МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка додатку для ефективного контролю та візуалізації персональних фінансів**

Виконав: студент 4 курсу, групи 6.1212-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)

освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)

Д.А. Микуленко
(підпис, ініціали та прізвище)

Керівник к.т.н., доцент,
О.М. Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Рецензент директор ТОВ «Дісітел»

П.О. Лютий

Запоріжжя

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра Електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)

Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Тетяна КРИТСЬКА
“_01_” _березня_2026_р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Микуленку Данилу Андрійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка додатку для ефективного контролю та візуалізації персональних фінансів
керівник роботи Міхайлуца Олена Миколаївна, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 23.12.2025 р. № 1830-с
2. Строк подання студентом кваліфікаційної роботи 2 червня 2026 р.
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- аналіз предметної області та постановка задачі;
- проєктування веб-додатку;
- програмна реалізація та тестування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прий-няв

7. Дата видачі завдання 01.03.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи бакалавра	Примітка
1	Аналіз предметної області	01.03 – 20.03.2026	виконано
2	Формулювання задачі кваліфікаційної роботи	21.03 – 28.03.2026	виконано
3	Огляд існуючих рішень і аналогів	01.04 – 10.04.2026	виконано
4	Порівняльний аналіз аналогів	11.04 – 17.04.2026	виконано
5	Формування вимог до системи	18.04 – 24.04.2026	виконано
6	Проєктування архітектури веб-додатку	25.04 – 31.04.2026	виконано
7	Проєктування інформаційної моделі та бази даних	01.05 – 07.05.2026	виконано
8	Проєктування користувацького інтерфейсу	08.05 – 12.05.2026	виконано
9	Реалізація модулів автентифікації, категорій і транзакцій	13.05 – 24.05.2026	виконано
10	Реалізація дашборду, звітів, бюджетів і валют	25.05 – 01.06.2026	виконано
11	Тестування основних сценаріїв роботи	02.06 – 04.06.2026	виконано
12	Аналіз результатів і визначення напрямів розвитку	05.06 – 07.06.2026	виконано
13	Оформлення кваліфікаційної роботи	08.06 – 09.06.2026	виконано

Студент _____ Д.А.Микуленко
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М.Міхайлуца
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Сторінок: 78

Рисунків: 26

Таблиць: 11

Джерел: 23

Микуленко Д.А. Розробка додатку для ефективного контролю та візуалізації персональних фінансів: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О.М. Міхайлуца. Запоріжжя : ЗНУ, 2026. 78 с.

Мета роботи полягає у проєктуванні та реалізації веб-додатку для управління персональними фінансами, призначеного для ручного обліку доходів і витрат, роботи з категоріями, бюджетами, валютами, звітами та графічним поданням фінансових показників.

У роботі проаналізовано предметну область персонального фінансового обліку, веб-орієнтований підхід до ведення фінансових записів і вимоги до захисту користувацьких даних. Розглянуто існуючі програмні рішення, сформульовано задачу розроблення та визначено вимоги до системи.

Спроєктовано архітектуру веб-додатку, інформаційну модель, структуру бази даних і користувацький інтерфейс. Реалізовано систему на основі ASP.NET Core MVC, ASP.NET Core Identity, Entity Framework Core та SQLite. Основні модулі охоплюють автентифікацію та авторизацію, роботу з категоріями і транзакціями, дашборд, звіти, бюджети та валюти.

Виконано автоматизоване і ручне тестування основних сценаріїв роботи. Перевірено коректність введення, обробки, фільтрації, розрахунків, формування звітів і розмежування доступу до даних.

Ключові слова: *персональні фінанси, веб-додаток, asp.net core, sqlite, транзакції, категорії, бюджети, валюти, звіти, дашборд*.

SUMMARY

Pages: 78

Figures: 26

Tables: 11

Sources: 23

Mikulenko D.A. Development of an application for effective personal finance control and visualization: bachelor's qualification work of specialty 121 "Software Engineering" / scientific supervisor O.M. Mikhailutsa. Zaporizhzhia : ZNU, 2026. 78 p.

The aim of the work is to design and implement a web application for personal finance management intended for manual income and expense tracking, category management, budget management, currency management, report generation, and graphical presentation of financial indicators.

The work analyzes the subject area of personal financial accounting, the web-oriented approach to maintaining financial records, and the requirements for protecting user data. Existing software solutions are reviewed, the development task is formulated, and system requirements are defined.

The application architecture, information model, database structure, and user interface are designed. The system is implemented using ASP.NET Core MVC, ASP.NET Core Identity, Entity Framework Core, and SQLite. The main modules include authentication and authorization, category and transaction management, dashboard, reports, budgets, and currencies.

Automated and manual testing of the main system scenarios is performed. The correctness of data input, processing, filtering, calculations, report generation, and access control is verified.

Keywords: personal finance, web application, asp.net core, sqlite, transactions, categories, budgets, currencies, reports, dashboard.

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 10
1.1 Інформаційні процеси персонального фінансового обліку 10
1.3 Захист і розмежування доступу до фінансових даних 15
1.4 Огляд існуючих програмних рішень 18
1.5 Постановка задачі та вимоги до системи 23
1.6 Висновки до розділу 25
2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ 26
2.1 Вибір архітектурного підходу 26
2.2 Засоби реалізації веб-додатку 29
2.3 Проєктування інформаційної моделі та бази даних 32
2.4 Проєктування користувацького інтерфейсу 37
2.5 Висновки до розділу 42
3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 43
3.1 Реалізація автентифікації та авторизації 43
3.2 Реалізація модулів категорій і транзакцій 48
3.3 Реалізація дашборду, звітів і візуалізації даних 55
3.4 Реалізація механізму бюджетів та валют 63
3.5 Тестування основних сценаріїв роботи 68
3.6 Аналіз результатів роботи системи 70
3.7 Висновки до розділу 73
ВИСНОВКИ 75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 76

ВСТУП

Персональний фінансовий облік пов’язаний із фіксацією доходів, витрат, поточного балансу, регулярних платежів і бюджетних обмежень користувача. За відсутності системного обліку ускладнюється аналіз витрат за категоріями, контроль фінансового стану за періодами та виявлення перевищення встановлених лімітів.

Наявні програмні засоби для обліку персональних фінансів не завжди відповідають потребі ручного обліку через веб-інтерфейс. Частина таких рішень має комерційну модель доступу, потребує підписки або передбачає зберігання даних у хмарному середовищі. Окремі системи орієнтовані на банківську синхронізацію, автоматичний імпорт транзакцій і розширене фінансове планування. Для базового обліку доходів і витрат такі функції можуть бути надлишковими.

Актуальність теми полягає у потребі створення додатку для ефективного контролю та візуалізації персональних фінансів. Такий додаток має забезпечувати ручне внесення фінансових операцій, групування записів за категоріями, формування звітів, графічне подання показників і контроль бюджетних обмежень у межах однієї системи.

Метою роботи є проєктування та реалізація додатку для ефективного контролю та візуалізації персональних фінансів, призначеного для ведення доходів і витрат, роботи з категоріями, формування звітів, графічного подання фінансових показників і контролю бюджетних лімітів.

Для досягнення поставленої мети необхідно розв’язати такі завдання: проаналізувати предметну область персонального фінансового обліку; розглянути існуючі програмні рішення; визначити вимоги до системи; спроєктувати архітектуру веб-додатку, інформаційну модель і структуру сутностей; розробити користувацький інтерфейс; реалізувати модулі автентифікації, категорій, транзакцій, бюджетів, валют, інформаційної панелі

та звітів; виконати тестування основних сценаріїв роботи системи; визначити результати реалізації та напрями подальшого розвитку.

Об’єктом дослідження є процес обліку та контролю персональних фінансів користувача у веб-середовищі.

Предметом дослідження є методи, моделі даних і програмні засоби реалізації веб-додатку для ведення фінансових транзакцій, формування звітів, візуалізації фінансових показників і контролю витрат за категоріями.

Під час виконання роботи використано метод аналізу предметної області для визначення основних операцій персонального фінансового обліку. Метод порівняльного аналізу застосовано для розгляду існуючих програмних рішень. Методи об’єктно-орієнтованого проєктування використано для побудови структури веб-додатку та розподілу функцій між окремими модулями. Метод моделювання даних застосовано для визначення сутностей, їхніх атрибутів і зв’язків. Метод тестування використано для перевірки основних функціональних сценаріїв.

Новизна одержаних результатів полягає у розробленні веб-додатку для ручного персонального фінансового обліку, у якому транзакції, категорії, бюджети, валюти, інформаційна панель і звіти поєднано в межах єдиної інформаційної моделі з розмежуванням даних за обліковими записами користувачів.

Практичне значення одержаних результатів полягає у створенні веб-додатку, який забезпечує ведення доходів і витрат, роботу з категоріями, перегляд балансу, формування звітів, аналіз структури витрат і контроль бюджетних лімітів. Розроблений програмний продукт може бути використаний для індивідуального фінансового обліку.

Особистий внесок автора полягає в аналізі предметної області, формуванні вимог до системи, проєктуванні архітектури веб-додатку, інформаційної моделі, структури сутностей і користувацького інтерфейсу, а також у програмній реалізації основних функціональних модулів системи.

Результати роботи висвітлено у тезах за темою «Розробка веб-додатку для управління персональними фінансами з використанням ASP.NET Core та SQLite».

Глосарій

БД – база даних.

СУБД – система управління базами даних.

API – програмний інтерфейс додатку.

ASP.NET Core – платформа для розробки веб-додатків на основі .NET.

CRUD – операції створення, читання, оновлення та видалення даних.

CSV – текстовий формат подання табличних даних.

CSS – мова опису зовнішнього вигляду вебсторінок.

HTML – мова розмітки вебсторінок.

HTTP – протокол передавання гіпертексту.

HTTPS – захищений протокол передавання гіпертексту.

MVC – архітектурний шаблон Model-View-Controller.

PDF – формат електронного документа.

SQL – мова структурованих запитів.

UI – користувацький інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Інформаційні процеси персонального фінансового обліку

Персональний фінансовий облік у межах розроблюваної системи розглядається як процес збирання, збереження, групування та аналізу даних про рух коштів користувача. Такий облік охоплює доходи, витрати, залишок коштів, регулярні платежі та бюджетні обмеження. Персональні фінанси мають індивідуальний характер, оскільки фінансові рішення пов’язані з особистими доходами, потребами та витратами конкретної людини [1].

Основою такого обліку є фінансова операція. Вона фіксує окремий факт надходження або використання коштів. Для подальшої обробки операція повинна містити суму, дату, тип, категорію, валюту, короткий опис і належність до конкретного користувача. Без цих реквізитів запис не може бути повноцінно використаний для розрахунку підсумків, фільтрації та формування звітів.

Доходи і витрати є основними типами фінансових операцій. Доходи збільшують доступний залишок коштів. Витрати зменшують цей залишок і показують напрями використання грошей. До доходів можуть належати заробітна плата, стипендія, соціальні виплати, додатковий заробіток або повернення коштів. До витрат належать покупки товарів, комунальні платежі, транспорт, підписки, медичні послуги та інші щоденні платежі.

Накопичення фінансових операцій потребує впорядкування. Простий список записів не дає повного уявлення про фінансовий стан користувача. Для аналізу операції потрібно групувати за датою, типом, категорією, валютою та періодом. Таке групування є основою для визначення загальної суми доходів, витрат, балансу та структури платежів.

Категорії використовуються для змістового поділу фінансових операцій. Вони пов’язують окремі записи з конкретними напрямами доходів або витрат. Для витрат можуть використовуватися категорії харчування, транспорт, комунальні послуги, покупки, підписки та медицина. Для доходів можуть

використовуватися категорії заробітна плата, додатковий дохід, соціальні виплати або повернення коштів. У підручнику з фінансів витрати домогосподарства розглядаються через поділ за напрямами використання коштів, що підтверджує потребу у структурованому групуванні платежів [1].

Бюджетний контроль є окремим інформаційним процесом. Він передбачає порівняння фактичних витрат із встановленими обмеженнями. У веб-додатку таке обмеження може задаватися для конкретної витратної категорії та періоду. Після внесення операцій система повинна визначати використану частину ліміту і фіксувати перевищення, якщо фактичні витрати більші за встановлене значення.

Періодичний аналіз потрібний для оцінки змін у фінансовому стані користувача. Операції можуть розглядатися за день, місяць, рік або довільний проміжок часу. Період дає змогу порівнювати доходи і витрати, визначати повторювані платежі та оцінювати динаміку балансу. Саме тому дата операції є обов’язковим реквізитом фінансового запису.

Управління особистими фінансами пов’язане з регулярними фінансовими рішеннями користувача. У Національній стратегії розвитку фінансової грамотності до 2030 року фінансова культура пов’язується з фінансовими звичками, очікуваннями та щоденними фінансовими рішеннями [2]. Для персонального обліку це означає потребу у постійному оновленні записів і контролі фактичних фінансових показників.

Ручне ведення фінансових записів у зошиті, нотатках або електронній таблиці ускладнює подальшу обробку даних. Користувач повинен самостійно групувати операції, обчислювати підсумки, перевіряти повтори, порівнювати періоди та створювати таблиці або діаграми. За великої кількості записів зростає ризик помилок у підрахунках і пропуску окремих платежів.

Окремою проблемою є відсутність швидкого наочного подання результатів. Перелік транзакцій не дає змоги одразу оцінити поточний баланс, частку витрат за категоріями, зміну доходів і витрат за періодами та

наближення до бюджетного ліміту. Для цього потрібні підсумкові показники, звіти, графіки та інформаційна панель.

Практика використання цифрових інструментів підтверджує потребу в автоматизованому обліку. За результатами дослідження фінансової грамотності в Україні у 2021 році 68% опитаних планували доходи та витрати, 23% робили записи витрат, а 13% використовували банківський додаток або інший інструмент управління грошовими коштами для відстеження витрат [3].

Отже, інформаційні процеси персонального фінансового обліку охоплюють введення операцій, їх збереження, групування, розрахунок підсумків, контроль бюджетних обмежень і подання результатів у зручному вигляді. Для розроблюваного веб-додатку це визначає потребу у підтримці транзакцій, категорій, валют, бюджетів, періодів, звітів, графіків і прив’язки записів до конкретного користувача.

1.2 Веб-орієнтований підхід до обліку персональних фінансів

Персональний фінансовий облік передбачає регулярне внесення і перегляд фінансових записів. Користувач фіксує операції після покупки, оплати рахунку або отримання доходу, а також переглядає залишок коштів, витрати за категоріями та стан бюджетних обмежень за вибраний період. Для такого обліку потрібний доступ до системи без прив’язки до одного робочого місця.

Браузерний формат роботи відповідає цій задачі, оскільки основним середовищем взаємодії є веб-браузер. За джерелом [4], веб-програма «працює через веб-браузер» і розміщується не на пристрої користувача. Система персонального фінансового обліку в такому форматі не потребує встановлення окремої клієнтської програми на кожний комп’ютер або мобільний пристрій.

Доступ через браузер підтримує роботу з фінансовими записами з різних пристроїв. Користувач може внести витрату після здійснення платежу, перевірити залишок коштів або переглянути бюджетний ліміт за наявності

мережевого з’єднання. У навчальній літературі серед переваг веб-програм зазначено роботу з різними операційними системами, відсутність завантаження на комп’ютер і доступ «з будь-якого пристрою» [4].

Кросплатформність зменшує залежність системи від конкретної операційної системи. Один інтерфейс відкривається на різних пристроях за умови наявності браузера. Для обліку персональних фінансів це відповідає характеру щоденної роботи з даними, оскільки фінансові записи створюються і переглядаються в різних ситуаціях.

Мережевий формат роботи пов’язаний із клієнт-серверною організацією. У джерелі [4] веб-додаток визначено як клієнт-серверну програму, у якій клієнт взаємодіє із сервером за допомогою браузера, а зберігання даних переважно здійснюється на сервері. Така організація відокремлює інтерфейс користувача від обробки фінансових записів і зберігання даних.

Браузер відповідає за взаємодію користувача з формами, таблицями, фільтрами, звітами і графіками. Серверна частина обробляє запити, перевіряє введені дані, виконує розрахунки та звертається до сховища. Такий розподіл забезпечує однакову обробку транзакцій, категорій, бюджетів і валют незалежно від пристрою користувача.

Система персонального фінансового обліку через браузер має підтримувати короткі форми введення, таблиці фінансових операцій, фільтрацію записів, повідомлення про помилки та подання підсумкових показників. У джерелі [5] серед властивостей веб-додатків зазначено доступність з різних пристроїв, швидкість роботи, безпеку та зрозумілу взаємодію користувача із системою.

Основними елементами інтерфейсу є форми введення, таблиці транзакцій, фільтри та звіти. Форми використовуються для роботи з транзакціями, категоріями, бюджетами і валютами. Таблиці відображають фінансові записи. Фільтри відбирають записи за датою, типом, категорією або валютою. Звіти узагальнюють дані за періодами.

Інформаційна панель відображає поточний баланс, суму доходів, суму витрат, структуру витрат за категоріями та використання бюджетних лімітів. Вона подає основні фінансові показники без перегляду повного списку транзакцій.

Централізоване зберігання даних зменшує розпорошення фінансової інформації між окремими файлами, нотатками або локальними таблицями. У джерелі [5] серед властивостей веб-додатків виділено централізоване розміщення даних і доступність через браузер. Для персонального фінансового обліку це стосується транзакцій, категорій, бюджетів, валют і підсумкових показників.

Під час вибору підходу до розроблення враховується характер задачі. За результатами порівняння PWA і React Native, вибір технологічного підходу залежить від потреб конкретного проєкту, архітектури, продуктивності та користувацького досвіду [6]. Для розроблюваної системи основними сценаріями є введення, збереження, фільтрація, розрахунок і візуалізація фінансових даних. Ці сценарії не потребують обов’язкового доступу до апаратних можливостей пристрою.

Порівняння з мобільними підходами підтверджує доцільність браузерного доступу для системи, орієнтованої на роботу з даними. У джерелі [6] зазначено, що PWA використовує веб-технології та підтримує кросплатформний підхід, тоді як React Native орієнтований на ближчу до нативної мобільну реалізацію. Для системи персонального фінансового обліку пріоритетними є єдиний інтерфейс, доступ через браузер і централізоване опрацювання фінансових записів.

Основним обмеженням такого підходу є залежність від мережевого з’єднання. Перегляд і внесення записів потребують доступу до системи через браузер. У джерелі [4] залежність від доступу до Інтернету визначено як один із недоліків веб-програми.

Фінансові записи містять дані про доходи, витрати, категорії платежів, бюджети і валюти користувача. Тому система потребує автентифікації,

авторизації та розмежування доступу. Кожен користувач повинен працювати лише з власними фінансовими даними.

Браузерний підхід відповідає задачі обліку персональних фінансів завдяки доступу з різних пристроїв, централізованому зберіганню записів і поданню результатів у вигляді таблиць, звітів, графіків та інформаційної панелі. Ці характеристики враховуються під час подальшого проєктування інтерфейсу, структури даних і механізмів захисту користувацьких записів.

1.3 Захист і розмежування доступу до фінансових даних

Фінансові дані користувача належать до даних з обмеженим доступом. До них входять записи про доходи, витрати, категорії платежів, бюджети, валюти та підсумкові показники. Після прив’язки таких записів до облікового запису вони характеризують фінансову поведінку конкретної особи.

У Законі України «Про захист персональних даних» персональні дані визначено як відомості про фізичну особу, яка ідентифікована або може бути конкретно ідентифікована. Обробка таких даних охоплює їх збирання, реєстрацію, накопичення, зберігання, зміну, використання і знищення, зокрема з використанням інформаційних систем [7]. Це стосується веб-додатку, у якому фінансові записи зберігаються в базі даних і пов’язуються з конкретним користувачем.

Захист фінансових записів починається з автентифікації. Вона встановлює, хто саме працює із системою. У рекомендаціях OWASP автентифікацію описано як перевірку того, що користувач або інша сутність є тим, за кого себе видає [8]. У веб-додатку персонального фінансового обліку автентифікація відповідає входу користувача за обліковими даними.

Автентифікація не охоплює всю задачу захисту. Після входу користувач має отримати доступ тільки до тих даних, які належать його обліковому запису. Для цього використовується авторизація. У рекомендаціях OWASP

авторизацію визначено як перевірку того, чи дозволена запитувана дія для певної сутності [9].

