МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБСИСТЕМИ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛЬНИМИ ФІНАНСАМИ З АНАЛІЗОМ ВИТРАТ ТА ПРОГНОЗУВАННЯМ БЮДЖЕТУ»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Д. Р. Романенко
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н Морозов Д. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Романенку Данилу Романовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Проєктування та розробка вебсистеми для управління персональними фінансами з аналізом витрат та прогнозуванням бюджету

керівник роботи Морозов Денис Миколайович, к.т.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та існуючих рішень у сфері персональних фінансів.
2. Проєктування вебсистеми для управління персональними фінансами.
3. Розробка та реалізація вебсистеми з аналізом витрат і прогнозуванням бюджету.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	20.02.2026	
2.	Збір вихідних даних.	18.03.2026	
3.	Обробка методичних та теоретичних	28.03.2026	
	джерел.		
4.	Розробка першого та другого розділу.	26.04.2026	
5.	Розробка третього розділу.	16.05.2026	
6.	Оформлення та нормоконтроль	04.06.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2026	

Студент _____ Д. Р. Романенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Д. М. Морозов
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування та розробка вебсистеми для управління персональними фінансами з аналізом витрат та прогнозуванням бюджету»: 77 с., 19 рис., 11 табл., 31 джерел, 2 додатки.

АНАЛІТИКА ДАНИХ, БАЗА ДАНИХ, БЮДЖЕТУВАННЯ, ВЕБЗАСТОСУНОК, ВІЗУАЛІЗАЦІЯ ДАНИХ, ПЕРСОНАЛЬНІ ФІНАНСИ, ПРОГНОЗУВАННЯ ВИТРАТ, УПРАВЛІННЯ ВИТРАТАМИ, SQLITE.

Об’єктом дослідження є процес управління персональними фінансами користувача в цифровому середовищі.

Предметом дослідження є методи та програмні засоби розробки вебсистем для обліку, аналізу та прогнозування фінансових даних.

Метою роботи є проєктування та розробка вебсистеми для управління персональними фінансами, яка забезпечує облік доходів і витрат, аналіз фінансової активності користувача та прогнозування бюджету на основі накопичених даних.

Методи дослідження – системний аналіз предметної області, порівняльний аналіз існуючих рішень, методи проєктування вебзастосунків і баз даних, методи агрегування даних SQL, метод лінійної регресії для прогнозування, а також тестування програмного забезпечення.

У роботі проаналізовано сучасні підходи до управління персональними фінансами та існуючі вебзастосунки для обліку витрат. Спроєктовано архітектуру системи, базу даних та модулі аналізу й бюджетування. Реалізовано функціонал обліку фінансових операцій, аналітики витрат із візуалізацією даних, а також модуль прогнозування бюджету на основі історичних значень. Розроблено клієнтський інтерфейс для зручної взаємодії користувача з системою. Проведене тестування підтвердило коректність роботи основних функцій та відповідність системи поставленим вимогам.

SUMMARY

Bachelor's Qualifying Thesis «Design and Development of a Web System for Personal Finance Management with Expense Analysis and Budget Forecasting»: 77 pages, 19 figures, 11 tables, 31 sources, 2 supplements.

DATA ANALYTICS, DATABASE, BUDGETING, WEB APPLICATION, DATA VISUALIZATION, PERSONAL FINANCE, EXPENSE FORECASTING, EXPENSE MANAGEMENT, SQLITE.

The object of the study is the process of managing a user's personal finances in a digital environment.

The subject of the study is methods and software tools for developing web systems for financial data accounting, analysis, and forecasting.

The aim of the work is to design and develop a web system for personal finance management that provides income and expense tracking, analysis of the user's financial activity, and budget forecasting based on accumulated data.

Research methods include system analysis of the subject area, comparative analysis of existing solutions, methods of web application and database design, SQL data aggregation methods, the least squares linear regression method for forecasting, and software testing.

The work analyzes modern approaches to personal finance management and existing web applications for expense tracking. The system architecture, database, and modules for financial analysis and budgeting are designed. The implemented system provides functionality for financial transaction management, expense analytics with data visualization, and a budget forecasting module based on historical values. A user interface is developed to ensure convenient interaction with the system. Testing confirmed the correctness of the main functions and compliance of the system with the defined requirements.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз предметної області та існуючих рішень у сфері персональних фінансів 9
1.1 Роль інформаційних систем у управлінні персональними фінансами 9
1.2 Функціональні можливості сучасних систем обліку доходів і витрат…. 11
1.3 Огляд існуючих вебзастосунків для фінансового планування 13
2 Проєктування вебсистеми для управління персональними фінансами 25
2.1 Аналіз вимог до вебсистеми 25
2.2 Проєктування архітектури вебзастосунку..... 28
2.3 Проєктування бази даних для фінансових операцій 32
2.4 Проєктування модулів аналізу витрат та бюджетування 35
2.5 Обґрунтування вибору технологій розробки 40
3 Розробка та реалізація вебсистеми з аналізом витрат і прогнозуванням бюджету 43
3.1 Реалізація серверної частини вебсистеми 43
3.2 Розробка клієнтського інтерфейсу користувача..... 47
3.3 Реалізація функціоналу обліку фінансових операцій 49
3.4 Реалізація аналітики витрат та прогнозування бюджету..... 53
3.5 Тестування вебсистеми 56
3.6 Інструкція користувача 57
Висновки 59
Перелік посилань..... 61
Додаток А Лістинг реалізації серверної частини вебсистеми..... 64
Додаток Б Інтерфейс користувача..... 74

ВСТУП

Сучасні інформаційні технології суттєво змінили підходи до управління фінансами як на рівні організацій, так і окремих користувачів. Розвиток вебсервісів і мобільних застосунків дозволив автоматизувати облік доходів і витрат, однак ефективність таких рішень залежить від їх функціональності та зручності використання. У цьому контексті актуальним є створення систем, які не лише фіксують фінансові операції, а й забезпечують аналіз витрат та прогнозування бюджету. Актуальність теми зумовлена потребою у підвищенні фінансової грамотності та зручних інструментах контролю особистих фінансів. Більшість існуючих рішень обмежуються базовим обліком транзакцій, тоді як користувачі потребують автоматичної категоризації витрат, аналітики та прогнозування бюджету на основі історичних даних.

Метою роботи є проєктування та розробка вебсистеми для управління персональними фінансами, яка забезпечує облік доходів і витрат, аналіз фінансової активності користувача та прогнозування бюджету на основі накопичених даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область управління персональними фінансами та існуючі програмні рішення;
- дослідити функціональні можливості сучасних систем обліку доходів і витрат;
- провести порівняльний аналіз існуючих вебзастосунків для фінансового планування;
- визначити вимоги до вебсистеми управління персональними фінансами;
- спроєктувати архітектуру вебзастосунку та структуру бази даних;

- реалізувати основний функціонал обліку та аналізу фінансових операцій.

Об’єктом дослідження є процес управління персональними фінансами користувача в цифровому середовищі. Предметом дослідження є методи та програмні засоби розробки вебсистем для обліку, аналізу та прогнозування фінансових даних.

Методи дослідження – системний аналіз предметної області, порівняльний аналіз існуючих рішень, методи проєктування вебзастосунків і баз даних, методи агрегування даних SQL, метод лінійної регресії для прогнозування, а також тестування програмного забезпечення.

Теоретичну основу дослідження становлять сучасні підходи до побудови вебзастосунків, методи обробки фінансових даних, аналізу часових рядів та принципи проєктування інформаційних систем. Практична частина роботи базується на розробці вебсистеми, яка реалізує функції обліку фінансових операцій, їх категоризації, аналітики витрат та прогнозування бюджету користувача.

Наукова новизна роботи полягає у поєднанні класичного підходу до обліку персональних фінансів із елементами аналітики та прогнозування, що дозволяє підвищити точність планування бюджету та рівень фінансового контролю користувача.

Практичне значення отриманих результатів полягає у можливості використання розробленої вебсистеми як інструменту для щоденного управління особистими фінансами, а також як основи для подальшого розвитку фінансових сервісів із елементами інтелектуального аналізу даних. Результати роботи можуть бути застосовані при створенні комерційних фінансових платформ, освітніх застосунків для підвищення фінансової грамотності та персональних систем бюджетування. Отримані результати можуть бути використані для подальшого вдосконалення систем управління персональними фінансами, зокрема шляхом інтеграції алгоритмів машинного навчання для більш точного прогнозування витрат та автоматичної оптимізації бюджету користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ У СФЕРІ ПЕРСОНАЛЬНИХ ФІНАНСІВ

1.1 Роль інформаційних систем у управлінні персональними фінансами

Інформаційні системи сьогодні є базовим інструментом організації та контролю персональних фінансів, оскільки дозволяють перетворювати хаотичні фінансові операції користувача на структуровані дані, придатні для аналізу та прийняття рішень. У традиційній практиці управління бюджетом часто базується на ручному обліку або використанні таблиць, що ускладнює оперативний контроль витрат і підвищує ризик помилок. Натомість сучасні інформаційні системи забезпечують автоматизований збір, збереження та обробку фінансової інформації, що значно підвищує точність і ефективність фінансового менеджменту [1].

Однією з ключових переваг інформаційних систем є автоматизація фінансового обліку. Сучасні застосунки можуть інтегруватися з банківськими сервісами, платіжними системами та електронними гаманцями, автоматично отримуючи дані про транзакції. Це дозволяє користувачу уникати ручного введення кожної операції, зменшує час на ведення обліку та мінімізує ймовірність пропусків або дублювання записів. У результаті формується більш повна та об’єктивна картина фінансового стану користувача в режимі реального часу [2].

Важливим функціональним аспектом є контроль витрат і бюджетування. Інформаційні системи дозволяють задавати фінансові ліміти за категоріями (наприклад, харчування, транспорт, розваги, комунальні послуги) та автоматично порівнювати фактичні витрати з запланованими. При перевищенні бюджету система може формувати сповіщення або візуальні індикатори, що допомагає користувачеві швидко реагувати та коригувати свої фінансові звички

[3]. Такий підхід сприяє формуванню дисциплінованого ставлення до витрат і підвищує рівень фінансового самоконтролю.

Не менш важливою функцією є аналітика фінансових даних. Інформаційні системи трансформують числові дані у зрозумілі візуальні форми – графіки, діаграми, звіти та дашборди. Це дозволяє користувачеві аналізувати динаміку доходів і витрат, виявляти сезонні коливання, визначати найбільш витратні категорії та оцінювати загальну фінансову стабільність [2]. Завдяки цьому процес прийняття рішень переходить від інтуїтивного до аналітично обґрунтованого.

Окрему роль відіграє прогнозування фінансового стану. На основі історичних даних системи можуть формувати прогнози майбутніх витрат або залишку бюджету. Це дозволяє користувачеві планувати фінанси на коротко- та довгострокову перспективу, оцінювати ризики дефіциту бюджету та заздалегідь коригувати фінансову поведінку. У більш розвинених системах для цього використовуються елементи статистичного аналізу та алгоритми машинного навчання [4].

Сучасні інформаційні системи також забезпечують персоналізацію фінансових рекомендацій. Вони аналізують поведінку користувача, його регулярні витрати, рівень доходів та фінансові цілі, щоб формувати індивідуальні поради. Наприклад, система може рекомендувати зменшити витрати на певні категорії або збільшити рівень заощаджень. Це перетворює систему з простого інструмента обліку на цифрового фінансового помічника, який підтримує користувача у прийнятті рішень [5].

Важливо також відзначити освітню та поведінкову функцію інформаційних систем. Постійна взаємодія з фінансовими звітами, графіками та нагадуваннями формує у користувача більш усвідомлене ставлення до грошей. З часом це сприяє розвитку фінансової грамотності, зменшенню імпульсивних витрат та формуванню стабільних навичок бюджетування [1]. Таким чином, система впливає не лише на облік фінансів, але й на фінансову поведінку користувача в цілому.

Разом із тим ефективність інформаційних систем залежить від низки факторів, серед яких особливо важливими є зручність інтерфейсу, якість обробки даних та рівень безпеки. Оскільки фінансова інформація є конфіденційною, системи повинні забезпечувати захист даних, автентифікацію користувачів і безпечне зберігання інформації. Крім того, складний або перевантажений інтерфейс може знизити залученість користувачів і зменшити ефективність використання системи [2].

Отже, інформаційні системи відіграють комплексну роль у персональному фінансовому менеджменті, поєднуючи функції обліку, контролю, аналізу, прогнозування та підтримки прийняття рішень. Їх використання дозволяє підвищити ефективність управління бюджетом, сформувати фінансову дисципліну та забезпечити більш усвідомлене ставлення користувачів до власних фінансів.

1.2 Функціональні можливості сучасних систем обліку доходів і витрат

Сучасні системи обліку доходів і витрат є багатофункціональними цифровими інструментами, що поєднують можливості автоматизованого збору фінансових даних, їх структурування, аналізу та підтримки прийняття фінансових рішень. На відміну від простих таблиць або ручного обліку, такі системи формують цілісне середовище фінансового менеджменту, яке дозволяє користувачеві контролювати власні грошові потоки в режимі реального часу та приймати більш обґрунтовані рішення щодо витрат і заощаджень. У науковій літературі підкреслюється, що сучасні фінансові застосунки вже виходять за межі базового обліку і виконують функції аналітики та прогнозування [6].

Однією з базових функцій є автоматизований облік фінансових операцій. Такі системи дозволяють користувачам вводити доходи та витрати вручну або автоматично синхронізувати дані з банківськими рахунками, платіжними картками та електронними сервісами. Це значно зменшує обсяг ручної роботи,

знижує ймовірність помилок і забезпечує актуальність фінансової інформації. Завдяки автоматизації користувач отримує точну картину своїх фінансів без необхідності постійного контролю кожної транзакції окремо.

Наступною важливою функцією є категоризація та групування фінансових операцій. Усі транзакції розподіляються за визначеними категоріями, такими як харчування, транспорт, житло, комунальні послуги, освіта, розваги або доходи з основних і додаткових джерел. Така структуризація дозволяє користувачеві аналізувати, на які сфери витрачається найбільше коштів, та виявляти потенційні можливості для оптимізації бюджету. У наукових дослідженнях зазначається, що саме категоризація є основою для подальшої аналітики фінансової поведінки користувача [7].

Важливим компонентом функціональності є бюджетування та фінансове планування. Системи дозволяють встановлювати фінансові ліміти для окремих категорій витрат або загального бюджету на певний період (місяць, тиждень, рік). У разі перевищення встановлених меж система може автоматично повідомляти користувача або візуально відображати перевищення бюджету. Це сприяє більш свідомому контролю витрат і допомагає уникати фінансових труднощів, пов’язаних із перевитратами. Бюджетування також виконує управлінську функцію, оскільки формує основу для фінансового планування.

Окрему роль відіграє аналітика та візуалізація фінансових даних. Сучасні системи перетворюють числову інформацію у зрозумілі графіки, діаграми та інтерактивні звіти. Це дозволяє користувачеві швидко оцінити структуру доходів і витрат, визначити найбільш витратні категорії та відстежувати зміни фінансового стану у динаміці. Візуалізація значно підвищує рівень розуміння фінансових процесів і сприяє більш ефективному прийняттю рішень [8].

Додатковою функцією сучасних систем є прогнозування фінансових показників. На основі історичних даних про доходи та витрати системи можуть формувати прогноз майбутнього стану бюджету, визначати можливий дефіцит коштів або надлишок. У більш розвинених рішеннях застосовуються елементи статистичного аналізу та алгоритми машинного навчання, що дозволяє

підвищити точність прогнозів [9]. Це дає користувачеві можливість заздалегідь планувати фінансові рішення та уникати небажаних ситуацій.

Сучасні системи також часто включають функції інтеграції з зовнішніми сервісами. Це може бути підключення до банківських систем, платіжних платформ, електронних гаманців або хмарних сховищ. Така інтеграція забезпечує автоматичне оновлення даних і дозволяє користувачеві працювати з фінансами в єдиному цифровому середовищі. У результаті підвищується зручність використання системи та зменшується необхідність ручного введення інформації.

Не менш важливою функціональною складовою є безпека даних і контроль доступу. Оскільки фінансові системи працюють із конфіденційною інформацією, вони повинні забезпечувати надійний захист даних користувачів. Для цього використовуються механізми автентифікації, шифрування даних, резервного копіювання та розмежування прав доступу. Дослідження показують, що рівень довіри користувачів до системи напряму залежить від якості реалізованих заходів безпеки [2].

Отже, функціональні можливості сучасних систем обліку доходів і витрат охоплюють автоматизований облік транзакцій, категоризацію фінансових операцій, бюджетування, аналітику, прогнозування, інтеграцію із зовнішніми сервісами та забезпечення безпеки даних. У сукупності ці функції формують комплексний інструмент управління особистими фінансами, який не лише фіксує грошові потоки, а й активно допомагає користувачеві планувати та контролювати власний бюджет.

1.3 Огляд існуючих вебзастосунків для фінансового планування

Стрімкий розвиток веброзробки та мобільних технологій сприяв появі значної кількості цифрових інструментів для управління особистими фінансами. Вебзастосунки для фінансового планування дозволяють користувачам у режимі

реального часу відстежувати витрати, формувати бюджети, ставити фінансові цілі та аналізувати свій фінансовий стан [10]. Нижче розглядаються найпоширеніші рішення, що представлені на ринку станом на 2025–2026 роки.

YNAB (You Need A Budget) – один із найпопулярніших вебзастосунків для персонального бюджетування, який базується на методології «нульового бюджету» (рис. 1.1). Його основна концепція полягає в тому, що кожна грошова одиниця, яка надходить користувачу, повинна мати чітко визначене призначення: витрати, заощадження або погашення боргів. Такий підхід дозволяє уникати «непризначених коштів» і формує у користувача більш усвідомлене ставлення до власних фінансів, підвищуючи рівень фінансової дисципліни та контролю [11].

