Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту)
Кафедра обчислювальної техніки
(повна назва кафедри )

**Комплексна бакалаврська кваліфікаційна робота на тему:**
«Онлайн-сервіс для фінансового планування.Частина 2. Клієнтський модуль»
08-54. КБКР.042.00.000 ТЗ

Виконав: студент 4 курсу, групи 2КІ-21б
спеціальності
123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
освітня програма «Комп'ютерна інженерія»
Самусь О.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. МБІС
Грицак А.В.
(прізвище та ініціали)

**Допущено до захисту**
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«18» червня 2025 р.

Вінниця 2025

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії

Кафедра Обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 - Інформаційні технології

Спеціальність 123 - «Комп'ютерна інженерія»

Освітня програма «Комп'ютерна інженерія»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д
«21» березня 2025 р.

# З А В Д А Н Н Я
## НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Самусю О.В.
(прізвище, ім'я, по батькові)

1 Тема роботи «Онлайн-сервіс для фінансового планування.Частина 2. Клієнтський модуль», керівник роботи к.т.н., доц. Тарновський М. Г, затверджені наказом ВНТУ від «20» березня 2025 року №97.

2 Термін подання студентом роботи 13.06.2025.

3 Вихідні дані до роботі: функціональні вимоги — облік доходів і витрат, його візуалізація; засоби розробки — HTML/CSS, JavaScript, фреймворки React/Vue, Python/Django, Postman, DBeaver, середовище розробки — PyCharm Professional.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз сучасних технологій з інформаційної підтримки управління фінансами, аналіз існуючих рішень у сфері фінансового планування; огляд архітектури даних у сервісі фінансового планування та технологій інтеграції з банківськими та фінансовими API, проектування клієнтської частини сервісу для фінансового планування, висновки.

5 Перелік графічного матеріалу: відображення сторінки реєстрації, авторизації,

відображення головної сторінки сайту, відображення компонентів карточки для перегляду категорій витрат, відображення аналізу криптовалюти, відображення новин на головній сторінці, відображення банківської карточки, відображення підписок, схема обміну даними.

6 Консультанти розділів роботи приведений в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Видав | Прийняв |
| 1-4 | Тарновський М. Г, доцент кафедри ОТ | 21.03.25 | 13.05.25 |
| Нормоконтроль | ас. каф. ОТ Швець С. І. | 14.05.25 | 30.05.25 |

7 Дата видачі завдання « 21 » березня 2025 р.

8 Календарний план виконання КБКР приведений в таблиці 2.

Таблиця 2 — Календарний план

| № з/п | Назва етапів виконання комплексної бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи | Підпис |
|---|---|---|---|
| 1 | Постановка задачі роботи | 21.03.25 — 30.05.25 | Виконано |
| 2 | Огляд існуючих методів для інтеграції банківської API | 21.03.25 — 30.05.25 | Виконано |
| 3 | Аналіз та вибір методів для розроки веб-сайту | 21.03.25 — 30.05.25 | Виконано |
| 4 | Розробка структури веб-сайту | 31.03.25—13.04.25 | Виконано |
| 5 | Розробка архітектури веб-сайту | 14.04.25—27.04.25 | Виконано |
| 6 | Підготовка матеріалів, та опис розробки онлайн-сервісу фінансового планування | 14.04.25—27.04.25 | Виконано |
| 7 | Оформлення пояснювальної записки | 14.04.25—27.04.25 | Виконано |
| 8 | Аналіз результатів виконання роботи | 28.04.25—11.05.25 | Виконано |

Студент _____ (підпис) Самусь О.В.

Керівник роботи _____ (підпис) Тарновський М. Г.

(підпис)

# АНОТАЦІЯ

УДК 004.77:004.738.5

Самусь О. В. Онлайн-сервіс фінансового планування. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Комп'ютерна інженерія». Вінниця: ВНТУ, 2025. 76с.