Для фінансового веб-додатку авторизація має виконуватися на рівні конкретного запису. Користувач може переглядати власні транзакції, категорії, бюджети і валюти. Записи іншого користувача не повинні відкриватися, редагуватися або використовуватися в розрахунках. Тому перевірка доступу має враховувати факт входу та власника даних.

Один із основних ризиків у таких системах пов’язаний із порушенням контролю доступу. У переліку OWASP Top 10 помилки контролю доступу описуються як причина несанкціонованого розкриття, зміни або знищення даних. Серед прикладів наведено ситуацію, коли користувач отримує доступ до чужого облікового запису через зміну ідентифікатора в запиті [10]. Для фінансової системи аналогічний ризик виникає під час відкриття транзакції, бюджету або категорії за ідентифікатором.

Перевірка власника запису має виконуватися на серверному боці. Дані, передані в адресі сторінки, параметрі запиту або прихованому полі форми, не можна вважати достатніми для надання доступу. Сервер повинен зіставити запитаний запис із поточним користувачем. Якщо запис належить іншому обліковому запису, дія не виконується.

Розмежування доступу має охоплювати всі операції з фінансовими даними. Це стосується перегляду списків, відкриття деталей, створення, редагування і видалення записів. Такий самий підхід застосовується до звітів, інформаційної панелі і графіків, оскільки вони формуються на основі тих самих фінансових операцій.

У рекомендаціях OWASP для авторизації виділено принцип мінімальних привілеїв. Користувач або компонент системи повинен мати лише ті дозволи, які потрібні для виконання його задач [9]. У межах розроблюваного веб-додатку звичайний користувач працює тільки з власними фінансовими записами. Адміністративні або службові дії не належать до його стандартного набору прав.

Іншим принципом є заборона доступу за замовчуванням. У рекомендаціях OWASP цей підхід описано як deny by default [9]. У системі персонального фінансового обліку доступ до запису надається лише після явного підтвердження права користувача на дію. Якщо належність запису не підтверджена, операція має бути відхилена.

Перевірка доступу потрібна під час кожного звернення до захищених даних. Одного входу в систему недостатньо для безпечної роботи з фінансовими записами. Кожна спроба перегляду, редагування, видалення або використання запису в аналітичному запиті повинна виконуватися з урахуванням поточного користувача.

Звіти і графіки також належать до захищених результатів обробки даних. Вони можуть не показувати окремі транзакції, але відображають підсумки доходів, витрат, бюджетів і категорій. Якщо до розрахунку потраплять чужі записи, буде порушено конфіденційність і сформовано неправильний результат.

Захист фінансових даних також пов’язаний із перевіркою введених значень. Форми транзакцій, категорій, бюджетів і валют мають приймати тільки коректні дані. Окремо перевіряються пов’язані об’єкти. Наприклад, вибрана категорія або бюджет мають належати тому самому користувачу, який створює або змінює запис.

У межах цієї роботи захист фінансових даних визначає потребу в облікових записах користувачів, вході до системи, завершенні сеансу, захищених сторінках і прив’язці фінансових записів до власника. Ці положення враховуються під час формування вимог до веб-додатку та подальшого проєктування його інформаційної моделі.

1.4 Огляд існуючих програмних рішень

Для аналізу обрано програмні рішення, які використовуються для персонального фінансового обліку, бюджетування, перегляду витрат і формування фінансових показників. До огляду включено YNAB, Wallet by BudgetBakers і Money Manager Ex. Ці рішення відрізняються способом використання, моделлю зберігання даних, рівнем автоматизації та складом функціональних можливостей.

YNAB є веб-сервісом і мобільним застосунком для бюджетування та контролю витрат. Основна логіка сервісу побудована навколо розподілу коштів за бюджетними категоріями, контролю залишків і регулярного перегляду фінансового стану. На офіційній сторінці функцій YNAB зазначено можливість підключення банківських рахунків, автоматичного імпорту транзакцій, доступу з комп’ютера, смартфона або планшета, а також синхронізації між пристроями [11].

Переваги YNAB:

- підтримка веб-версії та мобільних застосунків;
- бюджетування за категоріями;
- автоматичний імпорт транзакцій через банківське підключення;
- синхронізація між комп’ютером, телефоном і планшетом;
- наявність звітів і графічного подання фінансових показників.

Недоліки YNAB:

- комерційна модель доступу;
- орієнтація на власну методику бюджетування;
- надлишковість банківської синхронізації для простого ручного обліку;
- складніший функціональний склад порівняно з мінімальною системою обліку доходів, витрат, категорій і бюджетів.

Інтерфейс сервісу YNAB подано на рисунку 1.1.

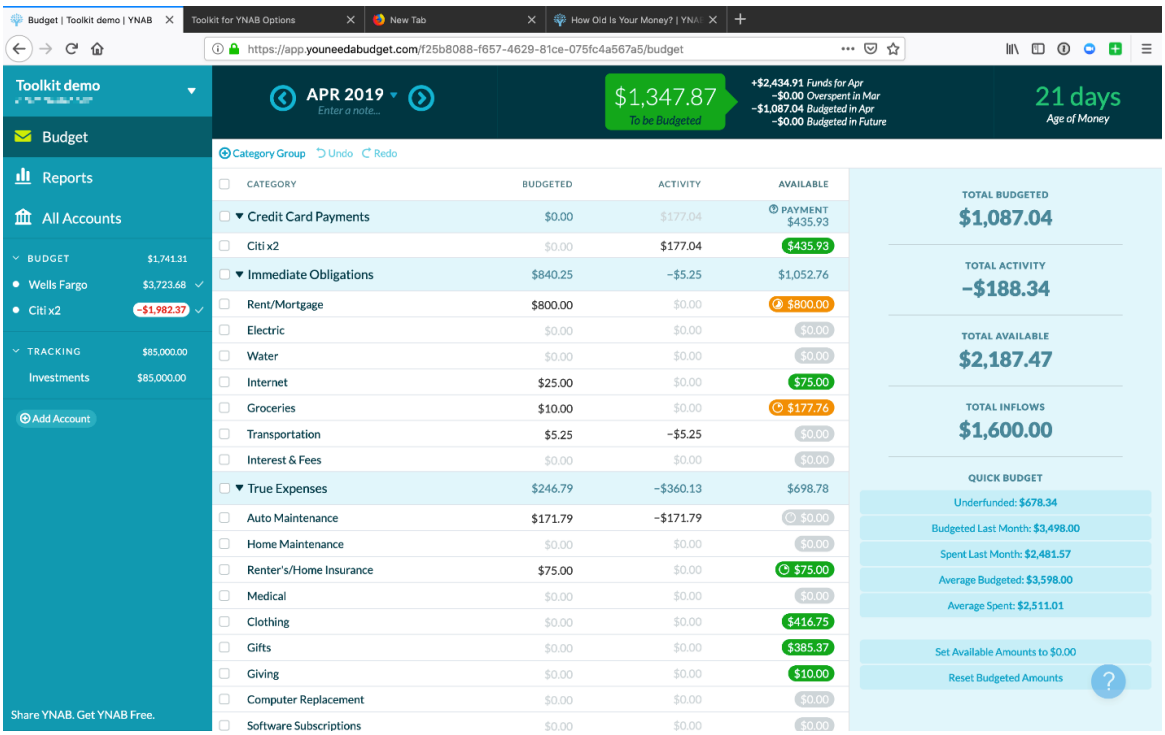


Рисунок *1.1 – Інтерфейс сервісу YNAB*

YNAB є прикладом системи, у якій фінансовий облік поєднано з бюджетуванням і візуальним аналізом показників. Для розроблюваного веб-додатку доцільно врахувати підхід до категоризації витрат, контролю бюджетів і подання зведених показників. Банківська синхронізація та складна методика планування не входять до меж розроблюваної системи.

Wallet by BudgetBakers є веб-орієнтованим і мобільним рішенням для персонального та сімейного фінансового обліку (див. рис. 1.2). Сервіс поєднує відстеження витрат, бюджетування, планові платежі, аналіз грошового потоку, автоматичну категоризацію та банківську синхронізацію. На офіційній сторінці продукту Wallet зазначено підтримку smart budgets, expense tracking, planned payments, cash flow insights, bank sync та API integrations [12].

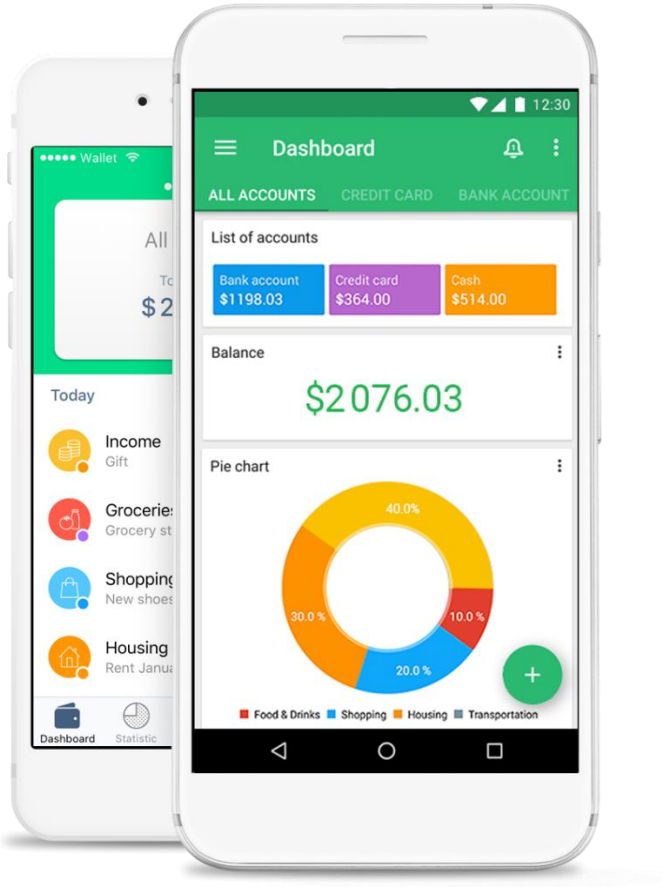


Рисунок *1.2 – Інтерфейс сервісу Wallet by BudgetBakers*

Переваги Wallet by BudgetBakers:

- підтримка веб-версії та мобільних застосунків;
- ведення витрат і бюджетів;
- планові платежі та нагадування;
- аналіз грошового потоку;
- автоматична категоризація транзакцій;
- банківська синхронізація.

Недоліки Wallet by BudgetBakers:

- значна частина функціональності пов’язана з банківською синхронізацією;
- автоматизація обліку є надлишковою для простої системи ручного введення;
- сервіс орієнтований на хмарну модель і зовнішні інтеграції;

Money Manager Ex є безкоштовним кросплатформним застосунком з відкритим кодом для персонального фінансового обліку. Офіційний сайт описує його як програмне забезпечення для організації фінансів і відстеження того, куди, коли і кому спрямовуються кошти. Система підтримує рахунки, валюти, транзакції, одержувачів платежів, категорії, заплановані операції, активи, бюджети, звіти та графіки [13].

Переваги Money Manager Ex:

- безкоштовне програмне забезпечення;
- відкритий код;
- кросплатформність;
- підтримка рахунків, валют, транзакцій і категорій;
- наявність бюджетів, запланованих операцій, звітів і графіків;
- орієнтація на ручний персональний облік.

Недоліки Money Manager Ex:

- основний формат використання пов’язаний із локальним застосунком;
- інтерфейс ближчий до настільного фінансового менеджера;
- для доступа з різних пристроїв потрібні додаткові механізми синхронізації;

Інтерфейс Money Manager Ex подано на рисунку 1.3.

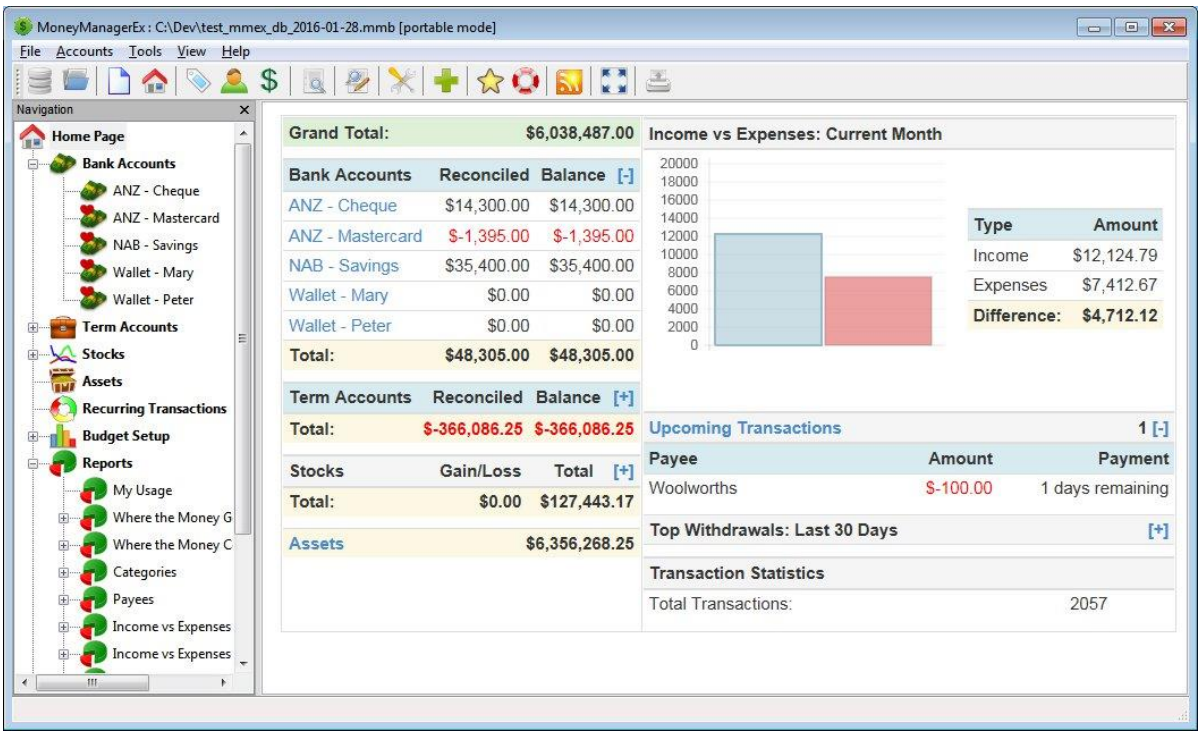


Рисунок *1.3 – Інтерфейс сервісу Money Manager Ex*

Money Manager Ex є близьким до задачі ручного персонального фінансового обліку. Його функціональність підтверджує доцільність підтримки транзакцій, категорій, валют, бюджетів і звітів. Водночас локальна модель використання не відповідає обраному напряму розроблення, за якого користувач працює із системою через браузер.

Розглянуті рішення охоплюють різні підходи до персонального фінансового обліку. YNAB орієнтований на бюджетування та планування витрат. Wallet by BudgetBakers робить акцент на автоматизації, банківській синхронізації та аналізі грошового потоку. Money Manager Ex представляє локальний підхід із відкритим кодом і розвиненим набором облікових функцій.

1.5 Постановка задачі та вимоги до системи

Задача роботи полягає у розробленні веб-додатку для ручного персонального фінансового обліку. Система призначена для збереження фінансових операцій користувача, обліку доходів і витрат, групування записів за категоріями, контролю бюджетних обмежень та формування звітів.

Предметом автоматизації є процес ведення персональних фінансових записів. Основною одиницею обліку є фінансова транзакція. Вона містить суму, дату, тип операції, категорію, валюту, спосіб оплати та коментар. Записи належать конкретному користувачу і використовуються для розрахунку балансу, доходів, витрат, бюджетних показників і звітів.

Суб’єктом взаємодії із системою є користувач. Основні варіанти використання веб-додатку наведено на рисунку 1.4.

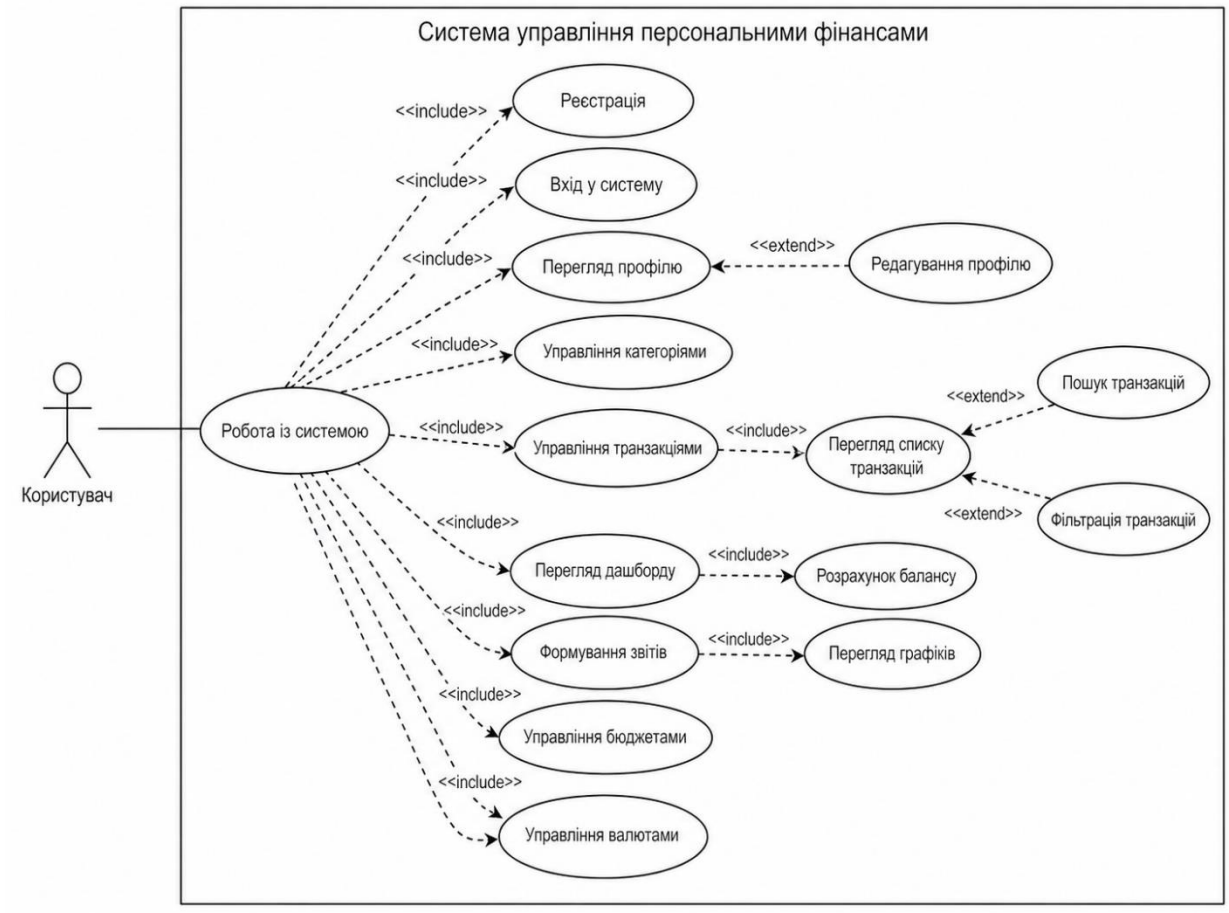


Рисунок *1.4 – Діаграма варіантів використання веб-додатку*

Діаграма відображає основні дії користувача на верхньому рівні: реєстрацію, вхід до системи, завершення сеансу, ведення категорій, ведення транзакцій, роботу з бюджетами, роботу з валютами, перегляд дашборду та формування звітів.

Вхідними даними системи є реєстраційні дані користувача, облікові дані для входу, назви категорій, параметри фінансових транзакцій, бюджетні ліміти, назви валют і курси валют до гривні.

Вихідними даними системи є списки транзакцій, перелік категорій, розрахований баланс, суми доходів і витрат, гривневий еквівалент операцій, стан бюджетних обмежень, звіти за періодом, графіки та зведені показники дашборду.

Для розв’язання поставленої задачі система має забезпечувати:

- реєстрацію користувача, вхід до системи та завершення сеансу;
- ведення категорій доходів і витрат;
- ведення фінансових транзакцій;
- пошук, фільтрацію та посторінковий перегляд транзакцій;
- розрахунок доходів, витрат і балансу за вибраний період;
- формування звітів і графічне подання фінансових показників;
- створення та контроль бюджетних обмежень;
- ведення валют користувача і розрахунок гривневого еквівалента операцій;
- обмеження доступу користувача до власних фінансових даних.

