Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти»

Виконав: студент 2-го курсу, групи 1КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Щупак Д. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Колодний В.В.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. САІТ
Варчук І.В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
11.10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Щупаку Денису Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти

керівник роботи_ к.т.н., доцент. кафедри КН Колодний В. В.
затверджені наказом вищого навчального закладу від "17" 09 2024 року № 310
2. Строк подання студентом роботи 11.11 2024 року
3. Вихідні дані до роботи:
вхідні дані - кількість банків – не менше 2; мова програмування – строго типізована; фреймворк для .NET 5.0; використання СУБД MySQL;

4. Зміст текстової частини:
Вступ, аналіз предметної області відстеження витрат з автоматичною синхронізацією транзакцій з карти, розробка інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карти, програмна реалізація інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карти, економічна частина, висновки, перелік використаних джерел, додатки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Алгоритм роботи програми, Структура компоновки інтерфейсу системи, графічне представлення прикладу роботи програми, таблиця результатів тестування розробленої програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультант	Підпис, дата	
		Завдання видав	Виконання прийняв
1-3	Колодний В.В. к.т.н., доц. каф. КН		
4	Лесько О.Й., к.е.н. проф., каф. ЕПВМ		

7. Дата видачі завдання 02.09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного рівня інформаційних технологій відстеження витрат з автоматичною синхронізацією транзакцій з карти. Постановка задач дослідження	02.09.24 – 18.09.24	
2	Моделювання інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карти	19.09.24 - 02.10.24	
3	Практичне застосування та оцінка ефективності розроблених моделей	03.10.24 - 17.10.24	
4	Підготовка економічної частини	18.10.24 – 28.10.24	
5	Апробація та/або впровадження результатів дослідження	29.10.24 – 04.11.24	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 - 11.11.24	

Студент _____ (підпис) Щупак Д. О.

Керівник роботи _____ (підпис) Колодиний В. В.

АНОТАЦІЯ

УДК 004.8

Щупак Д.О. Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп`ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 117 с.

Укр. мовою. Бібліогр.: 26 назв; рис.: 11; табл. 15.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології для відстеження витрат з автоматичною синхронізацією транзакцій з карти. Здійснено аналіз існуючих інформаційних ресурсів та систем-аналогів, що надають можливості автоматизації обліку фінансових операцій, та відстеження витрат.

Обрано метод для розв'язання завдань відстеження витрат та синхронізації транзакцій з карти. Спроектовано роботу інформаційної технології для автоматизації фінансових операцій та синхронізації даних із карткових транзакцій.

Обґрунтовано вибір мови програмування C#. Розроблено алгоритм роботи та проведено тестування програмного продукту. На основі проаналізованих результатів можна зробити висновок, що запропонована інформаційна технологія забезпечує точне та зручне відстеження витрат, що допомагає користувачам ефективно керувати своїми фінансами.

Ключові слова: інформаційні технології, відстеження витрат, синхронізація транзакцій, фінансовий облік, автоматизація, карткові транзакції

ABSTRACT

Shchupak D.O. Information Technology for Expense Tracking with Automatic Synchronization of Card Transactions. Master's Qualification Thesis in specialty 122 "Computer Science", educational program "Artificial Intelligence Systems". Vinnytsia: VNTU, 2024.117 pages.

In Ukrainian language. Bibliography: 26 sources; figures: 11; tables: 15.

This master's qualification work is dedicated to the development of an information technology for tracking expenses with automatic synchronization of card transactions. An analysis of existing information resources and analogous systems providing automation of financial operations accounting and expense tracking has been conducted.

A method for solving the tasks of expense tracking and synchronization of card transactions has been selected. The operation of the information technology for automating financial transactions and synchronizing data with card transactions has been designed.

The choice of the C# programming language has been substantiated. An algorithm for the system's operation has been developed, and the software product has been tested. Based on the analyzed results, it can be concluded that the proposed information technology ensures accurate and convenient expense tracking, helping users manage their finances effectively.

Keywords: information technology, expense tracking, transaction synchronization, financial accounting, automation, card transactions.

ЗМІСТ

ВСТУП

Актуальність. У сучасному суспільстві ефективне управління фінансами є однією з ключових складових фінансової грамотності та добробуту громадян. Зростаюча кількість фінансових операцій, здійснюваних через банківські картки, вимагає нових підходів до контролю та аналізу витрат [1]. В умовах швидкого розвитку цифрових технологій автоматизація відстеження витрат стає необхідною для забезпечення точності та оперативності фінансових операцій. Сьогодні, коли особиста фінансова дисципліна є важливою складовою успішного планування бюджету, важливо знаходити нові шляхи для оптимізації процесів управління витратами.

Інформаційні технології та автоматизація процесів відкривають нові можливості для вдосконалення управління фінансами. Використання технологій автоматичної синхронізації транзакцій з банківських карток дозволяє створити ефективні інструменти для контролю витрат у реальному часі. Це сприяє не лише підвищенню точності обліку витрат, але й забезпечує можливість оперативного аналізу фінансового стану, планування бюджету та уникнення непередбачених витрат.

Актуальність дослідження полягає у необхідності розробки сучасних інформаційних систем, які б сприяли ефективному відстеженню та управлінню особистими витратами. Інформаційна технологія з автоматичною синхронізацією транзакцій з карт дозволяє створити гнучкі та зручні системи, які адаптуються до потреб користувачів та забезпечують високий рівень автоматизації. Це забезпечує можливість точного обліку витрат, зменшення помилок та підвищення загальної ефективності управління фінансами.

Зв'язок роботи з науковими програмами, планами, темами.

Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карт, що сприятиме покращенню фінансового обліку та планування бюджету користувачів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Проаналізувати предметну галузь фінансового управління та обґрунтувати доцільність розробки інформаційної технології відстеження витрат.
- Провести огляд та порівняння існуючих технологій.
- Розробити загальну структурну схему системи.
- Здійснити програмну реалізацію системи з автоматичною синхронізацією транзакцій.
- Виконати тестування програми та провести аналіз отриманих результатів.
- Обґрунтувати економічну доцільність розробки.

Об’єктом дослідження є процес організації та управління витратами засобами інформаційних технологій.

Предметом дослідження є програмний засіб та технологія автоматичної синхронізації транзакцій з карт.

Методи дослідження. У роботі використані такі методи наукових досліджень: аналіз аналогічних технологій та програмних рішень, що дозволяє визначити більшість проблем та труднощів, які необхідно вирішити даній роботі.

Наукова новизна одержаних результатів. Вдосконалено інформаційну технологію відстеження витрат з автоматичною синхронізацією транзакцій з карт, яка відрізняється від існуючих наявністю більшої кількості банків для синхронізації та автоматичним визначенням категорії витрат, що дозволило розширити функціональні можливості інформаційної технології.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення відстеження витрат з автоматичною синхронізацією транзакцій з карти.

Запропонована інформаційно технологія сприяє розширеню можливостей використання програмного засобу, зокрема:

- Розроблено алгоритм роботи програмного забезпечення відстеження витрат з автоматичною синхронізацією транзакцій з карти
- Розроблено програмний засіб який дозволяє ефективніше організовувати та контролювати особисті витрати, забезпечуючи точне відображення транзакцій.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки задач, коректністю роботи програмних модулів та тестуванням програмної реалізації відстеження витрат з автоматичною синхронізацією транзакцій з карти.

Особистий внесок здобувача. Усі результати, що наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах опублікованих у співавторстві, автору належать [1]: розробка програмного додатку для інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карти.

Апробація: Результати дослідження були представлені на міжнародній науково-практичній конференції “Молодь в науці: дослідження, проблеми, перспективи" (МН-2025)” [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІДСТЕЖЕННЯ ВИТРАТ З АВТОМАТИЧНОЮ СИНХРОНІЗАЦІЄЮ ТРАНЗАКЦІЙ З КАРТИ

1.1 Аналіз предметної області

У галузі відстеження витрат з синхронізацією з картою існує ряд ключових концепцій та термінів, які формують основу розуміння цієї предметної області.

Фінансовий облік є фундаментальною концепцією, на якій базується відстеження витрат. Це процес систематичного запису, класифікації та аналізу фінансових транзакцій. У контексті особистих фінансів, фінансовий облік включає в себе документування всіх грошових потоків: доходів, витрат, заощаджень та інвестицій.

Транзакція є базовою одиницею в системі відстеження витрат. Вона представляє собою окрему фінансову операцію, таку як покупка товару, оплата послуги або отримання доходу. Кожна транзакція зазвичай характеризується такими атрибутами як дата, сума, категорія, опис та місце здійснення.

Категоризація витрат – це процес групування транзакцій за їх призначенням або типом. Типові категорії можуть включати "Продукти харчування", "Транспорт", "Розваги", "Комунальні послуги" тощо. Правильна категоризація є критично важливою для подальшого аналізу та розуміння структури витрат [2].

Бюджетування – це процес планування та контролю витрат. У контексті додатків для відстеження витрат, бюджетування часто включає встановлення лімітів витрат для різних категорій на певний період часу (наприклад, місяць або рік).

Геолокація – це процес визначення географічного розташування об'єкта, в даному випадку – користувача або місця здійснення покупки. У додатках для відстеження витрат геолокація використовується для автоматичного визначення місця здійснення транзакції.

Геокодування – це процес перетворення адреси або назви місця у географічні координати (широту і довготу). Зворотне геокодування, навпаки, перетворює географічні координати у зрозумілу для людини адресу або назву місця.

Точка інтересу (POI – Point of Interest) – це конкретне місце на карті, яке може бути цікавим або корисним для користувача. У контексті відстеження витрат, точками інтересу можуть бути магазини, ресторани, заправні станції тощо.

Синхронізація даних – це процес забезпечення узгодженості даних між різними пристроями або системами. У додатках для відстеження витрат це може включати синхронізацію між мобільним додатком та веб-інтерфейсом, або між додатком та банківським рахунком.

Візуалізація даних – це представлення даних у графічному форматі. У контексті відстеження витрат це може включати різноманітні графіки, діаграми, теплові карти тощо, які допомагають користувачам краще зрозуміти свої витрати [3].

Фінансова аналітика – це процес аналізу фінансових даних для отримання корисних висновків. У додатках для відстеження витрат це може включати аналіз трендів витрат, виявлення аномальних транзакцій, прогнозування майбутніх витрат тощо.

Персональний фінансовий менеджмент (PFM – Personal Financial Management) – це загальний термін, що описує процес управління особистими фінансами, включаючи бюджетування, відстеження витрат, планування заощаджень та інвестицій.

API (Application Programming Interface) – це набір протоколів, процедур та інструментів для створення програмного забезпечення. У контексті додатків для відстеження витрат, API можуть використовуватися для інтеграції з картографічними сервісами, банківськими системами тощо.

Шифрування – це процес кодування інформації таким чином, щоб вона була недоступною для несанкціонованого доступу. У додатках для відстеження

витрат шифрування використовується для захисту чутливої фінансової інформації користувачів [4].

Машинне навчання – це підгалузь штучного інтелекту, яка фокусується на розробці алгоритмів, які можуть навчатися та покращувати свою продуктивність з часом. У контексті відстеження витрат, машинне навчання може використовуватися для автоматичної категоризації транзакцій, виявлення шахрайських операцій або прогнозування майбутніх витрат.

Відстеження витрат з синхронізацією з картою є комплексною та багатогранною предметною областю, що поєднує в собі елементи фінансового управління, технології геолокації та аналізу даних. Дана сфера вирішує важливе завдання – допомогти користувачам ефективно контролювати свої фінанси, надаючи їм інструменти для детального обліку та аналізу власних витрат.

Основою цієї системи є процес фіксації та категоризації фінансових транзакцій. Кожна покупка, оплата послуги чи інша фінансова операція реєструється в системі, зберігаючи такі важливі дані, як сума, дата, місце здійснення та опис. Дана інформація стає фундаментом для подальшого аналізу та планування.

Однією з ключових особливостей сучасних систем відстеження витрат є їх інтеграція з картографічними сервісами. Це дозволяє не лише фіксувати суму витрат, але й прив'язувати кожну транзакцію до конкретного географічного місця. Такий підхід надає користувачам можливість візуалізувати свої витрати на карті, що може бути особливо корисним для аналізу витрат під час подорожей або для виявлення закономірностей у повсякденних витратах.

Процес геолокації та геокодування відіграє важливу роль у цій системі. Коли користувач здійснює покупку, система може автоматично визначити його місцезнаходження та зіставити ці дані з інформацією про найближчі точки інтересу. Особливості сучасних функцій визначення геолокації дозволяють автоматично заповнювати інформацію про місце здійснення покупки, що значно спрощує процес ведення обліку для користувача.

Важливим аспектом систем відстеження витрат є їх здатність до синхронізації даних. В свою чергу це означає, що інформація про витрати може оновлюватися в режимі реального часу на всіх пристроях користувача – смартфоні, планшеті, комп'ютері. До того ж, багато систем пропонують інтеграцію з банківськими рахунками та кредитними картками, що дозволяє автоматично імпортувати дані про транзакції.

Аналітичні можливості даних систем також заслуговують на увагу. Вони можуть генерувати різноманітні звіти та візуалізації, які допомагають користувачам краще зрозуміти свої витрати. Це можуть бути графіки розподілу витрат за категоріями, часові тренди, порівняння з попередніми періодами тощо. Деякі системи навіть використовують алгоритми машинного навчання для прогнозування майбутніх витрат на основі історичних даних.

Безпека даних є критично важливим аспектом у системах відстеження витрат, оскільки вони працюють з чутливою фінансовою інформацією. Тому такі системи зазвичай використовують складні методи шифрування для захисту даних користувачів.

Загалом, відстеження витрат з синхронізацією з картою є потужним інструментом персонального фінансового менеджменту. Воно допомагає користувачам не лише вести облік своїх витрат, але й отримувати глибоке розуміння своїх фінансових звичок, що в свою чергу сприяє більш усвідомленому та ефективному управлінню особистими фінансами.

1.2 Аналіз систем аналогів

Аналіз систем-аналогів є важливим етапом у розробці нового додатку для відстеження витрат з синхронізацією з картою. Це дозволяє зрозуміти поточний стан ринку, виявити сильні та слабкі сторони існуючих рішень, а також визначити можливості для інновацій. Розглянемо декілька відомих додатків у цій категорії.

Mint є одним з найпопулярніших додатків для управління особистими фінансами . Розроблений компанією Intuit, Mint пропонує широкий спектр

функцій для відстеження витрат. Додаток автоматично синхронізується з банківськими рахунками користувача, категоризує транзакції та створює бюджети. Mint також пропонує функцію відстеження цілей та інвестицій. Однією з ключових особливостей Mint є його здатність аналізувати витрати користувача та пропонувати персоналізовані фінансові поради. Однак, Mint має обмежену функціональність синхронізації з картою та в основному фокусується на американському ринку (рис.1.1). Система категоризації транзакцій в Mint є досить розвиненою. Додаток автоматично розподіляє витрати за різними категоріями, такими як "Продукти", "Розваги", "Комунальні послуги" тощо. Хоча автоматична категоризація не завжди ідеальна, користувачі мають можливість вручну коригувати категорії та створювати власні підкатегорії для більш точного відображення своїх витрат.

Функція бюджетування в Mint дозволяє користувачам встановлювати ліміти витрат для різних категорій. Додаток відстежує прогрес відносно встановлених бюджетів і надсилає сповіщення, коли користувач наближається до ліміту або перевищує його. Це допомагає користувачам краще контролювати свої витрати та дотримуватися фінансових цілей.

Mint також пропонує функцію відстеження фінансових цілей. Користувачі можуть встановлювати різноманітні цілі, такі як створення надзвичайного фонду, погашення боргів, заощадження на відпустку або на покупку будинку. Додаток допомагає відстежувати прогрес у досягненні цих цілей та пропонує поради щодо того, як їх досягти швидше.

Однією з найсильніших сторін Mint є його аналітичні можливості. Додаток аналізує фінансову поведінку користувача та генерує персоналізовані поради щодо поліпшення фінансового стану [5]. Це може включати рекомендації щодо зменшення витрат, збільшення заощаджень, або навіть пропозиції щодо кращих фінансових продуктів, таких як кредитні картки з нижчими відсотковими ставками.

Ще однією перевагою Mint є його інтеграція з кредитними звітами та рейтингами. Додаток дозволяє користувачам безкоштовно відстежувати свій

кредитний рейтинг, отримуючи оновлення та аналіз від TransUnion. Це корисний інструмент для тих, хто планує брати кредити або поліпшити свій кредитний рейтинг. Mint також попереджає про потенційні помилки в кредитній історії, що допомагає уникнути неприємних ситуацій у майбутньому. Окрім цього, користувачі можуть отримувати рекомендації щодо того, як підвищити свій кредитний бал, що робить додаток не тільки інструментом для бюджетування, але й потужним засобом для управління кредитним рейтингом.

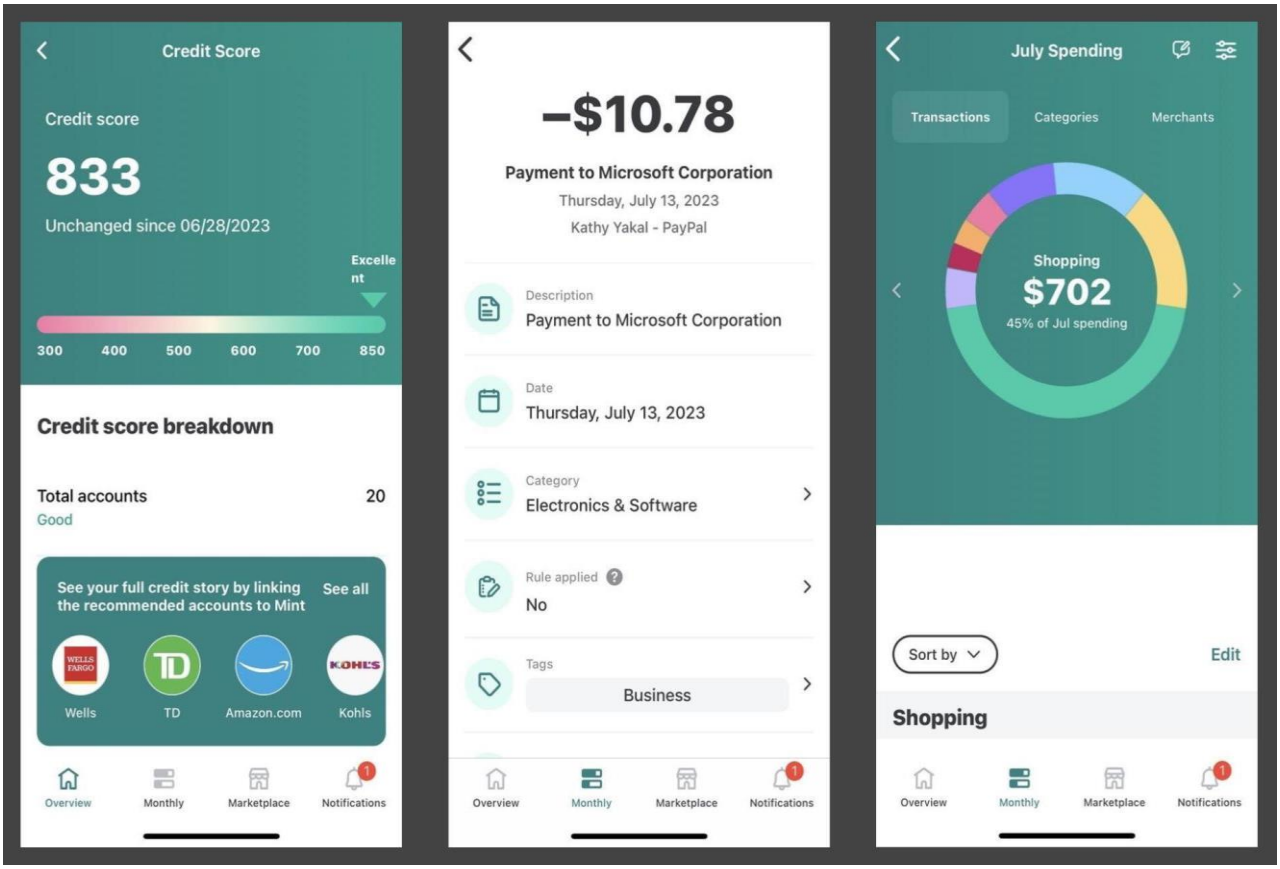

Рисунок 1.1 – Інтерфейс додатку Mint

YNAB (You Need A Budget) – це ще один популярний додаток для бюджетування та відстеження витрат [6]. YNAB використовує унікальну методологію бюджетування, яка базується на чотирьох правилах: надайте кожному долару роботу, охопіть великі нерегулярні витрати, будьте гнучкими, та працюйте зі старими грошима. YNAB пропонує детальну категоризацію витрат та потужні інструменти для аналізу бюджету. Однак, YNAB не має вбудованої

функції синхронізації з картою, що може бути недоліком для користувачів, які хочуть відстежувати свої витрати. (рис.1.2). Також одним із недоліків YNAB є його щомісячна або щорічна підписка, що може бути перешкодою для деяких користувачів. Водночас для тих, хто готовий інвестувати у фінансову стабільність, YNAB може бути потужним інструментом для досягнення довгострокових фінансових цілей.

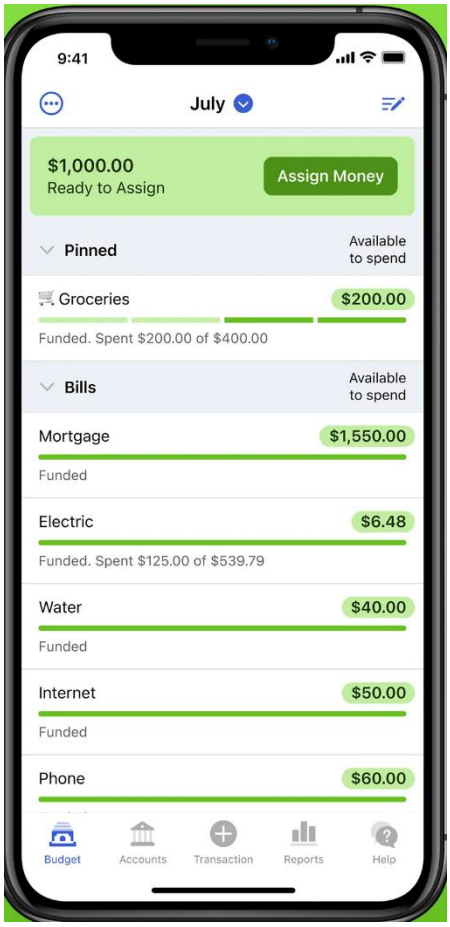

Рисунок 1.2 – Інтерфейс додатку YNAB

Основою методології YNAB є чотири правила, які формують фундамент для ефективного бюджетування:

– "Надайте кожному долару роботу" – це правило заохочує користувачів планувати використання кожного отриманого долара, що допомагає уникнути необдуманих витрат.

