МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБДОДАТКУ З ОБЛІКУ ОСОБИСТИХ ДОХОДІВ ТА ВИТРАТ»

Виконав: здобувач освіти 4 курсу, групи К121-22
Спеціальність 121 Інженерія
програмного забезпечення
(шифр і спеціальність)
Іван КОЛОМОЙЦЕВ
(ім'я та ПРІЗВИЩЕ)
Керівник Олена ШИТІКОВА
(ім'я та ПРІЗВИЩЕ)
викладач Запорізького авіаційного фахового коледжу ім. О.Г. Івченка
Рецензент Артур ТІМЕНКО
(посада, вчене звання, науковий ступінь, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2026

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо–кваліфікаційний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Т.М. Смолянкова
(підпис)

“ 05 ” червня 2026 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

КОЛОМОЙЦЕВУ Івану Олександровичу
(ПРІЗВИЩЕ, ім’я та по– батькові)

1. Тема роботи «Проєктування та розробка вебдодатку з обліку особистих доходів та витрат»

Керівник роботи к.т.н. ШИТІКОВА Олена Віткорівна
(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від «01» грудня 2025 р. № 1728-с

2. Строк подання студентом роботи 01.06.2026

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

4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити)
1. Сучасні вебдодатки
2. Розробка проєкту вебдодатку
3. Програмна реалізація вебдодатку

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Презентація до захисту

6. Консультанти розділів роботи

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

7. Дата видачі завдання 01.12.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	постановка задачі	Грудень 2025	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2026	виконано
3.	Розробка першого розділу:		
	огляд інформаційних систем та технологій для їхнього створення	Лютий 2026	виконано
4.	Розробка другого розділу:		
	розробка проєкту інформаційної системи та вибір технологій	Квітень 2026	виконано
5.	Розробка третього розділу:		
	розробка застосунку	Травень 2026	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2026	виконано
7.	Захист кваліфікаційної роботи	16.06.2026	виконано

Здобувач освіти _____ Іван КОЛОМОЙЦЕВ
(підпис) (ім’я та ПРІЗВИЩЕ)

Керівник роботи _____ Олена ШИТІКОВА
(підпис) (ім’я та ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я та ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 46 сторінок, 13 рисунків, 1 таблиця, 12 джерел.

Об’єкт дослідження – процес автоматизації обліку персональних фінансів користувачів за допомогою кросплатформених інформаційних систем.

Предмет дослідження – архітектура, функціональні можливості та UX/UI-принципи проєктування спрощеного вебзастосунку для мануального моніторингу доходів і витрат та інтерактивного керування балансом.

Мета роботи – скорочення часу на ведення персонального фінансового обліку та зменшення кількості помилок при фіксації транзакцій шляхом створення вебсервісу з інтуїтивним інтерфейсом, що забезпечує введення, збереження та перегляд фінансових операцій у структурованому вигляді.

Метод дослідження – аналітичний, експериментальний, порівняльний аналіз.

У кваліфікаційній роботі був спроєктований та розроблений вебдодаток, який має назву «MoneySpace» персональний фінансовий трекер із клієнт-серверною архітектурою та системою авторизації користувачів. Застосунок реалізує повний цикл роботи з фінансовими операціями: введення доходів і витрат, збереження у базі даних, перегляд історії транзакцій та автоматичний перерахунок балансу в режимі реального часу.

Клієнтська частина розроблена засобами HTML5, CSS3 та JavaScript без використання зовнішніх фреймворків. Інтерфейс побудовано на основі адаптивної сітки CSS Grid з підтримкою мобільних пристроїв. Серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js та забезпечує обробку HTTP-запитів, маршрутизацію та взаємодію з базою даних. Для зберігання даних використано БД MongoDB у поєднанні з ODM-бібліотекою Mongoose, що дозволяє описувати схеми документів та виконувати валідацію даних.

Безпека застосунку забезпечується системою автентифікації на основі JSON Web Token (JWT) та хешуванням паролів за допомогою бібліотеки bcryptjs. Кожен користувач має доступ виключно до власних фінансових даних завдяки прив'язці транзакцій до ідентифікатора облікового запису. Конфіденційні параметри конфігурації винесено у змінні середовища за допомогою бібліотеки dotenv.

Розроблений застосунок пройшов тестування у локальному середовищі з використанням інструментів браузерного відлагодження DevTools. Вихідний код проєкту опубліковано у репозиторії GitHub. Для спрощення розгортання на сторонніх пристроях реалізовано підтримку контейнеризації засобами Docker та Docker Compose, що дозволяє запустити повноцінний застосунок єдиною командою без необхідності окремого налаштування середовища.

ВЕБДОДАТОК, АВТОРИЗАЦІЯ, JWT, NODE.JS, EXPRESS, MONGODB, HTML, REST API, DOCKER, JAVASCRIPT, ФІНАНСОВИЙ ОБЛІК, ТРАНЗАКЦІЯ, БАЛАНС

SUMMARY

Qualification work: 46 pages, 13 figures, 1 table, 12 sources.

Object of research: the process of automating the accounting of users' personal finances using cross-platform information systems.

Subject of research: architecture, functionality and UX/UI principles of designing a simplified web application for manual monitoring of income and expenses and interactive balance management.

Purpose of work: reducing the time for maintaining personal financial accounting and reducing the number of errors when recording transactions by creating a web service with an intuitive interface that provides input, storage and viewing of financial transactions in a structured form.

Research method: analytical, experimental, comparative analysis.

In the qualification work, a web application called "MoneySpace" was designed and developed, a personal financial tracker with a client-server architecture and a user authorization system. The application implements a full cycle of work with financial transactions: entering income and expenses, saving in the database, viewing transaction history and automatic recalculation of the balance in real time.

The client part is developed using HTML5, CSS3 and JavaScript without using external frameworks. The interface is built on the basis of the adaptive CSS Grid grid with support for mobile devices. The server part is implemented on the Node.js platform using the Express.js framework and provides HTTP request processing, routing and interaction with the database. The MongoDB database is used to store data in combination with the Mongoose ODM library, which allows you to describe document schemas and perform data validation.

The application security is provided by an authentication system based on JSON Web Token (JWT) and password hashing using the bcryptjs library. Each user has access exclusively to their own financial data by linking transactions to the

account identifier. Confidential configuration parameters were transferred to environment variables using the dotenv library.

The developed application was tested in a local environment using the DevTools browser debugging tools. The project source code was published in the GitHub repository. To simplify deployment on third-party devices, containerization support was implemented using Docker and Docker Compose, which allows you to run a full-fledged application with a single command without the need for separate environment configuration.

WEB APPLICATION, AUTHORIZATION, JWT, NODE.JS, EXPRESS, MONGODB, HTML, REST API, DOCKER, JAVASCRIPT, FINANCIAL ACCOUNTING, TRANSACTION, BALANCE

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Реферат 4
Summary 6
Вступ 9
1 Основи веброзробки додатку 12
1.1 Поняття вебдодатку 12
1.2 Огляд існуючих аналогів 13
1.3 Функціональні можливості існуючих рішень 17
1.4 Виисновок до розділу 1 20
2 Проєктування вебдодатку 21
2.1 Визначення технічних вимог 21
2.2 Вибір інструментів розробки 22
2.3 Проєктування інтерфейсу користувача та макетів інтерфейсу 26
2.4 Проєктування функціональної структури додатка 29
2.5 Висновок до розділу 2 32
3 Розробка та реалізація вебдодатка «Moneyspace» 34
3.1 Порядок та реалізація вебдодатку «Moneyspace» 35
3.2 Результати розробки та тестування додатку 39
3.3 Висновок до розділу 3 42
Висновки 44
Перелік використаних джерел 45

ВСТУП

У сучасному світі фінансова грамотність та вміння ефективно керувати власними коштами є запорукою стабільності та добробуту кожної людини. Швидкий темп життя та велика кількість щоденних транзакцій роблять процес ручного обліку витрат складним та рутинним. У зв’язку з цим, розробка вебсервісів для автоматизації персонального фінансового обліку стає надзвичайно актуальною. Такі інструменти дозволяють користувачам не лише фіксувати рух коштів у реальному часі, а й аналізувати свою фінансову поведінку для оптимізації бюджету.

Дана кваліфікаційна робота присвячена проєктуванню та створенню вебдодатку для ведення обліку витрат та доходів. Метою роботи є скорочення часу на ведення персонального фінансового обліку та зменшення кількості помилок при фіксації транзакцій шляхом створення вебсервісу з інтуїтивним інтерфейсом, що забезпечує введення, збереження та перегляд фінансових операцій у структурованому вигляді.

Об’єкт дослідження – процес автоматизації обліку персональних фінансів користувачів за допомогою кросплатформених інформаційних систем.

Предмет дослідження – архітектура, функціональні можливості та UX/UI-принципи проєктування спрощеного вебзастосунку для мануального моніторингу доходів і витрат та інтерактивного керування балансом.

Метою роботи є скорочення часу на ведення персонального фінансового обліку та зменшення кількості помилок при фіксації транзакцій шляхом створення вебсервісу з інтуїтивним інтерфейсом, що забезпечує введення, збереження та перегляд фінансових операцій у структурованому вигляді.

Для реалізації мети дослідження необхідно виконати наступні завдання:

– проаналізувати теоретичні основи, архітектурні особливості та класифікацію сучасних вебдодатків;

– здійснити порівняльний аналіз існуючих програмних аналогів для визначення оптимального функціонального набору спрощеної системи;

– сформулювати технічні вимоги до проєктованої системи, орієнтуючись на високу швидкість мануального введення даних та реалізацію динамічного перерахунку уявного балансу в реальному часі;

– обрати стек технологій для реалізації проєкту;

– провести проєктування архітектури та логічної структури вебдодатку, структури бази даних для надійного збереження ретроспективної історії транзакцій та розробити користувацький інтерфейс за принципом Single Page Application (SPA).

– здійснити програмну реалізацію вебдодатку та провести тестування його компонентів.

Вступна частина роботи охоплює основні етапи та аспекти розробки вебдодатку, що базується на сучасних вебтехнологіях та контейнеризації.