Під час обробки даних мають виконуватися такі правила:

- транзакція повинна мати суму, дату, тип операції, категорію та валюту;
- категорія транзакції повинна відповідати типу операції;
- бюджетне обмеження створюється для витратної категорії;
- суми транзакцій, бюджетні ліміти та курси валют повинні мати коректні числові значення;

- звіти, дашборд і бюджетні розрахунки формуються лише за записами поточного користувача.

Робота з сайтом має відповідати таким умовам:

- основні сторінки доступні тільки після входу користувача;
- фінансові записи різних користувачів ізолюються між собою;
- робота з сайтом виконується через сучасний браузер;
- сторінки коректно відображаються на екранах різного розміру;
- таблиці фінансових записів підтримують фільтрацію та поділ на сторінки;
- повідомлення про помилки є зрозумілими для користувача.

1.6 Висновки до розділу

У першому розділі проаналізовано предметну область персонального фінансового обліку. Визначено основні інформаційні об’єкти системи: фінансові транзакції, категорії, бюджети, валюти, звіти та користувацькі дані.

Обґрунтовано доцільність веб-орієнтованого підходу. Такий формат забезпечує доступ через браузер, роботу з різних пристроїв, централізоване зберігання записів і можливість подання результатів у вигляді таблиць, звітів та графіків.

Розглянуто вимоги до захисту фінансових даних. Встановлено потребу в автентифікації, авторизації, прив’язці записів до користувача та обмеженні доступу до чужих даних.

Проаналізовано існуючі рішення: YNAB, Wallet by BudgetBakers і Money Manager Ex. Визначено, що вони підтримують персональний фінансовий облік, але мають обмеження для задачі простого ручного веб-обліку.

На основі аналізу сформульовано задачу розроблення веб-додатку. Система має забезпечувати ручне ведення доходів і витрат, роботу з категоріями, бюджетами та валютами, формування звітів, графічне подання показників і захищений доступ до персональних фінансових записів.

2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ

2.1 Вибір архітектурного підходу

Веб-додаток для управління персональними фінансами реалізується як серверний веб-застосунок із клієнт-серверною взаємодією. Клієнтська частина представлена веб-браузером користувача. Серверна частина приймає HTTP-запити, обробляє введені дані, виконує прикладну логіку та повертає відповідь у вигляді веб-сторінки.

За способом організації застосунок належить до монолітних веб-додатків. Основні компоненти розміщуються в межах одного програмного застосунку. У межах такого застосунку поєднано обробку запитів, формування сторінок, прикладну логіку, доступ до даних, роботу з обліковими записами та взаємодію з базою даних.

Монолітна організація не означає змішування всіх частин програми в одному рівні. Усередині веб-додатку використовується логічний поділ на рівень представлення, рівень контролерів, сервісний рівень, рівень доступу до даних і рівень зберігання даних. Такий поділ відокремлює інтерфейс користувача, обробку запитів, прикладну логіку та операції з базою даних.

Основою логічної організації серверної частини є архітектурний шаблон MVC. Він передбачає поділ веб-застосунку на модель, представлення та контролер. Модель описує дані предметної області. Представлення формує сторінку інтерфейсу. Контролер приймає запит користувача і визначає подальшу обробку.

У матеріалі D. Krishan наведено схему взаємодії компонентів MVC, де показано зв’язок між моделлю, представленням і контролером [14]. Схему архітектурного шаблону MVC подано на рисунку 2.1.

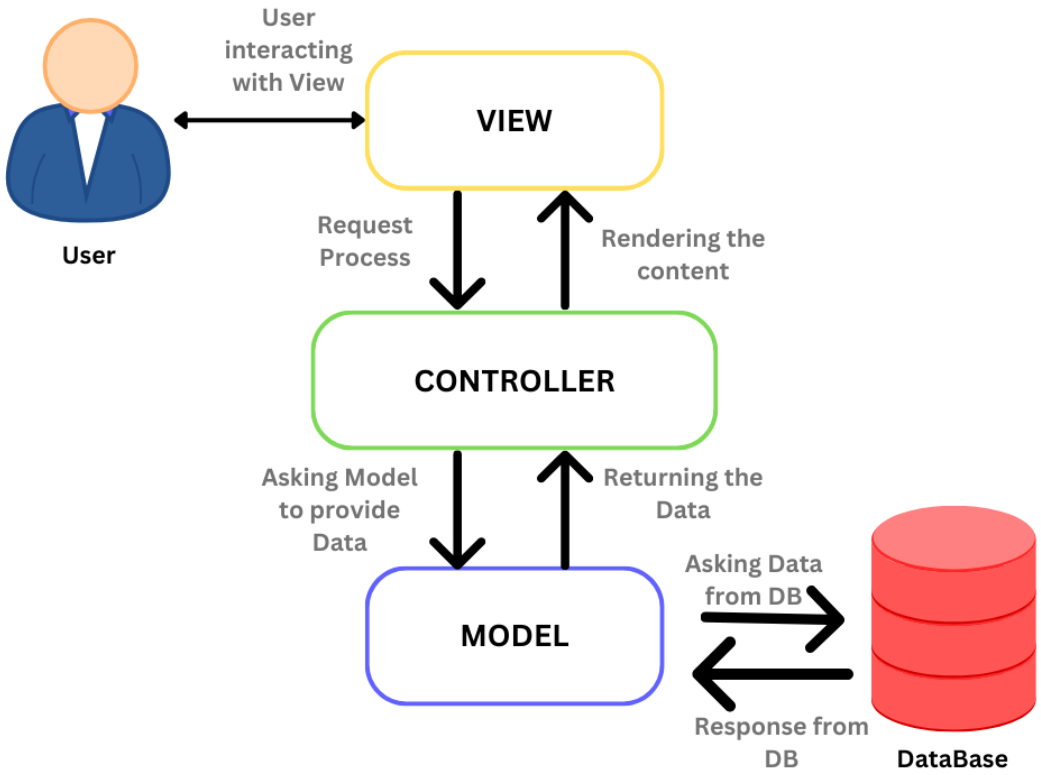


Рисунок 2.1 – *Схема архітектурного шаблону MVC*

На рисунку 2.1 показано базову взаємодію компонентів MVC. Запит користувача надходить до контролера. Контролер обробляє запит і взаємодіє з моделлю. Модель містить дані та правила їх подання в межах предметної області. Представлення отримує підготовлені дані та формує сторінку для користувача.

Для задачі персонального фінансового обліку базового поділу на модель, представлення і контролер недостатньо. Контролери не повинні містити розрахунок балансу, перевірку бюджетних обмежень, підготовку звітів і перевірку належності записів користувачу. Ці операції відносяться до прикладної логіки.

Прикладна логіка розміщується на сервісному рівні. На цьому рівні виконуються розрахунок доходів і витрат, визначення балансу, контроль бюджетних лімітів, підготовка звітних показників, обробка валютних значень

і перевірка доступу до фінансових записів. Сервісний рівень відокремлює правила предметної області від контролерів і сторінок інтерфейсу.

Рівень контролерів відповідає за приймання запитів і вибір сценарію обробки. Контролер отримує параметри запиту або дані форми, передає їх до сервісного рівня та повертає результат користувачу. Основні фінансові розрахунки на цьому рівні не виконуються.

Рівень представлення відповідає за відображення даних. До нього належать сторінки, форми, таблиці, фільтри, повідомлення і графічні блоки. Представлення працює з підготовленими даними і не звертається безпосередньо до бази даних.

Для передавання даних між контролерами та представленнями використовуються моделі подання. Вони містять набір полів, потрібних для конкретної сторінки, форми, таблиці або звіту. Моделі подання не є сутностями бази даних і не визначають структуру зберігання інформації.

Моделі предметної області описують основні об’єкти системи. До них належать користувач, фінансова транзакція, категорія, бюджет і валюта

Моделі описують основні сутності предметної області. До них належать користувач, категорія, транзакція, бюджет і валюта користувача. Моделі визначають структуру даних, їх атрибути та зв’язки.

Рівень доступу до даних забезпечує операції створення, читання, оновлення та видалення записів. Він відокремлює прикладну логіку від безпосередньої роботи з базою даних. Рівень зберігання даних містить таблиці та зв’язки, потрібні для збереження користувачів, транзакцій, категорій, бюджетів і валют.

Окремо в архітектурі виділяється компонент облікових записів і доступу. Він відповідає за реєстрацію користувача, вхід до системи, завершення сеансу та перевірку доступу до захищених сторінок. Через цей компонент фінансові записи пов’язуються з конкретним користувачем.

Загальну архітектуру веб-додатку подано на рисунку 2.2.

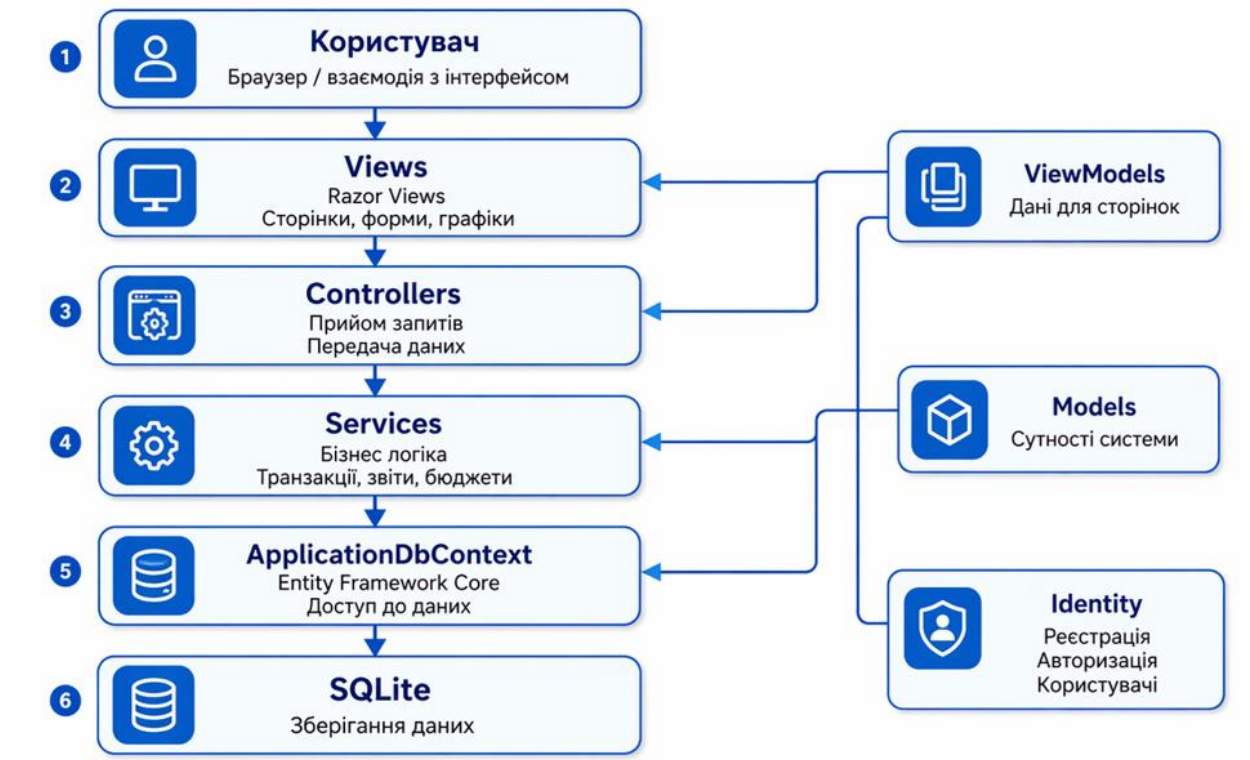


Рисунок 2.2 – *Загальна архітектура вебдодатку*

На рисунку 2.2 показано логічні рівні веб-додатку та напрям передавання даних між ними. Користувач працює із системою через браузер. Запит надходить до контролера. Контролер передає дані до сервісного рівня. Сервіс виконує прикладну логіку та звертається до рівня доступу до даних. Після отримання результату дані передаються до моделі подання і відображаються у представленні.

2.2 Засоби реалізації веб-додатку

Для реалізації веб-додатку обрано стек засобів, орієнтований на серверне формування сторінок, роботу з реляційною базою даних, автентифікацію користувачів, адаптивний браузерний інтерфейс і графічне подання фінансових показників. Склад засобів реалізації наведено в таблиці 2.1.

Таблиця 2.1 – *Засоби реалізації веб-додатку*

Група засобів	**Засоби**	**Призначення**
Серверна частина	C#, .NET, ASP.NET Core MVC	Обробка запитів, виконання прикладної логіки, формування сторінок
Облікові записи	ASP.NET Core Identity	Реєстрація, вхід, завершення сеансу, перевірка доступу
Доступ до даних	Entity Framework Core	Зв’язок моделей із базою даних, виконання запитів, міграції
Зберігання даних	SQLite	Збереження облікових записів і фінансових записів
Представлення	Razor Views	Формування HTML-сторінок на сервері
Клієнтська частина	HTML, CSS, JavaScript	Структура сторінок, оформлення, базова інтерактивність
Адаптивний інтерфейс	Bootstrap	Адаптація сторінок до різних розмірів екрана, компонування форм і таблиць
Візуалізація даних	Chart.js	Побудова діаграм і графіків для дашборду та звітів
Керування залежностями	NuGet	Підключення пакетів .NET-проєкту

Наведений набір засобів реалізації поділено за функціональними групами. Серверна частина відповідає за обробку запитів і виконання правил предметної області. Засоби доступу до даних забезпечують роботу з реляційною базою. Клієнтські засоби формують структуру сторінок, адаптивне відображення та графічне подання показників.

C# обрано як основну мову програмування серверної частини. Мова використовується для опису моделей, контролерів, сервісів, правил обробки даних і допоміжних класів. C# працює в середовищі .NET і підтримує об’єктно-орієнтовану організацію програмного коду [15].

.NET обрано як платформу виконання серверної частини. Платформа забезпечує запуск програми, роботу з бібліотеками, керування залежностями

та використання компонентів для побудови сайтів і серверних програмних систем. На її основі працюють засоби серверної частини, доступу до даних і автентифікації.

ASP.NET Core MVC обрано для побудови серверної частини сайту за шаблоном Model-View-Controller. Засіб використовується для обробки HTTP-запитів, роботи з контролерами, моделями та представленнями. У документації Microsoft ASP.NET Core MVC описано як фреймворк для створення сайтів і API із застосуванням шаблону MVC [16].

ASP.NET Core Identity обрано для роботи з обліковими записами користувачів. Засіб призначений для реєстрації, входу до системи, завершення сеансу, збереження хешів паролів і перевірки доступу до захищених частин системи. Офіційна документація визначає Identity як засіб додавання функцій входу користувача до програмних рішень на основі ASP.NET Core [17].

Entity Framework Core обрано для доступу до даних через об’єктно-реляційне відображення. Він забезпечує роботу з базою даних через класи моделей, виконання запитів, збереження змін і створення міграцій. У документації EF Core описано як легковагову, розширювану, відкриту і кросплатформну технологію доступу до даних, яка забезпечує роботу з базою через .NET-об’єкти [18].

SQLite обрано як реляційну базу даних для зберігання облікових записів, категорій, транзакцій, бюджетів, валют і службових даних. SQLite є вбудованим SQL-рушієм без окремого серверного процесу та без потреби в окремому налаштуванні сервера бази даних [19].

Razor Views обрано для серверного формування HTML-сторінок. Представлення використовуються для виведення форм, списків, таблиць, фільтрів, інформаційної панелі і звітів. Razor поєднує HTML-розмітку з C#-кодом, що забезпечує формування сторінок на основі підготовлених сервером даних [20].

HTML, CSS і JavaScript обрано як базові засоби клієнтської частини. HTML визначає структуру сторінок. CSS задає оформлення елементів

інтерфейсу. JavaScript використовується для базової інтерактивності та передавання підготовлених даних до графічних елементів.

Bootstrap обрано для адаптивного компонування сторінок. Його роль у проєкті пов’язана з коректним відображенням інтерфейсу на різних розмірах екрана, зокрема на персональних комп’ютерах, ноутбуках, планшетах і смартфонах. Засіб також використовується як основа для однакового оформлення форм, таблиць, кнопок, навігаційних елементів і карток [21].

Chart.js обрано для графічного подання фінансових показників. Бібліотека використовується для побудови діаграм структури витрат, співвідношення доходів і витрат та динаміки показників за періодами. Такий засіб відповідає задачі подання агрегованих фінансових даних в інформаційній панелі та звітах [22].

NuGet обрано для керування пакетами .NET-проєкту. Через нього підключаються бібліотеки, необхідні для серверної частини, доступу до даних, роботи з базою даних і автентифікації. NuGet є стандартним менеджером пакетів для .NET-проєктів [23].

2.3 Проєктування інформаційної моделі та бази даних

Інформаційна модель вебдодатку визначає склад даних, які потрібні для ведення персонального фінансового обліку. До моделі належать обліковий запис користувача, категорії доходів і витрат, фінансові транзакції, бюджетні обмеження та валюти користувача.

На схемі показано основні сутності системи, їхні атрибути та зв’язки (див. рис. 2.3). Центральною сутністю є користувач. З користувачем пов’язані категорії, транзакції, бюджети та валюти. Така структура забезпечує зберігання фінансових записів окремо для кожного облікового запису.

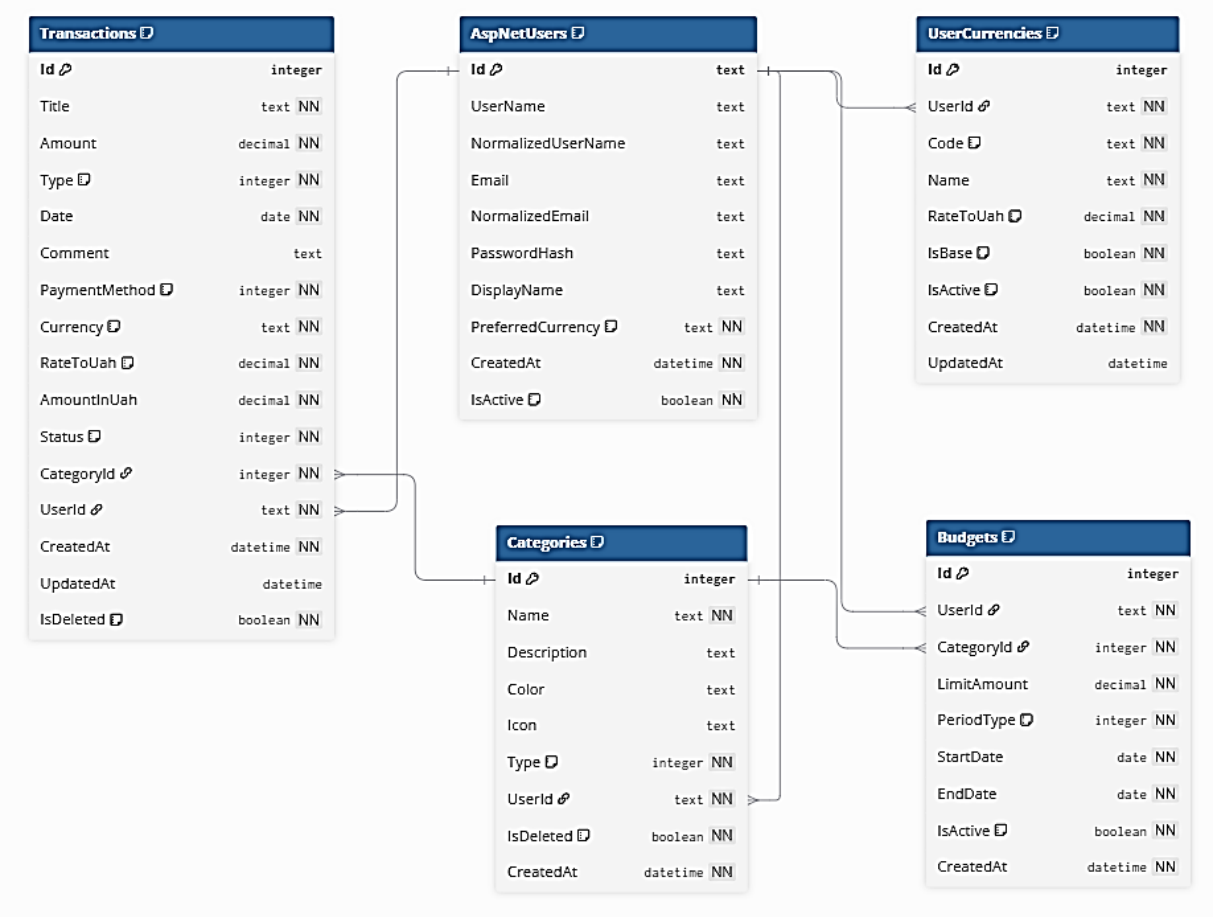


Рисунок *2.3 – Схема бази даних веб-додатку*

Сутність користувача зберігає дані облікового запису. Вона містить ідентифікатор, ім’я користувача, електронну пошту, хеш пароля, ім’я для відображення, основну валюту, дату створення та ознаку активності. Структуру сутності користувача наведено в таблиці 2.2.