На укр.мові. Бібліогр.: 12 назв, рис.: 22, табл.: 1

В даній бакалаврській роботі було розроблено онлайн-сервіс фінансового планування, який дозволяє користувачам здійснювати базовий моніторинг особистих витрат. У роботі обґрунтовано доцільність створення подібного сервісу в умовах зростаючої фінансової свідомості населення. Проведено аналіз існуючих програмних рішень, визначено слабкі та сильні сторони.

Запропонований сервіс реалізує інтеграцію з банківським API для перегляду категорій витрат користувача, сторінку з актуальними фінансовими новинами, а також розділ криптовалют, який відображає топ-5 цифрових валют за ринковою капіталізацією. Архітектура рішення враховує принципи масштабованості та зручності для кінцевого користувача.

Ключові слова: фінансове планування, Monobank API, фінансові новини, криптовалюта, вебсервіс.

# ABSTRACT

UDC 004.77:004.738.5

Samus O. V. Online Financial Planning Service. Bachelor's Thesis in the specialty 123 "Computer Engineering", educational program "Computer Engineering". Vinnytsia: VNTU, 2025. 76 p.

In English. Bibliography: 12 titles, figures: 22, tables: 1.

This bachelor's thesis presents the development of an online financial planning service that allows users to perform basic monitoring of personal expenses. The work substantiates the relevance of creating such a service in the context of growing financial awareness among the population. An analysis of existing software solutions was conducted, identifying their strengths and weaknesses.

The proposed service implements integration with a banking API for viewing user expense categories, a page with current financial news, and a cryptocurrency section displaying the top 5 digital currencies by market capitalization. The system architecture considers scalability and user-friendliness principles.

Keywords**:** financial planning, Monobank API, financial news, cryptocurrency, web service.

**ЗМІСТ**

# ВСТУП

У сучасному цифровому суспільстві питання ефективного управління особистими фінансами набуває особливої важливості. Зростання фінансової грамотності, поява нових інструментів інвестування та зміни економічної ситуації стимулюють людей до свідомого підходу до власних витрат і заощаджень. Водночас більшість громадян стикаються з труднощами у плануванні бюджету, розподілі доходів та витрат, а також у прийнятті рішень щодо накопичень і фінансових цілей. Традиційні способи фінансового обліку – паперові записи, таблиці чи стандартні мобільні додатки – не завжди враховують індивідуальні потреби користувачів і не забезпечують гнучкості для комплексного аналізу.

**Актуальність** теми зумовлена потребою у створенні адаптованого до українських реалій інструменту, що дозволить користувачам зручно планувати витрати, ставити фінансові цілі, отримувати рекомендації на основі реальних даних і забезпечить безпечну інтеграцію з банківськими сервісами. Зокрема, особливої уваги заслуговує можливість інтеграції з API українських банків, таких як Monobank, для отримання актуальної інформації про транзакції.

Проблематика ефективного фінансового планування особливо актуальна для молоді, яка тільки формує свої фінансові звички, та для людей, що не мають економічної освіти, але прагнуть контролювати свій бюджет. У відповідь на ці виклики активно розробляються онлайн-платформи, що пропонують автоматизовані підходи до фінансового планування. Серед таких рішень можна згадати додатки Mint, YNAB, Spendee, однак більшість із них орієнтовані на іноземні ринки та не враховують особливості української банківської системи, рівня доходів і місцевої ментальності.

**Метою комплексної бакалаврської кваліфікаційної роботи** є створення онлайн-сервісу фінансового планування з інтеграцією до банківського API та можливістю формування фінансової аналітики у режимі реального часу.

Для досягнення поставленої мети у роботі розв’язуються такі **задачі**:

— провдено аналіз сучасних технологій з інформаційної підтримки управління фінансами;

— проаналізовані існуючі рішення у сфері фінансового планування;

— вибрано архітектуру та стек технологій для створення онлайн-сервісу;

— розроблено функціональні модулі для обліку доходів, витрат, побудови бюджету та візуалізації фінансових показників;

— реалізувати захищену інтеграцію з банківським API для автоматичного отримання транзакцій;

**Об’єктом дослідження** є процеси обробки та аналізу фінансової інформації в онлайн-середовищі.

**Предметом дослідження** є програмна архітектура та технології реалізації веб-сервісу для персонального фінансового планування.