У першому розділі проводиться аналіз предметної області та огляд існуючих рішень для моніторингу персональних фінансів. Розглядаються переваги та недоліки аналогічних систем, проводиться аналіз потреб потенційних користувачів. Окрему увагу приділено обґрунтуванню вибору стеку технологій, зокрема перевагам використання NoSQL рішень та перевагам розгортання застосунку в ізольованих середовищах. Визначаються технічні, функціональні та нефункціональні вимоги до майбутнього програмного продукту.

Другий розділ розкриває процес проєктування системи. Описується архітектура вебсайту, структура бази даних MongoDB та логіка взаємодії між клієнтською та серверною частинами. Значна увага приділяється проєктуванню інтерфейсу користувача (UI/UX) з використанням HTML та CSS, щоб забезпечити зручність введення даних та наочність відображення списку транзакцій. Також розглядається функціональна структура модулів, відповідальних за обробку подій за допомогою JavaScript.

У третьому розділі представлена програмна реалізація вебсайту. Описується процес написання коду для фронтенд-частини та логіки взаємодії з базою даних. Окремим пунктом розглядається процес контейнеризації проєкту за допомогою Docker, що забезпечує стабільну роботу застосунку незалежно від оточення та спрощує його розгортання. Також описується етап тестування основних функцій: коректність введення сум, стабільність збереження даних у БД та правильність виведення історії операцій.

У висновку підводяться підсумки виконаної роботи. Результатом роботи є працездатний вебсайт, що відповідає поставленим завданням та сучасним стандартам веброзробки. Використання JavaScript у поєднанні з MongoDB дозволило створити гнучку систему обробки даних, а впровадження Docker забезпечило технологічність та масштабованість проєкту. Робота демонструє повний цикл створення вебпродукту: від аналізу вимог та проєктування архітектури до реалізації готового до використання інструменту фінансового обліку.

1 ОСНОВИ ВЕБРОЗРОБКИ ДОДАТКУ

1.1 Поняття вебдодатку

Вебдодаток – це тип прикладного програмного забезпечення, що працює на вебсервері та доступне користувачеві через веббраузер. Це означає, що відображення вмістимого сайтів залежить не від операційних систем, а тільки від браузерів та їхніх версій. Тому всі вебдодатки можна назвати кросплатформними програмними продуктами.

На відміну від класичних вебсайтів, які мають переважно інформаційний характер, вебдодатки орієнтовані на активну взаємодію з користувачем: введення, обробку, збереження та маніпуляцію даними.

Вебпрограмування – розділ програмування, орієнтований на розробку вебдодатків (програм, що забезпечують функціонування динамічних сайтів). Мови вебпрограмування призначені для роботи з вебтехнологіями. Мови вебпрограмування можна умовно розділити на дві групи, які часто перетинаються: клієнтські та серверні. Це означає, що відображення вмістимого сайтів залежить не від операційних систем, а тільки від браузерів та їхніх версій. Тому всі вебдодатки можна назвати кросплатформними програмними продуктами.

Вебдодатки, в свою чергу, складаються з «frontend»- та «backend»-частин, тобто клієнтської та серверної частин. Сама розробка сайтів, які можуть бути як статичними, так і динамічно взаємодіяти з користувачем, складається з вебдизайну та вебпрограмування. Розглянемо більш детально ці поняття.

Вебдизайн – галузь веброзробки та різновид дизайну, завданням якого є проєктування користувацьких інтерфейсів для вебдодатків. Вебдизайнери проєктують логічну структуру сторінок вебсайту; продумують найзручніші способи подання інформації; займаються художнім оформленням вебпроєкту.

Технологіями створення вебдизайну є HTML, CSS та інструменти створення адаптивної верстки GridLayout, Flexbox (для коректного відображення вебсторінок на будь-яких гаджетах).[1]

Використання вебдодатків для таких завдань, як облік витрат, надає низку стратегічних переваг як для розробника, так і для кінцевого споживача:

– кросплатформеність: Додаток працює на будь-якому пристрої (ПК, планшет, смартфон), де є браузер. Немає потреби розробляти окремі версії для Windows, Android чи iOS;

– відсутність інсталяції: Користувачу не потрібно завантажувати та встановлювати програмне забезпечення на свій пристрій, що економить пам'ять та час;

– централізоване оновлення: Будь-які зміни в коді (наприклад, виправлення багів у логіці підрахунку балансу) миттєво стають доступними всім користувачам без необхідності оновлювати додаток вручну;

– доступність даних: Оскільки інформація зберігається на сервері, користувач може отримати доступ до своєї історії транзакцій з будь-якої точки світу.

1.2 Огляд існуючих аналогів

На сучасному ринку представлено багато рішень для управління бюджетом, що варіюються від складних екосистем до простих односторінкових трекерів. До найбільш релевантних аналогів можна віднести:

1. Money Lover [2]: Один із найпопулярніших сервісів, який пропонує вебверсію для синхронізації фінансових потоків між різними пристроями. Він дозволяє фіксувати щоденні транзакції за лічені секунди та розподіляти їх за візуалізованими категоріями, такими як «Їжа», «Покупки» чи «Зарплата». Головне вікно застосунку представлено на рисунку 1.1.

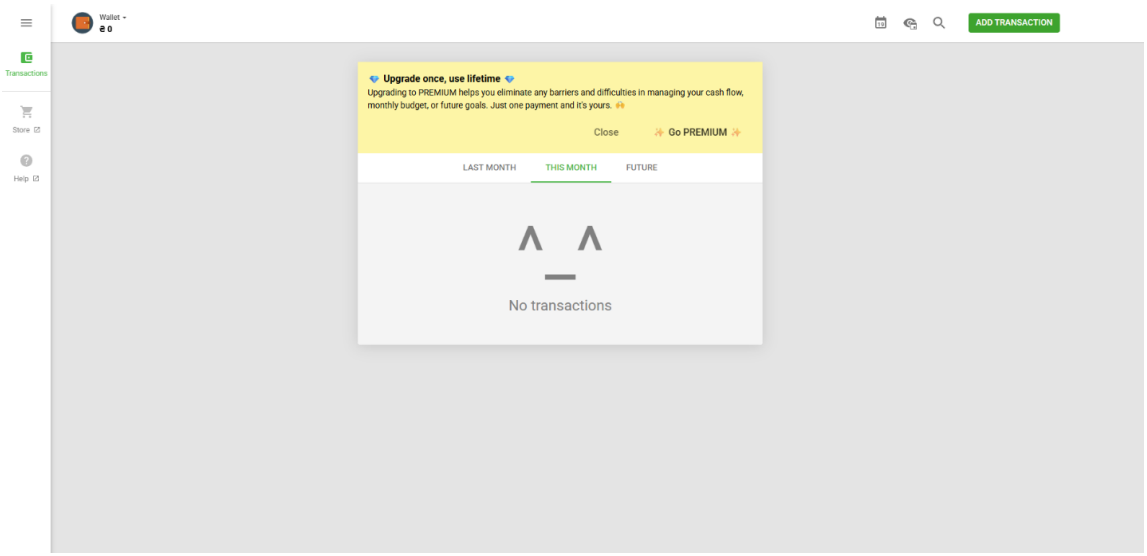


Рисунок 1.1 – Головна сторінка застосунку MoneyLover

2 Wallet by BudgetBakers[3]: Потужний інструмент, вебдодаток якого підтримує імпорт банківських виписок та автоматичну категоризацію витрат. Сервіс акцентує увагу на наданні чіткого огляду фінансового стану користувача для допомоги у дотриманні встановлених бюджетів. Головне вікно застосунку представлено на рисунку 1.2.

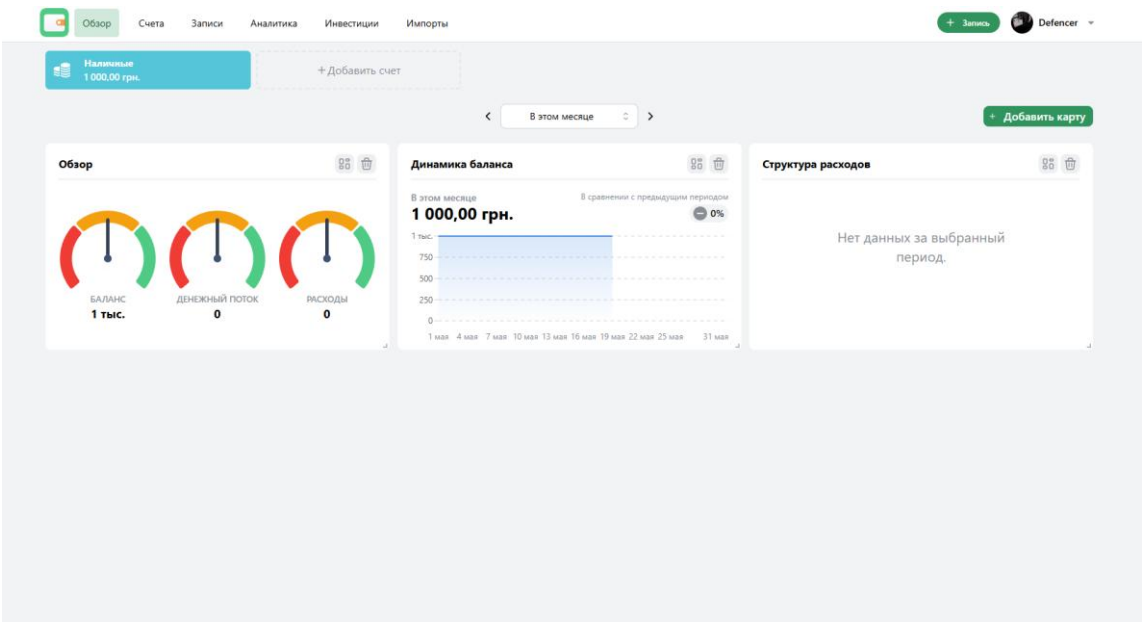


Рисунок 1.2 – Головне вікно застосунку Wallet by Budgetbakers

Порівняльна таблиця аналогів наведена у таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця аналогів

Критерії порівняння	Money Lover	Wallet by Budgetbakers	Обґрунтування впровадження у застосунку, що розробляється
Ергономіка мануального введення транзакцій	Оптимізована: мінімальна кількість кліків, наявність вбудованого калькулятора суми	Перевантажена: вимагає обов'язкового заповнення багатьох додаткових полів	Необхідно впровадити досвід Money Lover. Для спрощеного застосунку швидкість та простота ручного введення доходу/витрати є критично важливою.
Структура списку останніх транзакцій (історія)	Хронологічне групування за днями з чітким візуальним розділенням	Спільний безперервний список із фокусом на аналітичні теги	Необхідно впровадити. Групування списку транзакцій за датами (сьогодні, учора, конкретна дата) полегшує сприйняття історії операцій.
Візуалізація поточного балансу	Відображення загального віртуального балансу на головному екрані	Баланс жорстко прив'язаний до реальних або розрахункових рахунків	Необхідно впровадити. Має бути реалізовано просте інтерактивне поле «Уявний баланс», яке автоматично перераховується після кожної операції.
Кастомізація списку категорій	Дозволяє користувачу швидко створювати власні категорії витрат/доходів	Обмежена, орієнтована на фіксований системний набір категорій	Необхідно впровадити досвід Money Lover. Навіть у спрощеному додатку користувач повинен мати змогу адаптувати назви категорій під власні потреби.
Складність користувацького інтерфейсу (UI/UX)	Низький поріг входження, інтуїтивно зрозуміла навігація	Висока складність через велику кількість аналітичних вкладок	Необхідно впровадити досвід Money Lover. Інтерфейс має бути максимально лаконічним (одноекранна або двоекранна архітектура) без перевантаження зайвими елементами.