Таблиця 2.2 – Структура сутності користувача

Поле	Тип поля	Опис
Id	TEXT	Ідентифікатор користувача
UserName	TEXT	Ім’я користувача
Email	TEXT	Електронна пошта
PasswordHash	TEXT	Хеш пароля
DisplayName	TEXT	Ім’я для відображення в інтерфейсі
PreferredCurrency	TEXT	Основна валюта користувача

CreatedAt	TEXT	Дата створення облікового запису
IsActive	INTEGER	Ознака активності користувача

Категорія використовується для групування фінансових операцій за напрямами доходів або витрат. Категорія належить конкретному користувачу і містить назву, опис, тип, колір, позначення, дату створення та ознаку логічного видалення. Структуру сутності категорії наведено в таблиці 2.3.

Таблиця 2.3 – *Структура сутності категорії*

Поле	**Тип поля**	**Опис**
Id	INTEGER	Ідентифікатор категорії
Name	TEXT	Назва категорії
Description	TEXT	Опис категорії
Color	TEXT	Колір візуального позначення
Icon	TEXT	Позначення категорії
Type	INTEGER	Тип категорії
UserId	TEXT	Ідентифікатор користувача
IsDeleted	INTEGER	Ознака логічного видалення
CreatedAt	TEXT	Дата створення категорії

Фінансова транзакція фіксує окрему операцію користувача. Вона використовується для збереження даних про надходження або витрачання коштів, подальшого групування записів за категоріями, розрахунку підсумків і формування звітних показників. Транзакція є основною обліковою сутністю системи, оскільки на її основі визначаються суми доходів, витрат, баланс, структура платежів і стан бюджетних обмежень.

Сутність фінансової транзакції містить назву, суму, тип операції, дату, коментар, спосіб оплати, валюту, курс до гривні, суму у гривневому еквіваленті, статус, категорію, користувача, дату створення та ознаку логічного видалення. Поля валюти, курсу та гривневого еквівалента потрібні для приведення операцій у різних валютах до єдиного розрахункового значення.

Поля категорії та користувача забезпечують групування записів і розмежування даних між обліковими записами.

Таблиця 2.4 – Структура сутності фінансової транзакції

Поле	Тип поля	Опис
Id	INTEGER	Ідентифікатор транзакції
Title	TEXT	Назва операції
Amount	TEXT	Сума в початковій валюті
Type	INTEGER	Тип операції
Date	TEXT	Дата операції
Comment	TEXT	Коментар
PaymentMethod	INTEGER	Спосіб оплати
Currency	TEXT	Код валюти
RateToUah	TEXT	Курс до гривні
AmountInUah	TEXT	Сума у гривневому еквіваленті
Status	INTEGER	Статус операції
CategoryId	INTEGER	Ідентифікатор категорії
CreatedAt	TEXT	Дата створення
UserId	TEXT	Ідентифікатор користувача
IsDeleted	INTEGER	Ознака логічного видалення

Бюджетне обмеження використовується для контролю витрат за категорією. Бюджет належить користувачу, пов’язується з витратною категорією та містить суму ліміту, тип періоду, дату початку, дату завершення й ознаку активності. Структуру сутності бюджету наведено в таблиці 2.5.

Таблиця 2.5 – *Структура сутності бюджету*

Поле	Тип поля	Опис
Id	INTEGER	Ідентифікатор бюджету
UserId	TEXT	Ідентифікатор користувача
CategoryId	INTEGER	Ідентифікатор категорії
LimitAmount	TEXT	Сума ліміту

PeriodType	INTEGER	Тип періоду
StartDate	TEXT	Дата початку періоду
EndDate	TEXT	Дата завершення періоду
IsActive	INTEGER	Ознака активності

Валюта користувача зберігає перелік валют, доступних для фінансових операцій. Сутність містить код валюти, назву, курс до гривні, ознаку базової валюти, ознаку активності, дату створення та дату редагування. Структуру сутності валюти користувача наведено в таблиці 2.6.

Таблиця 2.6 – Структура сутності валюти користувача

Поле	**Тип поля**	**Опис**
Id	INTEGER	Ідентифікатор валюти
UserId	TEXT	Ідентифікатор користувача
Code	TEXT	Код валюти
Name	TEXT	Назва валюти
RateToUah	TEXT	Курс до гривні
IsBase	INTEGER	Ознака базової валюти
IsActive	INTEGER	Ознака активності
CreatedAt	TEXT	Дата створення
UpdatedAt	TEXT	Дата редагування

Зв’язки між сутностями побудовано навколо облікового запису користувача. Один користувач може мати багато категорій, транзакцій, бюджетів і валют. Кожна категорія, транзакція, бюджет і валюта належать одному користувачу.

Категорія пов’язана з фінансовими транзакціями. Одна категорія може використовуватися в багатьох транзакціях. Кожна транзакція має одну категорію. Такий зв’язок забезпечує групування доходів і витрат за напрямами обліку.

Категорія також пов’язана з бюджетними обмеженнями. Один бюджет створюється для однієї витратної категорії. Такий зв’язок потрібний для розрахунку використаної частини ліміту за вибраним напрямом витрат.

Валютні дані зберігаються у двох частинах моделі. Сутність валюти користувача містить доступні валюти та їхні поточні курси. Сутність транзакції фіксує валюту, курс і гривневий еквівалент на момент створення фінансового запису. Це зберігає історичні значення операцій незалежно від подальшої зміни курсу.

У структурі передбачено логічне видалення для категорій і транзакцій. Ознака логічного видалення дає змогу не вилучати запис фізично з бази даних. Такі записи можуть не відображатися в основних списках, але залишаються в структурі зберігання.

2.4 Проєктування користувацького інтерфейсу

Користувацький інтерфейс веб-додатку подано у форматі браузерного графічного інтерфейсу. Загальна структура сторінки складається з верхньої навігаційної області, центральної робочої області та службової нижньої частини. Основний вміст розміщується в центральній області у вигляді карток, таблиць, форм, фільтрів, повідомлень і графічних блоків. Загальний вигляд головного меню наведено на рисунку 2.4.

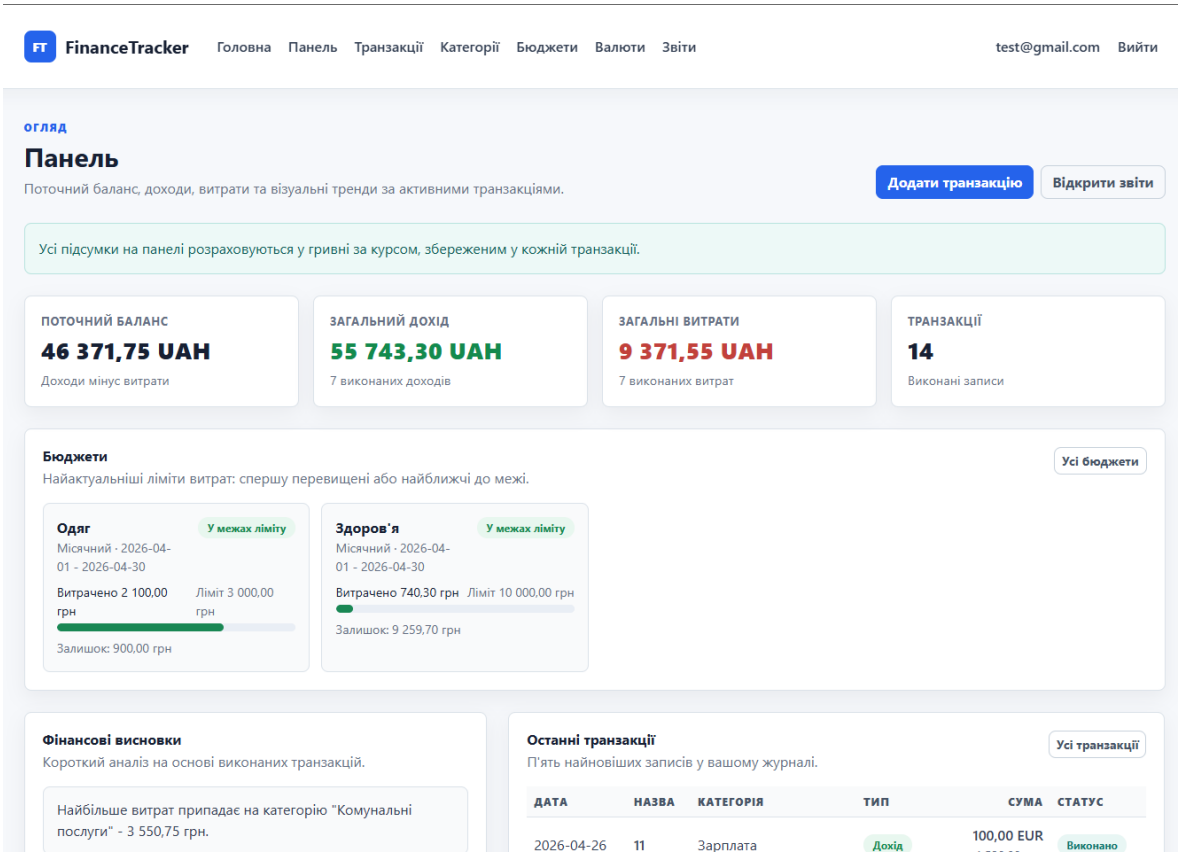


Рисунок *2.4 – Загальний вигляд головного меню веб-додатку*

Верхня навігаційна область містить назву системи, посилання на основні розділи та елементи облікового запису. Для робочої частини системи передбачено переходи до панелі, транзакцій, категорій, бюджетів, валют і звітів. Активний розділ виділяється кольором або станом пункту меню. Це відокремлює поточну сторінку від інших доступних переходів.

Візуальна концепція інтерфейсу відповідає стилю фінансової інформаційної панелі. Для неї використано світлий фон, чітке розділення блоків, картки з підсумковими показниками, таблиці для списків записів і кольорове виділення фінансових станів. Декоративні елементи не є основою оформлення. Акцент зроблено на читабельності числових даних, простому розташуванні елементів і швидкому розпізнаванні доходів, витрат, попереджень та помилок.

Центральна робоча область будується за єдиною схемою. У верхній частині області розміщується заголовок сторінки. Поруч або нижче подається

основна дія, наприклад створення нового запису. Далі розташовується основний вміст сторінки: таблиця, форма, набір карток або аналітичний блок. Для списків фільтри розміщуються перед таблицею. Для форм основний простір займають поля введення та кнопки підтвердження.

Колірна схема інтерфейсу побудована на світлій основі з окремими функціональними кольорами для фінансових станів. Нейтральні кольори використовуються для фону, меж, тексту та службових підписів. Акцентні кольори використовуються для дій, статусів, доходів, витрат і попереджень. Прийняту палітру наведено в таблиці 2.7.

Зелений колір застосовується для доходів, позитивних підсумків і успішних повідомлень. Червоний колір застосовується для витрат, помилок, видалення та перевищення бюджетного ліміту. Жовтий колір використовується для проміжних станів і попереджень. Синій колір використовується для основних кнопок, посилань і активних елементів навігації.

Таблиця 2.7 – Колірна палітра користувацького інтерфейсу

Призначення	**Колір**	**Код**
Основний фон сторінки	світло-сірий	#F8F9FA
Фон карток, таблиць і форм	білий	#FFFFFF
Основний текст	темно-сірий	#212529
Допоміжний текст	сірий	#6C757D
Межі блоків і таблиць	світло-сірий	#DEE2E6
Основна дія	синій	#0D6EFD
Активний або інформаційний стан	блакитний	#0DCAF0
Доходи та успішні дії	зелений	#198754
Витрати, помилки, небезпечні дії	червоний	#DC3545
Попередження	жовтий	#FFC107
Неактивні елементи	сірий	#ADB5BD

Шрифтова система орієнтована на читабельність таблиць, форм і числових показників. Використовується беззарубковий шрифт системного типу. Ієрархія тексту побудована через розмір, насиченість і колір. Основні параметри шрифтової системи наведено в таблиці 2.8.

Таблиця 2.8 – Шрифтова ієрархія інтерфейсу

Елемент тексту	**Розмір**	**Накреслення**	**Призначення**
Заголовок сторінки	24–32 px	напівжирне	назва поточного розділу
Назва блоку або картки	16–20 px	напівжирне	підпис логічного блоку
Основний текст	14–16 px	звичайне	дані у формах, таблицях і повідомленнях
Фінансова сума	20–28 px	напівжирне або жирне	баланс, дохід, витрати, ліміти
Допоміжний підпис	12–14 px	звичайне	пояснення, підказки, другорядні значення
Текст кнопки	14–16 px	напівжирне	назва дії
Текст бейджа	12–13 px	напівжирне	тип, статус або стан запису

Фінансові значення виділяються сильніше, ніж звичайний текст. Для балансу, доходів, витрат і бюджетних показників використовується більший розмір шрифту. Допоміжні підписи мають менший розмір і приглушений колір, щоб не конкурувати з основними значеннями.

Елементи керування мають єдину форму і відступи. Кнопки використовуються для створення, збереження, редагування, скасування, видалення або деактивації записів. Основна дія оформлюється акцентним кольором. Небезпечна дія оформлюється червоним кольором. Другорядні дії мають нейтральне оформлення.

Композиційні параметри інтерфейсу наведено в таблиці 2.9.

Таблиця 2.9 – *Основні параметри компонування інтерфейсу*

Параметр	**Значення**
Максимальна ширина робочого контейнера	1140–1320 px
Внутрішній відступ картки	16–24 px
Відступ між великими блоками	24–32 px
Відступ між полями форми	12–16 px
Висота основної кнопки	38–44 px
Радіус заокруглення карток і форм	8–12 px
Радіус заокруглення кнопок	6–8 px
Товщина меж таблиць і форм	1 px
Розміщення кнопок у формі	після основних полів
Розміщення фільтрів	перед таблицею
Розміщення дій над записом	останній стовпець таблиці

Картки призначені для коротких фінансових підсумків. У них подаються назва показника та основне числове значення. Таблиці використовуються для списків записів і містять заголовки стовпців, рядки даних та стовпець дій.

Фільтри розміщуються перед таблицею. Для відбору записів використовуються поле пошуку, випадаючі списки, поля дат і параметри сортування. Результати фільтрації відображаються у таблиці.

Графічні елементи використовуються для аналітичного подання даних. Кругові діаграми відображають структуру витрат за категоріями. Стовпчикові або лінійні графіки подають порівняння доходів і витрат за періодами.

Адаптивність інтерфейсу забезпечує зміну розміщення блоків залежно від ширини екрана. На широких екранах блоки можуть розміщуватися у кілька колонок. На вузьких екранах вони переходять у вертикальну послідовність. Таблиці переглядаються через горизонтальне прокручування.

2.5 Висновки до розділу

У другому розділі виконано проєктування веб-додатку для управління персональними фінансами. Визначено клієнт-серверну монолітну архітектуру з поділом на логічні рівні.

Описано засоби реалізації системи. Визначено стек для серверної частини, роботи з базою даних, автентифікації, формування інтерфейсу та графічного подання показників.

Спроєктовано інформаційну модель і структуру бази даних. Визначено основні сутності, їхні атрибути та зв’язки для збереження персональних фінансових записів.

Розроблено структуру користувацького інтерфейсу. Визначено навігацію, робочу область сторінок, палітру, шрифтову ієрархію, компонування, таблиці, форми, фільтри, картки та графіки.

Результати проєктування використано як основу для реалізації автентифікації, транзакцій, категорій, бюджетів, валют, дашборду та звітів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Реалізація автентифікації та авторизації

У веб-додатку реалізовано автентифікацію користувачів на основі стандартного механізму ASP.NET Core Identity. Реєстрація, вхід до системи, створення автентифікаційної cookie та збереження хешів паролів виконуються засобами Identity. Дані облікових записів зберігаються в тій самій базі даних, що й прикладні дані системи.

Модель користувача розширено класом ApplicationUser. Вона успадковує стандартні поля IdentityUser і містить додаткові поля, потрібні для роботи фінансового веб-додатку. До них належать ім’я для відображення, основна валюта користувача, дата створення запису, ознака активності та навігаційні колекції до категорій, транзакцій, бюджетів і валют користувача.

Лістинг 3.1 – *Фрагмент моделі користувача ApplicationUser*

```
public class ApplicationUser : IdentityUser
{
    [StringLength(100)]
    public string? DisplayName { get; set; }
    [Required]
    [StringLength(3, MinimumLength = 3)]
    public string PreferredCurrency { get; set; } = "UAH";
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
    public bool IsActive { get; set; } = true;
    public ICollection<Category> Categories { get; set; } = new
List<Category>();
    public ICollection<Transaction> Transactions { get; set; } =
new List<Transaction>();
    public ICollection<Budget> Budgets { get; set; } = new
List<Budget>();
    public ICollection<UserCurrency> UserCurrencies { get; set;
} = new List<UserCurrency>();
}
```

Поле DisplayName використовується для відображення користувача в інтерфейсі. Поле PreferredCurrency задає базову валюту облікового запису. Навігаційні колекції фіксують зв’язок користувача з його фінансовими даними.

Підключення Identity виконується у файлі Program.cs. Для роботи з користувачами використовується клас ApplicationUser, а сховище користувачів пов'язується з основним контекстом бази даних. Підтвердження облікового запису через електронну пошту не є обов'язковим.

Лістинг 3.2 – *Фрагмент налаштування Identity у Program.cs*

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));
builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
    options.SignIn.RequireConfirmedAccount = false)
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

Для коректної роботи автентифікації та авторизації в конвеєрі обробки запитів підключаються відповідні проміжні компоненти. Після цього реєструються маршрути контролерів і сторінки Identity.

Продовження лістингу 3.2

```
app.UseAuthentication();
app.UseAuthorization();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.MapRazorPages();
```

Закриті розділи системи захищаються атрибутом [Authorize]. Він застосовується до контролерів, які працюють з персональними фінансовими даними. До таких контролерів належать контролери транзакцій, категорій, бюджетів, валют, звітів, панелі та профілю. Якщо користувач не пройшов автентифікацію, доступ до цих сторінок не надається.

Лістинг 3.3 – *Приклад захисту контролера атрибутом [Authorize]*

```
[Authorize]
public class TransactionsController : Controller
{
    private readonly ITransactionService _transactionService;
    public TransactionsController(ITransactionService
transactionService)
    {
```

```
        _transactionService = transactionService;
    }
    private string GetCurrentUserId()
    {
        return User.FindFirstValue(ClaimTypes.NameIdentifier)
            ?? throw new
InvalidOperationException("Authenticated user id was not
found.");
    }
}
```

Атрибут [Authorize] обмежує доступ до контролера лише для авторизованих користувачів. Метод GetCurrentUserId() отримує ідентифікатор поточного користувача з claims. Далі цей ідентифікатор передається у сервіси, які працюють з фінансовими записами.

Окремо реалізовано ізоляцію даних за UserId. Авторизація на рівні контролера підтверджує факт входу користувача, але не визначає належність конкретного запису. Тому вибірка, перегляд, редагування і видалення прикладних даних виконуються тільки з урахуванням ідентифікатора поточного користувача.

Для отримання одного запису також використовується перевірка власника. Запис повертається лише тоді, коли його ідентифікатор відповідає запиту, а поле UserId збігається з поточним користувачем.

Лістинг 3.4 – *Фрагмент вибірки транзакцій поточного користувача*

```
var query = _context.Transactions
    .AsNoTracking()
    .Include(transaction => transaction.Category)
    .Where(transaction => transaction.UserId == userId);

return await _context.Transactions
    .AsNoTracking()
    .Include(transaction => transaction.Category)
    .FirstOrDefaultAsync(transaction =>
        transaction.Id == id && transaction.UserId == userId,
        cancellationToken);
```

Перевірка належності використовується і для пов’язаних даних. Наприклад, під час створення або редагування транзакції категорія повинна належати тому самому користувачу.

```
var category = await _context.Categories
    .FirstOrDefaultAsync(item =>
        item.Id == categoryId && item.UserId == userId,
        cancellationToken);
if (category == null)
{
    throw new InvalidOperationException("Обрана категорія
недоступна.");
}
```

Такий підхід відокремлює дані різних користувачів на рівні запитів до бази даних. Навіть у разі ручної зміни параметрів запиту користувач не отримує доступ до чужої транзакції, категорії, бюджету або валюти.