– "Охопіть великі нерегулярні витрати" – YNAB допомагає користувачам підготуватися до великих витрат, які трапляються нечасто (наприклад, страховка або податки), розбиваючи їх на менші щомісячні внески.

– "Будьте гнучкими" - це правило визнає, що життя непередбачуване, і заохочує користувачів адаптувати свій бюджет до змінних обставин.

– "Працюйте зі старими грошима" – YNAB прагне допомогти користувачам вирватися з циклу "від зарплати до зарплати", заохочуючи їх використовувати гроші, зароблені минулого місяця, а не поточного.

YNAB пропонує надзвичайно деталізовану систему категоризації витрат. Користувачі можуть створювати власні категорії та підкатегорії, що дозволяє досягти високого рівня гранулярності у відстеженні витрат. Це особливо корисно для тих, хто хоче мати повний контроль над своїми фінансами та розуміти, куди йде кожен долар.

Одним з ключових аспектів YNAB є його потужні інструменти для аналізу бюджету. Додаток пропонує різноманітні звіти та графіки, які допомагають користувачам візуалізувати свої витрати, доходи та прогрес у досягненні фінансових цілей. Це включає в себе аналіз тенденцій витрат з часом, порівняння бюджету з фактичними витратами, та відстеження чистої вартості.

YNAB також відрізняється своїм акцентом на освіті користувачів. Додаток пропонує численні навчальні ресурси, включаючи відео, статті та вебінари, які допомагають користувачам краще зрозуміти принципи ефективного бюджетування та фінансового планування [6].

Personal Capital – це додаток, який поєднує функції відстеження витрат з управлінням інвестиціями. Він пропонує автоматичну синхронізацію з банківськими рахунками та кредитними картками, а також надає детальний аналіз витрат за категоріями [7]. Personal Capital також включає потужні інструменти для аналізу інвестиційного портфеля та планування виходу на пенсію. Однак, як і Mint, Personal Capital має обмежену функціональність синхронізації з картою (рис.1.3).

У сфері відстеження витрат Personal Capital пропонує детальний аналіз за категоріями. Додаток автоматично категоризує транзакції та надає користувачам можливість переглядати свої витрати у вигляді зручних графіків та діаграм. Це допомагає користувачам зрозуміти свої звички витрат та виявити області, де можна заощадити.

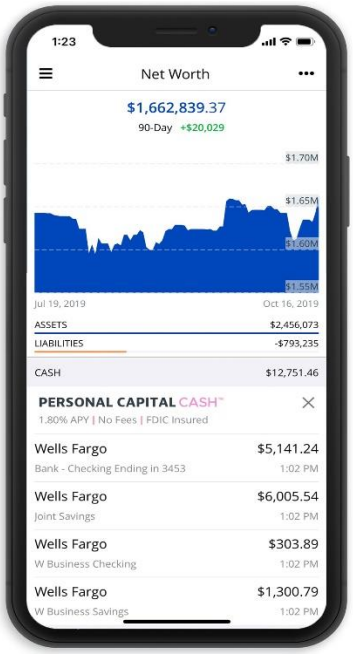

Рисунок 1.3 – Інтерфейс додатку Personal Capital

Однак, справжня особливість Personal Capital проявляється в його інструментах для управління інвестиціями. Додаток пропонує глибокий аналіз інвестиційного портфеля, включаючи оцінку розподілу активів, аналіз комісій та порівняння продуктивності з еталонними показниками. Користувачі можуть побачити, як їхні інвестиції розподілені за класами активів, секторами та географічними регіонами.

Особливо вражаючим є інструмент планування виходу на пенсію Personal Capital. Він враховує поточні заощадження, очікувані внески, соціальне забезпечення та інші фактори для створення детального прогнозу пенсійного доходу. Користувачі можуть експериментувати з різними сценаріями, щоб побачити, як зміни в заощадженнях або витратах можуть вплинути на їхнє пенсійне майбутнє.

Personal Capital також пропонує ряд інших корисних інструментів, таких як калькулятор освітніх витрат, аналізатор комісій за інвестиційними фондами та інструмент для відстеження чистої вартості активів.

Варто зазначити, що хоча базові функції Personal Capital безкоштовні, додаток також пропонує платні послуги управління капіталом для користувачів з великими інвестиційними портфелями.

Spendee – це додаток для відстеження витрат, який пропонує приємний візуальний інтерфейс та базові функції синхронізації з картою [8]. Spendee дозволяє користувачам вручну вводити свої витрати або імпортувати їх з банківських рахунків. Додаток також пропонує функцію створення спільних гаманців для відстеження спільних витрат з друзями або сім'єю. Однак, функціональність синхронізації з картою в Spendee обмежена і не пропонує глибокого аналізу географічних патернів витрат (рис.1.4).

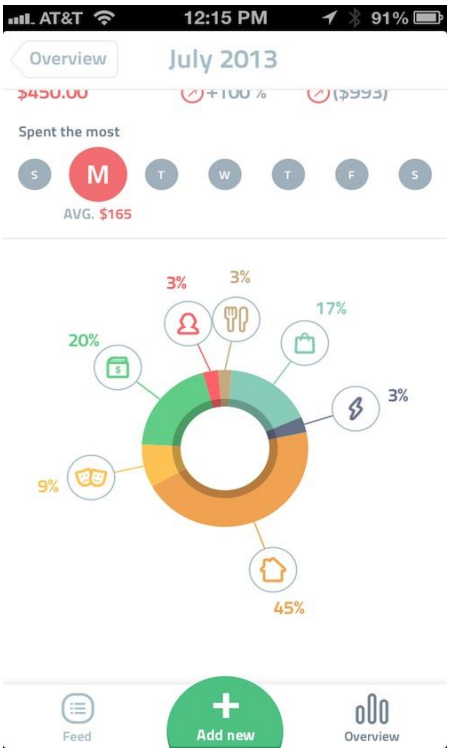

Рисунок 1.4 – Інтерфейс додатку Spendee

Основною особливістю Spendee є гнучкість у введенні даних про витрати. Користувачі можуть вибирати між ручним введенням транзакцій та автоматичним імпортом з банківських рахунків [8]. Ручне введення, хоча і вимагає більше зусиль, може сприяти більш усвідомленому ставленню до витрат. З іншого боку, автоматична синхронізація з банківськими рахунками забезпечує зручність та повноту даних.

Spendee пропонує базову систему категоризації витрат, яка дозволяє користувачам розподіляти свої витрати за різними категоріями, такими як "Їжа", "Транспорт", "Розваги" тощо. Користувачі також можуть створювати власні категорії, що додає гнучкості у відстеженні специфічних типів витрат.

Однією з унікальних функцій Spendee є можливість створення спільних гаманців. Ця функція особливо корисна для пар, сімей або груп друзів, які хочуть відстежувати спільні витрати. Наприклад, пара може створити спільний гаманець для відстеження витрат на домашнє господарство, а група друзів може використовувати цю функцію для планування спільної подорожі.

Spendee також пропонує функцію бюджетування, яка дозволяє користувачам встановлювати ліміти витрат для різних категорій. Додаток відстежує прогрес відносно цих лімітів та надсилає сповіщення, коли користувач наближається до перевищення бюджету.

Moneycontrol – це індійський додаток для управління фінансами, який пропонує функції відстеження витрат разом з інструментами для моніторингу інвестицій та фінансових новин. Додаток дозволяє користувачам вводити свої витрати вручну та категоризувати їх [9]. Також він дозволяє користувачам створювати портфелі, відстежувати рух акцій і вивчати динаміку своїх інвестицій в режимі реального часу. Окрім цього, додаток надає детальні аналітичні звіти, новини ринків, рекомендації аналітиків та інтерв’ю з фінансовими експертами, що допомагає користувачам приймати зважені інвестиційні рішення. Одна з ключових функцій Moneycontrol — це можливість порівнювати акції, взаємні фонди та інші інвестиційні продукти, що робить його універсальним інструментом для інвесторів різного рівня. Крім того, додаток надає доступ до

форумів, де користувачі можуть обмінюватися думками та обговорювати фінансові тенденції [9]. Moneycontrol також пропонує базову функціональність синхронізації з картою, дозволяючи користувачам додавати місцезнаходження до своїх витрат. Однак, ця функція не є повністю автоматизованою і не пропонує глибокого аналізу географічних даних (рис.1.5).

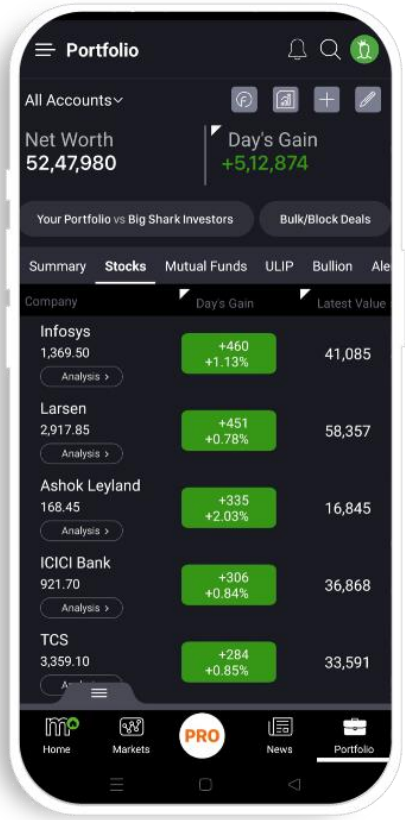

Рисунок 1.5 – Інтерфейс додатку Moneycontrol

Система категоризації в Moneycontrol досить гнучка. Додаток пропонує стандартний набір категорій витрат, таких як "Продукти харчування", "Транспорт", "Розваги" тощо, але також дозволяє користувачам створювати власні категорії. Це дає можливість адаптувати систему відстеження витрат під індивідуальні потреби та звички.

Що стосується синхронізації з картою, Moneycontrol дійсно пропонує базову функціональність у цьому напрямку. Користувачі можуть додавати місцезнаходження до своїх витрат, що дозволяє їм бачити, де були здійснені

конкретні покупки. Однак, ця функція не є повністю автоматизованою – користувачі повинні вручну вводити інформацію про місцезнаходження для кожної транзакції.

Переваги і недоліків даних додатків подані у табл. 1.1.

Таблиця 1.1 – Переваги та недоліки систем-аналогів

Додаток	**Переваги**	**Недоліки**
Mint	Автоматична синхронізація з банками Потужна категоризація Персоналізовані фінансові поради	Обмежена синхронізація з картою Фокус на американському ринку
YNAB	Унікальна методологія бюджетування Детальна категоризація Потужні інструменти аналізу	Відсутність синхронізації з картою Крута крива навчання
Personal Capital	Поєднання відстеження витрат та інвестицій Автоматична синхронізація Інструменти планування пенсії	Обмежена синхронізація з картою Складний для нових користувачів
Spendee	Приємний візуальний інтерфейс Базова синхронізація з картою Спільні гаманці	Обмежений аналіз географічних даних Менш потужні інструменти аналізу
Moneycontrol	Поєднання відстеження витрат та інвестицій Базова синхронізація з картою Фінансові новини	Обмежений аналіз географічних даних

1.3 Вирішення типових проблем додатків відстежування витрат

Інтеграція з геолокаційними сервісами є ключовим компонентом, що відрізняє ці додатки від традиційних систем обліку витрат. Використовуючи API карт, такі як Google Maps або OpenStreetMap, додаток може автоматично визначати місцезнаходження користувача під час здійснення покупки. Це не тільки спрощує процес введення даних, але й надає додаткову контекстну інформацію про витрати.

Синхронізація даних про витрати з картою вимагає розробки складного алгоритму, який повинен враховувати різні фактори. Наприклад, необхідно реалізувати механізм зіставлення координат GPS з конкретними точками інтересу на карті, такими як магазини, ресторани чи інші заклади. Це може бути досягнуто шляхом використання технологій геокодування та зворотного геокодування.

Важливим аспектом є також обробка та аналіз зібраних даних. Додаток повинен мати можливість генерувати різноманітні звіти та візуалізації, які допоможуть користувачам краще зрозуміти свої витрати. Це може включати теплові карти витрат по районах міста, графіки витрат за категоріями в різних локаціях, аналіз сезонних трендів витрат тощо. Для реалізації цих функцій можуть використовуватися бібліотеки візуалізації даних, такі як D3.js або Plotly [10].

Безпека даних є критично важливим аспектом у розробці таких додатків, оскільки вони працюють з чутливою фінансовою інформацією користувачів. Необхідно реалізувати надійні механізми шифрування даних як під час передачі, так і під час зберігання. Крім того, важливо забезпечити надійну аутентифікацію та авторизацію користувачів, можливо, з використанням багатофакторної аутентифікації.

Оптимізація продуктивності є ще одним важливим аспектом розробки. Враховуючи, що додаток може працювати з великими обсягами даних та виконувати складні обчислення для аналізу витрат, необхідно забезпечити ефективне використання ресурсів пристрою. Це може включати використання

кешування даних, асинхронну обробку запитів до сервера та оптимізацію алгоритмів аналізу даних.

Інтеграція з банківськими системами та платіжними сервісами може значно розширити функціональність додатку. Це дозволить автоматично імпортувати дані про транзакції, що зменшить необхідність ручного введення інформації користувачем. Однак, це також створює додаткові виклики з точки зору безпеки та відповідності регуляторним вимогам, таким як PSD2 в Європейському Союзі.

Розробка користувацького інтерфейсу для таких додатків вимагає особливої уваги до деталей. Інтерфейс повинен бути інтуїтивно зрозумілим та ергономічним, дозволяючи користувачам легко вводити дані про витрати та переглядати аналітику. Важливо також забезпечити адаптивний дизайн, який буде добре працювати на різних пристроях - від смартфонів до планшетів та десктопних комп'ютерів.

Галузь відстеження витрат з синхронізацією з картою постійно еволюціонує, і майбутні розробки можуть включати використання технологій штучного інтелекту та машинного навчання для прогнозування майбутніх витрат, виявлення аномальних транзакцій або надання більш персоналізованих фінансових рекомендацій. Це відкриває широкі можливості для подальших досліджень та інновацій у цій сфері.

1.4 Висновок до розділу 1

У даному розділі було здійснено детальний огляд галузі відстеження витрат.

Окрім того, було проаналізовано методи та засоби вирішення типових задач технології відстеження витрат з автоматичною синхронізацією транзакцій з карти.

Проведено аналіз програм-аналогів та наведено їх основні переваги та недоліки.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІДСТЕЖЕННЯ ВИТРАТ

2.1 Огляд методів візуалізації відстеження

Візуалізація відстеження вартості у додатках є критичним аспектом фінансового менеджменту та аналітики. Різноманітні методи візуального представлення даних дозволяють ефективно інтерпретувати складні фінансові показники та тенденції. Лінійні графіки слугують потужним інструментом для відображення динаміки вартості в часовому континуумі, забезпечуючи чітке розуміння довгострокових трендів та циклічних коливань. Стовпчикові діаграми надають можливість здійснювати компаративний аналіз вартісних показників за визначені часові інтервали, що особливо корисно при зіставленні планових та фактичних витрат [11].

Кругові діаграми є ефективним засобом візуалізації структурного розподілу витрат за категоріями, дозволяючи миттєво оцінити пропорційне співвідношення різних статей видатків. Теплові карти застосовуються для ідентифікації патернів інтенсивності витрат у багатовимірному просторі, виявляючи приховані закономірності та аномалії у фінансових даних. Діаграми Санкея демонструють складні фінансові потоки між різними категоріями, візуалізуючи процес розподілу коштів та сприяючи оптимізації бюджетного планування.

Бульбашкові діаграми надають можливість одночасного відображення трьох параметрів – категорії, вартості та частоти, що дозволяє виявляти найбільш значущі та впливові фактори у структурі витрат. Спідометри та датчики забезпечують інтуїтивно зрозуміле представлення поточного рівня витрат відносно встановлених лімітів, використовуючи кольорове кодування для швидкої оцінки фінансового стану.

Деревоподібні карти є потужним інструментом для відображення ієрархічної структури витрат, дозволяючи користувачам здійснювати глибинний аналіз на різних рівнях деталізації. Водоспадні діаграми візуалізують послідовні зміни в загальній сумі витрат, демонструючи вплив окремих факторів на кінцевий

фінансовий результат. Радіальні діаграми ефективно відображають циклічні патерни у витратах, що особливо корисно для аналізу сезонності та періодичних трендів у фінансових показниках.

Інтеграція цих методів візуалізації в додатки для відстеження вартості вимагає ретельного аналізу специфіки даних, цілей фінансового аналізу та потреб цільової аудиторії. Комбінування різних технік візуалізації дозволяє створювати комплексні аналітичні інструменти, здатні забезпечити глибоке розуміння фінансових процесів та підтримку прийняття обґрунтованих управлінських рішень. Розвиток інтерактивних можливостей та застосування алгоритмів машинного навчання для прогнозування тенденцій відкриває нові перспективи у сфері візуалізації фінансових даних, підвищуючи ефективність фінансового менеджменту та стратегічного планування.

При проектуванні додатку планується впровадити комплексний підхід до візуалізації фінансових даних, який поєднуватиме кілька ефективних методів. Передбачається, що центральне місце займатимуть кругові діаграми для відображення бюджету поточного місяця та розподілу витрат за категоріями. Цей вибір обумовлюється здатністю кругових діаграм наочно представляти пропорційні співвідношення між різними компонентами, що дозволить користувачам швидко оцінювати структуру витрат [11].

Для візуалізації річного розподілу витрат планується використання стовпчикових діаграм, які дозволять ефективно порівнювати видатки за різні місяці. Цей метод буде особливо корисним для виявлення сезонних трендів та аномалій у витратах протягом року. Комбінація стовпчикової та кругової діаграм на одному екрані забезпечить комплексний огляд фінансової ситуації, поєднуючи часовий аспект з категоріальним розподілом.

Передбачається також впровадження табличного представлення даних для детального відображення конкретних транзакцій та категорій витрат. Цей підхід забезпечить точність та деталізацію інформації, дозволяючи користувачам аналізувати окремі витрати.

У проекті планується реалізація інтерактивних елементів, таких як випадаючі списки для вибору року та можливість додавання нових категорій з кольоровим кодуванням, що підвищить функціональність та покращить користувацький досвід. Кольорове кодування категорій витрат буде послідовно застосовуватися у всіх візуалізаціях, що покращить сприйняття інформації та дозволить користувачам швидко ідентифікувати різні типи витрат.

Вибір цих методів візуалізації обумовлюється їх здатністю ефективно передавати різні аспекти фінансових даних. Кругові діаграми відмінно підходитимуть для відображення пропорцій, стовпчикові - для часових рядів, а таблиці - для детальної інформації. Така комбінація забезпечить всебічний аналіз витрат, дозволяючи користувачам отримувати як загальне уявлення про свої фінанси, так і детальну інформацію про конкретні витрати.

Додатково, у проекті передбачається реалізація темної теми, що підвищить ергономічність інтерфейсу, адаптуючи його до різних умов використання та особистих преференцій користувачів.

Загалом, обрані методи візуалізації формуватимуть цілісну та інтуїтивно зрозумілу систему фінансового трекінгу, яка дозволить користувачам ефективно аналізувати свої витрати та приймати обґрунтовані фінансові рішення. Проектування додатку з використанням цих методів візуалізації створить потужний інструмент для управління особистими фінансами, який буде зручним у використанні та інформативним для широкого кола користувачів. Окрім класичних діаграм та таблиць, розглядається можливість інтегрувати функцію звітів з автоматичним формуванням PDF-документів. Такі звіти будуть корисними для власників малого або ж людей, які люблять зберігати інформацію також у паперовому вигляді [12].

В таблицю 2.1 зведемо опис використання даних елементів візуалізації та їх переваги.

Таблиця 2.1 – Елементи візуалізації які планується використати в додатку

Метод візуалізації	**Опис**	**Переваги**	**Використання**
Кругові діаграми	Відображення бюджету поточного місяця та розподілу витрат за категоріями.	Наочно представляють пропорційні співвідношення між різними компонентами витрат.	Швидка оцінка структури витрат користувачами.
Стовпчикові діаграми	Візуалізація річного розподілу витрат.	Ефективне порівняння витрат за різні місяці.	Виявлення сезонних трендів та аномалій у витратах протягом року.
Таблиці	Детальне відображення конкретних транзакцій та категорій витрат.	Забезпечують точність та деталізацію інформації.	Аналіз окремих витрат користувачами.
Інтерактивні елементи	Випадаючі списки для вибору року, додавання нових категорій з кольоровим кодуванням.	Підвищують функціональність та покращують користувацький досвід.	Зручний вибір та управління категоріями витрат.
Темна тема	Адаптація інтерфейсу до різних умов використання та особистих преференцій користувачів.	Підвищує ергономічність інтерфейсу.	Демонструє увагу розробників до деталей користувацького досвіду.

2.2 Опис алгоритму роботи додатку

Алгоритм роботи додатку VUE EXPENSES базується на клієнт-серверній архітектурі, що забезпечує ефективну обробку та зберігання фінансових даних користувачів. При запуску додатку користувач потрапляє на головний екран, де відображається панель управління з основними фінансовими показниками та можливістю навігації між різними розділами: Dashboard, Expenses, Stats та Settings.

Процес взаємодії починається з автентифікації користувача. Після успішного входу клієнтська частина надсилає запит на сервер для отримання актуальних фінансових даних. Сервер, у свою чергу, звертається до бази даних, витягує необхідну інформацію та відправляє її назад на клієнт. Ця інформація включає загальні витрати, бюджет поточного місяця, розподіл витрат за категоріями та історію транзакцій.

На головному екрані Dashboard користувач бачить візуалізацію бюджету поточного місяця у вигляді кругової діаграми, що показує співвідношення витрачених та залишкових коштів. Нижче розташовані менші кругові діаграми, які відображають стан бюджету за різними категоріями витрат. Ця інформація динамічно оновлюється при кожному новому записі про витрати.

Додавання нових витрат відбувається через форму "Add New Expense", розташовану на тому ж екрані. Користувач вводить дані про витрату, включаючи категорію, тип, суму та опис. При натисканні кнопки "SUBMIT" ці дані відправляються на сервер, де вони проходять валідацію, обробляються та зберігаються в базі даних. Після успішного збереження сервер відправляє оновлені дані назад на клієнт, і всі відповідні візуалізації на екрані автоматично оновлюються.

У розділі Expenses користувач може переглядати детальну інформацію про свої витрати у вигляді таблиці з можливістю сортування та фільтрації. Кожен запис у цій таблиці представляє окрему транзакцію з інформацією про дату, категорію, суму та опис. Користувач може редагувати або видаляти існуючі

записи, що також викликає відповідні запити до сервера для оновлення бази даних.

Розділ Stats надає розширену аналітику витрат. Тут користувач може переглядати різноманітні графіки та діаграми, які відображають тенденції витрат за різні періоди часу. Стовпчикова діаграма показує розподіл витрат за місяцями поточного року, а кругова діаграма візуалізує загальний розподіл витрат за категоріями. Користувач може вибирати різні періоди для аналізу, що викликає відповідні запити до сервера для отримання та обробки необхідних даних.