Проведений порівняльний аналіз Money Lover та Wallet by Budgetbakers дозволив визначити оптимальний перелік функціональних вимог для розробки спрощеного вебзастосунку з ручним керуванням балансом. Основний акцент зміщено на швидкість взаємодії користувача з інтерфейсом та наочність даних.

До вимог майбутнього вебзастосунку доцільно імплементувати такі рішення:

– оптимізована форма реєстрації операцій (досвід Money Lover): реалізація лаконічної форми додавання транзакцій, яка потребує введення лише трьох обов'язкових параметрів: суми (із вбудованою можливістю простих арифметичних обчислень у самому полі), типу операції (дохід/витрата) та категорії. Це мінімізує часові витрати користувача на мануальне введення;

– динамічний модуль віртуального балансу: розробка центрального елемента інтерфейсу – поля «Уявний баланс», математична модель якого функціонує за принципом інкременту/декременту (додавання сум доходів та віднімання витрат від початкового значення) у режимі реального часу (Real-time update);

– хронологічний реєстр транзакцій: проєктування стрічки останніх дій за принципом блочного групування за датами. Кожен запис має містити чітку колірну індикацію (наприклад, зелений колір для доходів, червоний – для витрат) та відповідну іконку категорії для підвищення швидкості зчитування інформації;

– адаптивна дворівнева структура інтерфейсу: виключення складних аналітичних звітів на користь архітектури Single Page Application (SPA), де на одному екрані збалансовано розміщуються: поточний баланс, кнопка швидкого виклику форми введення та список останніх фінансових операцій.

Основним концептуальним та візуальним натхненням при розробці інтерфейсу «Moneyspace» став сервіс PayPal [4]. Попри те, що PayPal за своєю суттю є платіжною системою та електронним гаманцем, а не просто сервісом для ручного обліку витрат, його підхід до дизайну вважається еталоном у сфері фінансових технологій. Головне вікно застосунку PayPal представлено на рисунку 1.3.

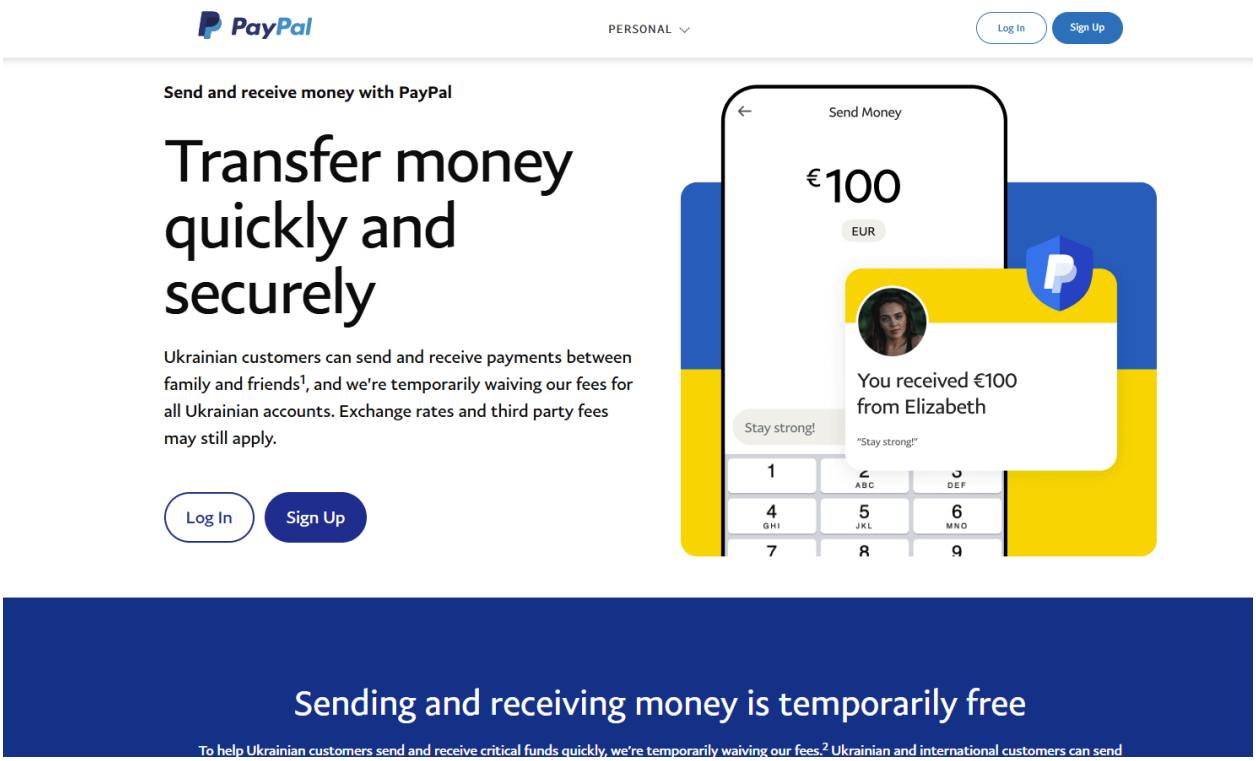


Рисунок 1.3 – Головна сторінка застосунку PayPal

Під час створення інтерфейсу було запозичено такі UX/UI-принципи PayPal:

– візуальний мінімалізм: Світла палітра та багато вільного простору полегшують сприйняття інформації й запобігають перевантаженню цифрами;

– фокус на дашборді: Головний екран об'єднує баланс та історію останніх транзакцій, даючи змогу миттєво оцінити фінансовий стан без зайвих кліків;

– інтуїтивні цільові дії: Контрастні та чіткі кнопки для основних операцій (за аналогією з "Send" та "Request" у PayPal) роблять взаємодію з додатком максимально простою.

1.3 Функціональні можливості існуючих рішень

Дослідження функціональних можливостей існуючих вебдодатків для управління персональними фінансами дозволило з’ясувати різноманітні

аспекти, які є критично важливими для користувача під час щоденного моніторингу доходів та витрат. Перш за все, були детально проаналізовані можливості внесення фінансових операцій у різних форматах. Це включає підтримку різних типів транзакцій, можливість швидкого введення суми, а також гнучкість системи у відображенні позитивного та негативного балансу залежно від характеру операції.

Також були вивчені можливості структурування даних, що дозволяє користувачам бачити чіткий розподіл між загальним капіталом та поточними потоками грошей. Різноманітність відображення інформації забезпечується використанням окремих карток для доходів і витрат, що допомагає користувачеві миттєво оцінити фінансову динаміку. Особлива увага приділена методам архівації та зберігання записів у базі даних, що забезпечує стабільність доступу до історії транзакцій та мінімізує ризик втрати важливих фінансових показників.

Іншим важливим аспектом є наявність функції візуалізації останніх дій у реальному часі, що дозволяє користувачеві контролювати свій бюджет «тут і зараз» та підвищує усвідомленість у витратах. Важливою є також функція миттєвого оновлення загального балансу після кожної нової транзакції, що дозволяє відразу бачити актуальний фінансовий стан без необхідності ручних підрахунків. Окремо розглядалася можливість керування сесіями, що забезпечує персоналізацію досвіду та дозволяє працювати з різними обліковими записами в межах одного інтерфейсу.

Завдяки детальному аналізу стали зрозумілими очікування користувачів від подібних сервісів та сформульовано конкретні вимоги до функціональності вебдодатку. Це дозволило визначити ключові функції, які необхідно реалізувати для створення зручного та ефективного інструменту для ведення обліку доходів та витрат на основі сучасних технологій.

На основі проведеного аналізу існуючих вебдодатків-аналогів виділено такий комплекс їхніх функціональних можливостей:

1 Модуль реєстрації та класифікації фінансових операцій:

– підтримка різних типів транзакцій (мультиформатність введення даних);

– функція експрес-введення (швидкої фіксації) числових значень суми операції;

– диференціація та адаптивне відображення позитивного (доходи) та негативного (витрати) балансу залежно від економічного характеру транзакції.

2 Модуль структурування та архітектури капіталу:

– сегрегація (розділення) фінансових даних на загальний статичний капітал та поточні динамічні грошові потоки;

– візуальна кластеризація інформації за допомогою автономних карток обліку доходів та витрат для експрес-оцінки фінансової динаміки.

3 Модуль збереження даних та ведення ретроспективного обліку:

– архівація та довготривале зберігання фінансових записів у реляційних/нереляційних базах даних;

– забезпечення перманентного (безперебійного) доступу до історичних даних транзакцій із мінімізацією ризиків втрати або пошкодження показників.

4 Модуль операційного моніторингу та інтерактивного оновлення (Real-time processing):

– візуалізація стрічки останніх дій користувача в режимі реального часу для забезпечення оперативного контролю бюджету;

– автоматичний миттєвий перерахунок та актуалізація загального балансу (уявного або реального рахунку) безпосередньо після фіксації кожної нової транзакції (без необхідності мануальних обчислень).

