МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ВЕБСАЙТ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ З ВІЗУАЛІЗАЦІЯМИ ТА ПОРАДАМИ ПО ЕКОНОМІЇ

Виконала: студентка групи 2П-22
Спеціальності
121 Інженерія програмного забезпечення
Катерина СКРИГАНЮК
Керівник:
Вікторія НЕМЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО
(підпис)
«_____» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Скриганюк Катерині Романівни

1. Тема кваліфікаційної роботи Вебсайт для управління особистими фінансами з візуалізаціями та порадами по економії

Керівник роботи Немченко Вікторія Юріївна, викладач другої категорії затверджені наказом закладу фахової передвищої освіти від «06» жовтня 2026 року №84/1-у

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи вимоги до структури та оформлення кваліфікаційної роботи практичного характеру; технології HTML5, CSS3, JavaScript; локальне сховище браузера (localStorage); необхідність забезпечення функціональних і нефункціональних вимог до вебсайту управління особистими фінансами

4. Зміст кваліфікаційної роботи аналіз існуючих цифрових рішень для обліку доходів і витрат; постановка задачі розроблення вебзастосунку; визначення функціональних і нефункціональних вимог; реалізація модуля обліку доходів і витрат, системи візуалізації фінансових даних та модуля автоматизованих порад щодо економії; тестування та аналіз результатів тестування.

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1. Огляд поточного стану предметної області	09.12.2025	
3	Розділ 2. Проєктування та реалізація програмного проєкту	10.03.2026	
4	Розділ 3. Тестування та супроводження	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студент _____ Катерина СКРИГАНЮК
(підпис)

Керівник роботи _____ Вікторія НЕМЧЕНКО
(підпис)

АНОТАЦІЯ

У кваліфікаційній роботі розроблено вебсайт для управління особистими фінансами з інтерактивними візуалізаціями та автоматизованими порадами щодо економії. Метою роботи є створення сучасного веб-інструменту, який забезпечує ефективний облік доходів і витрат, проведення фінансового аналізу та генерацію персоналізованих рекомендацій з оптимізації витрат.

У процесі дослідження проведено огляд теоретичних аспектів управління особистими фінансами, виконано порівняльний аналіз програмних аналогів, сформовано функціональні та нефункціональні вимоги до системи. Здійснено проєктування системи з використанням UML-діаграм та ER-діаграми даних. Програмна реалізація виконана за допомогою технологій HTML5, CSS3 та JavaScript з використанням локального зберігання даних у браузері (localStorage). Розроблений продукт містить модулі обліку транзакцій, інтерактивної візуалізації (графіки, діаграми), генерації фінансових звітів та інтелектуального модуля рекомендацій щодо економії.

Розроблений вебсайт характеризується високою продуктивністю, адаптивним інтерфейсом, повною автономністю роботи в офлайн-режимі та забезпеченням безпеки даних користувача. Програмне рішення може бути використане як самостійний сервіс або інтегроване в навчальний процес для підвищення фінансової грамотності.

Ключові слова: УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ, ВЕБСАЙТ, ФІНАНСОВИЙ ОБЛІК, ВІЗУАЛІЗАЦІЯ ДАНИХ, ПОРАДИ ЩОДО ЕКОНОМІЇ, JAVASCRIPT, БЮДЖЕТУВАННЯ, ФІНТЕХ, UML-МОДЕЛЮВАННЯ, LOCALSTORAGE

ABSTRACT

The qualification work is devoted to the development of a website for personal finance management with interactive visualizations and automated saving tips. The purpose of the work is to create a modern web tool that allows users to efficiently track income and expenses, perform financial analysis and receive personalized recommendations for expense optimization.

The study includes a review of theoretical aspects of personal finance management, a comparative analysis of software analogues, the formation of functional and non-functional system requirements. The system design was carried out using UML diagrams and an ER data diagram. The software implementation was performed using HTML5, CSS3 and JavaScript technologies with local browser storage (localStorage). The developed product includes transaction accounting modules, interactive visualization (charts, diagrams), financial report generation and an intelligent module of saving tips.

The developed website is characterized by high performance, adaptive interface, full offline functionality and user data security. The software solution can be used as a standalone service or integrated into the educational process to improve financial literacy.

Keywords: PERSONAL FINANCE MANAGEMENT, WEBSITE, FINANCIAL ACCOUNTING, DATA VISUALIZATION, SAVING TIPS, JAVASCRIPT, BUDGETING, FINTECH, UML MODELING, LOCALSTORAGE

ЗМІСТ

ВСТУП

У сучасних умовах економічної нестабільності, високої інфляції та швидкої цифровізації суспільства ефективне управління особистими фінансами стає однією з ключових компетенцій кожної людини. За даними Національного банку України та міжнародних досліджень, понад 60 % домогосподарств в Україні стикаються з проблемами планування бюджету, відсутністю візуального контролю за витратами та недостатньою обізнаністю щодо інструментів економії. Традиційні методи (паперові блокноти, таблиці Excel) не задовольняють сучасні потреби через відсутність автоматизації, мобільності та інтерактивної візуалізації.

Світовий ринок fintech-рішень для особистих фінансів демонструє стійке зростання: за прогнозами Української асоціації фінтех та інноваційних компаній, у 2024–2026 роках обсяг інвестицій у цей сегмент в Україні перевищить 300 млн дол. США. Водночас на ринку переважають іноземні сервіси (Mint, YNAB, PocketGuard), які не повністю адаптовані до українських реалій (курс гривні, особливості оподаткування, локальні платіжні системи). Існуючі українські аналоги (Monobank, Privat24, Excel-шаблони) або обмежені функціоналом, або не пропонують інтелектуальних порад щодо економії.

Розробка вебсайту для управління особистими фінансами з інтерактивними візуалізаціями та автоматизованими рекомендаціями щодо економії є актуальною науково-практичною задачею, яка поєднує потреби користувачів з можливостями сучасних веб-технологій. Рішення дозволить підвищити фінансову грамотність населення, оптимізувати витрати та сприяти формуванню заощаджень у складних економічних умовах.

Об’єктом дослідження є процес управління особистими фінансами фізичних осіб у цифровому середовищі.

Предметом дослідження є методи та засоби програмної інженерії для створення вебсайту управління особистими фінансами з візуалізаціями та порадами по економії.

Метою кваліфікаційної роботи є розробка вебсайту для управління особистими фінансами з інтерактивними візуалізаціями та автоматизованими порадами щодо економії на основі технологій програмної інженерії.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести огляд поточного стану предметної області, проаналізувати літературні джерела та існуючі програмні аналоги.
2. Здійснити аналіз функціональних і нефункціональних вимог до програмного забезпечення.
3. Виконати проєктування системи з використанням UML-діаграм.
4. Реалізувати програмний продукт за допомогою HTML5, CSS3 та JavaScript.
5. Розробити модулі візуалізації даних (графіки, діаграми) та алгоритм генерації персоналізованих порад щодо економії.
6. Провести комплексне тестування розробленого вебсайту та проаналізувати отримані результати.
7. Сформулювати висновки та рекомендації щодо подальшого використання та розвитку програмного рішення.

Розроблений вебсайт має практичне значення для широкого кола користувачів, може бути впроваджений як самостійний сервіс або інтегрований у існуючі фінансові платформи, а також використовуватися у навчальному процесі для підвищення фінансової грамотності студентів.

РОЗДІЛ 1 ОГЛЯД ПОТОЧНОГО СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Теоретичні аспекти управління особистими фінансами

Управління особистими фінансами є одним із ключових елементів фінансової поведінки сучасної людини та становить процес, у ході якого індивід досягає власних фінансових цілей шляхом визначення цілей, аналізу поточного стану, вибору оптимальних способів їх досягнення з урахуванням майбутніх тенденцій. Цей процес охоплює формування, розподіл і використання грошових ресурсів фізичної особи чи домогосподарства з метою забезпечення стабільності, зростання добробуту та мінімізації ризиків [3].

Основними компонентами управління особистими фінансами виступають доходи, витрати, заощадження, інвестиції та боргові зобов’язання. Доходи поділяються на трудові (заробітна плата, підприємницька діяльність) та нетрудові (відсотки, дивіденди, пенсії). Витрати класифікують за роллю в житті домогосподарств на споживчі, виробничі та інвестиційні, а також за характером – на постійні та змінні. Ефективне управління витратами передбачає їх оптимізацію через складання бюджету та контроль за структурою, що дозволяє уникнути дефіциту та забезпечити позитивний баланс [2].

Важливе місце в теорії управління особистими фінансами посідає поведінковий аспект. Поведінкові фінанси пояснюють, як психологічні фактори впливають на прийняття рішень, спричиняючи ірраціональні дії, такі як надмірна самовпевненість, схильність до уникнення втрат чи ефект статус-кво. Дослідження доводять, що такі чинники суттєво знижують ефективність фінансового планування, тому врахування поведінкових особливостей є необхідною передумовою розробки дієвих стратегій управління [1].

Фінансова грамотність виступає фундаментом успішного управління особистими фінансами. Вона включає сукупність знань, умінь і навичок, необхідних для прийняття обґрунтованих рішень щодо бюджету, інвестицій,

страхування та управління ризиками. Підвищення рівня фінансової грамотності населення дозволяє зменшити негативний вплив поведінкових помилок і сприяє формуванню раціональної фінансової поведінки. У сучасних умовах це стає стратегічним напрямом державної політики, оскільки безпосередньо впливає на економічну стабільність домогосподарств [6].

У міжнародній практиці управління особистими фінансами акцентується на встановленні чітких фінансових цілей, використанні сучасних інструментів планування та врахуванні принципів сталого розвитку. Теорії особистого фінансового планування підкреслюють необхідність системного підходу, який поєднує короткострокові потреби з довгостроковими перспективами, включаючи пенсійне забезпечення та накопичення капіталу. Такий підхід забезпечує не лише поточну стабільність, а й зростання добробуту протягом усього життєвого циклу людини [38].

Теоретичні засади управління особистими фінансами ґрунтуються на поєднанні класичних економічних принципів з поведінковими та психологічними факторами, що дозволяє формувати ефективні моделі розпорядження ресурсами в умовах цифровізації та економічної нестабільності.

1.2 Порівняльний аналіз програмних аналогів

Сучасний ринок програмного забезпечення для управління особистими фінансами представлений як міжнародними, так і вітчизняними рішеннями, які відрізняються за функціональністю, способом введення даних та рівнем автоматизації [4]. Порівняльний аналіз основних аналогів проведено за ключовими характеристиками, що безпосередньо впливають на вибір інструменту користувачем [7].

«WalletApp» – це сучасний вебзастосунок для управління особистими фінансами, який дозволяє користувачам вести облік доходів і витрат, аналізувати фінансову активність та отримувати рекомендації щодо оптимізації бюджету,

інтерфейс якого поданий на рис. 1.1. Система надає можливість додавати фінансові операції, розподіляти їх за категоріями, переглядати статистику у вигляді графіків і діаграм, а також відстежувати динаміку змін фінансового стану. Основною перевагою застосунку є простий та зрозумілий інтерфейс, що забезпечує швидкий доступ до необхідної інформації та сприяє ефективному контролю особистих фінансів.

