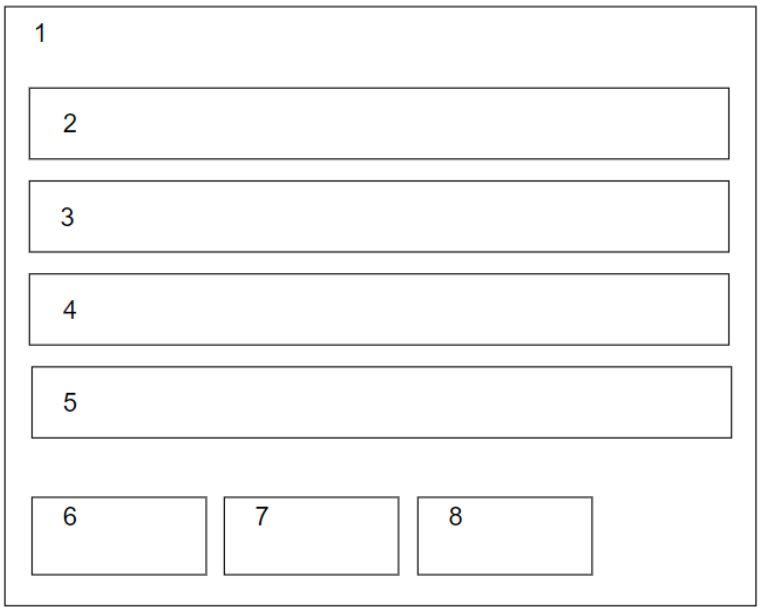


Рисунок 3.3 – Графічна схема вікна додавання витрат

Елементи форми інтерфейсу додатка

1. Робоча область
2. Поле вводу дати
3. Поле вводу інформації про витрату
4. Поле вводу суми витрат
5. Поле вибору категорії витрат
6. Кнопка “Додати”
7. Кнопка “Авто”
8. Кнопка “Відмінити”

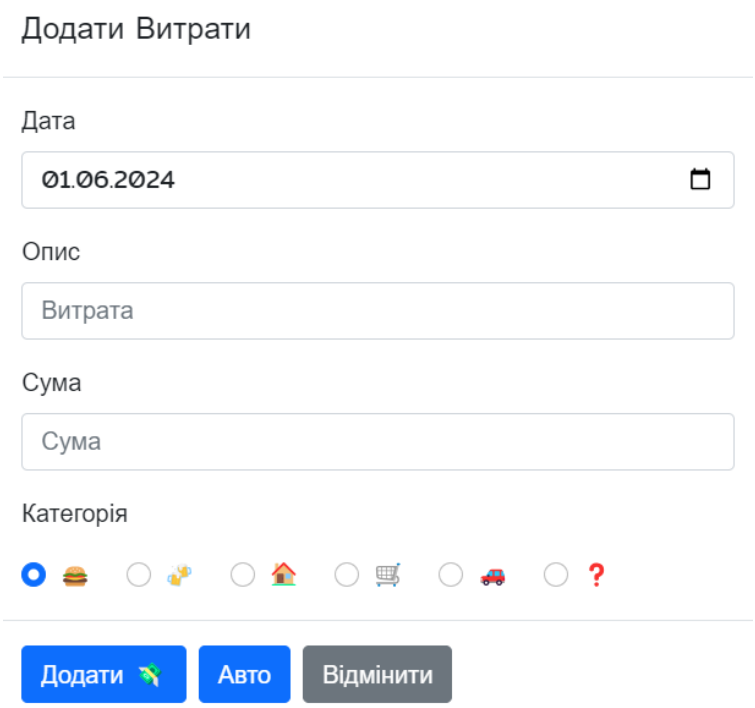


Рисунок 3.4 – Вікно додавання витрат

Відповідне вікно відкривається при натисканні на кнопку “Доходи” для додавання грошових надходжень з полями для вводу дати, опису та суми доходів. На рисунку 3.5 показано графічну схему вікна додавання доходів, а на рисунку 3.6 розроблений інтерфейс вікна.

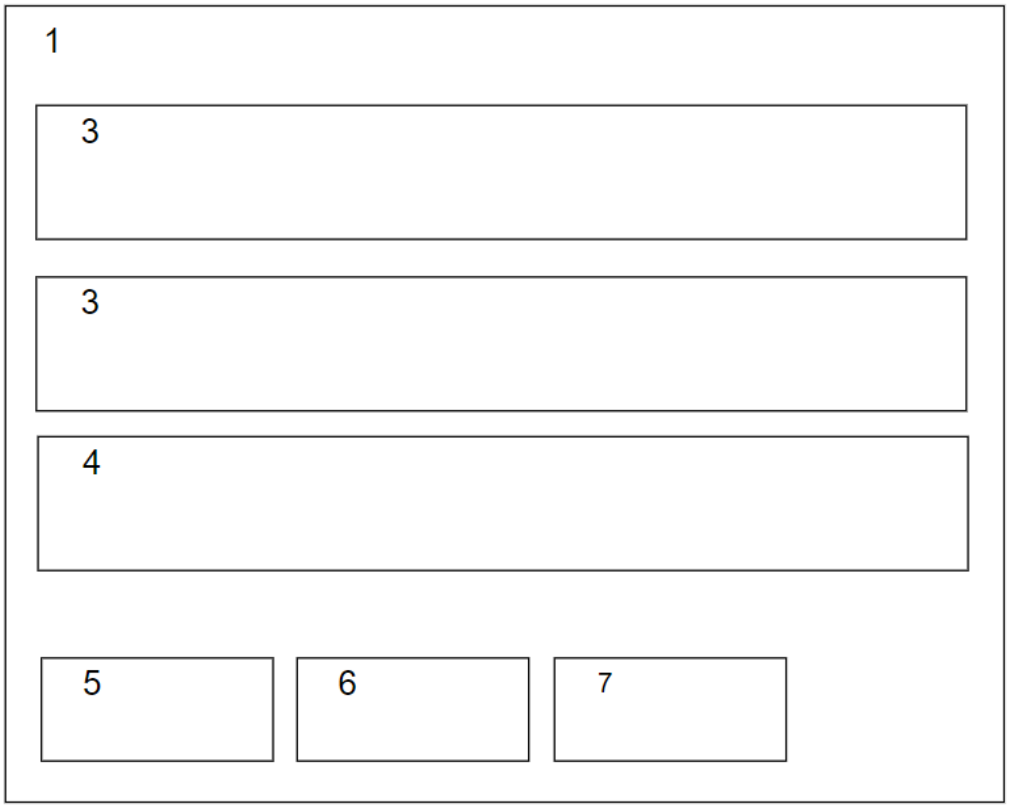


Рисунок 3.5 – Графічна схема вікна додавання доходів

Елементи форми інтерфейсу додатка

1. Робоча область
2. Поле вводу дати
3. Поле вводу інформації про доходи
4. Поле вводу суми доходів
5. Кнопка “Додати”
6. Кнопка “Авто”
7. Кнопка “Відмінити”

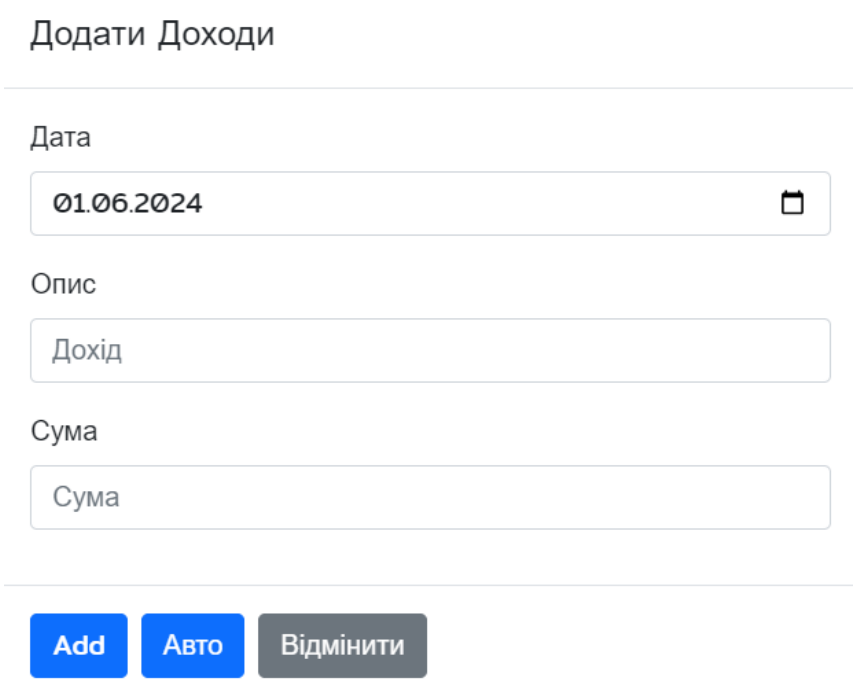


Рисунок 3.6 – Вікно додавання доходів

Під таблицями розташовані два графіки (див. рисунок 3.4), які візуалізують дані витрат та доходів. Для їх створення використовується бібліотека Chart.js. Графік витрат показує суму витрат за різними категоріями, а графік доходів показує суму доходів за різними категоріями. Це забезпечує візуальне представлення фінансової інформації, що допомагає користувачу швидко оцінити свої витрати та доходи.

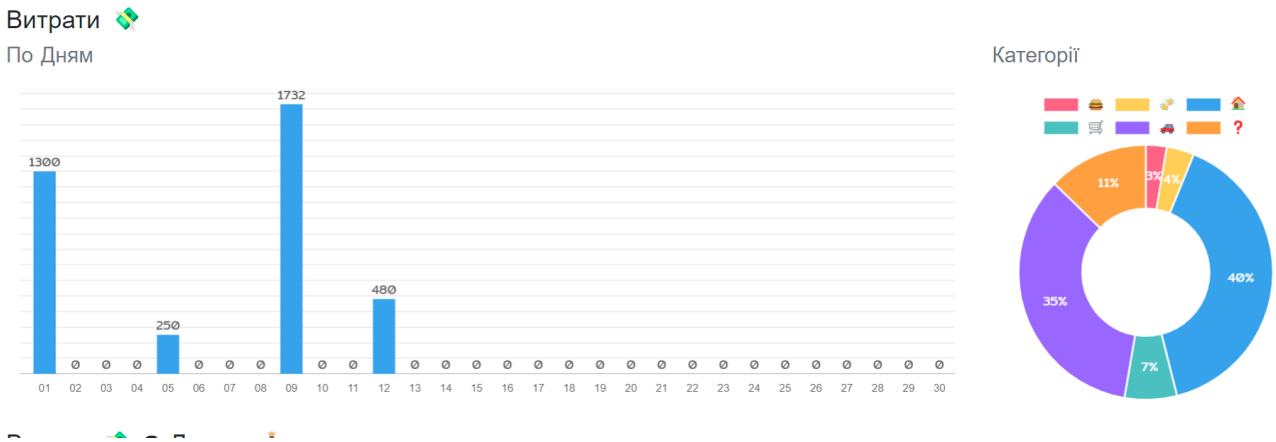


Рисунок 3.7 – Діаграми

Графіки автоматично оновлюються при додаванні або редагуванні даних в таблицях. Відсоткові значення на графіках розраховуються відносно загальної суми витрат або доходів, що забезпечує точне відображення фінансової інформації. Використання бібліотеки Chart.js дозволяє створювати динамічні та інтерактивні графіки, які легко адаптуються до змін у даних.

Далі на сторінці розташований випадаючий список для вибору місяця. Цей випадаючий список реалізований у вигляді елемента `select`, що надає користувачеві можливість обрати конкретний місяць для перегляду відповідних фінансових даних. Опції випадаючого списку охоплюють всі місяці року від січня до грудня. Це забезпечує зручний та інтуїтивно зрозумілий спосіб фільтрувати та переглядати фінансову інформацію за конкретними місяцями в рамках програмного продукту. Такий підхід дозволяє користувачам ефективно організовувати свої фінанси та аналізувати їх за різними періодами, що є важливим для правильного фінансового планування та управління бюджетом.

Праворуч від випадаючого списку вибору місяця розташовані кнопки "Import" та "Export". Кнопка "Import" дозволяє користувачу завантажувати дані з Excel файлу. Після натискання на цю кнопку відкривається діалогове вікно для вибору файлу, після чого дані з файлу зчитуються та додаються до відповідних таблиць витрат та доходів. Кнопка "Export" дозволяє користувачу зберігати поточні дані у вигляді Excel файлу. Це забезпечує зручний спосіб резервного копіювання та перенесення даних.