# 1 АНАЛІЗ ШЛЯХІВ СТВОРЕННЯ ОНЛАЙН-СЕРВІСУ ФІНАНСОВОГО ПЛАНУВАННЯ

Активне впровадження інформаційних технологій у всі сфери суспільного життя зумовлює зростаючу потребу в ефективних інструментах обробки, подання та візуалізації даних. Одним із ключових напрямів розвитку цифрових рішень є застосування веб-технологій, що базуються на інфраструктурі глобальної мережі Інтернет. У цьому контексті особливого значення набуває створення онлайн-сервісів, здатних забезпечити користувачам зручний доступ до аналітичних та планувальних інструментів у режимі реального часу.

Актуальність проєкту обумовлена необхідністю розробки веб-орієнтованої інформаційної системи для фінансового планування, яка дозволяє користувачам ефективно управляти власними коштами, формувати бюджет, моделювати сценарії витрат і накопичень, а також відслідковувати прогрес у досягненні фінансових цілей. Реалізація такого програмного рішення є головною метою даного бакалаврського дипломного проєкту.

У межах цього розділу буде проведено аналіз основних підходів до побудови подібних веб-сервісів, визначено доцільні методи розробки та обґрунтовано вибір технологічного стеку для створення повнофункціонального онлайн-сервісу фінансового планування.

## 1.1 Роль фінансового планування та сучасних інформаційних технологій у забезпеченні ефективного управління фінансами

У сучасних умовах функціонування підприємств, що характеризуються невизначеністю і високою динамічністю зовнішнього середовища, виникає об’єктивна необхідність застосування ефективного фінансового планування та прогнозування діяльності підприємств. Без використання цих інструментів у діяльності підприємства важко забезпечити стійкий фінансовий стан у майбутньому, сформувати достатній обсяг фінансових ресурсів на розвиток за мінімальних витрат, а отже, досягти конкурентоспроможності на ринку. За

допомогою фінансового планування та прогнозування конкретизуються намічені прогнози, визначаються взаємопов’язані завдання і послідовність їх реалізації у досягненні обраної мети. Важливість і актуальність усіх зазначених питань зумовили значний інтерес і увагу вчених до вивчення фінансового планування та прогнозування діяльності підприємства [1].

Фінансове планування виступає основою для прийняття стратегічних і тактичних управлінських рішень, спрямованих на оптимальне використання фінансових ресурсів, мінімізацію ризиків та забезпечення фінансової стабільності. Воно дає змогу оцінити майбутні доходи та витрати, забезпечити баланс між наявними і необхідними фінансовими ресурсами, а також встановити раціональні фінансові пріоритети. Саме тому фінансове планування вважається ключовим елементом загальної системи управління підприємством [2].

Функція планування в системі управління підприємством є однією з головних функцій, що визначає кінцеві результати виробничозбутової, економічної, фінансової й інвестиційної діяльності. У процесі планування визначаються основні напрямки розвитку підприємства. Планування забезпечує підприємству основу для прийняття управлінських рішень, знижує ризик та сприяє пошуку найбільш придатних напрямів дій.

Виділяють наступні основні завдання фінансового планування на підприємстві, що зображені на рисунку 1.1