У розділі Settings користувач може налаштовувати параметри додатку, такі як валюта, категорії витрат та типи платежів. Зміна цих налаштувань відправляє відповідні запити на сервер для оновлення користувацьких преференцій в базі даних. Тут також реалізована можливість перемикання між світлою та темною темами інтерфейсу, що змінює візуальне оформлення додатку без необхідності перезавантаження сторінки.

Важливим аспектом роботи додатку є постійна синхронізація даних між клієнтом та сервером. При кожній зміні даних (додавання нової витрати, редагування існуючої, зміна налаштувань) відбувається обмін даними між клієнтом та сервером. Це забезпечує актуальність інформації та можливість користувача працювати з додатком на різних пристроях, завжди маючи доступ до найсвіжих даних.

Алгоритм також передбачає обробку помилок та виняткових ситуацій. У разі виникнення проблем з підключенням до сервера або некоректних даних, додаток відображає відповідні повідомлення користувачеві та пропонує варіанти вирішення проблеми.

Безпека даних забезпечується шляхом шифрування всіх комунікацій між клієнтом та сервером, а також використанням токенів автентифікації для кожного запиту. Це захищає персональні фінансові дані користувачів від несанкціонованого доступу.

Таким чином, алгоритм роботи додатку VUE EXPENSES забезпечує комплексне управління особистими фінансами, поєднуючи зручний

користувацький інтерфейс з потужною серверною обробкою даних та їх візуалізацією.

Додаток також використовує повідомлення в реальному часі для інформування користувачів про важливі події, такі як перевищення бюджету або настання строків оплати. Це дозволяє користувачам бути завжди в курсі фінансових справ і вчасно реагувати на зміни. Наприклад, коли загальні витрати перевищують встановлений бюджет, система надсилає повідомлення, яке відображається на головному екрані. Це забезпечує більш проактивне управління фінансами.

Для забезпечення масштабованості додатку використовується модульна архітектура, що дозволяє легко додавати нові функції або змінювати існуючі без необхідності значних змін в основному коді. Це робить додаток більш гнучким та дозволяє швидко адаптуватися до потреб користувачів. Наприклад, можна додати нові категорії витрат або розширити аналітичний функціонал без порушення стабільної роботи інших частин додатку.

На рисунку 2.1 зображено алгоритм роботи додатку. Користувач може взаємодіяти з додатком різними способами: додавати нові витрати, переглядати детальну статистику, налаштовувати параметри додатку. При кожній дії, що змінює дані (наприклад, додавання нової витрати), відбувається відправка інформації на сервер, її обробка та збереження в базі даних. Після цього оновлені дані повертаються на клієнтську частину, де відбувається оновлення всіх відповідних візуалізацій. Процес синхронізації між клієнтом і сервером забезпечує актуальність даних та можливість роботи з додатком на різних пристроях.

Рисунок 2.1 – Алгоритм роботи додатку

Серверна частина веб-додатку виконує ключову роль у забезпеченні коректної роботи системи, оскільки саме вона відповідає за обробку та управління запитами, що надходять від клієнтського інтерфейсу. Основною задачею серверної частини є прийом запитів, їх валідація, виконання бізнес-логіки та відправлення відповідей клієнту, що дозволяє забезпечити зручний і ефективний досвід користувача.

Функціонал серверної частини включає декілька важливих операцій. По-перше, користувач має можливість додавати нові записи про транзакції. Це включає введення необхідних параметрів, таких як сума транзакції, вибір відповідної категорії або створення власної категорії для її класифікації. Сервер обробляє отримані дані, перевіряє їх на відповідність вимогам системи, зберігає в базі даних і відправляє клієнту підтвердження успішного додавання запису.

Окрім створення нових записів, користувачі можуть редагувати вже існуючі транзакції. Ця операція передбачає виклик відповідного API-методу, до якого передаються оновлені дані: змінена сума транзакції, нова категорія або відкоригована дата. Сервер виконує перевірку наявності транзакції, яку потрібно оновити, обробляє нові дані та вносить зміни до бази даних. Після успішного завершення операції користувач отримує відповідне повідомлення про оновлення.

Серверна частина веб-додатку не тільки обробляє запити користувачів, але й виконує ключову функцію у взаємодії з базою даних (БД), яка є основним місцем зберігання інформації про транзакції, категорії та інші пов’язані дані. У цьому проєкті використовується реляційна база даних MySQL, яка була приведена до третьої нормальної форми для забезпечення структурованого, ефективного та надійного зберігання даних.

Приведення БД до третьої нормальної форми означає, що всі таблиці в базі даних були організовані таким чином, щоб уникнути дублювання даних і забезпечити їх максимальну цілісність. У такій структурі кожен атрибут таблиці залежить виключно від первинного ключа, що дозволяє мінімізувати можливість аномалій при оновленні, вставці або видаленні даних. Наприклад, категорії транзакцій, дані користувача, картки користувача, та інші довідкові дані

зберігаються в окремих таблицях і зв’язані з основною таблицею транзакцій за допомогою зовнішніх ключів.

Використання MySQL у поєднанні з третьою нормальною формою забезпечує високу продуктивність системи. Завдяки цьому сервер може виконувати запити швидко й ефективно, навіть у разі великої кількості даних.

Алгоритм роботи оновлення даних на сервері зображено на рисунку 2.3

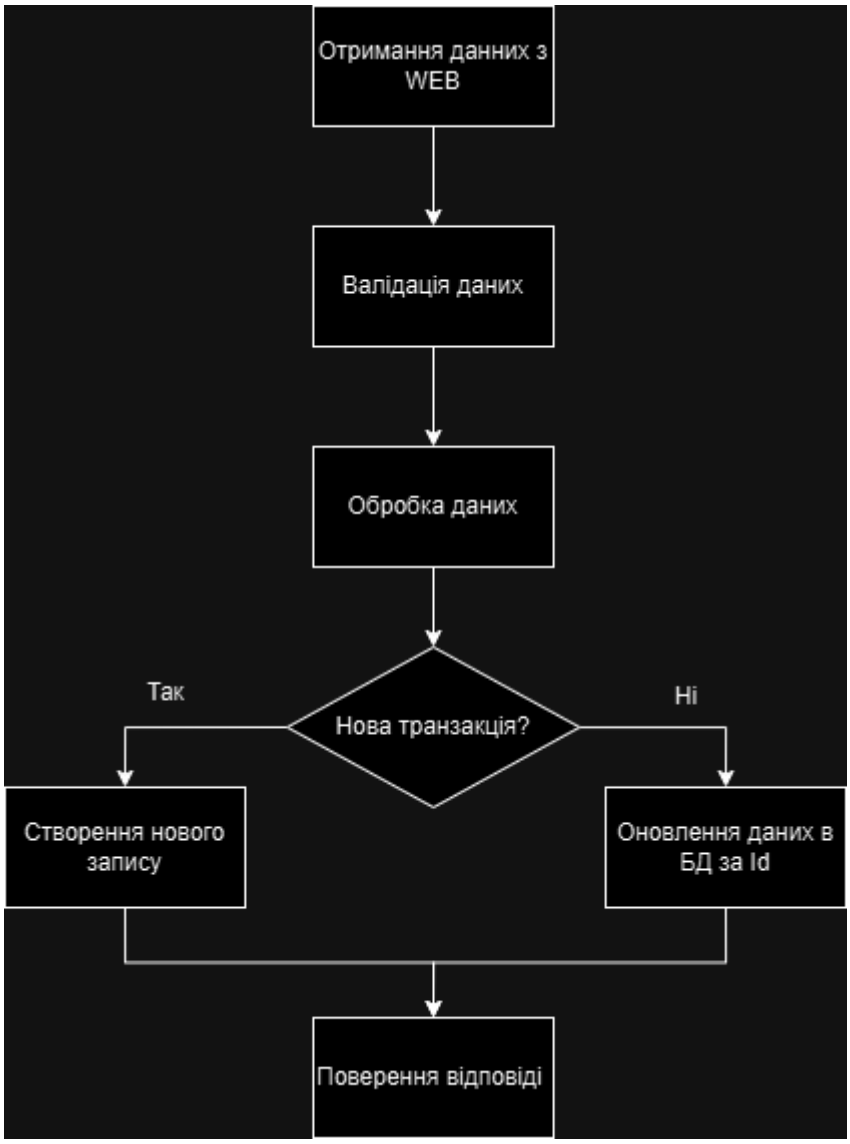

Рисунок 2.3 - Алгоритм роботи оновлення даних на сервері

Оскільки серверна частина веб-додатку не обмежується лише роботою з локальною базою даних MySQL. Вона також забезпечує інтеграцію зі сторонніми сервісами, зокрема з банківськими API, для автоматичної синхронізації

транзакцій. Це дозволяє користувачам отримувати актуальну інформацію про свої фінансові операції без необхідності ручного введення даних.

Одним із ключових етапів цієї інтеграції є автентифікація та отримання токена доступу. У рамках роботи з API банків, таких як Monobank та ПриватБанк, користувач повинен надати необхідні облікові дані, щоб отримати унікальний токен доступу. Сервер виконує такі дії:

Запит токена доступу:

- Користувач отримує токен при авторизації через QR код банку.
- Отриманий токен зберігається на сервері з урахуванням вимог безпеки, таких як шифрування.

Використання токена для запитів до API:

- Токен використовується як частина заголовків авторизації під час надсилання запитів до API Monobank або ПриватБанку.
- Сервер запитує дані про останні транзакції користувача які були здійснені після останньої синхронізації з сервером

Обробка отриманих даних:

- Банківські API повертають дані у форматі JSON. Сервер виконує парсинг отриманих даних, перевіряє їхню цілісність та відповідність системним вимогам.

Збереження даних у базі:

- Отримані та оброблені дані зберігаються в базі даних MySQL. Завдяки структурі, приведеній до третьої нормальної форми, сервер гарантує правильне відображення транзакцій у системі.
- У разі, якщо категорія транзакції, надана банком, не відповідає наявним категоріям у системі, сервер залишає транзакцію не приєднанню до жодної категорії, щоб користувач міг сам змінити категорію.

Після обробки синхронізовані транзакції передаються клієнтському інтерфейсу, що дозволяє користувачу переглядати оновлену інформацію в реальному часі.

Схему синхронізації данних з банківськими АПІ зображено на рисунку 2.4

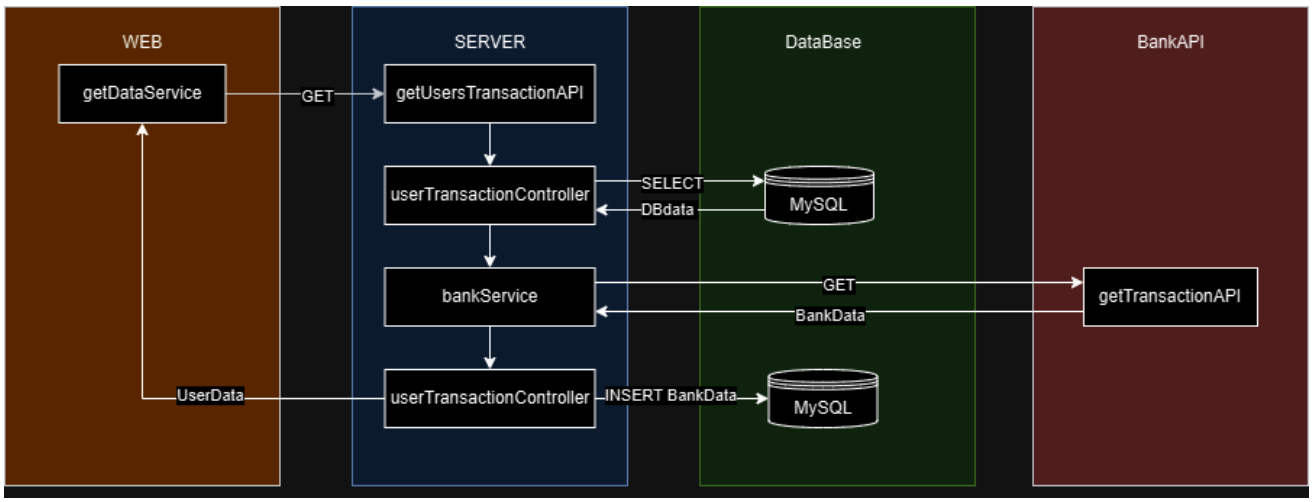

Рисунок 2.4 – схема синхронізації даних з банківськими АПІ

2.3 Аналіз технологій на реалізацію додатку

На серверній стороні вибір .NET 5.0 для створення API обґрунтований його високою продуктивністю, кросплатформеністю та багатим екосистемом. Використання Fluent Validation дозволяє реалізувати надійну валідацію даних, що критично важливо для фінансового додатку. Впровадження архітектури CQRS (Command Query Responsibility Segregation) з використанням MediatR сприяє розділенню операцій читання та запису, що оптимізує продуктивність та масштабованість системи. Entity Framework Core на SQLite забезпечує зручну роботу з базою даних, а Dapper для запитів додає гнучкість та швидкодію при складних вибірках.

Інтеграція Swashbuckle.AspNetCore для Swagger полегшує документування та тестування API, що є важливим для подальшої підтримки та розвитку проекту. Імплементація ASP.NET Core JWT Bearer Authentication з підтримкою токенів оновлення забезпечує надійний механізм автентифікації та авторизації користувачів. Використання Serilog для логування дозволяє ефективно відстежувати та аналізувати роботу системи. Референсна архітектура ContosoUniversity надає перевірені шаблони та практики для структурування проекту.

На клієнтській стороні вибір VueJs як основного фреймворку обумовлений його легкістю, гнучкістю та високою продуктивністю. Vuex з Vuex-persistedstate

забезпечує ефективне управління станом додатку з можливістю збереження даних між сесіями. Vue-router надає зручний механізм клієнтської маршрутизації, що покращує користувацький досвід. Axios використовується для здійснення HTTP-запитів, забезпечуючи простий та потужний інтерфейс для комунікації з сервером. Vuetify як компонентний фреймворк надає багатий набір готових UI-компонентів, що прискорює розробку та забезпечує консистентний дизайн. Lodash надає набір утилітарних функцій для роботи з даними на клієнтській стороні, спрощуючи маніпуляції з об'єктами та масивами. Vue-echarts (Echarts) обраний для створення візуалізацій та графіків, надаючи широкі можливості для представлення фінансових даних.

Такий стек забезпечить оптимальний баланс між продуктивністю, масштабованістю та зручністю розробки, дозволяючи створити надійний та ефективний додаток. Вибір кожної технології обґрунтований її специфічними перевагами та синергією з іншими компонентами системи, що в сукупності створює потужну платформу для реалізації всіх необхідних функціональних вимог VUE EXPENSES. Всі засоби розбробик представлені в таблиці 2.2

Таблиця 2.2 – Засоби розробки

Технологія	**Переваги**	**Аналоги**
.NET 5.0	Висока продуктивність, кросплатформеність, багатий екосистем	Java Spring, Node.js
Fluent Validation	Гнучкість, зрозумілий синтаксис, легка інтеграція з .NET	Data Annotations, JSON Schema
CQRS (MediatR)	Розділення операцій читання та запису, покращена масштабованість	Traditional N-tier Architecture
Entity Framework Core	ORM з широкими можливостями, легка інтеграція з .NET	Hibernate (Java), Sequelize (Node.js)

Продовження таблиці 2.2 – Засоби розробки

Технологія	**Переваги**	**Аналоги**
Dapper	Висока продуктивність для складних запитів, легковісність	ADO.NET, NHibernate
Swashbuckle.AspNetCore	Автоматична генерація документації API, інтерактивний інтерфейс	NSwag, Swagger for other platforms
ASP.NET Core JWT	Стандартизований механізм автентифікації, підтримка refresh токенів	OAuth 2.0, Custom authentication systems
Serilog	Гнучке налаштування, підтримка структурованого логування	log4net, NLog
VueJs	Легкість вивчення, висока продуктивність, реактивність	React, Angular
Vuex	Централізоване управління станом, інтеграція з Vue devtools	Redux (React), NgRx (Angular)
Vue-router	Нативна інтеграція з Vue, декларативна маршрутизація	React Router, Angular Router
Axios	Простий API, підтримка перехоплювачів, широка браузерна сумісність	Fetch API, jQuery AJAX
Vuetify	Багатий набір готових компонентів, матеріальний дизайн	Bootstrap Vue, Element UI
Lodash	Консистентний API, оптимізовані функції для роботи з даними	Underscore.js, Ramda
Vue-echarts (Echarts)	Широкі можливості візуалізації, гнучкість налаштувань	Chart.js, D3.js

Детально розглянемо всі винесені в таблицю засоби розробки.

.NET 5.0 є потужною і сучасною платформою для розробки різноманітних додатків. Вона пропонує високу продуктивність завдяки оптимізованому середовищу виконання та компілятору. Кросплатформеність .NET 5.0 дозволяє розробляти та запускати додатки на різних операційних системах, включаючи Windows, Linux та macOS. Ця платформа має багату екосистему з великою кількістю бібліотек та інструментів, що прискорює процес розробки. .NET 5.0 підтримує різні парадигми програмування та типи додатків, від веб-додатків до мобільних та десктопних рішень [13].

Fluent Validation – це бібліотека для .NET, яка дозволяє створювати гнучкі та потужні правила валідації. Її головною перевагою є зрозумілий та виразний синтаксис, який дозволяє писати складні правила валідації у вигляді ланцюжків методів [13]. Дана бібліотека легко інтегрується з іншими компонентами .NET екосистеми, такими як ASP.NET Core MVC. Fluent Validation підтримує локалізацію повідомлень про помилки та дозволяє створювати власні валідатори, що робить її дуже гнучким інструментом.

CQRS (Command Query Responsibility Segregation) – це архітектурний патерн, який розділяє операції читання та запису даних. У контексті .NET, MediatR є популярною бібліотекою для реалізації CQRS [13]. Цей підхід дозволяє оптимізувати продуктивність складних систем, розділяючи моделі для читання та запису даних. CQRS покращує масштабованість системи, дозволяючи незалежно масштабувати операції читання та запису. Крім того, цей патерн сприяє кращій організації коду та полегшує тестування окремих компонентів системи.

Entity Framework Core є потужним ORM (Object-Relational Mapping) фреймворком для .NET. Він дозволяє працювати з базами даних, використовуючи об'єктно-орієнтований код замість прямих SQL-запитів. Entity Framework Core підтримує широкий спектр баз даних та пропонує як підхід Code First, так і Database First. Фреймворк забезпечує високу продуктивність завдяки оптимізованим запитам та кешуванню. Він також надає зручні інструменти для

міграцій бази даних, що полегшує управління схемою бази даних під час розробки.

Dapper – це простий і легковісний ORM для .NET, який особливо ефективний для складних запитів. Він забезпечує високу продуктивність, будучи тонкою абстракцією над ADO.NET [13]. Dapper дозволяє писати чисті SQL-запити, при цьому автоматично маппінг результатів на об'єкти .NET. Ця бібліотека особливо корисна в сценаріях, де потрібен повний контроль над SQL-запитами при збереженні зручності роботи з об'єктами.

Swashbuckle.AspNetCore – це бібліотека для автоматичної генерації документації API в проектах ASP.NET Core. Вона створює інтерактивний інтерфейс Swagger UI, який дозволяє розробникам легко тестувати та вивчати API. Swashbuckle автоматично генерує специфікацію OpenAPI (раніше відому як Swagger) на основі коду контролерів та моделей. Це значно спрощує процес документування API та забезпечує актуальність документації.

ASP.NET Core JWT (JSON Web Tokens) – це механізм для реалізації автентифікації та авторизації в веб-додатках. JWT забезпечує безпечний спосіб передачі інформації між клієнтом та сервером у вигляді JSON об'єкта. ASP.NET Core має вбудовану підтримку JWT, що дозволяє легко реалізувати токен-базовану автентифікацію. Ця технологія також підтримує refresh токени, що підвищує безпеку та зручність використання.

Serilog – це потужна бібліотека для логування в .NET додатках. Вона підтримує структуроване логування, що дозволяє ефективно аналізувати логи. Serilog має гнучку систему конфігурації та підтримує широкий спектр "стоків" (sinks) для зберігання логів, включаючи файли, бази даних та хмарні сервіси. Бібліотека також дозволяє легко фільтрувати та форматувати логи, що робить її потужним інструментом для діагностики та моніторингу додатків.

Vue.js – це прогресивний JavaScript фреймворк для побудови користувацьких інтерфейсів. Він відомий своєю легкістю вивчення та високою продуктивністю. Vue.js використовує реактивну модель даних, що дозволяє автоматично оновлювати DOM при зміні даних. Фреймворк має модульну

архітектуру, яка дозволяє поступово впроваджувати його в існуючі проекти. Vue.js також пропонує потужні інструменти розробки та велику екосистему додаткових бібліотек.

Vuex – це бібліотека для управління станом у Vue.js додатках. Вона реалізує паттерн "єдине джерело істини" для даних додатку, що спрощує управління складними станами. Vuex забезпечує передбачуваний потік даних, що полегшує відладку та тестування. Бібліотека тісно інтегрується з Vue Devtools, надаючи потужні інструменти для інспекції та налагодження стану додатку.

Vue-router – це офіційна бібліотека маршрутизації для Vue.js. Вона дозволяє створювати одностореінкові додатки (SPA) з декларативним визначенням маршрутів. Vue-router підтримує вкладені маршрути, програмну навігацію та захист маршрутів. Бібліотека тісно інтегрується з Vue.js, що дозволяє легко створювати динамічні та інтерактивні інтерфейси [14].

Axios – це популярна JavaScript бібліотека для виконання HTTP-запитів. Вона пропонує простий та потужний API для роботи з AJAX. Axios підтримує перехоплювачі запитів та відповідей, що дозволяє легко модифікувати або логувати дані. Бібліотека автоматично перетворює дані в JSON та підтримує скасування запитів. Axios також має широку підтримку браузерів та може використовуватися як в клієнтському, так і в серверному JavaScript.

Vuetify – це бібліотека компонентів для Vue.js, яка реалізує принципи Material Design [14]. Вона надає багатий набір готових до використання компонентів, які дозволяють швидко створювати красиві та функціональні інтерфейси. Vuetify підтримує адаптивний дизайн та має вбудовану підтримку для різних розмірів екранів. Бібліотека також пропонує теми та стилі, які можна легко налаштувати.

Lodash – це утилітна JavaScript бібліотека, яка надає велику кількість корисних функцій для роботи з масивами, об'єктами, рядками та іншими типами даних. Lodash пропонує консистентний API та оптимізовані реалізації часто використовуваних операцій. Бібліотека особливо корисна для маніпуляцій з даними, функціонального програмування та покращення продуктивності коду.

Vue-echarts (Echarts) – це бібліотека для створення інтерактивних та потужних візуалізацій даних у Vue.js додатках. Echarts пропонує широкий спектр типів графіків та діаграм, від простих лінійних графіків до складних 3D візуалізацій. Бібліотека підтримує анімації, інтерактивність та адаптивний дизайн. Vue-echarts забезпечує зручну інтеграцію Echarts з Vue.js, дозволяючи легко створювати реактивні та динамічні візуалізації даних.