Під кнопками імпорту та експорту розташовані таблиці для відображення витрат та доходів (див. рисунок 3.5). Таблиця витрат, з ідентифікатором, містить стовпці для дати, назви, вартості, категорії та кнопки редагування. Таблиця доходів, з ідентифікатором, містить стовпці для дати, назви, вартості та кнопки редагування. Дані в таблиці додаються та редагуються користувачем, що забезпечує зручне управління фінансовою інформацією.

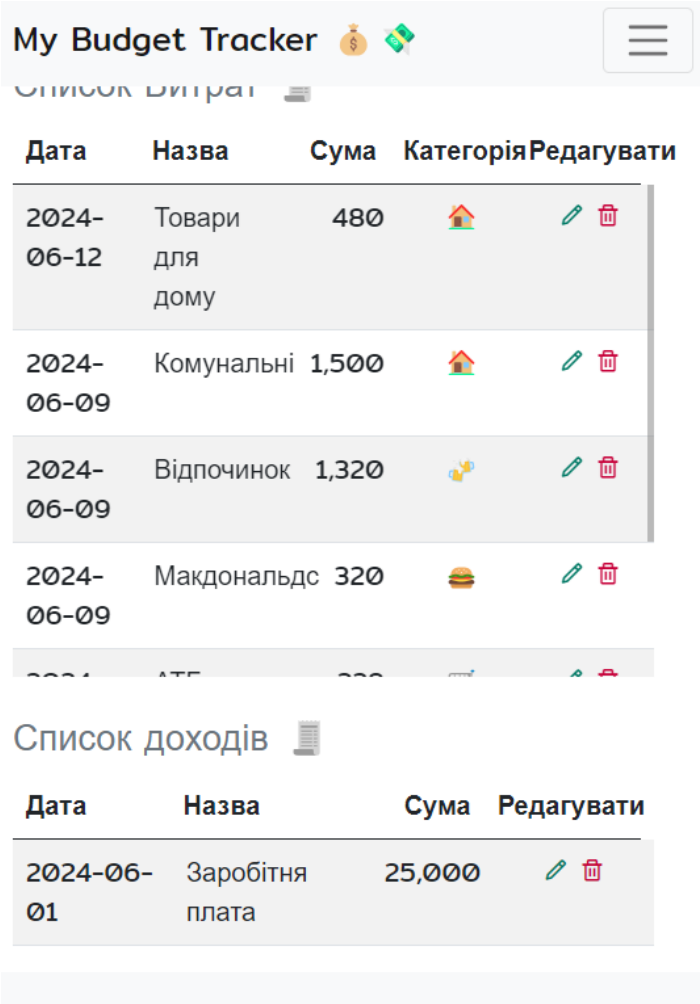


Рисунок 3.8 – Таблиці зі списками витрат та доходів

Основна функціональність додатку реалізована за допомогою мови JavaScript у файлі `app.js`. Код обробляє події натискання кнопок, імпорту файлів та оновлення графіків. Використовується бібліотека `xlsx` для роботи з Excel файлами та бібліотека `Chart.js` для створення графіків. Це забезпечує високу продуктивність та надійність роботи додатку.

Загальний дизайн інтерфейсу додатку є простим, інтуїтивно зрозумілим і забезпечує зручне управління фінансовими даними користувача.

3.5 Висновки

У третьому розділі було детально розглянуто вибір технологій, мови програмування, бібліотек та інструментів, що використовувались під час

розробки застосунку. Після ретельного вивчення вимог і можливостей було прийнято рішення на використання Visual Studio Code як середовище розробки а також мову програмування JavaScript через велику кількість підтримуваних бібліотек як допоможуть в виконанні різних функцій в проекті. Використання VS Code надає змогу ефективно керувати проектом завдяки широкому вибору інструментів функцій та розширень які допомагають при написанні коду та форматуванні його. Для реалізації графічної частини інтерфейсу було використано бібліотеку Bootstrap CSS, яка забезпечує сучасний, адаптивний та зручний у використанні дизайн компонентів.

4 ТЕСТУВАННЯ ПРОГРАМИ ТА ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Тестування системи

Тестування програмного забезпечення[21] – це комплексний процес, спрямований на перевірку програмного продукту з метою виявлення потенційних помилок, дефектів або неполадок, а також перевірки відповідності вимогам замовника та стандартам якості. Тестування є невід'ємною частиною розробки програмного забезпечення та виконується на різних етапах життєвого циклу продукту. Воно дозволяє виявити та усунути помилки ще до випуску продукту на ринок, забезпечуючи його надійність, функціональність та задоволення потреб користувачів.

Проведене тестування мало на меті перевірити відповідність розробленого програмного забезпечення вимогам, визначеним на етапі проектування, а також виявити та виправити можливі помилки та недоліки у функціонуванні системи.

Першочергово було проведено модульне тестування, під час якого перевірялася коректність роботи окремих функцій та компонентів програми. Для цього використовувалися спеціальні тестові набори, які охоплювали різноманітні сценарії використання та різні вхідні дані.

Під час проведення тестування виявлено дефект, що дозволяв користувачам вводити букви замість числових значень у полях фінансових даних. Це призводило до некоректних результатів при подальших обчисленнях та аналізі даних. Для усунення цієї проблеми було вжито заходів шляхом обмеження можливості введення будь-яких символів, крім цифр, у полях з сумою та кількістю витрат. Ця виправлення забезпечує правильну обробку та інтерпретацію введених користувачем даних, зберігаючи точність та надійність фінансових обчислень (див. рисунок 4.1).

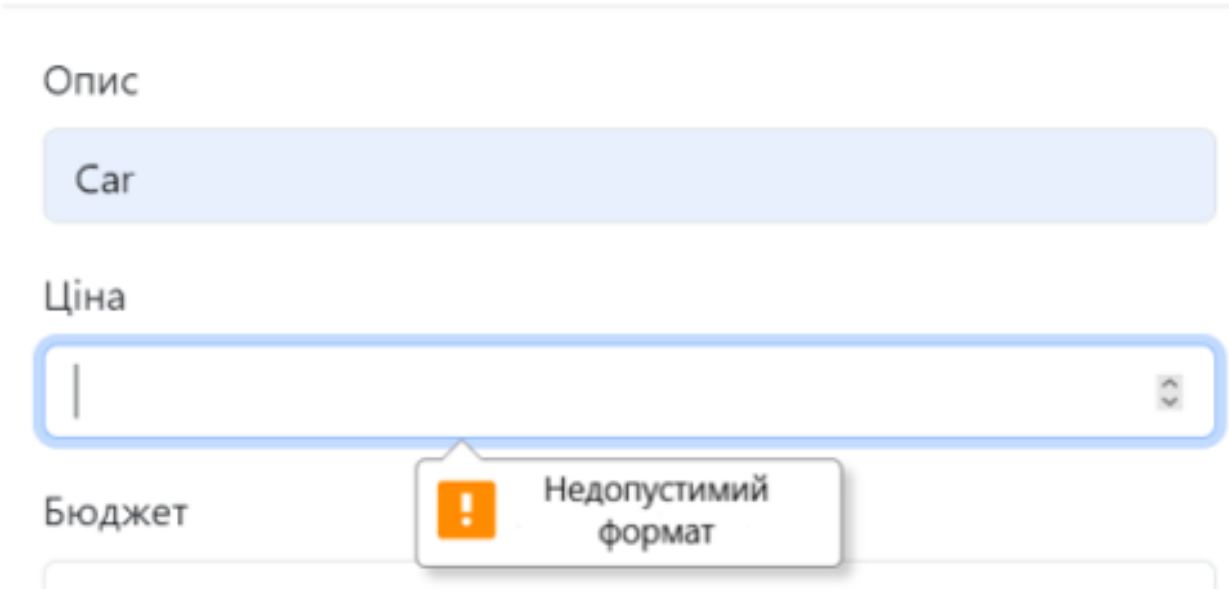


Рисунок 4.1 – Спроба вводу букв в поле для цифр

Далі під час проведення інтеграційного тестування було перевірено взаємодію між різними модулями системи та їх вплив на загальну продуктивність. Цей процес дозволив виявити потенційні конфлікти та невідповідності у роботі різних компонентів, що могли б вплинути на ефективність та стабільність системи. Детальний аналіз взаємодії допоміг нам вчасно виявити та виправити будь-які проблеми, забезпечуючи таким чином безперебійну та оптимальну роботу всієї системи.

У ході подальшого тестування було виявлено що на круговій діаграмі деякі кольори змішуються з іншими і не зовсім зрозуміло які витрати відносяться до конкретного сектору діаграми. Тому було прийнято рішення вибрати більш контрастні кольори та додати відображення категорії при наведені на кожен сектор діаграми (див. рисунок 4.2).

Завершальним етапом тестування стало комплексне тестування, під час якого вже тестувалася вся система в цілому. Здійснювалися різні сценарії використання, що моделювали реальні умови користування програмою. Також важливим аспектом було тестування системи на різних пристроях та під різними операційними системами з метою виявлення можливих проблем з сумісністю. Результати тестування наведено в таблиці 4.1.

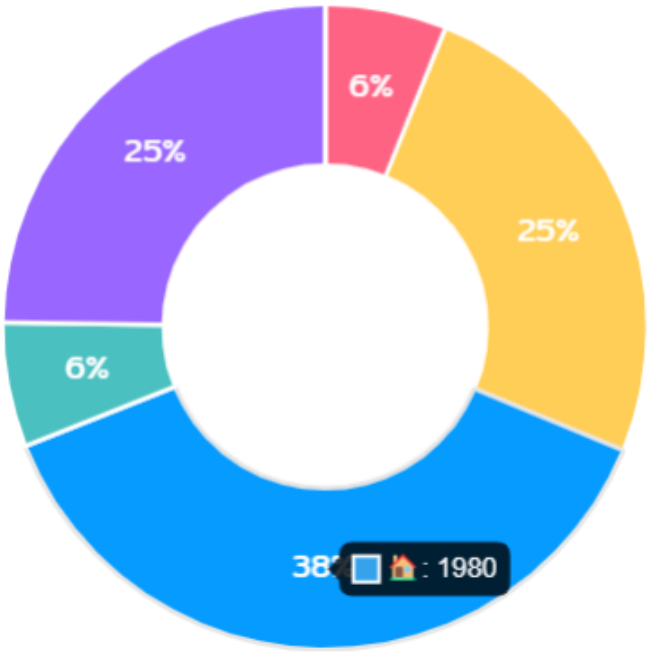


Рисунок 4.2 – Зображення діаграми

Таблиця 4.1– Результати тестування

№	Опис тесту	Кроки виконання	Очікуваний результат	Фактичний результат	Оцінка
1	2	3	4	5	6
1	Вхід в систему Google Pay	1. Відкрити додаток	Відкриття додатка	Відображення особистого кабінету Google Pay	Успішно
		2. Ввести дані для входу	Відображення особистого кабінету		
		3. Натиснути кнопку входу	Успішний вхід в систему		

Продовження таблиці 4.1

1	2	3	4	5	6
2	Додавання транзакції у додатку	1. Відкрити додаток	Відображення основного екрану додатку	Відображення основного екрану додатку	Успішно
		2. Вибрати опцію "Додати транзакцію"	Відкриття форми додавання транзакції	Відкриття форми додавання транзакції	Успішно
		3. Ввести дані про транзакцію	Успішне додавання транзакції	Успішне додавання транзакції	Успішно
		4. Зберегти транзакцію	Транзакція збережена у системі	Транзакція збережена у системі	Успішно
3	Аналіз витрат за певний період	1. Відкрити додаток	Відображення основного екрану додатку	Відображення основного екрану додатку	Успішно
		2. Вибрати період для аналізу	Відображення статистики за обраний період	Відображення статистики за обраний період	Успішно
		3. Переглянути витрати за обраний період	Коректне відображення статистики	Коректне відображення статистики	Успішно

Продовження таблиці 4.1

1	2	3	4	5	6
4	Видалення транзакції	1. Відкрити додаток	Відображення основного екрану додатку	Відображення основного екрану додатку	Успішно
		2. Знайти потрібну транзакцію	Транзакція знайдена у списку	Транзакція знайдена у списку	Успішно
		3. Вибрати опцію "Видалити"	Підтвердження видалення транзакції	Підтвердження видалення транзакції	Успішно
		4. Підтвердити видалення	Транзакція успішно видалена	Транзакція успішно видалена	Успішно

Після проведення комплексного тестування було отримано підтвердження високої функціональності та надійності розробленої мобільної системи управління та моніторингу особистих фінансів. Були ідентифіковані деякі помилки та недоліки, які були усунені згідно з вимогами та вимогами користувачів. Цей процес виправлення допоміг забезпечити стабільну та безперебійну роботу програми, що в свою чергу забезпечує користувачам комфортне та ефективне використання системи.