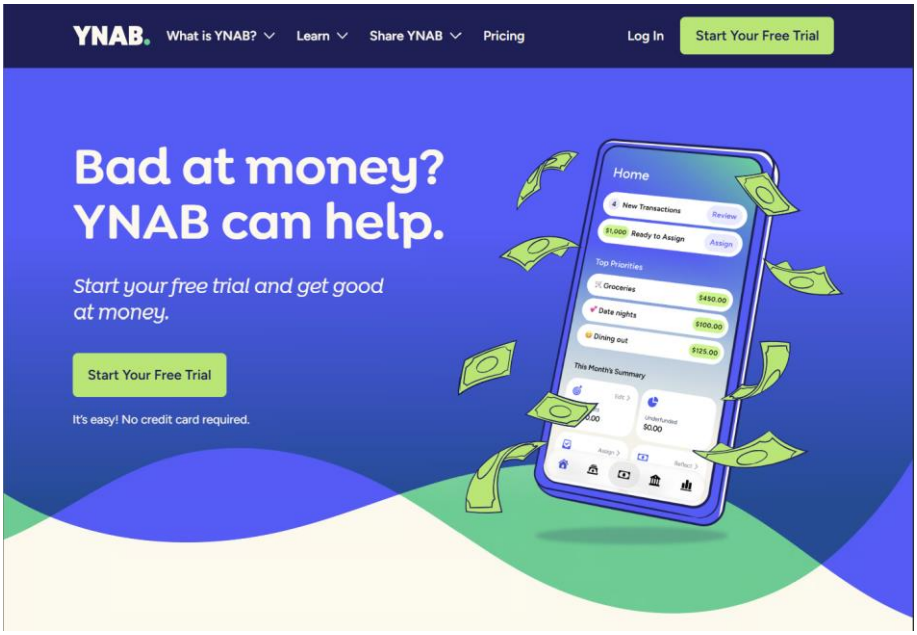


Рисунок 1.1 – YNAB

Функціонально YNAB є комплексною системою управління особистим бюджетом. Вона забезпечує синхронізацію з банківськими рахунками, кредитними картками та фінансовими сервісами, що дозволяє автоматично імпортувати транзакції та підтримувати актуальний стан фінансових даних у режимі реального часу. Окрім автоматичного імпорту, система підтримує ручне

введення операцій, що дає можливість користувачам повністю контролювати структуру обліку або працювати в офлайн-режимі.

Однією з ключових функцій YNAB є інструменти постановки фінансових цілей. Користувач може визначати як короткострокові цілі (наприклад, щомісячне накопичення або планування великих покупок), так і довгострокові (формування резервного фонду, погашення кредитів, інвестиційні цілі). Система візуалізує прогрес досягнення цих цілей, що підсилює мотиваційний ефект і сприяє регулярному дотриманню бюджету.

Окрему важливу роль відіграє функціонал управління боргами. YNAB дозволяє структурувати кредитні зобов'язання, визначати пріоритетність їх погашення та прогнозувати терміни повного закриття боргу залежно від фінансової поведінки користувача. Це робить систему ефективним інструментом не лише для обліку витрат, а й для стратегічного фінансового планування.

Значною перевагою YNAB є також освітня складова. Платформа активно використовує навчальні матеріали, інтерактивні курси та вебінари, спрямовані на формування правильних фінансових звичок. Такий підхід дозволяє користувачам не просто використовувати інструмент, а поступово змінювати власну фінансову поведінку та підхід до управління грошима.

З погляду доступності, застосунок є кросплатформним і доступний через веббраузер, а також мобільні додатки для iOS та Android. Це забезпечує постійний доступ до фінансових даних незалежно від пристрою користувача. За оцінками магазинів застосунків, YNAB має високі рейтинги – 4,8 в App Store та 4,7 у Google Play, що свідчить про високу якість і задоволеність користувачів [12].

Модель монетизації YNAB побудована на платній підписці, яка становить приблизно 109 USD на рік або 14,99 USD на місяць. Така модель дозволяє забезпечувати регулярний розвиток платформи, однак може бути обмежувальним фактором для частини користувачів.

Empower Personal Dashboard (раніше Personal Capital) – це фінансова платформа, орієнтована насамперед на комплексне управління інвестиціями та

оцінку загального фінансового стану користувача (рис. 1.2). На відміну від класичних бюджетних застосунків, основний акцент тут зроблено не на щоденному бюджетуванні, а на аналізі активів, зобов’язань та довгостроковому фінансовому плануванні [13].

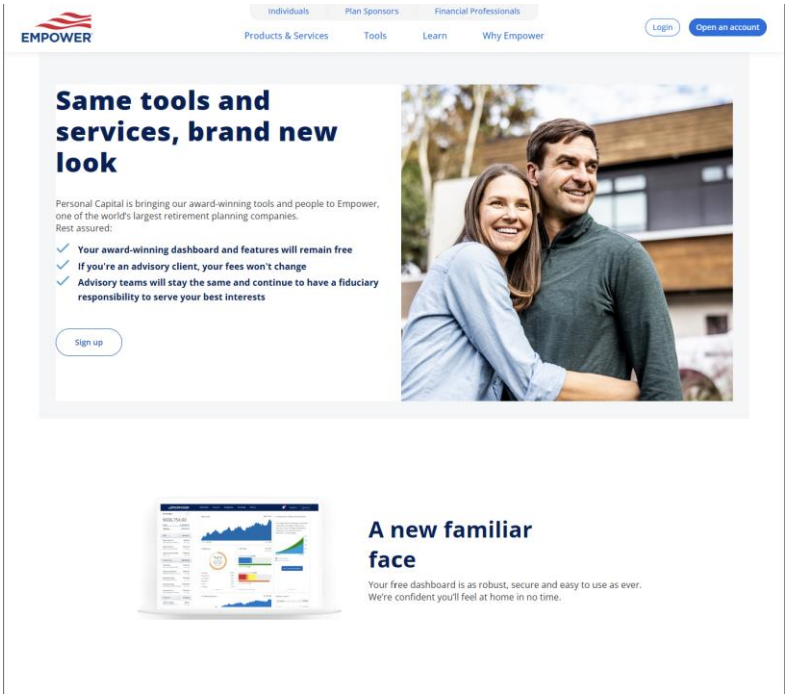


Рисунок 1.2 – Empower Personal Dashboard

Система дозволяє об’єднувати в одному інтерфейсі різні типи фінансових рахунків: банківські депозити, кредитні картки, іпотечні кредити, пенсійні рахунки (IRA, 401(k)) та інвестиційні портфелі. Завдяки цьому формується цілісне уявлення про фінансовий стан користувача, включаючи як активи, так і боргові зобов’язання. Однією з ключових функцій Empower є інструменти аналізу інвестицій. Користувач отримує можливість оцінювати структуру свого інвестиційного портфеля, рівень ризику та ефективність вкладень. Особливу увагу приділено аналізу прихованих комісій (Fee Analyzer), які можуть суттєво впливати на довгострокову дохідність інвестиційних інструментів.

Також платформа містить функції фінансового планування, зокрема інструменти прогнозування виходу на пенсію. Вони дозволяють моделювати майбутній фінансовий стан користувача з урахуванням поточних доходів,

витрат, інвестиційної дохідності та довгострокових фінансових цілей. Це робить систему більш орієнтованою на стратегічне, а не оперативне управління фінансами. Empower Personal Dashboard доступний через веббраузер, а також мобільні та планшетні пристрої, що забезпечує зручний доступ до фінансової інформації в будь-який час. Застосунок має високий рейтинг користувачів (близько 4,8 в App Store), що свідчить про його стабільність і якість реалізації.

Водночас варто зазначити, що функціональність класичного бюджетування в Empower є обмеженою. Основний фокус системи спрямований на інвестиційний аналіз і управління капіталом, тому вона менш ефективна для користувачів, яким необхідний детальний контроль щоденних витрат.

PocketGuard (рис. 1.3) є спрощеним фінансовим застосунком, орієнтованим на швидке визначення доступного залишку коштів користувача. Основна концепція сервісу полягає у відповіді на запитання «скільки я можу витратити зараз», що робить його особливо зручним для щоденного контролю витрат [14].

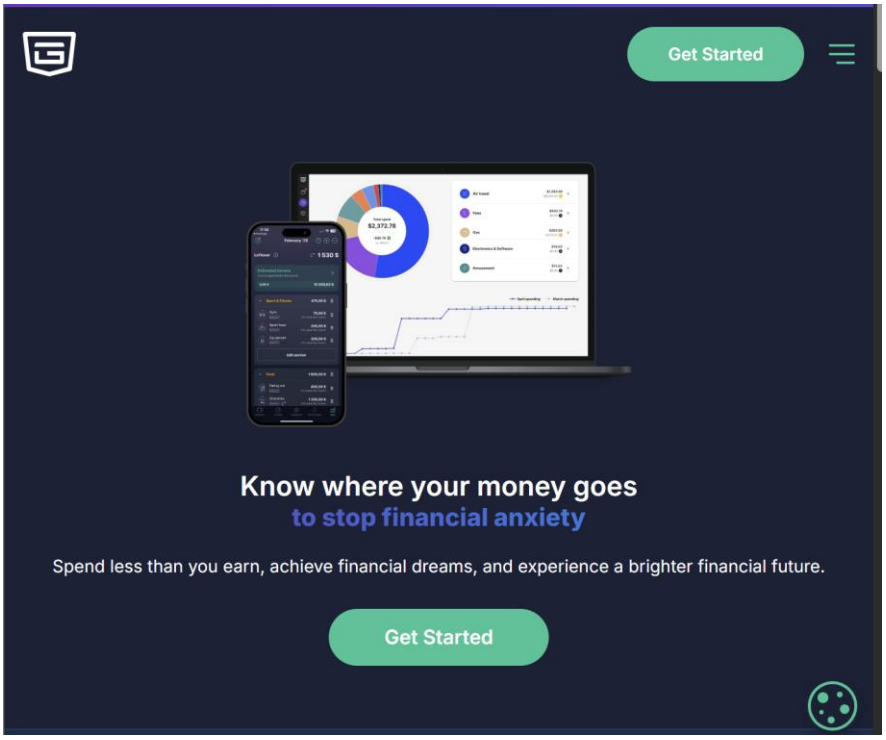


Рисунок 1.3 – PocketGuard

Після підключення банківських рахунків система автоматично аналізує доходи, регулярні витрати, рахунки та фінансові цілі, після чого розраховує доступний бюджет. Це дозволяє користувачеві оперативно оцінювати свої фінансові можливості без складного аналізу даних.

Однією з ключових функцій PocketGuard є механізм контролю темпу витрат. Інструмент «Pace» відстежує швидкість витрачання коштів і попереджає користувача у випадку перевищення запланованого бюджету. Це сприяє більш дисциплінованому фінансовому поведінковому контролю та запобігає надмірним витратам.

Система також забезпечує базове бюджетування та категоризацію витрат, однак не пропонує розширених інвестиційних або кредитних інструментів. Це робить PocketGuard більш орієнтованим на користувачів, які потребують простого та швидкого контролю щоденних фінансів.

Рейтинг застосунку становить близько 4,5 в App Store та 4,0 у Google Play, що свідчить про загальну задоволеність користувачів. Монетизація реалізована через підписку, яка становить приблизно 6,25 USD на місяць при річній оплаті.

Spendee (рис. 1.4) є сучасним веб- та мобільним застосунком для управління особистими фінансами, який вирізняється інтуїтивно зрозумілим інтерфейсом та орієнтацією на користувачів із різними фінансовими потребами. Зокрема, система активно використовується тревелерами, фрилансерами та користувачами, які працюють із кількома валютами, що робить її зручною для міжнародних фінансових операцій та нерегулярних доходів [15].

Однією з ключових особливостей Spendee є підтримка мультивалютності. Користувач може одночасно вести облік фінансів у різних валютах, що автоматично конвертуються та відображаються у зведеній аналітиці. Це особливо важливо для осіб, які подорожують або отримують дохід із різних країн, оскільки дозволяє уникнути помилок при ручному перерахунку валют і забезпечує більш точну фінансову картину.

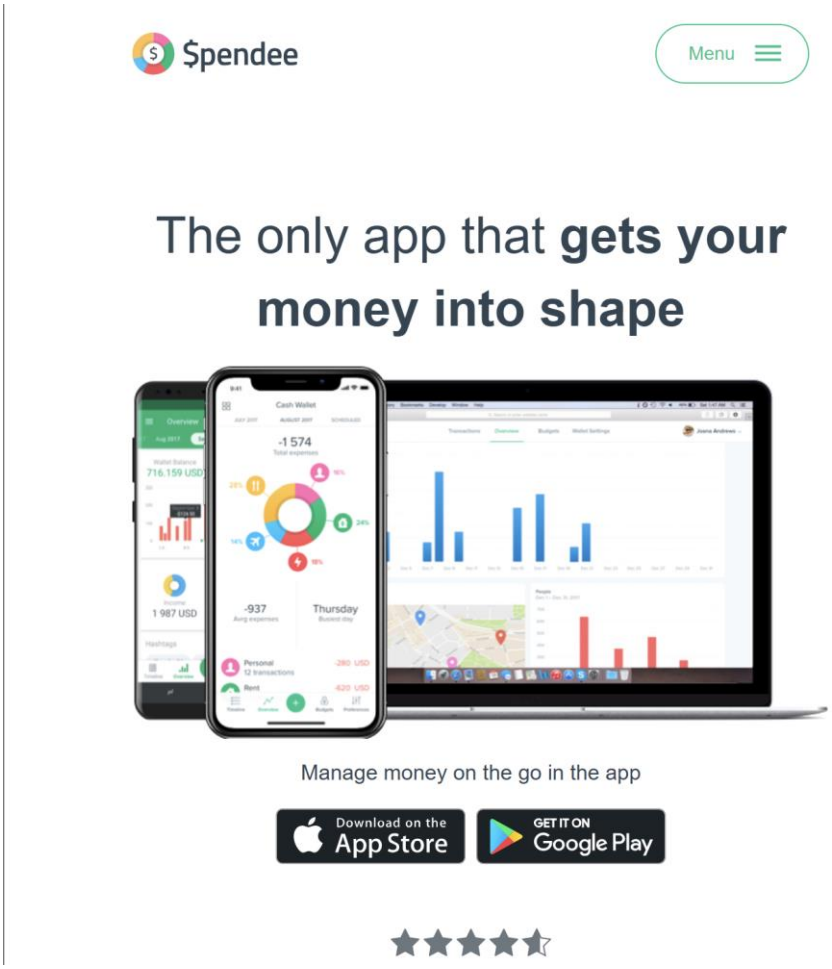


Рисунок 1.4 – Spendee

Важливою функціональною можливістю є спільне бюджетування. Spendee дозволяє створювати спільні гаманці, які можуть використовувати кілька користувачів одночасно. Це актуально для сімей, пар або груп осіб, які спільно ведуть витрати, наприклад під час подорожей або спільного проживання. Система забезпечує синхронізацію даних у реальному часі, що дозволяє всім учасникам бачити актуальний стан бюджету, а також отримувати сповіщення про транзакції або перевищення лімітів.

Окремо слід відзначити функціонал категоризації витрат і аналітики. Користувач може створювати власні категорії або використовувати стандартні шаблони для класифікації витрат і доходів. На основі цих даних система формує графіки та звіти, які дозволяють оцінювати структуру витрат і визначати фінансові звички користувача.

Spendee працює за моделлю freemium, що передбачає наявність безкоштовної версії з базовим функціоналом та платної підписки з розширеними можливостями. Преміум-версія включає доступ до розширеної аналітики, синхронізації з банками та додаткових інструментів бюджетування. Вартість підписки починається приблизно від 2,99 USD на місяць, що робить сервіс доступним для широкого кола користувачів.

Загалом Spendee можна охарактеризувати як гнучку систему для персонального та спільного фінансового управління, яка поєднує простоту використання з базовими аналітичними можливостями, орієнтовану на мобільний спосіб життя.

Monarch Money (рис. 1.5) є сучасною фінансовою платформою, яка була запущена у 2020 році та швидко здобула популярність серед фінансових консультантів, індивідуальних інвесторів і домогосподарств, що потребують комплексного контролю фінансів [16]. Основною ідеєю системи є надання користувачеві єдиного централізованого огляду всіх фінансових активів і зобов’язань.

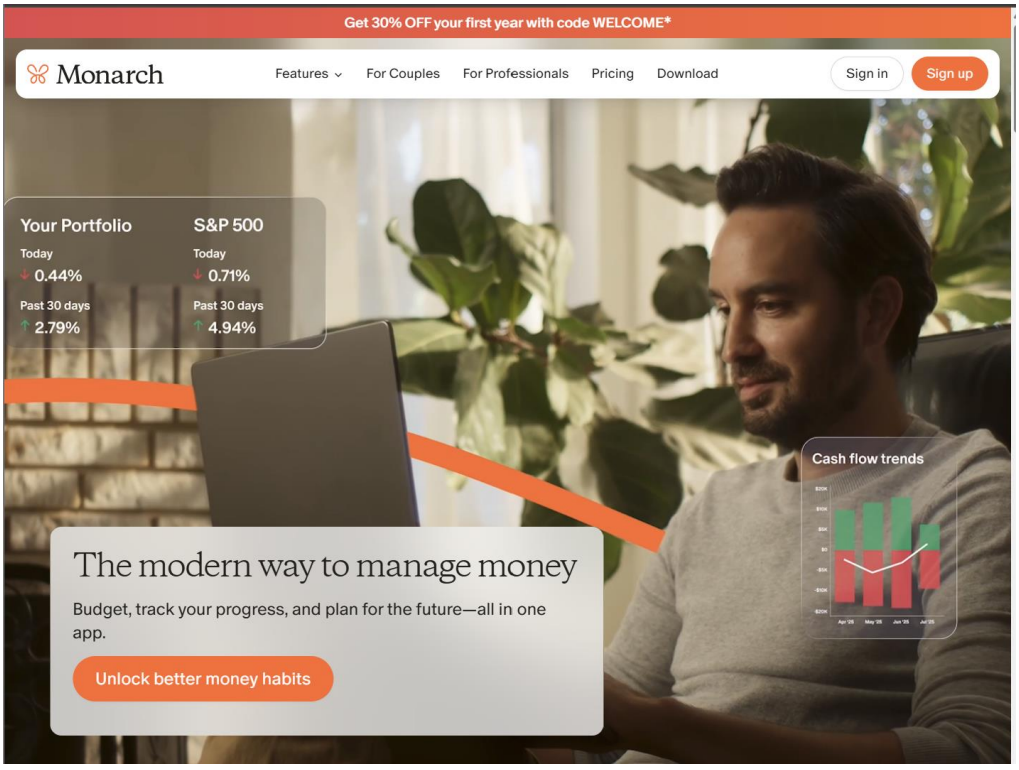


Рисунок 1.5 – Monarch Money

Функціонально Monarch Money дозволяє об’єднувати в одному інтерфейсі різні фінансові інструменти, включаючи банківські рахунки, кредитні картки, інвестиційні портфелі, кредити та інші фінансові зобов’язання. Це забезпечує повну прозорість фінансового стану користувача, включаючи розрахунок чистого капіталу (net worth) та аналіз грошових потоків.