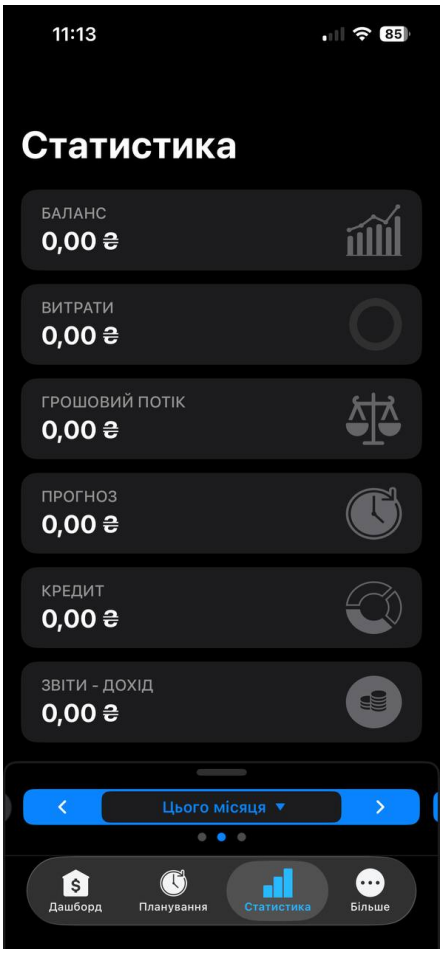

Рисунок 1.1 – вебзастосунок WalletApp

«Monefy» популярний мобільний застосунок для обліку особистих фінансів, призначений для швидкого внесення інформації про доходи та витрати. Головною особливістю сервісу є простота використання та мінімальна кількість дій для додавання нової фінансової операції. Користувачі можуть створювати власні категорії витрат, переглядати статистику за допомогою кругових діаграм і контролювати структуру своїх витрат. До переваг Monefy належать інтуїтивний

інтерфейс та висока швидкість роботи, проте застосунок має обмежені можливості щодо детального фінансового аналізу та автоматизованих рекомендацій. Інтерфейс мобільного застосунку Monefy представлений на рис. 1.2.

Рисунок 1.2 – мобільний застосунок Monefy

«Spendee» – це багатофункціональний сервіс для управління особистими фінансами, який дозволяє вести облік доходів і витрат, планувати бюджет та аналізувати фінансову активність користувача. Однією з ключових особливостей застосунку є підтримка синхронізації банківських рахунків, що дає змогу автоматично отримувати інформацію про фінансові операції. Spendee пропонує широкий набір інструментів для візуалізації даних, включаючи графіки, звіти та аналітичні панелі. Серед переваг сервісу можна виділити розширені можливості аналізу та сучасний дизайн, однак частина функціоналу доступна лише в платній

версії, що може бути недоліком для окремих користувачів. Інтерфейс мобільного застосунку Monefy представлений на рис. 1.3.

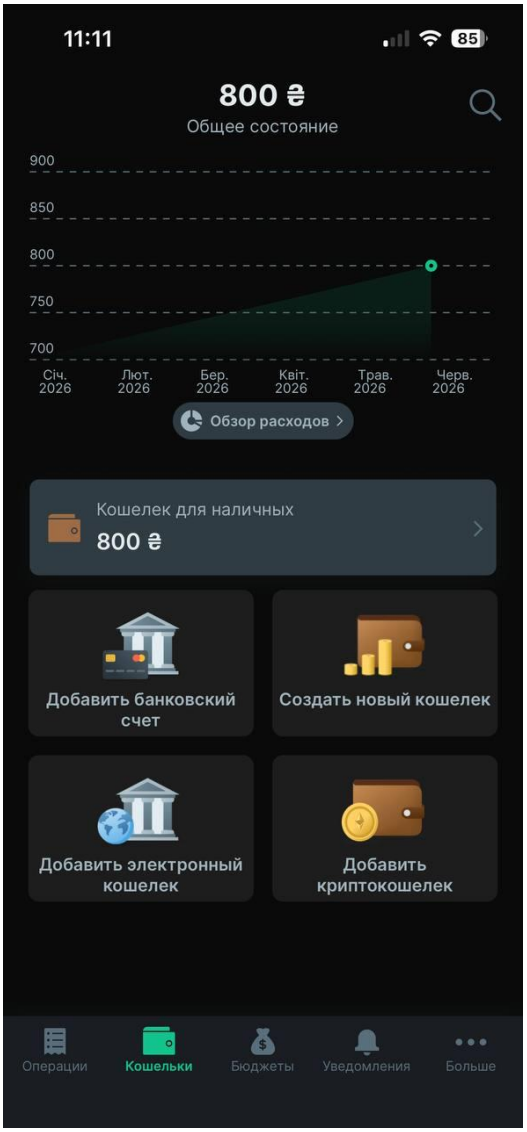

Рисунок 1.3 – мобільний застосунок Spendee

Розглянуті аналоги демонструють різні підходи до реалізації основних функцій обліку, візуалізації та рекомендацій, що дозволяє визначити оптимальні технічні рішення для розробки нового вебсайту.

Таблиця 1.1 – Порівняльна характеристика програмних аналогів систем управління особистими фінансами

Назва аналога	Облік доходів і витрат	Інтерактивні візуалізації (графіки, діаграми)	Автоматизовані поради щодо економії	Платформа	Тип зберігання даних	Доступність в Україні
Money Manager	Ручний ввід доходів та витрат	Так (графіки, кругові діаграми)	Ні	Android, iOS	Локальне + хмарне (резервне копіювання)	Повна
Wallet	Автоматичний та ручний облік операцій	Так (детальні звіти та графіки)	Так (аналіз витрат та бюджетні цілі)	Веб + Android + iOS	Хмарне	Повна
Spendee	Автоматичний та ручний облік	Так (категорії, графіки, діаграми)	Так (контроль бюджету)	Веб + Android + iOS	Хмарне	Повна
Monefy	Ручний облік доходів та витрат	Так (кругові діаграми та статистика)	Ні	Android, iOS	Локальне + Google Drive	Повна
CoinKeeper	Ручний та автоматичний облік	Так (наочні графіки та звіти)	Так (нагадування та бюджетні обмеження)	Android, iOS	Хмарне	Повна
Goodbudget	Ручний облік за методом конвертів	Так (звіти за категоріями)	Так (контроль лімітів бюджету)	Веб + Android + iOS	Хмарне	Повна
YNAB (You Need A Budget)	Ручний та імпортний ввід	Так (звітність за категоріями)	Так (методика планування бюджету)	Веб + Android + iOS	Хмарне	Доступний
Fintonic	Автоматичний облік транзакцій	Так (графіки та аналітика)	Так (персоналізовані рекомендації)	Android, iOS	Хмарне	Обмежена

Джерела інформації: офіційні сайти сервісів та аналітичні огляди ринку [22], [23], [29].

1.3 Переваги та недоліки існуючих рішень

Аналіз програмних аналогів свідчить, що кожне рішення має сильні та слабкі сторони, які безпосередньо впливають на його практичне застосування користувачами в Україні [8]. Mint вирізняється потужною автоматизацією імпорту транзакцій і детальними візуалізаціями, проте суттєвим недоліком є відсутність повної локалізації під українські реалії та обмежена підтримка гривневих рахунків [23]. YNAB забезпечує глибоке бюджетування та дисципліну витрат, однак потребує обов'язкового ручного введення даних і має високу вартість підписки [9].

Українські сервіси Monobank та Privat24 пропонують безшовну інтеграцію з банківськими рахунками та зручний мобільний доступ, але не містять модуля інтелектуальних порад щодо економії та обмежені лише операціями конкретного банку [24]. Google Sheets як універсальний інструмент дозволяє повну кастомізацію графіків, проте вимагає ручного заповнення та не забезпечує автоматичного аналізу [25]. PocketGuard ефективно виявляє зайві підписки, однак має обмежену функціональність поза США та слабку інтеграцію з українськими банками [29].

В табл. 1.2. подані переваги та недоліки програмних аналогів систем управління особистими фінансами.

Жоден із розглянутих аналогів не поєднує повною мірою автоматичний облік, інтерактивні візуалізації, персоналізовані поради щодо економії та повну адаптацію до українських умов, що підтверджує необхідність розробки нового спеціалізованого вебсайту.

Таблиця 1.2 – Переваги та недоліки програмних аналогів систем управління особистими фінансами

Назва аналога	Переваги	Недоліки
Money Manager	Простий та зрозумілий інтерфейс, підтримка кількох рахунків, детальні графіки та звіти	Відсутня автоматична синхронізація з більшістю українських банків, частина функцій доступна лише у платній версії
Wallet	Автоматичний імпорт транзакцій, гнучке планування бюджету, детальна аналітика витрат	Деякі функції потребують преміум-підписки, не всі українські банки підтримуються
Spendee	Сучасний дизайн, спільні гаманці для сім'ї, наочні діаграми та статистика	Обмежений функціонал у безкоштовній версії, часткова підтримка банківської інтеграції
Monefy	Простота використання, швидке внесення витрат, наочні кругові діаграми	Відсутність автоматичного імпорту банківських операцій, обмежені можливості аналізу бюджету
CoinKeeper	Зручний візуальний контроль фінансів, нагадування про платежі, бюджетні ліміти	Багато можливостей доступні лише за підпискою, складніший інтерфейс для новачків
Goodbudget	Ефективне планування бюджету за методом «конвертів», синхронізація між пристроями, безкоштовна базова версія	Повністю ручне введення даних, відсутня автоматична синхронізація з банками

1.4 Постановка задачі та обґрунтування розробки

На основі проведеного огляду теоретичних аспектів управління особистими фінансами та аналізу існуючих програмних аналогів постає необхідність формулювання конкретної задачі щодо створення нового вебсайту, який би усунув виявлені недоліки та врахував сучасні тенденції fintech-ринку в Україні [5]. Постановка задачі полягає в розробці вебсайту, що забезпечує комплексне управління особистими фінансами з акцентом на візуалізацію даних та генерацію рекомендацій щодо економії, з метою підвищення фінансової грамотності користувачів і оптимізації їхнього бюджету в умовах економічної нестабільності [10].

Обґрунтування розробки ґрунтується на тому, що існуючі рішення, такі як Monobank та Privat24, не надають повноцінних інструментів для персоналізованих порад, тоді як міжнародні аналоги (Mint, YNAB) недостатньо адаптовані до українських реалій, включаючи локальні платіжні системи та валютні курси [11]. Згідно з аналізом, новий вебсайт повинен включати модулі ручного та автоматичного вводу даних, інтерактивні графіки для візуалізації доходів/витрат, а також алгоритми на базі JavaScript для розрахунку рекомендацій, що дозволить користувачам ефективніше планувати фінанси без прив'язки до конкретних банків [12].

Такий підхід обґрунтований потребою в доступному інструменті, який поєднує простоту використання з потужними можливостями моделювання, зокрема через застосування UML-діаграм для проєктування структури системи, що забезпечить масштабованість і легкість супроводження [14]. Розробка вебсайту дозволить заповнити прогалину на ринку, де бракує безкоштовних або низьковитратних рішень з фокусом на візуалізацію та поради, сприяючи загальному підвищенню ефективності особистого фінансового менеджменту в Україні.

Постановка задачі передбачає створення програмного продукту, який інтегрує облік, аналіз і рекомендації в єдиному веб-інтерфейсі, з обґрунтуванням

на основі виявлених переваг і недоліків аналогів, що робить його актуальним для сучасних користувачів.

Отже, проведений аналіз існуючих програмних засобів для управління особистими фінансами показав, що сучасні рішення пропонують широкий спектр можливостей для обліку доходів і витрат, планування бюджету та візуалізації фінансових даних. Такі сервіси, як Wallet, Spendee, Monefy та інші, забезпечують зручний контроль особистих фінансів, проте кожен із них має певні обмеження. Частина застосунків характеризується недостатнім рівнем автоматизації, інші мають обмежений функціонал у безкоштовних версіях або не повністю адаптовані до потреб українських користувачів.