4.2 Розробка інструкції користувача

Інструкція користувача не лише описує функціонал програмного продукту, але й встановлює технічні вимоги для його оптимального функціонування. У цьому документі визначені як мінімальні, так і рекомендовані конфігурації, необхідні для запуску програми. Ця інформація включає в себе

деталі щодо операційних систем, обсягу оперативної пам'яті, типу та швидкості процесора та інших параметрів, які можуть впливати на ефективність роботи програми. Більш того, в таблиці 4.2 наведено рекомендації щодо оптимальних характеристик пристрою для кращого використання програмного продукту. Такий підхід дозволяє користувачам забезпечити необхідні умови для стабільної та ефективної роботи програми, забезпечуючи їм максимальний комфорт та задоволення від використання продукту.

Таблиця 4.2 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	**Рекомендовано**	**Мінімально**
Операційна система	Android 10.0	Android 7.0
Процесор	Octa-core 2.0 GHz	Quad-core 1.2 GHz
Оперативна пам'ять	4 ГБ	2 ГБ
Внутрішня пам'ять	100 МБ	100 МБ
Екран	Роздільна здатність HD (1280 x 720 пікселів) або вище, діагональ 5,5 дюйма або більше	Роздільна здатність HD (1280 x 720 пікселів) або вище, діагональ 4,5 дюйма або більше

Після запуску додатку користувач повинен авторизуватися в системі або зареєструватись якщо немає акаунта. Потім він опиняється на головному екрані додатку де він отримує доступ до всього функціоналу додатку а саме додавання витрат та надходжень за допомогою відповідних кнопок “Додати витрати” та “Додати доходи”. За допомогою кнопок користувач має можливість як зберегти всі введені дані в Excel файл так і зчитати дані з зовнішнього файлу. Внизу сторінки у користувача є список витрат та надходжень де він може редагувати або видаляти їх

У кожному віконці бюджетів користувач має можливість здійснювати ряд дій, включаючи перегляд витрат. Натиснувши кнопку "Витрати", відкривається

вікно, де користувач може детально ознайомитися з усіма витратами, що належать до цієї категорії. Крім того, в цьому вікні надається можливість видалення окремих витрат або видалення всієї категорії в цілому. Ця функція дозволяє користувачу легко та зручно керувати своїми фінансами, надаючи повний контроль над витратами та категоріями, що значно сприяє ефективному управлінню бюджетом.

4.3 Висновки

У четвертому розділі було приділено час та увагу тестуванню модулів програмного забезпечення для мобільної системи управління та моніторингу особистих фінансів. Це етап вирішальної важливості, оскільки від якості тестування залежить надійність та ефективність роботи програмного продукту.

Крім того, розроблено докладну інструкцію користувача, яка допоможе новим користувачам ознайомитися з програмним продуктом та його можливостями. Це включає в себе посібник по початковій експлуатації, пояснення основних функцій та функціональності програми, а також кроки з встановлення та налаштування.

Ця інструкція є важливим елементом для забезпечення зручності та ефективності використання програмного продукту для кінцевого користувача. Вона допоможе користувачам швидко та ефективно освоїти програму, зменшуючи час на навчання та підвищуючи загальну задоволеність від користування.

ВИСНОВКИ

У ході бакалаврської дипломної роботи розроблено мобільний застосунок аналізу грошових витрат у системі Google Pay. В процесі виконання роботи розв’язані такі задачі проекту: проведено аналіз існуючих мобільних застосунків і систем для аналізу грошових витрат, розроблено метод контролю грошових витрат і надходжень користувача, розроблено базу данних грошових витрат у системі Google Pay, розроблено алгоритми аналізу грошових витрат, проведено тестування та оцінку ефективності розробленої мобільної системи в реальних умовах.

У процесі досліджень використано такі теорії та методи: теорія чисел та чисельних методів, комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень, теорія баз даних, UML-діаграми, прототипування, методи оптимізації даних, методи тестування та оцінки ефективності системи в реальних умовах для аналізу зручності використання та надійності управління фінансовими даними.

Подальшого розвитку набув метод контролю грошових витрат і надходжень, який надає можливість автоматично визначати рух коштів на рахунках за допомогою системи безконтактного способу оплати Google Pay, що дозволило оптимізувати процес витрат, стабілізувати рівень залишків на рахунках, і як наслідок покращити фінансовий стан користувача.

У першому розділі було проведено детальний та ґрунтовний аналіз різноманітних програмних рішень, призначених для управління особистими фінансами. Серед цих рішень були розглянуті такі популярні додатки, як Mint, Personal Capital, YNAB (You Need a Budget), PocketGuard та Monefy. Кожен з цих додатків має свої унікальні особливості, переваги та недоліки, які були ретельно досліджені та проаналізовані у рамках даного розділу.

У другому розділі проведено детальний аналіз вхідних і вихідних даних, необхідних для ефективної роботи мобільної системи управління та моніторингу особистих фінансів. Встановлено, що для оптимального функціонування додатку

необхідно обробляти широкий спектр особистих та фінансових даних користувачів, включаючи інформацію про доходи, витрати, банківські рахунки, інвестиції, а також плановані фінансові цілі та бюджети. На виході система повинна надавати детальні звіти, аналітичні дані та інтерактивні функції для взаємодії з користувачем. Цей підхід забезпечує створення мобільного застосунку, який максимально відповідає потребам користувачів і сприяє покращенню їх фінансового здоров'я.

Розроблено модель роботи програмного продукту за допомогою UML діаграми. Модель надає чітке уявлення про функцірнування застосунку у реальному середовищі.

У третьому розділі було детально розглянуто вибір технологій, мови програмування, бібліотек та інструментів, що використовувались під час розробки застосунку. Після ретельного вивчення вимог і можливостей було прийнято рішення використовувати Visual Studio Code як основне середовище розробки, а також мову програмування JavaScript. Використання Visual Studio Code забезпечує ефективне керування проектом завдяки широкому вибору інструментів, функцій та розширень, які допомагають при написанні, налагодженні та форматуванні коду. Вибір JavaScript обумовлений його універсальністю та широкою підтримкою в екосистемі веб-розробки, що дозволяє створювати як фронтенд, так і бекенд додатку. Для реалізації графічної частини інтерфейсу було використано бібліотеку Bootstrap CSS, яка забезпечує сучасний, адаптивний та зручний у використанні дизайн компонентів. Bootstrap надає широкий набір готових стилів і компонентів, що значно прискорює розробку інтерфейсу користувача, робить його більш консистентним і професійним на вигляд. Також було інтегровано кілька інструментів для управління станом додатку та обробки даних, таких як Redux, що дозволяє ефективно управляти станом програми, та Axios для зручної роботи з API-запитами. Використання цих технологій та інструментів забезпечує високу продуктивність, зручність у розробці та подальшому обслуговуванні додатку, роблячи його надійним та функціональним для кінцевих користувачів.

У четвертому розділі було проведено комплексне тестування всіх модулів програмного застосунку мобільної системи управління та моніторингу особистих фінансів. Тестування включало функціональне та інтеграційне тестування, а також тестування відповідності вимогам і тестування безпеки для забезпечення стабільності та надійності додатку. Були створені різні сценарії тестування, що охоплювали усі можливі варіанти використання програмного застосунку, щоб виявити та усунути всі можливі помилки та недоліки.

Крім того, у цьому розділі була розроблена інструкція користувача, яка детально пояснює користувачам, як користуватися програмним застосунком для досягнення їхніх фінансових цілей. Інструкція включає пояснення основних функцій, методів введення даних, керування бюджетом, використання аналітичних звітів та інші аспекти, які допомагають користувачам максимально ефективно використовувати програму. Такий підхід забезпечує зручність взаємодії з програмою та підвищує її корисність для кінцевих користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Finance Database [Електронний ресурс] – Режим доступу до ресурсу https://www.jeroenbouma.com/projects/financedatabase (дата звернення: 04.04.2024)
2. Трайтека Н. О. Розробка мобільного додатку для управління та моніторингу особистих фінансів // Матеріали Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 2024. [Електронний ресурс] Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21677
3. Що таке фінансова грамотність [Електронний ресурс] – Режим доступу до ресурсу: https://spe.org.ua/blog/groshi/shcho-take-finansova-hramotnist/ (дата звернення: 07.04.2024)
4. Що таке шифрування та як воно працює? [Електронний ресурс] – Режим доступу до ресурсу https://www.kingston.com/ua/blog/data-security/what-is-encryption (дата звернення: 07.04.2024)
5. Що таке двофакторна автентифікація або 2FA? [Електронний ресурс] – Режим доступу до ресурсу: https://experience.dropbox.com/uk-ua/resources/what-is-2fa (дата звернення: 08.04.2024)
6. MINT finance [Електронний ресурс] – Режим доступу до ресурсу: https://mint.intuit.com (дата звернення: 09.04.2024)
7. Personal Capital [Електронний ресурс] – Режим доступу до ресурсу: https://www.empower.com/empower-personal-wealth-transition (дата звернення: 09.04.2024)
8. YNAB [Електронний ресурс] – Режим доступу до ресурсу: https://www.ynab.com (дата звернення: 09.04.2024)
9. Pocket Guard [Електронний ресурс] – Режим доступу до ресурсу: https://pocketguard.com (дата звернення: 09.04.2024)

10. Monefy [Електронний ресурс] – Режим доступу до ресурсу: https://monefy.com.ua (дата звернення: 09.04.2024)
11. Як будувати UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/ (дата звернення: 18.04.2024)
12. DrawIO [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net (дата звернення: 18.04.2024)
13. Таблиці та бази даних все, що потрібно знати [Електронний ресурс] – Режим доступу до ресурсу: https://brander.ua/blog/tablytsi-ta-bazy-danykh-vse-shcho-potribno-znaty (дата звернення: 28.04.2024)
14. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs (дата звернення: 30.04.2024)
15. Boostrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 3.05.2024)
16. Box Icons [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 3.05.2024)
17. Chart JS [Електронний ресурс] – Режим доступу до ресурсу: https://www.chartjs.org/docs/latest/getting-started/ (дата звернення: 3.05.2024)
18. CountUp JS [Електронний ресурс] – Режим доступу до ресурсу: https://inorganik.github.io/countUp.js/ (дата звернення: 3.05.2024)
19. Sweet Alert JS [Електронний ресурс] – Режим доступу до ресурсу: https://sweetalert2.github.io (дата звернення: 3.05.2024)
20. Sheet JS [Електронний ресурс] – Режим доступу до ресурсу: https://docs.sheetjs.com/docs/ (дата звернення: 3.05.2024)
21. Що таке тестування [Електронний ресурс] – Режим доступу до ресурсу: https://university.sigma.software/what-is-software-testing/ (дата звернення: 10.05.2024)
22. Методичні вказівки до виконання бакалаврських кваліфікаційних

робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка мобільного застосунку аналізу грошових витрат у системі Google Pay» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. каф. О. М. Рейда
«12» березня 2024 р.
Виконав:
_____ студент гр. 3ПІ-20б Н. О. Трайтека
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка мобільного застосунку аналізу грошових витрат у системі Google Pay».

Галузь застосування – фінансова.

2. Підстава для розробки

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

3. Мета та призначення розробки

Метою дослідження є розробка інтуїтивно зрозумілої та ефективної системи управління особистими фінансами, що забезпечує точне відслідковування доходів та витрат користувачів, зручне імпортування та експортування даних у форматі Excel, а також візуалізацію фінансової інформації за допомогою графіків.

4. Вихідні дані для проведення НДР

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

1. Філліпс, Б., Стюарт, К., Марсікано, К. Android Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2019. 624 с.
2. Скін, Дж., Ґрінгалх, Д. Kotlin Programming: The Big Nerd Ranch Guide. Атланта: Big Nerd Ranch, 2018. 480 с.
3. Джонсон, Джеф, Фінн, Кейт. Designing User Interfaces for an Aging Population: Towards Universal Design. Амстердам: Morgan Kaufmann, 2017. 304 с.