Після реєстрації або входу в систему перевіряється завершення профілю користувача. Якщо поле DisplayName не заповнене, користувач перенаправляється на сторінку завершення профілю. Базова валюта облікового запису встановлюється як UAH.

Лістинг 3.5 – *Фрагмент перевірки завершення профілю*

```
if (string.IsNullOrWhiteSpace(user.DisplayName))
{
    var returnUrl = Uri.EscapeDataString(
        context.Request.Path + context.Request.QueryString);
context.Response.Redirect($"/Profile/Complete?returnUrl={returnU
rl}");
    return;
}
```

Збереження профільних даних виконується через UserManager<ApplicationUser>. Це дає змогу оновлювати прикладні поля користувача через стандартний механізм Identity.

Сторінку входу користувача до системи наведено на рисунку 3.1. Вона використовується для автентифікації зареєстрованого користувача перед переходом до закритих розділів веб-додатку.

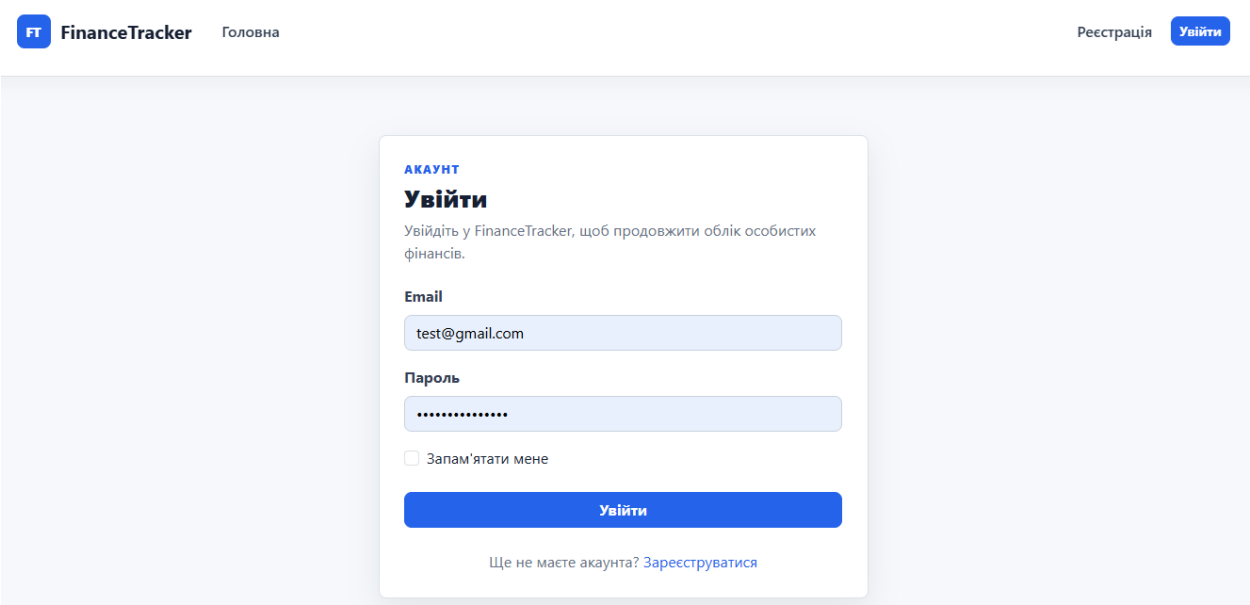


Рисунок *3.1 – Сторінка входу користувача до системи*

Сторінку завершення профілю після автентифікації наведено на рисунку 3.2. Вона використовується для введення імені, яке надалі відображається в інтерфейсі системи.

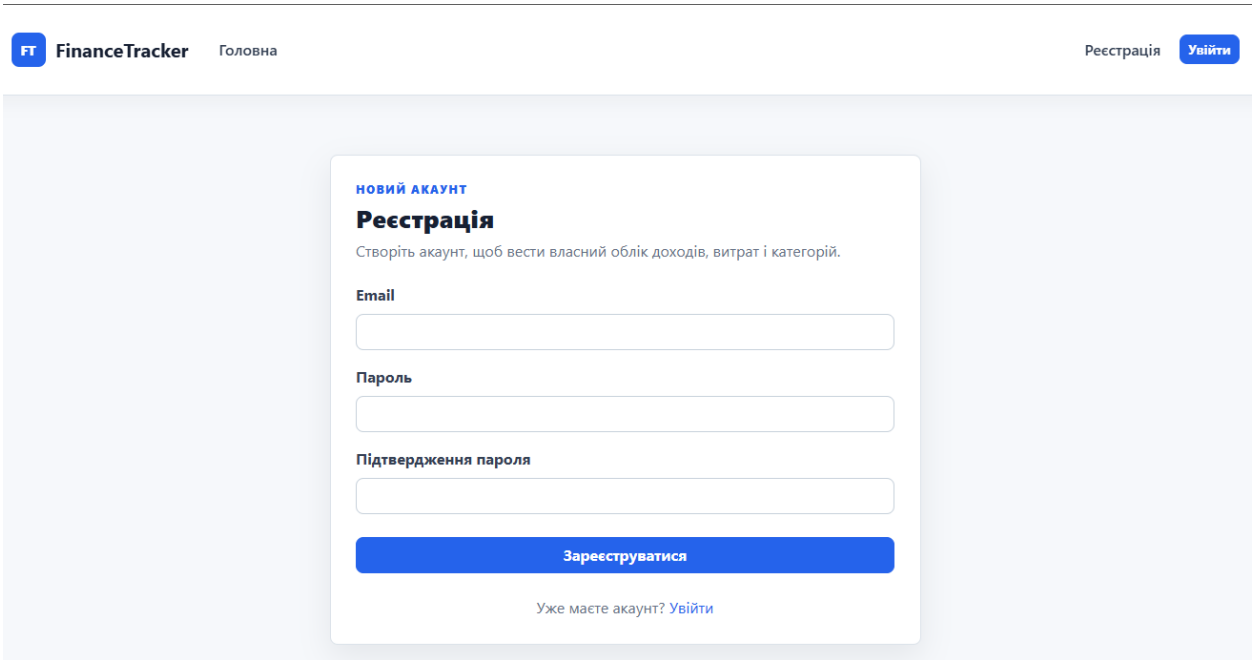


Рисунок *3.2 – Сторінка завершення профілю користувача*

У результаті реалізації модуль автентифікації та авторизації виконує дві основні функції. Перша пов’язана з входом користувача до системи та захистом закритих сторінок. Друга пов’язана з перевіркою належності прикладних даних конкретному користувачу через UserId.

3.2 Реалізація модулів категорій і транзакцій

Модулі категорій і транзакцій формують основну прикладну частину веб-додатку. Категорії використовуються для групування доходів і витрат. Транзакції зберігають окремі фінансові операції користувача та є джерелом даних для балансу, дашборду, звітів і бюджетного контролю.

Реалізація цих модулів побудована за загальною структурою застосунку. Контролери приймають HTTP-запити, отримують ідентифікатор поточного користувача та передають дані до сервісного рівня. Сервіси виконують перевірки, формують запити до бази даних, обробляють фільтри, виконують м’яке видалення та готують дані для відображення. Для сторінок списків і форм використовуються окремі ViewModel, що відокремлюють дані інтерфейсу від сутностей бази даних.

Модуль категорій реалізує перегляд, створення, редагування та м’яке видалення категорій. Категорія має належність до конкретного користувача через UserId, тому всі запити до категорій виконуються тільки в межах поточного облікового запису. Список категорій сортується за типом і назвою, що спрощує поділ записів на доходи та витрати.

Лістинг 3.6 – *Фрагмент вибірки категорій поточного користувача*

```
var query = _context.Categories
    .AsNoTracking()
    .Where(category => category.UserId == userId)
    .OrderBy(category => category.Type)
    .ThenBy(category => category.Name);
```

У наведеному фрагменті категорії відбираються за ідентифікатором користувача. Метод AsNoTracking() застосовується для запитів перегляду, де не потрібне відстеження змін об’єктів.

Під час створення категорії виконується перевірка дубліката. У межах одного користувача не допускається створення активної категорії з однаковою назвою і типом. Це обмеження запобігає дублюванню однакових класифікаторів у формах транзакцій і бюджетів.

Лістинг 3.7 – *Перевірка дубліката категорії*

```
var exists = await _context.Categories.AnyAsync(category =>
    category.UserId == userId &&
    category.Type == model.Type &&
    category.Name == model.Name &&
    !category.IsDeleted,
    cancellationToken);
if (exists)
{
    throw new InvalidOperationException("Категорія з такою назвою
вже існує.");
}
```

М’яке видалення категорії реалізовано через ознаку IsDeleted. Запис не вилучається фізично з бази даних, а приховується з активних списків. Такий спосіб не порушує зв’язки з раніше створеними транзакціями.

Сторінка категорій містить список записів користувача, тип категорії, візуальне позначення та дії редагування або видалення. Її вигляд подано на рисунку 3.3.

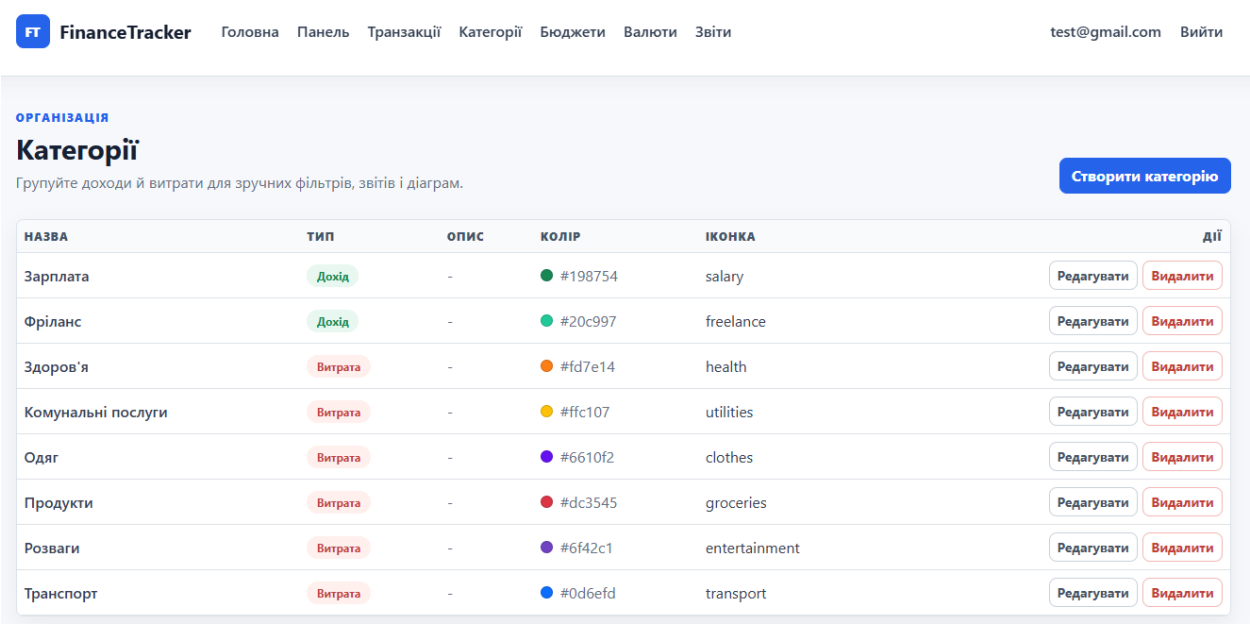


Рисунок *3.3 – Сторінка категорій*

Створення категорії виконується через окрему форму. У ній вводяться назва, тип, опис і візуальні параметри. Сторінку створення категорії наведено на рисунку 3.4.

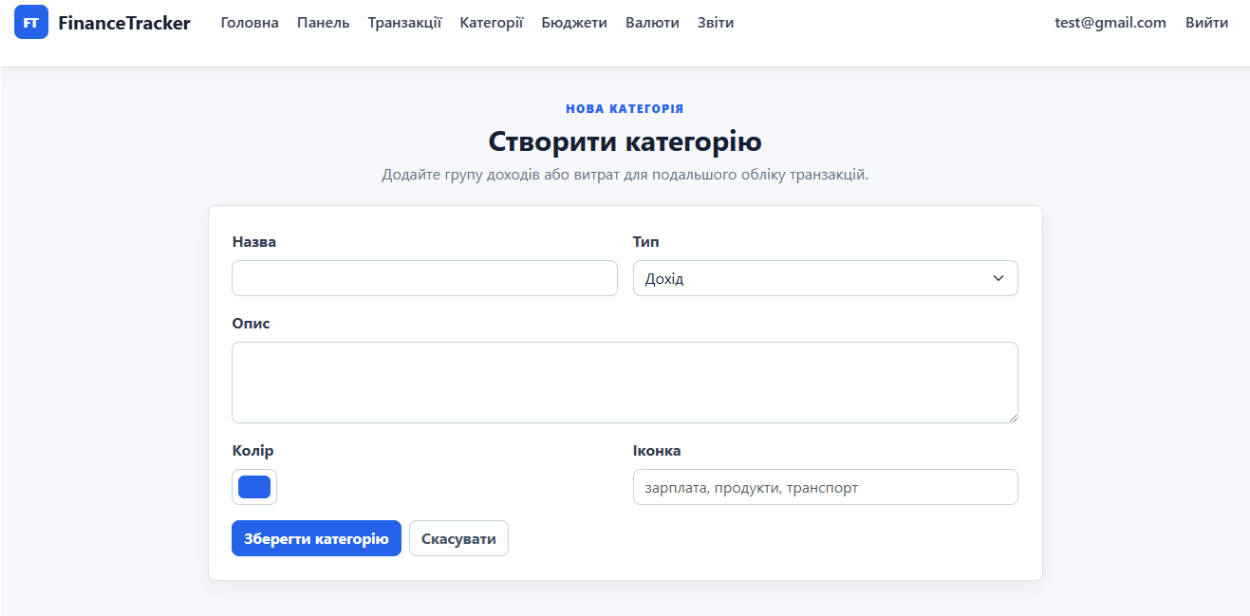


Рисунок *3.4 – Сторінка створення категорії*

Модуль транзакцій реалізує ведення фінансових операцій користувача. Для транзакцій передбачено створення, перегляд списку, редагування, м’яке

видалення, пошук, фільтрацію, сортування, пагінацію та роботу з валютними параметрами. Кожна транзакція належить користувачу і пов’язується з однією категорією.

Контролер транзакцій не містить основної прикладної логіки. Він приймає параметри запиту, отримує UserId, викликає метод сервісу та повертає підготовлену модель у представлення.

Лістинг 3.8 – *Фрагмент контролера транзакцій*

```
[Authorize]
public class TransactionsController : Controller
{
    private readonly ITransactionService _transactionService;
    public TransactionsController(ITransactionService
transactionService)
    {
        _transactionService = transactionService;
    }
    public async Task<IActionResult> Index(
        TransactionListFilterViewModel filter,
        CancellationToken cancellationToken)
    {
        var model = await
_transactionService.GetUserTransactionsAsync(
            GetCurrentUserId(),
            filter,
            cancellationToken);
        return View(model);
    }
}
```

Список транзакцій формується у сервісному рівні. Базовий запит відбирає тільки записи поточного користувача, підключає пов’язану категорію та виключає видалені записи з активного перегляду.

Лістинг 3.9 – *Базова вибірка транзакцій користувача*

```
var query = _context.Transactions
    .AsNoTracking()
    .Include(transaction => transaction.Category)
    .Where(transaction =>
        transaction.UserId == userId &&
        !transaction.IsDeleted);
```

Пошук транзакцій виконується за текстовими полями. У запиті перевіряється назва операції та коментар. Якщо пошуковий рядок порожній, умова пошуку не застосовується.

Лістинг 3.10 – *Фрагмент пошуку транзакцій*

```
if (!string.IsNullOrWhiteSpace(filter.SearchTerm))
{
    query = query.Where(transaction =>
        transaction.Title.Contains(filter.SearchTerm) ||
        (transaction.Comment != null &&
         transaction.Comment.Contains(filter.SearchTerm)));
}
```

Фільтрація списку транзакцій виконується за параметрами, які передаються з форми перегляду. До них належать тип операції, статус, категорія, валюта та межі періоду. Кожна умова додається до запиту тільки за наявності відповідного параметра.

Лістинг 3.11 – *Фрагмент фільтрації транзакцій*

```
if (filter.Type.HasValue)
{
    query = query.Where(transaction => transaction.Type ==
filter.Type.Value);
}
if (filter.CategoryId.HasValue)
{
    query = query.Where(transaction => transaction.CategoryId ==
filter.CategoryId.Value);
}
if (!string.IsNullOrWhiteSpace(filter.Currency))
{
    query = query.Where(transaction => transaction.Currency ==
filter.Currency);
}
if (filter.DateFrom.HasValue)
{
    query = query.Where(transaction => transaction.Date >=
filter.DateFrom.Value);
}
if (filter.DateTo.HasValue)
{
    query = query.Where(transaction => transaction.Date <=
filter.DateTo.Value);
}
```

Сортування транзакцій реалізовано через вибір поля та напряму сортування. За замовчуванням записи впорядковуються за датою у спадному порядку, а для стабільності порядку додатково використовується ідентифікатор запису.

Лістинг 3.12 – *Фрагмент сортування транзакцій*

```
query = filter.SortBy switch
{
    "title" => filter.SortDirection == "asc"
        ? query.OrderBy(transaction => transaction.Title)
        : query.OrderByDescending(transaction =>
transaction.Title),
    "date" => filter.SortDirection == "asc"
        ? query.OrderBy(transaction => transaction.Date)
               .ThenBy(transaction => transaction.Id)
        : query.OrderByDescending(transaction =>
transaction.Date)
               .ThenByDescending(transaction => transaction.Id),
    "amount" => filter.SortDirection == "asc"
        ? query.OrderBy(transaction => transaction.AmountInUah)
        : query.OrderByDescending(transaction =>
transaction.AmountInUah),

    _ => query.OrderByDescending(transaction =>
transaction.Date)
              .ThenByDescending(transaction => transaction.Id)
};
```

Під час створення або редагування транзакції перевіряється доступність категорії. Категорія повинна належати поточному користувачу. Також перевіряється відповідність типу транзакції типу категорії. Наприклад, витратна транзакція не може бути прив’язана до категорії доходу.

Лістинг 3.13 – *Перевірка належності та типу категорії*

```
var category = await _context.Categories
    .FirstOrDefaultAsync(item =>
        item.Id == categoryId &&
        item.UserId == userId &&
        !item.IsDeleted,
        cancellationToken);
if (category == null)
{
```

```
    throw new InvalidOperationException("Обрана категорія
недоступна.");
}
if ((int)category.Type != (int)transactionType)
{
    throw new InvalidOperationException(
        "Тип транзакції не відповідає типу категорії.");
}
```

Сторінка транзакцій містить фільтри, таблицю записів, валютні значення, статуси та дії над записами. Її вигляд подано на рисунку 3.5.