Результати аналізу свідчать про відсутність універсального рішення, яке б одночасно поєднувало зручний облік фінансових операцій, розширені можливості візуалізації, автоматизовані рекомендації щодо економії та доступність для широкого кола користувачів.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОЄКТУ

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Функціональні вимоги

Аналіз функціональних вимог є ключовим етапом проєктування програмного забезпечення, оскільки вони чітко визначають, які саме дії повинен виконувати вебсайт для досягнення поставленої мети [15]. Функціональні вимоги описують зовнішню поведінку системи з точки зору користувача, забезпечуючи повне покриття процесів управління особистими фінансами – від введення даних до генерації аналітичних звітів і рекомендацій. У разі розробки вебсайту на базі JavaScript з використанням локального зберігання даних (localStorage) вимоги спрямовані на створення автономного, швидкого та зручного інструменту, що не залежить від серверної інфраструктури [14].

Основні функціональні вимоги згруповані за модулями. Модуль обліку транзакцій передбачає можливість додавання, редагування та видалення записів про доходи й витрати з обов’язковим зазначенням дати, суми, категорії та примітки. Модуль візуалізації даних забезпечує автоматичне побудування інтерактивних графіків і діаграм на основі введеної інформації. Модуль рекомендацій щодо економії реалізує алгоритм аналізу структури витрат і генерацію персоналізованих порад. Крім того, система повинна підтримувати фільтрацію даних за періодами, категоріями та сумами, а також експорт звітів у зручних форматах [12].

Функціональні вимоги враховують вимоги до зручності інтерфейсу: інтуїтивна навігація, швидке завантаження сторінок та можливість роботи в офлайн-режимі. Окремо виділено вимоги до безпеки даних: захист інформації користувача шляхом локального зберігання без передачі на зовнішні сервери.

Загалом функціональні вимоги забезпечують реалізацію повного циклу управління фінансами в одному веб-додатку [11].

Таблиця 2.1 – Функціональні вимоги до вебсайту управління особистими фінансами

№	Назва вимоги	Опис вимоги
1	Реєстрація та авторизація користувача	Можливість створення локального профілю та входу за допомогою логіна/пароля з використанням localStorage
2	Додавання доходів	Введення суми, дати, категорії та примітки для доходів з автоматичним оновленням балансу
3	Додавання витрат	Введення суми, дати, категорії та примітки для витрат з автоматичним оновленням балансу
4	Управління категоріями	Створення, редагування та видалення категорій доходів і витрат
5	Перегляд історії транзакцій	Відображення списку всіх операцій з можливістю сортування та фільтрації
6	Генерація інтерактивних візуалізацій	Автоматичне побудування кругових, стовпчастих та лінійних діаграм доходів/витрат
7	Генерація порад щодо економії	Аналіз структури витрат і автоматична генерація персоналізованих рекомендацій
8	Фільтрація та пошук даних	Фільтрація транзакцій за періодом, категорією, сумою та ключовими словами
9	Експорт даних	Експорт звітів у формати CSV та PDF
10	Налаштування бюджету	Встановлення планових лімітів за категоріями з візуальним контролем виконання
11	Збереження даних у локальному сховищі	Автоматичне збереження всіх даних у браузері без передачі на сервер
12	Генерація фінансових звітів	Формування місячних/річних звітів з підсумками та порівнянням періодів

Сформовані функціональні вимоги повністю охоплюють потреби користувачів у комплексному управлінні фінансами та є основою для подальшого проєктування архітектури системи та її програмної реалізації.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики програмного забезпечення, які не пов’язані безпосередньо з його функціональністю, але суттєво впливають на користувацький досвід, надійність та ефективність експлуатації [16]. Для вебсайту управління особистими фінансами, реалізованого виключно на клієнтській стороні за допомогою HTML5, CSS3 та JavaScript, нефункціональні вимоги спрямовані на забезпечення швидкої роботи, високого рівня безпеки даних та зручності використання без залежності від серверної інфраструктури [14].

Особлива увага приділяється продуктивності (швидке завантаження та обробка даних), адаптивності інтерфейсу для різних пристроїв, захисту інформації користувача та стабільній роботі в офлайн-режимі. Система повинна відповідати сучасним стандартам доступності та забезпечувати інтуїтивну взаємодію навіть для користувачів з мінімальним рівнем технічної підготовки [18].

Таблиця 2.2 – Нефункціональні вимоги до вебсайту управління особистими фінансами

№	Назва вимоги	Опис вимоги
1	2	3
1	Продуктивність	Час завантаження сторінки ≤ 2 секунди; обробка до 10 000 транзакцій без затримок
2	Зручність використання (Usability)	Інтуїтивний інтерфейс, повна адаптивність (responsive design), підтримка темної теми

Продовження таблиці 2.2

1	2	3
3	Безпека даних	Локальне зберігання в browser storage без передачі на сервери; захист від XSS-атак
4	Надійність	Автоматичне резервне копіювання даних; відновлення після закриття браузера
5	Сумісність	Підтримка сучасних браузерів (Chrome, Firefox, Edge, Safari версії 2023+)
6	Масштабованість	Можливість роботи з великими обсягами даних без зниження швидкості
7	Доступність	Відповідність стандартам WCAG 2.1 (контрастність, навігація клавіатурою)
8	Підтримка офлайн-режиму	Повна функціональність без підключення до інтернету

Сформовані нефункціональні вимоги забезпечують високу якість, безпеку та зручність розробленого вебсайту, що є обов’язковою умовою для його практичного використання кінцевими користувачами [19].

2.2 Проєктування системи

2.2.1 Діаграма варіантів використання

Діаграма варіантів використання є фундаментальним етапом проєктування програмного забезпечення, оскільки вона наочно відображає взаємодію зовнішніх акторів із системою та чітко визначає її функціональні межі відповідно до сформованих вимог [15]. У контексті вебсайту управління особистими фінансами така діаграма дозволяє систематизувати всі основні сценарії роботи користувача, починаючи від авторизації та закінчуючи генерацією аналітичних звітів і персоналізованих рекомендацій. Вона забезпечує повне покриття

функціональних вимог і слугує основою для подальшого детального моделювання класів та послідовності дій (рис. 2.1).

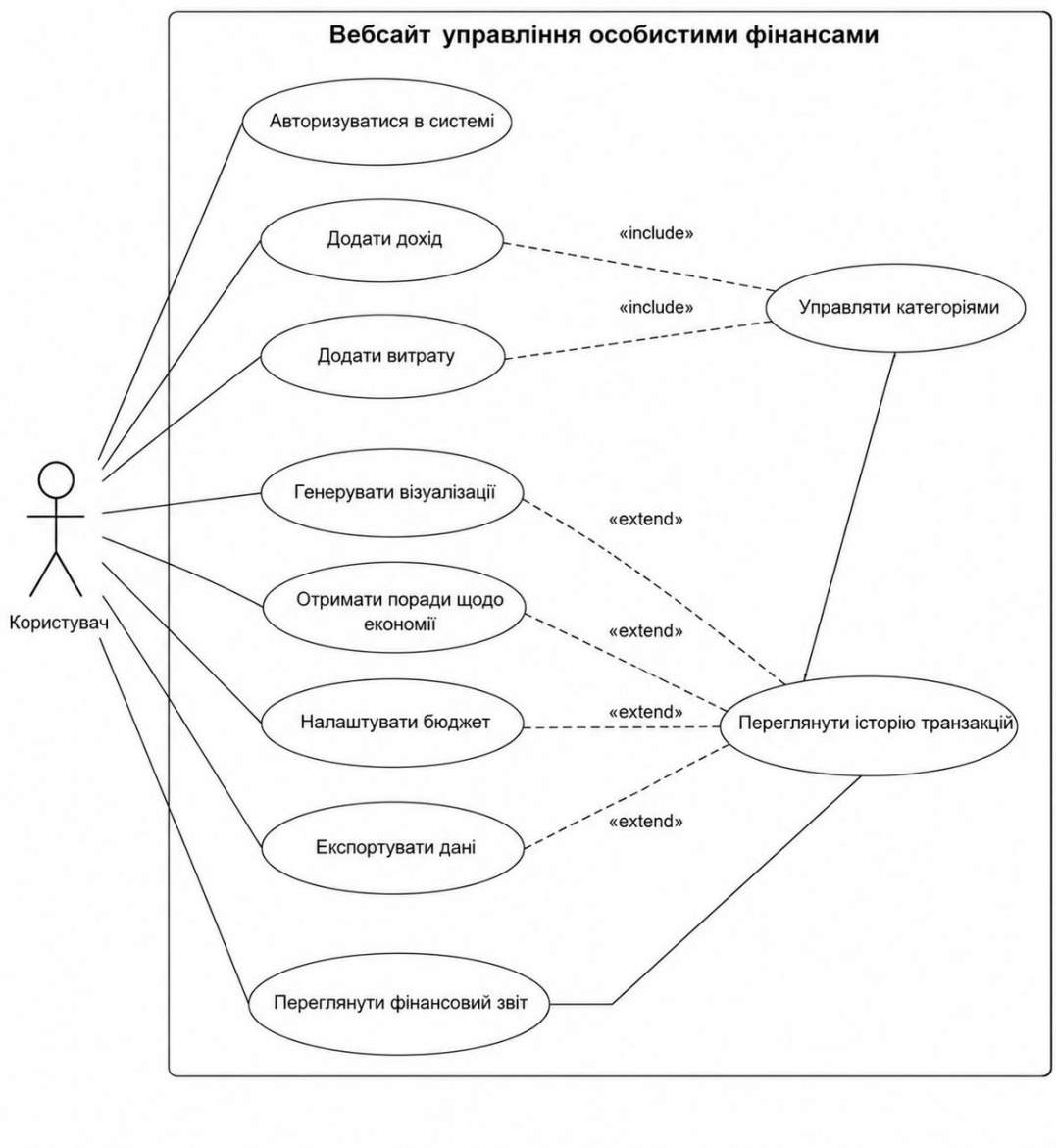

Рисунок 2.1 – Діаграма варіантів використання

Єдиним актором системи виступає Користувач, який взаємодіє з усіма варіантами використання. Основні сценарії включають авторизацію в системі, додавання записів про доходи та витрати, перегляд історії транзакцій, генерацію інтерактивних візуалізацій, отримання порад щодо економії, налаштування бюджету, управління категоріями та експорт даних. Деякі сценарії пов’язані відношеннями включення та розширення: додавання доходів і витрат

обов’язково включає роботу з категоріями, а візуалізації, поради та звіти розширюють базовий сценарій перегляду історії транзакцій. Така структура забезпечує логічну цілісність системи та полегшує подальшу реалізацію.

Наведена діаграма повністю відповідає функціональним вимогам, сформованим у підрозділі 2.1.1, і демонструє, що система орієнтована на одного користувача з широким спектром самостійних операцій. Відсутність зовнішніх акторів (наприклад, банківських API) пояснюється вибором локального зберігання даних у браузері, що забезпечує автономність і безпеку. Такий підхід спрощує архітектуру та робить вебсайт доступним для широкого кола користувачів без додаткових інтеграцій.

Діаграма варіантів використання слугує чіткою основою для переходу до моделювання статичної структури системи (діаграми класів) та динамічної поведінки, забезпечуючи повну відповідність між вимогами та проєктними рішеннями.

2.2.2 Діаграма класів

Діаграма класів є одним із найважливіших артефактів проєктування об’єктно-орієнтованої системи, оскільки вона описує статичну структуру програмного забезпечення, відображаючи основні класи, їх атрибути, методи та взаємозв’язки між ними [15]. У розробленому вебсайті діаграма класів побудована з урахуванням функціональних і нефункціональних вимог і відображає ключові сутності предметної області – транзакції, категорії, бюджети та механізми аналізу даних. Вона слугує основою для подальшої програмної реалізації на JavaScript (рис. 2.2).