5 Модуль автентифікації та адміністрування сесій:

– керування користувацькими сесіями для забезпечення персоналізації інтерфейсу;

– підтримка ізольованої роботи з різними обліковими записами (акаунтами) в межах єдиного середовища (інтерфейсу) вебдодатка.

1.4 Виисновок до розділу 1

Розуміння основ розробки вебдодатків є фундаментальним етапом у створенні сучасних інформаційних систем. Цей розділ розглядає ключові поняття, переваги та класифікацію вебдодатків, огляд існуючих аналогів - таких як «Money Lover» та «Wallet», а також їхніх функціональних можливостей. Аналіз усіх цих пунктів дає глибоке розуміння очікувань користувачів від сервісів для управління персональними фінансами.

Огляд існуючих аналогів дозволяє визначити найкращі практики у сфері фінтех-дизайну, оцінити їхні переваги та недоліки, а також уникнути типових помилок під час розробки архітектури. Було приділено увагу стилістичному оформленню сервісу «PayPal», що дозволило запозичити принципи лаконічності та функціонального акцентування. Аналіз функціональних можливостей конкурентів допоміг виокремити важливі аспекти: швидкість введення даних, динамічний перерахунок балансу та зручну візуалізацію історії транзакцій.

Ретельний аналіз цих аспектів та розуміння очікувань користувачів створює міцну основу для розробки ефективного вебдодатку, який відповідатиме потребам сучасного користувача у зручному фінансовому обліку та забезпечуватиме надійне зберігання даних за допомогою сучасних технологій.

2 ПРОЄКТУВАННЯ ВЕБДОДАТКУ

Розробка вебдодатку – головний етап у створенні програмного продукту, яка охоплює побудову зручного й інтуїтивного інтерфейсу та формування логіки опрацювання даних, що задовольняє потреби сучасних користувачів у ефективному управлінні особистими фінансами. У цьому розділі докладно висвітлено процес проєктування вебсайту для обліку доходів і витрат, окреслено технічні вимоги, структуру бази даних та застосовувані інструменти розробки.

До головних етапів і ключових складових процесу побудови вебдодатку належать проєктування системної архітектури, розробка макетів сторінок і планування взаємодії з базою даних. Опрацювання концепції вебзастосунку передбачає визначення структури сайту, створення форм для введення фінансових операцій, проєктування блоку останніх транзакцій та механізму збереження інформації у базі даних. Попереднє проєктування дає змогу завчасно сформувати цілісну архітектуру майбутнього додатка.

2.1 Визначення технічних вимог

Вимоги до інтерфейсу користувача: інтерфейс вебдодатку розробляється відповідно до сучасних принципів вебдизайну із застосуванням світлої кольорової гами. Мінімалістичний стиль з акцентом на типографіку забезпечує швидке сприйняття фінансової інформації. Розташування карток балансу, кнопок додавання операцій та списку транзакцій на головній сторінці має бути інтуїтивно зрозумілим. Контрастні кольори для ключових кнопок («Додати дохід», «Додати витрату») та чітка візуальна ієрархія дозволяють користувачу миттєво орієнтуватися у функціоналі.

Вимоги до функціональності: облік фінансових операцій. Додаток повинен надавати можливість вносити дані про доходи та витрати із зазначенням суми та типу операції. Після введення система автоматично перераховує загальний баланс і оновлює показники відповідних блоків.

Збереження даних: кожна введена транзакція має автоматично зберігатися у базі даних, що гарантує цілісність інформації та її доступність при кожному новому сеансі роботи.

Моніторинг транзакцій: блок «Останні транзакції» повинен відображати історію операцій у хронологічному порядку, що забезпечує швидкий перегляд активності без переходу на додаткові сторінки.

Керування обліковим записом: додаток має підтримувати функціонал зміни поточного акаунту та безпечного виходу із системи для забезпечення персоналізації та захисту фінансових даних.

Вимоги до середовища розгортання: вебдодаток розробляється з урахуванням стандартів адаптивної верстки для коректного відображення на різних пристроях. Підтримка актуальних версій браузерів є обов'язковою: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge. Система має бути здатною до розгортання на серверах з будь-якою операційною системою (Linux, Windows, macOS) без додаткового налаштування середовища.

2.2 Вибір інструментів розробки

Для реалізації вебдодатку «MoneySpace» було проаналізовано ряд технологічних рішень і обрано оптимальний стек інструментів, виходячи з вимог проєкту.

1. Мова програмування: JavaScript [5]

Серед мов програмування для вебдодатку розглядалися JavaScript, TypeScript та Python:

– TypeScript є надбудовою над JavaScript з підтримкою статичної типізації, що підвищує надійність великих корпоративних систем, однак для проєкту такого масштабу вносить зайву складність і збільшує час розробки;

– Python (у зв'язці з Flask або Django) добре підходить для серверної логіки, проте потребує використання окремої мови для фронтенду, що ускладнює підтримку єдиної кодової бази;

– JavaScript дозволяє використовувати єдину мову як на клієнтській, так і на серверній стороні (через Node.js), що спрощує архітектуру проєкту, зменшує кількість залежностей і прискорює розробку. Саме тому JavaScript було обрано як основну мову реалізації.

2. Технології розмітки та стилізації: HTML та CSS [6],[7]

Для побудови інтерфейсу розглядалися HTML + CSS, а також фреймворки Bootstrap і Tailwind CSS:

– Bootstrap надає готові UI-компоненти, проте нав'язує власну стилістику і значно збільшує розмір підключених файлів, що негативно впливає на швидкість завантаження;

– Tailwind CSS пропонує гнучке утилітарне стилювання, але вимагає додаткового налаштування збірки та суттєво збільшує поріг входження;

– чистий HTML + CSS забезпечує повний контроль над розміткою і зовнішнім виглядом без зайвих залежностей. Для вебдодатку з фіксованим набором сторінок це є оптимальним рішенням, що гарантує легкість підтримки і мінімальний розмір проєкту.

3. Серверне середовище: Node.js [8]

Як серверна платформа розглядалися Node.js, Django (Python) та Laravel (PHP).

– Django є потужним фреймворком із вбудованою адміністративною панеллю, однак вимагає використання Python на бекенді, що суперечить принципу єдиного мовного стеку проєкту;

– Laravel надає зручні інструменти для побудови MVC-застосунків на PHP, але PHP не забезпечує нативної асинхронності та поступається Node.js за продуктивністю при обробці паралельних запитів;

– Node.js дозволяє використовувати JavaScript на сервері, що забезпечує уніфікованість стеку. Асинхронна модель виконання Node.js ефективно обробляє численні одночасні запити (наприклад, записи транзакцій), що є актуальним для фінансового додатку з частими зверненнями до бази даних.

4. База даних: MongoDB [9]

Для зберігання даних розглядалися MongoDB, PostgreSQL та MySQL:

– MySQL і PostgreSQL є реляційними СУБД із жорсткою схемою таблиць. Вони добре підходять для складних зв'язків між даними, однак для зберігання фінансових транзакцій, які мають просту й однорідну структуру, надлишково ускладнюють розробку;

– MongoDB – документоорієнтована NoSQL база даних, яка зберігає дані у форматі BSON (бінарний JSON). Її гнучка схема дозволяє легко розширювати структуру документа транзакції без міграцій бази. Нативна інтеграція з JavaScript-стеком (через бібліотеку Mongoose) і JSON-подібний формат даних роблять MongoDB природним вибором у межах обраної технологічної платформи.

5. Контейнеризація: Docker [10]

Для розгортання застосунку розглядалися Docker, ручне розгортання на сервері та Vagrant:

– Vagrant дозволяє керувати віртуальними машинами, але потребує значних апаратних ресурсів і не забезпечує такої ж швидкості запуску середовища, як контейнери;

– ручне розгортання є найпростішим підходом, проте суттєво ускладнює відтворення середовища на різних машинах і підвищує ризик конфліктів залежностей;

– Docker ізолює застосунок разом із усіма залежностями в контейнер, що гарантує однакову поведінку середовища незалежно від операційної системи (Linux, Windows, macOS). Це критично важливо для забезпечення стабільності розгортання «MoneySpace» на різних серверах.

6. Редактор коду: Visual Studio Code [11]

Серед редакторів розглядалися VS Code, WebStorm та Sublime Text:

– WebStorm є повноцінним IDE з глибокою підтримкою JavaScript, однак є платним продуктом і споживає значно більше системних ресурсів;

– Sublime Text – легкий і швидкий редактор, проте має обмежену екосистему розширень порівняно з конкурентами;

– VS Code поєднує легкість редактора з широкими можливостями IDE: підтримує інтегрований Git, налагодження Node.js, підсвічування синтаксису та автодоповнення для всіх технологій стеку проєкту. Безкоштовність і кросплатформність роблять його практичним вибором для розробки «MoneySpace».

7. Система контролю версій: Git [12]

Серед систем контролю версій розглядалися Git, SVN та Mercurial:

– SVN і Mercurial є централізованими або менш поширеними системами, що ускладнює інтеграцію з сучасними хостинговими платформами та командну роботу;

– Git є де-факто стандартом галузі з розподіленою архітектурою, підтримкою гілкування та повною інтеграцією з GitHub/GitLab. Використання Git дозволяє відстежувати всі зміни кодової бази «MoneySpace», повертатися до попередніх версій і безпечно вносити зміни через окремі гілки розробки.

Таким чином, обраний стек технологій - JavaScript / HTML / CSS / Node.js / MongoDB / Docker / VS Code / Git – є цілісним і взаємоузгодженим рішенням, де кожен інструмент доповнює інші та відповідає конкретним вимогам проєкту.

2.3 Проєктування інтерфейсу користувача та макетів інтерфейсу

Проєктування інтерфейсу користувача в контексті розробки вебдодатку «Moneyspace» охоплює процес створення зручного, привабливого та легко зрозумілого інтерфейсу для управління персональними фінансами. Це ключовий етап розробки, оскільки інтерфейс визначає швидкість взаємодії з даними та комфорт користувача при щоденному обліку витрат. Макет інтерфейсу вебдодатку – це візуальна концепція, що відображає організацію елементів на сторінці, включаючи картки балансу, кнопки транзакцій та історію операцій.