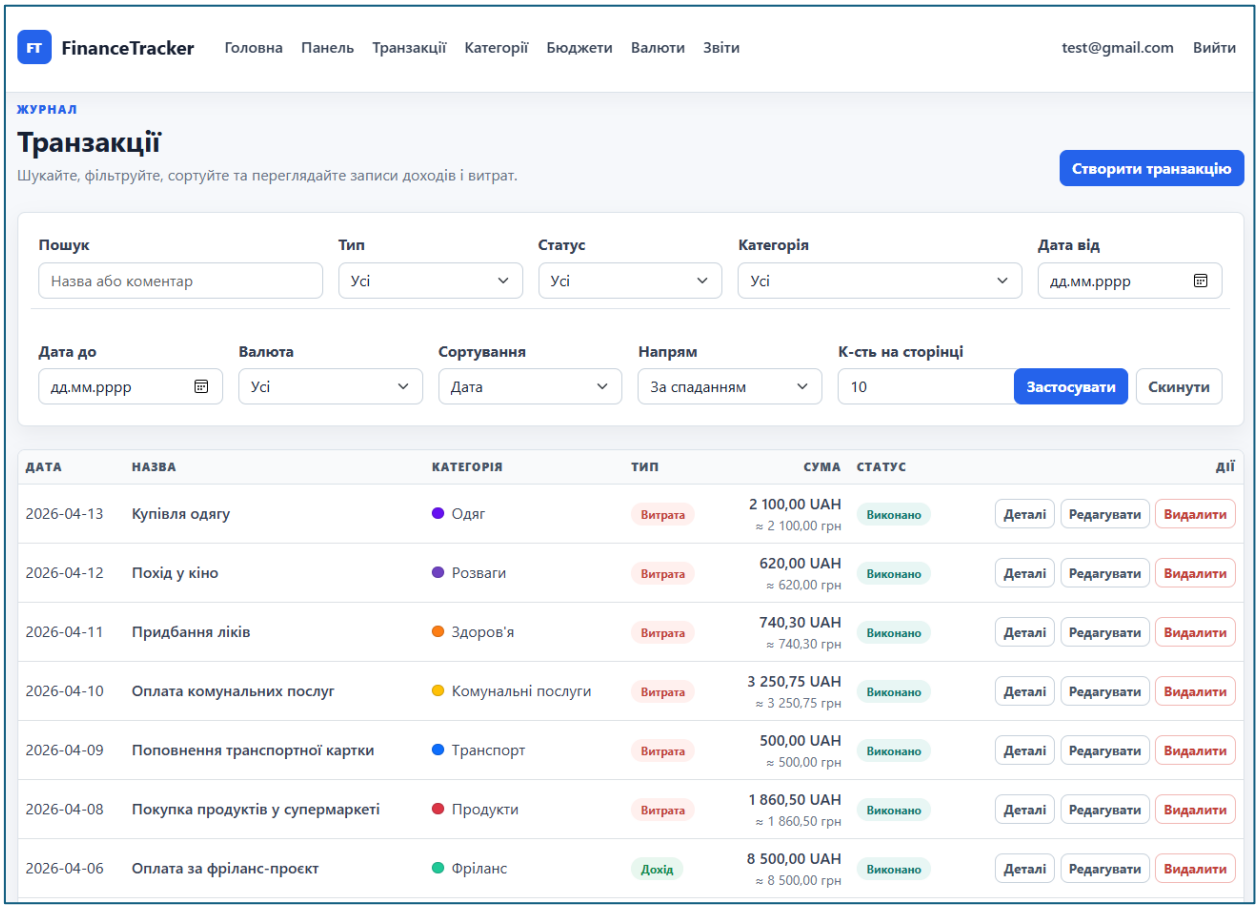


Рисунок *3.5 – Сторінка транзакцій*

Створення транзакції виконується через форму, у якій задаються назва, сума, валюта, тип операції, категорія, спосіб оплати, статус, дата та коментар. Сторінку створення транзакції наведено на рисунку 3.6.

У результаті реалізації модулі категорій і транзакцій забезпечують базову роботу з фінансовими записами. Категорії задають структуру класифікації доходів і витрат. Транзакції зберігають фінансові операції, підтримують

фільтрацію, сортування, пошук, м’яке видалення та мультивалютний облік. Дані обох модулів ізольовані за користувачем через UserId.

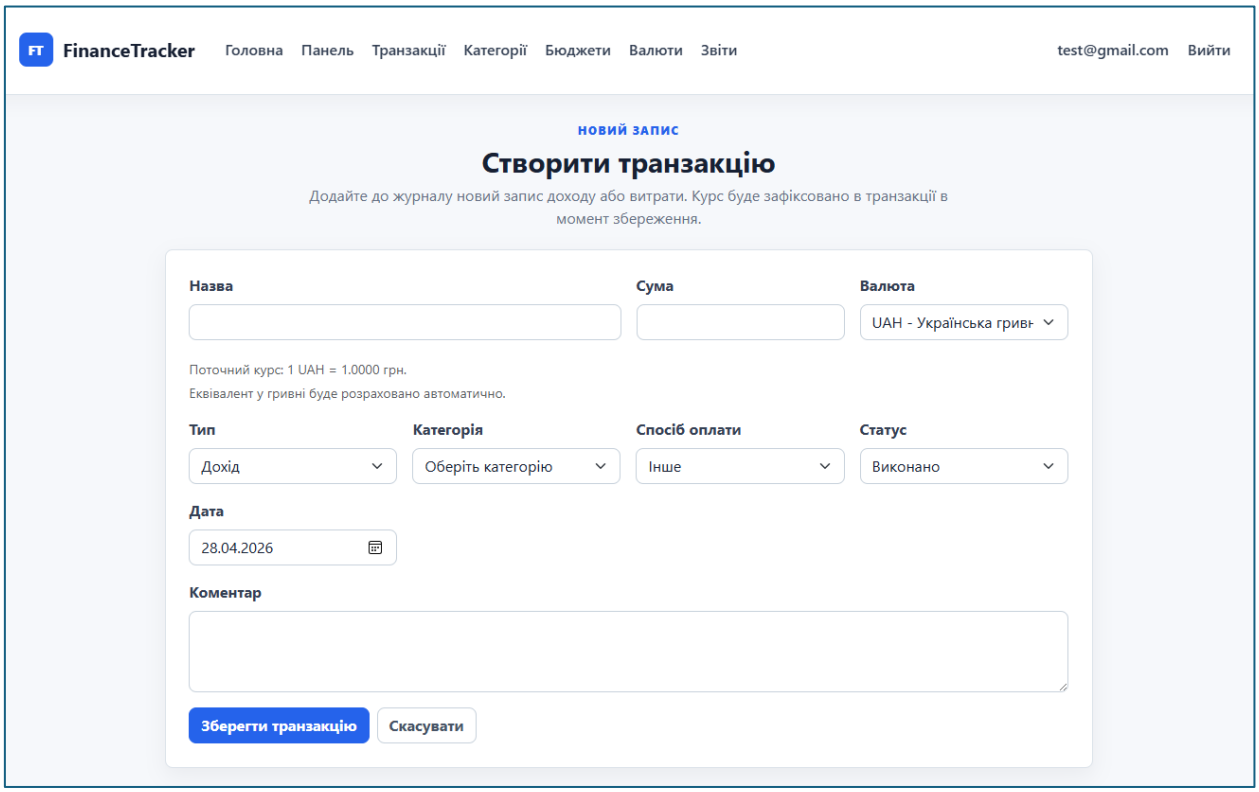


Рисунок *3.6 – Сторінка створення транзакцій*

3.3 Реалізація дашборду, звітів і візуалізації даних

Дашборд і звіти реалізують аналітичну частину вебдодатку. Ці модулі не створюють окремі фінансові записи, а працюють з уже збереженими транзакціями користувача. На їх основі формуються підсумкові значення, категоріальна аналітика, місячне порівняння доходів і витрат, останні операції та графічне подання даних.

Аналітична логіка винесена в сервісний рівень. Для дашборду використовується DashboardService, для звітів ReportService. Контролери отримують ідентифікатор поточного користувача, викликають відповідний

сервіс і передають у представлення готову модель даних. Обчислення не виконуються безпосередньо у представленнях.

Дашборд виконує роль головної робочої панелі користувача. На ньому подаються основні фінансові показники: баланс, загальні доходи, загальні витрати та кількість транзакцій. Дані дашборду формуються тільки для поточного користувача. У розрахунках використовуються завершені транзакції, які не були логічно видалені.

Лістинг 3.14 – *Фрагмент вибірки транзакцій для дашборду*

```
var transactions = await _context.Transactions
    .AsNoTracking()
    .Include(transaction => transaction.Category)
    .Where(transaction =>
        transaction.UserId == userId &&
        transaction.Status == TransactionStatus.Completed &&
        !transaction.IsDeleted)
    .OrderByDescending(transaction => transaction.Date)
    .ToListAsync(cancellationToken);
```

У наведеному фрагменті вибираються транзакції поточного користувача. До запиту підключається категорія, оскільки вона використовується для подальшого групування витрат. У дашборді не враховуються видалені та незавершені записи.

Основні фінансові показники розраховуються окремо для доходів і витрат. Для підсумків використовується поле AmountInUah, у якому зберігається гривневий еквівалент операції. Це потрібно для коректного обчислення балансу при використанні різних валют.

Лістинг 3.15 – *Обчислення основних показників дашборду*

```
var totalIncome = transactions
    .Where(transaction => transaction.Type == 
TransactionType.Income)
    .Sum(transaction => transaction.AmountInUah);
var totalExpenses = transactions
    .Where(transaction => transaction.Type == 
TransactionType.Expense)
    .Sum(transaction => transaction.AmountInUah);
var balance = totalIncome - totalExpenses;
```

```
var transactionCount = transactions.Count;
```

У цьому фрагменті доходи і витрати обчислюються окремими сумами. Баланс формується як різниця між доходами та витратами. Кількість транзакцій використовується як окремий показник активності користувача.

Першу частину сторінки дашборду з картками фінансових показників наведено на рисунку 3.7.

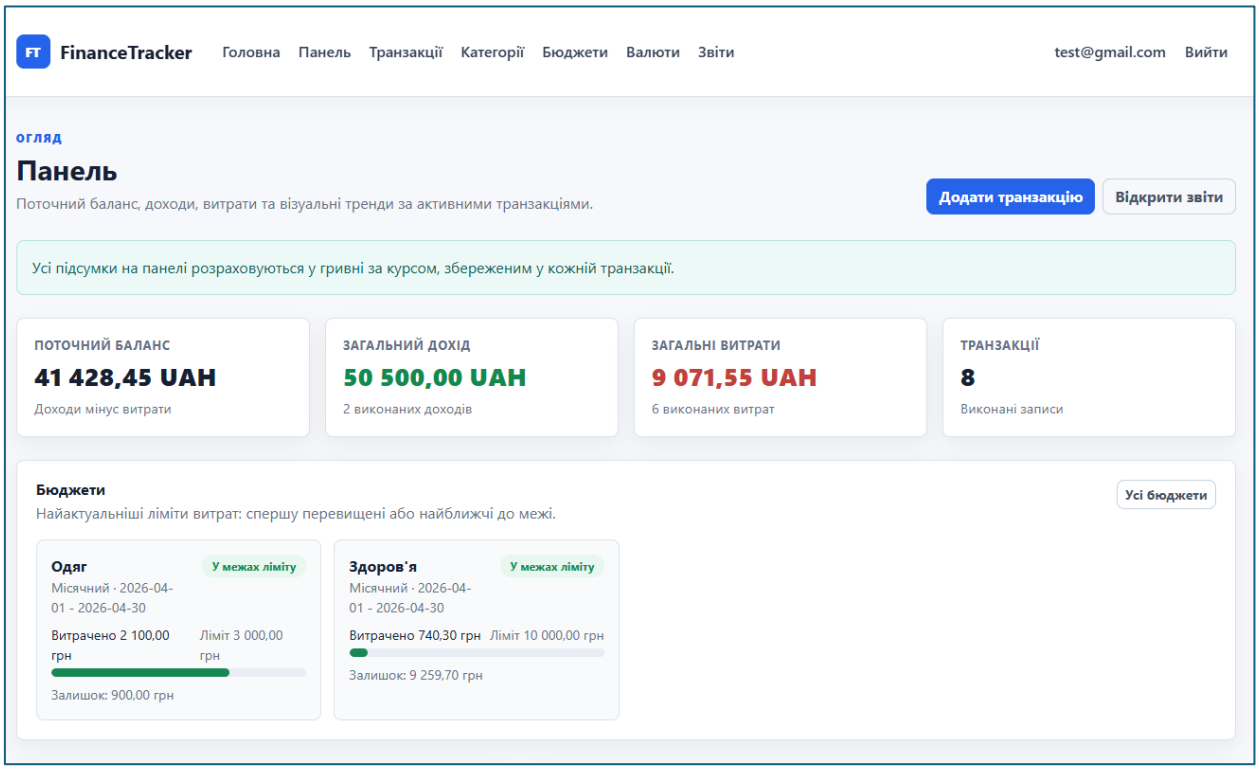


Рисунок *3.7 – Сторінка дашборду*

На дашборді також відображаються останні фінансові операції. Для цього з повного набору транзакцій вибирається обмежена кількість останніх записів. У модель передаються дата, назва, категорія, тип, сума, валюта та статус операції.

Лістинг 3.16 – *Формування блоку останніх транзакцій*

```
var recentTransactions = transactions
    .OrderByDescending(transaction => transaction.Date)
```

```
    .ThenByDescending(transaction => transaction.Id)
    .Take(5)
    .Select(transaction => new
DashboardRecentTransactionViewModel
    {
        Date = transaction.Date,
        Title = transaction.Title,
        CategoryName = transaction.Category.Name,
        Type = transaction.Type,
        Amount = transaction.Amount,
        Currency = transaction.Currency,
        Status = transaction.Status
    })
    .ToList();
```

Блок останніх транзакцій використовується для короткого перегляду останніх записів без переходу до повного журналу. Друга частина сторінки дашборду з додатковими блоками аналітики подана на рисунку 3.8.

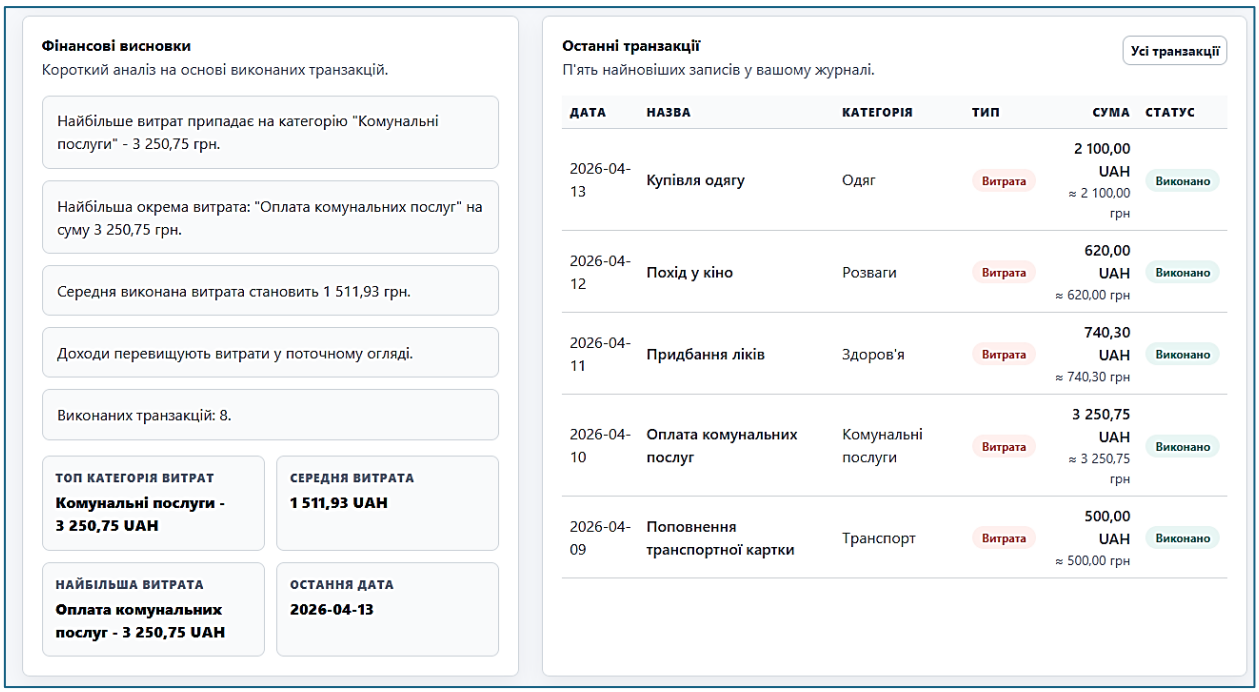


Рисунок *3.8 – Сторінка дашборду, блок останніх транзакцій*

Для швидкого аналізу витрат формується найбільша витратна категорія. Витратні транзакції групуються за назвою категорії, після чого для кожної

групи обчислюється загальна сума у гривнях. Найбільша група використовується в аналітичному блоці дашборду.

Лістинг 3.17 – *Визначення категорії з найбільшими витратами*

```
var topExpenseCategory = transactions
    .Where(transaction       =>       transaction.Type       ==
TransactionType.Expense)
    .GroupBy(transaction => transaction.Category.Name)
    .Select(group => new
    {
        CategoryName = group.Key,
        Total = group.Sum(transaction => transaction.AmountInUah)
    })
    .OrderByDescending(item => item.Total)
    .FirstOrDefault();
```

У дашборді використовуються не лише числові підсумки, а й графічні блоки. Кругова діаграма відображає структуру витрат за категоріями. Стовпчиковий графік використовується для порівняння доходів і витрат за місяцями. Дані для графіків готуються в сервісі, а представлення лише передає їх у клієнтську частину сторінки.

Місячне порівняння формується шляхом групування транзакцій за роком і місяцем. У кожній групі окремо підраховуються доходи та витрати.

Лістинг 3.18 – *Формування місячного порівняння доходів і витрат*

```
var monthlyComparison = transactions
    .GroupBy(transaction => new
    {
        transaction.Date.Year,
        transaction.Date.Month
    })
    .Select(group => new MonthlyComparisonViewModel
    {
        Label = $"{group.Key.Month:D2}.{group.Key.Year}",
        Income = group
            .Where(transaction => transaction.Type ==
TransactionType.Income)
            .Sum(transaction => transaction.AmountInUah),
        Expense = group
            .Where(transaction => transaction.Type ==
TransactionType.Expense)
            .Sum(transaction => transaction.AmountInUah)
```

```
    })
    .OrderBy(item => item.Label)
    .ToList();
```

Третю частину сторінки дашборду з графічними та додатковими аналітичними блоками наведено на рисунку 3.9.

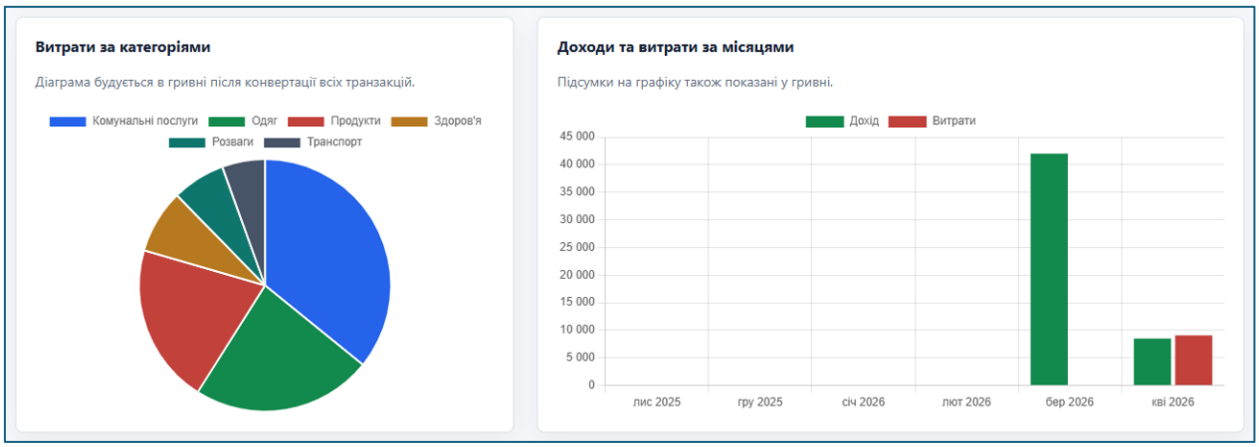


Рисунок 3.9 – *Сторінка дашборду, графіки*

Модуль звітів призначений для детального аналізу фінансових записів за вибраними параметрами. На відміну від дашборду, звіти формуються не тільки за всіма доступними завершеними транзакціями, а й за умовами періоду та валюти. Параметри фільтрації передаються до сервісу звітів і застосовуються під час формування запиту.

Лістинг 3.19 – *Формування вибірки даних для звіту*

```
var query = _context.Transactions
    .AsNoTracking()
    .Include(transaction => transaction.Category)
    .Where(transaction =>
        transaction.UserId == userId &&
        transaction.Status == TransactionStatus.Completed &&
        !transaction.IsDeleted);
if (fromDate.HasValue)
{
    query = query.Where(transaction => transaction.Date >=
fromDate.Value);
}
if (toDate.HasValue)
```

```
{
    query = query.Where(transaction => transaction.Date <=
toDate.Value);
}
if (!string.IsNullOrWhiteSpace(currency))
{
    query = query.Where(transaction => transaction.Currency ==
currency);
}
```

У цьому фрагменті базова вибірка обмежується поточним користувачем, завершеним статусом і відсутністю логічного видалення. Після цього за наявності параметрів додаються умови за датою початку, датою завершення та валютою.