5. Технічні вимоги

Для розробки системи управління фінансами використовуються мова програмування JavaScript, бібліотеки для роботи з Excel файлами, для створення

графіків, і Bootstrap CSS для інтерфейсу користувача. Вхідними даними є інформація про доходи та витрати, включаючи дату транзакції, назву, суму, категорію та тип, а також файли формату Excel, які містять ці дані для імпорту. Системні вимоги включають будь-яку сучасну операційну систему, сучасний веб-браузер, мінімум 2 ГБ оперативної пам'яті, 50 МБ простору на жорсткому диску, та доступ до Інтернету для завантаження зовнішніх ресурсів.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

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

8. Вимоги до рівня уніфікації та стандартизації

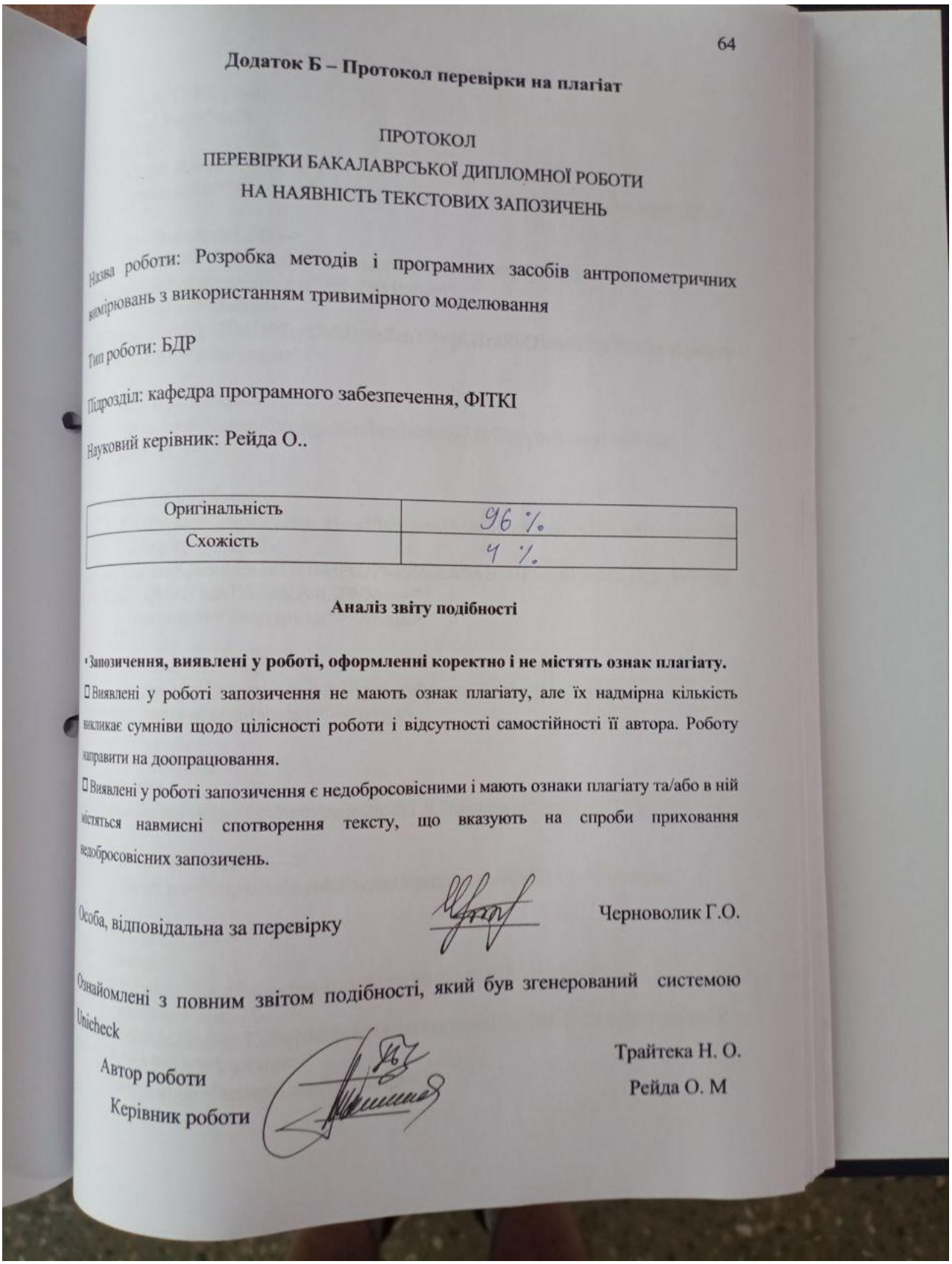
При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку застосунків для моніторингу фінансових витрат	13.03.24 –22.03.24
2	Розробка методів аналізу грошових витрат	13.03.24 –22.03.24
3	Розробка алгоритмів роботи програмного забезпечення	25.03.24 –19.04.24
4	Розробка програмного забезпечення	22.04.24 –10.05.24
5	Тестування програмного забезпечення	13.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	27.05.24 – 03.06.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.



Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка методів і програмних засобів антропометричних вимірювань з використанням тривимірного моделювання

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Рейда О..

Оригінальність	96 %
Схожість	4 %

Аналіз звіту подібності

- **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

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

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

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Трайтека Н. О.

Керівник роботи Рейда О. М

Додаток В – Лістинг програми

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKm
P1" crossorigin="anonymous" />

  <!-- Box Icons -->
  <link href="https://unpkg.com/boxicons@2.0.7/css/boxicons.min.css"
rel="stylesheet" />

  <!-- Chart JS -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"
    integrity="sha512-
d9xgZrVZpmmQlfonhQUvTR7lMPtO7NkZMkA0ABN3PHCbKA5nqylQ/yWlFAy
Y6hYgdF1Qh6nYiuADWwKB4C2WSw=="
    crossorigin="anonymous"></script>

  <!-- Chart JS Plugin -->
  <script src="https://cdn.jsdelivr.net/gh/emn178/chartjs-plugin-
labels/src/chartjs-plugin-labels.js"></script>

  <!-- CountUp JS -->
  <script
src="https://cdn.jsdelivr.net/npm/countup@1.8.2/dist/countUp.min.js"></script>

  <!-- Sweet Aleart JS -->
  <script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>

  <!-- Sheet JS -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.16.9/xlsx.mini.min.js"
    integrity="sha512-
edsVBJ+Mhf6lc2Sm3k87X8q794hL9g++dgr4yetkaHJfLJfz6e6SG5wq921i2YHjQC
AtJ+91ZzbhxP+L9WVwOA=="
    crossorigin="anonymous"></script>
```

```
        <!-- File Save JS -->
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.0/FileSaver.min.js"
          integrity="sha512-
csNcFYJniKjJxRWRV1R7fvnXrycHP6qDR21mgz1ZP55xY5d+aHLfo9/FcGDQLfn
2IfngbAHd8LdfsagcCqgTcQ=="
          crossorigin="anonymous"></script>

        <!-- Style -->
        <link rel="stylesheet" href="style.css" />

        <title>My Budget Tracker</title>
      </head>

      <body>
        <!-- Navbar -->
        <nav class="navbar sticky-top navbar-expand-lg navbar-light bg-light">
          <div class="container-fluid">
            <a class="navbar-brand" href="#">My Budget Tracker 💰💸</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbar"
              aria-controls="navbar" aria-expanded="false" aria-label="Toggle
navigation">
              <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbar">
              <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                  <a class="nav-link active" aria-current="page" href="#"></a>
                </li>
                <li class="nav-item dropdown">
                  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown"
                    aria-expanded="false">
                    Витрати
                  </a>
                  <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <li>
                      <a class="dropdown-item" href="#" data-bs-toggle="modal" data-bs-
target="#add-expend-item-modal"><i
                          class="bx bx-plus-circle"></i> Додати Витрати</a>
                    </li>
                  </ul>
                </li>
                <li class="nav-item dropdown">
```

```
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown"
                  aria-expanded="false">
                  Доходи
                </a>
                <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                  <li>
                    <a class="dropdown-item" href="#" data-bs-toggle="modal" data-bs-
target="#add-income-item-modal"><i
                        class="bx bx-plus-circle"></i> Додати Доходи</a>
                  </li>
                </ul>
              </li>
            </ul>

            <ul class="navbar-nav">
              <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle active" href="#"
id="navbarDropdown" role="button"
                  data-bs-toggle="dropdown" aria-expanded="false">
                  Привіт👋, Nikita
                </a>
                <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                  <li>
                    <a class="dropdown-item" href="#"><i class="bx bx-log-out"></i>
Вийти</a>
                  </li>
                </ul>
              </li>
            </ul>

            <img class="img-profile" src="./images/profile-pic.png" />
          </div>
        </div>
      </nav>
      <!-- Navbar -->

      <!-- Container -->
      <div class="container-fluid">
        <!-- Cards Content Row -->
        <div class="row mt-3">
          <!-- Card-1 -->
          <div class="col-lg-3 col-sm-6 col-12 mt-2">
            <div class="card">
              <div class="card-body text-danger">
```

```
        <h5 class="card-title text-muted">💸Витрати Сьогодні</h5>
        <h3 class="card-text counter">0</h3>
      </div>
    </div>
  </div>
  <!-- Card-1 -->

  <!-- Card-2 -->
  <div class="col-lg-3 col-sm-6 col-12 mt-2">
    <div class="card">
      <div class="card-body text-danger">
        <h5 class="card-title text-muted">Витрати(Місяць)</h5>
        <h3 class="card-text counter">0</h3>
      </div>
    </div>
  </div>
  <!-- Card-2 -->

  <!-- Card-3 -->
  <div class="col-lg-3 col-sm-6 col-12 mt-2">
    <div class="card">
      <div class="card-body text-danger">
        <h5 class="card-title text-muted">💸Витрати(Рік)</h5>
        <h3 class="card-text counter">0</h3>
      </div>
    </div>
  </div>
  <!-- Card-3 -->

  <!-- Card-4 -->
  <div class="col-lg-3 col-sm-6 col-12 mt-2">
    <div class="card">
      <div class="card-body text-success">
        <h5 class="card-title text-muted">💰Дохід(Рік)</h5>
        <h3 class="card-text counter">0</h3>
      </div>
    </div>
  </div>
  <!-- Card-4 -->
</div>
<!-- Cards Content Row -->

<!-- Charts Content Row -->
<div class="row mt-3">
```

```
            <!-- Charts Header -->
            <h3>Витрати 💸</h3>
            <!-- Charts Header -->

            <!-- Line Chart -->
            <div class="col-lg-9 col-sm-12 col-12">
              <h4 class="text-muted">По Днях</h4>
              <div class="card" style="border: 0px">
                <div id="chart-days-body" class="card-body" style="width: 100%;
height: 100%">
                  <canvas id="chart-days" height="350"></canvas>
                </div>
              </div>
            </div>
            <!-- Line Chart -->

            <!-- Pie Chart -->
            <div class="col-lg-3 col-sm-12 col-12">
              <h4 class="text-muted">Категорії</h4>
              <div class="card" style="border: 0px">
                <div id="chart-categories-body" class="card-body" style="width: 100%;
height: 100%">
                  <canvas id="chart-categories" width="100" height="350"></canvas>
                </div>
              </div>
            </div>
            <!-- Pie Chart -->
          </div>
          <!-- Charts Content Row -->

          <!-- List Contents Row -->
          <div class="row mt-3">
            <!-- Header List -->
            <h3>Витрати💸 & Доходи💰</h3>
            <!-- Header List -->

            <!-- Month Select -->
            <div class="row mt-2 col-lg-6">
              <div class="input-group">
                <select id="month-select" class="form-select">
                  <option selected disabled>Виберіть Місяць</option>
                  <option value="01">Січень</option>
                  <option value="02">Лютий</option>
                  <option value="03">Березень</option>
```

```
                <option value="04">Квітень</option>
                <option value="05">Травень</option>
                <option value="06">Червень</option>
                <option value="07">Липень</option>
                <option value="08">Серпень</option>
                <option value="09">Вересень</option>
                <option value="10">Жовтень</option>
                <option value="11">Листопад</option>
                <option value="12">Грудень</option>
              </select>
              <button id="import" type="button" class="btn btn-outline-primary btn-
import">
                Завантажити <i class="bx bxs-file-import"></i>
              </button>
              <button id="export" type="button" class="btn btn-outline-primary btn-
export">
                Скачати <i class="bx bxs-save"></i>
              </button>
            </div>
          </div>
          <!-- Month Select -->