Макет інтерфейсу допомагає забезпечити однорідність та спростити розробку, оскільки він дає змогу візуально спланувати розміщення інформаційних блоків перед початком написання коду на HTML та CSS. Крім того, наявність детального макета дозволяє ще на етапі проєктування перевірити, чи не перевантажена сторінка зайвими деталями, що особливо важливо для фінансових інструментів.

Основні аспекти проєктування інтерфейсу включають наступні.

1. Дизайн і стиль інтерфейсу.

Визначення загального вигляду та стилю вебдодатку. Для проєкту «Moneyspace» обрано сучасний «Clean Design» із використанням світлої кольорової гами та акцентного синього кольору для ключових елементів керування. Цей стиль відповідає сучасним тенденціям веброзробки та забезпечує високу читабельність цифрових показників. Використання чіткого беззасічкового шрифту та великих карток із закругленими кутами дозволяє користувачеві з першого погляду зрозуміти свій фінансовий стан. Привабливий та охайний інтерфейс сприяє довірі користувача до системи та підвищує задоволеність від щоденного моніторингу витрат.

2. Розташування елементів інтерфейсу.

Організація та розміщення блоків на екрані. На головній сторінці «Moneyspace» реалізована чітка ієрархія: зверху розташовано блок загального

балансу, під ним – розділені картки доходів та витрат, що дозволяє миттєво оцінити фінансову ситуацію. Кнопки «Додати дохід» та «Додати витрату» розміщені в центрі робочої області для швидкого доступу. Нижню частину екрана займає розділ «Останні транзакції», який візуально відокремлений від блоків статистики, що забезпечує логічний розподіл між підсумковими даними та детальною історією дій.

3. Навігація.

Проєктування системи навігації у вебдодатку. В «Moneyspace» реалізована спрощена навігаційна модель, де всі основні функції зосереджені на одному екрані, що мінімізує кількість кліків. У верхній частині сторінки (хедері) розташовані допоміжні елементи: зміна акаунту та кнопка виходу. Це дозволяє користувачеві легко перемикатися між обліковими записами, не втрачаючи фокус на основному контенті. Така архітектура забезпечує швидкий перехід між функціями введення даних та перегляду статистики без плутанини.

4. Взаємодія з елементами.

Розробка інтерфейсу, який забезпечує інтуїтивну взаємодію за допомогою JavaScript. Взаємодія з елементами «Moneyspace» побудована на принципі миттєвого відгуку. При натисканні на кнопки додавання операцій користувач отримує можливість ввести суму, а зміни в блоці недавніх транзакцій та перерахунок балансу відбуваються динамічно. Також передбачено відображення інформаційних повідомлень, наприклад, текст «Транзакцій ще немає», який підказує користувачеві перші кроки, що значно покращує загальне враження від використання вебдодатку та полегшує процес освоєння інструменту.

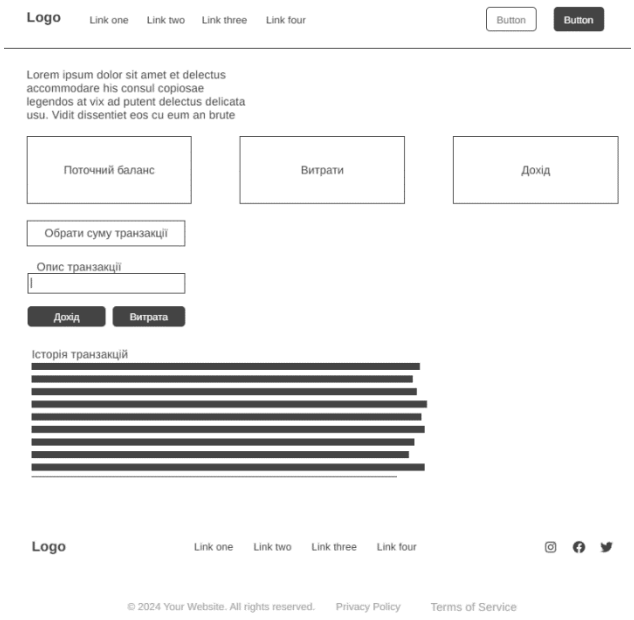


Рисунок 2.1 – Макет головної сторінки вебсайту

Після розгляду головних аспектів проєктування інтерфейсу були розроблені макети сторінок додатку, які представлені на рисунку 2.1 та 2.2.

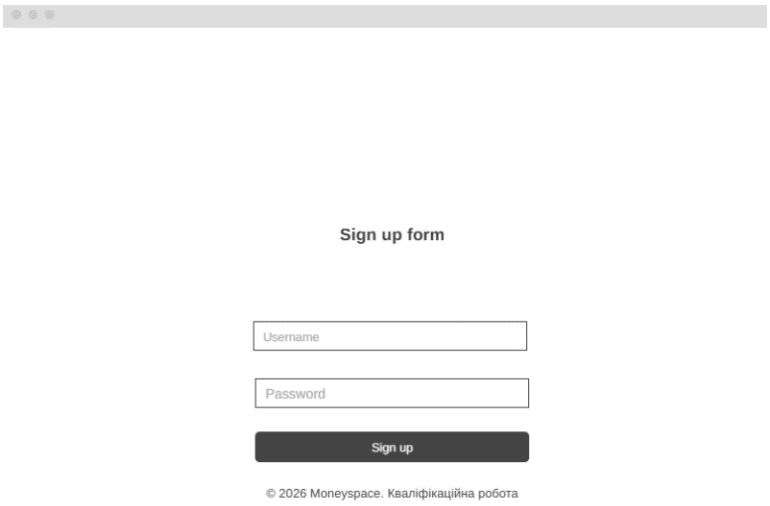


Рисунок 2.2 – Макет логін-сторінки вебсайту

2.4 Проєктування функціональної структури додатка

Функціональна структура вебдодатку описує сукупність взаємопов’язаних модулів, які забезпечують виконання цільових завдань користувача: введення даних, їх обробку, збереження та візуалізацію. Завдяки використанню технологій JavaScript та MongoDB, структура є гнучкою та дозволяє обробляти фінансові операції в режимі реального часу.

Основні компоненти функціональної структури:

1 Модуль аутентифікації та керування доступом: забезпечує ідентифікацію користувача, можливість зміни акаунту та безпечний вихід із системи.

2 Модуль управління транзакціями: відповідає за обробку подій при натисканні кнопок «Додати дохід» та «Додати витрату», включаючи валідацію введених сум

3 .Модуль розрахунку балансу: динамічно обчислює різницю між загальними доходами та витратами для відображення актуального стану рахунку.

4 Модуль взаємодії з БД (Data Access Layer): забезпечує запис нових транзакцій у MongoDB та отримання історії операцій.

5 Модуль відображення (UI Update): відповідає за миттєве оновлення інтерфейсу без перезавантаження сторінки (AJAX/Fetch API).

Унаслідок аналізу всіх компонентів системи була побудована діаграма компонентів, що зображена на рисунку 2.3.

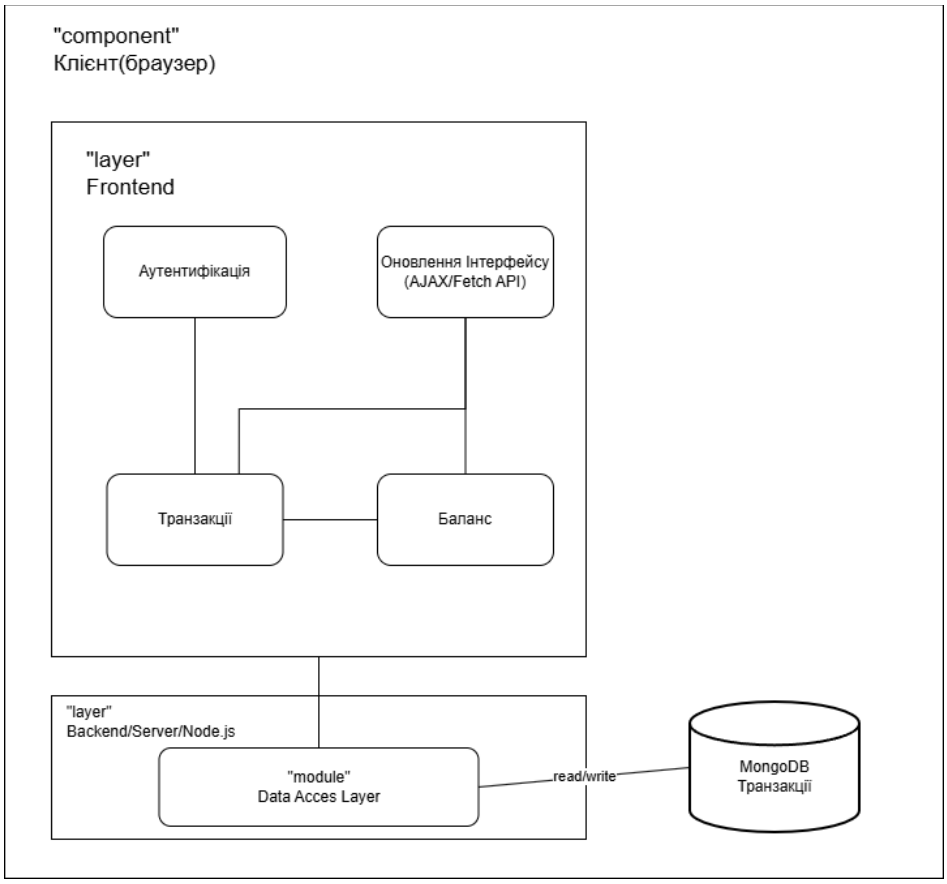


Рисунок 2.3 – Діаграма компонентів «MoneySpace»

Взаємодія модулів та порядок виконання операцій. Процес роботи вебдодатку відбувається за наступним алгоритмом:

– ініціалізація: При завантаженні сторінки модуль відображення надсилає запит до бази даних через серверний прошарок для отримання списку останніх транзакцій;

– введення даних: Користувач активує форму введення. Після натискання кнопки підтвердження, модуль управління транзакціями перевіряє коректність даних (наприклад, щоб сума не була від’ємною або порожньою);

– опрацювання та збереження: Дані передаються в базу даних MongoDB. Після успішного підтвердження запису, модуль розрахунку перераховує «Загальний баланс», а модуль відображення додає нову операцію у вікно «Останні транзакції».

Взаємодія модулів зображена на рисунку 2.4.