2.4 Висновок до розділу 2

У даному розділі було оглянуто методи візуалізації даних для відстеження витрат. Описано алгоритми роботи веб додатку, алгоритм оновлення даних на сервері при запиті з клієнта. А також описано алгоритм синхронізації транзакцій з банківськими АПІ.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІДСТЕЖЕННЯ ВИТРАТ

3.1 Програмна реалізація інтерфейсу додатку відстеження витрат

Візуалізація основної частини додатку для відстеження витрат була ретельно спроєктована, використовуючи сучасні інструменти та технології, зокрема Vue.js та Vuetify. Кожен елемент інтерфейсу створено з урахуванням зручності для користувачів, забезпечуючи інтуїтивно зрозумілий доступ до ключових функцій програми.

Інтерфейс розділений на декілька основних секцій, кожна з яких виконує свою конкретну функцію. Для кращої організації та чіткого візуального розділення кожна секція представлена окремою карткою (v-card), що дозволяє легко ідентифікувати її призначення. Такий підхід не лише полегшує сприйняття інформації, але й сприяє її ефективному представленню на різних пристроях.

Vuetify — це потужна бібліотека компонентів для Vue.js, яка слідує принципам Material Design. Її перевагою є широкий набір готових до використання елементів, які інтегруються без значних зусиль. У даному випадку v-card використовується для створення окремих блоків контенту з унікальним стилем і функціональністю. Завдяки гнучкості Vuetify забезпечується адаптивність дизайну, що дозволяє програмі однаково добре виглядати як на настільних комп’ютерах, так і на мобільних пристроях.

Для структурування інтерфейсу використовуються v-container, v-layout, і v-flex, які створюють зручну сітку. Наприклад, на мобільних пристроях секції автоматично перебудовуються у вертикальний стек для забезпечення зручного перегляду.

У верхньому лівому куті інтерфейсу розташована секція для додавання нових витрат. Ця секція представлена карткою з компонентом ExpenseForm, який включає в себе поля для введення назви витрати, суми та вибору категорії. Заголовок "Add New Expense" виділяється своїм оформленням і чітко демонструє призначення цієї секції. Користувачам також доступна кнопка для збереження

введених даних, а додаткові підказки допомагають уникнути помилок під час заповнення форми. Особливий стиль оформлення картки забезпечує її видимість та виділення серед інших елементів.

У верхній правій частині інтерфейсу розташована секція для відображення бюджету поточного місяця. Тут використовується компонент DoughnutChart, який представляє співвідношення витрачених коштів до загального бюджету у вигляді кругової діаграми. Нижче діаграми розташована текстова інформація, яка демонструє конкретні числові показники, такі як загальний ліміт бюджету, фактичні витрати та залишок коштів. Це дозволяє користувачам швидко оцінити свій фінансовий стан, не витрачаючи час на додаткові розрахунки чи аналіз.

Під основними секціями знаходиться область, яка демонструє бюджети за категоріями витрат. Ця секція включає набір мініатюрних кругових діаграм, кожна з яких відповідає окремій категорії, наприклад, "Харчування", "Розваги" чи "Транспорт". Для уникнення перевантаження інтерфейсу використовується горизонтальний скролінг, що дозволяє розміщувати значну кількість категорій у компактному вигляді. Компонент DoughnutChart тут також виконує роль інструменту для візуалізації даних, забезпечуючи легке сприйняття інформації.

Остання секція, яка займає нижню частину інтерфейсу, надає користувачам річний огляд їхніх фінансів. Дана картка розділена на дві частини: ліворуч розміщений компонент BarChart для відображення динаміки річних витрат, а праворуч – компонент PieChart, який показує розподіл витрат за категоріями в рамках року. Таке поєднання різних типів графіків дозволяє користувачам отримати комплексне розуміння своїх довгострокових фінансових тенденцій.

Форма додавання нових витрат розташована у верхній лівій частині інтерфейсу. Вона представлена компонентом ExpenseForm, який дозволяє користувачам вводити дані про нові витрати. Форма має заголовок "Add New Expense" та кнопку для збереження введених даних.

Секція бюджетування знаходиться у верхній правій частині. Дана секція використовує компонент DoughnutChart для візуалізації місячного бюджету. Кругова діаграма показує співвідношення витрачених коштів до загального

бюджету. Під діаграмою відображаються конкретні суми ліміту бюджету та витрачених коштів.

Бюджети під категоріями розташовані під основними секціями. Модуль представляє собою набір маленьких кругових діаграм (також використовуючи DoughnutChart), кожна з яких відображає бюджет для окремої категорії витрат. Діаграми розміщені в горизонтальному ряду з можливістю прокрутки.

Модульна частина річного огляду займає нижню частину інтерфейсу і розділена на дві частини: BarChart для відображення річних витрат і PieChart для візуалізації розподілу витрат за категоріями.

В цілому, в інтерфейсі доцільно використати адаптивний дизайн (завдяки Vuetify), який забезпечує коректне відображення на різних пристроях. Наприклад, на менших екранах елементи можуть перебудовуватися для вертикального відображення.

Кольорова схема інтерфейсу базується на темі користувача (user.theme), що дозволяє персоналізувати вигляд додатку.

Додаток активно використовує можливості графічних компонентів для представлення даних у зрозумілому форматі. Це значно полегшує аналіз фінансової інформації та допомагає користувачам приймати зважені рішення щодо своїх витрат. Завдяки адаптивному дизайну, реалізованому за допомогою Vuetify, програма зберігає свою функціональність та зручність незалежно від типу пристрою. Візуалізація даних, інтеграція графіків та використання компонентів Material Design дозволили створити інтерфейс, який поєднує у собі зручність, сучасний вигляд та багатофункціональність, що робить додаток незамінним інструментом для управління особистими фінансами.

Використання компонентів Vuetify [15], таких як v-container, v-layout, та v-flex, забезпечує адаптивність дизайну, дозволяючи інтерфейсу коректно відображатися на різних пристроях та розмірах екранів. Наприклад, на мобільних пристроях секції можуть перебудовуватися у вертикальний стек для оптимального відображення на менших екранах.

Функціональність використаних компонентів опишемо в таблиці 3.1.

Таблиця 3.1 – Основні функції застосовних компонентів Vuetify

Компонент	**Функціональність**	**Основні параметри**
v-container	Обмежує ширину контенту та забезпечує відступи з боків. Центрує контент на сторінці. Підтримує адаптивний дизайн на різних розмірах екранів.	fluid: робить контейнер повношириннім (100%). tag: визначає HTML-тег для контейнера (наприклад, , <section>).
v-layout	Організовує елементи в рядок або колонку завдяки Flexbox. Керує вирівнюванням та відступами між елементами. Забезпечує адаптивність і гнучкість макету сторінки.	row, column: задає напрямок (горизонтально або вертикально). align-start, align-center, align-end: налаштовує вертикальне вирівнювання. justify-start, justify-center, justify-end: налаштовує горизонтальне вирівнювання. wrap: дозволяє переносити елементи на новий рядок.
v-flex	Задає ширину та розташування елементів всередині v-layout. Підтримує адаптивне масштабування контенту відповідно до екрана. Керує розміром та порядком елементів, дозволяючи створювати адаптивні та структуровані макети.	xs12, sm6, md4, lg3: задають ширину елемента на екранах різних розмірів (від 1 до 12 колонок). offset-sm2, offset-md4: зміщує елемент на певну кількість колонок. order-sm1, order-lg2: змінює порядок елементів в макеті в залежності від розміру екрана.

Продовження таблиці 3.1 – Основні функції застосовних компонентів Vuetify

Компонент	Функціональність	Основні параметри
v-spacer	Допоміжний компонент, що використовується для заповнення простору між елементами в макеті. Він забезпечує правильне вирівнювання.	v-col-auto: підлаштовується під вміст. Забезпечує динамічну адаптацію ширини в межах макета. Автоматична ширина, гнучкість у макеті, підтримка адаптивності

Кольорова схема інтерфейсу є важливим елементом, який базується на темі користувача, що дозволяє персоналізувати візуальний стиль додатку відповідно до індивідуальних уподобань та забезпечує унікальний досвід для кожного користувача. Ця персоналізація досягається шляхом передачі параметрів теми в усі ключові компоненти візуалізації, такі як графіки, кнопки, фонові елементи та текстові поля, що автоматично адаптують свою кольорову палітру відповідно до обраних налаштувань.

Наприклад, у темній темі основний акцент робиться на контрастних тонах, що сприяє зручному сприйняттю інформації в умовах низької освітленості, зменшуючи навантаження на очі. У світлій темі, навпаки, використовуються м'які пастельні кольори, які створюють враження легкості й простоти. Такі зміни зачіпають усі аспекти інтерфейсу, включаючи колір графіків, фон карток (v-card), виділення активних елементів, а також кольорові акценти, що допомагають фокусувати увагу користувача на важливих деталях.

Особливість цієї реалізації полягає в тому, що обрана тема не лише змінює зовнішній вигляд інтерфейсу, а й інтегрується з логікою програми, створюючи відчуття гармонії між візуальною частиною і функціональністю. Наприклад, компоненти візуалізації, такі як DoughnutChart, BarChart та PieChart, автоматично використовують кольорову палітру, що відповідає темі користувача, забезпечуючи естетичну цілісність навіть при зміні теми в режимі реального часу.

Крім того, кольорова схема підтримує можливість налаштувань користувачем. Це означає, що кожен користувач може обирати власні кольорові акценти для конкретних елементів, таких як категорії витрат чи фонові кольори. Наприклад, витрати на транспорт можуть бути позначені синім кольором, розваги – зеленим, а харчування – помаранчевим, що сприяє швидкому та зручному аналізу фінансових даних. Такий підхід робить використання додатку не лише корисним, але й приємним з точки зору естетики.

Додатковою перевагою є динамічність кольорової схеми. Якщо користувач змінює тему (наприклад, з темної на світлу), усі елементи інтерфейсу автоматично підлаштовуються без необхідності перезавантаження сторінки або втрати поточного стану програми. Це досягається завдяки інтеграції параметрів теми в глобальні стилі додатку, що забезпечує безшовний досвід користування. Такий підхід дозволяє додатку виглядати однаково добре на різних пристроях та в різних умовах використання, незалежно від обраної теми.

Кольорова схема також відіграє важливу роль у створенні емоційного сприйняття додатку. Правильно підібрана палітра кольорів може впливати на настрій користувача, створюючи відчуття спокою, впевненості або енергійності під час роботи з додатком. Наприклад, теплі тони, такі як помаранчевий чи жовтий, додають дружності та мотивації, тоді як холодні відтінки, як-от синій чи зелений, асоціюються зі стабільністю та надійністю. Такий ретельно продуманий дизайн допомагає створити позитивне враження від використання додатку, підвищуючи задоволеність користувача та його прихильність до продукту.

Таким чином, кольорова схема не лише додає естетичної привабливості додатку, але й сприяє зручності використання, підвищує впізнаваність інтерфейсу та створює відчуття персоналізованого досвіду для кожного користувача. Це робить додаток більш привабливим та функціональним, задовольняючи сучасні вимоги до дизайну інтерфейсів.

На рисунку 3.1 можна побачити структуру компоновки інтерфейсу системи.

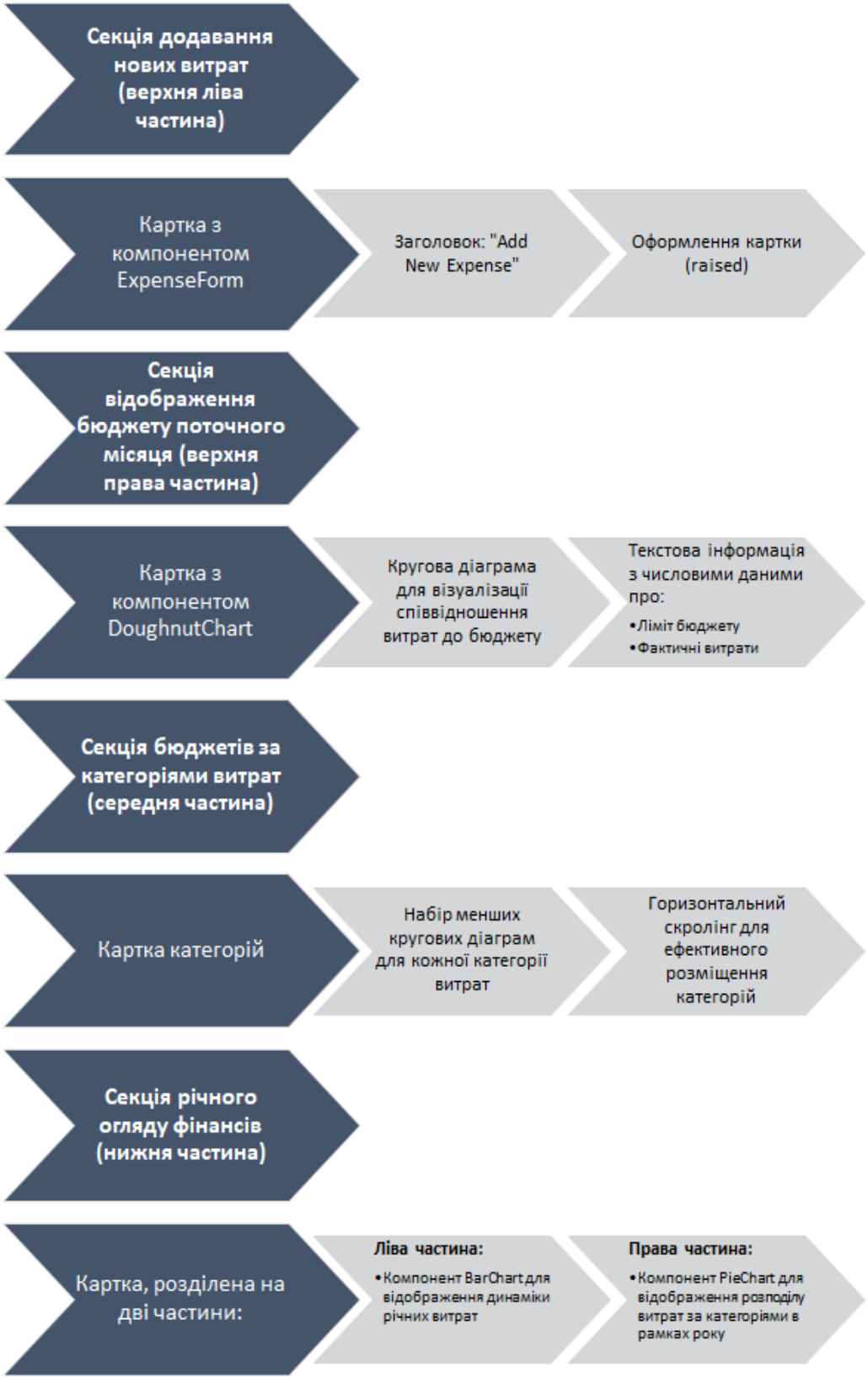

Рисунок 3.1 – Структура компоновки інтерфейсу системи

Структура компонента Expences відповідає стандартній архітектурі Vue.js, складаючись з шаблонної та скриптової частин. Шаблонна частина використовує компоненти фреймворку Vuetify для створення інтуїтивно зрозумілого та естетичного користувацького інтерфейсу. Центральним елементом інтерфейсу є таблиця даних (v-data-table), яка відображає список витрат з можливістю сортування. Для додавання нових та редагування існуючих витрат передбачено діалогове вікно (v-dialog), що активується кнопкою "New Expense". Шаблон також використовує слоти (slots) для кастомізації відображення певних колонок таблиці, таких як форматування значень валюти та кнопок дій.

Скриптова частина компонента забезпечує логіку роботи з даними та взаємодію з користувацьким інтерфейсом. Вона починається з імпорту необхідних залежностей, включаючи Vuex для управління станом додатку та компонент ExpenseForm. Компонент визначає локальний стан, який включає налаштування таблиці, стан діалогового вікна та структуру об'єкта витрат. Обчислювані властивості (computed properties) використовуються для доступу до даних зі сховища Vuex, забезпечуючи реактивність інтерфейсу. Методи компонента реалізують основні операції з витратами: додавання, редагування та видалення. Метод saveExpense обробляє як створення нових, так і редагування існуючих витрат, викликаючи відповідні дії Vuex. Компонент також реалізує обробку помилок та скидання форми після закриття діалогового вікна.

Архітектурно компонент дотримується принципу одностороннього потоку даних, характерного для Vue.js, що спрощує відстеження змін стану та полегшує налагодження. Використання Vuex для централізованого управління станом додатку забезпечує кращу масштабованість та підтримку коду. Декларативний підхід до побудови інтерфейсу, властивий Vue.js, дозволяє чітко розділити логіку відображення та бізнес-логіку компонента.

3.2 Програмна реалізація додаткових сторінок клієнтської частини додатку відстеження витрат

Для ефективного управління клієнт-серверною системою додатку критично важливо правильно спроектувати головний компонент Vue.js, який виконує роль центрального вузла для всіх ключових функцій і структур веб-додатку. Цей компонент виступає основою для організації всіх інших компонентів, забезпечуючи узгодженість дизайну, функціональність маршрутизації та ініціалізацію необхідних даних.

Основна структура компонента включає навігаційну панель, головний контейнер для відображення контенту та інші базові елементи, які формують основу всього інтерфейсу.Він використовує систему маршрутизації Vue.js через елемент <router-view>, що дозволяє динамічно відображати різні сторінки або компоненти залежно від поточного URL. Елемент <router-view> є ключовим компонентом у системі маршрутизації Vue Router. Він функціонує як динамічний контейнер, який відображає компоненти, що відповідають поточному маршруту URL. Коли користувач переміщається між різними сторінками додатку, Vue Router автоматично визначає, який компонент повинен бути відображений у <router-view>, базуючись на конфігурації маршрутів. Це дозволяє створювати однопагінні додатки (SPA), де зміна контенту відбувається без перезавантаження всієї сторінки. <router-view> може бути вкладеним, що дозволяє реалізовувати складні ієрархічні структури маршрутів. Даний компонент є центральним елементом для реалізації навігації в Vue.js додатках, забезпечуючи плавний перехід між різними частинами додатку та підтримуючи концепцію компонентно-орієнтованої архітектури.

Важливою функцією компонента є ініціалізація даних при завантаженні додатку. У методі mounted викликається ряд дій Vuex для завантаження необхідних даних, таких як типи витрат, категорії, статистика та валюти. Таким чином, всі необхідні дані доступні для використання в інших частинах додатку.

Компонент також відповідає за управління темою додатку. Він встановлює та змінює тему (світлу або темну) на основі налаштувань користувача, що

зберігаються у стані Vuex. Адаптивний дизайн реалізується через використання умовних класів у шаблоні, які змінюються залежно від розміру екрану (рис.3.2). Це забезпечує максимально зручний доступ до всіх функцій додатку, незалежно від типу пристрою, який використовує користувач.

```
<script>
import { mapState } from "vuex";
import Navbar from "@/components/TheNavbar";
import {
  LOAD_EXPENSE_TYPES,
  LOAD_CATEGORIES,
  LOAD_CATEGORIES_BREAKDOWN,
  LOAD_EXPENSES_BREAKDOWN,
  LOAD_EXPENSES,
  LOAD_CURRENCIES
} from "@/store/_actiontypes";
export default {
  components: {
    Navbar
  },
  mounted() {
    this.$store.dispatch(`expenseTypes/${LOAD_EXPENSE_TYPES}`);
    this.$store.dispatch(`expenseCategories/${LOAD_CATEGORIES}`);
    this.$store.dispatch(`statistics/${LOAD_CATEGORIES_BREAKDOWN}`);
    this.$store.dispatch(`statistics/${LOAD_EXPENSES_BREAKDOWN}`);
    this.$store.dispatch(`expenses/${LOAD_EXPENSES}`);
    this.$store.dispatch(`account/${LOAD_CURRENCIES}`);
    this.$vuetify.theme.dark = this.theme === "dark";
  },
  computed: {
    ...mapState({
      theme: state => (state.account.user ? state.account.user.theme : "")
    })
  },
  watch: {
    theme(newTheme, oldTheme) {
      this.$vuetify.theme.dark = newTheme === "dark";
    }
  },
  data: () => ({
    //
  })
};
</script>
```

Рисунок 3.2 – Фрагмент скриптової частини домашньої сторінки додатку

Наступний фрагмент коду представляє собою компонент сторінки логіну у Vue.js додатку, використовуючи фреймворк Vuetify для створення

користувацького інтерфейсу. Архітектура цього компонента побудована з урахуванням принципів однопагних додатків (SPA) та реактивного програмування.

Шаблонна частина компонента визначає структуру сторінки логіну. Вона використовує компоненти Vuetify для створення респонсивного та естетичного інтерфейсу. Центральним елементом є картка (v-card), яка містить форму логіну з полями для введення електронної пошти та пароля. Форма включає валідацію полів, а також можливість показати/приховати пароль. Нижче форми розташовані кнопки для реєстрації та входу.

Шаблонна частина компонента відповідає за визначення структури та дизайну сторінки логіну, створюючи гармонійний і зручний інтерфейс за допомогою компонентів Vuetify. Основним елементом є картка (v-card), яка виступає візуальним і функціональним центром сторінки. Вона має чітке обрамлення та підняте оформлення, що акцентує увагу користувача на формі входу, виділяючи її серед інших елементів інтерфейсу. Завдяки гнучкості Vuetify, картка забезпечує адаптивність, автоматично підлаштовуючись до розміру екрана пристрою, що гарантує комфортне використання на різних платформах, включаючи мобільні телефони, планшети та настільні комп’ютери.

Форма логіну, яка розміщується всередині картки, містить два основні поля для введення електронної пошти та пароля. Ці поля інтегрують валідацію, яка забезпечує перевірку коректності введених даних у режимі реального часу. Наприклад, поле для електронної пошти перевіряє формат адреси, тоді як поле для пароля контролює мінімальну довжину або інші параметри, задані бізнес-логікою. Крім того, поле для пароля оснащене опцією показати/приховати введений текст за допомогою інтерактивної іконки, що покращує зручність і безпеку роботи, особливо в умовах використання додатку в публічних місцях.

Нижче форми логіну розташовані кнопки, які виконують кілька важливих функцій. Кнопка входу, оформлена в стилі основної кольорової схеми додатку, служить для підтвердження дій користувача та відправлення даних форми на сервер для авторизації. Її інтерактивність включає анімацію натискання та стан

завантаження, що інформує користувача про процес обробки запиту. Друга кнопка перенаправляє користувача на сторінку реєстрації, дозволяючи легко створити новий акаунт, якщо його ще не існує.У методі created відбувається скидання теми Vuetify та стану логіну користувача. Це забезпечує коректний початковий стан при завантаженні сторінки логіну. Методи компонента обробляють подання форми логіну та перенаправлення на сторінку реєстрації.

Архітектурно, цей компонент демонструє розділення відповідальності між представленням (шаблон), логікою (методи) та управлінням станом (Vuex). Використання Vuex для операцій логіну та управління станом завантаження забезпечує централізоване управління даними та полегшує масштабування додатку.