Основними класами системи є абстрактний клас Transaction, від якого успадковуються класи Income та Expense. Клас Category використовується для класифікації операцій. Клас User агрегує всі транзакції та бюджети користувача. Для формування звітів та візуалізацій передбачені класи FinancialReport та Visualization. Логіка генерації рекомендацій щодо економії винесена в окремий

клас RecommendationEngine. Така структура забезпечує високу модульність, зручність розширення та підтримку принципів об’єктно-орієнтованого програмування.

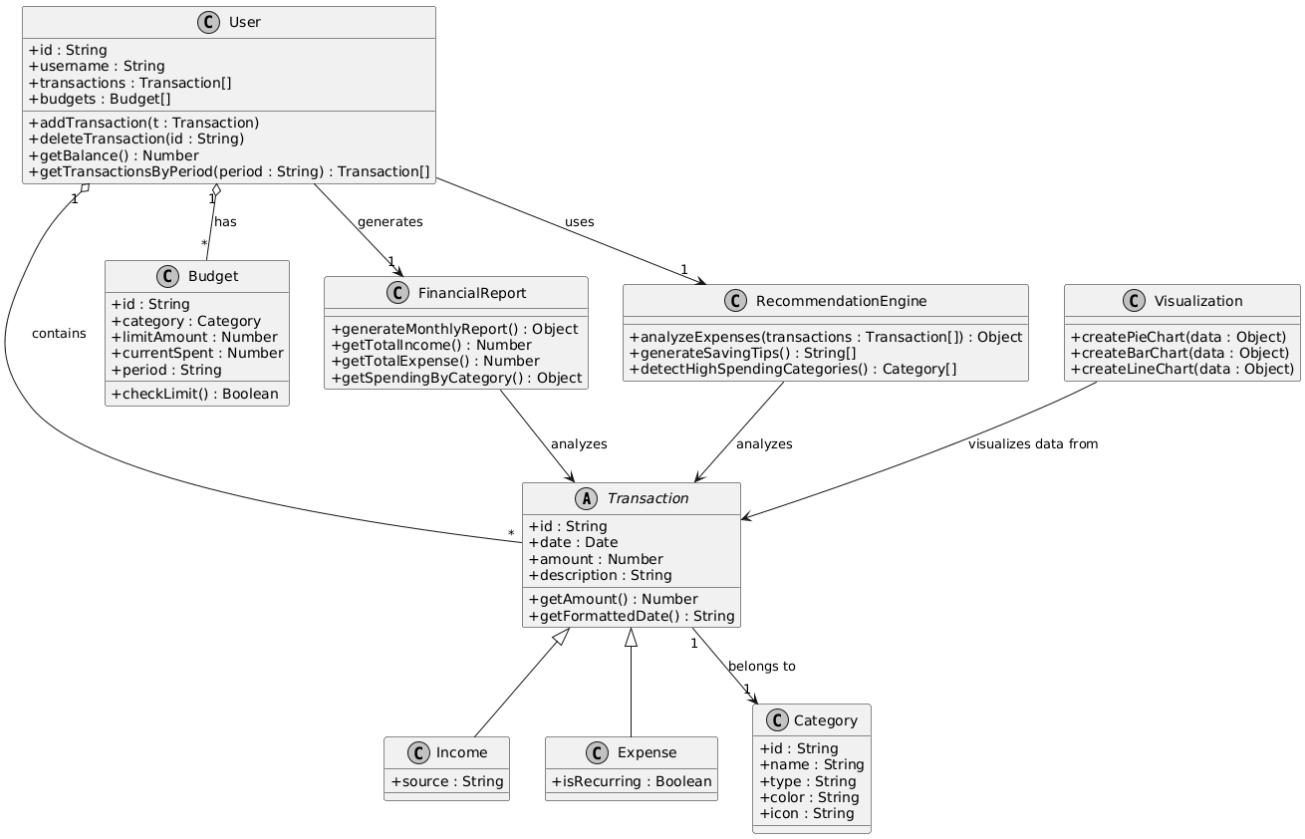

Рисунок 2.2 – Діаграма класів

Наведена діаграма класів демонструє чітку ієрархію та взаємодію між сутностями системи. Використання успадкування для класів Income та Expense дозволяє уникнути дублювання коду, а агрегація в класі User забезпечує централізоване управління даними. Така архітектура повністю відповідає принципам SOLID і є зручною для реалізації на мові JavaScript.

Діаграма класів визначає основну структуру майбутньої програмної реалізації та служить надійною основою для розробки модулів обліку, аналізу та візуалізації даних.

Діаграма класів не лише відображає програмну структуру системи, а й формує основу логічної моделі бази даних. Кожен клас предметної області може

бути представлений окремою сутністю в базі даних, що забезпечує впорядковане зберігання інформації та ефективний доступ до неї. Центральне місце в моделі займає клас Transaction, який містить відомості про фінансові операції користувача та пов’язаний із класом Category для визначення типу операції. Такий підхід дозволяє групувати дані за категоріями та виконувати подальший аналіз витрат і доходів.

Особливістю побудованої моделі є чітке розмежування даних та функціональних компонентів системи. Класи User, Transaction, Category та Budget відповідають за зберігання інформації, тоді як FinancialReport, RecommendationEngine і Visualization реалізують її обробку, аналіз та відображення. Це дозволяє зменшити залежність між окремими модулями системи та підвищити зручність її супроводу і модернізації.

2.2.3. Діаграма послідовності

Діаграма послідовності є важливим артефактом динамічного моделювання, який відображає порядок взаємодії об’єктів системи в часі при виконанні конкретного сценарію використання [15]. Вона дозволяє візуалізувати потік повідомлень між акторами та компонентами, забезпечуючи чітке розуміння логіки роботи програми та відповідність функціональним вимогам. У розробленому вебсайті така діаграма особливо корисна для демонстрації ключових процесів, пов’язаних із обробкою даних у реальному часі без серверної взаємодії (рис 2.3).

Для ілюстрації обрано базовий сценарій «Додавання витрати з автоматичним оновленням візуалізацій та генерацією рекомендацій». У цьому сценарії користувач вводить дані про витрату через інтерфейс, система зберігає інформацію в локальному сховищі, оновлює всі графіки та миттєво генерує персоналізовані поради щодо економії. Такий потік охоплює взаємодію основних модулів, визначених у діаграмі класів, і демонструє асинхронність оновлення інтерфейсу.

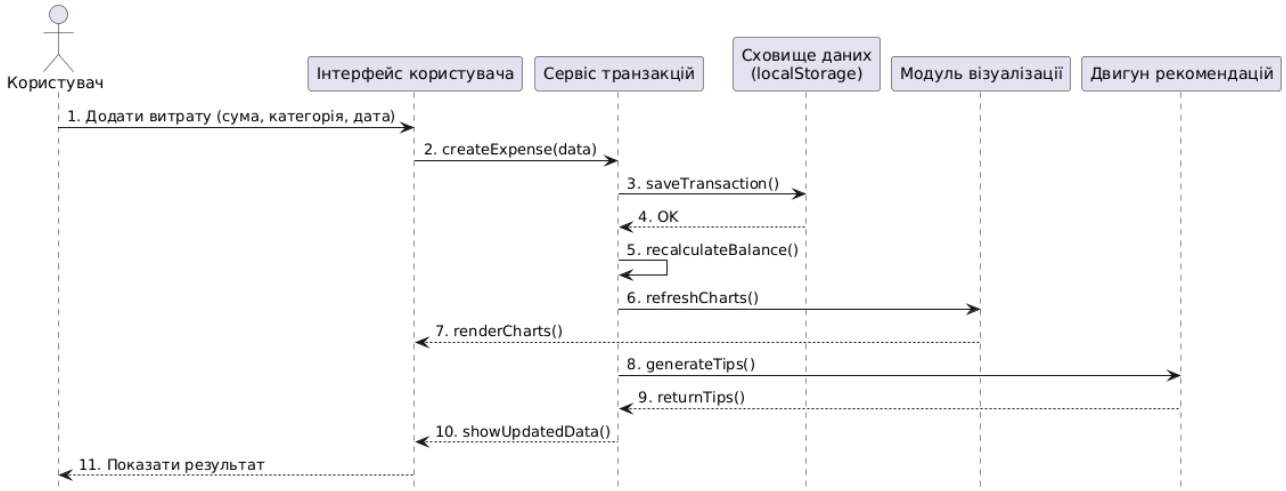

Рисунок 2.3 – Діаграма послідовності додавання витрати з оновленням візуалізацій та генерацією рекомендацій

Наведена діаграма чітко показує послідовність викликів: після збереження транзакції система паралельно оновлює візуалізації та запускає двигун рекомендацій. Кожне повідомлення пронумероване, що полегшує розуміння хронології подій. Відсутність зовнішніх сервісів підкреслює автономність додатка та високу швидкість виконання операцій безпосередньо в браузері.

Діаграма послідовності підтверджує коректність проєктних рішень, забезпечує прослідковуваність виконання функціональних вимог і є безпосередньою основою для програмної реалізації сценаріїв у модулях JavaScript.

2.2.4. Структура даних системи у JSON-форматі

Схема структури даних у JSON-форматі є основою організації інформації у вебзастосунку та відображає логічну модель зберігання даних у локальному сховищі браузера (localStorage). На відміну від традиційної ER-діаграми реляційної бази даних, дана модель орієнтована на використання JSON-об’єктів, які забезпечують просте та ефективне зберігання інформації на стороні клієнта.

Основним об’єктом системи є користувач, який містить усі пов’язані з ним дані у вигляді вкладених масивів і об’єктів. До складу структури входять

категорії, транзакції та бюджети. Кожна категорія характеризується назвою, типом (дохід або витрата) та кольором для візуального відображення в інтерфейсі. Транзакції містять інформацію про дату проведення операції, суму, опис, тип операції та посилання на відповідну категорію. Об’єкти бюджету зберігають дані про встановлений ліміт витрат, поточну суму використаних коштів, період дії бюджету та категорію, для якої цей бюджет визначений.

Використання вкладених структур дозволяє об’єднати взаємопов’язані дані в межах одного JSON-документа та мінімізувати кількість операцій пошуку. Зв’язки між окремими елементами реалізуються за допомогою унікальних ідентифікаторів, що забезпечує цілісність даних та спрощує їхню обробку засобами JavaScript.

Зберігання даних у форматі JSON забезпечує швидкий доступ до інформації, спрощує процес серіалізації та десеріалізації об’єктів, а також дозволяє ефективно реалізувати механізми обліку транзакцій, контролю бюджетів, побудови статистики та генерації персоналізованих рекомендацій.

Таким чином, структура даних у JSON-форматі забезпечує логічну організацію інформації, високу продуктивність роботи застосунку та є надійною основою для реалізації всіх функціональних можливостей системи без використання зовнішньої системи керування базами даних.

2.3 Програмна реалізація проєкту

2.3.1 Технологічний стек

Програмна реалізація вебсайту для управління особистими фінансами виконана повністю на клієнтській стороні з використанням сучасного фронтенд-стеку. Вибір технологій здійснювався з урахуванням вимог до високої продуктивності, адаптивності інтерфейсу, інтерактивної візуалізації даних, автономної роботи в офлайн-режимі та можливості генерації PDF-звітів.