Рисунок 2.4 – Блоксхема алгоритму роботи вебдодатку

Обробка даних у «Moneyspace» буде здійснюватись на двох рівнях:

1 Клієнтський рівень (Frontend): JavaScript виконує первинну обробку та форматування чисел (додавання знаку «грн», зміна кольору тексту: зелений для доходів, червоний для витрат).

2 Серверний рівень (Backend): Забезпечує безпечне передавання JSON-об'єктів між інтерфейсом та БД. Використання Docker дозволяє ізолювати середовище виконання, гарантуючи, що методи обробки даних працюватимуть ідентично на будь-якому хостингу.

Проєктована структура передбачає легке масштабування системи в майбутньому:

– категоризація: додавання можливості призначати категорію кожній транзакції (їжа, транспорт, відпочинок);

– візуалізація: інтеграція бібліотек для побудови графіків та діаграм на основі накопичених даних з MongoDB;

– експорт звітів: реалізація функції вивантаження історії транзакцій у форматах PDF або Excel;

– мультивалютність: впровадження конвертації валют за актуальним курсом;

– можливість зміни мови: інтегрування можливості змінювати мову інтерфейсу.

Дана функціональна структура забезпечує стабільну роботу поточної версії «Moneyspace» та створює надійний фундамент для подальшого розвитку вебсервісу.

2.5 Висновок до розділу 2

Підсумовуючи процес проєктування вебдодатку «Moneyspace», можна зробити висновок, що цей етап став фундаментом для створення цілісної та функціональної системи управління персональними фінансами. У ході роботи

було детально опрацьовано архітектуру застосунку, що дозволило поєднати сучасні технологічні рішення з потребами кінцевого користувача.

Основні результати проєктування включають:

– обґрунтування технологічного стеку: Вибір комбінації HTML/CSS/JavaScript для фронтенду, MongoDB для збереження даних та Docker для контейнеризації забезпечив високу продуктивність, гнучкість обробки неструктурованих даних та легкість розгортання проєкту в будь-якому середовищі.

– створення візуальної концепції: Розроблений макет інтерфейсу базується на принципах мінімалізму та «Clean Design». Чітке структурування інформаційних блоків – від загального балансу до вікна недавніх транзакцій – забезпечує інтуїтивне розуміння функціоналу сайту з першого погляду;

– визначення технічних вимог: Встановлені стандарти щодо адаптивності, кросбраузерності та безпеки (модулі аутентифікації та зміни акаунту) гарантують стабільну роботу додатка на різних пристроях та захист фінансової інформації;

– функціональна архітектура: Проєктування функціональної структури дозволило налагодити чітку взаємодію між діями користувача (введення доходів/витрат) та серверною частиною. Динамічне оновлення інтерфейсу без перезавантаження сторінки значно підвищує якість користувацького досвіду.

Таким чином, етап проєктування дозволив трансформувати абстрактну ідею обліку витрат у чіткий технічний план. Створена архітектура не лише повністю задовольняє поточні функціональні вимоги щодо ведення обліку доходів і витрат, а й має високий потенціал для подальшого масштабування, включаючи впровадження аналітичних графіків та розширеної категорії операцій. Це створює надійну базу для програмної реалізації проєкту, яка буде розглянута в наступному розділі.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ВЕБДОДАТКА «MONEYSPACE»

Розробка вебдодатку є ключовим етапом у процесі створення програмного забезпечення для обліку особистих фінансів. Цей етап охоплює безпосередню реалізацію задуманої функціональності та інтерфейсу користувача, програмування серверної та клієнтської частин, налаштування бази даних, а також тестування й оптимізацію додатку відповідно до визначених вимог.

В процесі роботи використовувалось інтегроване середовище розробки Visual Studio Code, яке забезпечує зручний інтерфейс для написання коду, налагодження застосунку та роботи з системою контролю версій. VS Code також надає розширену підтримку синтаксису для всіх використаних технологій завдяки відповідним розширенням.

Клієнтська частина вебдодатку «Moneyspace» реалізована засобами HTML, CSS та JavaScript. HTML-розмітка визначає структуру сторінок, зокрема форми введення доходів і витрат та елементи відображення списків транзакцій. CSS забезпечує стилізацію інтерфейсу та адаптивне відображення елементів, а JavaScript відповідає за динамічну взаємодію з користувачем: обробку подій, валідацію введених даних та асинхронний обмін даними із сервером.

Серверна частина розроблена на основі платформи Node.js, яка дозволяє виконувати JavaScript-код на стороні сервера та забезпечує обробку HTTP-запитів від клієнта. За допомогою Node.js реалізовано RESTful API для операцій створення, читання, оновлення та видалення фінансових записів користувача.

Для зберігання даних використовується СУБД MongoDB, яка розгорнута локально. MongoDB є документоорієнтованою базою даних, що дозволяє зберігати записи про доходи та витрати у вигляді гнучких JSON-подібних документів без необхідності жорсткої схеми. Кожен запис містить тип

операції, суму, категорію та дату, що забезпечує зручне подальше сортування й фільтрацію.

Для спрощення розгортання та забезпечення ізольованого середовища виконання застосовувався Docker. Контейнеризація дозволяє уникнути залежності від конфігурації конкретної операційної системи та гарантує відтворюваність середовища розробки й запуску застосунку. MongoDB запускалась у вигляді окремого Docker-контейнера, що спрощує керування її версією та налаштуваннями.

3.1 Порядок та реалізація вебдодатку «Moneyspace»

Спочатку було встановлене робоче середовище розробки Visual Studio Code. Було створене базове дерево проєкту, що передбачає розташування файлів у робочому середовищі. Повна ієрархія проєкту представлена на рисунку 3.1.

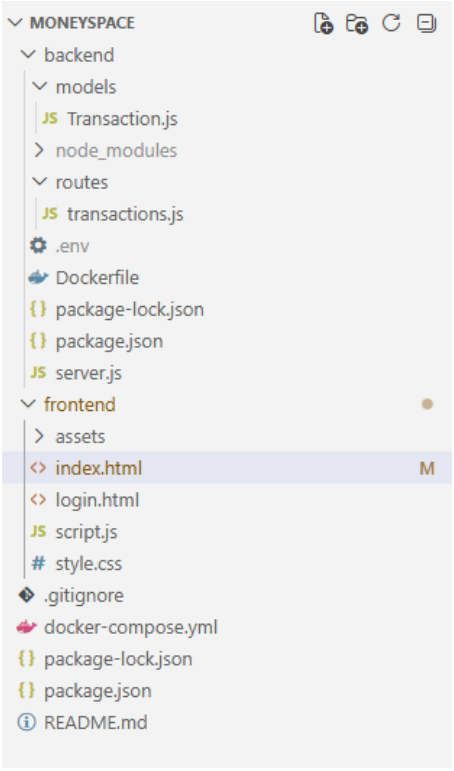


Рисунок 3.1 – Ієрархія файлів вебзастосунку

На першому етапі було розроблено клієнтську частину застосунку. Створено HTML-розмітку головної сторінки index.html, яка містить шапку з логотипом, який був згенерований штучним інтелектом, та назвою проєкту, інформаційні картки для відображення загального балансу, доходів і витрат, кнопки додавання операцій, модальне вікно з формою введення даних, а також секцію для відображення історії транзакцій. Для стилізації інтерфейсу створено файл style.css з використанням шрифту Inter, кольорової палітри у синьо-білих тонах та адаптивної сітки на основі CSS Grid. Шаблонний вигляд вебсторінки зображений на рисунку 3.2.

Початковий скрипт script.js реалізував логіку відкриття та закриття модального вікна з підтримкою клавіші Escape і закриття по кліку на фон, модальне вікно зображено на рисунку 3.3.