          <!-- List Table -->
          <div class="row mt-2">
            <!-- Expend List -->
            <div class="col-lg-6 col-sm-12 col-12 mt-2">
              <h4 class="text-muted">Список Витрат 📜</h4>
              <table id="expend-table" class="table table-striped">
                <thead>
                  <tr>
                    <th>Дата</th>
                    <th>Назва</th>
                    <th class="text-end">Сума</th>
                    <th class="text-center">Категорія</th>
                    <th class="text-center">Редагувати</th>
                  </tr>
                </thead>
                <tbody id="expend-items">
                  <!-- Dynamic Data wiht JS -->
                </tbody>
              </table>
            </div>
            <!-- Expend List -->

            <!-- Income List -->
```

```
            <div class="col-lg-6 col-sm-12 col-12 mt-2">
              <h4 class="text-muted">Список доходів 📜</h4>
              <table id="income-table" class="table table-striped">
                <thead>
                  <tr>
                    <th>Дата</th>
                    <th>Назва</th>
                    <th class="text-end">Сума</th>
                    <th class="text-center">Редагувати</th>
                  </tr>
                </thead>
                <tbody id="income-items">
                  <!-- Dynamic Data wiht JS -->
                </tbody>
              </table>
            </div>
            <!-- Income List -->
          </div>
          <!-- List Table -->
        </div>
        <!-- List Contents Row -->
      </div>
      <!-- Container -->

      <!-- Footer -->
      <footer class="footer mt-auto py-3 bg-light">
        <div class="container-fluid text-center">

        </div>
      </footer>
      <!-- Footer -->

      <!-- Add Expend Item Modal -->
      <div class="modal fade" id="add-expend-item-modal" data-bs-
backdrop="static" data-bs-keyboard="false" tabindex="-1"
        aria-hidden="true">
        <div class="modal-dialog modal-dialog-centered">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title">Додати Витрати</h5>
            </div>
            <div class="modal-body">
              <form id="expend-item-form">
                <div class="mb-3">
                  <label for="expend-date" class="form-label">Дата</label>
```

```
                <input type="date" class="form-control" id="expend-date" />
              </div>
              <div class="mb-3">
                <label for="expend-title" class="form-label">Опис</label>
                <input type="text" class="form-control" id="expend-title"
placeholder="Витрата" />
              </div>
              <div class="mb-3">
                <label for="expend-cost" class="form-label">Сума</label>
                <input type="number" min="0" class="form-control" id="expend-
cost" placeholder="Сума" />
              </div>
              <div class="mb-3">
                <label>Категорія</label>
              </div>
              <div class="form-check form-check-inline">
                <input class="form-check-input" type="radio" name="category"
id="food-category" value="🍔"
                  checked="checked" />
                <label class="form-check-label" for="food-category">🍔</label>
              </div>
              <div class="form-check form-check-inline">
                <input class="form-check-input" type="radio" name="category"
id="party-category" value="🍻" />
                <label class="form-check-label" for="party-category">🍻</label>
              </div>
              <div class="form-check form-check-inline">
                <input class="form-check-input" type="radio" name="category"
id="house-category" value="🏠" />
                <label class="form-check-label" for="house-category">🏠</label>
              </div>
              <div class="form-check form-check-inline">
                <input class="form-check-input" type="radio" name="category"
id="shopping-category" value="🛒" />
                <label class="form-check-label" for="shopping-
category">🛒</label>
              </div>
              <div class="form-check form-check-inline">
                <input class="form-check-input" type="radio" name="category"
id="car-category" value="🚗" />
                <label class="form-check-label" for="car-category">🚗</label>
              </div>
              <div class="form-check form-check-inline">
```

```
                <input class="form-check-input" type="radio" name="category"
id="other-category" value="❓" />
                <label class="form-check-label" for="other-category">❓</label>
              </div>
            </form>
          </div>
          <div class="modal-footer justify-content-start">
            <button type="button" class="btn btn-primary" id="expend-item-submit"
data-id="">
              Додати 💸
            </button>
            <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal" id="expend-item-cancel">
              Відмінити
            </button>
          </div>
        </div>
      </div>
      <!-- Add Expend Item Modal -->

      <!-- Add Income Item Modal -->
      <div class="modal fade" id="add-income-item-modal" data-bs-
backdrop="static" data-bs-keyboard="false" tabindex="-1"
        aria-labelledby="staticBackdropLabel" aria-hidden="true">
        <div class="modal-dialog modal-dialog-centered">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="staticBackdropLabel">Додати
Доходи</h5>
            </div>
            <div class="modal-body">
              <form id="income-item-form">
                <div class="mb-3">
                  <label for="income-date" class="form-label">Дата</label>
                  <input type="date" class="form-control" id="income-date" />
                </div>
                <div class="mb-3">
                  <label for="income-title" class="form-label">Опис</label>
                  <input type="text" class="form-control" id="income-title"
placeholder="Дохід" />
                </div>
                <div class="mb-3">
                  <label for="income-cost" class="form-label">Сума</label>
```

```
                    <input type="number" min="0" class="form-control" id="income-
cost" placeholder="Сума" />
                  </div>
                </form>
              </div>
              <div class="modal-footer justify-content-start">
                <button type="button" class="btn btn-primary" id="income-item-
submit" data-id="">
                  Додати 💰
                </button>
                <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal" id="income-item-cancel">
                  Відмінити
                </button>
              </div>
            </div>
          </div>
        </div>
        <!-- Add Income Modal -->

        <!-- Option 1: Bootstrap Bundle with Popper -->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"
          integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW
"
          crossorigin="anonymous"></script>

        <!-- App JS -->
        <script src="app.js"></script>
      </body>

      </html>


      // Expend Item Class
      class ExpendItem {
        constructor(id, date, title, cost, category) {
          this.id = id;
          this.date = date;
          this.title = title;
          this.cost = cost;
          this.category = category;
        }
      }
```

```
// Income Item Class
class IncomeItem {
  constructor(id, date, title, cost) {
    this.id = id;
    this.date = date;
    this.title = title;
    this.cost = cost;
  }
}

// UI Class
class UI {
  constructor() {
    // Expend and Income table body
    this.expendItems = document.getElementById("expend-items");
    this.incomeItems = document.getElementById("income-items");
    // Expend inputs
    this.expendDateInput = document.getElementById("expend-date");
    this.expendTitleInput = document.getElementById("expend-title");
    this.expendCostInput = document.getElementById("expend-cost");
    this.expendCategoryInput = document.getElementById("food-category");
    // Income inputs
    this.incomeDateInput = document.getElementById("income-date");
    this.incomeTitleInput = document.getElementById("income-title");
    this.incomeCostInput = document.getElementById("income-cost");
    // Expend and Income submit button
    this.expendSubmitBtn = document.getElementById("expend-item-submit");
    this.incomeSubmitBtn = document.getElementById("income-item-submit");
    // Add Expend and Add Income modal
    this.expendModal = document.getElementById("add-expend-item-modal");
    this.incomeModal = document.getElementById("add-income-item-modal");
  }

  // Add Expend Item
  addExpendToList(expendItem) {
    // Create tr element
    const row = document.createElement("tr");

    // Insert cols
    row.innerHTML = `
        <td>${expendItem.date}</td>
        <td>${expendItem.title}</td>
        <td class="text-end">${parseInt(expendItem.cost).toLocaleString(
      "th-TH"
```

```
  )}</td>
     <td class="text-center">${expendItem.category}</td>
     <td class="text-center">
        <a href="#"
        class="edit"
        data-bs-toggle="modal"
        data-bs-target="#add-expend-item-modal"
        data-id="${expendItem.id}"><i class='bx bx-pencil'></i></a>
        <a href="#" class="delete" data-id="${expendItem.id
    }"><i class='bx bx-trash'></i></a>
     </td>
    `;

  // Write tr element on the top of expend table body
  this.expendItems.insertBefore(row, this.expendItems.childNodes[0]);
}

// Add Income Item
addIncomeToList(incomeItem) {
  // Create tr element
  const row = document.createElement("tr");

  // Insert cols
  row.innerHTML = `
     <td>${incomeItem.date}</td>
     <td>${incomeItem.title}</td>
     <td class="text-end">${parseInt(incomeItem.cost).toLocaleString(
    "th-TH"
  )}</td>
     <td class="text-center">
        <a href="#"
        class="edit"
        data-bs-toggle="modal"
        data-bs-target="#add-income-item-modal"
        data-id="${incomeItem.id}"><i class='bx bx-pencil'></i></a>
        <a href="#" class="delete" data-id="${incomeItem.id
    }"><i class='bx bx-trash'></i></a>
     </td>
    `;

  // Write tr element on the top of income table body
  this.incomeItems.insertBefore(row, this.incomeItems.childNodes[0]);
}

// Delete item from list
```

```
deleteItem(target) {
  if (target.classList.contains("delete")) {
    // Delete item from table body
    target.parentElement.parentElement.remove();
  }
}

// Clear expend form fields
clearExpendFields() {
  // Set default date
  this.expendDateInput.value = new Date().toISOString().split("T")[0];
  this.expendTitleInput.value = "";
  this.expendCostInput.value = "";
  // Set to default checked
  this.expendCategoryInput.checked = true;
}

// Clear income form fields
clearIncomeFields() {
  // Set default date
  this.incomeDateInput.value = new Date().toISOString().split("T")[0];
  this.incomeTitleInput.value = "";
  this.incomeCostInput.value = "";
}

// Change expend form to edit
changeExpendForm(type, id) {
  if (type === "edit") {
    this.expendSubmitBtn.textContent = "Edit";

    this.expendSubmitBtn.className = "btn btn-warning";

    let dataAtt = document.createAttribute("data-id");

    dataAtt.value = id;

    this.expendSubmitBtn.setAttributeNode(dataAtt);
  } else {
    this.expendSubmitBtn.textContent = "Add";

    this.expendSubmitBtn.className = "btn btn-primary";

    let dataAtt = document.createAttribute("data-id");

    dataAtt.value = id;
```

```
      this.expendSubmitBtn.setAttributeNode(dataAtt);
    }
  }

  // Change income form to edit
  changeIncomeForm(type, id) {
    if (type === "edit") {
      this.incomeSubmitBtn.textContent = "Edit";

      this.incomeSubmitBtn.className = "btn btn-warning";

      let dataAtt = document.createAttribute("data-id");

      dataAtt.value = id;

      this.incomeSubmitBtn.setAttributeNode(dataAtt);
    } else {
      this.incomeSubmitBtn.textContent = "Add";

      this.incomeSubmitBtn.className = "btn btn-primary";

      let dataAtt = document.createAttribute("data-id");

      dataAtt.value = id;

      this.incomeSubmitBtn.setAttributeNode(dataAtt);
    }
  }

  // Fill data to expend form fields
  fillExpendForm(item) {
    // Change to edit form
    this.changeExpendForm("edit", item.id);

    // Set edit item value to expend form
    this.expendDateInput.value = item.date;
    this.expendTitleInput.value = item.title;
    this.expendCostInput.value = item.cost.split(",").join("");

    document.querySelectorAll('input[name="category"]').forEach((category)
=> {
      if (category.value === item.category) {
        category.checked = true;
      }
```

```
    });
  }

  // Fill data to income form fields
  fillIncomeForm(item) {
    // Change to edit form
    this.changeIncomeForm("edit", item.id);

    // Set edit item value to income form
    this.incomeDateInput.value = item.date;
    this.incomeTitleInput.value = item.title;
    this.incomeCostInput.value = item.cost.split(",").join("");
  }

  // Clear expend table
  clearExpendTable() {
    while (this.expendItems.firstChild) {
      this.expendItems.removeChild(this.expendItems.firstChild);
    }
  }

  // Clear income table
  clearIncomeTable() {
    while (this.incomeItems.firstChild) {
      this.incomeItems.removeChild(this.incomeItems.firstChild);
    }
  }