Компонент також демонструє ефективне використання допоміжних функцій для валідації, що є важливим кроком у покращенні якості коду, його повторного використання та спрощенні підтримки у майбутньому. Наприклад, валідація електронної пошти чи пароля винесена в окремі утилітарні функції, які можуть бути застосовані в інших частинах додатку, де потрібна аналогічна логіка. Це дозволяє уникнути дублювання коду, зменшує ризик помилок та забезпечує легкість внесення змін у разі необхідності оновлення правил валідації.

Інтеграція компонента із системою маршрутизації Vue Router відіграє важливу роль у забезпеченні навігації між сторінками додатку (рис 3.3). Завдяки цьому користувачі можуть швидко переходити між сторінкою логіну, реєстрації та іншими модулями, такими як відновлення пароля чи головна панель управління. Маршрутизація реалізована за допомогою методу push Vue Router, що дозволяє плавно змінювати URL без необхідності повного перезавантаження сторінки. Крім того, для захисту сторінок із чутливими даними, наприклад, панелі управління, компонент інтегрує механізми перевірки авторизації, які перенаправляють користувача назад на сторінку логіну у разі відсутності дійсного токена доступу. Отже, компонент сторінки логіну у Vue.js є важливим елементом для забезпечення зручного користувацького досвіду.

```
<script>
import { mapState, mapActions } from "vuex";
import { LOGIN, LOGOUT } from "@/store/_actiontypes";
import validations from "@/helpers/validations";
import router from "@/router/index";

export default {
  data() {
    return {
      loginForm: {
        email: "test@demo.com",
        password: ""
      },
      showPassword: false,
      ...validations
    };
  },
  computed: {
    ...mapState({
      loading: state => state.loader.loading
    })
  },
  created() {
    //reset theme
    this.$vuetify.theme.dark = 0;
    // reset login status
    this.LOGOUT();
  },
  methods: {
    ...mapActions("account", [LOGIN, LOGOUT]),
    handleLoginSubmit() {
      if (!this.$refs.loginForm.validate()) return;

      const { email, password } = this.loginForm;
      if (email && password) {
        this.LOGIN({ email, password });
      }
    },
    handleRegisterClick() {
      router.push("/register");
    }
  }
};
</script>
```

Рисунок 3.3 – Фрагмент скриптової частини сторінки логіну

Скрипт реєстраційної сторінки реалізує адаптивну форму реєстрації користувача на Vue.js, використовуючи компоненти Vuetify для побудови інтерфейсу, таких як v-app, v-content, v-container, v-layout, v-flex, v-card, та інші. Форма включає валідацію полів введення (наприклад, електронна пошта, ім’я, прізвище, пароль) з використанням правил з модуля валідацій. Взаємодія з Vuex забезпечується через mapState та mapActions, дозволяючи керувати станом завантаження і виконанням асинхронних дій, таких як REGISTER. При натисканні кнопки реєстрації метод handleRegisterSubmit здійснює перевірку даних форми через валідацію і, якщо всі поля коректні, викликає відповідну дію з Vuex для надсилання даних на сервер. Зокрема, в скрипті реалізовано і перемикання видимості пароля за допомогою іконок та маршрутизацію на сторінку входу через router.push при натисканні кнопки входу.

mapState використовується для відображення (мапінгу) властивостей стану Vuex у властивості computed компонента. Це дозволяє зручно отримувати доступ до глобального стану прямо з компоненту, що особливо важливо для реактивного оновлення UI при зміні стану. Замість того, щоб звертатися до сховища через this.$store.state, можна скористатися mapState, що надає більш зручний синтаксис і підтримує структуровану організацію коду.

У скрипті mapActions використовується для мапінгу методу REGISTER з модуля account у Vuex. Це дозволяє легко викликати дію без необхідності явно звертатися до this.$store.dispatch, що робить код більш компактним і зрозумілим. При виклику this.REGISTER, відповідна дія запускається з передачею даних форми реєстрації, забезпечуючи обробку користувацьких даних на сервері [16].

Використання mapState і mapActions у компонентах Vue.js значно підвищує захист системи та ефективність реєстрації, оскільки сприяє чіткій організації взаємодії з глобальним станом і діями. По-перше, відокремлення логіки управління станом і асинхронних операцій у Vuex дозволяє централізовано контролювати і змінювати поведінку системи, що підвищує безпеку, запобігаючи прямим маніпуляціям з даними компонентів. Це унеможливлює несанкціоновані

зміни стану, оскільки всі зміни проходять через строго визначені дії і мутації, де можуть бути реалізовані перевірки, валідації та захисні механізми (рис.3.4).

Ефективність реєстрації покращується за рахунок асинхронної обробки запитів до серверу, що реалізується через Vuex-дії, такі як REGISTER. Завдяки використанню mapActions, компоненти можуть легко викликати потрібні дії без прямого доступу до внутрішньої логіки, що зменшує складність і ризик помилок при обробці даних користувачів. Це також забезпечує масштабованість, дозволяючи швидко вносити зміни в логіку реєстрації, наприклад, додавання нових перевірок або захисту від ботів, без потреби модифікувати компоненти (табл.3.2).

Таблиця 3.2 – Порівняння методів mapState і mapActions

Метод	**Призначення**	**Покращення захисту системи**	**Підвищення ефективності реєстрації**
mapState	Відображає властивості стану Vuex у computed властивості компонента.	Дозволяє централізовано керувати станом, запобігаючи несанкціонованим змінам з компонентів.	Реактивно оновлює інтерфейс на основі змін стану, забезпечуючи швидкий і точний фідбек.
mapActions	Відображає дії Vuex як методи компонента, дозволяючи викликати асинхронні операції і змінювати стан.	Виконує всі операції через строго контрольовані дії, де можуть бути реалізовані додаткові перевірки.	Оптимізує асинхронну обробку реєстрації, спрощуючи логіку компонента і мінімізуючи помилки.

Не менш важливим є те, що використання mapState для відображення стану завантаження, як-от відображення спінера під час реєстрації, створює кращий досвід для користувачів, знижуючи ймовірність дублювання запитів або помилок

через повторні кліки. Це також запобігає ситуаціям, коли форма могла б відправити неповні або некоректні дані через неправильно оброблений стан, підвищуючи загальну надійність процесу реєстрації. Таким чином, впровадження цих технологій робить систему більш захищеною, адаптивною та ефективною у роботі з критичними даними користувачів.

```
<script>
import { mapState, mapActions } from "vuex";
import { REGISTER } from "@/store/_actiontypes";
import validations from "@/helpers/validations";
import router from "@/router/index";

export default {
  data() {
    return {
      loginForm: {
        email: "test@demo.com",
        password: ""
      },
      registerForm: {
        email: "",
        firstName: "",
        lastName: "",
        password: ""
      },
      showRegisterPassword: false,
      showPassword: false,
      ...validations
    };
  },
  computed: {
    ...mapState({
      loading: state => state.loader.loading
    })
  },
  methods: {
    ...mapActions("account", [REGISTER]),
    handleRegisterSubmit() {
      if (!this.$refs.registerForm.validate()) return;

      const { email, firstName, lastName, password } = this.registerForm;
      this.REGISTER({ email, firstName, lastName, password });
    },
    handleLoginClick() {
     router.push("/login");
    }
  }
};
</script>
```

Рисунок 3.4 – Фрагмент скриптової частини сторінки реєстрації

Надалі опишемо скрипт, що реалізує статистичний модуль системи управління витратами на основі компонентів Vue.js і Vuetify, забезпечуючи відображення ключових фінансових показників і статистики витрат користувача.

У верхній частині макета представлено чотири картки (v-card), кожна з яких відображає важливі фінансові показники: загальні витрати (overallSpent), найбільші витрати за категорією (mostSpentBy), найбільші витрати на конкретний об'єкт (mostSpentOn), і загальні витрати за поточний рік (spentThisYear). Дані отримуються з Vuex через допоміжну функцію mapGetters, що дозволяє компоненту легко підключитися до стану модуля expenses та динамічно відображати актуальні значення.

Для забезпечення адаптивності дизайну кожна картка має умови для зміни класів залежно від поточного розміру екрану, використовуючи систему breakpoint Vuetify ($vuetify.breakpoint). Завдяки цій особливості є можливість оптимізувати відступи і розміщення карток у відповідності до ширини вікна, покращуючи користувацький досвід.

У середній частині компонента розміщені три вкладені компоненти: ExpensesStats, ExpenseCategoryStats, і ExpenseTypeStats. Вони відповідають за візуалізацію детальної статистики витрат за роками, категоріями та типами витрат відповідно. Компоненти отримують вхідні статистичні параметри years, months, та theme, що налаштовують відображення даних згідно з вибором користувача. Метод getYears генерує масив із значеннями років, починаючи від поточного року мінус чотири, що дозволяє переглядати дані за останні кілька років.

Властивість months визначає перелік місяців року, що дозволяє вибирати дані за конкретний місяць або переглядати сукупну статистику за всі місяці (рис.3.5).

```
  computed: {
    ...mapGetters("expenses", [
      "overallSpent",
      "mostSpentBy",
      "mostSpentOn",
      "spentThisYear"
    ]),
    ...mapState({
      theme: state => (state.account.user ? state.account.user.theme : "")
    })
  },
  data: () => ({
    months: [
      { name: "All", value: "" },
      { name: "January", value: 1 },
      { name: "February", value: 2 },
      { name: "March", value: 3 },
      { name: "April", value: 4 },
      { name: "May", value: 5 },
      { name: "June", value: 6 },
      { name: "July", value: 7 },
      { name: "August", value: 8 },
      { name: "September", value: 9 },
      { name: "October", value: 10 },
      { name: "November", value: 11 },
      { name: "December", value: 12 }
    ]
  })
};
</script>

<style scoped>
```

Рисунок 3.5 – Фрагмент скрипту статистичного модуля клієнтської частини додатку

3.3 Програмна реалізація серверної частини додатку відстеження витрат

Проєктування структури серверної частини додатку для відстеження витрат здійснюється з дотриманням принципів архітектурного дизайну та з використанням сучасних інструментів і технологій, що забезпечують масштабованість, безпеку та продуктивність системи. Розробка цього модуля базується на стеку .NET Core і включає використання таких технологій, як

MediatR для обробки запитів, FluentValidation для перевірки даних, Entity Framework Core для доступу до бази даних, та JWT для автентифікації.

– Основною точкою входу в додаток є клас Startup, який відповідає за початкове налаштування програми, ініціалізацію служб і конфігурацію HTTP-конвеєра. Конфігурація починається з методу ConfigureServices, де додаються необхідні служби, налаштовуються бази даних, забезпечується підключення до MediatR та інтеграція FluentValidation у конвеєр обробки запитів.

– Конфігурація служб: У цьому блоці налаштовуються служби MediatR для керування запитами та відповідями між різними частинами системи. Крім того, тут відбувається налаштування Entity Framework Core з використанням SQLite як сховища даних.

– Підключення валідації: FluentValidation інтегрується в MediatR pipeline через ValidationPipelineBehavior, що забезпечує автоматичну перевірку даних перед їх обробкою.

– Налаштування Swagger: Додається документація API з використанням Swagger, що дозволяє описати точки доступу, використовувані для взаємодії з API.

Безпека додатку реалізується за допомогою JWT (JSON Web Token) аутентифікації. Винесення налаштувань JWT в окремі параметри дозволяє гнучко керувати параметрами безпеки, такими як валідність токену, правила перевірки підпису, видавець і аудиторія. Метод AddJwt визначає параметри валідації токенів та обробку подій, пов'язаних з автентифікацією.

Структура даних додатку підтримується за допомогою класу ExpensesContext, який реалізує шаблон DbContext. Цей контекст відповідає за управління зв'язками між об'єктами, відстеження змін, та забезпечення доступу до бази даних SQLite. У моделі ExpensesContext описуються основні таблиці: користувачі (Users), витрати (Expenses), категорії витрат (ExpenseCategories), типи витрат (ExpenseTypes) і токени оновлення (RefreshTokens).

Для захисту системи від несанкціонованих дій з боку користувачів буде використано ряд відомих технологій криптографії з метою захисту даних, зведемо їх у таблицю (табл. 3.3).

Таблиця 3.3 – Методи використані для захисту даних при проектуванні додатку

Метод	Мета	Реалізація	Забезпечена безпека
Хешування паролів (HMACSHA512)	Забезпечення безпечного зберігання паролів у базі даних.	Використання алгоритму HMACSHA512	Захист від атак типу «таблиць розтину» або «веселкових таблиць».
Використання JWT	Надання користувачам безпечного методу автентифікації без зберігання даних на сервері.	Створення підписаних JWT, що містять інформацію про користувача і мають обмежений термін дії.	Захист від підробки токенів і обмеження дії токенів у часі.
Генерація унікальних токенів і оновлення токенів	Підтримка авторизації без повторного введення облікових даних.	Використання унікальних ідентифікаторів JWT і токенів оновлення, що генеруються за допомогою криптографічних методів.	Захист від атак на повторне використання токенів і забезпечення унікальності токенів.

Продовження таблиці 3.3 – Методи використані для захисту даних при проектуванні додатку

Метод	Мета	Реалізація	Забезпечена безпека
Захист від експірації токенів	Мінімізувати ризики використання прострочених токенів.	Встановлення терміну дії токенів та перевірка відповідності поточному часу; додавання заголовка Token-Expired у разі використання простроченого токена.	Запобігання використанню викрадених токенів після закінчення їх терміну дії.
Керування підписом токенів за допомогою криптографічного ключа	Запобігання підробці токенів.	Використання SigningCredentials для підпису JWT з використанням секретного ключа, відомого тільки серверу.	Забезпечення цілісності та автентичності токенів, захист від підробки запитів.
Обробка помилок автентифікації	Сигналізація клієнту про необхідність дій у випадку помилок автентифікації.	Додавання спеціальних заголовків у відповідь на невдалі спроби автентифікації (наприклад, повідомлення про прострочені токени).	Поліпшення взаємодії з користувачем та підвищення надійності системи через своєчасне інформування про помилки.

3.4 Тестування програмного додатку відстеження витрат

Опишемо основні види тестування, які застосуємо при проведенні тестування десктопної версії програмного додатку відстеження витрат, і продумаємо основні тест-кейси.

1. Функціональне тестування

Мета: Перевірка правильності виконання основних функцій додатку, таких як додавання, редагування, та видалення витрат, категоризація витрат, генерація звітів тощо.

Особливості: Функції мають працювати однаково стабільно незалежно від теми (світла чи темна), типу пристрою, або роздільної здатності екрану. Важливо перевірити всі можливі сценарії взаємодії користувача з програмою [17].

2. UI/UX тестування

Мета: Забезпечення зручності та інтуїтивності інтерфейсу користувача.

Особливості: Важливе значення мають перевірки на коректне відображення елементів інтерфейсу, зокрема кнопок, текстових полів, іконок, таблиць, графіків тощо. Користувач повинен легко орієнтуватися у додатку без необхідності вдаватися до інструкцій. Особливу увагу приділяють дизайну та інтерфейсу у різних темах [18].

3. Тестування у світлій і темній темах

Мета: Забезпечення коректного відображення інтерфейсу у світлій та темній темах, що є особливо актуальним з огляду на зростаючу популярність темних режимів серед користувачів.

Особливості:

Контрастність і видимість елементів: У темній темі елементи інтерфейсу повинні залишатися чіткими і контрастними, щоб не погіршувалась читабельність тексту, помітність кнопок, іконок та інших важливих компонентів. У світлій темі кольори мають бути приємними для очей і не дратувати користувача яскравістю.

Тестування адаптації кольорової гами: Кольори мають автоматично підлаштовуватись під обрану тему, а зміна теми має відбуватись плавно без візуальних помилок або некоректного відображення елементів.

Іконки та графіка: У різних темах можуть виникати проблеми з іконками та зображеннями, які спочатку розроблені під одну колірну схему. Необхідно переконатися, що всі елементи залишаються видимими та виглядають органічно в обох режимах.

Анімації та ефекти: Темна і світла теми можуть по-різному впливати на сприйняття анімаційних ефектів. Необхідно переконатися, що всі анімації працюють однаково добре, незалежно від обраної теми.

Тестування доступності (Accessibility Testing): Перевірка на коректність використання кольорів та розмірів шрифтів для людей із порушеннями зору. Контраст у темній темі повинен відповідати стандартам доступності, забезпечуючи достатню видимість усіх елементів.

4. Тестування на різних роздільних здатностях екрану

Мета: Забезпечення коректного вигляду додатку на різних моніторах та пристроях.

Особливості: Потрібно перевірити адаптивність інтерфейсу, щоб уникнути проблем з відображенням на моніторах з різною роздільною здатністю та співвідношенням сторін. Перевірка повинна включати як стандартні монітори, так і високоякісні дисплеї (4K).

5. Тестування продуктивності та навантаження

Мета: Забезпечення стабільної роботи додатку при високому навантаженні та багатозадачності.

Особливості: Перевірка на витривалість при обробці великої кількості даних, одночасному використанні декількох користувачів або відкритті великих звітів.

6. Безпекове тестування

Мета: Виявлення вразливостей у додатку, зокрема при роботі з конфіденційними даними користувачів.

Особливості: Перевірка безпечності автентифікації, зберігання паролів, коректності роботи захищених з’єднань та механізмів доступу до даних [18].

7. Тестування синхронізації

Мета: перевірка коректності обробки, збереження та передачі даних між банківськими АПІ і сервером у реальному часі.

Основні екрани додатку наведемо на рисунках 3.6-3.10 (наведемо в темній темі).