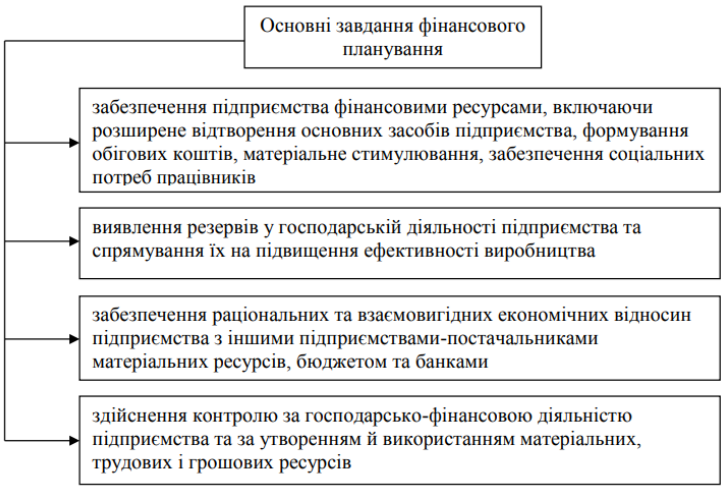


Рисунок 1.1 — Основні завдання фінансового планування

Актуальність теми фінансового планування також зростає в умовах цифрової трансформації економіки. Сучасні інформаційні та комп’ютерні технології суттєво розширюють можливості фінансових аналітиків, забезпечуючи високу швидкість обробки великих обсягів даних, точність розрахунків та автоматизацію рутинних операцій. Інтеграція штучного інтелекту, машинного навчання та хмарних сервісів у фінансове планування дозволяє підприємствам оперативно реагувати на зміни ринку, моделювати різні сценарії розвитку, здійснювати гнучке бюджетування та прогнозування [3].

Крім того, впровадження ІТ-рішень у сфері фінансового планування сприяє підвищенню прозорості та обґрунтованості фінансових рішень. Завдяки цьому керівництво підприємства отримує доступ до актуальної інформації в режимі реального часу, що значно покращує якість управління та знижує ймовірність помилкових рішень. У контексті діджиталізації бізнес-процесів це дозволяє досягати високої ефективності навіть у складних і конкурентних ринкових умовах [4].

Таким чином, поєднання фінансового планування з сучасними інформаційними технологіями є важливим напрямом підвищення загальної ефективності управління підприємством. Це обумовлює доцільність глибокого вивчення як теоретичних засад, так і прикладних аспектів розробки відповідних інформаційних систем, що будуть адаптовані до потреб конкретного підприємства.

## 1.2 Підходи та технології для фінансового планування

Моделювання, включаючи аналіз за сценарієм «що, якщо», обробка інформаційних потоків в обох напрямках (знизу вверх і зверху вниз), а також інтеграція даних, перегляд інформації та створення звітів і графічних представлень є неможливими без застосування спеціалізованих програмних засобів.

Оскільки процес розробки фінансових планів і бюджетів є циклічним, для досягнення їх остаточної версії необхідно кілька разів коригувати і уточнювати

дані. Багато часу витрачається на узгодження операцій, збір необхідної інформації та підготовку звітів. Крім того, контроль і аналіз виконання бюджетів у комерційних організаціях передбачають обробку великих обсягів даних, що робить використання інформаційних технологій необхідним.

Інформаційні технології — поняття більш широке, ніж комп'ютерні продукти. Інформаційні технології – це сукупність програмного забезпечення (інформаційна складова), технічних засобів і телекомунікацій (інструментальна складова), кадрів та їх організація (соціальна складова).

Інформаційні технології (ІТ) охоплюють комплекс засобів і методів, що забезпечують обробку, зберігання, передачу та захист інформації. Вони складаються з трьох основних складових:

— програмне забезпечення включає операційні системи, прикладні програми, бази даних та інші програмні продукти, що забезпечують функціонування інформаційних систем;

— технічні засоби і телекомунікації охоплюють комп'ютери, сервери, мережі, канали зв'язку та інші апаратні засоби, необхідні для обробки та передачі інформації;

— кадри та їх організація включають фахівців з інформаційних технологій, їхню кваліфікацію, організаційну структуру та управлінські процеси, що забезпечують ефективне використання ІТ.

Ці складові взаємодіють між собою, створюючи єдину інформаційну інфраструктуру, що підтримує функціонування організацій та суспільства в цілому.