Першу частину сторінки звітів з фільтрами та підсумковими показниками наведено на рисунку 3.10.

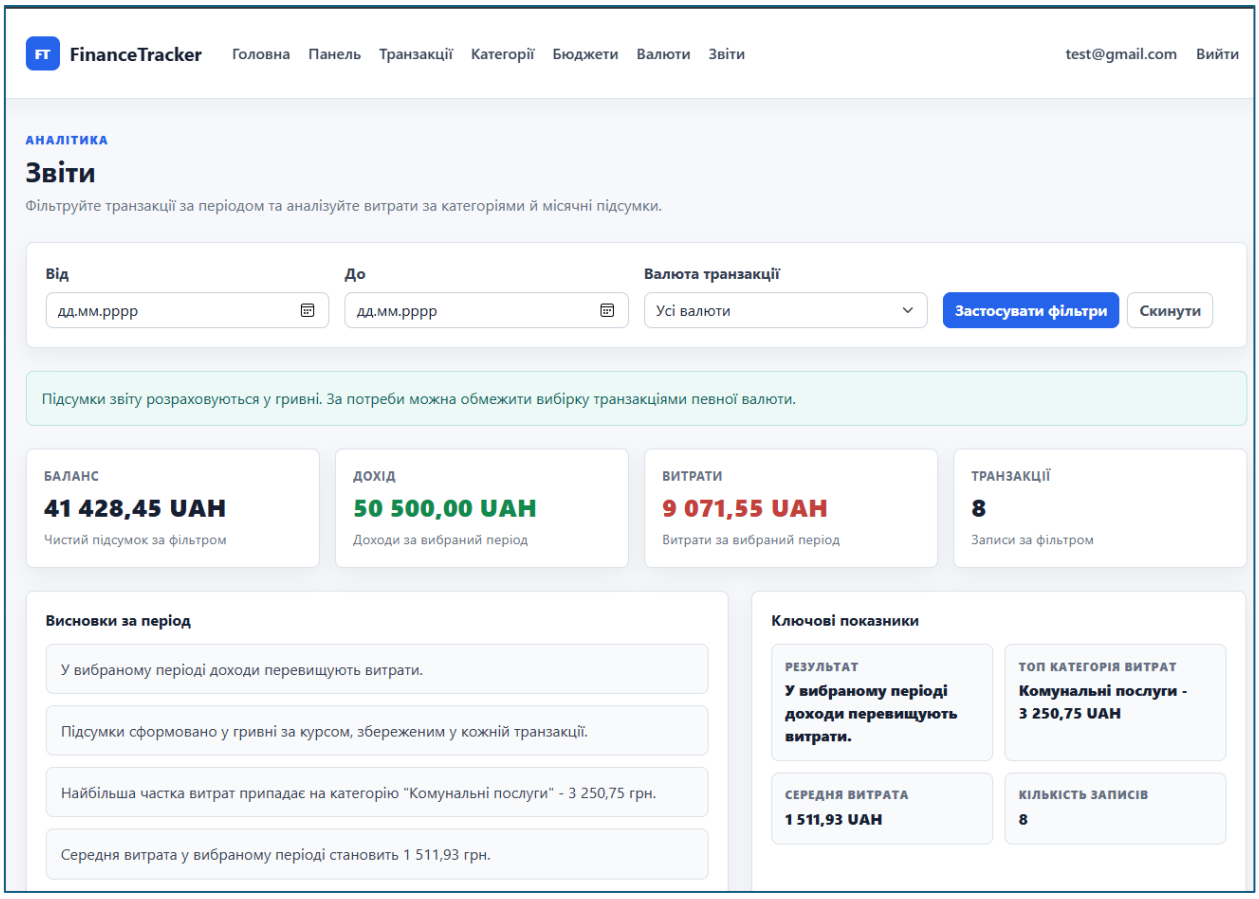


Рисунок *3.10 – Сторінка звітів*

Для категоріального аналізу вибираються лише витратні транзакції. Вони групуються за категоріями, після чого результат сортується за сумою витрат у спадному порядку. Такі дані використовуються для таблиці або діаграми структури витрат.

Лістинг 3.20 – *Агрегація витрат за категоріями*

```
var expenseByCategory = transactions
    .Where(transaction => transaction.Type == TransactionType.Expense)
    .GroupBy(transaction => transaction.Category.Name)
    .Select(group => new CategoryExpenseViewModel
    {
        CategoryName = group.Key,
        TotalAmount = group.Sum(transaction => transaction.AmountInUah)
    })
    .OrderByDescending(item => item.TotalAmount)
    .ToList();
```

На основі цієї агрегації формується розподіл витрат за категоріями. Найбільші категорії показують основні напрями витрат за вибраний період.

Друга частина сторінки звітів із графічним поданням даних наведена на рисунку 3.11.

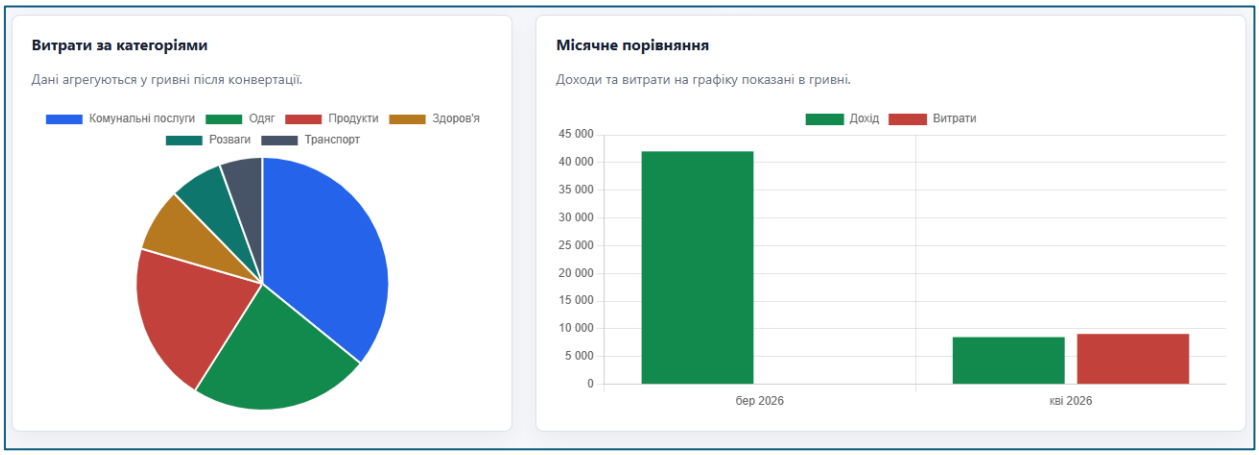


Рисунок *3.11 – Сторінка звітів*

Візуалізація даних у дашборді та звітах виконує допоміжну аналітичну роль. Числові показники подають точні значення, а графіки показують структуру та динаміку. Кругові діаграми використовуються для структури витрат за категоріями. Стовпчикові графіки застосовуються для порівняння доходів і витрат за місяцями.

Порожні стани також враховуються під час формування аналітики. Якщо користувач не має транзакцій або фільтри не повернули записів, підсумкові значення дорівнюють нулю, а графічні блоки не викликають помилки обробки

3.4 Реалізація механізму бюджетів та валют

Механізм бюджетів і валют доповнює модуль транзакцій. Бюджети використовуються для контролю витрат за категоріями. Валюти використовуються для збереження транзакцій у різних валютах і приведення фінансових підсумків до гривневого еквівалента. Реалізація цих модулів побудована через BudgetService, CurrencyService, відповідні контролери та ViewModel для сторінок списку і форм введення.

Бюджет створюється для конкретної витратної категорії користувача. Запис містить суму ліміту, тип періоду, дату початку, дату завершення та ознаку активності.

Сторінку бюджетів наведено на рисунку 3.12. На ній відображаються категорія, ліміт, фактичні витрати, залишок, відсоток використання та стан бюджету.

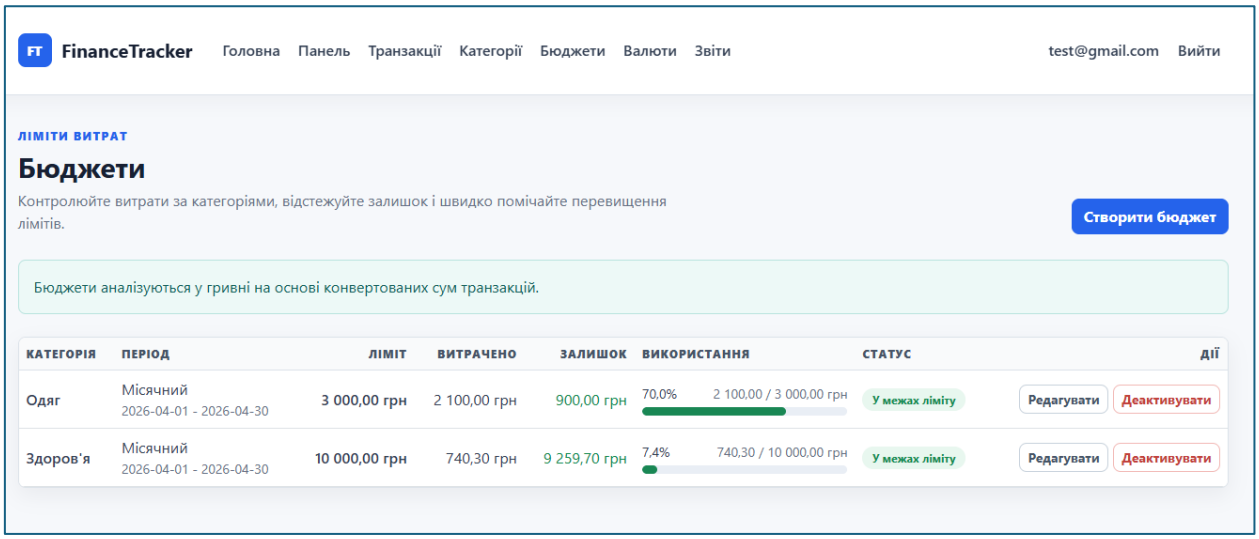


Рисунок *3.12 – Сторінка бюджетів*

Створення бюджету виконується через форму з вибором категорії, введенням ліміту та зазначенням періоду дії. Сторінку створення бюджету наведено на рисунку 3.13.

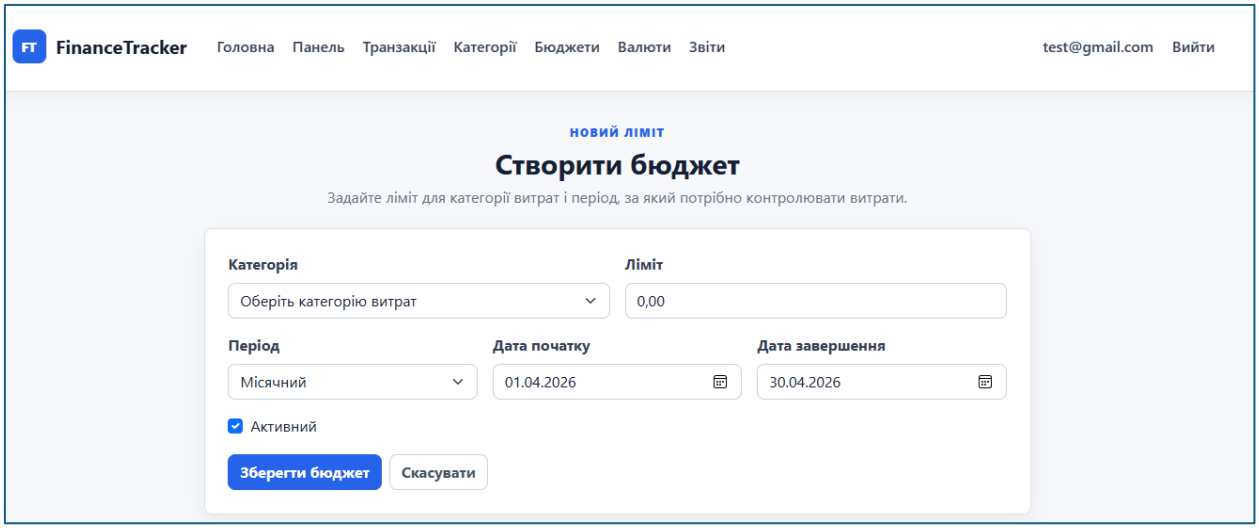


Рисунок *3.13 – Сторінка створення бюджету*

Фактичне використання бюджету розраховується на основі транзакцій. У розрахунок потрапляють лише витрати поточного користувача, які належать

до категорії бюджету, мають статус Completed і входять у межі періоду бюджету. Для суми використовується поле AmountInUah.

Лістинг 3.21 – *Розрахунок фактичних витрат у межах бюджету*

```
var spentAmount = await _context.Transactions
    .AsNoTracking()
    .Where(transaction =>
        transaction.UserId == userId &&
        transaction.CategoryId == budget.CategoryId &&
        transaction.Type == TransactionType.Expense &&
        transaction.Status == TransactionStatus.Completed &&
        transaction.Date >= budget.StartDate &&
        transaction.Date <= budget.EndDate)
    .SumAsync(transaction => transaction.AmountInUah,
cancellationToken);
```

Після визначення фактичних витрат обчислюється залишок, відсоток використання та ознака перевищення ліміту. Ці значення передаються у ViewModel і використовуються для числового подання та прогрес-бара.

Лістинг 3.22 – *Обчислення стану використання бюджету*

```
var usagePercent = budget.LimitAmount <= 0
    ? 0
    : Math.Round((spentAmount / budget.LimitAmount) * 100, 2);

var remainingAmount = budget.LimitAmount - spentAmount;
var isOverLimit = spentAmount > budget.LimitAmount;
```

Модуль валют зберігає перелік валют користувача. Для кожної валюти задається код, назва, курс до гривні, ознака базової валюти та ознака активності. Базовою валютою є UAH, для якої курс дорівнює 1.

Сторінку валют наведено на рисунку 3.14. На ній відображаються доступні користувачу валюти, їхні коди, назви, курси та стани активності.

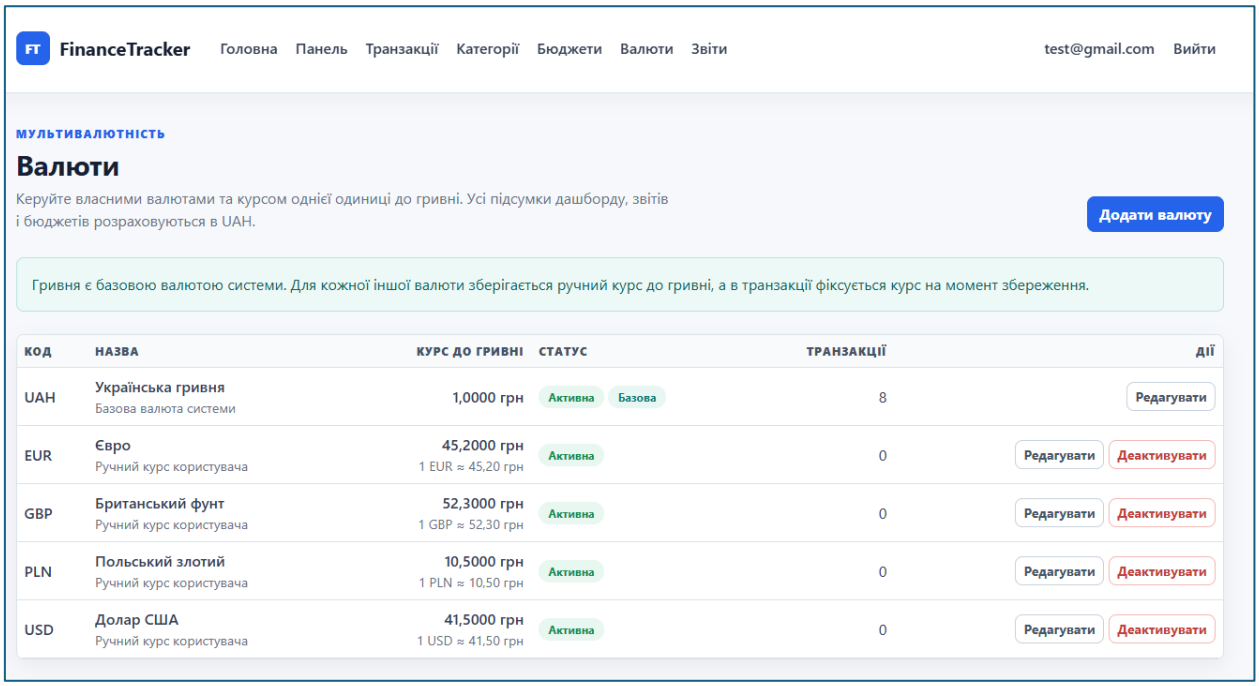


Рисунок *3.14 – Сторінка Валют*

Створення валюти виконується через форму введення коду, назви та курсу до гривні. Код валюти нормалізується до верхнього регістру. Сторінку створення валюти наведено на рисунку 3.15.

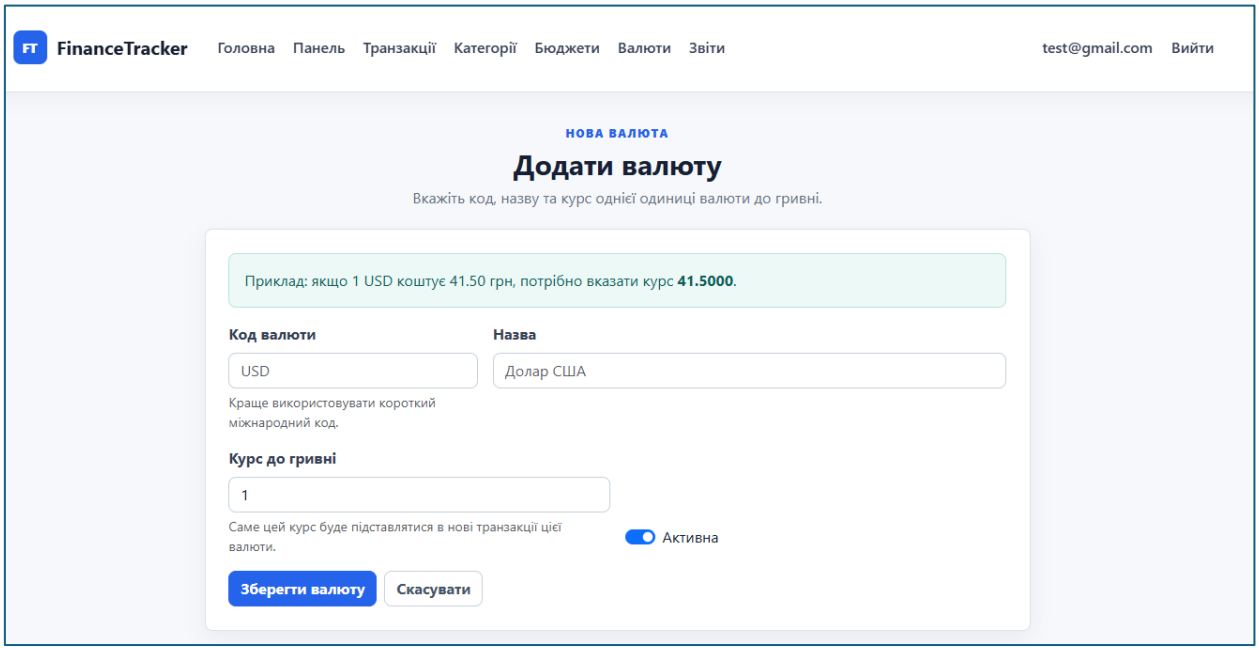


Рисунок *3.15 – Сторінка створення валюти*

Для нового користувача формується початковий набір валют.

Лістинг 3.23 – *Формування початкового набору валют користувача*

```
if (!await _context.UserCurrencies
    .AnyAsync(currency => currency.UserId == userId,
cancellationToken))
{
    _context.UserCurrencies.AddRange(
        new UserCurrency
        {
            UserId = userId,
            Code = "UAH",
            Name = "Українська гривня",
            RateToUah = 1m,
            IsBase = true
        },
        new UserCurrency
        {
            UserId = userId,
            Code = "USD",
            Name = "Долар США",
            RateToUah = 41.50m
        },
        new UserCurrency
        {
            UserId = userId,
            Code = "EUR",
            Name = "Євро",
            RateToUah = 45.20m
        }
    );
    await _context.SaveChangesAsync(cancellationToken);
}
```

Під час створення транзакції сервіс отримує курс вибраної валюти до гривні. У транзакції зберігається не лише початкова сума і код валюти, а й курс на момент створення запису та гривневий еквівалент.