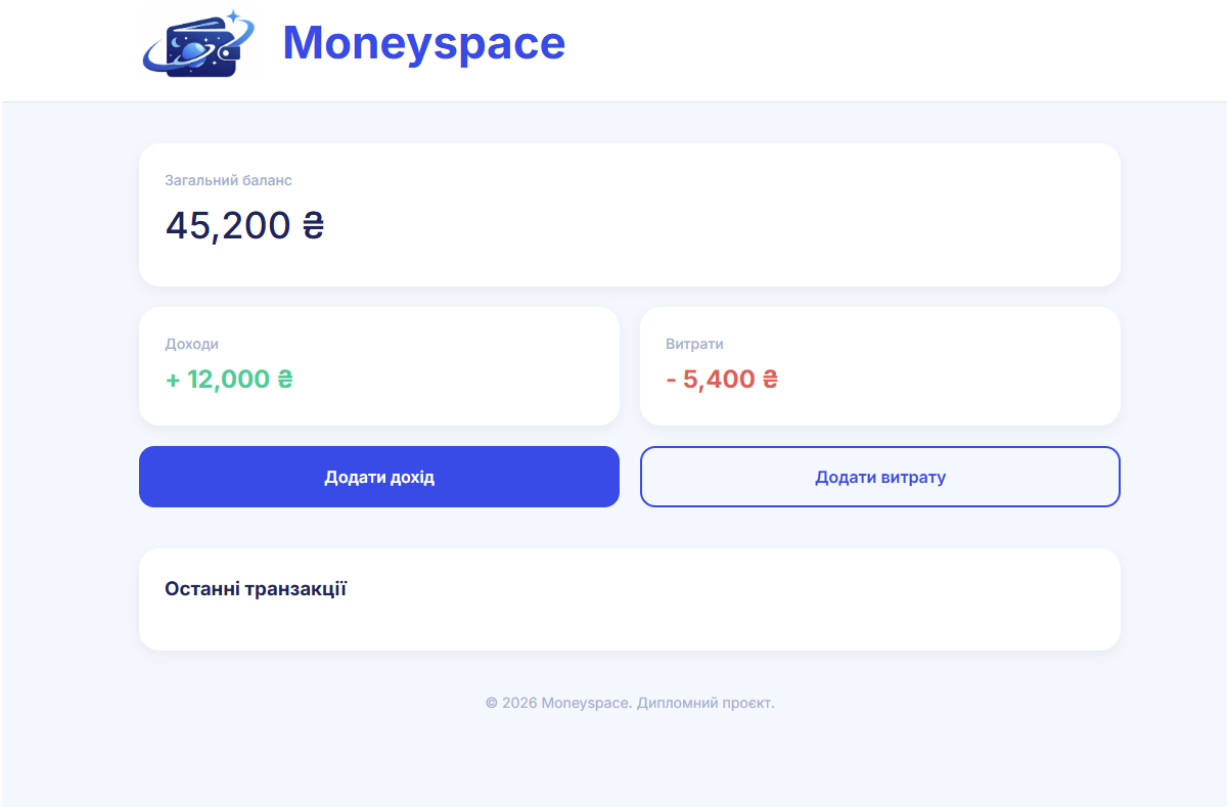


Рисунок 0.2 – Шаблон вебсторінки

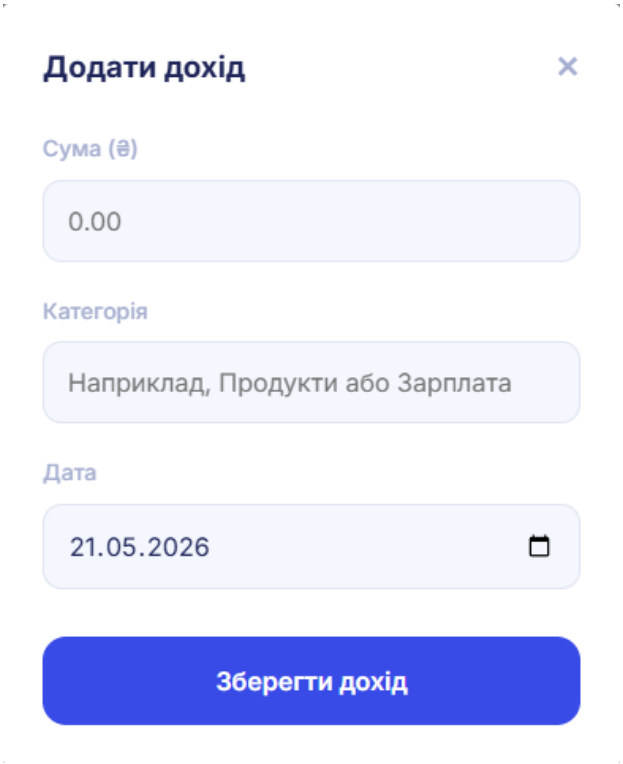


Рисунок 0.3 – Модальне вікно

На другому етапі реалізовано серверну частину застосунку на основі платформи Node.js з використанням фреймворку Express. Для зберігання даних обрано СУБД MongoDB у локальному режимі розгортання, що забезпечило незалежність від зовнішніх мережевих сервісів. Для взаємодії з базою даних використано бібліотеку Mongoose, яка дозволяє описувати схеми документів та виконувати валідацію даних на рівні моделі.

Було визначено схему транзакції з полями типу операції, суми, категорії та дати. Сервер реалізує три маршрути REST API: GET /api/transactions для отримання всіх транзакцій, POST /api/transactions для збереження нової операції та DELETE /api/transactions/:id для видалення запису за ідентифікатором. Конфіденційні параметри підключення винесено у файл змінних середовища .env.

Клієнтський скрипт було розширено функціями взаємодії з API. Реалізовано асинхронні функції fetchTransactions, saveTransaction та deleteTransaction, що використовують Fetch API для виконання HTTP-запитів до серверної частини. Додано функції форматування валюти та дати

відповідно до української локалі, а також систему сповіщень у вигляді тостів для інформування користувача про результат операцій. При завантаженні сторінки автоматично викликається функція loadAll, яка отримує всі транзакції з бази даних, обчислює загальний баланс, суму доходів і витрат та відображає їх на сторінці.

На третьому етапі реалізовано систему автентифікації користувачів на основі технології JWT (JSON Web Token). До серверної частини додано модель користувача з полями логіну та хешованого паролю. Для хешування паролів використано бібліотеку bcryptjs, що забезпечує безпечне зберігання облікових даних. Реалізовано маршрут POST /api/auth/login, який перевіряє введені облікові дані та у разі успіху повертає підписаний токен з терміном дії сім днів. Створено захищений службовий маршрут POST /api/auth/create-user для додавання нових користувачів через термінал з використанням адміністративного ключа. Всі маршрути транзакцій захищено middleware-функцією protect, яка перевіряє наявність та валідність токена у заголовку запиту Authorization. Транзакції прив'язано до конкретного користувача через поле user, що містить ідентифікатор відповідного запису в колекції користувачів.

Створено окрему сторінку автентифікації login.html з формою введення логіну та паролю, яка при успішному вході зберігає токен і ім'я користувача у localStorage браузера та виконує перенаправлення на головну сторінку. Головна сторінка при завантаженні перевіряє наявність токена і за його відсутності перенаправляє користувача на сторінку входу. У шапку додано блок з відображенням імені поточного користувача та кнопками виходу з облікового запису і зміни акаунту. Клієнтський скрипт оновлено таким чином, що всі запити до API передають токен у заголовку, а у разі отримання відповіді зі статусом 401 автоматично виконується перенаправлення на сторінку входу.

На завершальному етапі проєкт підготовлено до публікації та передачі. Створено файл .gitignore для виключення з репозиторію конфіденційних даних і залежностей, а саме директорії node_modules та файлу .env.

Репозиторій опубліковано на платформі GitHub. Для спрощення розгортання застосунку на сторонніх пристроях додано підтримку Docker: створено файл Dockerfile з описом образу серверної частини на основі Node.js та файл docker-compose.yml, який описує два сервіси – серверний застосунок та базу даних MongoDB. Це дозволяє розгорнути повноцінний застосунок на будь-якому комп'ютері з встановленим Docker за допомогою єдиної команди docker-compose up без необхідності окремого встановлення Node.js та MongoDB.

3.2 Результати розробки та тестування додатку

В цьому розділі представлені результати розробки вебдодатку «MoneySpace» для ведення обліку з особистих доходів і витрат.

Розроблено вебдодаток, за допомогою якого користувач може вести облік власних доходів і витрат. Інтерфейс головної сторінки додатка спроєктовано з урахуванням сучасних вимог до вигляду вебдодатків. Програмне забезпечення надає користувачеві повний набір інструментів для ефективного контролю фінансового стану в режимі реального часу.

У верхній панелі вебдодатка реалізовано відображення імені поточного авторизованого користувача, а також інтегровано кнопки керування сесією, які дозволяють змінити обліковий запис та безпечно вийти із системи з автоматичним очищенням токенів.

Основний блок системи складається з трьох інформаційних карток. Головна панель відображає поточний загальний баланс користувача, який автоматично розраховується як різниця між усіма зафіксованими надходженнями та витратами. Нижче розташовані дві окремі картки для деталізації фінансових потоків: панель доходів із зеленим підсвічуванням та панель витрат із червоним, що забезпечує швидке сприйняття інформації. Готова головна сторінка вебдодатку зображена на рисунку 3.4.

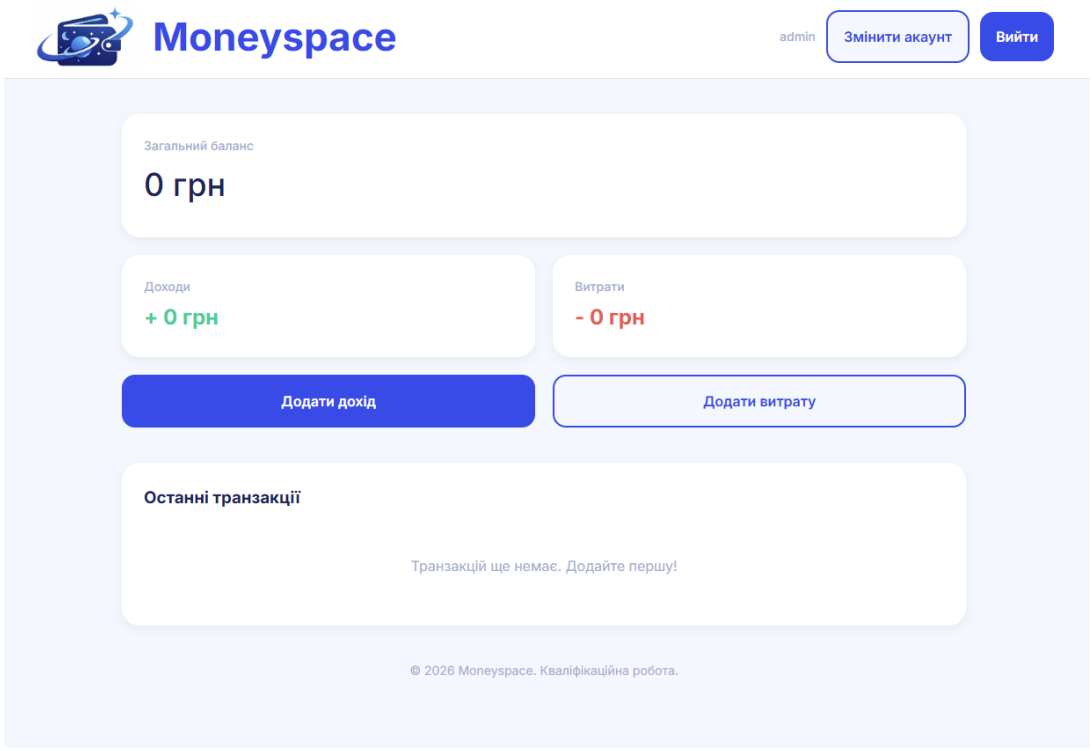


Рисунок 0.4 – Вигляд головної сторінки вебдодатку

Для взаємодії із додатком користувач повинен спочатку зайти у систему під своїм логіном і паролем. Інакше до вебдодатку користувач зайти не зможе, так як інформація про доходи і витрати є конфіденційною. Форма входу зображена на рисунку 3.5.