Сучасні підходи та технології фінансового планування базуються на інтеграції традиційних методів з новітніми інформаційними технологіями. Основні підходи:

— системний підхід передбачає комплексний аналіз усіх факторів, що впливають на фінансову ситуацію підприємства, з урахуванням внутрішніх та зовнішніх умов;

— стратегічне планування орієнтоване на довгострокові цілі підприємства, визначення напрямків розвитку та ресурсного забезпечення;

— оперативне планування зосереджене на короткострокових цілях, бюджетуванні та контролі за виконанням фінансових планів;

— цифровізація фінансового планування: використання сучасних інформаційних технологій для автоматизації процесів планування, прогнозування та аналізу фінансових даних;

Розглянемо ці ключові підходи та технології, які лежать в основі сучасного фінансового планування, їх роль у досягненні бізнес-цілей та забезпеченні ефективності фінансових процесів.

Системний підхід в фінансовому плануванні є основою для глибокого аналізу усіх внутрішніх і зовнішніх факторів, що впливають на фінансову ситуацію підприємства. Це означає, що кожен елемент фінансової системи — від ресурсів до ризиків — оцінюється в контексті взаємозв'язків і впливів на загальну фінансову стратегію. Важливим аспектом є комплексний підхід до аналізу не лише фінансових показників, а й макроекономічних умов, ринкових тенденцій, політичної ситуації, а також технологічних і соціальних змін. Такий підхід дозволяє забезпечити стійкість і адаптивність підприємства до змінних умов на ринку.

Стратегічне планування орієнтоване на визначення довгострокових цілей підприємства. Воно охоплює розробку планів, які дозволяють організації досягти бажаних результатів, забезпечуючи зростання та розвиток на багато років вперед. В цьому контексті підприємства визначають напрямки свого розвитку, враховуючи такі аспекти, як розширення ринків, інвестиції в інновації, модернізацію інфраструктури та ефективне використання ресурсів. Стратегічне планування базується на ретельному аналізі можливостей і загроз, прогнозуванні фінансових показників, а також на розробці моделей росту та розвитку, що узгоджуються з реальними фінансовими умовами.

Оперативне планування зосереджується на короткострокових фінансових цілях та управлінні бюджетами. Цей підхід охоплює управління повсякденними

фінансовими операціями, включаючи розподіл ресурсів, контроль за витратами, оптимізацію фінансових потоків і забезпечення фінансової стабільності на поточний період. Оперативне планування забезпечує ефективну реалізацію стратегічних цілей через точне виконання фінансових завдань на кожному етапі та коригування бюджету відповідно до змін на ринку. Важливим аспектом є постійний моніторинг та контроль за виконанням фінансових планів.

Цифровізація фінансового планування має на меті автоматизацію та оптимізацію процесів планування, прогнозування та аналізу фінансових даних через використання сучасних інформаційних технологій. Це дозволяє значно підвищити точність прогнозів, знизити витрати на управління фінансами та покращити реакцію на зміни в бізнес-середовищі. Використання спеціалізованих програмних платформ та інструментів для обробки великих обсягів даних дозволяє фінансовим менеджерам приймати обґрунтовані рішення на основі реальних даних у реальному часі.

Для фінансового аналізу використовуються інструменти бізнес-аналітики (BI), які дозволяють візуалізувати та аналізувати фінансові дані у зручному вигляді.

Power BI — це програмне забезпечення для візуалізації даних, яке допомагає створювати інтерактивні звіти та графіки на основі фінансової інформації. Завдяки інтеграції з іншими системами Power BI надає глибоке розуміння фінансових показників у реальному часі.

Tableau ще один популярний інструмент для аналізу та візуалізації даних, який дозволяє інтегрувати різноманітні джерела фінансової інформації та створювати динамічні звіти та панелі управління.

Сучасне фінансове планування все частіше враховує використання штучного інтелекту (ШІ) та машинного навчання (МН) для прогнозування і оптимізації фінансових стратегій. Особливо важливим є інтегрування криптовалют у фінансові системи, оскільки вони здатні значно змінити спосіб, яким інвестори і компанії управляють своїми активами. Криптовалюти, зокрема біткоїн, ефіріум та інші альткоїни, стали популярним інструментом для

інвестицій та зберігання вартості, завдяки своїй децентралізованій природі та потенціалу для високої прибутковості.