Рисунок 3.6 – Вкладка Дашборд

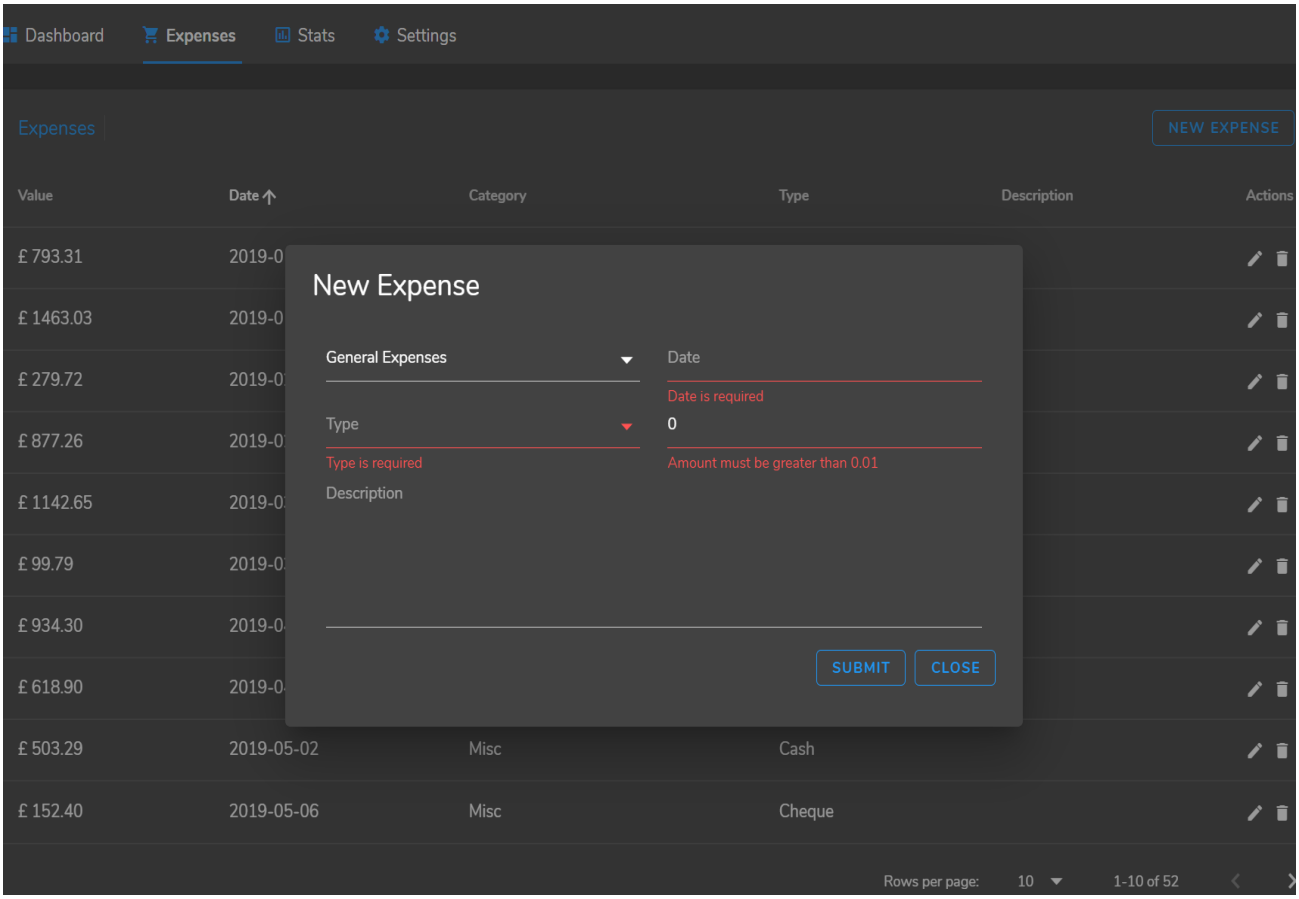

Рисунок 3.7 – Додавання нових витрат в БД

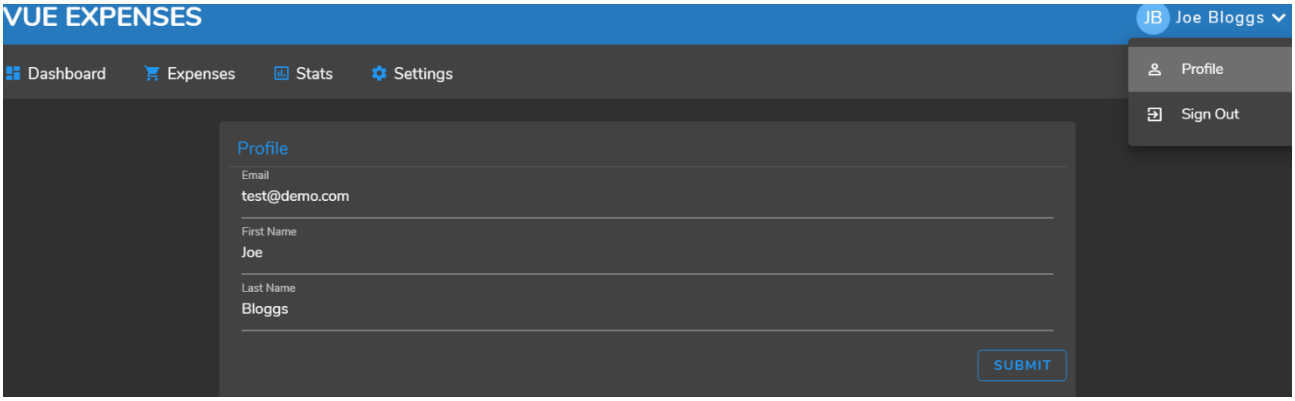

Рисунок 3.8 – Верифікація профіля

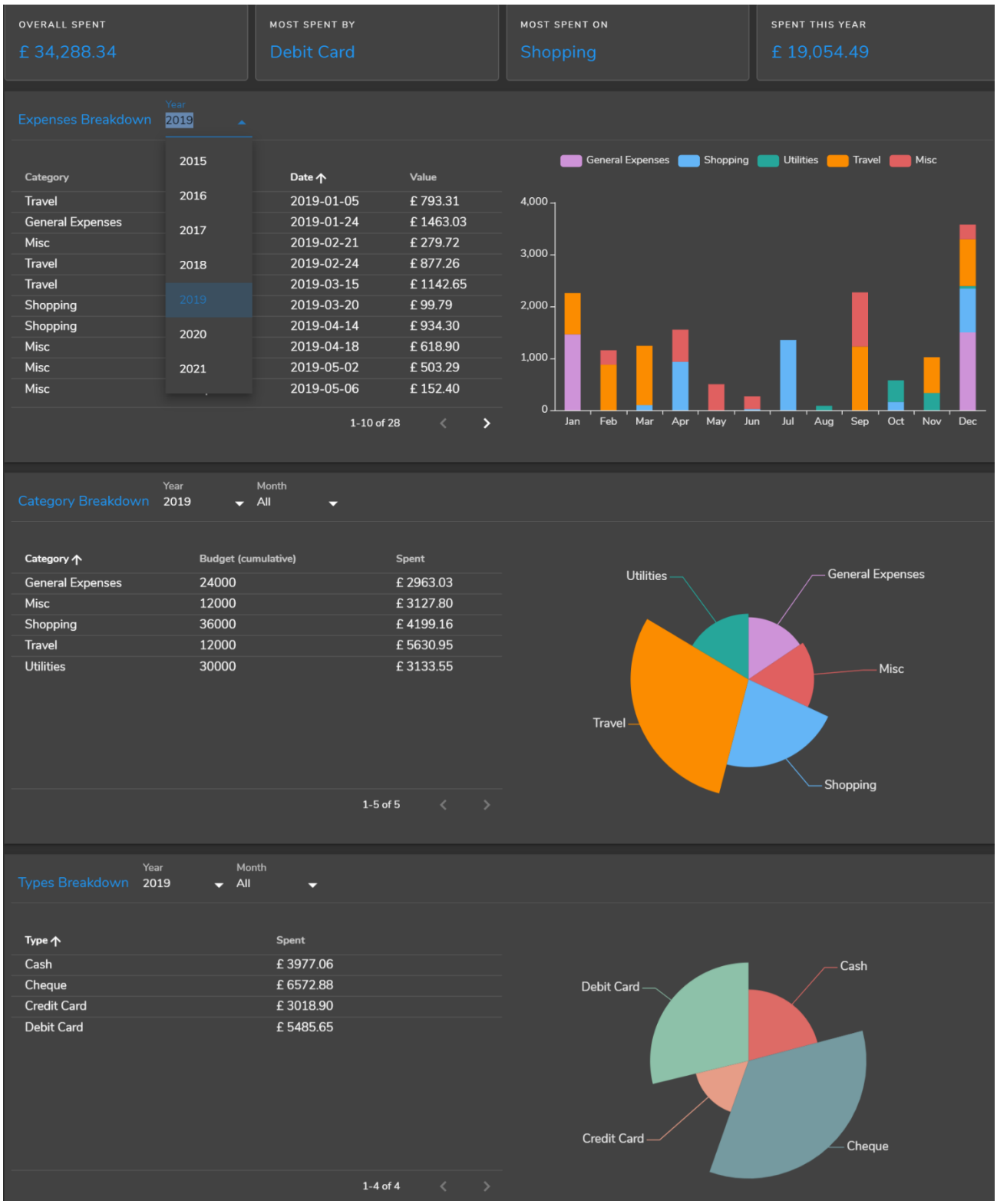

Рисунок 3.9 – Статистика витрат

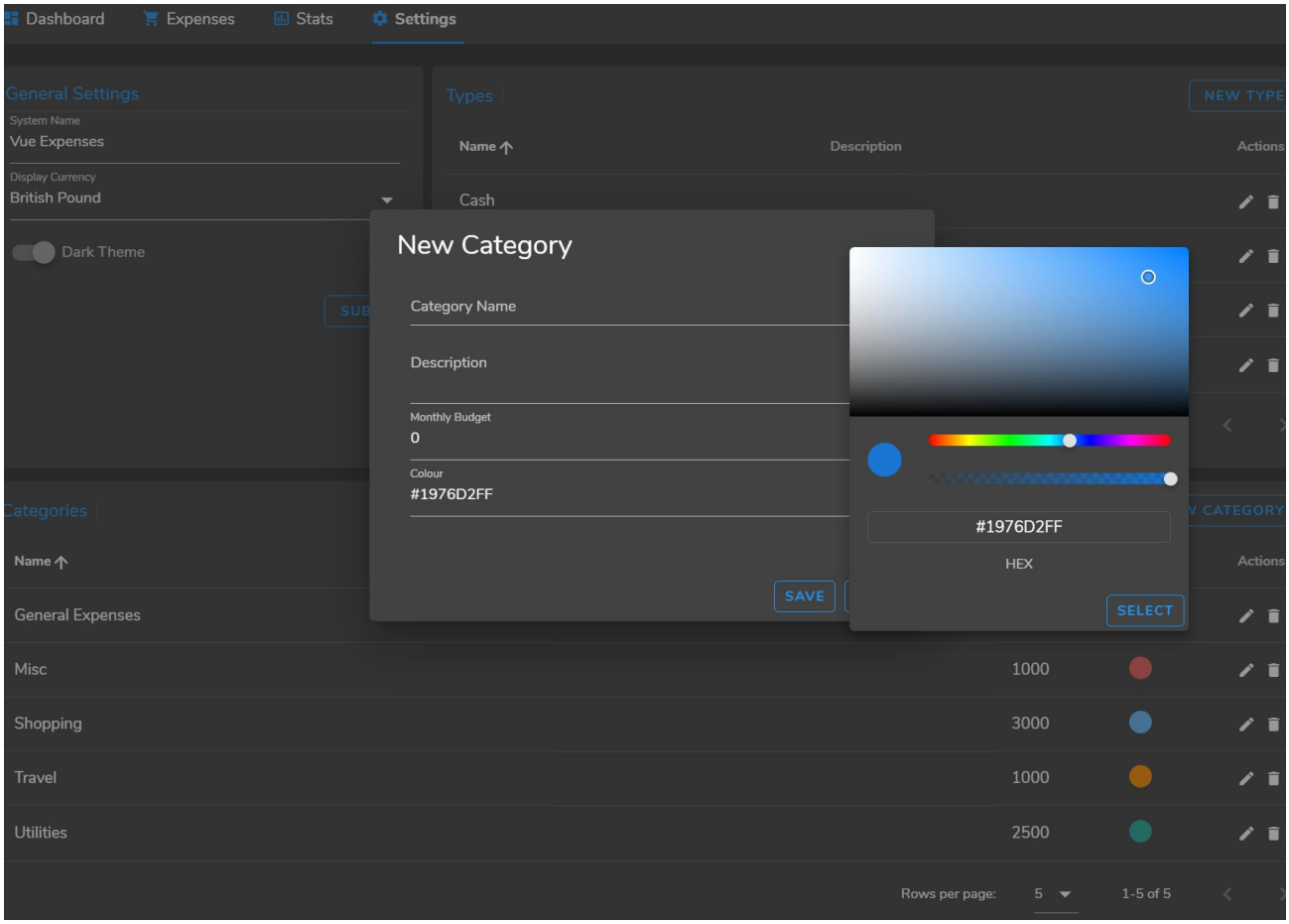

Рисунок 3.10 – Налаштування додатку

В таблиці 3.4 наведено основні тест-кейси, які застосовувались під час тестування програмної реалізації додатку для відстеження витрат з автоматичною синхронізацією транзакцій із банківської карти. Тестування було спрямоване на перевірку працездатності всіх основних модулів додатку, оцінку відповідності функціоналу визначеним цілям, а також забезпечення стабільної роботи системи у різних сценаріях.

Крім функціонального тестування, до тест-кейсів включено перевірки на продуктивність додатку при обробці великих обсягів транзакцій, тестування відображення інтерфейсу на різних пристроях та перевірку адаптивності дизайну. Це дозволило оцінити, як додаток реагує на різні сценарії використання, включаючи роботу на мобільних пристроях з різною роздільною здатністю екрана.

Таблиця 3.4 – Основні тест-кейси тестування додатку відстеження витрат з автоматичною синхронізацією транзакцій з карти

ID Тест-Кейсу	Назва Тест-Кейсу	Опис Тесту	Попередні Умови	Кроки	Очікуваний Результат	Статус
TC-001	Тестування додавання витрат	Перевірка функціональності додавання нової витрати	Відкритий додаток, авторизований користувач	1. Відкрити форму додавання витрат. 2. Ввести дані витрати. 3. Натиснути "Зберегти".	Витрата успішно додається і відображається у списку.	Пройдено
TC-002	Тестування редагування витрат	Перевірка функціональності редагування існуючої витрати	Існує хоча б одна витрата у списку	1. Обрати витрату зі списку. 2. Внести зміни. 3. Натиснути "Зберегти".	Зміни збережено, оновлена витрата відображається.	Пройдено
TC-003	Тестування видалення витрат	Перевірка функціональності видалення витрати	Існує хоча б одна витрата у списку	1. Обрати витрату зі списку. 2. Натиснути "Видалити".	Витрата успішно видаляється зі списку.	Пройдено
TC-005	Тестування світлої теми	Перевірка інтерфейсу у світлій темі	Увімкнена світла тема	1. Переконатися, що тема обрана. 2. Перевірити відображення елементів інтерфейсу.	Всі елементи чітко видимі, немає помилок у відображенні.	Пройдено

Продовження таблиці 3.4

ID Тест-Кейсу	Назва Тест-Кейсу	Опис Тесту	Попередні Умови	Кроки	Очікуваний Результат	Статус
TC-006	Тестування темної теми	Перевірка інтерфейсу у темній темі	Увімкнена темна тема	1. Переконатися, що тема обрана. 2. Перевірити відображення елементів інтерфейсу.	Всі елементи чітко видимі, немає помилок у відображенні.	Пройдено
TC-007	Тестування адаптивності інтерфейсу	Перевірка відображення додатку на різних роздільних здатностях	Підключено декілька моніторів з різними здатностями	1. Відкрити додаток на кожному з моніторів. 2. Перевірити відображення інтерфейсу.	Інтерфейс коректно відображається на всіх моніторах.	Пройдено
TC-008	Тестування продуктивності при навантаженні	Перевірка стабільності роботи додатку при великих обсягах даних	Є багато витрат в базі даних	1. Додати велику кількість витрат. 2. Виконати навантажувальні операції (генерація звітів).	Додаток стабільно працює без затримок і помилок.	Пройдено

Продовження таблиці 3.4

ID Тест-Кейсу	Назва Тест-Кейсу	Опис Тесту	Попередні Умови	Кроки	Очікуваний Результат	Статус
ТС-009	Тестування безпеки автентифікації	Перевірка роботи механізмів автентифікації та захисту паролів	Додаток налаштовано для роботи з акаунтами	1. Виконати вхід з різними обліковими записами. 2. Перевірити роботу збереження паролів.	Вхід успішний, паролі зберігаються і захищені належним чином.	Пройдено

2.4 Висновок до розділу 3

У даному розділі було програмно реалізовано веб частину додатку, використовуючи можливості vue expense. Також було реалізовано додаткові сторінки клієнтської частини, і реалізовано серверну частину програми, з впровадженням синхронізації зі сторонніми сервісами, і даному випадку – банківськими АПІ.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудити науково технічної розробки

Мета проведення комерційного і технологічного аудиту дослідження на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» полягає у визначенні науково-технічного рівня та комерційного потенціалу розробки, отриманої в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня та комерційного потенціалу розробки рекомендується проводити за допомогою 5-бальної системи за 12 критеріями, які наведені в таблиці 4.1 [26].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівцівабо витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовують ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштівта часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-якірегламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	4	3	3
7. Ринкові перспективи (конкуренція)	3	2	2
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	5	4	5
12. Практична здійсненність (розробка документів)	5	5	4
Сума балів	45	45	43
Середньоарифметична сума балів $СБ_c$	44,3		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [26].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» становить 44,3 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

4.2 Оцінювання рівня новизни розробки

Виводячи новий продукт на ринок, виробник зазвичай вважає, що інноваційність нової розробки є достатньою для того, щоб споживач сприйняв її як нову. Однак часто думки виробника та споживача щодо рівня новизни розбігаються. Тому важливо оцінити рівень новизни розробки, створеної в результаті дослідження на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти».

Визначення рівня новизни та ступеня інтегральної інноваційності є важливим, оскільки воно дозволяє забезпечити однакове позитивне сприйняття новизни як виробником, так і споживачем, що, у свою чергу, гарантує успішність продукту на ринку. Це підвищує його шанси стати популярним, забезпечити попит і дозволити відшкодувати витрати, вкладені у розробку та виробництво.

Для оцінювання рівня новизни використовуємо експертний метод, у якому новий продукт порівнюється з існуючими на ринку аналогами за ключовими характеристиками у системі «краще-гірше». Відносний рівень новизни

визначається на основі порівняння з аналогами або продуктами, що є досить близькими до них.

Для кожного виду новизни використовуються різні чинники, що впливають на її рівень, і вони оцінюються в балах. Чим вищий бал — тим вищий рівень новизни. Оцінка базується на експертних думках, де кожен чинник оцінюється в діапазоні від -5 (значно гірше за аналог) до +5 (значно краще за аналог). Отримані результати впорядковуються в лист оцінювання (таблиця 4.4).

Таблиця 4.4 – Лист оцінювання рівня новизни експертами

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
1		*2*	*3*	*4*
Споживча новизна	Питома вага 0,26	Максимальний бал $Б_{i\ max}$		25
1. Зміна поведінкових звичок споживача		2	2	2
2. Ступінь задоволення потреб і запитів		4	4	4
3. Спосіб задоволення потреби		2	2	2
4. Формування нової потреби		4	3	3
5. Формування нового споживача		0	0	0
Середній бал експертів $Б_{i\ отр}$		11		
Товарна новизна	Питома вага 0,21	Максимальний бал $Б_{i\ max}$		30
1. Параметричні зміни показників продукції				
1.1. Якісні		3	3	4
1.2. Технічні		3	4	3
1.3. Економічні		3	3	3
1.4. Сервісні		4	4	4
2. Якість продукції по відношенню до конкурентів		3	3	3
3. Функціональні зміни		4	4	4
Середній бал експертів $Б_{i\ отр}$		21		
Виробнича новизна	Питома вага 0,014	Максимальний бал $Б_{i\ max}$		25
1. Рівень унікальності товару для підприємства		5	5	5
2. Рівень унікальності для галузі		3	3	3
3. Рівень унікальності товару для країни		1	1	1
4. Зміна виробничої системи		4	4	4
5. Відносно існуючого асортименту		3	2	3
Середній бал експертів $Б_{i\ отр}$		16		
Прогресивна новизна	Питома вага 0,2	Максимальний бал $Б_{i\ max}$		25
1. Зміна технології виготовлення		4	4	4

Продовження таблиці 4.4

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
4. Зміна конструктивного виконання		3	3	3
5. Рівень застосування інновацій		2	2	2
Середній бал експертів $Б_{i\ отр}$		10		
Ринкова новизна	Питома вага 0,1	Максимальний бал $Б_{i\ max}$		20
1. Новий виріб на новому ринку		0	0	0
2. Новий виріб на відомому ринку		4	4	4
3. Модернізований виріб		3	3	3
4. Нова модель		1	1	1
Середній бал експертів $Б_{i\ отр}$		8		
Екологічна новизна	Питома вага 0,035	Максимальний бал $Б_{i\ max}$		20
1. Рівень екологічної чистоти технологіївиробництва		5	5	5
2. Рівень впровадження мало- та безвідходнихтехнологій		5	5	5
3. Рівень екологічно небезпечних режимівексплуатації продукції		5	5	5
4. Рівень забруднення навколишнього середовища		5	5	5
Середній бал експертів $Б_{i\ отр}$		20		
Соціальна новизна	Питома вага 0,036	Максимальний бал $Б_{i\ max}$		20
1. Використання нового товару приводить допокращення стану здоров’я нації		0	0	0
2. Використання нового товару приводить дозростання доходів населення		0	0	0
3. Виробництво нового товару приводить до збільшення (зменшення) кількості робочих місцьна підприємстві		4	5	4
4. Виробництво нового товару приводить допідвищення кваліфікації персоналу		3	4	3
Середній бал експертів $Б_{i\ отр}$		8		
Маркетингова новизна	Питома вага 0,145	Максимальний бал $Б_{i\ max}$		20
1. Нові методи маркетингових досліджень		0	0	0
2. Вживання нових стратегій сегментації ринку		3	3	3
3. Вибір нової маркетингової стратегії обхвату і розвитку цільового сегмента		1	1	1
4. Побудова нових каналів збуту		2	1	1
Середній бал експертів $Б_{i\ отр}$		5		

Значення *i*-го виду новизни розрахуємо за формулою [27]:

$$I_i = \frac{Б_{i\,отр}}{Б_{i\,MAX}}, \qquad (4.1)$$

де Бі отр – отримана кількість балів за шкалою оцінок чинників, що визначають і-й вид новизни;

Бі МАХ – максимальна кількість балів, що може бути отримана за і-м видом новизни.

Загальний рівень інтегральної новизни розраховуємо шляхом перемноження отриманого значення і-го виду новизни на її вагомість, причому вагомість і-го виду новизни визначаємо експертним методом, за формулою [27]:

$$N_{інт} = \sum_{i}^{n} W_i \cdot I_i, \qquad (4.2)$$

де Nінт – рівень інтегральної (сукупної) новизни;

Wi – вагомість (питома вага) і-го виду новизни;

n – загальна кількість видів новизни.

$$N_{інт} = (0{,}26 \cdot 11/25)+(0{,}21 \cdot 21/30)+(0{,}014 \cdot 16/25)+(0{,}2 \cdot 10/25)+(0{,}1 \cdot 8/20)+$$
$$(0{,}035 \cdot 20/20)+(0{,}036 \cdot 8/20)+(0{,}145 \cdot 5/20) = 0{,}481.$$

Отримане значення інтегрального рівня новизни зіставляємо зі шкалою, що наведена в табл. 4.5 [26].

Таблиця 4.5 – Рівні новизни нового товару та їхня характеристика

Рівні новизни товару	Значення інтег-ральної новизни	Характеристика товару	Вид нового товару
Найвища	1,00	Абсолютно новий товар	Новий товар, що наділений ознаками інноваційності (інноваційний товар)
Висока	0,8…0,99	Товар, який не має аналогів	
Значуща	0,6…0,79	Принципова зміна споживчих властивостей товару	
Достатня	0,4…0,59	Принципова технологічна модифікація товару	
Незначна	0,2…0,39	Кардинальна зміна параметрів	Новий товар
Помилкова	0,00…0,19	Малоістотна модифікація	

Згідно таблиці 4.5 розробка відповідає рівню при значенні інтегральної новизни 0,481 - достатня новизна; за характеристикою: принципова технологічна модифікація товару; вид розробки - новий товар, що наділений ознаками інноваційності (інноваційний товар).

4.3 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення розрахуємо за формулою [27]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \qquad (4.3)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

αі – коефіцієнт, який враховує питому вагу і-го технічного показника в загальній якості розробки. Коефіцієнт αі визначається експертним шляхом

і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

βі – відносне значення і-го технічного показника якості нової розробки.

Відносні значення βі для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{ai}}, \qquad (4.4)$$

де Іні та Іна – чисельні значення конкретного і-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \qquad (4.5)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.6.

Таблиця 4.6 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектоване програмне забезпечення	Відношення параметрів нової розробки до аналога	Питома вага показника
Оперування стандартними даними	бал	7	9	1,28	0,35
Можливість працювати з нечіткими параметрами	бал	1	7	7	0,3
Ідеальний експерт	бал	2	8	4	0,1
Професійне надання рекомендацій необізнаному користувачеві	бал	2	9	4,5	0,1
Стабільний доступв інтернет в інтернет	бал	9	9	1	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}28 \cdot 0{,}35 + 7 \cdot 0{,}3 + 4 \cdot 0{,}1 + 4{,}5 \cdot 0{,}1 + 1 \cdot 0{,}15 = 3{,}55.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,55 рази.

4.4 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.4.1 Витрати на оплату праці

До статті «Витрати на оплату праці» входять витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, а також науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим співробітникам, безпосередньо зайнятим виконанням конкретної теми. Оплата здійснюється згідно з посадовими окладами, відрядними розцінками або тарифними ставками відповідно до чинних у організаціях систем оплати праці.

Основна заробітна плата дослідників ($З_о$) розраховується на основі посадових окладів працівників за наведеною формулою [26].

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$З_о$ = 18250,00 · 36 / 21 = 31285,71 *грн.*

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число дні вроботи	Витрати на заробітну плату, грн
Керівник проекту	18250,00	869,05	36	31285,71
Відповіданьний виконавець (інженер-програміст вищоїкатегорії)	16000,00	761,90	36	27428,57
Консультант-аналітик	16500,00	785,71	7	5500,00
Всього				64214,29

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (4.8)$$

де ММ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо Мм=8000,00 грн;

Кі – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [32];

Кс – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

Тр – середнє число робочих днів в місяці, приблизно Тр = 21 дн;

tзм – тривалість зміни, год.

$C1 = 8000,00 \cdot 1,10 \cdot 1,35 / (21 \cdot 8) = 70,71$ грн.

$Зр1 = 70,71 \cdot 12,00 = 848,52$ грн.

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення обладнання робочих місць інженерно-технічних працівників (розробників і дослідників)	12,00	2	1,10	70,71	848,52
Установка обчислювального обладнання для проведення моделювання та досліджень	8,00	3	1,35	72,68	581,46
Встановлення програмного забезпечення інструментарію розробки системи надання рекомендацій	7,50	4	1,50	80,76	605,69
Введення кодів базових модулів програмного забезпечення	15,00	5	1,70	91,53	1372,90
Введення тестувально-наладочної бази даних	16,00	4	1,50	80,76	1292,14
Тестування системи надання рекомендацій	12,50	3	1,35	72,68	908,54
Всього					5609,42

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (64214,29 + 5609,42) · 11 / 100% = 7680.60 грн.

4.4.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (64214,29 + 5609,42 + 7665,43) · 22 / 100% = 17047.61грн.