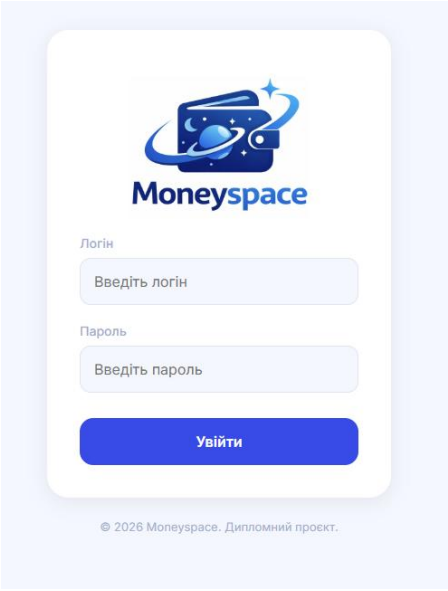


Рисунок 0.5 – Вікно входу

Щоб протестувати додаток, були введені дані через відповідну форму. Після чого транзакції відобразились у списку останніх транзакцій. Скрипт відобразив всі введені дані як у списку транзакцій, так і у відповідних картках. Результат тестування зображений на рисунку 3.6.

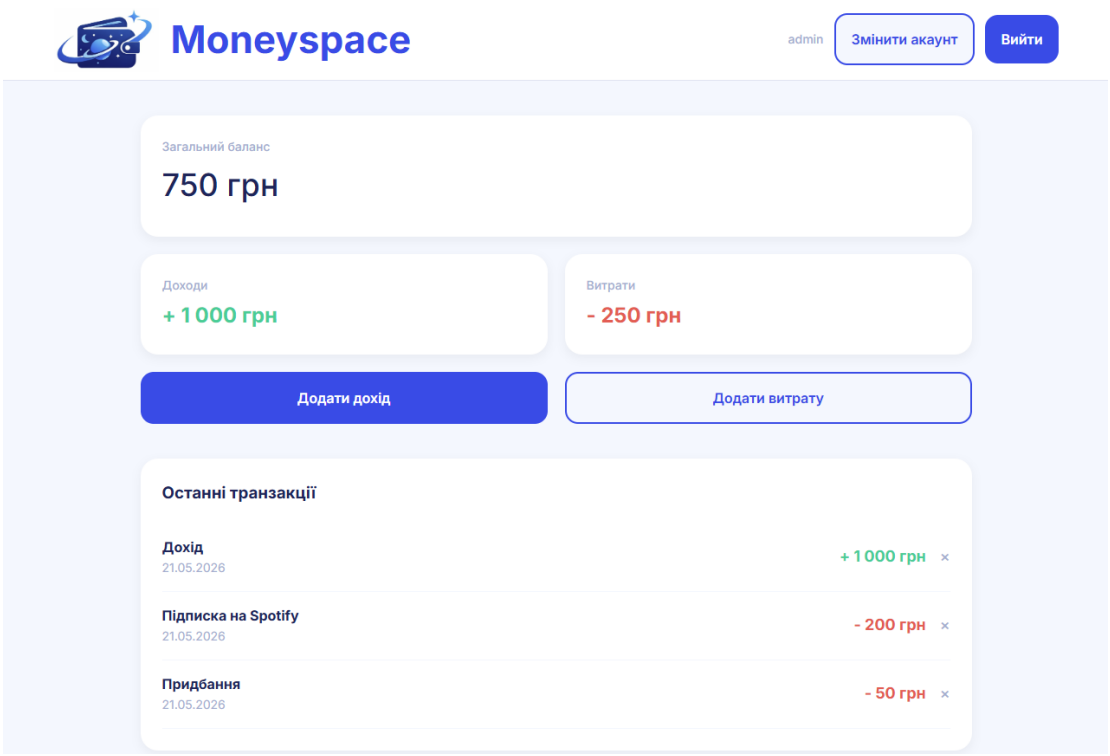


Рисунок 0.6 – Тестування основних функцій

У підсумку можна зазначити, що додаток «Moneyspace» є ефективним та зручним інструментом для обліку власних доходів і витрат.

У процесі самостійного вивчення сучасних технологій веброзробки та роботи над проєктом було отримано практичні знання, які безпосередньо застосовувались під час створення «MoneySpace».

1 JavaScript та Node.js. Засвоєно ключові концепції JavaScript: асинхронне програмування з async/await, робота з Fetch API, обробка подій та маніпуляції з DOM. На серверному рівні опановано побудову REST API засобами Express.js – маршрутизацію запитів, написання middleware-функцій та обробку помилок. Отримано практичний досвід роботи з бібліотеками:

jsonwebtoken для авторизації, bcryptjs для захисту паролів та dotenv для керування конфігурацією.

2 Бази даних та MongoDB. Набуто навички проєктування схем даних через Mongoose – створення моделей користувачів і транзакцій, налаштування зв'язків між колекціями та валідації полів. Отримано досвід інтеграції локальної бази даних у реальний проєкт: зберігання, сортування, фільтрація та безпечне видалення користувацьких записів.

3 Інструменти розробки. Опановано повний цикл розробки у середовищі VS Code, керування залежностями через npm та налагодження коду за допомогою браузерних DevTools. Сформовано розуміння того, як правильно інтегрувати фронтенд із захищеним бекендом у повноцінному вебпроєкті.

4 Професійний розвиток. Робота над проєктом дала не лише технічні знання, а й реальний досвід прийняття архітектурних рішень. Поглиблено розуміння веббезпеки: робота з JWT-токенами, захист маршрутів та ізоляція даних користувачів. Отримано практичні навички аналізу HTTP-запитів, обробки кодів відповідей сервера, налаштування CORS та побудови динамічних інтерфейсів без перезавантаження сторінки.

3.3 Висновок до розділу 3

В цьому розділі було детально розглянуто процес розробки, реалізації та тестування вебдодатка «MoneySpace».

Розглянуто архітектуру створеного додатка, який складається з декількох компонентів: серверної частини на базі Node.js та Express, системи авторизації за допомогою вебтокенів (JWT), бази даних MongoDB, а також динамічного клієнтського інтерфейсу. Кожен компонент виконує специфічні функції, забезпечуючи загальну працездатність, безпеку персональних даних та зручність використання додатка. Структура додатка була спроєктована з

дотриманням принципів модульності та розширюваності, що дозволяє легко додавати нові функції, такі як категорії для аналітики чи генерація фінансових звітів, у майбутньому.

Завдяки ретельному проєктуванню, правильному вибору технологій та грамотній реалізації вдалося створити повноцінний вебдодаток «MoneySpace», який відповідає поставленим вимогам та задовольняє потреби користувачів у зручному веденні обліку власних доходів і витрат. Додаток має інтуїтивно зрозумілий інтерфейс, забезпечує надійне зберігання фінансової історії, ізоляцію даних різних користувачів та надає зворотній зв’язок у вигляді миттєвого перерахунку загального балансу в режимі реального часу.

ВИСНОВКИ

Результатом даної кваліфікаційної роботи стало створення вебдодатку «MoneySpace» для персонального обліку доходів і витрат. У процесі роботи були проаналізовані потреби цільової аудиторії, існуючі аналоги та сучасні підходи до архітектури клієнт-серверних систем, що дозволило визначити оптимальні технології для реалізації проєкту.

Завдяки правильному вибору технологій та грамотній реалізації вдалося створити повноцінний вебдодаток, що відповідає всім поставленим вимогам. Застосунок має інтуїтивно зрозумілий інтерфейс, забезпечує надійну ізоляцію даних через систему авторизації та динамічний перерахунок балансу в режимі реального часу. Додаток пройшов комплексне тестування у локальному середовищі з використанням інструментів вебвідлагодження DevTools, а вихідний код опубліковано на GitHub для вільного розгортання та тестування.

«MoneySpace» має добрий потенціал для подальшого розвитку завдяки модульній структурі коду та гнучкій REST API архітектурі, що відкриває можливості для додавання візуалізації статистики, розширених категорій витрат та інших функцій.

Робота над проєктом стала важливим етапом професійного становлення. У процесі розробки було отримано практичний досвід серверного програмування на Node.js та Express.js, опановано взаємодію з базою даних MongoDB через Mongoose, а також реалізовано захист даних за допомогою JWT-токенів та хешування паролів.

Навчання у ВСП «Економіко-правничий фаховий коледж ЗНУ» за спеціальністю 121 Інженерія програмного забезпечення надало теоретичне підґрунтя для успішної реалізації цього проєкту та відкрило широкі перспективи для подальшого професійного зростання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. К. В. Двірничук, Д. О. Вацек Навчальний посібник “ВЕБПРОГРАМУВАННЯ ТА ВЕБДИЗАЙН”, URL: https://abitap.com/wp-content/uploads/2022/12/all_web_26_grudnya_2022.pdf (дата звернення: 19.05.2026).

2. MoneyLover Official Website. URL: https://web.moneylover.me/ (дата звернення: 19.05.2026).

3. Wallet by. by BudgetBakers Official Website. URL: https://budgetbakers.com/en/ (дата звернення: 19.05.2026).

4. PayPal Official Website. URL: https://www.paypal.com/home (дата останнього звернення: 19.05.2026).

5. JavaScript. Вікіпедія: вільна енцикл. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 21.05.2026).

6. HTML. Вікіпедія: вільна енцикл. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 19.05.2026).

7. CSS. Вікіпедія: вільна енцикл. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення: 19.05.2026).

8. Node.js Free Download. Гіперпосилання: https://nodejs.org/en/download (дата звернення: 19.05.2026).

9. MongoDB. URL: https://www.mongodb.com/ (дата звернення: 19.05.2026).

10. Docker. URL: https://www.docker.com/ (дата звернення: 19.05.2026).

11. Visual Studio Code Official Website. Free download. URL: https://code.visualstudio.com/ (дата звернення: 19.05.2026).

12. Git. URL: https://git-scm.com/book/uk/v2/%D0%92%D1%81%D1%82%D1%83%D0%BF-%D0%9F%D1%80%D0%BE-

%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%83-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8E-%D0%B2%D0%B5%D1%80%D1%81%D1%96%D0%B9 (дата звернення: 19.05.2026).

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничий фаховий коледж ЗНУ»

Я, Коломойцев Іван Олександрович, здобувач освіти 4 курсу, спеціальності 121 Інженерія програмного забезпечення, освітньо-професійної програми Розробка програмного забезпечення, групи К121-22, адреса електронної пошти vanakolomojcev30@gmail.com

- підтверджую, що написана мною дипломна робота на тему «Проєктування та розробка вебдодатку з обліку особистих доходів і витрат» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;

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

- згоден/згодна 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 27.05.2026 Підпис  Іван КОЛОМОЙЦЕВ

Дата _____ Підпис _____ Олена ШИТІКОВА