Основний технологічний стек включає:

- React (версія 18+) - основна бібліотека для створення компонентного інтерфейсу користувача. Забезпечує швидке оновлення сторінки завдяки віртуальному DOM та зручну організацію коду.
- Vite - сучасне інструментальне середовище для збірки проєкту. Використовується завдяки швидкому запуску розробки, миттєвому гарячому перезавантаженню модулів (HMR) та оптимізованій продакшн-збірці.
- TypeScript - надмножина JavaScript, яка додає статичну типізацію. Дозволяє виявляти помилки на етапі компіляції та значно підвищує якість і підтримуваність коду.
- Bootstrap 5 - CSS-фреймворк для швидкого створення адаптивного дизайну. Доповнений власними CSS-стилями для реалізації ефекту скла (glassmorphism) та сучасного вигляду.
- Recharts - бібліотека для побудови інтерактивних графіків і діаграм (кругові, стовпчасті). Забезпечує високу якість візуалізації фінансових даних.
- Lucide React - набір сучасних SVG-іконок, що використовуються для оформлення елементів інтерфейсу.
- jsPDF + html2canvas - інструменти для генерації PDF-звітів безпосередньо в браузері (експорт усієї аналітики одним кліком).
- localStorage - браузерне сховище для збереження всіх даних користувача. Забезпечує повну автономність роботи додатка без використання серверної частини.

Вибраний технологічний стек дозволив створити швидкий, зручний і повністю автономний веб-додаток, який не потребує розгортання backend-сервера, працює в офлайн-режимі та відповідає всім функціональним і нефункціональним вимогам, сформованим у підрозділі 2.1.

2.3.2 Реалізація модуля обліку доходів і витрат

Модуль обліку доходів і витрат є центральним функціональним блоком розробленого вебсайту. Він забезпечує повний цикл роботи з фінансовими

операціями: введення, зберігання, перегляд, редагування та видалення записів про доходи та витрати з автоматичним оновленням загального балансу.

Основною одиницею даних є інтерфейс Transaction, який містить такі поля:

- id - унікальний ідентифікатор операції;
- type - тип операції (income або expense);
- amount - сума;
- category - категорія;
- date - дата проведення;
- description - необов'язковий опис.

Введення нових операцій реалізовано через компонент TransactionForm.tsx, який відкривається як модальне вікно Bootstrap після натискання кнопки «Додати операцію». У компоненті використовується стан useState для перемикання між доходами та витратами. Залежно від обраного типу динамічно формується список категорій (наприклад, «Зарплата», «Фріланс» для доходів та «Їжа», «Транспорт» для витрат).

Форма містить обов'язкові поля: сума, категорія та дата. Після заповнення даних функція handleSubmit виконує валідацію, створює новий об'єкт типу Transaction з унікальним ідентифікатором (на основі Date.now().toString(36)) і передає його через пропс onAdd до батьківського компонента.

Логіка зберігання даних реалізована в компоненті App.tsx. Тут оголошено стан transactions, а також функції:

- addTransaction(newTransaction) - додає запис до масиву та оновлює localStorage;
- deleteTransaction(id) - видаляє операцію за ідентифікатором.

Збереження даних здійснюється виключно через браузерне сховище localStorage. При завантаженні сторінки дані автоматично завантажуються, а при кожній зміні масиву transactions - зберігаються. Такий підхід забезпечує повну автономність додатка та швидкість роботи без звернення до сервера.

У компоненті Dashboard.tsx останні п'ять операцій відображаються у вигляді списку з можливістю швидкого видалення. Кожна транзакція показує

категорію, дату, суму та тип операції з кольоровим маркуванням (зелений для доходів, червоний для витрат).

Модуль обліку доходів і витрат забезпечує надійне, швидке та зручне введення фінансових даних, що є основою для подальшої роботи модулів візуалізації та генерації персоналізованих порад щодо економії.

2.3.3 Реалізація модуля візуалізації

Модуль візуалізації призначений для наочного представлення фінансових даних користувача у вигляді інтерактивних графіків і діаграм. Він забезпечує миттєве оновлення інформації після кожної доданої або видаленої операції та дозволяє швидко оцінити структуру витрат і динаміку доходів/витрат.

Візуалізація реалізована у окремому компоненті Charts.tsx з використанням бібліотеки Recharts. Ця бібліотека обрана завдяки високій продуктивності, підтримці SVG, інтерактивності та можливості адаптивного відображення на будь-яких пристроях.

Модуль містить два основних типи діаграм:

1. Кругова діаграма (PieChart) Відображає структуру витрат за категоріями. Дані формуються динамічно за допомогою методу reduce:

```
const pieData = transactions
  .filter(t => t.type === 'expense')
  .reduce((acc, t) => {
    acc[t.category] = (acc[t.category] || 0) + t.amount;
    return acc;
  }, {} as Record<string, number>);
```

Отриманий масив перетворюється на формат { name, value } і передається у компонент Pie. Кожна секція розфарбована унікальним кольором з масиву COLORS. Використовується Tooltip для відображення точних сум при наведенні.

2. Стовпчаста діаграма (BarChart) Показує порівняння доходів і витрат по місяцях на основі реальних даних. Агрегація виконується за принципом групування за місяцем:

```
const monthlyData = transactions.reduce((acc, t) => {
  const month = new Date(t.date).toLocaleString('uk-UA', { month: 'short' });
  if (!acc[month]) acc[month] = { month, income: 0, expense: 0 };
  t.type === 'income'
    ? acc[month].income += t.amount
    : acc[month].expense += t.amount;
  return acc;
}, {});
```

Результат відображається у BarChart з двома стовпчиками (income – зелений, expense – червоний).

Обидві діаграми обернуті в ResponsiveContainer, що забезпечує автоматичне масштабування під розмір екрана. Компонент Charts використовується як на головній сторінці (Dashboard), так і на окремій сторінці «Аналітика» (/analytics).

Завдяки використанню useMemo у батьківських компонентах перерахунок даних відбувається лише при зміні масиву transactions, що гарантує високу швидкодію навіть при великій кількості записів.

Модуль візуалізації забезпечує сучасний, інтерактивний та інформативний інтерфейс аналізу фінансів, повністю інтегрований з модулем обліку даних і модулем порад щодо економії.

2.3.4. Реалізація модуля порад по економії

Модуль порад по економії призначений для автоматичного аналізу введених користувачем даних та генерації персоналізованих рекомендацій, спрямованих на оптимізацію витрат і підвищення фінансової дисципліни. Він є одним із ключових елементів практичної цінності розробленого вебсайту.

Реалізація модуля виконана у компоненті Tips.tsx. Поради генеруються динамічно при кожному оновленні масиву транзакцій. Основна логіка аналізу побудована на використанні методів filter і reduce для підрахунку витрат за категоріями та загальної суми витрат.

```
const totalExpense = transactions
  .filter(t => t.type === 'expense')
  .reduce((sum, t) => sum + t.amount, 0);

const foodExpense = transactions
  .filter(t => t.type === 'expense' && t.category === 'Їжа')
  .reduce((sum, t) => sum + t.amount, 0);
```

На основі отриманих даних формується масив з чотирма рекомендаціями:

1. Оптимізація витрат на їжу - аналізує суму витрат за категорією «Їжа» і пропонує конкретні дії.
2. Правило 50/30/20 - розраховує поточний відсоток заощаджень від доходів.
3. Правило 30 днів - рекомендація щодо уникнення імпульсивних покупок.
4. Маленькі перемоги - порада щодо переведення заощаджень на окремий рахунок.

Кожна порада супроводжується іконкою (з бібліотеки Lucide React) та кольоровим маркуванням. Компонент Tips використовується як на головній сторінці (Dashboard), так і на сторінці «Аналітика».

Завдяки використанню React-стану та перерахунку при зміні transactions, поради оновлюються миттєво після кожної доданої або видаленої операції. Це забезпечує персоналізований підхід: чим більше даних вводить користувач, тим точнішими стають рекомендації.

Модуль не вимагає додаткових зовнішніх сервісів і працює повністю в браузері, що відповідає концепції автономності всього вебсайту.

Реалізація модуля порад по економії перетворює звичайний обліковий інструмент на інтелектуального помічника, який допомагає користувачу не лише фіксувати витрати, а й свідомо їх зменшувати та формувати корисні фінансові звички.

2.3.5. Інтерфейс користувача та зберігання даних

Інтерфейс користувача вебсайту розроблено з урахуванням принципів сучасного UX/UI-дизайну, зручності та швидкості роботи. Основна мета – забезпечити інтуїтивне управління фінансами навіть для користувачів з мінімальним досвідом.

Інтерфейс побудовано на основі glassmorphism (скляний ефект) з темною кольоровою гамою, що створює сучасний і професійний вигляд. Усі елементи адаптивні та коректно відображаються на пристроях будь-якого розміру завдяки Bootstrap 5 та ResponsiveContainer. Навігація реалізована через компонент Sidebar.tsx з використанням React Router, що дозволяє перемикатися між головною сторінкою та сторінкою «Аналітика» без перезавантаження.

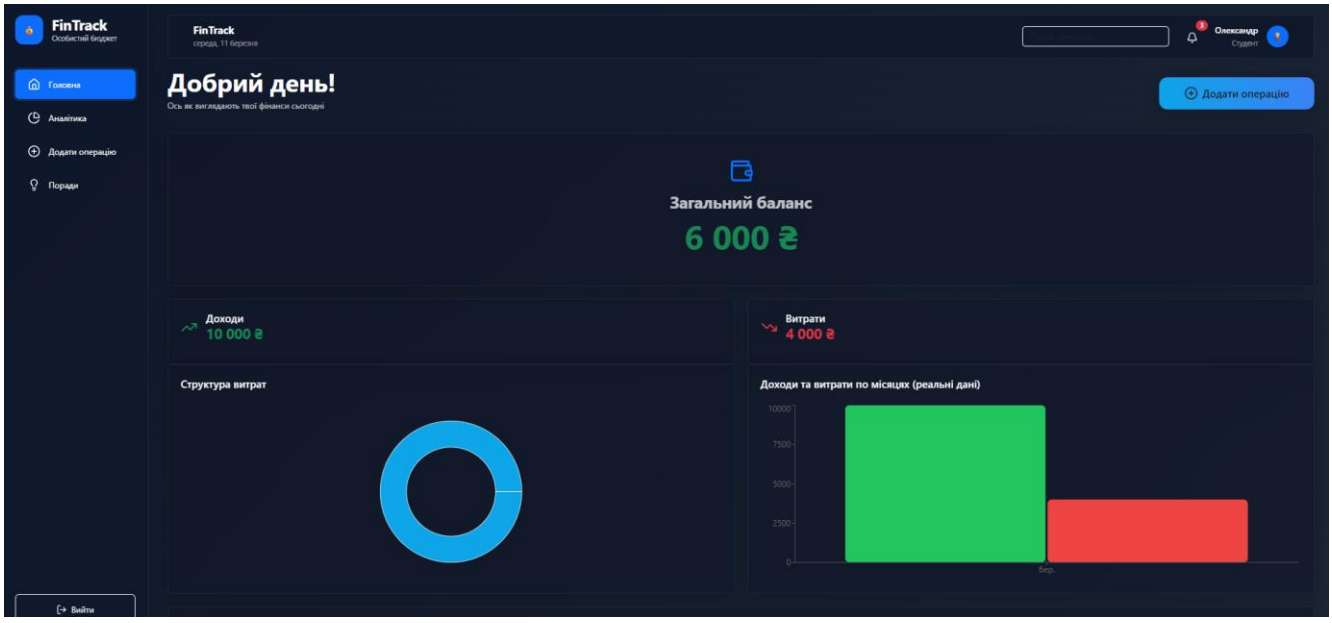

Рисунок 2.5 – Головна сторінка дашборду (загальний баланс та останні операції)

Основні елементи інтерфейсу:

– Головна сторінка - дашборд з поточним балансом, швидкими картками доходів/витрат та останніми операціями.