4.4.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основніта допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \qquad (4.11)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 3,0 \cdot 195,00 \cdot 1,05 - 0 \cdot 0 = 614,25$ *грн.*

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
ПАПІР CANYO A4 STAND (PAPER_MS80/MS .A4.80.ST)	195,00	3,0	0	0	614,25
Папір CANYO A5, 80 г, 500 арк. Premium, клас А (149366)	120,00	4,0	0	0	504,00
Настільний набір Buromax 16 items, black (BM.6302-01)	210,00	4,0	0	0	882,00
Органайзер настільний металевий, 22x14x13 см, чорний H-Tone (JJ41220)	240,00	4,0	0	0	1008,00
Картридж Canon 725 Black (3484B002)	1230,00	3,0	0	0	3874,50
USB накопичувач DRIVE 128GB Silver (L)	240,00	2,0	0	0	504,00
Всього					7701,75

4.4.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі (Кв), які використовують при проведенні НДР на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.12)$$

де Нj – кількість комплектуючих j-го виду, шт.;

Цj – покупна ціна комплектуючих j-го виду, грн;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15).

Кв = 1 · 2369,00 · 1,05 = 2487,45 грн.

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній жорсткий диск 2.5" 2TB TOSHIBA (HDTB520EK3AA)	1	2369,00	2487,45
Система інтерфейсного забезпечення	1	4299,00	4513,95
NVidia GeForce GTX 1050, 4 ГБ	1	6520,00	6846,00
Всього			13847,40

4.4.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.13)$$

де Ці – ціна придбання одиниці спецустаткування даного виду, марки, грн;

Спр.і –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

Кі – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (Кі = 1,10…1,12);

k – кількість найменувань устаткування.

Вспец = 13630,00 · 1 · 1,1 = 14993,00 грн.

Отримані результати зведемо до таблиці 4.11.

Таблиця 4.11 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Джерело безперебійного живлення LPE-B-PSW-2300BA/1600Вт	1	13630,00	14993,00
Всього			14993,00

4.4.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.14)$$

де Ціпрг – ціна придбання одиниці програмного засобу даного виду, грн;

Спрг.і – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

Кі – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10…1,12);

k – кількість найменувань програмних засобів.

Впрг = 14300,00 · 1 · 1,05 = 15015,00 грн.

Отримані результати зведемо до таблиці 4.12.

Таблиця 4.12 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладне середовище розробки ПЗ	1	14300,00	15015,00
Програмне забезпечення з відкритим кодом Google Colab	1	2100,00	2205,00
Всього			17220,00

4.4.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (4.15)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо,які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень підчас досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів,приміщень тощо, років.

$A_{обл} = (92569{,}00 \cdot 2) / (3 \cdot 12) = 5142{,}72$ грн.

Проведені розрахунки зведемо до таблиці 4.13.

Таблиця 4.13 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Електронно-обчислювальний центр моделювання та тестування системи КОМП'ЮТЕР HP ELITEONE 870 G9 AIO / I7-13700 (7B0P5EA)	92569,00	3	2	5142,72
Робоче місце Project Manager 2/0	9899,00	5	2	329,97
Робоче місце інженера-дослідника Ноутбук Lenovo IdeaPad 330-15ICH (81FK00G1RA)	16599,00	3	2	922,17
Пристрій виводу інформації CANON Laser 6900	8699,00	4	2	362,46
Система мережевого обладнання передачі даних	19200,00	4	2	800,00
Офісна оргтехніка	10100,00	5	2	336,67
Приміщення лабораторії	255500,00	25	2	1703,33
ОС Windows 11	5645,00	3	2	313,61
Прикладний пакет Microsoft Office 2019	5155,00	3	2	286,39
Прикладний пакет моделювання	7646,00	3	2	424,78
Всього				10622,09

4.4.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.16)$$

де Wyi – встановлена потужність обладнання на визначеному етапі розробки, кВт;

ti – тривалість роботи обладнання на етапі дослідження, год;

Це – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 10 грн;

Квпі – коефіцієнт, що враховує використання потужності, $K_{вni}$ <1;

ηі – коефіцієнт корисної дії обладнання, ηі<1. Ве = 0,32 · 200,0 · 10 · 0,95 / 0,97 = 626,00 грн.

Проведені розрахунки зведемо до таблиці 4.14.

Таблиця 4.14 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Електронно-обчислювальний центр моделювання та тестування системи КОМП'ЮТЕР HP ELITEONE 870 G9 AIO / I7-13700 (7B0P5EA)	0,32	200,0	626,00
Робоче місце Project Manager 2/0	0,06	280,0	126,00
Робоче місце інженера-дослідника Ноутбук Lenovo IdeaPad 330-15ICH (81FK00G1RA)	0,05	260,0	97,50
Пристрій виводу інформації CANON Laser 6900	0,25	5,0	9,38
Система мережевого обладнання передачі даних	0,10	200,0	150,00
Офісна оргтехніка	0,55	2,0	8,25
Фізичне обладнання серверу бази даних	0,20	200,0	300,00
Всього			1317,13

4.4.9 Службові відрядження

Витрати за статтею «Службові відрядження» відсутні.

4.4.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (4.17)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (64214,29 + 5471,42) · 30 / 100% = 20905,71 грн.

4.4.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (4.18)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (64214,29 + 5471,42) · 60 / 100% = 41811,42 грн.

4.4.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків;

витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (4.19)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = *(64214,29 + 5471,42) · 100 / 100% = 69685,70 грн.*

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \qquad (4.20)$$

$B_{заг}$= *64214,29 +5471,42 +7665,43 +17017,25 +7701,75 +13847,40 +14993,00 + 17220,00 + 10622,09 +1171,13 +0,00 +20905,71 +41811,42 +69685,70 =292326,58 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (4.21)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,95.

ЗВ = 292326,58 / 0,95 = 307712,19 грн.

4.5 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів за темами «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти»

$$ЗВ_1 = 307712,19 \text{ грн.}$$

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвесторавеличини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	25000	30000	20000	10000

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 100000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 60,00 грн;

$\pm\Delta Ц_о$ - зміна вартості послуги від впровадження результатів, приймемо 32,88 грн

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.22)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор,у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(32,88·100000,00+92,88·25000)·0,83·0,4·(1-0,18/100%)=1527266,40 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(32,88·100000,00+92,88·55000)·0,83·0,4·(1-0,18/100%)=2285835,94 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(32,88·100000,00+92,88·75000)·0,83·0,4·(1-0,18/100%)=2791548,96 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(32,88·100000,00+92,88·85000)·0,83·0,4·(1-0,18/100%)=3044405,47 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.23)$$

ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =1527266,40/(1+0,15)^1+2285835,94/(1+0,15)^2+2791548,96/(1+0,15)^3+
+3044405,47/(1+0,15)^4=1328057,74+1728420,37+1835488,75+1740648,71=
=6632615,57 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.24)$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію; приймаємо кінв = 1,7;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів; приймаємо 536 923,03 грн.

PV = $k_{інв}$ · ЗВ = 1,7 · 536 923,03 = 916 843,37 грн.

Абсолютний економічний ефект Еабс для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \qquad (4.25)$$

ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 6 632 615,57 грн;

PV – теперішня вартість початкових інвестицій, 916 843,37 грн.

$E_{абс} = ПП - PV = 6\,632\,615{,}57 - 916\,843{,}37 = 5\,715\,772{,}20$ *грн.*

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.26)$$

Де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 5 715 772,20 грн;

PV – теперішня вартість початкових інвестицій, 916 843,37 грн;

Тж – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до завершення отримання позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+5715772{,}20/916843{,}37)^{1/4} = 0{,}64.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \qquad (4.27)$$

де:

d – середньозважена ставка за депозитними операціями в комерційних банках; у 2023 році в Україні d = 0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймаємо 0,3.

$\tau_{мін}$ = 0,1+0,3 = 0,4 < 0,64 свідчить про те, що внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, є вищою за мінімальну внутрішню дохідність. Тобто, інвестувати в науково-дослідну роботу за темою «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» доцільно.

Період окупності інвестицій Ток, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.28)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ОК}$ = 1 / 0,64 = 1,56 р.

$T_{ОК}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.6 Висновок до розділу 4

Згідно з результатами проведених досліджень, рівень комерційного потенціалу розробки на тему «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти» становить 44,3 бала, що свідчить про високу комерційну цінність проведених досліджень.

При оцінюванні технічних параметрів за узагальненим коефіцієнтом якості, науково-технічна розробка приблизно в 3,55 рази перевершує існуючі аналоги.

Також термін окупності становить 1,56 року, що є меншим за 3 роки, що свідчить про комерційну привабливість розробки і може стимулювати потенційного інвестора профінансувати її впровадження та вихід на ринок.

Таким чином, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія відстеження витрат з автоматичною синхронізацією транзакцій з карти».

ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки відстеження витрат з автоматичною синхронізацією транзакцій з карти.

У першому розділі магістерської роботи було проведено аналіз предметної області управління фінансами, вивчено існуючі системи-аналогі, що дозволяють автоматизувати процес обліку фінансових операцій, а також виявлено типові проблеми таких додатків. Це дало змогу обґрунтувати доцільність розробки нової інформаційної технології, яка вирішує недоліки існуючих систем.

У другому розділі розроблено модель системи відстеження витрат, описано загальний алгоритм її роботи, а також виконано аналіз сучасних технологій для реалізації додатка. Особливу увагу було приділено візуалізації даних, щоб забезпечити зручність і ефективність користування. Було спроектовано загальну структурну схему системи, яка інтегрує автоматичну синхронізацію транзакцій із карткових рахунків.

У третьому розділі виконано програмну реалізацію додатку для відстеження витрат з автоматичною синхронізацією транзакцій з карти. Було обрано мову програмування C# , що забезпечує високу продуктивність та зручність у розробці. Реалізовано функціональні модулі для автоматизації обліку витрат, а також синхронізації даних із карткових транзакцій у режимі реального часу. Також було використано методи візуалізації, такі як лінійні графіки та стовпчикові діаграми, які забезпечують глибоке розуміння фінансових тенденцій і дозволяють користувачам краще управляти своїм бюджетом. Алгоритм роботи додатку VUE EXPENSES демонструє зручність та ефективність у зборі, обробці та відображенні фінансових даних, що сприяє зменшенню помилок і підвищенню точності інформації.

У четвертому розділі магістерської роботи було проведено дослідження, яке показало, що комерційний потенціал розробки інформаційної технології відстеження витрат з автоматичною синхронізацією транзакцій з карти оцінюється як високий, досягаючи 44,3 бала. Технічно ця розробка значно

перевершує існуючі аналоги, з узагальненим коефіцієнтом якості, що виявляється в 3,55 рази вищим. Термін окупності становить 1,56 року, що відповідає вимогам комерційної привабливості та може зацікавити потенційних інвесторів.

Тестування додатку показало, що він працює коректно і відповідає встановленим вимогам. Основний акцент у тестуванні був зроблений на інтерфейс, оскільки функціональність дублюється, але оптимізовано. Аналіз ефективності розробленого додатку в порівнянні з аналогами виявив його конкурентоспроможність, підкресливши значні переваги у зручності використання та точності відстеження витрат.

Таким чином було розширено функціональкі можливості за допомогою збільшення кількості доступних банків для снихронізації до 2. А також додаванням атоматичного розподілення витрат за категоріями на основі коду бізнесу в якому було здійснено оплату

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. В. Колодний, Д. О. Щупак. Удосконалення відстеження витрат з автомтичною синхронізацією з карти. Матеріали Всеукраїнської науково-пракичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» – [Електронний ресурс] – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22775/18772
2. С. Г. Арбузов. Банківська енциклопедія – 2011 р. – С. 504.
3. Кулик В. А., Щербак С. В. "Основи синхронізації в системах з розподіленими базами даних". – 2019– С. 67–73.
4. Павлюк В. П., Сметанін О. О. "Розробка REST API для розподілених інформаційних систем" – 2019. – С. 54–62.
5. Mint – [Електронний ресурс] – Режим доступу: https://mint.intuit.com/
6. YNAB – [Електронний ресурс] — Режим доступу: https://www.ynab.com/features
7. Personal Capital – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Personal_Capital
8. Spendee – [Електронний ресурс] – Режим доступу: https://medium.com/spendee/grow-your-wealth-in-2022-with-spendee-lifetime-access-573076a6cd18
9. MoneyControl – [Електронний ресурс] – Режим доступу: https://medium.com/@rleeb02/control-your-money-control-your-life-b4b9646b0c33
10. Романенко В. М., Федоренко О. О. "Застосування D3.js для візуалізації великих даних" – 2021– С. 67-74.
11. Романенко В. М., Білоножко О. А. "Візуалізація фінансових даних за допомогою графічних бібліотек D3.js та Chart.js" – 2021. – С. 56-62.
12. Кравченко О. П., Мельник М. С. "Автоматизація формування звітів у фінансових додатках" – 2020. – С. 89-96.
13. Павленко М. В., Іванченко О. С. "Розробка високопродуктивних додатків з використанням .NET 5.0 та CQRS патерну" – 2021– С. 45-52.
14. Соловйов А. М., Дмитренко В. С. "Розробка адаптивних інтерфейсів з використанням Vue.js та Vuetify" – 2021. – С. 58-66.

14. Vuetify Documentation – [Електронний ресурс] – Режим доступу: https://vuetifyjs.com/en/getting-started/overview/

15. Vue.js 3.2 для розробників. Методика створення компонентів – 2022. – С. 12-20.

16. Microsoft Docs – UI/UX Testing Best Practices. – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/

17. Сидоренко А. “Особливості тестування продуктивності веб додатків” –2020. – С. 102-109.

18. JWT.io – JSON Web Token Introduction. – [Електронний ресурс] – Режим доступу: https://jwt.io/introduction

19. Adams S. “Implementing JWT Authentication in .NET Core” – 2021. – P. 78-94.

20. Microsoft Docs – .NET Security and Cryptography. – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/standard/security/

21. Security Testing Best Practices – [Електронний ресурс] – Режим доступу: https://owasp.org/www-project-top-ten/

22. W3C Web Accessibility Initiative (WAI). – [Електронний ресурс] – Режим доступу: https://www.w3.org/WAI/

23. Testing Vue.js Applications by Edd Yerburgh – 2018. – P. 78-90.

24. Vue Test Utils Documentation. – [Електронний ресурс] – Режим доступу: https://vue-test-utils.vuejs.org/

25. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

26. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ,2016. – 113 с.

ДОДАТКИ

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 97,00% Схожість 3,00%

Аналіз звіту подібності (відмітити потрібне):

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи Щупак Д.О.