Лістинг 3.24 – *Збереження курсу та гривневого еквівалента транзакції*

```
var rateToUah = await _currencyService.GetRateToUahAsync(
    userId,
    model.Currency,
    cancellationToken);
transaction.Amount = model.Amount;
transaction.Currency = model.Currency;
transaction.RateToUah = rateToUah;
```

```
transaction.AmountInUah = model.Amount * rateToUah;
```

Бюджети і валюти пов’язані з транзакціями через розрахункові дані. Бюджети використовують витратні транзакції у межах категорії та періоду. Валюти використовуються під час створення транзакції для фіксації курсу і розрахунку AmountInUah.

3.5 Тестування основних сценаріїв роботи

Для тестування створено окремий проєкт FinanceTracker.Tests. Тести запускаються через команду dotnet test. Як тестове сховище використано SQLite in-memory, що дає змогу перевіряти реальні запити Entity Framework Core без використання постійного файлу бази даних. Усього реалізовано 8 автоматизованих тестів, усі тести виконано успішно: Passed: 8, Failed: 0, Skipped: 0

Запуск усіх тестів виконується командою dotnet test FinanceTracker.Tests\FinanceTracker.Tests.csproj.

Також тести можуть запускатися через файл рішення dotnet test FinanceTracker.slnx.

Перелік перевірених сценаріїв наведено в таблиці 3.1.

Таблиця 3.1 – *Автоматизовані тести сервісної логіки*

№	Назва тесту	Об’єкт перевірки	Очікуваний результат
1	Створення початкового набору валют	CurrencyService	для нового користувача створюються базові валюти; гривня має курс 1 і позначається як базова
2	Захист базової валюти від некоректного редагування	CurrencyService	базова валюта UAH не змінює код, курс, активність і базовий статус після оновлення

3	Розрахунок гривневого еквівалента транзакції	TransactionService	під час створення транзакції зберігаються валюта, курс до гривні та значення AmountInUah
4	Перевірка відповідності транзакції та категорії	TransactionService	транзакція не створюється, якщо її тип не відповідає типу вибраної категорії
5	Ізоляція транзакцій за користувачем і пошук записів	TransactionService	користувач отримує лише власні транзакції; пошук повертає записи за назвою або коментарем
6	Заборона бюджету для категорії доходів	BudgetService	бюджет не створюється для категорії, що належить до доходів
7	Розрахунок стану використання бюджету	BudgetService	правильно обчислюються витрати, залишок, відсоток використання та ознака перевищення ліміту
8	Формування звіту за завершеними транзакціями	ReportService	звіт враховує лише завершені транзакції, застосовує валютний фільтр і правильно рахує підсумки

Результат виконання автоматизованих тестів подано на рисунку 3.16.

```
PS D:\Studwork\FinanceTracker> dotnet test FinanceTracker.slnx
Restore complete (1,7s)
  FinanceTracker net9.0 succeeded (3,8s) → FinanceTracker\bin\Debug\net9.0\FinanceTracker.dll
  FinanceTracker.Tests net9.0 succeeded (0,7s) → FinanceTracker.Tests\bin\Debug\net9.0\FinanceTracker.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.8.2+699d445a1a (64-bit .NET 9.0.13)
[xUnit.net 00:00:00.10]   Discovering: FinanceTracker.Tests
[xUnit.net 00:00:00.15]   Discovered:  FinanceTracker.Tests
[xUnit.net 00:00:00.15]   Starting:    FinanceTracker.Tests
[xUnit.net 00:00:02.59]   Finished:    FinanceTracker.Tests
  FinanceTracker.Tests test net9.0 succeeded (4,5s)

Test summary: total: 8; failed: 0; succeeded: 8; skipped: 0; duration: 4,5s
Build succeeded in 11,4s
PS D:\Studwork\FinanceTracker>
```

Рисунок 3.16 – *Результат виконання автоматизованих тестів*

Перевірка валют охоплює створення початкового набору валют і захист базової валюти від некоректного редагування. Для UAH перевіряється

фіксований курс 1, базовий статус і неможливість деактивації через звичайне оновлення.

Тести транзакцій перевіряють мультивалютний розрахунок, зв’язок транзакції з категорією та ізоляцію даних користувача. Окремо перевіряється правило відповідності типу транзакції типу категорії. Наприклад, дохід не може бути створений у витратній категорії.

Бюджетні тести перевіряють обмеження на створення бюджету для категорій доходів і правильність обчислення використання бюджету. У розрахунок потрапляють лише витратні транзакції поточного користувача зі статусом Completed, які входять у період бюджету.

Тест звітів перевіряє формування підсумків за завершеними транзакціями. Додатково перевіряється фільтр за валютою, правильність розрахунку доходів, витрат, балансу та визначення основної витратної категорії.

3.6 Аналіз результатів роботи системи

Автоматизована перевірка сервісної логіки завершилася без помилок. Запущено 8 тестів, з них 8 пройдено успішно, невдалих і пропущених тестів не зафіксовано. Перевірка охопила роботу валют, транзакцій, бюджетів і звітів, зокрема розрахунок гривневого еквівалента, ізоляцію даних за користувачем, перевірку типу категорії, бюджетні обчислення та формування звітних підсумків.

Додатково проведено ручну перевірку основних сценаріїв у браузері. Перевірка виконувалася для дій, які відповідають основному призначенню веб-додатку: вхід користувача, створення фінансових записів, перегляд списків, формування аналітичних показників, робота з бюджетами та валютами. Результати наведено в таблиці 3.2.

Перевірка форм введення показала коректну роботу валідації. Обов’язкові поля контролюються перед збереженням. Некоректні числові

значення не записуються до бази даних. Повідомлення про помилки відображаються біля відповідних полів або в межах форми.

Приклад спрацювання валідації форми створення транзакції наведено на рисунку 3.17.

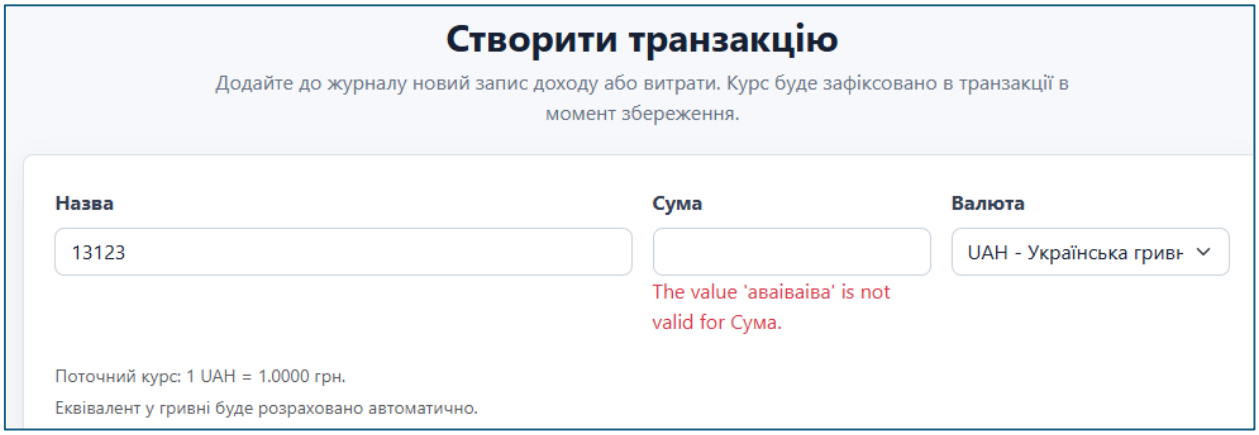


Рисунок *3.17 – Перевірка валідації форми створення транзакції*

Таблиця 3.2 – *Результати ручної перевірки основних сценаріїв*

№	Сценарій перевірки	Результат перевірки	Стан
1	Реєстрація користувача	Новий обліковий запис створюється, користувач переходить до роботи із системою	успішно
2	Вхід користувача	Після автентифікації відкривається доступ до захищених розділів системи	успішно
3	Створення категорії	Категорія зберігається з назвою, типом і візуальним позначенням	успішно
4	Створення транзакції	Фінансова операція зберігається із сумою, валютою, категорією, датою і статусом	успішно
5	Пошук транзакцій	Список містить записи, що відповідають назві або коментарю	успішно
6	Фільтрація транзакцій	Записи відбираються за типом, категорією, валютою і датою	успішно

7	Розрахунок інформаційної панелі	Баланс, доходи і витрати формуються на основі завершених транзакцій	успішно
8	Формування звіту	Звіт і графіки формуються за вибраним періодом і валютою	успішно
9	Створення бюджету	Бюджет відображає ліміт, витрати, залишок і відсоток використання	успішно
10	Додавання валюти	Валюта додається до списку, її курс використовується для розрахунку гривневого еквівалента	успішно
11	Розмежування доступу	Користувачу відображаються лише власні записи за ідентифікатором облікового запису	успішно
12	Логічне видалення запису	Запис позначається як видалений і не відображається в основному списку	успішно

Перевірка форм введення показала коректну роботу валідації. Обов’язкові поля контролюються перед збереженням. Некоректні числові значення не записуються до бази даних. Повідомлення про помилки відображаються біля відповідних полів або в межах форми.

Приклад спрацювання валідації форми створення транзакції наведено на рисунку 3.17.

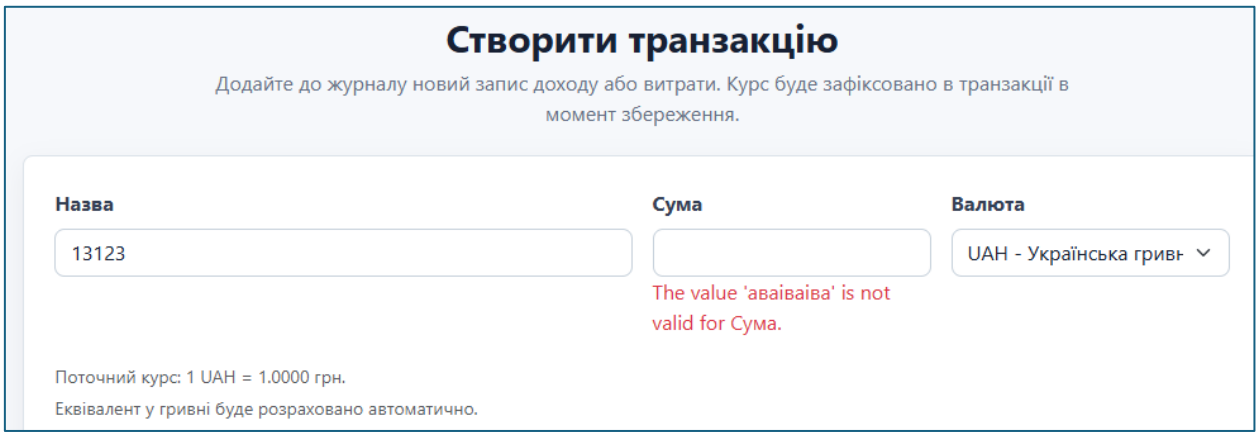


Рисунок *3.17 – Перевірка валідації форми створення транзакції*

Для форми бюджету перевірено введення ліміту та дат періоду. Некоректний ліміт, порожні обов’язкові поля або неправильний діапазон дат блокують збереження запису. Приклад перевірки подано на рисунку 3.18.

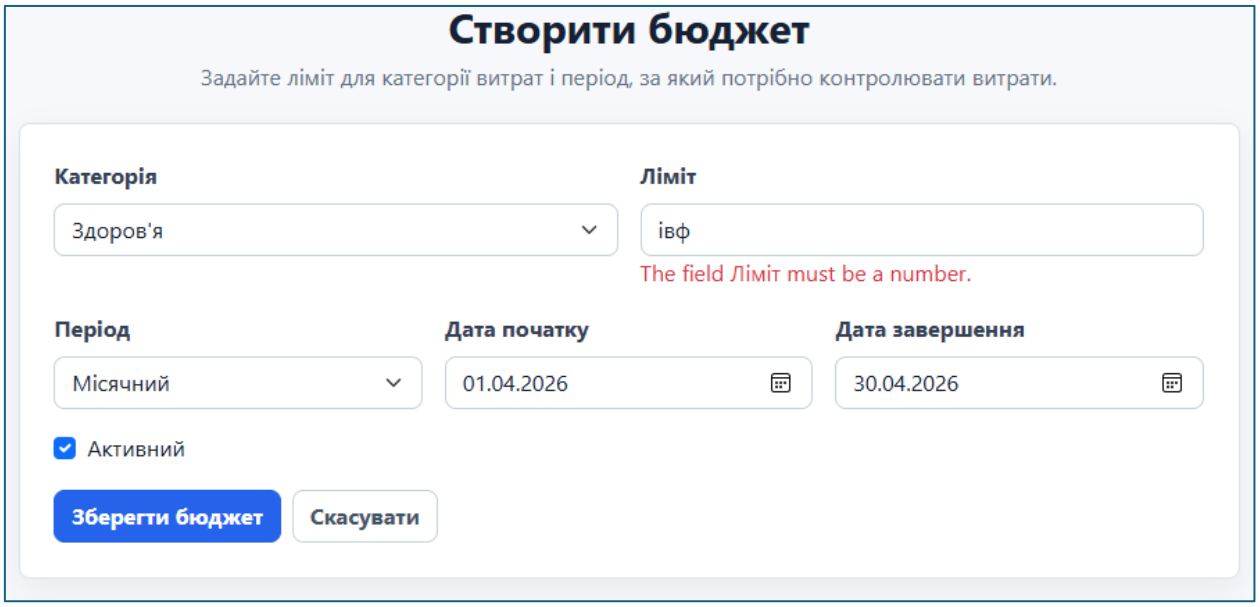


Рисунок *3.18 – Перевірка валідації форми створення бюджету*

Проведена перевірка підтвердила працездатність основних модулів вебдодатку.

3.7 Висновки до розділу

У третьому розділі виконано програмну реалізацію веб-додатку для управління персональними фінансами. Реалізовано основні функціональні модулі, передбачені під час проєктування системи.

Реалізовано автентифікацію та авторизацію користувачів. Забезпечено реєстрацію, вхід до системи, завершення сеансу, захист сторінок і прив’язку фінансових записів до облікового запису користувача.

Реалізовано модулі категорій і транзакцій. Категорії забезпечують групування доходів і витрат, а транзакції забезпечують збереження фінансових

операцій, пошук, фільтрацію, сортування, посторінковий перегляд, м’яке видалення та облік валютних значень.

Реалізовано дашборд, звіти та графічне подання даних. Система формує підсумки доходів, витрат і балансу, відображає структуру витрат за категоріями та показує динаміку фінансових показників за періодами.

Реалізовано механізм бюджетів і валют. Бюджети використовуються для контролю витрат за категоріями, а валюти забезпечують збереження операцій у різних валютах і розрахунок гривневого еквівалента.

Проведено автоматизоване та ручне тестування основних сценаріїв роботи. Перевірено створення і обробку фінансових записів, розрахунок підсумків, роботу бюджетів, валют, звітів, фільтрів, валідації форм і розмежування доступу до даних.

За результатами тестування підтверджено працездатність реалізованих модулів. Розроблений веб-додаток виконує поставлені функції та може використовуватися для ручного обліку персональних фінансів.

ВИСНОВКИ

1. У кваліфікаційній роботі розроблено вебд-одаток для управління персональними фінансами з використанням ASP.NET Core та SQLite.
2. Проаналізовано предметну область персонального фінансового обліку. Визначено основні об’єкти системи: транзакції, категорії, бюджети, валюти, звіти та користувацькі дані.
3. Розглянуто існуючі рішення для обліку персональних фінансів. Встановлено, що вони мають функції фінансового обліку, але не повністю відповідають задачі простого ручного веб-обліку.
4. Сформовано вимоги до системи. Визначено потребу в реєстрації користувачів, веденні доходів і витрат, роботі з категоріями, бюджетами, валютами, звітами та захисті персональних даних.
5. Спроєктовано архітектуру веб-додатку, інформаційну модель, структуру бази даних і користувацький інтерфейс.
6. Реалізовано основні модулі системи: автентифікацію, категорії, транзакції, дашборд, звіти, бюджети та валюти.
7. Виконано тестування основних сценаріїв роботи. Перевірено коректність введення, обробки, фільтрації, розрахунків і розмежування доступу до даних.
8. Розроблений веб-додаток виконує поставлені функції та може використовуватися для ручного обліку доходів, витрат і контролю персональних фінансових показників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Крисоватий А. І. Фінанси : підручник. Тернопіль : ЗУНУ, 2024. 632 с.

2. Національна стратегія розвитку фінансової грамотності до 2030 року / Національний банк України. Київ, 2024. 50 с.

3. Фінансова грамотність, фінансова інклюзія та фінансовий добробут в Україні у 2021 році : звіт за результатами дослідження / USAID, Info Sapiens. Київ, 2021. 83 с.

4. Пономарьов І. В. Технології розробки кросплатформених веб-додатків : конспект лекцій. Дніпро : ДНУ, 2023. 113 с.

5. Шкарупа А. О., Стрельцов О. А. Аналіз особливостей сучасних веб-додатків. Automation and Development of Electronic Devices. 2022. Part 2. С. 102–107.

6. Jagatha V., Khamesipour A., Chung S. Cross-Platform App Development: A Comparative Study of PWAs and React Native Mobile Apps. Proceedings of the ISCAP Conference. 2023. Vol. 9, № 5944

7. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI. / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 05.05.2026).

8. OWASP Foundation. Authentication Cheat Sheet. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (date of access: 05.05.2026).

9. OWASP Foundation. Authorization Cheat Sheet. OWASP Cheat Sheet Series. URL:https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html (date of access: 05.05.2026).

10. OWASP Foundation. A01:2021 Broken Access Control. OWASP Top 10:2021. URL: https://owasp.org/Top10/2021/A01_2021-Broken_Access_Control/ (date of access: 05.05.2026).

11. YNAB. Features. URL: https://www.ynab.com/features (date of access: 05.05.2026).

12. BudgetBakers. Wallet: Personal & Family Finance Manager. URL: https://web.budgetbakers.com/ (date of access: 05.05.2026).

13. Money Manager Ex. Official Website. URL: https://moneymanagerex.org/ (date of access: 05.05.2026).

14. Krishan D. Understanding the Difference Between MVC Architecture and Clean Architecture in ASP.NET. Medium. URL: https://medium.com/@darshankrishan12/understanding-the-difference-between-mvc-architecture-and-clean-architecture-in-asp-net-0acd67228d1f (date of access: 05.05.2026).

15. Microsoft. C# documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (date of access: 05.05.2026).

16. Microsoft. Overview of ASP.NET Core MVC. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview (date of access: 05.05.2026).

17. Microsoft. Introduction to Identity on ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity (date of access: 05.05.2026).

18. Microsoft. Overview of Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (date of access: 05.05.2026).

19. SQLite. About SQLite. URL: https://sqlite.org/about.html (date of access: 05.05.2026).

20. Microsoft. Razor syntax reference for ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor (date of access: 05.05.2026).

21. Bootstrap. Get started with Bootstrap. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (date of access: 05.05.2026).

22. Chart.js. Getting Started. URL: https://www.chartjs.org/docs/latest/getting-started/ (date of access: 05.05.2026).

23. Microsoft. NuGet documentation. URL: https://learn.microsoft.com/en-us/nuget/ (date of access: 05.05.2026).

24. Микуленко Д. Проєктування та реалізація вебзастосунку для управління персональними фінансами. *Збірник наукових праць студентів, здобувачів PhD, докторантів і молодих вчених, університетська конференція «МОЛОДА НАУКА-2026» Том III «Секції Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ*. Запоріжжя, 2026. С. 100–101.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Микуленко Данило Андрійович , студентка 4 курсу, форми навчання заочної, Інженерного навчально-наукового інституту ім. Ю. М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми «Програмне забезпечення систем», адреса електронної пошти se121-32@stu.zsea.edu.ua , — підтверджую, що написана мною кваліфікаційна робота на тему «Розробка додатку для ефективного контролю та візуалізації персональних фінансів відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

-згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 20.05.2026 _____Микуленко Данило Андрійович
(студентка)

Дата 20.05.2026 _____Міхайлуца Олена Миколаївна
(керівник)