  // Close expend modal
  closeExpendModal() {
    const expendModal = bootstrap.Modal.getInstance(this.expendModal);

    expendModal.hide();
  }

  // Close income modal
  closeIncomeModal() {
    const incomeModal = bootstrap.Modal.getInstance(this.incomeModal);

    incomeModal.hide();
  }
}

// Local Storage Class
class Store {
```

```
static getExpendItems() {
  let expendItems;

  if (localStorage.getItem("expendItems") === null) {
    expendItems = [];
  } else {
    expendItems = JSON.parse(localStorage.getItem("expendItems"));
  }

  return expendItems;
}

static getIncomeItems() {
  let incomeItems;

  if (localStorage.getItem("incomeItems") === null) {
    incomeItems = [];
  } else {
    incomeItems = JSON.parse(localStorage.getItem("incomeItems"));
  }

  return incomeItems;
}

static displayExpendItems() {
  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  // Filter expend items in month selected with present year
  let expendItemsMonth = Store.filterMonth(expendItems);

  // Ascending sort expend items by date
  expendItemsMonth.sort(function (a, b) {
    return new Date(a.date) - new Date(b.date);
  });

  expendItemsMonth.forEach(function (expendItem) {
    const ui = new UI();

    // Add expend items to UI
    ui.addExpendToList(expendItem);
  });
}

static displayIncomeItems() {
```

```
  // Get income items from local storage
  const incomeItems = Store.getIncomeItems();

  // Filter expend items in month selected with present year
  let incomeItemsMonth = Store.filterMonth(incomeItems);

  // Ascending sort income items by date
  incomeItemsMonth.sort(function (a, b) {
    return new Date(a.date) - new Date(b.date);
  });

  incomeItemsMonth.forEach(function (incomeItems) {
    const ui = new UI();

    // Add income items to UI
    ui.addIncomeToList(incomeItems);
  });
}

static addExpendItem(expendItem) {
  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  expendItems.push(expendItem);

  localStorage.setItem("expendItems", JSON.stringify(expendItems));
}

static addIncomeItem(incomeItem) {
  // Get income items from local storage
  const incomeItems = Store.getIncomeItems();

  incomeItems.push(incomeItem);

  localStorage.setItem("incomeItems", JSON.stringify(incomeItems));
}

static updateExpendItem(updatedItem) {
  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  expendItems.forEach(function (expendItem, index) {
    if (expendItem.id == updatedItem.id) {
      expendItems[index] = updatedItem;
    }
```

```
  });

  localStorage.setItem("expendItems", JSON.stringify(expendItems));
}

static updateIncomeItem(updatedItem) {
  // Get income items from local storage
  const incomeItems = Store.getIncomeItems();

  incomeItems.forEach(function (incomeItem, index) {
    if (incomeItem.id == updatedItem.id) {
      incomeItems[index] = updatedItem;
    }
  });

  localStorage.setItem("incomeItems", JSON.stringify(incomeItems));
}

static deleteExpendItem(id) {
  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  expendItems.forEach(function (expendItem, index) {
    if (expendItem.id == id) {
      expendItems.splice(index, 1);
    }
  });

  localStorage.setItem("expendItems", JSON.stringify(expendItems));
}

static deleteIncomeItem(id) {
  // Get income items from local storage
  const incomeItems = Store.getIncomeItems();

  incomeItems.forEach(function (incomeItem, index) {
    if (incomeItem.id == id) {
      incomeItems.splice(index, 1);
    }
  });

  localStorage.setItem("incomeItems", JSON.stringify(incomeItems));
}

// Filter items with present year
```

```
    static filterYear(items) {
      const year = new Date().getFullYear();

      let yearItems = items.filter((item) => item.date.split("-")[0] == year);

      return yearItems;
    }

    // Filter items with month select value and present year
    static filterMonth(items) {
      const month = document.getElementById("month-select").value;

      let yearItems = Store.filterYear(items);

      let monthItems = yearItems.filter((item) => item.date.split("-")[1] ==
month);

      return monthItems;
    }

    // Filter items with present day with month select value and present year
    static filterToDay(items) {
      const today = new Date().toISOString().split("T")[0];

      // Filter expend items in month selected with present year
      let monthItems = Store.filterMonth(items);

      let dayItems = monthItems.filter((item) => item.date == today);

      return dayItems;
    }

    // Write JSON data to sheet (xlsx)
    static writeToSheet(expend, income) {
      const month = document.getElementById("month-select").value;

      const worksheetExpend = XLSX.utils.json_to_sheet(expend);

      const worksheetIncome = XLSX.utils.json_to_sheet(income);

      const workbook = XLSX.utils.book_new();

      XLSX.utils.book_append_sheet(workbook, worksheetExpend, `Expend-
${month}`);
```

```
        XLSX.utils.book_append_sheet(workbook, worksheetIncome, `Income-${month}`);

        let workbookOut = XLSX.write(workbook, {
          bookType: "xlsx",
          bookSST: false,
          type: "array",
        });

        saveAs(
          new Blob([workbookOut], { type: "application/octet-stream" }),
          `Expend-Income-${month}.xlsx`
        );
      }

      static exportData() {
        // Get expend items from local storage
        const expendItems = Store.getExpendItems();

        // Filter expend items in month selected with present year
        let expendItemsMonth = Store.filterMonth(expendItems);

        expendItemsMonth.sort(function (a, b) {
          return new Date(a.date) - new Date(b.date);
        });

        // Get income items from local storage
        const incomeItems = Store.getIncomeItems();

        // Filter income items in month selected with present year
        let incomeItemsMonth = Store.filterMonth(incomeItems);

        incomeItemsMonth.sort(function (a, b) {
          return new Date(a.date) - new Date(b.date);
        });

        // Export json to sheet
        Store.writeToSheet(expendItemsMonth, incomeItemsMonth);
      }


    }

    // Show cards class
    class DrawCard {
```

```
// Counter Up Summary Number
counterUp() {
  let cardDatas = []; // [expendDay, expendMonth, expendYear, incomeYear]

  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  // Filter expend items in present year
  const expendItemsYear = Store.filterYear(expendItems);

  // Filter expend items in month selected with present year
  const expendItemsMonth = Store.filterMonth(expendItems);

  // Filter expend items in today with month selected and present year
  const expendItemsDay = Store.filterToDay(expendItems);

  // Sum expend cost in a days
  cardDatas.push(
    expendItemsDay.reduce((sum, item) => {
      return sum + parseInt(item.cost);
    }, 0)
  );

  // Sum expend cost in month
  cardDatas.push(
    expendItemsMonth.reduce((sum, item) => {
      return sum + parseInt(item.cost);
    }, 0)
  );

  // Sum expend cost in year
  cardDatas.push(
    expendItemsYear.reduce((sum, item) => {
      return sum + parseInt(item.cost);
    }, 0)
  );

  // Get income items from local storage
  const incomeItems = Store.getIncomeItems();

  // Filter expend items in month selected with present year
  const incomeItemsMonth = Store.filterMonth(incomeItems);

  // Sum income cost in year
  cardDatas.push(
```

```
      incomeItemsMonth.reduce((sum, item) => {
        return sum + parseInt(item.cost);
      }, 0)
    );

    const counterTexts = document.querySelectorAll(".counter");
    if (cardDatas !== undefined && cardDatas.length !== 0) {
      counterTexts.forEach(function (textEle, index) {
        new CountUp(textEle, 0, cardDatas[index]).start();
      });
    }
  }
}

// Show chart class
class DrawChart {
  constructor() {
    this.chartDays = document.getElementById("chart-days-body");
    this.chartCategory = document.getElementById("chart-categories-body");
  }
  // Get days in month
  static daysInMonth() {
    const month = document.getElementById("month-select").value;

    const year = new Date().getFullYear();

    // Get last day in month
    const lastDayInMonth = new Date(year, Number(month), 0).getDate();

    // Set days in month to [1,2,3,...,last day]
    const daysInMonthArr = [...Array(lastDayInMonth).keys()].map((i) =>
      (i + 1).toString()
    );

    return daysInMonthArr.map((day) => {
      if (day.length !== 2) {
        return "0" + day;
      } else {
        return day;
      }
    });
  }

  // Draw Bar Chart
  drawBarChart() {
```

```
// Get expend items from local storage
const expendItems = Store.getExpendItems();

// Filter expend items in month selected with present year
const expendItemsMonth = Store.filterMonth(expendItems);

// Set expend items to data for bar chart
let dataBarChart = [];

DrawChart.daysInMonth().forEach((day) => {
  // Sum cost in a days
  dataBarChart.push(
    expendItemsMonth
      .filter((item) => {
        return item.date.split("-")[2] == day;
      })
      .reduce((sum, item) => {
        return sum + parseInt(item.cost);
      }, 0)
  );
});

// Draw bar chart
const chartDays = document.getElementById("chart-days");
let ctx = chartDays.getContext("2d");
new Chart(ctx, {
  type: "bar",
  data: {
    labels: DrawChart.daysInMonth(),
    datasets: [
      {
        label: "💸",
        data: dataBarChart,
        backgroundColor: "rgba(54, 162, 235, 1)",
      },
    ],
  },
  options: {
    defaultFontSize: 14,
    defaultFontFamily: "Mitr",
    scales: {
      yAxes: [
        {
          ticks: {
            beginAtZero: true,
```

```
            max: Math.max(...dataBarChart) + 100,
            stepSize: 100,
            display: false,
          },
          gridLines: {
            display: true,
            drawBorder: false,
          },
        },
      ],
      xAxes: [
        {
          gridLines: {
            display: false,
          },
        },
      ],
    },
    legend: {
      display: false,
    },
    animation: {
      duration: 2000,
    },
    maintainAspectRatio: false,
    tooltips: false,
    plugins: {
      labels: {
        render: "value",
      },
    },
  },
});
}

// Draw Doughnut Chart
drawDoughnutChart() {
  // Get expend items from local sotrage
  const categorys = ["🍔", "👊", "🏠", "🛒", "🚗", "❓"];

  // Get expend items from local storage
  const expendItems = Store.getExpendItems();

  // Filter expend items in month selected with present year
  const expendItemsMonth = Store.filterMonth(expendItems);
```

```
// Set expend items to data for doughnut chart
let dataDoughnutChart = [];

categorys.forEach((category) => {
  // Sum cost in a category
  dataDoughnutChart.push(
    expendItemsMonth
      .filter((item) => {
        return item.category == category;
      })
      .reduce((sum, item) => {
        return sum + parseInt(item.cost);
      }, 0)
  );
});

// Draw doughnut chart
const chartCategories = document.getElementById("chart-categories");
let ctx = chartCategories.getContext("2d");
new Chart(ctx, {
  type: "doughnut",
  data: {
    datasets: [
      {
        data: dataDoughnutChart,
        backgroundColor: [
          "rgba(255, 99, 132, 1)",
          "rgba(255, 206, 86, 1)",
          "rgba(54, 162, 235, 1)",
          "rgba(75, 192, 192, 1)",
          "rgba(153, 102, 255, 1)",
          "rgba(255, 159, 64, 1)",
        ],
      },
    ],

    labels: categorys,
  },
  options: {
    defaultFontSize: 14,
    defaultFontFamily: "Mitr",
    legend: {
      labels: {
        fontSize: 16,
```

```
        },
      },
      animation: {
        duration: 2000,
      },
      maintainAspectRatio: false,
      plugins: {
        labels: { render: "percentage", fontColor: "white" },
      },
    },
  });
 }

 // Reset existing bar chart
 clearBarChart() {
  while (this.chartDays.firstChild) {
   this.chartDays.removeChild(this.chartDays.firstChild);
  }

  let canvas = document.createElement("canvas");
  canvas.id = "chart-days";
  canvas.height = 350;

  this.chartDays.appendChild(canvas);
 }

 // Reset existing doughnut chart
 clearDoughnutChart() {
  while (this.chartCategory.firstChild) {
   this.chartCategory.removeChild(this.chartCategory.firstChild);
  }

  let canvas = document.createElement("canvas");
  canvas.id = "chart-categories";
  canvas.width = 100;
  canvas.height = 350;

  this.chartCategory.appendChild(canvas);
 }
}

// DOM load event
document.addEventListener(
 "DOMContentLoader",
 (function () {
```

```
        // Set defalut date
        const date = new Date();
        const today = date.toISOString().split("T")[0];
        const month = date.toISOString().split("T")[0].split("-")[1];

        // Set date in expend modal to today
        document.getElementById("expend-date").value = today;

        // Set date in income modal to today
        document.getElementById("income-date").value = today;

        // Set month select to this month
        document.getElementById("month-select").value = month;

        // Instantiate DrawCard
        const card = new DrawCard();

        // Show cards value
        card.counterUp();

        // Instantiate DrawChart
        const chart = new DrawChart();
        // Show bar chart
        chart.drawBarChart();

        // Show doughnut chart
        chart.drawDoughnutChart();