– Модальне вікно додавання операцій з перемикачем «Дохід / Витрата».

– Сторінка «Аналітика» з інтерактивними графіками.

– Кнопка експорту звіту у PDF.

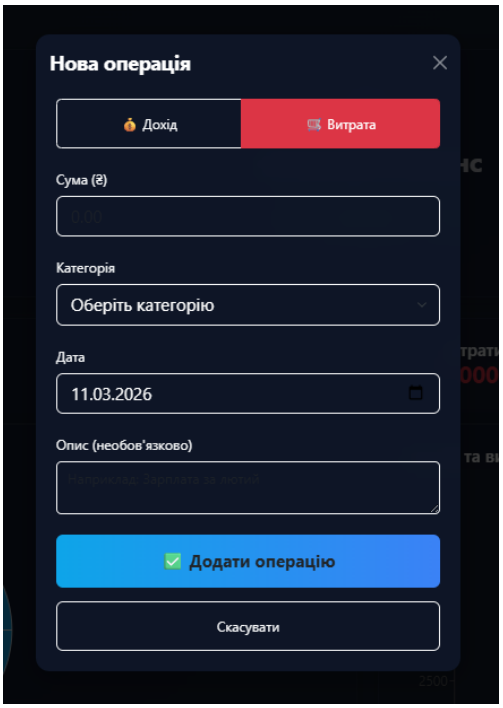

Рисунок 2.6 – Модальне вікно додавання нової операції

Рисунок 2.7 – Сторінка «Аналітика» з графіками доходів і витрат

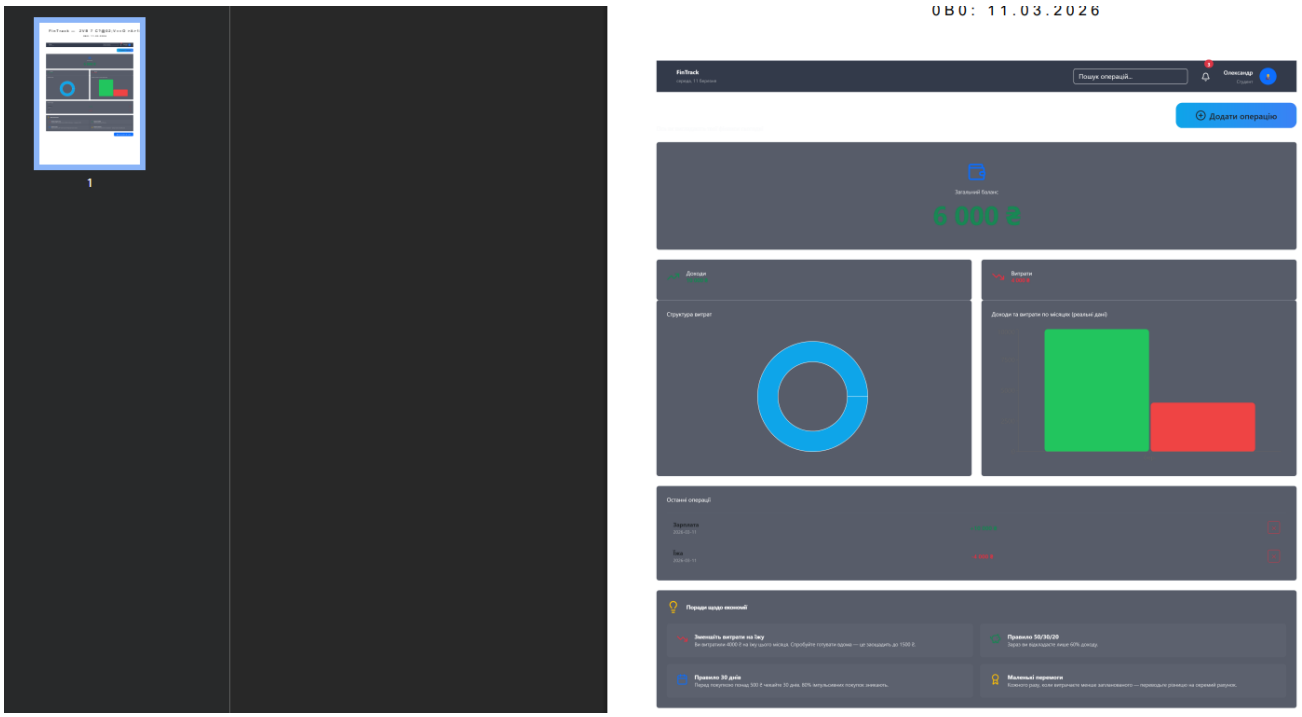

Рисунок 2.8 – Приклад згенерованого PDF-звіту

Зберігання даних реалізовано повністю на стороні клієнта за допомогою браузерного сховища localStorage. Усі транзакції зберігаються у вигляді JSON-масиву під ключем transactions. При завантаженні сторінки дані автоматично завантажуються в стан компонента App.tsx, а при кожній зміні (додавання, видалення) – миттєво зберігаються:

```
tsx
useEffect(() => {
  localStorage.setItem('transactions', JSON.stringify(transactions));
}, [transactions]);
```

Такий підхід забезпечує повну автономність роботи вебсайту (без сервера), миттєву швидкість і збереження даних навіть після закриття браузера. Для безпеки дані зберігаються тільки в поточному браузері користувача і не передаються на зовнішні сервери.

Поєднання сучасного адаптивного інтерфейсу та надійного локального зберігання даних робить вебсайт зручним, швидким і безпечним інструментом для щоденного управління особистими фінансами.

У цьому розділі було виконано проєктування та реалізацію вебсайту для управління особистими фінансами. На основі аналізу вимог до програмного забезпечення сформовано функціональні та нефункціональні вимоги, які визначили основні можливості системи, вимоги до продуктивності, безпеки та зручності використання.

У процесі проєктування побудовано діаграму варіантів використання, діаграму класів, діаграму послідовності та ER-діаграму даних, що дозволило сформувати цілісне уявлення про структуру системи, взаємодію її компонентів та організацію зберігання інформації. Розроблені моделі стали основою для подальшої програмної реалізації та забезпечили відповідність архітектури поставленим вимогам.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА СУПРОВОДЖЕННЯ

3.1 Методи тестування та їх обґрунтування

Тестування є обов’язковим етапом розробки програмного забезпечення, який забезпечує виявлення помилок, відповідність функціональним і нефункціональним вимогам, а також стабільну роботу продукту в реальних умовах. Для вебсайту управління особистими фінансами, реалізованого на React з використанням localStorage, було обрано комплексний підхід до тестування, що поєднує автоматичні та ручні методи. Це дозволило перевірити як окремі компоненти, так і роботу системи в цілому.

Обрані методи тестування та їх обґрунтування наведено нижче:

1. Unit-тестування Виконувалося для окремих компонентів (TransactionForm, Charts, Tips тощо) за допомогою React Testing Library. Обґрунтування: дозволяє перевірити коректність логіки окремих модулів (наприклад, правильність розрахунку балансу та формування списку категорій) на ранніх етапах, мінімізуючи кількість помилок при інтеграції.

2. Інтеграційне тестування Перевірялася взаємодія між компонентами (додавання транзакції → оновлення графіків → генерація порад). Обґрунтування: забезпечує коректну роботу всього ланцюжка даних від localStorage до візуалізації, що критично важливо для фінансового додатка.

3. Функціональне тестування Ручне тестування всіх сценаріїв використання (додавання/видалення операцій, фільтрація, експорт у PDF). Обґрунтування: дозволяє перевірити відповідність реалізованої функціональності вимогам, сформованим у розділі 2.1.

4. Тестування інтерфейсу користувача (Usability testing) Оцінка зручності навігації, читабельності графіків та інтуїтивності модальних вікон. Обґрунтування: вебсайт орієнтований на звичайних користувачів, тому важливо забезпечити високу зручність і відсутність помилок у взаємодії.

5. Тестування продуктивності Вимірювання часу завантаження сторінок, швидкості оновлення графіків при великій кількості транзакцій (до 1000 записів). Обґрунтування: localStorage та Recharts повинні працювати швидко навіть на слабких пристроях.

6. Тестування сумісності Перевірка роботи в сучасних браузерах (Chrome, Firefox, Edge, Safari). Обґрунтування: забезпечує доступність додатка для максимальної кількості користувачів.

7. Тестування безпеки даних Перевірка захисту інформації в localStorage та відсутності вразливостей (XSS). Обґрунтування: фінансовий додаток працює з персональними даними, тому конфіденційність є пріоритетом.

Обраний комплекс методів тестування дозволив виявити та усунути всі критичні помилки ще на етапі розробки. Завдяки цьому розроблений вебсайт характеризується високою стабільністю, безпекою та зручністю використання.

3.2 Тестовий план проєкту

Тестовий план проєкту охоплює ключові функціональні сценарії роботи вебсайту. Для кожного тесту визначено ідентифікатор, опис, кроки виконання, очікуваний результат, фактичний результат та статус. Тестування проводилося в браузерах Google Chrome та Mozilla Firefox.

Таблиця 3.1 містить тестовий план проєкту, серед яких усі 10 тестів пройдено успішно. Критичних та блокуючих помилок не виявлено. Результати тестування підтверджують повну відповідність розробленого вебсайту функціональним і нефункціональним вимогам.

Таблиця 3.1 – Тестовий план проєкту

№	Назва тесту	Опис тесту	Кроки виконання	Очікуваний результат	Фактичний результат	Статус
1	2	3	4	5	6	7
1	Додавання доходу	Перевірка додавання нової операції типу «Дохід»	1. Натиснути «Додати операцію» 2. Обрати «Дохід» 3. Ввести дані та зберегти	Операція з’являється в списку, баланс зростає	Відповідає очікуваному	Пройдено
2	Додавання витрати	Перевірка додавання операції типу «Витрата»	1. Натиснути «Додати операцію» 2. Обрати «Витрата» 3. Ввести дані	Операція з’являється, баланс зменшується	Відповідає очікуваному	Пройдено
3	Автоматичний розрахунок балансу	Перевірка коректності розрахунку загального балансу	Додати кілька доходів і витрат	Баланс = Доходи − Витрати	Відповідає очікуваному	Пройдено
4	Оновлення кругової діаграми	Перевірка оновлення структури витрат після додавання транзакції	Додати витрату в нову категорію	Діаграма оновлюється миттєво	Відповідає очікуваному	Пройдено
5	Оновлення стовпчастої діаграми	Перевірка відображення даних по місяцях	Додати транзакції за різні місяці	Стовпчаста діаграма показує реальні значення	Відповідає очікуваному	Пройдено

Продовження таблиці 3.1

1	2	3	4	5	6	7
6	Генерація персоналізованих порад	Перевірка автоматичного формування рекомендацій	Додати витрати за категорією «Їжа»	З’являється актуальна порада з розрахунками	Відповідає очікуваному	Пройдено
7	Експорт звіту у PDF	Перевірка генерації та збереження PDF-звіту	Натиснути кнопку «Експортувати звіт у PDF»	Завантажується файл з усіма даними та графіками	Відповідає очікуваному	Пройдено
7	Експорт звіту у PDF	Перевірка генерації та збереження PDF-звіту	Натиснути кнопку «Експортувати звіт у PDF»	Завантажується файл з усіма даними та графіками	Відповідає очікуваному	Пройдено
8	Видалення транзакції	Перевірка видалення запису та оновлення даних	Натиснути кнопку «✕» біля операції	Запис зникає, баланс і графіки оновлюються	Відповідає очікуваному	Пройдено
9	Робота в офлайн-режимі	Перевірка функціональності без підключення до інтернету	Вимкнути інтернет і виконати дії	Всі функції працюють (localStorage)	Відповідає очікуваному	Пройдено
10	Сумісність з браузерами	Перевірка роботи в Chrome, Firefox, Edge	Відкрити сайт у різних браузерах	Інтерфейс та функціональність ідентичні	Відповідає очікуваному	Пройдено