Однією з ключових переваг платформи є розширена система бюджетування. Користувачі можуть створювати гнучкі бюджетні категорії, налаштовувати індивідуальні фінансові правила та відстежувати витрати в реальному часі. Система також підтримує сповіщення про перевищення бюджету або незвичні фінансові операції, що підвищує рівень фінансового контролю. Особливу увагу в Monarch приділено спільному фінансовому плануванню. Платформа дозволяє кільком користувачам (наприклад, подружжю або членам родини) вести спільний бюджет, що є важливим для домогосподарств із загальними витратами. Це сприяє більш прозорому та узгодженому управлінню фінансами в межах сім’ї або партнерства.

Monarch Money доступний через веббраузер, а також мобільні застосунки для iOS та Android. Система працює за підписною моделлю, вартість якої становить приблизно 99,99 USD на рік. Незважаючи на платний доступ, платформа позиціонується як преміальне рішення для користувачів, які потребують глибокого фінансового аналізу та зручного інтерфейсу.

Quicken Simplifi (рис. 1.6) є хмарною версією відомої фінансової екосистеми Quicken, орієнтованою переважно на молодшу та цифрово активну аудиторію [17]. Основною метою платформи є спрощення фінансового управління без втрати функціональної глибини, характерної для класичних фінансових систем.

Система пропонує динамічне бюджетування, яке автоматично адаптується до змін доходів і витрат користувача. На відміну від статичних бюджетних моделей, Simplifi враховує реальні фінансові потоки та дозволяє оперативно коригувати фінансові плани. Це робить систему більш гнучкою та придатною для користувачів із нестабільними доходами.

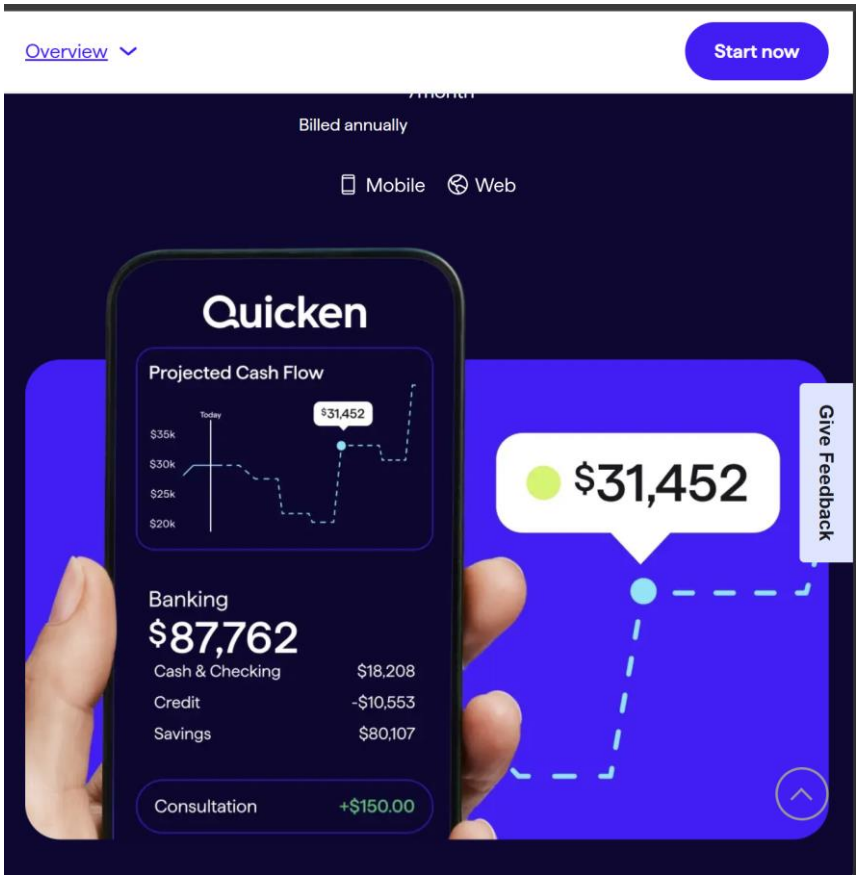


Рисунок 1.6 – Quicken Simplifi

Окремою функцією є прогнозування грошових потоків, яке дозволяє оцінювати майбутній фінансовий стан на основі поточних витрат і доходів. Користувач може бачити потенційний баланс у майбутні періоди, що допомагає уникати дефіциту коштів і краще планувати великі витрати.

Також Quicken Simplifi включає інструменти для відстеження інвестиційного портфеля та базового пенсійного планування. Це дозволяє користувачам не лише контролювати щоденні витрати, але й формувати довгострокову фінансову стратегію.

На відміну від деяких конкурентів, Quicken Simplifi надає користувачам додаткову підтримку через чат і телефон, що підвищує рівень сервісу та зручність використання. Вартість підписки становить приблизно 40 USD на рік (з можливими акційними знижками для нових користувачів).

Загалом Quicken Simplifi можна охарактеризувати як збалансоване рішення, яке поєднує простоту використання з розширеними функціями бюджетування та прогнозування фінансів.

Аналіз наведених застосунків дозволяє виділити ключові тенденції розвитку ринку фінансових вебзастосунків (табл. 1.1). По-перше, спостерігається зміщення від суто безкоштовних рішень (зокрема, закриття Mint у 2024 р.) до підписних моделей із розширеним функціоналом. По-друге, зростає попит на інструменти спільного бюджетування для домогосподарств і пар. По-третє, інтеграція з банківськими рахунками та автоматична категоризація транзакцій стали стандартом галузі. Жоден із розглянутих застосунків не локалізований для українського ринку та не інтегрований із вітчизняними банківськими системами, що визначає нішу для розробки відповідного рішення.

Таблиця 1.1 – Порівняльна характеристика вебзастосунків для фінансового планування

Застосунок	**Тип доступу**	**Веб-версія**	**Мобільний**	**Бюджетування**	**Інвестиційні інструменти**	**Мультивалютність**	**Вартість (USD/рік)**
YNAB	SaaS	✓	✓	Нульовий бюджет	–	–	109
Empower	SaaS	✓	✓	Базове	✓	–	Безкоштовно
PocketGuard	SaaS	–	✓	Автоматичне	–	–	~75
Spendee	SaaS	✓	✓	Категорійне	–	✓	~36
Monarch Money	SaaS	✓	✓	Гнучке	✓	–	99,99
Quicken Simplifi	SaaS	✓	✓	Динамічне	✓	–	~40–60

Отже, аналіз сучасних вебзастосунків для фінансового планування показав, що ринок персональних фінансових систем активно розвивається та орієнтується на автоматизацію обліку, інтеграцію з банківськими сервісами, аналітику витрат і прогнозування бюджету. Найбільш поширеними є SaaS-

рішення з підписною моделлю, які забезпечують користувачам доступ до розширених функцій бюджетування, візуалізації даних та фінансового планування.

Водночас встановлено, що більшість розглянутих систем не адаптовані до українського ринку, не підтримують локальні банківські сервіси та мають обмежену персоналізацію під регіональні фінансові умови. Це формує потребу у створенні більш гнучкого та локалізованого вебзастосунку, який поєднуватиме базові функції обліку фінансів із аналітикою та прогнозуванням бюджету.

2 ПРОЄКТУВАННЯ ВЕБСИСТЕМИ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛЬНИМИ ФІНАНСАМИ

2.1 Аналіз вимог до вебсистеми

Першим етапом проєктування вебсистеми для управління персональними фінансами є формування, аналіз та структуризація вимог до майбутнього програмного продукту. Саме на цьому етапі визначається функціональна повнота системи, її обмеження, а також ключові параметри якості, які впливають на ефективність використання вебзастосунку.

У межах даного дослідження вимоги поділено на дві основні групи: функціональні вимоги, що описують перелік можливостей системи, та нефункціональні вимоги, які визначають її якісні характеристики, продуктивність, безпеку та зручність використання.

Функціональні вимоги (табл. 2.1) визначають набір дій, які вебсистема повинна виконувати для забезпечення повноцінного управління персональними фінансами користувача. Вони охоплюють увесь життєвий цикл фінансових операцій – від їх внесення до аналітичної обробки та прогнозування.

Узагальнюючи функціональні вимоги, можна відзначити, що система повинна забезпечувати повний цикл управління персональними фінансами – від первинного обліку транзакцій до формування аналітичних звітів і прогнозів. Особливу роль відіграють функції бюджетування та аналізу витрат, які забезпечують користувачеві інструменти фінансового контролю та планування.

Нефункціональні вимоги (табл. 2.2) визначають якісні характеристики роботи вебсистеми, що впливають на її продуктивність, надійність, безпеку та зручність використання. Вони є критичними для забезпечення стабільної та коректної роботи фінансового застосунку.

Таблиця 2.1 – Функціональні вимоги до вебсистеми

Код	Найменування вимоги	Опис	Пріоритет
FR-01	Облік транзакцій	Додавання, редагування, перегляд і видалення доходів та витрат користувача	Високий
FR-02	Категоризація операцій	Розподіл фінансових операцій за категоріями (продукти, транспорт, житло тощо)	Високий
FR-03	Розрахунок балансу	Автоматичне обчислення доходів, витрат і загального фінансового балансу	Високий
FR-04	Аналіз витрат	Побудова аналітики витрат за категоріями та періодами	Високий
FR-05	Бюджетування	Встановлення та контроль місячних фінансових лімітів	Середній
FR-06	Прогнозування витрат	Оцінка майбутніх витрат на основі історичних даних	Середній
FR-07	Розрахунок заощаджень	Визначення частки заощаджень від загального доходу	Низький
FR-08	Сповіщення	Повідомлення користувача про перевищення бюджету або аномальні витрати	Низький

Таблиця 2.2 – Нефункціональні вимоги до вебсистеми

Код	Категорія	Вимога
NFR-01	Продуктивність	Час відгуку інтерфейсу не перевищує 2 секунд при обробці до 10 000 транзакцій
NFR-02	Зручність використання	Інтерфейс адаптивний (mobile-first), світла кольорова тема
NFR-03	Надійність	Забезпечення цілісності даних через транзакційність та використання зовнішніх ключів
NFR-04	Переносимість	Використання локальної бази даних SQLite без залежності від зовнішніх сервісів
NFR-05	Безпека	Захист від SQL-ін’єкцій, перевірка та валідація всіх користувацьких даних
NFR-06	Супровідність	Модульна архітектура з чітким поділом на шари (frontend, backend, data layer)

Згідно з визначеними вимогами, система повинна забезпечувати високу швидкодію навіть при значних обсягах фінансових даних. Окремо підкреслюється важливість інтуїтивного інтерфейсу, який дозволяє користувачеві швидко взаємодіяти з функціоналом без додаткового навчання.

Критично важливим є також аспект безпеки, оскільки система працює з фінансовими даними користувача. Використання параметризованих запитів, перевірки введених даних та структурованого доступу до бази даних мінімізує ризики несанкціонованого доступу та втрати інформації.

Основним актором розроблюваної вебсистеми є Користувач, який взаємодіє з усіма функціональними модулями системи та здійснює повний цикл управління персональними фінансами.

Основні сценарії використання системи включають (рис. 2.1):

- керування фінансовими транзакціями (додавання, редагування, видалення);
- перегляд фінансового дашборду;
- аналіз структури витрат;
- налаштування бюджетів за категоріями;
- перегляд прогнозу фінансового стану;
- управління категоріями доходів і витрат.

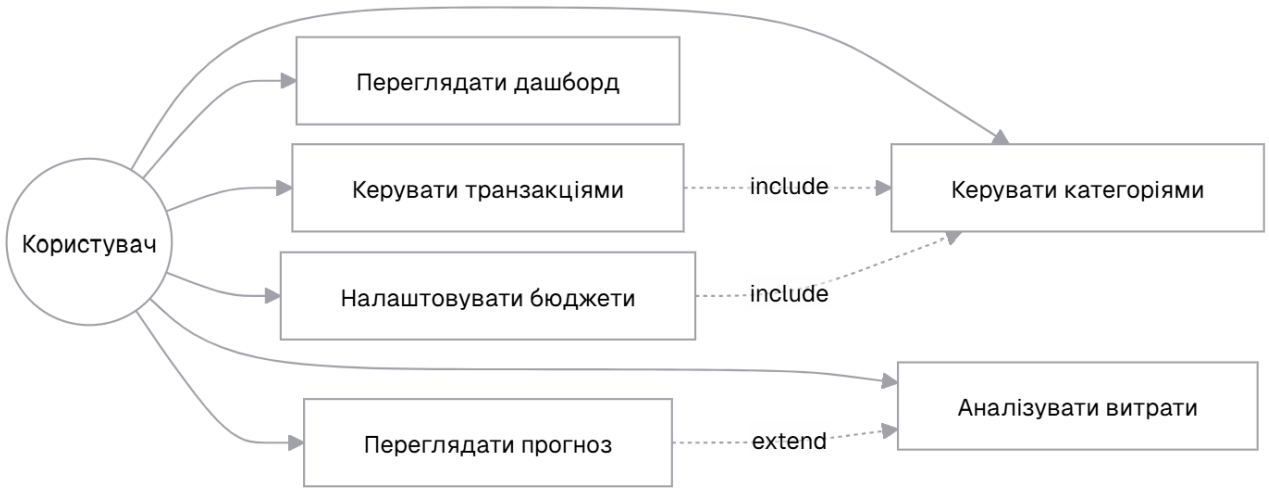


Рисунок 2.1 – Основні сценарії використання системи

Аналіз сценаріїв використання показує, що система має модульну структуру, де кожен функціональний блок відповідає окремому аспекту фінансового менеджменту. Це дозволяє забезпечити гнучкість системи та її масштабованість у майбутньому.

У результаті аналізу вимог визначено, що розроблювана вебсистема повинна поєднувати функції обліку, аналітики, бюджетування та прогнозування фінансових даних. Встановлено ключові функціональні можливості та

нефункціональні характеристики, які забезпечують ефективність, безпеку та зручність використання системи. Це створює основу для подальшого проєктування архітектури та реалізації вебзастосунку.

2.2 Проєктування архітектури вебзастосунку

Проєктування архітектури вебзастосунку є ключовим етапом розробки системи управління персональними фінансами, оскільки саме на цьому рівні визначаються принципи організації програмного коду, взаємодія між компонентами та спосіб обробки даних. У межах даної роботи вебсистема реалізується на основі сучасного фреймворку Next.js (App Router), що дозволяє поєднувати серверну та клієнтську логіку в єдиному середовищі та мінімізувати потребу у створенні окремого REST API.

Архітектура вебзастосунку (табл. 2.3) побудована за багаторівневим (layered) принципом, що забезпечує чіткий поділ відповідальності між компонентами системи та спрощує її супровід і масштабування. В основі підходу лежить розділення логіки на рівні представлення, застосунку, доступу до даних та зберігання інформації.

Таблиця 2.3 – Рівні архітектури вебзастосунку та їх призначення

Рівень	Компоненти	Відповідальність
Рівень представлення	сторінки (page.tsx), UI-компоненти React	відображення інтерфейсу та взаємодія з користувачем
Рівень застосунку	Server Actions (actions.ts)	обробка бізнес-логіки, валідація та координація операцій
Рівень доступу до даних	queries.ts, db.ts	формування SQL-запитів та робота з моделлю даних
Рівень зберігання	SQLite (finance.db)	фізичне збереження фінансової інформації

Рівень представлення у проєктованій архітектурі відповідає за формування користувацького інтерфейсу вебзастосунку та реалізується через сторінки (page.tsx) і UI-компоненти React. Його основною функцією є відображення даних

у зручному для користувача вигляді, а також забезпечення інтерактивної взаємодії користувача із системою.

Рівень застосунку формує логіку обробки бізнес-процесів системи та представлений механізмом Server Actions (actions.ts). На цьому рівні передбачається обробка користувацьких запитів, виконання перевірки та валідації вхідних даних, а також координація взаємодії між інтерфейсним рівнем і рівнем доступу до даних.

Рівень доступу до даних включає модулі queries.ts та db.ts, які забезпечують формування SQL-запитів і організацію взаємодії з базою даних. Його функціональне призначення полягає у виконанні операцій отримання, збереження та модифікації даних відповідно до запитів, що надходять від рівня застосунку.

Рівень зберігання представлений локальною базою даних SQLite (finance.db) і забезпечує фізичне збереження всієї фінансової інформації користувача, включаючи транзакції, категорії витрат та бюджетні показники. Використання локальної бази даних передбачає автономність системи та відсутність залежності від зовнішніх сервісів зберігання.

Архітектурна структура вебзастосунку представлена у вигляді компонентної діаграми (рис. 2.2), яка відображає загальну взаємодію між клієнтською частиною, серверною логікою та рівнем збереження даних.

Згідно з проєктною структурою, клієнтська частина представлена браузером користувача, у якому відображаються інтерфейсні компоненти та ініціюються користувацькі дії. Ці дії передбачено передавати до серверної частини Next.js, де вони обробляються на рівні Server Components та Server Actions.

Рівень застосунку виконує функції координації логіки системи, включаючи валідацію даних та організацію звернень до рівня доступу до даних. Рівень доступу до даних, у свою чергу, передбачає використання окремих модулів для формування SQL-запитів та взаємодії з базою даних SQLite, яка виступає як основне сховище фінансової інформації.