        // Show expend items in expend table
        Store.displayExpendItems();

        // Show income items in income table
        Store.displayIncomeItems();
      })()
    );

    // Event listener for month selected change
    document.getElementById("month-select").addEventListener("change",
function () {
      // Instantiate UI
      const ui = new UI();

      // Instantiate DrawChart
      const chart = new DrawChart();
```

```
  // Reset existing bar chart
  chart.clearBarChart();

  // Draw bar chart
  chart.drawBarChart();

  // Reset existing doughnut chart
  chart.clearDoughnutChart();

  // Show doughnut chart
  chart.drawDoughnutChart();

  // Reset expend table
  ui.clearExpendTable();

  // Show expend items in expend table
  Store.displayExpendItems();

  // Reset income table
  ui.clearIncomeTable();

  // Show income items in income table
  Store.displayIncomeItems();
});

// Event listener for add or update expend item
document
  .getElementById("expend-item-submit")
  .addEventListener("click", function (e) {
    // Instantiate UI
    const ui = new UI();

    // Instantiate DrawChart
    const chart = new DrawChart();

    // Instantiate DrawCard
    const card = new DrawCard();

    // Add new expend item
    if (e.target.dataset.id === "") {
      // Get form values
      const date = document.getElementById("expend-date").value;
      const title = document.getElementById("expend-title").value;
      const cost = document.getElementById("expend-cost").value;
      const category =
```

```
document.querySelector('input[name="category"]:checked')
            .value;

          // Get time stamp for id
          const id = new Date().getTime();

          // Indtantiate Expend Item
          const expendItem = new ExpendItem(id, date, title, cost, category);

          // Validate
          if (title === "" || cost === "" || category === "") {
            // Error aleart
            Swal.fire({
              icon: "error",
              title: "Oops...",
              text: "Please fill in all fields",
            });
          } else {
            // Add expend item to local storage
            Store.addExpendItem(expendItem);

            // Reset expend table
            ui.clearExpendTable();

            // Show expend items in expend table
            Store.displayExpendItems();

            // Show cards value
            card.counterUp();

            // Reset existing bar chart
            chart.clearBarChart();

            // Draw bar chart
            chart.drawBarChart();

            // Reset existing doughnut chart
            chart.clearDoughnutChart();

            // Show doughnut chart
            chart.drawDoughnutChart();

            // Complete aleart
            Swal.fire({
              icon: "success",
```

```
            title: "Your expend item has been saved",
            showConfirmButton: false,
            timer: 1500,
          });

          // Clear fields
          ui.clearExpendFields();

          // Close modal
          ui.closeExpendModal();
        }
      }
      // Update expend item
      else {
        // Get form values
        const date = document.getElementById("expend-date").value;
        const title = document.getElementById("expend-title").value;
        const cost = document.getElementById("expend-cost").value;
        const category =
document.querySelector('input[name="category"]:checked')
          .value;

        // Get expend edit item id
        const id = parseInt(e.target.dataset.id);

        // Instantiate Expend Item
        const expendItem = new ExpendItem(id, date, title, cost, category);

        // Validate
        if (title === "" || cost === "" || category === "") {
          // Error aleart
          Swal.fire({
            icon: "error",
            title: "Oops...",
            text: "Please fill in all fields",
          });
        } else {
          // Add updte expend item to local storage
          Store.updateExpendItem(expendItem);

          // Reset expend table
          ui.clearExpendTable();

          // Show expend items in expend table
          Store.displayExpendItems();
```

```
        // Show cards value
        card.counterUp();

        // Reset existing bar chart
        chart.clearBarChart();

        // Draw bar chart
        chart.drawBarChart();

        // Reset existing doughnut chart
        chart.clearDoughnutChart();

        // Show doughnut chart
        chart.drawDoughnutChart();

        // Clear fields
        ui.clearExpendFields();

        // Change modal state
        ui.changeExpendForm("add", "");

        // Close modal
        ui.closeExpendModal();

        // Complete aleart
        Swal.fire({
          icon: "success",
          title: "Your update expend item has been saved",
          showConfirmButton: false,
          timer: 1500,
        });
      }
    }

    e.preventDefault();
  });

// Event listener for add or update income item
document
  .getElementById("income-item-submit")
  .addEventListener("click", function (e) {
    // Instantiate UI
    const ui = new UI();
```

```
// Instantiate DrawCard
const card = new DrawCard();

// Add new expend item
if (e.target.dataset.id === "") {
  // Get form values
  const date = document.getElementById("income-date").value;
  const title = document.getElementById("income-title").value;
  const cost = document.getElementById("income-cost").value;

  // Get time stamp for id
  const id = new Date().getTime();

  // Indtantiate Income Item
  const incomeItem = new IncomeItem(id, date, title, cost);

  // Validate
  if (title === "" || cost === "") {
    // Error aleart
    Swal.fire({
      icon: "error",
      title: "Oops...",
      text: "Please fill in all fields",
    });
  } else {
    // Add income item to local storage
    Store.addIncomeItem(incomeItem);

    // Reset income table
    ui.clearIncomeTable();

    // Show income items in income table
    Store.displayIncomeItems();

    // Show cards value
    card.counterUp();

    // Complete aleart
    Swal.fire({
      icon: "success",
      title: "Your income item has been saved",
      showConfirmButton: false,
      timer: 1500,
    });
```

```
    // Clear fields
    ui.clearIncomeFields();

    // Close modal
    ui.closeIncomeModal();
  }
}
// Update income item
else {
  // Get form values
  const date = document.getElementById("income-date").value;
  const title = document.getElementById("income-title").value;
  const cost = document.getElementById("income-cost").value;

  // Get income edit item id
  const id = parseInt(e.target.dataset.id);

  // Instantiate income item
  const incomeItem = new IncomeItem(id, date, title, cost);

  // Validate
  if (title === "" || cost === "") {
    // Error aleart
    Swal.fire({
      icon: "error",
      title: "Oops...",
      text: "Please fill in all fields",
    });
  } else {
    // Add updte income item to local storage
    Store.updateIncomeItem(incomeItem);

    // Reset income table
    ui.clearIncomeTable();

    // Show income items in income table
    Store.displayIncomeItems();

    // Show cards value
    card.counterUp();

    // Clear fields
    ui.clearIncomeFields();

    // Change modal state
```

```
            ui.changeIncomeForm("add", "");

            // Close modal
            ui.closeIncomeModal();

            // Complete aleart
            Swal.fire({
              icon: "success",
              title: "Your update income item has been saved",
              showConfirmButton: false,
              timer: 1500,
            });
          }
        }

        e.preventDefault();
      });

    // Event listener for expend item delete
    document.getElementById("expend-items").addEventListener("click", function
(e) {
      // Instantiate UI
      const ui = new UI();

      // Instantiate DrawChart
      const chart = new DrawChart();

      // Instantiate DrawCard
      const card = new DrawCard();

      if (e.target.classList.contains("delete")) {
        // Confirm alert
        Swal.fire({
          title: "Are you sure?",
          text: "You won't be able to revert this!",
          icon: "warning",
          showCancelButton: true,
          confirmButtonColor: "#3085d6",
          cancelButtonColor: "#d33",
          confirmButtonText: "Yes, delete it!",
        }).then((result) => {
          if (result.isConfirmed) {
            // Delete expend item from list
            ui.deleteItem(e.target);
```

```
            // Delete expend item from local storage
            Store.deleteExpendItem(e.target.dataset.id);

            // Show cards value
            card.counterUp();

            // Reset existing bar chart
            chart.clearBarChart();

            // Draw bar chart
            chart.drawBarChart();

            // Reset existing doughnut chart
            chart.clearDoughnutChart();

            // Show doughnut chart
            chart.drawDoughnutChart();

            Swal.fire({
              icon: "success",
              title: "Deleted!, Your expend has been deleted.",
              showConfirmButton: false,
              timer: 1500,
            });
          }
        });
      }

      e.preventDefault();
    });

    // Event listener for income item delete
    document.getElementById("income-items").addEventListener("click",
function (e) {
      // Instantiate UI
      const ui = new UI();

      // Instantiate DrawCard
      const card = new DrawCard();

      if (e.target.classList.contains("delete")) {
        Swal.fire({
          title: "Are you sure?",
          text: "You won't be able to revert this!",
          icon: "warning",
```

```
            showCancelButton: true,
            confirmButtonColor: "#3085d6",
            cancelButtonColor: "#d33",
            confirmButtonText: "Yes, delete it!",
          }).then((result) => {
            if (result.isConfirmed) {
              ui.deleteItem(e.target);
              Store.deleteIncomeItem(e.target.dataset.id);
              card.counterUp();
              Swal.fire({
                icon: "success",
                title: "Deleted!, Your income has been deleted.",
                showConfirmButton: false,
                timer: 1500,
              });
            }
          });
        }

        e.preventDefault();
      });

      document.getElementById("expend-items").addEventListener("click", function
(e) {
        const ui = new UI();

        if (e.target.classList.contains("edit")) {
          const id = e.target.dataset.id;
          const date =
            e.target.parentElement.previousElementSibling.previousElementSibling
              .previousElementSibling.previousElementSibling.textContent;
          const title =
            e.target.parentElement.previousElementSibling.previousElementSibling
              .previousElementSibling.textContent;
          const cost =
            e.target.parentElement.previousElementSibling.previousElementSibling
              .textContent;
          const category =
e.target.parentElement.previousElementSibling.textContent;

          const item = {
            id,
            date,
            title,
            cost,
```

```
            category,
          };

          ui.fillExpendForm(item);
        }
      });

    document.getElementById("income-items").addEventListener("click",
function (e) {
      const ui = new UI();

      if (e.target.classList.contains("edit")) {
        const id = e.target.dataset.id;
        const date =
          e.target.parentElement.previousElementSibling.previousElementSibling
            .previousElementSibling.textContent;
        const title =
          e.target.parentElement.previousElementSibling.previousElementSibling
            .textContent;
        const cost = e.target.parentElement.previousElementSibling.textContent;

        const item = {
          id,
          date,
          title,
          cost,
        };

        ui.fillIncomeForm(item);
      }
    });

    document.getElementById("expend-item-cancel").addEventListener("click",
function () {
      const ui = new UI();

      ui.clearExpendFields();

      // Change expend form state
      ui.changeExpendForm("add", "");
    });

    document.getElementById("income-item-cancel").addEventListener("click",
function () {
      const ui = new UI();
```

```
  ui.clearIncomeFields();

  ui.changeIncomeForm("add", "");
});

document.getElementById("export").addEventListener("click", function () {
  Store.exportData();
});
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ АНАЛІЗУ ГРОШОВИХ ВИТРАТ У СИСТЕМІ GOOGLE PAY

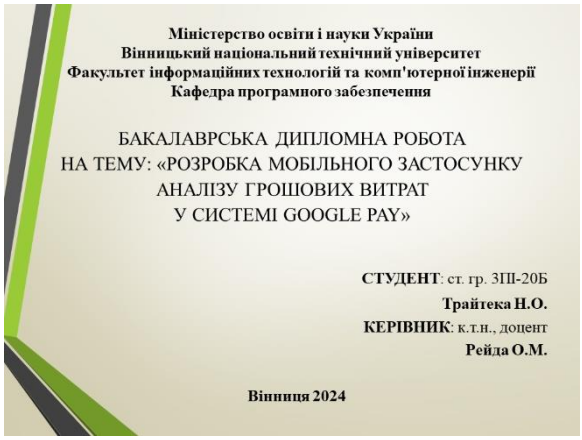


Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Слайд «Актуальність, Мета і Завдання, предмет та область дослідження»

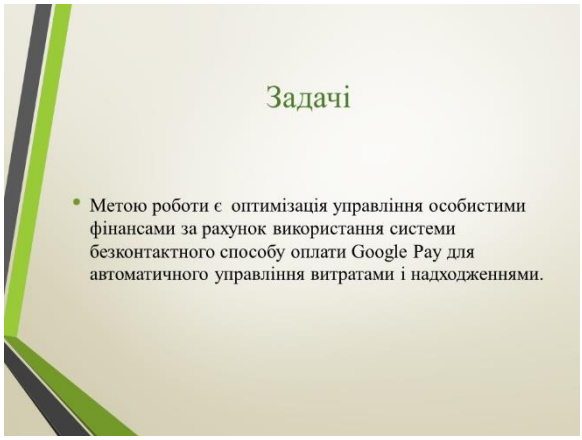


Рисунок Г.3 – Слайд «Задачі»