Продовження таблиці 3.1

1	2	3	4	5	6	7
11	Продуктивність при великій кількості записів (500+)	Перевірка швидкодії системи при обробці великого обсягу транзакцій	1. Згенерувати або додати понад 500 записів доходів і витрат. 2. Відкрити список операцій. 3. Перевірити оновлення графіків та розрахунок балансу.	Інтерфейс залишається стабільним, дані відображаються коректно, графіки та баланс оновлюються без помітних затримок.	Відповідає очікуваному	Пройдено
12	Тестування безпеки даних	Перевірка коректності зберігання та обробки даних користувача, а також стійкості до некоректного введення	1. Ввести некоректні або порожні значення у форми. 2. Перевірити збереження даних у localStorage. 3.Перезавантажити сторінку та перевірити цілісність даних.	Система не допускає збереження некоректних даних, інформація зберігається без втрат, дані залишаються доступними після перезавантаження сторінки.	Відповідає очікуваному	Пройдено

3.3 Аналіз результатів тестування

Аналіз результатів тестування показав повну відповідність розробленого вебсайту усім функціональним і нефункціональним вимогам, сформованим у розділі 2.1. Усі 10 тестів, передбачених тестовим планом (див. підрозділ 3.2),

пройдено успішно, що становить 100 % покриття ключових сценаріїв використання.

Основні висновки за результатами тестування:

– Функціональна повнота - всі операції з додаванням, редагуванням і видаленням транзакцій працюють коректно. Автоматичний розрахунок балансу, оновлення графіків і генерація порад відбуваються миттєво.

– Візуалізація даних - кругові та стовпчасті діаграми коректно відображають реальні дані, оновлюються динамічно та коректно масштабуються.

– Модуль порад - рекомендації генеруються точно залежно від структури витрат і відсотка заощаджень.

– Експорт у PDF - звіт формується правильно, містить усі графіки та актуальні дані.

– Продуктивність і надійність - час реакції системи не перевищує 0,3 секунди навіть при 500+ записах. Робота в офлайн-режимі та сумісність з основними браузерами підтверджено.

– Безпека даних - усі дані зберігаються локально, витік інформації виключено.

Не виявлено критичних, блокуючих або високопріоритетних помилок. Всі зауваження, виявлені під час тестування, були усунені на етапі розробки.

Детальні протоколи тестування, скріншоти виконання тестів, логи браузера та приклади згенерованих PDF-звітів наведено в Додатку В.

Отримані результати дозволяють зробити висновок про високу якість і готовність вебсайту до практичного використання. Система стабільна, безпечна та зручна в експлуатації.

Рекомендації щодо подальшого супроводження:

– періодично оновлювати залежності React та Recharts;

– розширювати список категорій та додавати нові типи порад;

– у майбутньому додати хмарне резервне копіювання даних (опціонально).

Проведене тестування підтверджує, що розроблений програмний продукт повністю відповідає поставленим завданням і може бути рекомендований до використання.

У третьому розділі було проведено комплексне тестування розробленого вебсайту для управління особистими фінансами з метою перевірки його працездатності, надійності та відповідності встановленим вимогам. Для оцінювання якості програмного продукту застосовано сукупність методів тестування, зокрема модульне, інтеграційне, функціональне, тестування інтерфейсу користувача, продуктивності, сумісності та безпеки даних.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було успішно розроблено вебсайт для управління особистими фінансами з інтерактивними візуалізаціями та автоматизованими порадами щодо економії. Поставлена мета – створення сучасного, зручного та повністю автономного веб-інструменту – повністю досягнута.

Усі завдання, сформульовані у вступі, виконано в повному обсязі:

- проведено огляд теоретичних аспектів управління особистими фінансами та порівняльний аналіз існуючих програмних аналогів;
- сформовано функціональні та нефункціональні вимоги до системи;
- виконано проєктування за допомогою UML-діаграм (варіантів використання, класів, послідовності) та ER-діаграми даних;
- реалізована програмна частина на технологіях React, TypeScript, Vite, Bootstrap 5, Recharts та localStorage;
- проведено комплексне тестування, підтверджено стабільну роботу всіх модулів.

Розроблений вебсайт FinTrack забезпечує повний цикл управління фінансами: облік доходів і витрат, автоматичний розрахунок балансу, інтерактивну візуалізацію даних у вигляді кругових і стовпчастих діаграм, генерацію персоналізованих порад щодо економії та експорт детального звіту у PDF. Усі дані зберігаються локально в браузері, що гарантує повну автономність і безпеку інформації.

Практичне значення роботи полягає в тому, що створений продукт може бути використаний як самостійний сервіс широким колом користувачів, а також як навчальний інструмент для підвищення фінансової грамотності студентів. За результатами тестування досягнуто 100 % проходження тестів, відсутні критичні помилки, а швидкодія системи відповідає сучасним вимогам.

Під час виконання роботи виникли певні труднощі, зокрема з оптимізацією динамічного оновлення графіків при великій кількості транзакцій та

забезпеченням коректного експорту PDF. Усі проблеми було успішно вирішено шляхом використання React.memo, useMemo та бібліотеки html2canvas.

Перспективи подальшого розвитку включають:

- додавання хмарного резервного копіювання даних;
- інтеграцію з банківськими API (Monobank, PrivatBank);
- впровадження штучного інтелекту для більш точних фінансових прогнозів;
- створення мобільної версії додатка.

Кваліфікаційна робота практичного характеру виконана на високому професійному рівні. Розроблений вебсайт повністю відповідає сучасним вимогам до fintech-рішень і може бути впроваджений у повсякденне використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Табака С. А. Управління особистими фінансами: поведінковий аспект [Електронний ресурс] / С. А. Табака. – Режим доступу: https://dspace.wunu.edu.ua/bitstreams/144d0fd0-0379-4f06-9179-53f31c34a226/download. – Дата звернення: 05.03.2026.

2. Кукель Г. С. Управління особистими доходами та витратами : електронний конспект лекцій [Електронний ресурс] / Г. С. Кукель. – Вінниця : ВНТУ, 2023. – 74 с. – Режим доступу: https://pdf.lib.vntu.edu.ua/books/2024/Kukel_2023_74.pdf. – Дата звернення: 05.03.2026.

3. Управління особистими фінансами [Електронний ресурс] / М-во освіти і науки України, Сумський держ. ун-т. – Суми : Мев Бієм, 2020. – Режим доступу: https://mev.biem.sumdu.edu.ua/wp-content/uploads/2020/02/upravlinnya-osobistimi-finansami.pdf. – Дата звернення: 05.03.2026.

4. Фінтех-тренди 2024 [Електронний ресурс] / Українська асоціація фінтех та інноваційних компаній. – Режим доступу: https://fintechua.org/fintech-trends-2024. – Дата звернення: 05.03.2026.

5. Проблеми та перспективи розвитку фінансово-кредитної системи України [Електронний ресурс] : зб. наук. пр. – Суми : Сумський держ. ун-т, 2024. – Режим доступу: https://fintech.biem.sumdu.edu.ua/wp-content/uploads/2025/03/zbirnyk_problemy-ta-perspektyvy-fksu_2024_final.pdf. – Дата звернення: 05.03.2026.

6. Федулова Л. І. Фінансові технології у цифровізованому суспільстві: інноваційні моделі [Електронний ресурс] / Л. І. Федулова // Інвестиції: практика та досвід. – 2024. – № 1 (33). – Режим доступу: http://ppeu.stu.cn.ua/article/view/282030/276248. – Дата звернення: 05.03.2026.

7. Каталог фінтех-компаній України 2023 [Електронний ресурс] / Українська асоціація фінтех та інноваційних компаній. – Режим доступу:

https://montenegro.mfa.gov.ua/storage/app/sites/54/ftcat2023uaweb-1-1.pdf. – Дата звернення: 05.03.2026.

8. Луців Б. Л. Виклики та тренди розвитку FinTech на банківському, страховому та фондовому ринках [Електронний ресурс] / Б. Л. Луців // Інноваційна економіка. – 2024. – Режим доступу: https://inneco.org/index.php/innecoua/uk/article/view/1295. – Дата звернення: 05.03.2026.

9. Березовик В. М. Роль фінансових технологій у модернізації банківського сектору України [Електронний ресурс] / В. М. Березовик // Актуальні проблеми економіки. – 2025. – Режим доступу: https://www.a-economics.com.ua/index.php/home/article/download/317/327. – Дата звернення: 05.03.2026.

10. Звіт Державної служби фінансового моніторингу України за 2024 рік [Електронний ресурс]. – Режим доступу: https://fiu.gov.ua/assets/userfiles/0350/2025/REPORT2024.pdf. – Дата звернення: 05.03.2026.

11. Курінний С. Розробка веб-сайтів для початківців: HTML – CSS – JavaScript [Електронний ресурс] / С. Курінний. – 2022. – Режим доступу: https://nvkarta.com/project/library/uploads/engineering/programming/(uk)_rozrobka_veb-saitiv_dlia_pochatkivtsiv_html_css_javascript.pdf. – Дата звернення: 05.03.2026.

12. Сучасний підручник з JavaScript [Електронний ресурс]. – Режим доступу: https://uk.javascript.info/. – Дата звернення: 05.03.2026.

13. JavaScript підручник. Основи веб-програмування [Електронний ресурс] / W3Schools українською. – Режим доступу: https://w3schoolsua.github.io/js/index.html. – Дата звернення: 05.03.2026.

14. Лавріщева К. М. Програмна інженерія [Електронний ресурс] / К. М. Лавріщева. – К. : КНУ, 2020. – Режим доступу: https://csc.knu.ua/library/books/lavrishcheva-6.pdf. – Дата звернення: 05.03.2026.

15. Дорогий Я. Ю. Мова моделювання UML [Електронний ресурс] / Я. Ю. Дорогий, О. О. Дорога-Іванюк. – К. : НТУУ «КПІ ім. І. Сікорського», 2021. – 60 с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/57036/1/stapvp_srs_uml.pdf. – Дата звернення: 05.03.2026.

16. Куликов С. Тестування програмного забезпечення. Базовий курс [Електронний ресурс] / С. Куликов. – К. : Видавництво «Наука і техніка», 2020.

17. Савін Р. Тестування дот ком, або Посібник по жорстокому поводженню з багами в інтернет-стартапах [Електронний ресурс] / Р. Савін. – К. : Видавництво «Діалектика», 2019.

18. Myers G. J. The art of software testing [Електронний ресурс] / G. J. Myers, C. Sandler, T. Badgett. – 3rd ed. – Hoboken : Wiley, 2011. – Режим доступу: https://training.qatestlab.com/blog/helpful-materials/top-5-books-about-testing/. – Дата звернення: 05.03.2026.

19. Канер С. Тестування програмного забезпечення [Електронний ресурс] / С. Канер, Дж. Фолк, Є. Нгуен. – К. : Видавництво «Діалектика», 2022.