Прогнозування цін на криптовалюти: Моделі машинного навчання, зокрема нейронні мережі, можуть аналізувати історичні дані щодо цін на криптовалюти, а також інші фактори, які впливають на їх вартість, наприклад, новини, соціальні медіа та загальний стан ринку. За допомогою таких моделей можна спрогнозувати ціни на криптовалюти в короткостроковій та довгостроковій перспективі, що допомагає інвесторам приймати більш обґрунтовані рішення.

Автоматична торгівля (робо-торгівля): Штучний інтелект і машинне навчання активно використовуються для розробки алгоритмів автоматичної торгівлі на криптовалютних біржах. Ці алгоритми аналізують ринкові сигнали і проводять операції на основі заданих критеріїв без участі людини, дозволяючи отримувати вигоду від короткострокових коливань ринку. Багато криптовалютних трейдерів використовують такі системи для оптимізації своїх торгівельних стратегій, що значно знижує людську помилку і підвищує ефективність.

Аналіз настроїв: Використання ШІ для аналізу настроїв на ринку криптовалют стало важливим інструментом для прогнозування можливих рухів цін. Збираючи та аналізуючи дані з соціальних мереж, новинних сайтів і форумів, моделі машинного навчання можуть оцінювати загальний емоційний настрій серед інвесторів і учасників ринку. Ці дані можуть бути використані для прогнозування майбутніх коливань цін на криптовалюти, оскільки настрої часто відіграють важливу роль у поведінці ринку.

Ідентифікація шахрайства та аномалій: Штучний інтелект активно застосовується для виявлення аномальних фінансових операцій і запобігання шахрайству на криптовалютних платформах. Алгоритми машинного навчання можуть аналізувати транзакції, шукати незвичні або підозрілі патерни поведінки, що дозволяє вчасно виявляти шахрайські операції і мінімізувати фінансові

втрати. Це особливо важливо в умовах відсутності централізованого контролю, що є характерним для криптовалютних мереж.

### 1.3 Аналіз існуючих рішень у сфері фінансового планування

У сучасних умовах стрімкого розвитку цифрових технологій та зростання попиту на ефективне управління особистими фінансами фінансове планування набуває особливої актуальності. Зростання кількості фінансових інструментів, збільшення обсягів даних і потреба в прийнятті обґрунтованих рішень сприяли появі великої кількості рішень у сфері фінансового планування — від мобільних застосунків до комплексних веб-сервісів. Цей розділ присвячено аналізу найпоширеніших та найефективніших сучасних рішень, які дозволяють користувачам контролювати витрати, формувати бюджети, інвестувати та досягати фінансових цілей. Оцінка існуючих продуктів дозволить виявити їх сильні сторони, недоліки та напрями для потенційного вдосконалення, що є необхідним етапом при розробці власної системи фінансового планування.

У межах цього підрозділу розглянемо найбільш популярні аналоги програм у сфері фінансового планування, їх функціональні можливості та підходи до реалізації.

Рисунок 1.2 — Застосунок Saldo Finance

Saldo Finance (рис. 1.2) — застосунок для обліку грошей для бізнесу та контролю особистих і сімейних фінансів. Він дає можливість отримати транзакції з рахунків monobank, додавати свої витрати та доходи з інших джерел вручну. У результаті користувач може побачити свої сумарні витрати, доходи та різницю між ними щомісяця. Додаток дає можливість запрошувати інших

користувачів для ведення спільного бюджету, а також працювати з P&L та Cashflow (касовий метод та метод нарахування).

Fast Budget (рис. 1.3) — дає можливість будувати різні типи графіків і таблиць та вести фінансовий календар. Користувач може синхронізувати дані між пристроями та підключити свої банківські рахунки. Із додаткових функцій: автоматичне завантаження операцій із банківського рахунку, можливість створення власних категорій з функцією їх редагування та переміщення.