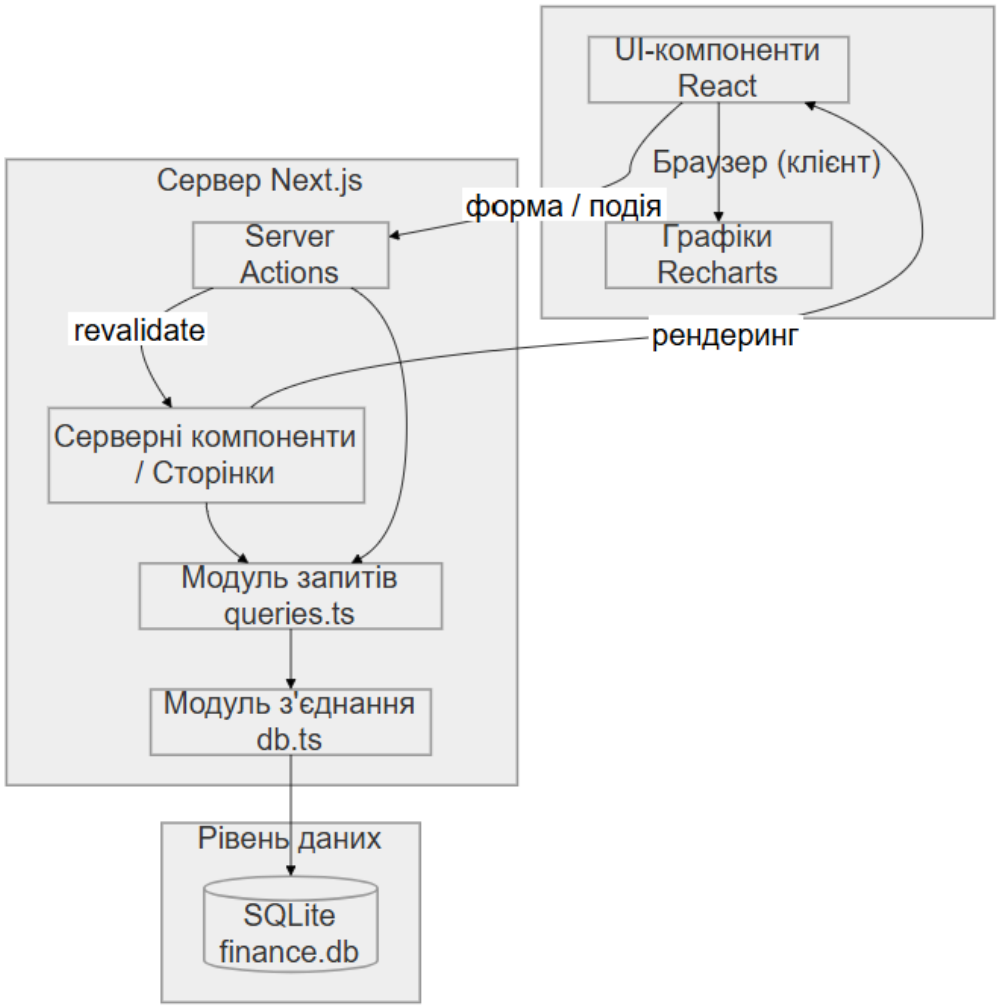


Рисунок 2.2 – Діаграма компонентів вебсистеми

Для деталізації логіки взаємодії компонентів системи у проєктуванні розглянуто типовий сценарій додавання фінансової транзакції (рис. 2.3). Даний сценарій відображає передбачену послідовність взаємодії між рівнями системи без прив’язки до реалізованого коду.

Процес передбачає, що користувач вводить дані транзакції через інтерфейс форми, після чого ці дані передаються до серверного рівня у вигляді структурованого запиту. На рівні Server Actions здійснюється перевірка коректності отриманих даних відповідно до визначених правил валідації.

Після цього формується запит до рівня доступу до даних, де передбачено виконання SQL-операції вставки нового запису у таблицю транзакцій бази даних. У відповідь система отримує результат операції, який у подальшому

використовується для оновлення стану інтерфейсу користувача через механізм повторного отримання даних.

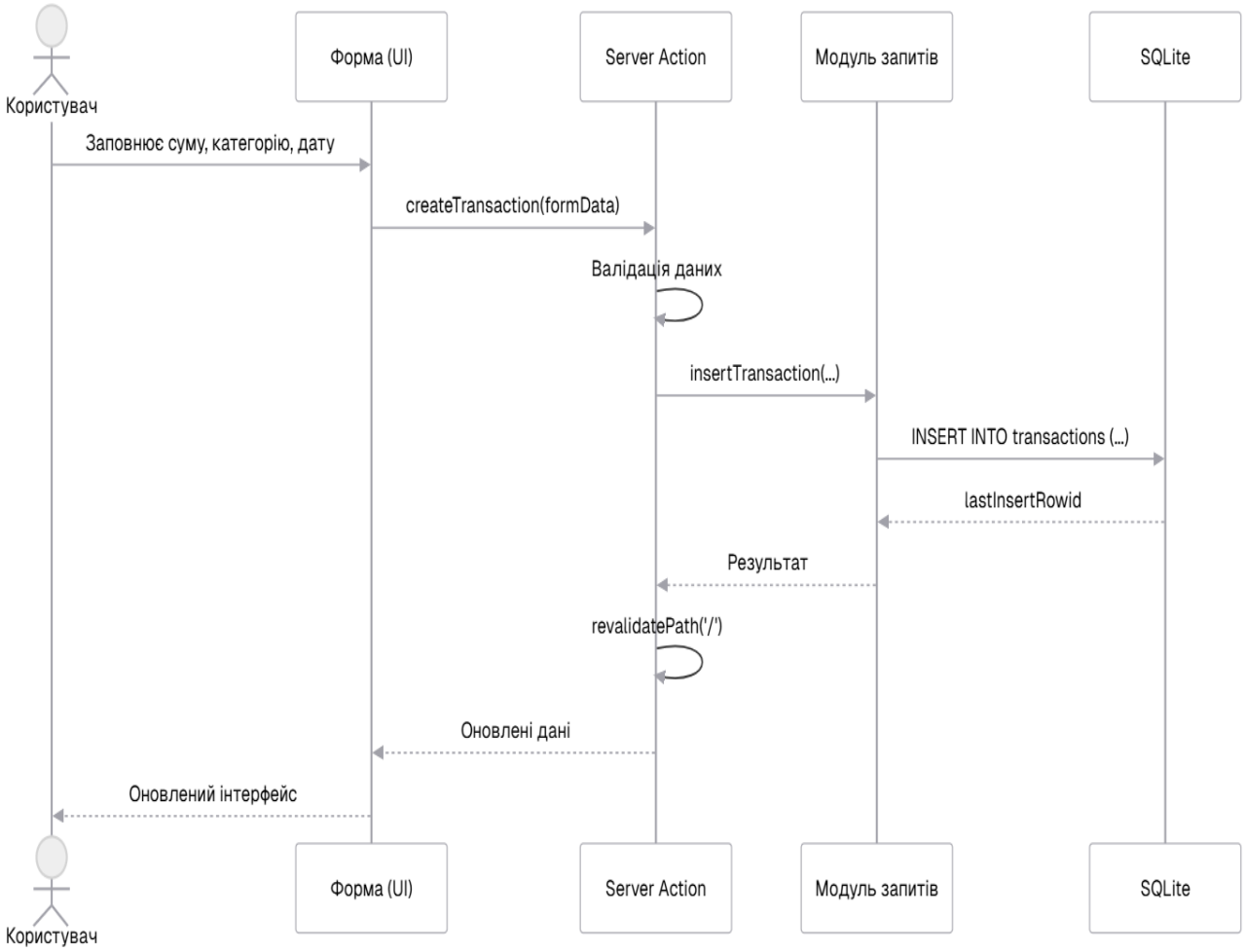


Рисунок 2.3 – Діаграма послідовності операції додавання транзакції

Тобто у проєктній моделі забезпечується чіткий поділ відповідальності між рівнями системи, що дозволяє зменшити зв’язність компонентів і спростити подальшу реалізацію вебзастосунку.

Отже, запропонована архітектура вебзастосунку формує структуровану багаторівневу модель системи, яка забезпечує логічне розділення інтерфейсу, бізнес-логіки та рівня зберігання даних. Це створює основу для подальшого етапу проєктування – розробки алгоритмів та реалізації програмного забезпечення.

2.3 Проєктування бази даних для фінансових операцій

Проєктування бази даних є одним із ключових етапів розробки вебзастосунку для управління персональними фінансами, оскільки саме структура збереження даних визначає ефективність подальшої обробки, аналітики та масштабування системи. Основною метою даного етапу є створення логічно узгодженої реляційної моделі, яка забезпечує цілісність фінансових даних користувача, мінімізує надлишковість інформації та підтримує швидкий доступ до операцій обліку, аналізу та бюджетування.

У межах проєктування використано реляційний підхід, що є доцільним для даного класу систем, оскільки фінансові операції мають чітко визначені зв’язки між сутностями, а також потребують забезпечення цілісності та транзакційності. База даних містить три основні сутності: категорії витрат і доходів, транзакції та бюджети, які утворюють основу для подальшого аналізу фінансової інформації.

Концептуальна модель бази даних відображає загальну структуру предметної області та взаємозв’язки між основними сутностями системи. У межах розробленої системи виділено три ключові сутності: categories, transactions та budgets. Сутність категорій використовується для класифікації фінансових операцій, транзакції відображають безпосередні доходи та витрати користувача, а бюджети визначають фінансові обмеження для контролю витрат у межах певного періоду.

Зв’язки між сутностями реалізують наступну логіку: кожна транзакція обов’язково належить до певної категорії, тоді як бюджет встановлюється також у прив’язці до категорії, що дозволяє здійснювати контроль витрат у розрізі окремих фінансових груп. Така структура забезпечує узгодженість даних і дозволяє виконувати аналітичні запити щодо витрат та доходів у різних розрізах.

Логічна структура бази даних представлена у вигляді ER-діаграми (рис. 2.4), яка ілюструє зв’язки між основними сутностями системи.

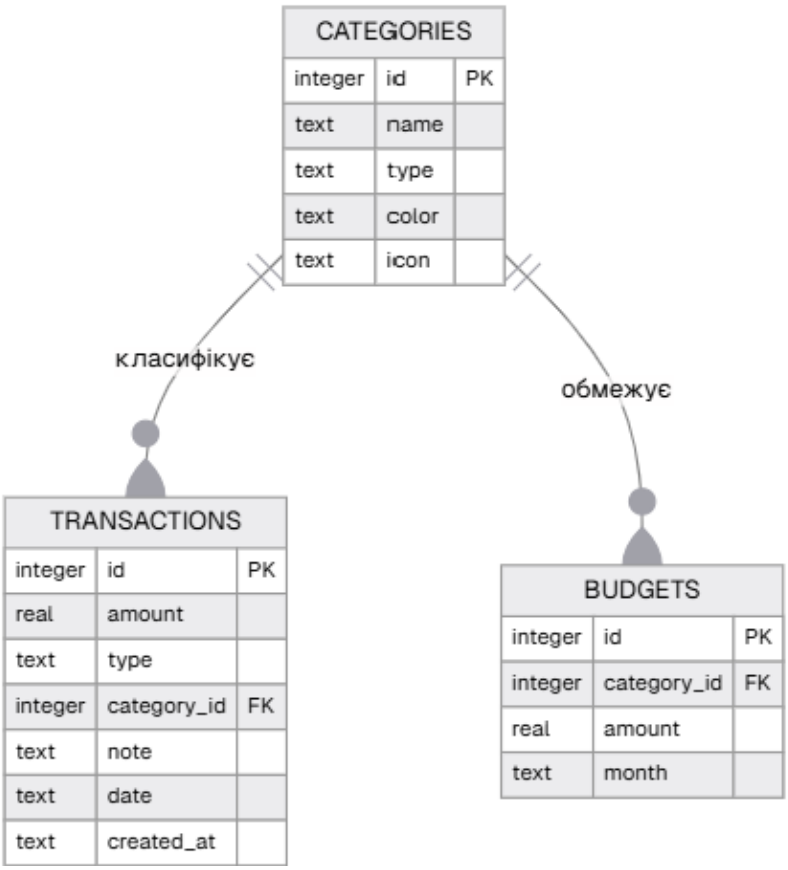


Рисунок 2.4 – ER-діаграма бази даних фінансових операцій

Фізична реалізація бази даних передбачає використання трьох основних таблиць, кожна з яких відповідає окремій сутності предметної області та має чітко визначену структуру полів, типів даних і обмежень цілісності.

Таблиця categories (табл. 2.4) призначена для зберігання інформації про фінансові категорії, які використовуються для групування транзакцій. Вона дозволяє розділяти доходи та витрати, а також забезпечує візуальне кодування через кольори та іконки, що надалі використовується в інтерфейсі користувача для аналітичної візуалізації.

Таблиця transactions (табл. 2.5) є центральною в системі, оскільки саме вона містить усі фінансові операції користувача. Кожен запис у цій таблиці фіксує суму операції, її тип, дату та зв’язок із відповідною категорією. Додатково передбачено поле примітки для розширення контексту фінансових записів.

Таблиця 2.4 – Структура таблиці categories

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор категорії
name	TEXT	NOT NULL	Назва категорії
type	TEXT	NOT NULL, CHECK (income/expense)	Тип категорії (дохід або витрата)
color	TEXT	NOT NULL	Колір для візуалізації в інтерфейсі
icon	TEXT	–	Іконка категорії

Таблиця 2.5 – Структура таблиці transactions

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор транзакції
amount	REAL	NOT NULL	Сума фінансової операції
type	TEXT	NOT NULL, CHECK (income/expense)	Тип операції
category_id	INTEGER	FK – categories(id)	Посилання на категорію
note	TEXT	–	Додаткова інформація
date	TEXT	NOT NULL	Дата операції (ISO формат)
created_at	TEXT	DEFAULT CURRENT_TIMESTAMP	Час створення запису

Таблиця budgets (табл. 2.6) використовується для реалізації механізму фінансового планування та контролю витрат. Вона дозволяє задавати граничні значення витрат для кожної категорії в межах певного місяця, що забезпечує базову функціональність бюджетування у системі.

Таблиця 2.6 – Структура таблиці budgets

Поле	Тип	Обмеження	Призначення
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор бюджету
category_id	INTEGER	FK – categories(id)	Категорія бюджету
amount	REAL	NOT NULL	Встановлений ліміт витрат
month	TEXT	NOT NULL	Місяць у форматі YYYY-MM

Для забезпечення надійності та ефективності роботи бази даних у процесі проєктування передбачено низку механізмів контролю цілісності та оптимізації продуктивності. Зокрема, використання зовнішніх ключів забезпечує підтримку

зв’язків між таблицями та запобігає виникненню неконсистентних даних при видаленні або зміні пов’язаних записів.

Додатково застосовано обмеження CHECK, які гарантують коректність значень типу транзакції та категорії, виключаючи можливість внесення некоректних фінансових даних. Для підвищення швидкодії виконання аналітичних запитів передбачено індексацію ключових полів, таких як date, category_id та month, що є критично важливим при формуванні звітів та графічної аналітики.

Окрему увагу приділено налаштуванню режиму роботи бази даних у режимі WAL (Write-Ahead Logging), що дозволяє підвищити продуктивність паралельного читання та запису даних. Крім того, усі операції модифікації даних виконуються в межах транзакцій, що забезпечує атомарність і цілісність змін навіть у випадку помилок виконання.

Таким чином, розроблена структура бази даних забезпечує узгоджене, ефективне та масштабоване зберігання фінансової інформації, створюючи надійну основу для реалізації функціональних можливостей вебзастосунку, зокрема обліку транзакцій, аналітики витрат та бюджетного контролю.

2.4 Проєктування модулів аналізу витрат та бюджетування

Проєктування модулів аналізу витрат та бюджетування є важливою складовою архітектури вебзастосунку, оскільки саме ці компоненти забезпечують перетворення первинних фінансових даних у аналітичну інформацію, придатну для прийняття рішень користувачем. У межах системи ці модулі реалізують обробку транзакційних даних, формування агрегованих показників, побудову візуалізацій, а також виконання розрахунків бюджетних обмежень і прогнозування фінансових витрат.

Загальна логіка побудови цих модулів базується на принципі поділу відповідальності: модуль аналізу витрат відповідає за статистичну обробку

історичних даних, тоді як модуль бюджетування та прогнозування – за контроль фінансових лімітів і формування майбутніх значень на основі наявних тенденцій.

Модуль аналізу витрат призначений для формування узагальнених фінансових показників на основі транзакцій користувача. Його робота ґрунтується на використанні агрегувальних SQL-запитів, які дозволяють перетворювати первинні записи про доходи та витрати у структуровані аналітичні зрізи.

Основними показниками (табл. 2.7), що формуються в межах модуля, є сумарні доходи та витрати, структура витрат за категоріями, динаміка змін у часі, а також показники балансу та рівня заощаджень. Кожен із цих показників має відповідний метод обчислення та форму візуального представлення, що дозволяє користувачу інтерпретувати фінансову інформацію у зручному графічному вигляді.

Таблиця 2.7 – Аналітичні показники модуля аналізу витрат

Показник	**Метод обчислення**	**Візуалізація**
Сумарні доходи / витрати	`SUM(amount)` з групуванням за типом	Картки-показники
Структура витрат	`SUM(amount)` GROUP BY category	Кругова діаграма
Динаміка за місяцями	`SUM(amount)` GROUP BY `substr(date,1,7)`	Стовпчикова діаграма
Динаміка балансу	Доходи − Витрати за місяць	Лінійний графік
Норма заощаджень	$S = \frac{Income - Expense}{Income} \cdot 100%$	Картка-показник

У наведеній таблиці представлено основні аналітичні показники модуля аналізу витрат, а також методи їх обчислення та способи візуалізації. Зокрема, сумарні доходи та витрати визначаються за допомогою агрегувальної функції SUM(amount) з групуванням за типом операції та відображаються у вигляді карток-показників. Структура витрат формується шляхом групування даних за категоріями, що дозволяє представити розподіл витрат у вигляді кругової діаграми.

Динаміка витрат за місяцями обчислюється через групування транзакцій за часовим проміжком (місяцем) і візуалізується у вигляді стовпчикової діаграми,

що дозволяє оцінити зміни фінансової активності в часі. Динаміка балансу визначається як різниця між доходами та витратами за кожен місяць і відображається лінійним графіком, що демонструє загальну фінансову тенденцію. Окремо розраховується норма заощаджень за формулою відношення різниці доходів і витрат до доходів у відсотках, яка подається у вигляді узагальненого карткового показника для швидкої оцінки фінансової ефективності користувача.