Керівник роботи Колодний В.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов’язковий)
Лістинг програми

```
using System;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using vue_expenses_api.Domain;

namespace vue_expenses_api.Infrastructure;

public class ExpensesContext : DbContext
{
    private readonly IConfiguration _configuration;
    private readonly ILoggerFactory _loggerFactory;

    public ExpensesContext(
        DbContextOptions options,
        IConfiguration configuration,
        ILoggerFactory loggerFactory) : base(options)
    {
        _configuration = configuration;
        _loggerFactory = loggerFactory;
    }

    public DbSet<User> Users { get; set; }
    public DbSet<Expense> Expenses { get; set; }
    public DbSet<ExpenseCategory> ExpenseCategories { get; set; }
    public DbSet<ExpenseType> ExpenseTypes { get; set; }
    public DbSet<RefreshToken> RefreshTokens { get; set; }
```

```
protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlite(_configuration.GetConnectionString("DefaultConnection"));
    optionsBuilder.UseLoggerFactory(_loggerFactory);
}

public override Task<int> SaveChangesAsync(
    CancellationToken cancellationToken = new())
{
    UpdateAuditFields();
    return base.SaveChangesAsync(cancellationToken);
}

public override int SaveChanges()
{
    UpdateAuditFields();
    return base.SaveChanges();
}

private void UpdateAuditFields()
{
    // get entries that are being Added or Updated
    var modifiedEntries = ChangeTracker.Entries()
        .Where(x => (x.State == EntityState.Added || x.State == EntityState.Modified));

    foreach (var entry in modifiedEntries)
    {
        var entity = entry.Entity as Entity;
        var now = DateTime.Now;

        if (entry.State == EntityState.Added)
        {
            entity.CreatedAt = now;
        }
```

```
            entity.UpdatedAt = now;
        }
    }

    protected override void OnModelCreating(
        ModelBuilder modelBuilder)
    {
        var navigation = modelBuilder.Entity<User>()
            .Metadata.FindNavigation(nameof(User.RefreshTokens));
        navigation.SetPropertyAccessMode(PropertyAccessMode.Field);

        modelBuilder.Entity<RefreshToken>()
            .HasOne(d => d.User)
            .WithMany(e => e.RefreshTokens)
            .IsRequired()
            .OnDelete(DeleteBehavior.Cascade);;

        modelBuilder.Entity<Expense>()
            .HasOne(d => d.Category)
            .WithMany(e => e.Expenses);

        modelBuilder.Entity<Expense>()
            .HasOne(d => d.Type)
            .WithMany(e => e.Expenses);
    }
}
<template>
  <div>
    <v-container>
      <v-layout row justify-space-between>
        <v-flex xs12 md6>
          <v-card class="pa-2 mr-2" raised min-height="350px">
            <div class="blue--text px-2 py-1 text-capitalize font-weight-medium">Add New
Expense</div>
```

```
          <v-divider></v-divider>
          <ExpenseForm
            :expense="expense"
            :onSubmitClick="saveExpense"
            :loading="loading"
            ref="form"
          />
        </v-card>
      </v-flex>
      <v-flex xs12 md6>
        <v-card
          :class="{'pa-2 mr-2 mt-2': $vuetify.breakpoint.smAndDown, 'pa-2 mr-2':
$vuetify.breakpoint.mdAndUp}"
          tile
          min-height="340px"
          height="100%"
        >
          <div
            class="blue--text px-2 py-1 text-capitalize font-weight-medium"
          >Budget (Current Month)</div>
          <v-divider></v-divider>
          <DoughnutChart
            :height="75"
            :theme="user.theme"
            :showLabel="true"
            :showLabelLines="true"
            :seriesData="monthlyBudget.data"
            :centerY="50"
            :pieRadiusOuter="75"
          />
          <div class="d-flex justify-space-around text-subtitle-2 px-12 mx-12">
            <div>
              <div>Limit</div>
              <div>{{`${user.displayCurrency} ${monthlyBudget.totalBudget}`}}</div>
            </div>
```

```
      <div>
        <div>Spent</div>
        <div>{{`${user.displayCurrency} ${monthlyBudget.totalSpent}`}}</div>
      </div>
    </div>
  </v-card>
</v-flex>
<v-flex xs12 md12>
  <v-container px-0 pb-0>
    <v-card class="pa-2 mr-2" tile>
      <div
        class="blue--text px-2 py-1 text-capitalize font-weight-medium"
      >Budgets By Categories (Current Month)</div>
      <v-divider></v-divider>
      <div class="category-budgets">
        <div
          v-for="budget in monthlyBudgetsByCategory"
          :key="budget.name"
          class="category-budgets-budget"
        >
          <DoughnutChart
            :titleText="budget.name"
            :showTitle="true"
            :height="90"
            :titleFontSize="14"
            :theme="user.theme"
            :showTooltip="false"
            :seriesData="budget.monthlyBudget"
          />
        </div>
      </div>
    </v-card>
  </v-container>
</v-flex>
<v-flex xs12 md12>
```

```
      <v-flex>
        <v-container>
          <v-layout row wrap>
            <v-flex xs12 md12>
              <v-card class="pa-2 mr-2" tile>
                <div
                  class="blue--text px-2 py-1 text-capitalize font-weight-medium"
                >Breakdown (Current Year)</div>
                <v-divider></v-divider>
                <v-container>
                  <v-layout row wrap>
                    <v-flex xs12 md6 style="min-height:340px;height=100%">
                      <BarChart
                        :theme="user.theme"
                        titleText="Expenses"
                        :seriesData="yearlyExpenses"
                      />
                    </v-flex>
                    <v-flex xs12 md6 style="min-height:340px;height=100%">
                      <PieChart
                        :theme="user.theme"
                        titleText="Category"
                        :seriesData="categoryExpenses"
                      />
                    </v-flex>
                  </v-layout>
                </v-container>
              </v-card>
            </v-flex>
          </v-layout>
        </v-container>
      </v-flex>
    </v-flex>
  </v-layout>
</v-container>
```

```
  </div>
</template>
<script>
import ExpenseForm from "@/components/ExpenseForm";
import DoughnutChart from "@/components/Charts/DoughnutChart";
import BarChart from "@/components/Charts/BarChart";
import PieChart from "@/components/Charts/PieChart";
import { mapState, mapActions, mapGetters } from "vuex";
import { CREATE_EXPENSE } from "@/store/_actiontypes";

export default {
  components: { ExpenseForm, DoughnutChart, BarChart, PieChart },
  computed: {
    ...mapGetters("statistics", [
      "monthlyBudget",
      "monthlyBudgetsByCategory",
      "yearlyExpenses",
      "categoryExpenses"
    ]),
    ...mapState({
      user: state => state.account.user
    })
  },
  data() {
    return {
      loading: false,
      dateMenu: false,
      expense: {}
    };
  },
  methods: {
    ...mapActions("expenses", [CREATE_EXPENSE]),
    saveExpense() {
      this.loading = true;
      this.CREATE_EXPENSE({
```

```
        expense: this.expense
      })
        .then(() => {
          this.expense = {};
          this.$refs.form.reset();
        })
        .finally(() => {
          this.loading = false;
        });
    }
  }
};
</script>

<style scoped>
.category-budgets {
  display: flex;
  justify-content: space-between;
  overflow: hidden;
  height: 180px;
  padding-left: 10px;
}
.category-budgets:hover {
  overflow-x: scroll;
}
.category-budgets-budget {
  width: 180px;
  flex: 0 0 auto;
}
</style>
<template>
  <div>
    <v-container>
      <v-layout row>
        <v-flex xs12 sm6 md3>
```

```
        <v-card class="pa-4 mr-2" outlined>
          <div class="text-overline mb-2">Overall Spent</div>
          <div class="text-h6 blue--text text-capitalize">{{overallSpent}}</div>
        </v-card>
      </v-flex>
      <v-flex xs12 sm6 md3>
        <v-card
          :class="{'pa-4 mt-2 mr-2': $vuetify.breakpoint.xs, 'pa-4 mr-2':
$vuetify.breakpoint.smAndUp}"
          outlined
        >
          <div class="text-overline mb-2">Most Spent By</div>
          <div class="text-h6 blue--text text-capitalize">{{mostSpentBy}}</div>
        </v-card>
      </v-flex>
      <v-flex xs12 sm6 md3>
        <v-card
          :class="{'pa-4 mt-2 mr-2': $vuetify.breakpoint.smAndDown, 'pa-4 mr-2':
$vuetify.breakpoint.mdAndUp}"
          outlined
        >
          <div class="text-overline mb-2">Most Spent On</div>
          <div class="text-h6 blue--text text-capitalize">{{mostSpentOn}}</div>
        </v-card>
      </v-flex>
      <v-flex xs12 sm6 md3>
        <v-card
          :class="{'pa-4 mt-2 mr-2': $vuetify.breakpoint.xs, 'pa-4 mr-2 mt-2':
$vuetify.breakpoint.sm, 'pa-4 mr-2': $vuetify.breakpoint.mdAndUp}"
          outlined
        >
          <div class="text-overline mb-2">Spent This Year</div>
          <div class="text-h6 blue--text text-capitalize">{{spentThisYear}}</div>
        </v-card>
      </v-flex>
```

```
      <v-flex xs12 md12 my-3>
        <v-flex>
          <ExpensesStats :years="getYears()" :theme="theme" />
        </v-flex>
      </v-flex>
      <v-flex xs12 md12 my-3>
        <v-flex>
          <ExpenseCategoryStats :years="getYears()" :months="months" :theme="theme" />
        </v-flex>
      </v-flex>
      <v-flex xs12 md12 my-3>
        <v-flex>
          <ExpenseTypeStats :years="getYears()" :months="months" :theme="theme" />
        </v-flex>
      </v-flex>
    </v-layout>
  </v-container>
 </div>
</template>
<script>
import ExpensesStats from "@/components/ExpensesStats";
import ExpenseCategoryStats from "@/components/ExpenseCategoryStats";
import ExpenseTypeStats from "@/components/ExpenseTypeStats";
import { mapState, mapGetters } from "vuex";

export default {
 components: { ExpensesStats, ExpenseCategoryStats, ExpenseTypeStats },
 methods: {
  getYears() {
   var items = [];
   var startYear = new Date().getFullYear() - 4;
   for (var i = 0; i < 7; i++) {
    items.push(startYear++);
   }
   return items;
```

```
    }
  },
  computed: {
    ...mapGetters("expenses", [
      "overallSpent",
      "mostSpentBy",
      "mostSpentOn",
      "spentThisYear"
    ]),
    ...mapState({
      theme: state => (state.account.user ? state.account.user.theme : "")
    })
  },
  data: () => ({
    months: [
      { name: "All", value: "" },
      { name: "January", value: 1 },
      { name: "February", value: 2 },
      { name: "March", value: 3 },
      { name: "April", value: 4 },
      { name: "May", value: 5 },
      { name: "June", value: 6 },
      { name: "July", value: 7 },
      { name: "August", value: 8 },
      { name: "September", value: 9 },
      { name: "October", value: 10 },
      { name: "November", value: 11 },
      { name: "December", value: 12 }
    ]
  })
};
</script>

<style scoped>
::v-deep .v-text-field__details {
```

```
  display: none;
}
::v-deep .v-data-footer .v-data-footer__select .v-input {
  margin: 0;
  margin-left: 10px;
}
::v-deep .v-select__slot {
  font-size: smaller;
  width: 100px;
}
::v-deep .v-text-field > .v-input__control > .v-input__slot:before {
  border: 0px;
}

::v-deep .v-select__slot input {
  width: 50px;
}
</style>
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІДСТЕЖЕННЯ ВИТРАТ З АВТОМАТИЧНОЮ СИНХРОНІЗАЦІЄЮ ТРАНЗАКЦІЙ З КАРТИ

Виконав: студент 2-го курсу,
групи 1КН-23м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Щупак Д.О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Колодний В.В.
(прізвище та ініціали)
«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Алгоритм роботи програми

Рисунок Г. 1 - Алгоритм роботи програми

Структура компоновки інтерфейсу системи

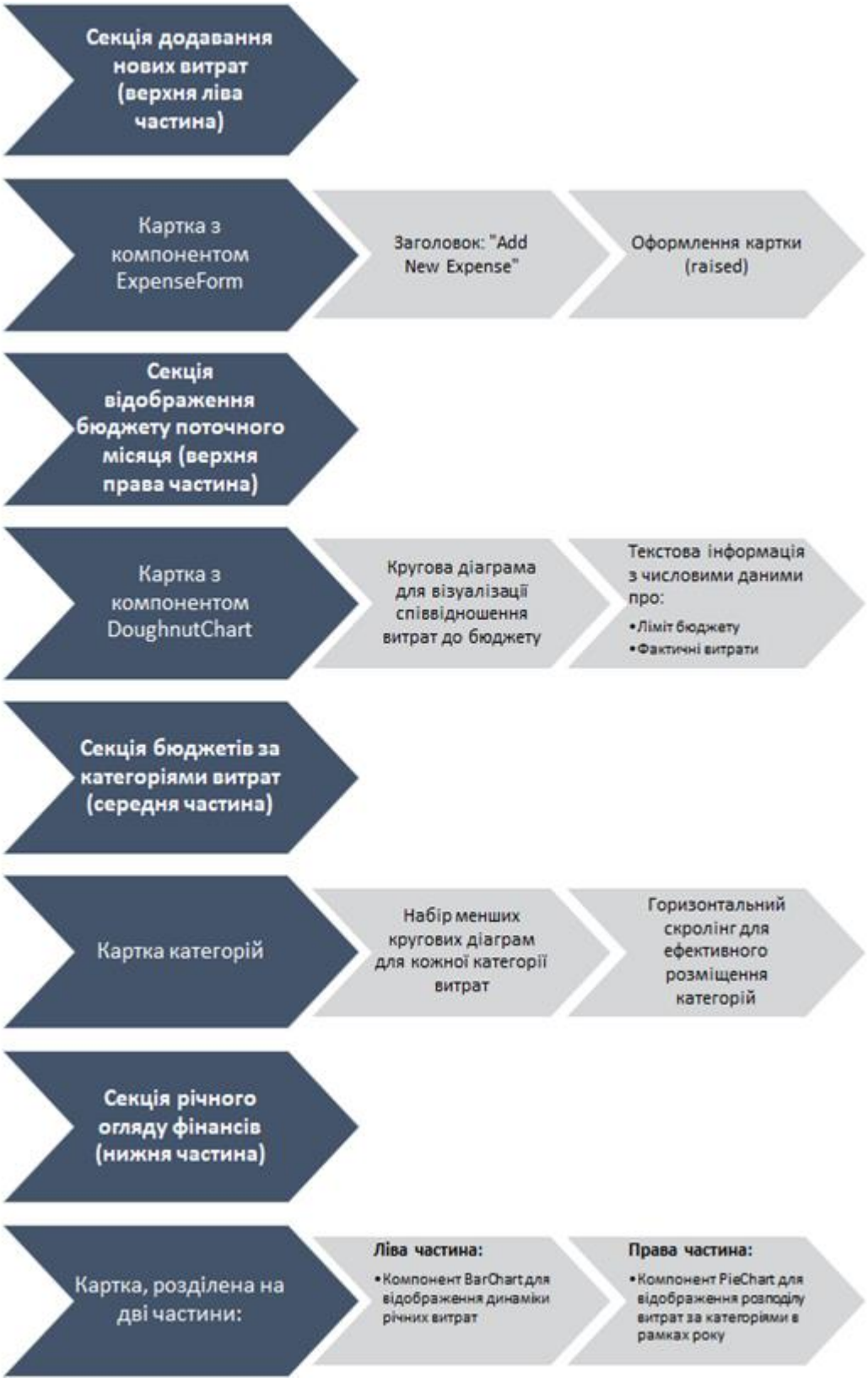

Рисунок Г.2 - Структура компоновки інтерфейсу системи

Приклад роботи програми

Рисунок Г.3 - Вкладка Дашборд

Рисунок Г.4 - Верифікація профіля

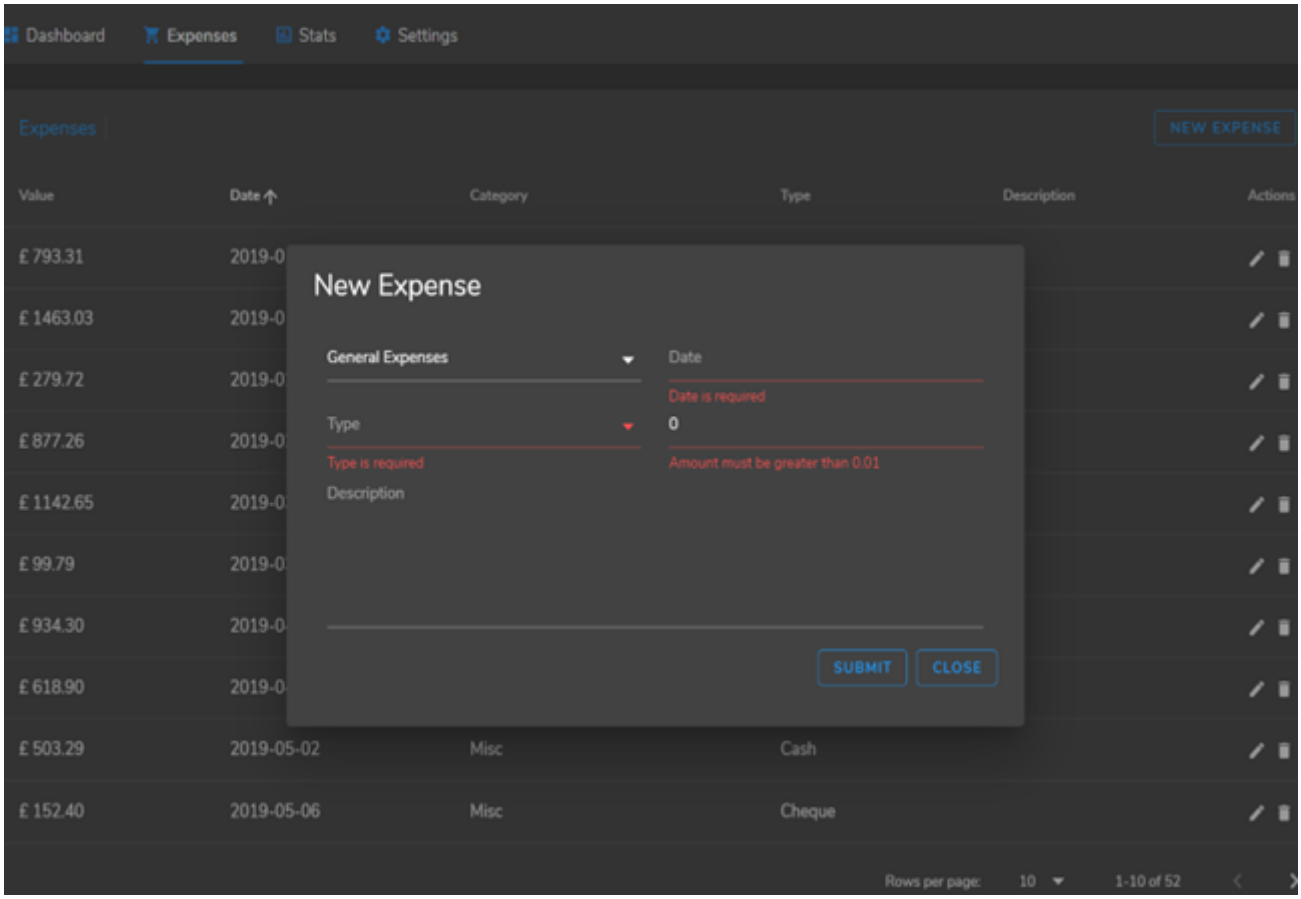

Рисунок Г.5 - Додавання нових витрат в БД

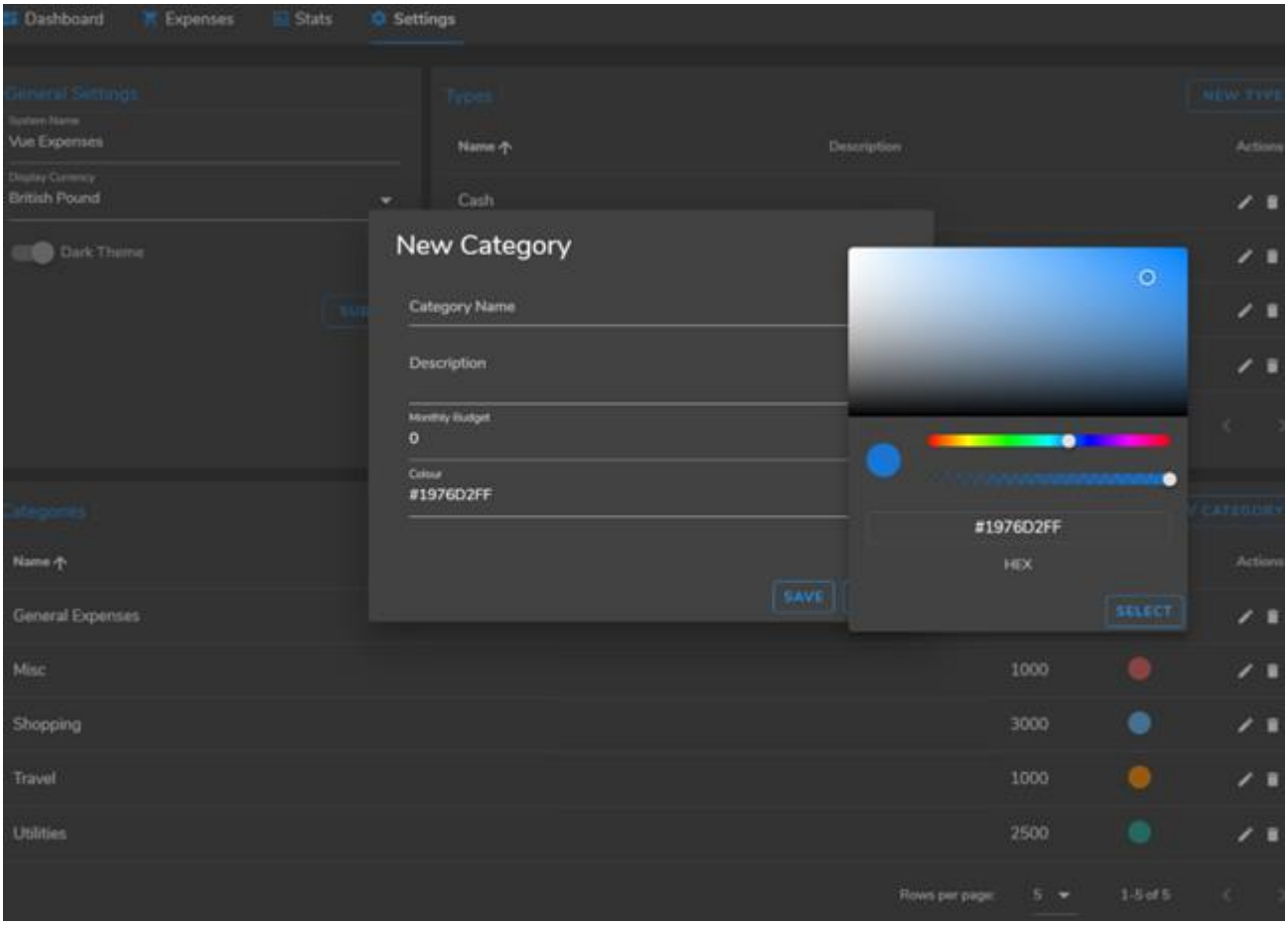

Рисунок Г.6 - Налаштування додатку

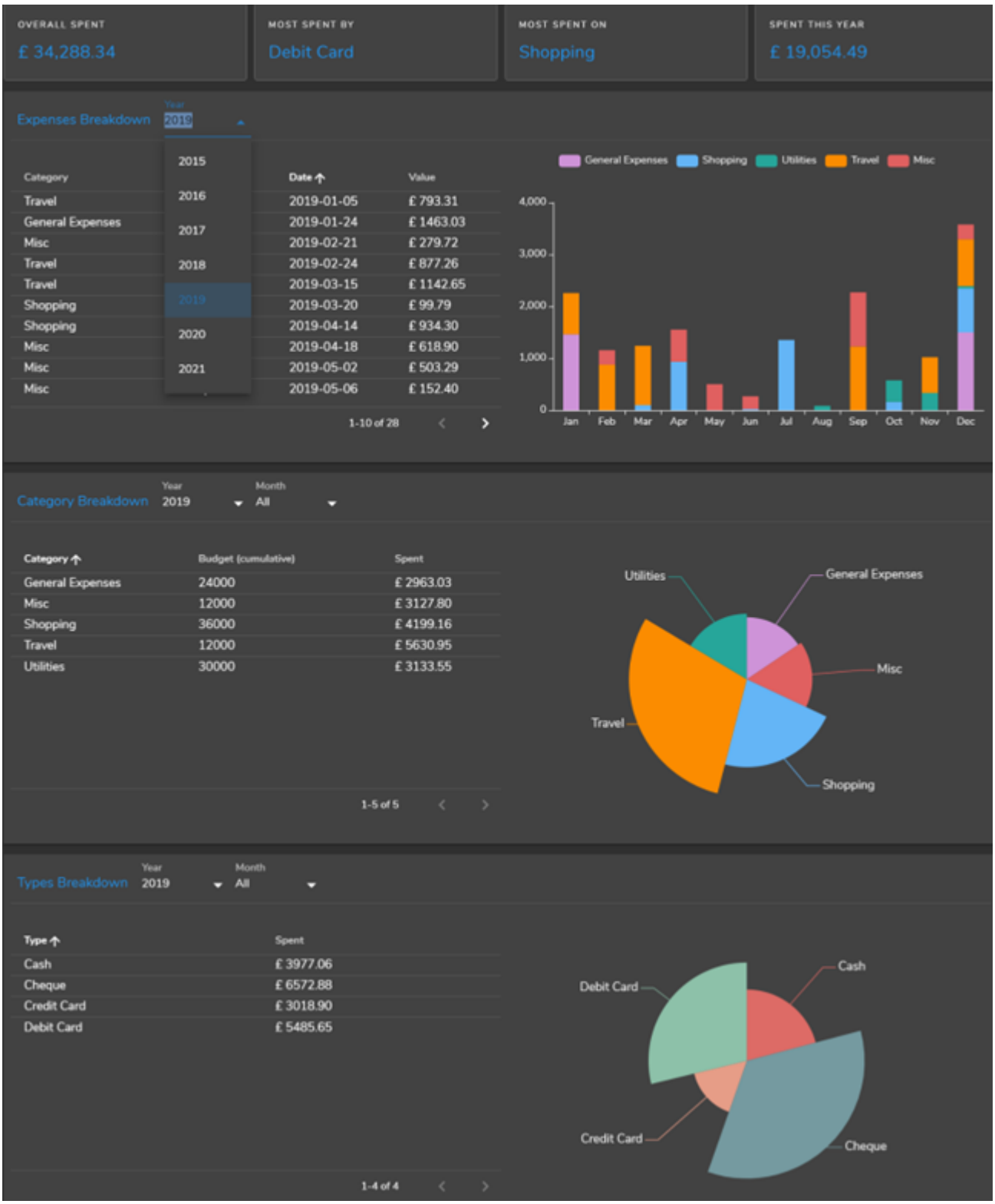

Рисунок Г.7 - Статистика витрат

Результати тестування розробленої програми

ID Тест-Кейсу	Назва Тест-Кейсу	Опис Тесту	Попередні Умови	Кроки	Очікуваний Результат	Статус
TC-001	Тестування додавання витрат	Перевірка функціональності додавання нової витрати	Відкритий додаток, авторизований користувач	1. Відкрити форму додавання витрат. 2. Ввести дані витрати. 3. Натиснути "Зберегти".	Витрата успішно додається і відображається у списку.	Пройдено
TC-002	Тестування редагування витрат	Перевірка функціональності редагування існуючої витрати	Існує хоча б одна витрата у списку	1. Обрати витрату зі списку. 2. Внести зміни. 3. Натиснути "Зберегти".	Зміни збережено, оновлена витрата відображається.	Пройдено
TC-003	Тестування видалення витрат	Перевірка функціональності видалення витрати	Існує хоча б одна витрата у списку	1. Обрати витрату зі списку. 2. Натиснути "Видалити".	Витрата успішно видаляється зі списку.	Пройдено
TC-004	Тестування генерації звітів	Перевірка коректності генерації фінансових звітів	Існують витрати у різних категоріях	1. Відкрити розділ "Звіти". 2. Обрати потрібний період. 3. Натиснути "Згенерувати".	Генерується звіт з актуальними даними по витратах.	Пройдено

Рисунок Г.8 – Результати тестування розробленої програми

ID Тест-Кейсу	Назва Тест-Кейсу	Опис Тесту	Попередні Умови	Кроки	Очікуваний Результат	Статус
TC-005	Тестування світлої теми	Перевірка інтерфейсу у світлій темі	Увімкнена світла тема	1.Переконатися, що тема обрана. 2. Перевірити відображення елементів інтерфейсу.	Всі елементи чітко видимі, немає помилок у відображенні.	Пройдено
TC-006	Тестування темної теми	Перевірка інтерфейсу у темній темі	Увімкнена темна тема	1. Переконатися, що тема обрана. 2. Перевірити відображення елементів інтерфейсу.	Всі елементи чітко видимі, немає помилок у відображенні.	Пройдено
TC-007	Тестування адаптивності інтерфейсу	Перевірка відображення додатку на різних роздільних здатностях	Підключено декілька моніторів з різними здатностями	1. Відкрити додаток на кожному з моніторів. 2. Перевірити відображення інтерфейсу.	Інтерфейс коректно відображається на всіх моніторах.	Пройдено
TC-008	Тестування продуктивності при навантаженні	Перевірка стабільності роботи додатку при великих обсягах даних	Є багато витрат в базі даних	1. Додати велику кількість витрат. 2. Виконати навантажувальні операції (генерація звітів).	Додаток стабільно працює без затримок і помилок.	Пройдено
TC-009	Тестування безпеки автентифікації	Перевірка роботи механізмів автентифікації та захисту паролів	Додаток налаштовано для роботи з акаунтами	1. Виконати вхід з різними обліковими записами. 2. Перевірити роботу збереження паролів.	Вхід успішний, паролі зберігаються і захищені належним чином.	Пройдено
TC-010	Тестування зміни теми	Перевірка коректності зміни теми інтерфейсу	Відкритий додаток	1. Змінити тему зі світлої на темну та навпаки.	Тема змінюється без помилок, всі елементи коректно підлаштовуються.	Пройдено

Рисунок Г.9 – Результати тестування розробленої програми

Додаток Г (довідниковий)

Інструкція користувача

Для запуску розробленої програми необхідно запустити клієнтську частину командною з консолі: npm run vue-cli-service serve, після чого буде запущено веб частину додатку.

Для запуску серверу варто використати команду dotnet run, запустити виконная серверного коду.

Після запуску програми відкрити в браузері веб додаток за url: http://localhost:3000. На сторінці авторизації ввести свої данні для входу, або ж розпочати процес реєстрації. Після закінчення реєстрації та верифікації електронної адреси, перейти в налаштування для додавання нової картки, для додавання карти Monobank потрібно пройти авторизацію відсканувавши QR код, в додатку Monobank. Для додавання карти приват банку, потрібно зайти в приват24 для бізнесу, зайти на вкладку Автоклієнт, та його API після чого вам буде надано токен, який вам потрібно скопіювати та ввести в додаток.

Після чого будуть автоматично синхронізовані усі ваші витрати з цієї карти які будуть зроблені в подальшому.

Щоб додати власні витрати потрібно на цій ж вкладці натиснути кнопку new expense і вказати данні. Також тут можна добавити також власну категорію витрат.

Щоб переглянути свою статистику, потрібно перейти на вкладку stats і обрати необхідний період, та критерії фільтрації, після чого буде виведено статистику та розбивку ваших витрат за обраний період.