У застосунку можна вводити свої щоденні доходи та витрати або планувати їх повторення, створити шаблони транзакцій, налаштувати нагадування про необхідність здійснення транзакції, тримати на контролі свої фінанси за допомогою п’яти різних типів діаграм (рис.1.3).

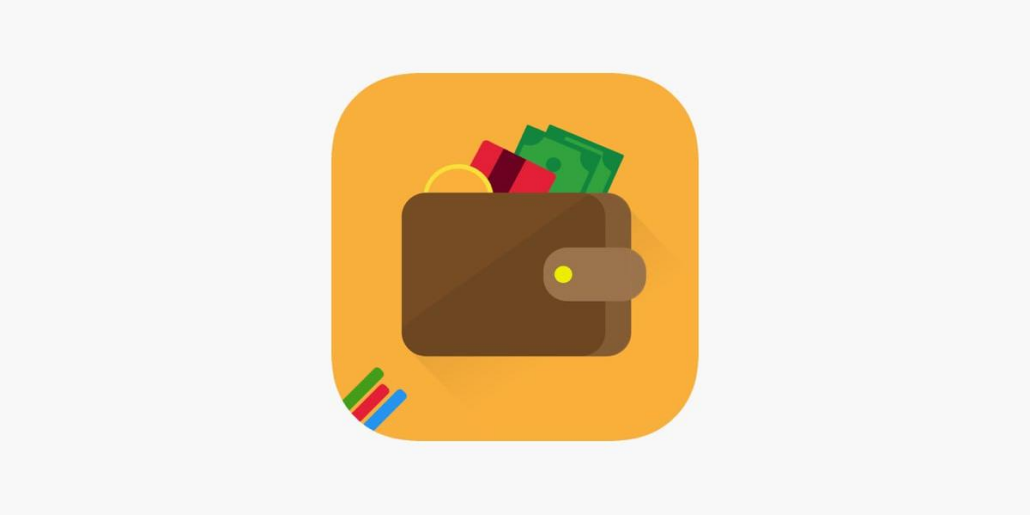

Рисунок 1.3 — Застосунок Fast Budget

Money Lover (рис. 1.4) — менеджер рахунків — мобільний додаток для обліку доходів та видатків, прогнозування бюджету, ведення статистики в подорожах. Програма підтримується на всіх смартфонах та планшетах на базі операційних систем Android та iOS. Вона дає змогу визначити бюджет за встановленими або самостійно створеними категоріями, поставити фінансову мету, а також відстежити витрати під час подорожі.

Усі операції у програмі групуються по гаманцях. Наприклад, можна переглянути інформацію про власне операції, борги, тенденції та категорії. Для підрахунку фінансів передбачені конвертер валют, СМС-банкінг, калькулятор

чайових, процентна ставка та інше. Експортувати статистику можна у форматах CSV та XLS. Крім того, є функції завантаження на сервер і створення резервної копії даних.

Рисунок 1.4 — Застосунок Money Lover

Money Manager (рис. 1.5) — програма для фінансового планування, огляду, відстеження витрат і управління особистими активами для Android. Вона надає щотижневий і помісячний звіт фінансових операцій, пропонує можливість зберігати фото, переглядати транзакції за певними фільтрами, планувати за допомогою календаря, створювати графіки активів та встановлювати місячний бюджет для кожної категорії [5].