Реалізаційна логіка модуля передбачає послідовну обробку даних, починаючи з отримання транзакцій із бази даних і завершуючи формуванням візуальних компонентів інтерфейсу. Спочатку виконується вибірка даних за заданий період, після чого застосовується відповідний тип агрегації – за категоріями або часовими інтервалами. Далі обчислюються похідні показники, зокрема частки витрат і баланс за період, які передаються до компонентів візуалізації. Завершальним етапом є відображення результатів у вигляді графіків та аналітичних карток.

Модуль бюджетування та прогнозування забезпечує контроль фінансових витрат користувача в межах встановлених лімітів, а також формує прогнозні значення на основі історичних даних. Його функціональність умовно поділяється на два взаємопов’язані підмодулі: бюджетний контроль та аналітичне прогнозування.

У межах підсистеми бюджетування для кожної категорії витрат встановлюється місячний ліміт, який порівнюється з фактичними витратами користувача. Ступінь використання бюджету визначається як відсоткове відношення витрачених коштів до встановленого ліміту. Такий підхід дозволяє не лише контролювати поточний стан бюджету, але й оперативно виявляти перевищення допустимих витрат:

$$U = \frac{Spent}{Limit} * 100\%.$$

Якщо U > 100%, система позначає категорію як перевитрачену.

Для прогнозу майбутніх витрат застосовано метод лінійної регресії (метод найменших квадратів). За наявними точками (x_i, y_i), де x_i – порядковий номер місяця, а y_i – сума витрат, обчислюються коефіцієнти прямої $y = a + bx$:

$$b = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}, a = \frac{\sum y_i - b\sum x_i}{n}.$$

Прогнозоване значення на наступний місяць обчислюється підстановкою x_{n+1} у рівняння регресії. Логіку модуля прогнозування показано на рисунку 2.5.

Діаграма відображає процес побудови прогнозу витрат на основі історичних фінансових даних за N місяців. Спочатку здійснюється формування пар значень (x_i, y_i), Далі виконуються розрахунки необхідних статистичних величин (Σx, Σy, Σxy, Σx²), на основі яких визначаються коефіцієнти лінійної регресії a та b.

Після цього будується рівняння тренду y = a + bx, яке використовується для прогнозування значень витрат на наступні місяці (N+1, N+2, N+3). Отримані результати проходять перевірку на коректність, зокрема значення менші за нуль обмежуються нульовим рівнем. На завершальному етапі фактичні та прогнозні дані об’єднуються та відображаються у вигляді лінійного графіка з використанням суцільної та пунктирної ліній для розрізнення реальних і прогнозованих значень.

Таким чином, модулі аналізу витрат та бюджетування утворюють єдиний аналітичний контур системи, який поєднує обробку історичних фінансових даних із механізмами контролю та прогнозування. Це забезпечує користувачу можливість не лише фіксувати власні фінансові операції, але й оцінювати їх у динаміці, контролювати бюджетні обмеження та планувати майбутні витрати на основі статистично обґрунтованих моделей.

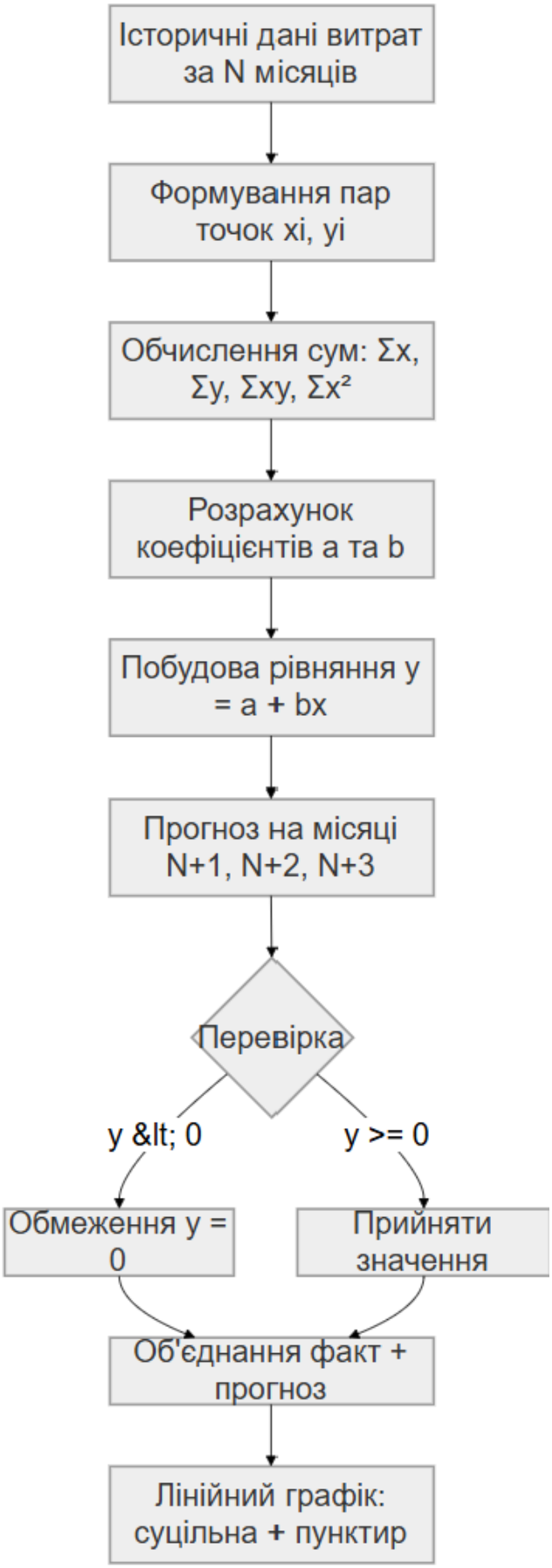


Рисунок 2.5 – Алгоритм прогнозування бюджету методом лінійної регресії

2.5 Обґрунтування вибору технологій розробки

Вибір технологічного стеку для розробки вебзастосунку для управління персональними фінансами здійснювався з урахуванням вимог до продуктивності, масштабованості, простоти розгортання, а також відповідності нефункціональним вимогам системи (NFR), зокрема щодо швидкодії, безпеки, переносимості та зручності супроводу. Основна увага приділялася технологіям, які дозволяють мінімізувати складність архітектури та зменшити кількість зовнішніх залежностей, забезпечуючи при цьому достатній рівень функціональності для персонального застосунку.

Результати порівняльного аналізу альтернативних технологій наведено у таблиці 2.8, де зіставлено обрані рішення з їх можливими альтернативами та обґрунтовано доцільність їх використання в межах проєкту.

Таблиця 2.8 – Порівняння альтернативних технологій

Категорія	**Обраний інструмент**	**Альтернативи**	**Обґрунтування вибору**
Фреймворк	Next.js (App Router) [18]	Vue/Nuxt [19], Angular [20]	Використання серверних компонентів та Server Actions дозволяє зменшити потребу в окремому API, а SSR підвищує продуктивність і швидкість завантаження сторінок
Мова програмування	TypeScript [21]	JavaScript [22]	Статична типізація зменшує кількість помилок під час розробки та підвищує підтримуваність коду
База даних	SQLite (`node:sqlite`) [23]	PostgreSQL [24], MySQL [25]	Вбудована база даних не потребує окремого серверного розгортання, що спрощує встановлення та використання застосунку
UI-бібліотека	shadcn/ui [26] + Tailwind CSS [27]	Bootstrap [28], Material UI [29]	Забезпечує гнучкість стилізації, сучасний дизайн та мінімальне навантаження на фронтенд
Візуалізація даних	Recharts [30]	Chart.js [31]	Декларативний підхід і тісна інтеграція з React-компонентами спрощують реалізацію графіків

Проведений аналіз свідчить, що обраний технологічний стек є оптимальним для реалізації персональної фінансової системи, оскільки забезпечує баланс між продуктивністю, простотою розробки та можливістю подальшого масштабування.

Відповідно до вимог нефункціональної характеристики NFR-04 (переносимість), а також з урахуванням специфіки застосунку як персональної системи обліку фінансів, було обрано вбудовану базу даних SQLite з використанням модуля node:sqlite. Дане рішення є доцільним для застосунків, які не потребують складної серверної інфраструктури та орієнтовані на локальне або індивідуальне використання.

SQLite має низку суттєвих переваг у контексті даного проєкту. По-перше, відсутня необхідність у розгортанні окремого серверу бази даних, що значно спрощує інсталяцію та запуск системи.

По-друге, усі дані зберігаються в єдиному файлі finance.db, що забезпечує зручність резервного копіювання, перенесення та відновлення інформації.

По-третє, система повністю підтримує стандартні SQL-операції, включаючи транзакції, індекси та зовнішні ключі, що є достатнім для реалізації фінансової логіки застосунку. Додатковою перевагою є висока швидкодія при роботі з відносно невеликими обсягами даних, характерними для персонального використання.

Дизайн користувацького інтерфейсу розроблено з урахуванням принципів мінімалізму, зручності сприйняття інформації та логічної структурованості фінансових даних. Основною кольоровою концепцією обрано зелену палітру, яка асоціюється зі стабільністю, фінансовим зростанням та позитивною динамікою. Це дозволяє підсилити тематичну спрямованість застосунку та забезпечити інтуїтивно зрозуміле візуальне середовище для користувача.

Технічно реалізація інтерфейсу базується на використанні моделі кольорів OKLCH, що забезпечує більш рівномірне та передбачуване сприйняття яскравості й контрасту між елементами інтерфейсу. Застосування цієї моделі

дозволяє досягти візуальної гармонії та покращити читабельність фінансових показників.

Окремо слід зазначити, що інтерфейс реалізовано без підтримки темної теми, що спрощує структуру стилів та забезпечує єдину візуальну логіку застосунку. Такий підхід є доцільним для персональної фінансової системи, де пріоритетом є функціональність та швидкість взаємодії, а не складна система кастомізації зовнішнього вигляду.

У результаті проведеного аналізу обґрунтовано вибір технологічного стеку для розробки вебзастосунку персонального фінансового обліку. Встановлено, що використання Next.js, TypeScript, SQLite, Tailwind CSS та Recharts є доцільним з огляду на продуктивність, простоту реалізації, зручність супроводу та відповідність нефункціональним вимогам системи. Обрані технології забезпечують ефективну взаємодію між рівнями системи, високу швидкодію та мінімальну складність розгортання.

Додатково обґрунтовано вибір SQLite як оптимального рішення для персонального застосунку, а також визначено принципи побудови інтерфейсу, орієнтовані на зручність користувача та візуальну узгодженість. Загалом обраний технологічний стек формує стабільну основу для подальшої реалізації та розвитку вебзастосунку.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ З АНАЛІЗОМ ВИТРАТ І ПРОГНОЗУВАННЯМ БЮДЖЕТУ

3.1 Реалізація серверної частини вебсистеми

Реалізація серверної частини вебзастосунку здійснюється на основі фреймворку Next.js із використанням архітектури App Router, що дозволяє поєднувати серверну та клієнтську логіку в межах одного проєкту. Такий підхід забезпечує зменшення складності системи, оскільки відпадає необхідність у побудові окремого REST API, а обробка запитів до бази даних виконується безпосередньо через серверні компоненти та Server Actions.

У межах реалізації серверної частини особлива увага приділена структурі проєкту, організації взаємодії з базою даних, а також реалізації бізнес-логіки, що забезпечує облік, аналіз та прогнозування фінансових операцій користувача.

Структура вебзастосунку побудована таким чином, щоб розділити відповідальність між рівнями інтерфейсу, логіки та доступу до даних. Основні каталоги та файли проєкту наведено у таблиці 3.1.

Таблиця 3.1 – Структура файлів проєкту

Шлях	Призначення
app/layout.tsx	Кореневий макет застосунку, підключення загальної структури інтерфейсу, шрифтів та навігації
app/page.tsx	Головна сторінка (дашборд) із ключовими фінансовими показниками
app/transactions/page.tsx	Сторінка керування транзакціями
app/analytics/page.tsx	Модуль аналітики витрат
app/budget/page.tsx	Модуль бюджетування
app/forecast/page.tsx	Модуль прогнозування фінансових показників
lib/db.ts	Налаштування підключення до SQLite та ініціалізація схеми
lib/queries.ts	Функції для читання даних із бази даних
lib/actions.ts	Server Actions для створення та зміни даних
lib/types.ts	Опис структур типів даних
lib/format.ts	Допоміжні функції форматування валют і дат
components/	UI-компоненти (форми, таблиці, графіки, картки показників)

Структура проєкту відображає модульний підхід до організації коду вебзастосунку на базі Next.js. Каталог app/ містить сторінки застосунку, де layout.tsx відповідає за загальний каркас інтерфейсу, навігацію та підключення спільних ресурсів, а page.tsx реалізує головний дашборд із ключовими фінансовими показниками. Окремі підсторінки, такі як transactions/page.tsx, analytics/page.tsx, budget/page.tsx і forecast/page.tsx, реалізують відповідні функціональні модулі системи – управління транзакціями, аналіз витрат, бюджетування та прогнозування.

Логіка роботи з даними винесена у директорію lib/, що забезпечує чітке розділення відповідальностей. Файл db.ts відповідає за ініціалізацію та налаштування підключення до SQLite, тоді як queries.ts містить функції для читання даних із бази. Операції зміни стану системи реалізовані у actions.ts через Server Actions. Додатково types.ts визначає структури даних, а format.ts містить допоміжні функції для форматування дат і валют.

UI-компоненти винесені в окрему директорію components/, де зосереджені повторно використовувані елементи інтерфейсу, такі як форми введення, таблиці, графіки та картки фінансових показників. Така структура забезпечує логічне розділення фронтенд- та бекенд-логіки, спрощує масштабування системи та підвищує її підтримуваність.

Підключення до бази даних реалізовано за допомогою вбудованого модуля node:sqlite, що дозволяє працювати з SQLite без додаткових зовнішніх залежностей. Для забезпечення стабільності з’єднання та уникнення повторної ініціалізації бази даних під час гарячого перезавантаження застосовано підхід Singleton із глобальним кешуванням екземпляра.

Ініціалізація з’єднання передбачає також налаштування параметрів продуктивності та цілісності даних, зокрема режиму WAL для паралельного читання/запису та увімкнення зовнішніх ключів для контролю зв’язків між таблицями (додаток А.1).

Наведений фрагмент коду реалізує модуль підключення до локальної бази даних SQLite у середовищі Node.js та забезпечує її ініціалізацію за принципом Singleton для уникнення дублювання з’єднань.

Спочатку імпортується клас DatabaseSync із вбудованого модуля node:sqlite, який використовується для синхронної роботи з базою даних. Додатково підключаються модулі path та fs, що забезпечують роботу з файловою системою та формування шляхів до файлів незалежно від операційної системи.

Далі створюється директорія для зберігання бази даних. Шлях DATA_DIR формується як підкаталог .data у кореневій директорії проєкту (process.cwd()). Якщо така директорія не існує, вона автоматично створюється за допомогою fs.mkdirSync, що гарантує наявність середовища для збереження файлу бази даних.

Після цього визначається шлях до самої бази даних – файл finance.db, який розміщується всередині створеної директорії.

Основна логіка інкапсульована у функції createConnection(). У ній створюється екземпляр бази даних через new DatabaseSync(DB_PATH), після чого виконується її конфігурація. Зокрема, встановлюється режим WAL (Write-Ahead Logging), який покращує продуктивність та дозволяє одночасне читання і запис, а також вмикається підтримка зовнішніх ключів через PRAGMA foreign_keys = ON, що забезпечує цілісність зв’язків між таблицями.

Після налаштування викликаються функції initSchema(db) та seedIfEmpty(db), які відповідають за створення структури таблиць і первинне заповнення бази даних початковими даними, якщо вона порожня.

Експортований об’єкт db реалізовано за патерном Singleton: спочатку перевіряється, чи існує глобальний екземпляр global.__financeDb. Якщо він уже створений, використовується він, інакше викликається createConnection() для створення нового підключення. Це запобігає повторному відкриттю бази даних при гарячому перезавантаженні застосунку в режимі розробки.

У кінці передбачено збереження екземпляра у глобальній області видимості під час розробки (NODE_ENV !== "production"), що додатково оптимізує роботу застосунку та зменшує навантаження на файлову систему.

Бізнес-логіка запису даних реалізована за допомогою Server Actions, що дозволяє виконувати серверні операції безпосередньо з компонентів інтерфейсу. Основним завданням даного рівня є прийом даних від користувача, їх валідація, виконання SQL-операцій та оновлення відображення інтерфейсу.

Прикладом є реалізація функції додавання нової транзакції, яка включає перевірку коректності введених даних, виконання параметризованого SQL-запиту та інвалідацію кешу сторінок для оновлення відображуваних даних (додаток А.2).

Цей серверний код реалізує створення нової фінансової транзакції у вебзастосунку.

Функція createTransaction отримує дані з форми (FormData), перетворює їх у потрібні типи та виконує базову валідацію: перевіряє коректність суми та наявність дати. У разі помилки виконання переривається з відповідним повідомленням.

Після перевірки даних виконується SQL-запит INSERT INTO transactions, який додає новий запис у таблицю транзакцій у базі даних SQLite. Для запобігання SQL-ін’єкціям використано параметризовані значення (?).

Після успішного додавання запису викликається revalidatePath, який оновлює відповідні сторінки (/, /transactions, /analytics), щоб інтерфейс відобразив актуальні дані без перезавантаження застосунку.

Отже, реалізація серверної частини забезпечує коректну обробку фінансових операцій, включаючи валідацію даних, безпечне збереження інформації у базі даних SQLite та оперативне оновлення інтерфейсу через механізм revalidation. Використання Server Actions дозволило відмовитися від окремого API-шару, що спростило архітектуру та підвищило ефективність взаємодії між клієнтською та серверною частинами застосунку.

3.2 Розробка клієнтського інтерфейсу користувача

Клієнтська частина вебзастосунку реалізована як набір React-компонентів у середовищі Next.js та відповідає за відображення даних, взаємодію з користувачем і побудову інтерактивного інтерфейсу для аналізу фінансової інформації. Основний акцент зроблено на зручності використання, візуальній зрозумілості та узгодженості дизайну в межах усієї системи.