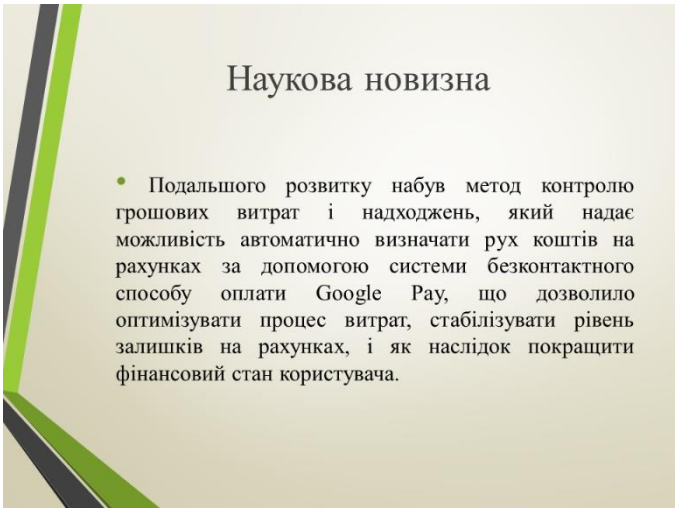


Рисунок Г.4 – Слайд «Наукова новизна»

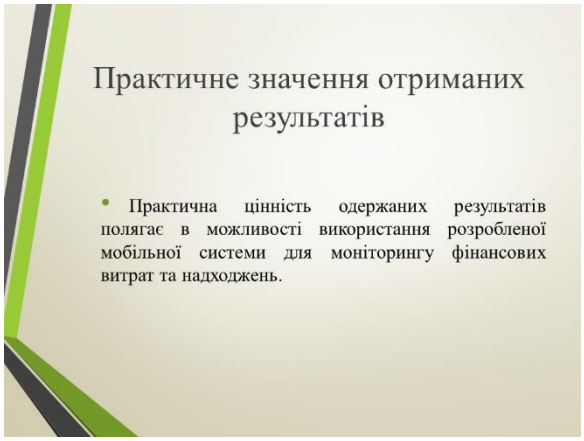


Рисунок Г.5 – Слайд «Практичне значення отриманих результатів»

Рисунок Г.6 – Слайд «Аналог Mint»

Рисунок Г.7 – Слайд «Аналог Personal Capital»

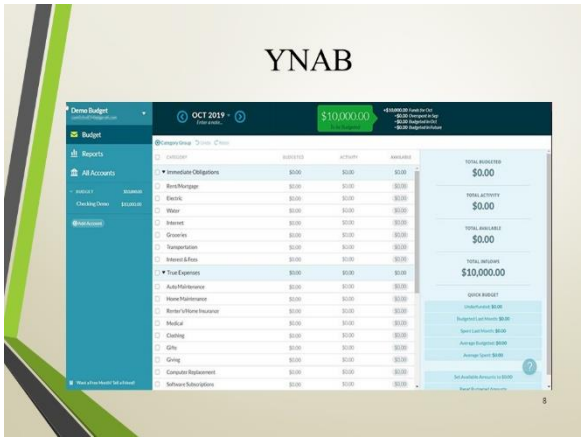


Рисунок Г.8 – Слайд «Аналог YNAB»

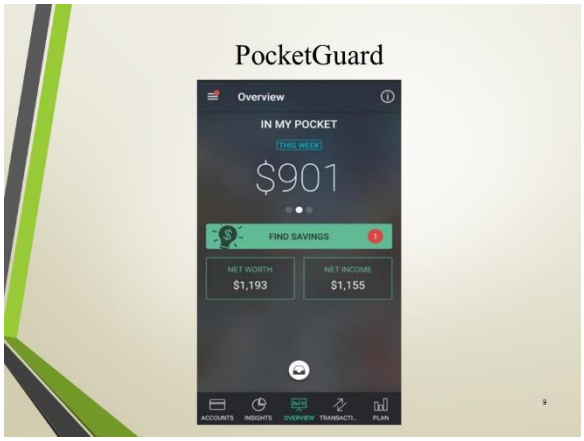


Рисунок Г.9 – Слайд «Аналог PocketGuard»

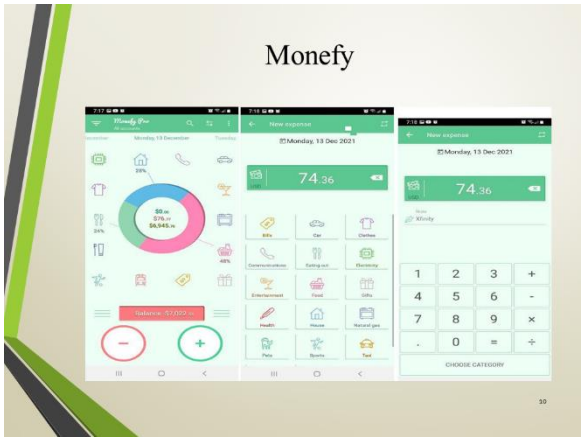


Рисунок Г.10 – Слайд «Аналог Monefy»

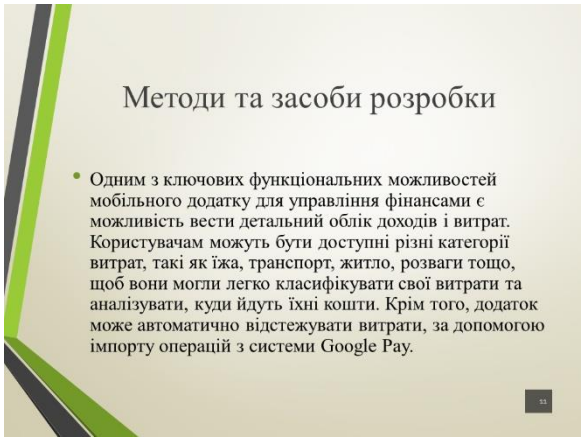


Рисунок Г.11 – Слайд «Методи та засоби розробки»

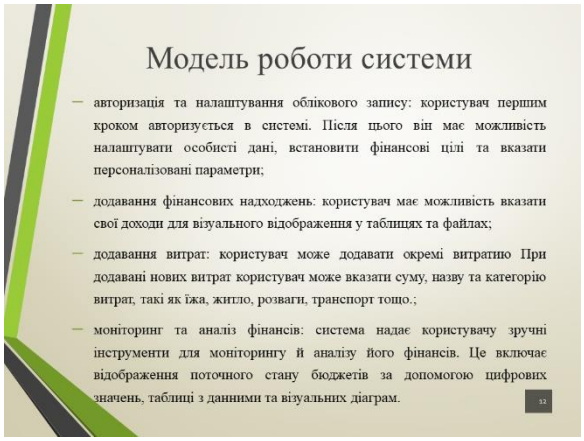


Рисунок Г.12 – Слайд «Модель роботи системи»

Рисунок Г.13 – Слайд «Структурна схема системи»

Рисунок Г.14 – Слайд «Таблиці бази даних»

Рисунок Г.15 – Слайд «Алгоритм роботи системи»

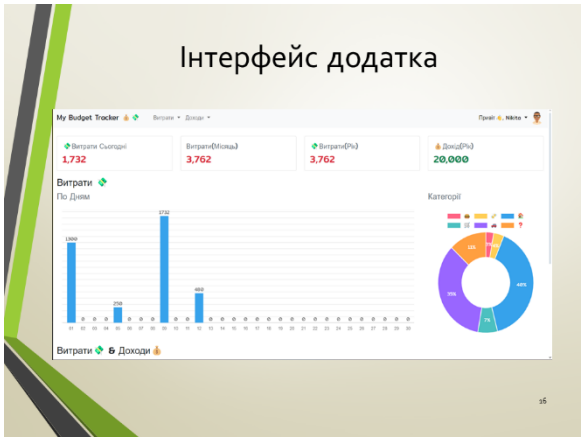


Рисунок Г.16 – Слайд «Інтерфейс додатка»

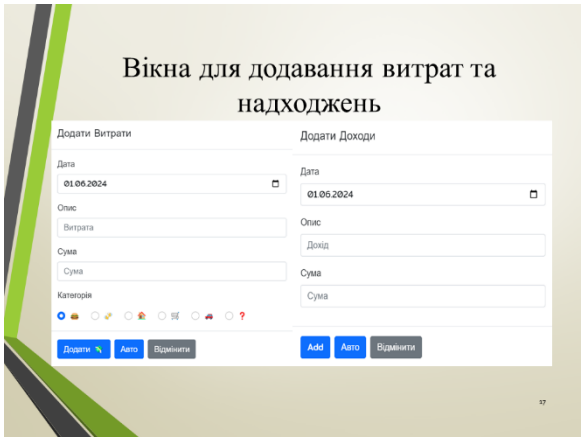


Рисунок Г.17 – Слайд «Вікна для додавання витрат та надходжень»

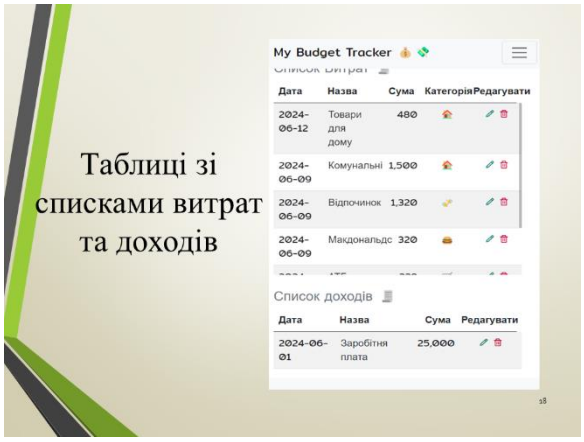


Рисунок Г.18 – Слайд «Таблиці зі списками витрат та доходів»

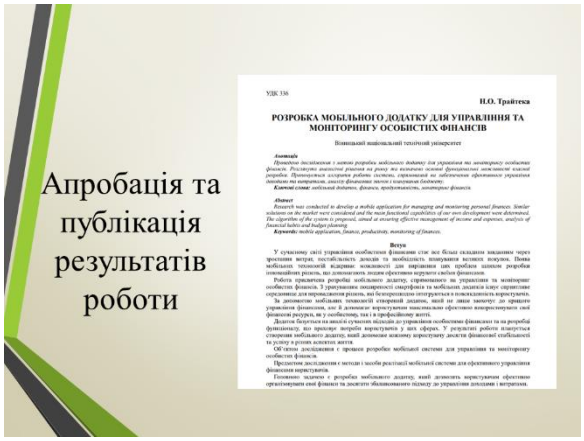


Рисунок Г.19 – Слайд «Апробації та бублікація результатів роботи»

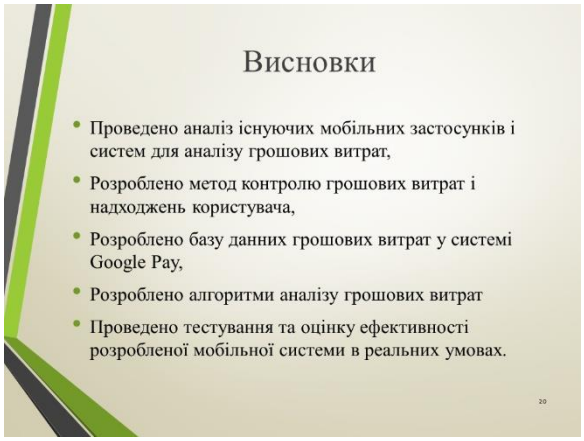


Рисунок Г.20 – Слайд «Висновки»

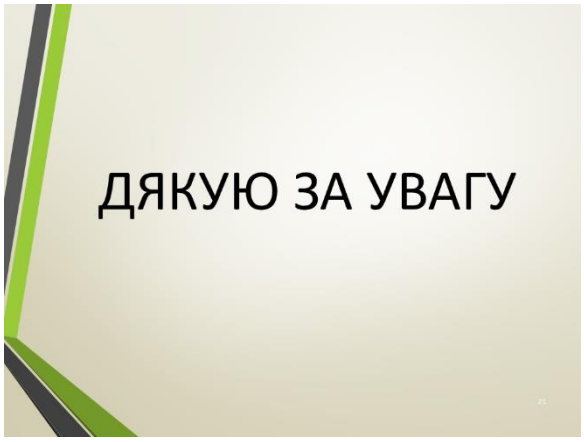


Рисунок Г.20 – Фінальний слайд