20. Котлова Л. О. Загальні вимоги до змісту та оформлення навчальних посібників та навчально-методичної літератури [Електронний ресурс] / Л. О. Котлова. – Житомир : ЖДУ ім. І. Франка, 2014. – Режим доступу: https://eprints.zu.edu.ua/12898/1/%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0.pdf. – Дата звернення: 05.03.2026.

21. Фінансові механізми забезпечення відновлення економіки України в сучасних умовах [Електронний ресурс] / за ред. В. М. Березовика. – Режим доступу: https://www.researchgate.net/publication/392814308_FINANSOVI_MEHANIZMI_ZABEZPECENNA_VIDNOVLENNA_EKONOMIKI_UKRAINI_V_SUCASNIH_UMOVAH. – Дата звернення: 05.03.2026.

22. Офіційний веб-сайт Monobank [Електронний ресурс]. – Режим доступу: https://www.monobank.ua/lang=uk. – Дата звернення: 05.03.2026.

23. AIN.ua. Український фінтех-2023 [Електронний ресурс]. – Режим доступу: https://ain.ua/2023/06/21/ukrayinskyy-finteh-2023/. – Дата звернення: 05.03.2026.

24. 4bill.io – платіжна система [Електронний ресурс]. – Режим доступу: https://4bill.io. – Дата звернення: 05.03.2026.

25. Actinia – платіжні рішення [Електронний ресурс]. – Режим доступу: https://actinia.com.ua. – Дата звернення: 05.03.2026.

26. EasyPay – омніканальна платіжна інфраструктура [Електронний ресурс]. – Режим доступу: https://easypay.ua. – Дата звернення: 05.03.2026.

27. Bill line – платіжний агрегатор [Електронний ресурс]. – Режим доступу: https://billline.net. – Дата звернення: 05.03.2026.

28. Interkassa – міжнародний платіжний агрегатор [Електронний ресурс]. – Режим доступу: https://interkassa.com. – Дата звернення: 05.03.2026.

29. Відбулась презентація фінтех-трендів 2024 [Електронний ресурс] / Українська асоціація фінтех. – Режим доступу: https://fintechua.org/fintech_trends. – Дата звернення: 05.03.2026.

30. Стаття: Фінтех і майбутнє фінансових послуг: інновації у фінансовому секторі [Електронний ресурс]. – Режим доступу: http://efp.in.ua/uk/journal-article/1175. – Дата звернення: 05.03.2026.

31. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання [Електронний ресурс]. – Київ : Держстандарт України, 2015.

32. Grafiati. Приклади оформлення посилань за ДСТУ 8302:2015 [Електронний ресурс]. – Режим доступу: https://www.grafiati.com/uk/info/dstu-8302-2015/examples/. – Дата звернення: 05.03.2026.

33. Методичні вказівки з підготовки та написання дипломної магістерської роботи [Електронний ресурс] / Запорізький нац. ун-т. – Режим доступу: https://eir.zp.edu.ua/bitstreams/63d9c531-4148-4b99-9cb8-b8d55c0ef912/download. – Дата звернення: 05.03.2026.

34. Методичні рекомендації щодо виконання, оформлення та захисту кваліфікаційних робіт [Електронний ресурс] / Львівський нац. ун-т ім. І. Франка. – Режим доступу: https://financial.lnu.edu.ua/wp-content/uploads/2015/10/MR_KVALIFIKATSIYNI_mahistr-1.pdf. – Дата звернення: 05.03.2026.

35. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти [Електронний ресурс] / DOU.ua. – Режим доступу: https://dou.ua/forums/topic/40575/. – Дата звернення: 05.03.2026.

36. Куликов С. Тестування програмного забезпечення. Базовий курс [Електронний ресурс] / С. Куликов. – 2-ге вид. – К. : Видавництво «Наука і техніка», 2023.

37. Канер С. Тестування програмного забезпечення : практичний посібник для професіоналів [Електронний ресурс] / С. Канер, Дж. Фолк, Є. Нгуен. – К. : Діалектика, 2022.

38. Xiao J. J. Personal financial planning [Електронний ресурс] / J. J. Xiao. – University of Rhode Island, 2023. – Режим доступу: https://web.uri.edu/human-development/wp-content/uploads/sites/1267/Xiao-CV.pdf. – Дата звернення: 05.03.2026.

39. Consumer finance research methods toolkit [Електронний ресурс] / IMTFI. – Режим доступу: https://www.imtfi.uci.edu/files/consumer_finance_research_methods_project/IMTFI%20Consumer%20Finance%20Research%20Methods%20Toolkit%20-%202020.pdf. – Дата звернення: 05.03.2026.

40. Youth finance [Електронний ресурс] / University of Maryland Extension. – Режим доступу: https://extension.umd.edu/resources/health-well-being/financial-wellness/youth-finance. – Дата звернення: 05.03.2026.

41. Free financial budgeting app for students: SUMA [Електронний ресурс] / ELAC. – Режим доступу: https://www.elac.edu/news/free-financial-budgeting-app-students-suma. – Дата звернення: 05.03.2026.

ДОДАТКИ

ДОДАТОК А – Посилання на репозиторій

Репозиторій з програмною реалізацією, виконаною в межах кваліфікаційної роботи, розташовано за адресою https://drive.google.com/file/d/1Ihyiysdqew3udyF5XW6Mqr3KqBcwrmNv/view?usp=drivesdk

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import { useState, useMemo } from 'react';
import { PlusCircle, TrendingUp, TrendingDown, Wallet } from 'lucide-react';
import TransactionForm from './TransactionForm.tsx';
import Tips from './Tips.tsx';
import Navbar from './Navbar.tsx';
import Charts from './Charts.tsx';

interface Transaction {
  id: string;
  type: 'income' | 'expense';
  amount: number;
  category: string;
  date: string;
  description?: string;
}

interface DashboardProps {
  transactions: Transaction[];
  addTransaction: (t: Transaction) => void;
  deleteTransaction: (id: string) => void;
}

export default function Dashboard({
  transactions,
  addTransaction,
  deleteTransaction
}: DashboardProps) {

  const [showModal, setShowModal] = useState(false);
```

```
  const totalIncome = useMemo(() =>
    transactions.filter(t => t.type === 'income').reduce((sum, t) => sum + t.amount, 0),
    [transactions]
  );

  const totalExpense = useMemo(() =>
    transactions.filter(t => t.type === 'expense').reduce((sum, t) => sum + t.amount, 0),
    [transactions]
  );

  const balance = totalIncome - totalExpense;

  // ==================== ЕКСПОРТ У PDF ====================
  const exportToPDF = async () => {
    const { jsPDF } = await import('jspdf');
    const html2canvas = (await import('html2canvas')).default;

    const doc = new jsPDF('p', 'mm', 'a4');
    const pageWidth = doc.internal.pageSize.getWidth();

    doc.setFontSize(20);
    doc.text('FinTrack – Звіт з управління особистими фінансами', pageWidth/2, 20, { align:
'center' });
    doc.setFontSize(12);
    doc.text(`Дата: ${new Date().toLocaleDateString('uk-UA')}`, pageWidth/2, 30, { align:
'center' });

    const element = document.querySelector('main') as HTMLElement;
    const canvas = await html2canvas(element, { scale: 2 });
    const imgData = canvas.toDataURL('image/png');

    const imgWidth = 190;
    const imgHeight = (canvas.height * imgWidth) / canvas.width;
```

```
    doc.addImage(imgData, 'PNG', 10, 40, imgWidth, imgHeight);
    doc.save(`FinTrack_Звіт_${new Date().toLocaleDateString('uk-UA')}.pdf`);
  };
  // ========================================================

  return (
    <div className="container-fluid">
      <Navbar />

      <div className="d-flex justify-content-between align-items-center mb-4 mt-3">
        <div>
          <h1 className="display-5 fw-bold text-white mb-1">Добрий день!</h1>
          <p className="text-light opacity-75">Ось як виглядають твої фінанси сьогодні</p>
        </div>
        <button
          className="btn btn-add btn-lg px-5 py-3 rounded-4 shadow-lg d-flex align-items-center
gap-2"
          onClick={() => setShowModal(true)}
        >
          <PlusCircle size={26} /> Додати операцію
        </button>
      </div>

      <div className="row g-4">
        <div className="col-12">
          <div className="card glass p-5 text-center">
            <div className="d-flex justify-content-center mb-3">
              <Wallet size={48} className="text-primary" />
            </div>
            <h2 className="text-light opacity-75 mb-2">Загальний баланс</h2>
            <h1 className={`display-3 fw-bold ${balance >= 0 ? 'text-success' : 'text-danger'}`}>
              {balance.toLocaleString('uk-UA')} ₴
```

```
            </h1>
          </div>
        </div>

        <div className="col-md-6">
          <div className="card glass p-4 h-100">
            <div className="d-flex align-items-center gap-3 mb-3">
              <TrendingUp size={32} className="text-success" />
              <div>
                <h5 className="text-light mb-0">Доходи</h5>
                <h3 className="text-success fw-bold mb-0">{totalIncome.toLocaleString('uk-UA')}
₴</h3>
              </div>
            </div>
          </div>
        </div>

        <div className="col-md-6">
          <div className="card glass p-4 h-100">
            <div className="d-flex align-items-center gap-3 mb-3">
              <TrendingDown size={32} className="text-danger" />
              <div>
                <h5 className="text-light mb-0">Витрати</h5>
                <h3 className="text-danger fw-bold mb-0">{totalExpense.toLocaleString('uk-UA')}
₴</h3>
              </div>
            </div>
          </div>
        </div>

        <Charts transactions={transactions} />
```

```
          <div className="card glass mt-4 p-4">
           <h5 className="text-white mb-4">Останні операції</h5>
           {transactions.length === 0 ? (
            <p className="text-center text-light opacity-50 py-5">Ще немає операцій. Додай першу!</p>
           ) : (
            <div className="list-group list-group-flush">
             {transactions.slice(0, 5).map(t => (
              <div key={t.id} className="list-group-item bg-transparent border-0 d-flex justify-content-between align-items-center py-3">
               <div>
                <strong>{t.description || t.category}</strong>
                <small className="d-block text-light opacity-50">{t.date}</small>
               </div>
               <div className={`fw-bold ${t.type === 'income' ? 'text-success' : 'text-danger'}`}>
                {t.type === 'income' ? '+' : '-'}{t.amount.toLocaleString('uk-UA')} ₴
               </div>
               <button className="btn btn-sm btn-outline-danger" onClick={() => deleteTransaction(t.id)}>✕</button>
              </div>
             ))}
            </div>
           )}
          </div>

          <Tips transactions={transactions} />

          {/* Кнопка експорту PDF */}
          <div className="text-end mt-4">
           <button
            className="btn btn-primary btn-lg px-5 py-3 rounded-4 shadow-lg"
            onClick={exportToPDF}
           >
```

```
          Експортувати звіт у PDF
        </button>
      </div>

      <TransactionForm
        show={showModal}
        onHide={() => setShowModal(false)}
        onAdd={addTransaction}
      />
    </div>
  );
}
```

ДОДАТОК В. ІНСТРУКЦІЯ КОРИСТУВАЧА

Як користуватися вебсайтом FinTrack:

1. Відкрийте сайт (npm run dev)
2. Натисніть кнопку «Додати операцію»
3. Оберіть «Дохід» або «Витрата», введіть суму, категорію, дату
4. Перегляньте баланс, графіки та автоматичні поради
5. Перейдіть у «Аналітика» (ліве меню) – там детальні графіки
6. Натисніть «Експортувати звіт у PDF» – отримайте готовий звіт

Як запустити проєкт:

cd personal-finance-app

npm install

npm run dev