Візуальний стиль інтерфейсу побудовано на основі зеленої кольорової гами, яка асоціюється з фінансовим зростанням і стабільністю. Для реалізації єдиного стилістичного підходу використано CSS-змінні (design tokens), визначені у файлі globals.css. Кольори задаються у просторі OKLCH, що забезпечує більш точне та рівномірне сприйняття відтінків на різних пристроях і екранах.

Основні змінні включають фоновий колір, головний акцентний колір, кольори графіків та колір для позначення критичних фінансових станів, зокрема перевитрат. Такий підхід дозволяє легко масштабувати дизайн та підтримувати консистентність відображення даних у різних модулях системи.

Структура інтерфейсу побудована навколо постійної бічної навігаційної панелі (рис. 3.1), що забезпечує швидкий доступ до основних функціональних розділів системи. Інтерфейс розділено на логічні модулі, кожен з яких відповідає окремому бізнес-процесу: огляд фінансових показників, управління транзакціями, аналітика, бюджетування та прогнозування.

Така організація дозволяє користувачу інтуїтивно переміщатися між функціональними блоками без втрати контексту, а також підтримує масштабованість системи при додаванні нових модулів у майбутньому.

Для відображення аналітичних даних використовується бібліотека Recharts, яка інтегрується як набір клієнтських React-компонентів і дозволяє будувати інтерактивні графіки на основі отриманих із серверної частини даних.

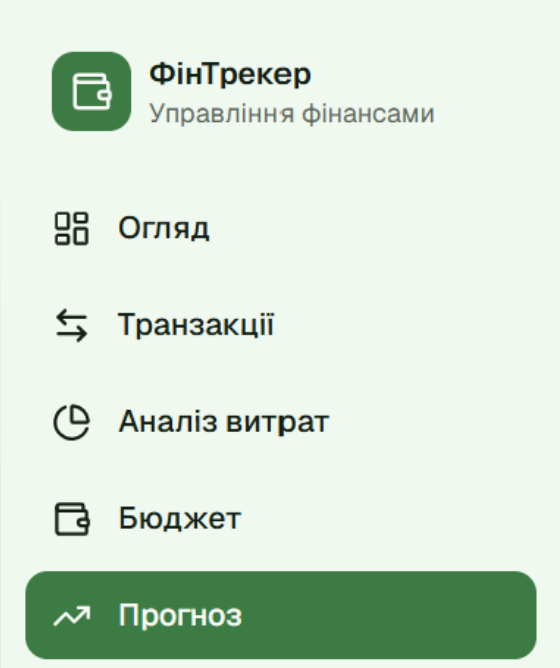


Рисунок 3.1 – Панель навігації

Прикладом є кругова діаграма структури витрат за категоріями, де дані передаються у вигляді масиву об’єктів, а кожен сегмент діаграми відображає частку відповідної категорії. Для кожного елемента задається індивідуальний колір, що підвищує візуальну читабельність і дозволяє швидко аналізувати розподіл витрат.

Додатково реалізовано інтерактивні підказки (Tooltip), які відображають точні числові значення у форматі валюти, що забезпечує більш точне сприйняття фінансових даних та підвищує аналітичну ефективність інтерфейсу.

Інтерфейс розробленої вебсистеми наведено в додатку Б.

Отже, розробка клієнтського інтерфейсу забезпечила створення зручного, інтуїтивно зрозумілого та візуально узгодженого середовища для роботи з фінансовими даними. Використання компонентного підходу React та бібліотеки Recharts дозволило реалізувати інтерактивну візуалізацію показників і забезпечити ефективну взаємодію користувача з системою. Єдина кольорова схема та продумана навігація підвищують зручність використання та сприяють швидкому доступу до основних функціональних модулів.

3.3 Реалізація функціоналу обліку фінансових операцій

Реалізація функціоналу обліку фінансових операцій у вебсистемі побудована навколо принципу серверно-орієнтованої обробки даних, де всі зміни стану виконуються через Server Actions, а відображення оновлюється автоматично після інвалідації кешу сторінок. Основною одиницею роботи системи є фінансова транзакція, яка містить інформацію про суму, тип операції, категорію, дату та додаткові примітки.

Введення нових фінансових операцій реалізовано через модальну форму, яка передає дані безпосередньо у серверну дію за допомогою атрибута action. Такий підхід дозволяє уникнути додаткового рівня API та забезпечує мінімальну затримку між введенням даних і їх збереженням у базі даних. Перед відправкою форма проходить клієнтську валідацію, що дозволяє відфільтрувати некоректні значення ще до взаємодії із сервером.

Загальний процес взаємодії компонентів під час додавання транзакції подано на рисунку 3.2.

Діаграма відображає послідовність обробки операції додавання транзакції: користувач вводить дані у форму, після чого виконується клієнтська валідація. У разі виявлення помилки система повертає повідомлення про некоректні дані, а при успішній перевірці дані передаються до серверної дії createTransaction.

На сервері виконується запис інформації у базу даних SQLite через SQL INSERT-запит. Після успішного збереження викликається механізм revalidatePath, який ініціалізує оновлення відповідних сторінок застосунку. У результаті автоматично оновлюються список транзакцій, аналітичні графіки та зведені показники фінансової системи, забезпечуючи актуальність відображених даних без перезавантаження застосунку.

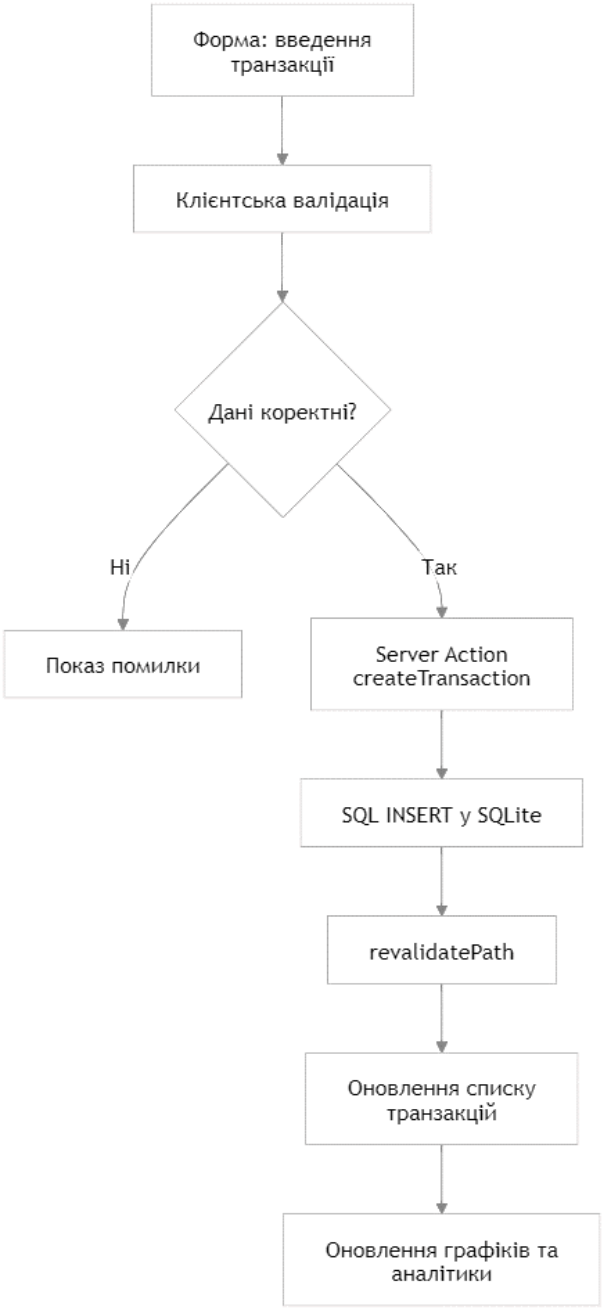


Рисунок 3.2 – Потік даних операції додавання фінансової транзакції

Відображення списку транзакцій у системі реалізовано як серверний компонент, який безпосередньо звертається до бази даних через модуль запитів. Такий підхід дозволяє отримувати завжди актуальні дані без додаткових проміжних API-шарів. Кожен запис містить структуровану інформацію про фінансову операцію, зокрема суму, тип (дохід або витрата), категорію та дату, що забезпечує коректну подальшу аналітичну обробку та відображення у вигляді таблиць і графіків.

Окремо реалізовано функціонал видалення транзакцій, який виконується через серверну дію:

```
export async function deleteTransaction(id: number) {
  db.prepare("DELETE FROM transactions WHERE id = ?").run(id)
  revalidatePath("/")
  revalidatePath("/transactions")
  revalidatePath("/analytics")
}
```

У цьому фрагменті функція deleteTransaction приймає ідентифікатор запису та виконує SQL-команду DELETE, яка видаляє відповідний рядок із таблиці transactions. Використання параметра ? забезпечує захист від SQL-ін’єкцій. Після видалення викликається revalidatePath, який примусово оновлює кеш сторінок дашборду, списку транзакцій та аналітики. Таким чином гарантується синхронність відображення даних у всіх частинах інтерфейсу.

Формування зведених фінансових показників базується на використанні агрегувальних SQL-запитів, що дозволяє виконувати обчислення безпосередньо на рівні бази даних. Це значно зменшує навантаження на клієнтську частину та підвищує продуктивність системи при роботі з великим обсягом транзакцій.

```
export function getMonthTotals(month: string) {
  const row = db.prepare(
    `SELECT
      COALESCE(SUM(CASE WHEN type='income' THEN amount END),0)
AS income,
      COALESCE(SUM(CASE WHEN type='expense' THEN amount END),0)
AS expense
     FROM transactions WHERE substr(date,1,7) = ?`
  ).get(month) as { income: number; expense: number }
```

```
  return {
    income: row.income,
    expense: row.expense,
    balance: row.income - row.expense
  }
}
```

У цьому коді виконується умовна агрегація даних із таблиці transactions. За допомогою конструкції CASE WHEN окремо підраховуються суми доходів і витрат за вибраний місяць, а функція COALESCE гарантує повернення нульового значення у випадку відсутності даних. Далі на основі отриманих результатів розраховується баланс як різниця між доходами та витратами. Отримані показники використовуються у дашборді та модулях аналітики для побудови графіків і карток фінансового стану користувача.

Отже, реалізовано основний функціонал обліку фінансових операцій, який включає додавання, перегляд, видалення транзакцій та формування зведених фінансових показників. Система побудована на основі серверних дій Next.js, що дозволяє обробляти всі операції безпосередньо на сервері та забезпечує цілісність і актуальність даних. Застосування параметризованих SQL-запитів підвищує рівень безпеки за рахунок захисту від SQL-ін’єкцій, а використання механізму інвалідації кешу забезпечує автоматичне оновлення інтерфейсу після змін у базі даних. Крім того, реалізація агрегувальних запитів дозволяє ефективно формувати аналітичні показники без додаткового навантаження на клієнтську частину.

3.4 Реалізація аналітики витрат та прогнозування бюджету

У даному підрозділі реалізовано функціонал аналітичної обробки фінансових даних та прогнозування майбутніх витрат на основі історичних значень. Основу модуля становлять агрегувальні SQL-запити та алгоритм лінійної регресії, що дозволяє поєднати статистичний аналіз із візуалізацією даних у клієнтській частині системи.

Аналітика витрат у системі базується на обробці даних безпосередньо на рівні бази даних із використанням SQL-агрегацій. Зокрема, структура витрат за категоріями визначається шляхом групування транзакцій за category_id, а помісячна динаміка розраховується на основі частини дати substr(date,1,7), що дозволяє виділяти рік і місяць із повної дати.

Окремим елементом є модуль бюджетного контролю, який поєднує планові та фактичні значення витрат. Для цього використовується запит із вкладеним підзапитом, що дозволяє обчислювати суму витрат у межах кожної категорії бюджету та порівнювати її із запланованим лімітом.

```
export function getBudgets(month: string): Budget[] {
  return plain<Budget[]>(db.prepare(
    `SELECT b.*, c.name AS category_name, c.color AS category_color,
      COALESCE((
        SELECT SUM(t.amount) FROM transactions t
        WHERE t.category_id = b.category_id
          AND t.type='expense'
          AND substr(t.date,1,7) = b.month
      ), 0) AS spent
    FROM budgets b
    JOIN categories c ON c.id = b.category_id
    WHERE b.month = ? ORDER BY c.name`
  ).all(month))
}
```

У наведеному коді виконується об’єднання таблиць budgets та categories для отримання повної інформації про кожен бюджетний запис, включаючи назву та колір категорії. Підзапит розраховує фактичні витрати (spent) у межах відповідного місяця та категорії. Функція COALESCE забезпечує повернення нульового значення у випадку відсутності транзакцій. У результаті формується структурований набір даних, який використовується для побудови індикаторів виконання бюджету.

Прогнозування витрат реалізовано на основі методу лінійної регресії, що дозволяє визначити загальну тенденцію зміни фінансових показників у часі. Вхідними даними виступає послідовність історичних значень місячних витрат, на основі яких обчислюється рівняння прямої $y = a + bx$.

Коефіцієнти регресії розраховуються за класичними формулами методу найменших квадратів, що дозволяє мінімізувати похибку між фактичними та апроксимованими значеннями.

```
export function linearForecast(history: number[], periods = 3): number[] {
  const n = history.length
  if (n < 2) return Array(periods).fill(history[0] ?? 0)

  let sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0
  history.forEach((y, i) => {
    const x = i + 1
    sumX += x; sumY += y; sumXY += x * y; sumX2 += x * x
  })

  const b = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX * sumX)
  const a = (sumY - b * sumX) / n

  const result: number[] = []
```

```
  for (let k = 1; k <= periods; k++) {
    const x = n + k
    result.push(Math.max(0, a + b * x))
  }
  return result
}
```

У даній реалізації спочатку обчислюються сумарні значення змінних, після чого визначаються коефіцієнти нахилу та перетину прямої. Далі формується прогноз на задану кількість періодів, причому додатково застосовується обмеження Math.max(0, ...), що гарантує недопущення від’ємних значень витрат.

Отримані прогнозні дані використовуються у візуалізації у вигляді поєднання фактичної (суцільної) та прогнозної (пунктирної) ліній, що дозволяє користувачеві наочно оцінити тенденції зміни витрат та планувати бюджет на майбутні періоди.

Отже, реалізовано модуль аналітики витрат та прогнозування бюджету, який забезпечує комплексну обробку фінансових даних на основі агрегувальних SQL-запитів та алгоритму лінійної регресії. Це дозволило отримувати структуровані аналітичні показники щодо витрат за категоріями, часом та виконанням бюджету без додаткового навантаження на клієнтську частину системи.

Розроблений механізм прогнозування дає змогу визначати тенденції зміни витрат у майбутніх періодах, що підвищує практичну цінність застосунку та дозволяє користувачеві ефективніше планувати фінансові ресурси. Загалом, модуль забезпечує поєднання аналітичної обробки та візуалізації даних у єдиній інтегрованій системі.

3.5 Тестування вебсистеми

У процесі тестування вебсистеми було застосовано метод функціонального тестування (метод «чорної скриньки»), що передбачає перевірку основних сценаріїв використання без аналізу внутрішньої реалізації коду. Додатково здійснювалася перевірка коректності відображення інтерфейсу в сучасних браузерах та відповідності поведінки системи очікуваним результатам.

Основні результати тестування наведено у таблиці 3.2, де перевірялися ключові операції з фінансовими даними, включаючи додавання, видалення транзакцій, валідацію введених значень, побудову графіків, встановлення бюджету та формування прогнозів. Усі тестові сценарії були успішно виконані, що підтверджує коректність реалізації функціональних модулів системи.

Таблиця 3.2 – Результати функціонального тестування

№	Тестовий сценарій	Очікуваний результат	Статус
Т-01	Додавання витрати	Операція додається, баланс оновлюється	Пройдено
Т-02	Додавання доходу	Баланс збільшується, оновлюються графіки	Пройдено
Т-03	Видалення транзакції	Запис видаляється, дані перераховуються	Пройдено
Т-04	Некоректна сума	Поява повідомлення про помилку	Пройдено
Т-05	Структура витрат	Відображення кругової діаграми	Пройдено
Т-06	Бюджетування	Збереження ліміту та індикатор виконання	Пройдено
Т-07	Перевищення бюджету	Позначення перевитрати	Пройдено
Т-08	Прогнозування	Побудова прогнозної лінії	Пройдено

Методика тестування базувалася на перевірці поведінки системи у відповідь на типові дії користувача. Окрему увагу приділено сценаріям обробки некоректних даних, зокрема введенню від’ємних значень, нульової суми або відсутності обов’язкових полів. Це дозволило перевірити надійність механізмів валідації та захисту цілісності даних.

Результати тестування показали, що система коректно обробляє всі передбачені сценарії. Додавання транзакцій призводить до автоматичного оновлення списків, аналітики та дашборду, а видалення записів забезпечує синхронне оновлення пов’язаних компонентів. Валідація вхідних даних працює стабільно та запобігає збереженню некоректної інформації в базі даних. Також

підтверджено коректність роботи бюджетного контролю та алгоритму прогнозування витрат.

Загалом, тестування засвідчило стабільність роботи вебсистеми, відсутність критичних помилок та відповідність реалізованого функціоналу поставленим вимогам.

3.6 Інструкція користувача

Після запуску вебзастосунку користувач потрапляє на головну сторінку «Огляд», де відображаються основні фінансові показники за обраний період: загальні доходи, витрати, баланс, а також ключові графіки, що демонструють структуру та динаміку фінансових потоків.

Для роботи з фінансовими операціями користувач переходить до розділу «Транзакції», де може додавати нові записи через спеціальну форму. При створенні транзакції необхідно вказати тип операції (дохід або витрата), суму, категорію, дату та, за потреби, додаткову примітку. Після збереження запис автоматично з’являється у списку, а всі аналітичні показники оновлюються.

Видалення операцій здійснюється безпосередньо зі списку транзакцій шляхом натискання відповідної кнопки. Після видалення система автоматично перераховує баланс та оновлює графіки на всіх пов’язаних сторінках.

У розділі «Аналіз витрат» користувач може переглядати структуру витрат за категоріями та їхню динаміку в часі. Графічні елементи дозволяють швидко оцінити основні напрями витрат і виявити фінансові тенденції.

Розділ «Бюджет» призначений для встановлення місячних лімітів витрат за окремими категоріями. Система автоматично відстежує фактичні витрати та сигналізує про перевищення встановленого бюджету.

У розділі «Прогноз» відображається прогнозована динаміка витрат, побудована на основі історичних даних. Це дозволяє користувачеві оцінити майбутні фінансові тенденції та заздалегідь коригувати свої витрати.

Для отримання коректних аналітичних результатів рекомендується регулярно оновлювати дані, вносячи всі фінансові операції одразу після їх здійснення, а також підтримувати достатній обсяг історичних записів для точнішого прогнозування.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було повністю реалізовано всі поставлені завдання, що забезпечило досягнення визначеної мети дослідження – проєктування та розробку вебсистеми для управління персональними фінансами з функціями аналізу витрат та прогнозування бюджету.

На першому етапі було здійснено детальний аналіз предметної області управління персональними фінансами. Розглянуто основні принципи ведення фінансового обліку в цифровому середовищі, визначено роль інформаційних систем у підвищенні ефективності контролю доходів і витрат користувача. Дослідження існуючих програмних рішень показало, що більшість сучасних застосунків забезпечують базовий функціонал обліку транзакцій, однак часто мають обмежені можливості гнучкої аналітики, візуалізації даних та прогнозування фінансових показників.

На основі аналізу було досліджено функціональні можливості сучасних систем обліку доходів і витрат. Встановлено, що ключовими компонентами таких систем є модулі управління транзакціями, категоризації фінансових операцій, формування звітності та базова аналітика. Це дозволило сформувати перелік функцій, які повинна містити розроблювана вебсистема для забезпечення повноцінного фінансового контролю користувача.

Проведено порівняльний аналіз існуючих вебзастосунків для фінансового планування. У результаті визначено їх основні переваги, такі як зручний інтерфейс та наявність базових аналітичних функцій, а також недоліки — недостатня гнучкість налаштувань, обмежені можливості персоналізованого аналізу та відсутність або спрощеність прогнозування. Це дозволило обґрунтувати необхідність створення власного рішення з розширеною аналітичною складовою.

На наступному етапі було сформовано вимоги до вебсистеми управління персональними фінансами. Визначено функціональні вимоги, що включають

облік доходів і витрат, управління категоріями, формування аналітичних звітів, бюджетування та прогнозування витрат. Також враховано нефункціональні вимоги, зокрема продуктивність, простота використання, швидкість обробки даних та надійність зберігання фінансової інформації.

Було спроєктовано архітектуру вебзастосунку, яка передбачає поділ на рівні представлення, бізнес-логіки та доступу до даних. Така структура забезпечує модульність системи, спрощує супровід і дозволяє масштабувати функціонал у майбутньому. Окремо розроблено структуру бази даних, що включає основні сутності: транзакції, категорії та бюджети, а також зв’язки між ними для забезпечення цілісності та узгодженості фінансових даних.

У процесі реалізації було створено основний функціонал вебсистеми. Реалізовано механізми додавання, перегляду та видалення фінансових операцій, їх категоризацію, а також автоматичне оновлення аналітичних показників. Окрему увагу приділено модулю аналітики, який забезпечує формування узагальнених фінансових даних за різними періодами, а також модулю бюджетування, що дозволяє контролювати перевищення встановлених лімітів.

Крім того, реалізовано функціонал прогнозування витрат на основі історичних даних із використанням математичного методу лінійної регресії. Це дозволяє оцінювати майбутні фінансові тенденції та підвищує практичну цінність системи для користувача.

Отже, виконані завдання підтверджують повне досягнення мети роботи та створення функціональної вебсистеми, яка поєднує облік фінансових операцій, аналітику, бюджетування та прогнозування, забезпечуючи ефективне управління персональними фінансами користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Financial Information Systems | Business and Management | Research Starters | EBSCO Research. URL : https://www.ebsco.com/research-starters/business-and-management/financial-information-systems (дата звернення : 28.03.2026).

2. Chahar P., Vishwakarma Y. K., Mishra R., Paliwal G. Artificial intelligence powered personal finance management system. SSRN Electronic Journal. 2026. DOI : https://doi.org/10.2139/ssrn.6377518.

3. Grados-Espinoza J., Velasquez-Jimenez L. Design and implementation of personal finance software for controlling financial efficiency. International Journal of Engineering Trends and Technology. 2025. Vol. 73, No. 2. P. 107–118. DOI : https://doi.org/10.14445/22315381/ijett-v73i2p109.

4. Jasim L., Gatea A. K. The role of financial information systems in financial performance. Socio-economic relations in the digital society. 2025. Vol. 1, No. 55. P. 19–27. DOI : https://doi.org/10.55643/ser.5.55.2025.592.

5. Alenazi M., Sas C. Creating and managing transactions and budgets: analysis of marketplace descriptions and functionality review of budgeting apps. Interacting with Computers. 2024. Vol. 37, No. 5. P. 381–396. DOI : https://doi.org/10.1093/iwc/iwae041.

6. Гевчук А. В., Савчук Б. П. Роль цифрових інструментів у вдосконаленні обліку доходів та витрат – результат ефективності сучасного підприємства. International Scientific Journal “Internauka”. URL : https://www.inter-nauka.com/uploads/public/17708301649622.pdf (дата звернення : 28.03.2026).

7. Мінфін. Топ-10 додатків для ведення бюджету. Як стати багатим? URL : https://new.minfin.com.ua/ua/ideabank/10-prilozhenij-kotorye-pomogut-razbogatet (дата звернення : 28.03.2026).

8. Бразілій Н., Чичужко М., Крот Ю., Пастернак Я. Мобільні додатки для автоматизації обліку: нові можливості для підприємців і бухгалтерів. Herald

of Khmelnytskyi National University. Economic Sciences. 2025. No. 342(3(1)). P. 205–215. DOI : https://doi.org/10.31891/2307-5740-2025-342-3(1)-30.

9. Бенько В. Облік доходів і витрат: вітчизняний та зарубіжний досвід. Світ фінансів. 2024. No. 2(79). P. 130–143. URL : https://sf.wunu.edu.ua/index.php/sf/article/view/1699 (дата звернення : 28.03.2026).

10. Dianapps. Top 10 personal finance software for 2025. URL : https://dianapps.com/blog/top-10-personal-finance-software-for-2025 (дата звернення : 28.03.2026).

11. YNAB. YNAB – a better Mint alternative. URL : https://www.ynab.com/blog/ynab-a-better-mint-alternative (дата звернення : 28.03.2026).

12. Nasdaq. 10 best budgeting apps 2025: stay in control of your finances. URL : https://www.nasdaq.com/articles/10-best-budget-apps-2025-stay-control-your-finances (дата звернення : 28.03.2026).

13. NerdWallet. The best budget apps for 2026. URL : https://www.nerdwallet.com/article/finance/best-budget-apps (дата звернення : 28.03.2026).

14. Yahoo Tech / PCMag. PocketGuard review. URL : https://tech.yahoo.com/apps/articles/pocketguard-181714446.html (дата звернення : 28.03.2026).

15. Eastern Peak. Personal finance app development: features & cost. URL : https://easternpeak.com/blog/personal-finance-app-development/ (дата звернення : 28.03.2026).

16. Marriage, Kids and Money. Simplifi vs Monarch Money. URL : https://marriagekidsandmoney.com/simplifi-vs-monarch-money-which-is-the-best-mint-alternative/ (дата звернення : 28.03.2026).

17. Quicken. Quicken Simplifi vs Monarch: get more, pay less. URL : https://www.quicken.com/lp/quicken-vs-monarch/ (дата звернення : 28.03.2026).

18. Next.js by Vercel – the React framework. URL : https://nextjs.org/ (дата звернення : 28.03.2026).

19. Nuxt. NUXT: the full-stack Vue framework. URL : https://nuxt.com/ (дата звернення : 28.03.2026).

20. Angular. URL : https://angular.dev/ (дата звернення : 28.03.2026).

21. TypeScript: JavaScript with syntax for types. URL : https://www.typescriptlang.org/ (дата звернення : 28.03.2026).

22. JavaScript. MDN Web Docs. URL : https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення : 28.03.2026).

23. SQLite documentation. URL : https://sqlite.org/docs.html (дата звернення : 28.03.2026).

24. PostgreSQL. URL : https://www.postgresql.org/ (дата звернення : 28.03.2026).

25. MySQL documentation. URL : https://dev.mysql.com/doc/ (дата звернення : 28.03.2026).

26. shadcn/ui documentation. URL : https://ui.shadcn.com/ (дата звернення : 28.03.2026).

27. Tailwind CSS documentation. URL : https://tailwindcss.com/ (дата звернення : 28.03.2026).

28. Bootstrap. URL : https://getbootstrap.com/ (дата звернення : 28.03.2026).

29. Material UI. URL : https://mui.com/material-ui/ (дата звернення : 28.03.2026).

30. Recharts. URL : https://recharts.github.io/ (дата звернення : 28.03.2026).

31. Chart.js. Open Source HTML5 Charts for Your Website. URL : https://www.chartjs.org/ (дата звернення : 28.03.2026).

ДОДАТОК А

Лістинг реалізації серверної частини вебсистеми

Лістинг А.1 Реалізація з’єднання з базою даних

```
import { DatabaseSync } from "node:sqlite"
import path from "path"
import fs from "fs"

// ── Налаштування шляху до бази даних ──────────────────────────

const DATA_DIR = path.join(process.cwd(), ".data")
if (!fs.existsSync(DATA_DIR)) {
  fs.mkdirSync(DATA_DIR, { recursive: true })
}
const DB_PATH = path.join(DATA_DIR, "finance.db")

// ── Singleton-екземпляр з'єднання ─────────────────────────────

declare global {
  // eslint-disable-next-line no-var
  var __financeDb: DatabaseSync | undefined
}

function createConnection(): DatabaseSync {
  const db = new DatabaseSync(DB_PATH)
  db.exec("PRAGMA journal_mode = WAL;")
```

```
  db.exec("PRAGMA foreign_keys = ON;")
  initSchema(db)
  seedIfEmpty(db)
  return db
}

export const db: DatabaseSync = global.__financeDb ?? createConnection()
if (process.env.NODE_ENV !== "production") {
  global.__financeDb = db
}

// ── Схема бази даних ──────────────────────────────────────────

function initSchema(db: DatabaseSync) {
  db.exec(`
    CREATE TABLE IF NOT EXISTS categories (
      id         INTEGER PRIMARY KEY AUTOINCREMENT,
      name       TEXT    NOT NULL UNIQUE,
      type       TEXT    NOT NULL CHECK (type IN ('income','expense')),
      color      TEXT    NOT NULL DEFAULT '#3f9142',
      icon       TEXT    NOT NULL DEFAULT 'wallet'
    );

    CREATE TABLE IF NOT EXISTS transactions (
      id         INTEGER PRIMARY KEY AUTOINCREMENT,
      amount     REAL    NOT NULL,
      type       TEXT    NOT NULL CHECK (type IN ('income','expense')),
      category_id INTEGER REFERENCES categories(id) ON DELETE SET
NULL,
      note       TEXT    NOT NULL DEFAULT '',
```

```
    date        TEXT    NOT NULL,
    created_at  TEXT    NOT NULL DEFAULT (datetime('now'))
   );

   CREATE TABLE IF NOT EXISTS budgets (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    category_id INTEGER NOT NULL REFERENCES categories(id) ON DELETE CASCADE,
    amount      REAL    NOT NULL,
    month       TEXT    NOT NULL,
    UNIQUE (category_id, month)
   );

   CREATE INDEX IF NOT EXISTS idx_tx_date ON transactions(date);
   CREATE INDEX IF NOT EXISTS idx_tx_category ON transactions(category_id);
  `)
 }

 // ── Початкові дані для демонстрації ──────────────────────────

 function seedIfEmpty(db: DatabaseSync) {
  const count = db.prepare("SELECT COUNT(*) AS c FROM categories").get() as { c: number }
  if (count.c > 0) return

  db.exec("BEGIN")
  try {
   const insertCat = db.prepare(
    "INSERT INTO categories (name, type, color, icon) VALUES (?, ?, ?, ?)",
```

```
  )
  const categories: [string, string, string, string][] = [
    ["Зарплата", "income", "#2f8f46", "briefcase"],
    ["Фріланс", "income", "#46a06a", "laptop"],
    ["Інвестиції", "income", "#5fb37e", "trending-up"],
    ["Продукти", "expense", "#3f9142", "shopping-cart"],
    ["Транспорт", "expense", "#5aa364", "car"],
    ["Житло", "expense", "#2d7a3e", "home"],
    ["Розваги", "expense", "#74b87f", "gamepad-2"],
    ["Здоров'я", "expense", "#4c9d5a", "heart-pulse"],
    ["Кафе та ресторани", "expense", "#88c499", "utensils"],
    ["Інше", "expense", "#9bcea9", "ellipsis"],
  ]
  const catIds: Record<string, number> = {}
  for (const c of categories) {
    const r = insertCat.run(...c)
    catIds[c[0]] = Number(r.lastInsertRowid)
  }

  // Генеруємо транзакції за останні 5 місяців
  const insertTx = db.prepare(
    "INSERT INTO transactions (amount, type, category_id, note, date) VALUES (?, ?, ?, ?, ?)",
  )
  const now = new Date()
  const pad = (n: number) => String(n).padStart(2, "0")

  for (let m = 4; m >= 0; m--) {
    const d = new Date(now.getFullYear(), now.getMonth() - m, 1)
    const y = d.getFullYear()
```

```
    const mo = d.getMonth()

    // Доходи
    insertTx.run(42000, "income", catIds["Зарплата"], "Місячна зарплата", `${y}-${pad(mo + 1)}-05`)
    if (m % 2 === 0)
      insertTx.run(
        6500 + Math.round(Math.random() * 4000),
        "income",
        catIds["Фріланс"],
        "Проєкт",
        `${y}-${pad(mo + 1)}-18`,
      )

    // Витрати — кілька на місяць
    const expenseCats = ["Продукти", "Транспорт", "Житло", "Розваги", "Здоров'я", "Кафе та ресторани"]
    const baseAmounts: Record<string, number> = {
      Продукти: 1200,
      Транспорт: 600,
      Житло: 9000,
      Розваги: 800,
      "Здоров'я": 700,
      "Кафе та ресторани": 900,
    }
    for (const name of expenseCats) {
      const occurrences = name === "Продукти" || name === "Кафе та ресторани" ? 4 : name === "Житло" ? 1 : 2
      for (let i = 0; i < occurrences; i++) {
        const day = 2 + Math.floor(Math.random() * 26)
```

```
            const amt = Math.round(baseAmounts[name] * (0.7 + Math.random() *
0.6))
            insertTx.run(amt, "expense", catIds[name], name, `${y}-${pad(mo + 1)}-
${pad(day)}`)
          }
        }
      }

      // Бюджети на поточний місяць
      const month = `${now.getFullYear()}-${pad(now.getMonth() + 1)}`
      const insertBudget = db.prepare(
        "INSERT INTO budgets (category_id, amount, month) VALUES (?, ?, ?)",
      )
      const budgets: [string, number][] = [
        ["Продукти", 6000],
        ["Транспорт", 1500],
        ["Житло", 9500],
        ["Розваги", 2000],
        ["Здоров'я", 1500],
        ["Кафе та ресторани", 3000],
      ]
      for (const [name, amount] of budgets) {
        insertBudget.run(catIds[name], amount, month)
      }

      db.exec("COMMIT")
    } catch (e) {
      db.exec("ROLLBACK")
      throw e
    }
```

```
}
```

Лістинг А.2 Реалізація серверних дій

```
"use server"

import { db } from "@/lib/db"
import { revalidatePath } from "next/cache"

//                    ──                    Транзакції
──────────────────────────────────────────────────────────────────────
────

export async function createTransaction(formData: FormData) {
  const amount = Number(formData.get("amount"))
  const type = String(formData.get("type"))
  const categoryId = Number(formData.get("category_id"))
  const note = String(formData.get("note") ?? "")
  const date = String(formData.get("date"))

  if (!amount || amount <= 0) return { error: "Введіть коректну суму" }
  if (type !== "income" && type !== "expense") return { error: "Невірний тип"
}
  if (!date) return { error: "Вкажіть дату" }

  db.prepare(
    "INSERT INTO transactions (amount, type, category_id, note, date)
VALUES (?, ?, ?, ?, ?)",
  ).run(amount, type, categoryId || null, note, date)
```

```
  revalidatePath("/")
  revalidatePath("/transactions")
  revalidatePath("/analytics")
  revalidatePath("/forecast")
  return { success: true }
}

export async function deleteTransaction(id: number) {
  db.prepare("DELETE FROM transactions WHERE id = ?").run(id)
  revalidatePath("/")
  revalidatePath("/transactions")
  revalidatePath("/analytics")
  revalidatePath("/forecast")
  return { success: true }
}

// ── Категорії ──────────────────────────────────────────────────────

export async function createCategory(formData: FormData) {
  const name = String(formData.get("name") ?? "").trim()
  const type = String(formData.get("type"))
  const color = String(formData.get("color") ?? "#3f9142")

  if (!name) return { error: "Введіть назву категорії" }
  try {
    db.prepare("INSERT INTO categories (name, type, color, icon) VALUES (?, ?, ?, 'wallet')").run(
      name,
      type,
```

```
      color,
    )
  } catch {
    return { error: "Категорія з такою назвою вже існує" }
  }
  revalidatePath("/")
  revalidatePath("/transactions")
  revalidatePath("/budget")
  return { success: true }
}

// ── Бюджети ──────────────────────────────────────────────────────────

export async function setBudget(formData: FormData) {
  const categoryId = Number(formData.get("category_id"))
  const amount = Number(formData.get("amount"))
  const month = String(formData.get("month"))

  if (!categoryId) return { error: "Оберіть категорію" }
  if (!amount || amount <= 0) return { error: "Введіть суму бюджету" }

  db.prepare(
    `INSERT INTO budgets (category_id, amount, month) VALUES (?, ?, ?)
     ON CONFLICT(category_id, month) DO UPDATE SET amount = excluded.amount`,
  ).run(categoryId, amount, month)

  revalidatePath("/budget")
  revalidatePath("/forecast")
```

```
  return { success: true }
}

export async function deleteBudget(id: number) {
  db.prepare("DELETE FROM budgets WHERE id = ?").run(id)
  revalidatePath("/budget")
  return { success: true }
}
```

ДОДАТОК Б

Інтерфейс користувача

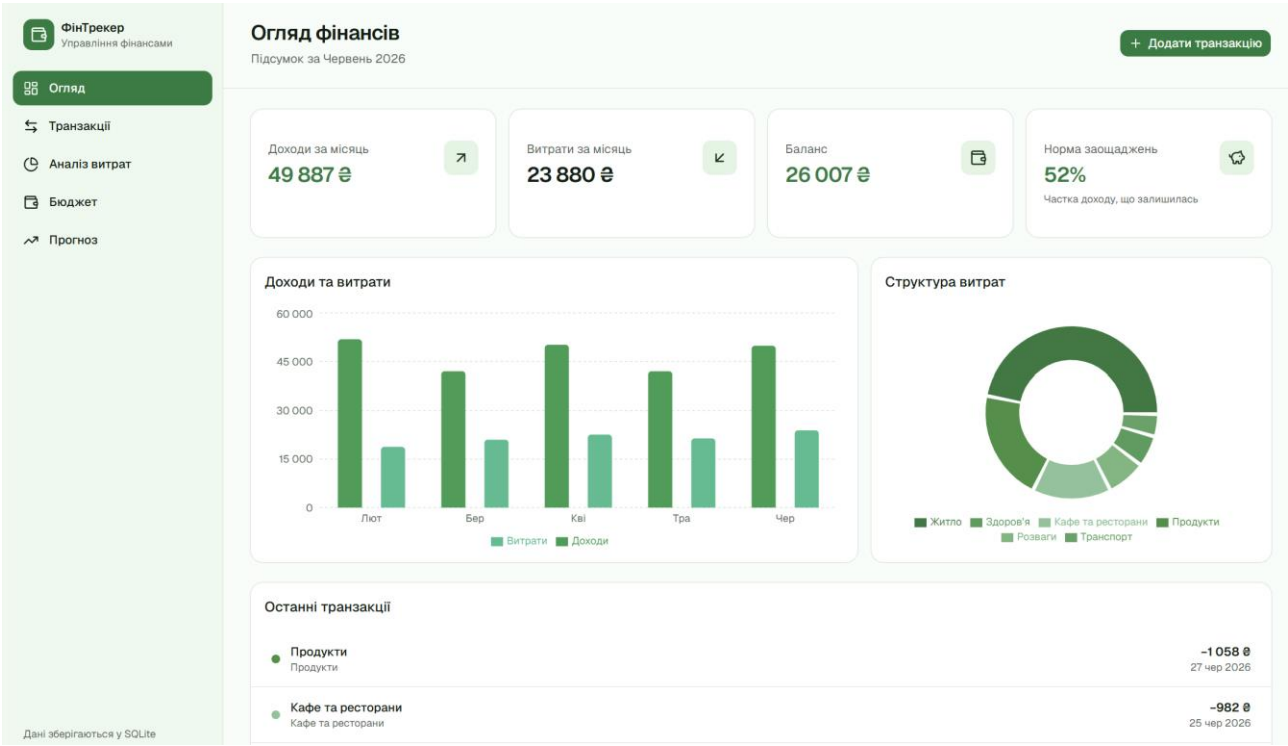


Рисунок Б.1 – Сторінка огляду даних

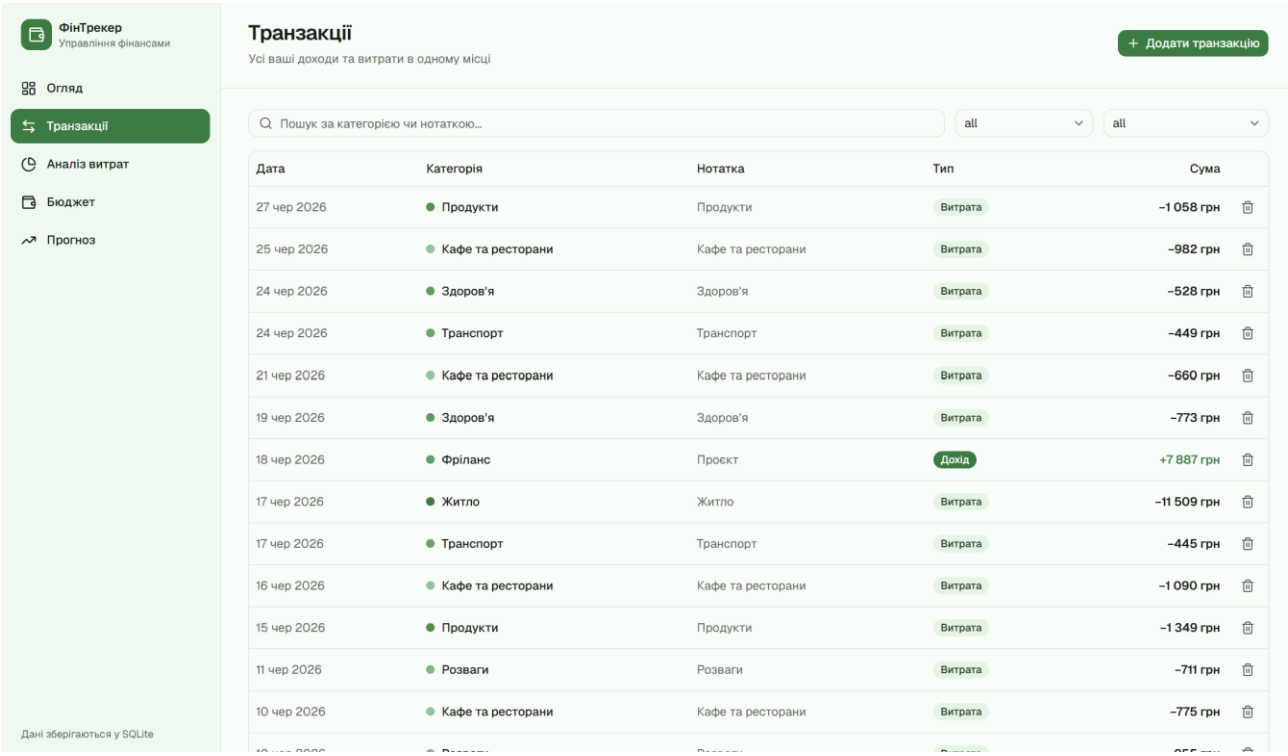


Рисунок Б.2 – Сторінка транзакцій

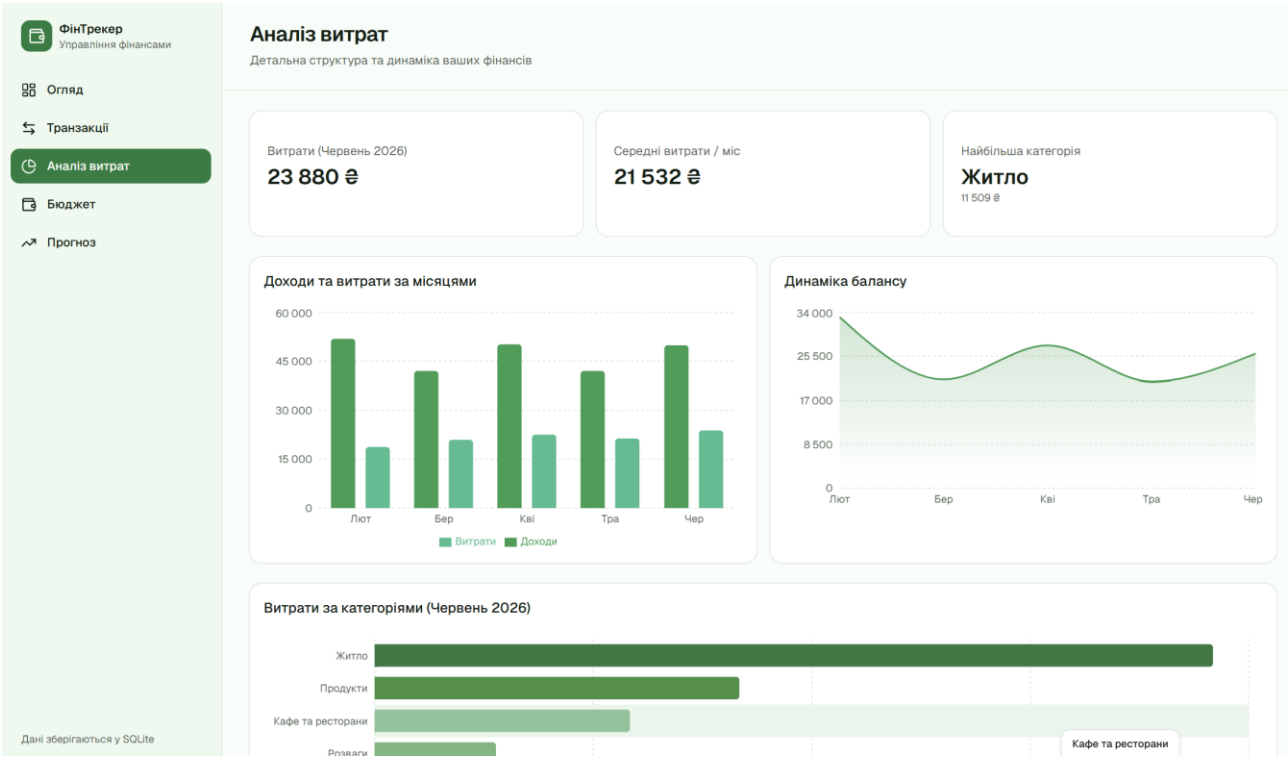


Рисунок Б.3 – Сторінка аналізу витрат

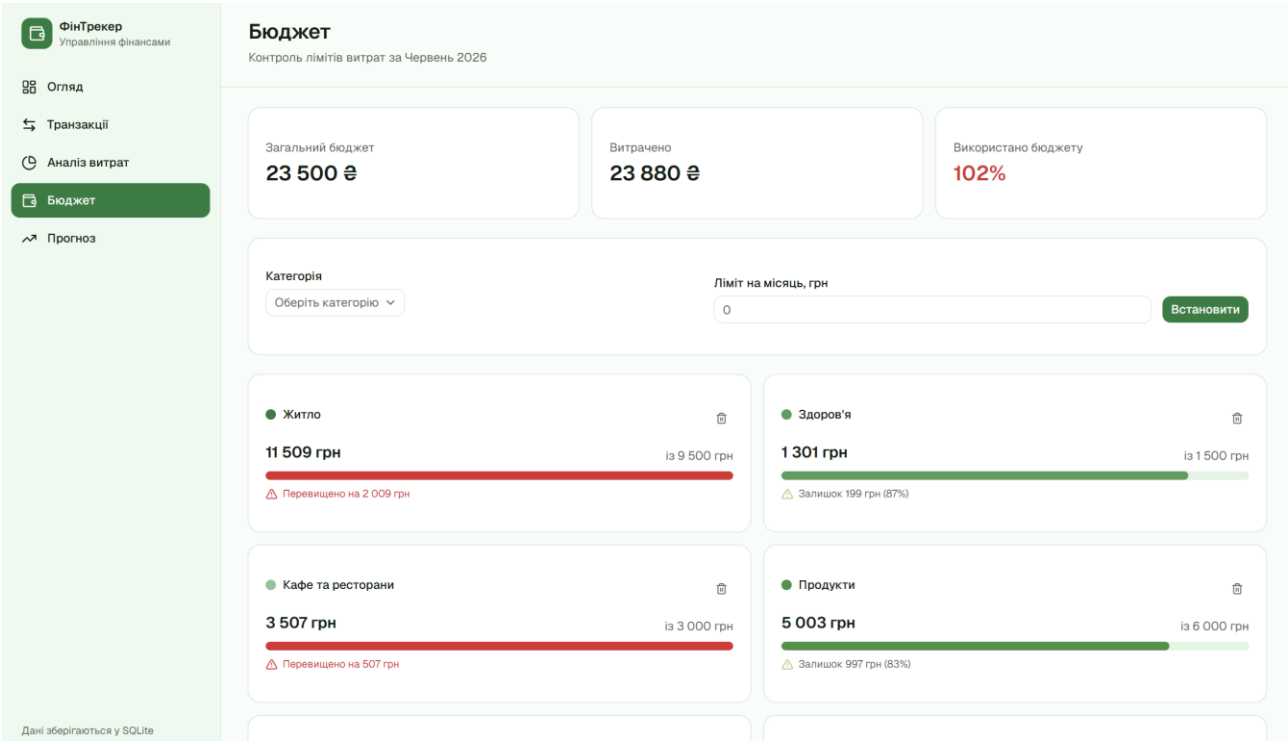


Рисунок Б.4 – Сторінка бюджету

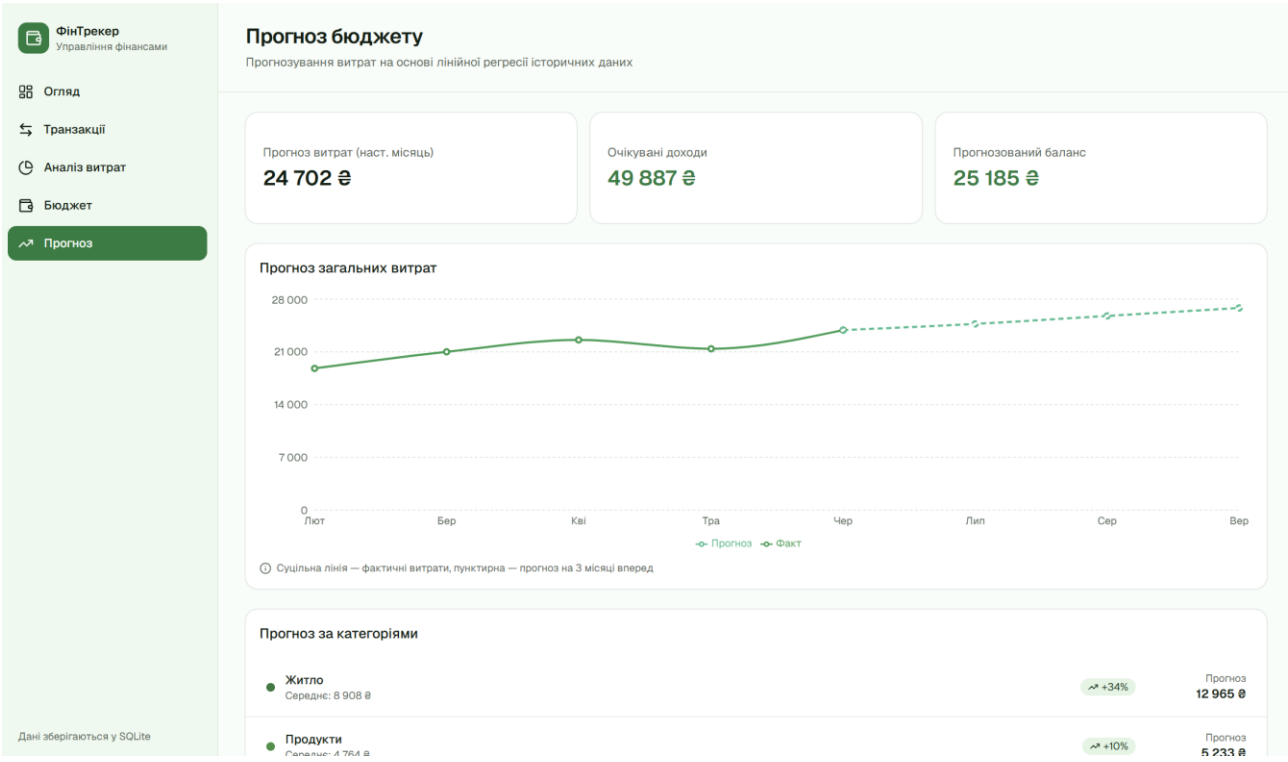


Рисунок Б.5 – Сторінка прогнозування

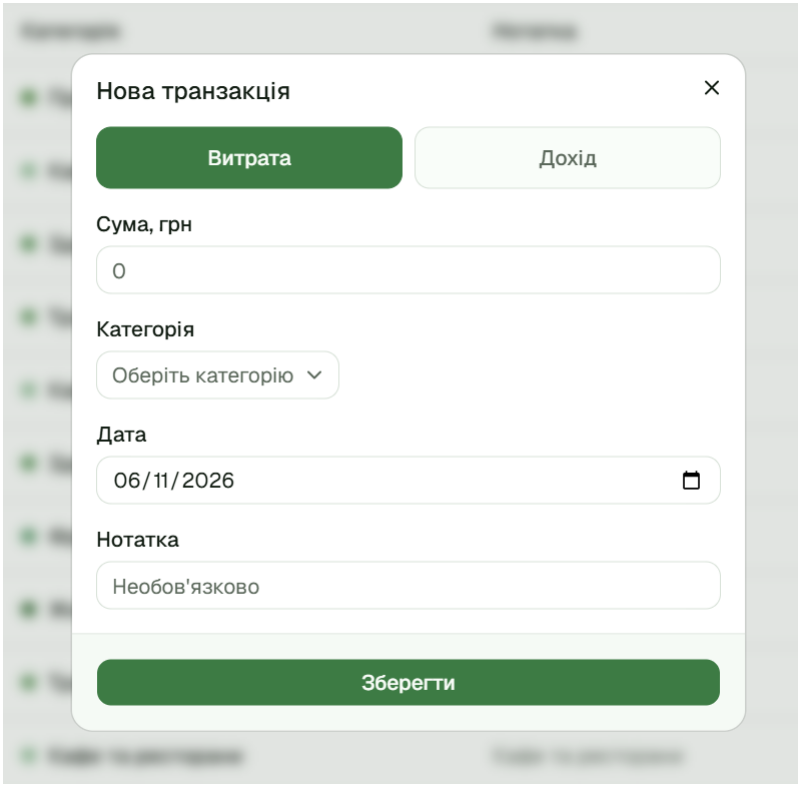


Рисунок Б.6 – Вікно створення транзакції