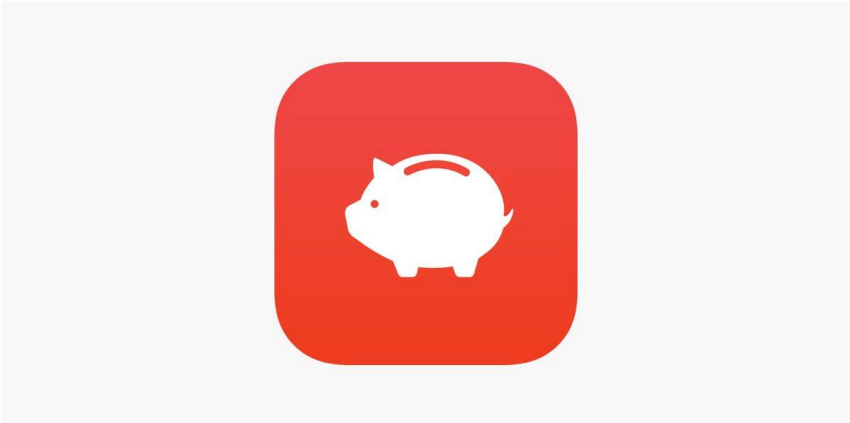

Рисунок 1.5 — Застосунок Money Manager

Підсумовуючи проведений аналіз, можна зробити висновок, що на сучасному ринку існує велика кількість програмних рішень, орієнтованих на фінансове планування, контроль витрат, формування бюджету та досягнення особистих фінансових цілей. Більшість з них мають схожий функціонал, який

включає базові інструменти для обліку доходів і витрат, побудову категорій витрат, візуалізацію фінансових потоків, а також формування звітів. Частина сервісів інтегрується з банківськими рахунками або пропонує спрощені інструменти для створення заощаджень та контролю над борговими зобов’язаннями. При цьому, хоча інтерфейси таких програм зазвичай зручні для користувача, а функціонал охоплює основні потреби в побутовому фінансовому менеджменті, вони значною мірою обмежуються стандартними рішеннями, які майже не відрізняються одне від одного.

Проте варто відзначити, що більшість із розглянутих застосунків не враховують виклики та можливості сучасного етапу цифровізації. Зокрема, практично повністю відсутня інтеграція штучного інтелекту, яка могла б забезпечити персоналізовані фінансові поради, прогнозування витрат на основі поведінкової аналітики або автоматичну адаптацію бюджету відповідно до змін у фінансовій ситуації користувача. Також майже не представлений аналіз криптовалютного портфеля, що є особливо актуальним у контексті зростання популярності цифрових активів серед молодого покоління користувачів. Більше того, сучасні інструменти рідко враховують складні фінансові сценарії, пов’язані з інвестиціями, пасивним доходом, ризик-менеджментом або податковим плануванням, не кажучи вже про крос-платформеність, API-доступи до сторонніх сервісів чи модулі прогнозування на основі великих даних.

Таким чином, можна стверджувати, що попри велику кількість наявних програм, ринок усе ще має значний потенціал для інновацій. Існує очевидна потреба у створенні більш гнучких, інтелектуальних і технологічно просунутих рішень, які б поєднували сучасні цифрові можливості з високим рівнем персоналізації, безпеки та аналітичної потужності. Саме у цьому напрямі варто орієнтуватися при розробці нового програмного продукту у сфері фінансового планування, здатного задовольнити вимоги сьогодення та адаптуватися до фінансової поведінки майбутнього.

# 2 ІНФОРМАЦІЙНА ВЗАЄМОДІЯ В ОНЛАЙН-СЕРВІСІ ФІНАНСОВОГО ПЛАНУВАННЯ

## 2.1 Загальна архітектура даних у сервісі фінансового планування

Архітектура обміну даними у сучасному онлайн-сервісі фінансового планування є багаторівневою системою, яка забезпечує надійний, безпечний та швидкий обмін інформацією між користувачем, базами даних, банківськими API та аналітичними модулями. Вона охоплює фронтенд-інтерфейс, бекенд-логіку, базу даних, а також модулі зовнішньої інтеграції. Кожен із компонентів архітектури виконує специфічну роль і взаємодіє з іншими за допомогою чітко визначених протоколів та структур даних.

Прикладом архітектури обміну даними у сучасному онлайн-сервісі фінансового планування, може слугувати блок-схема відображена в додатку В.

У центрі системи знаходиться бекенд-сервер, реалізований за допомогою мови програмування Python з використанням Django або FastAPI. Це дозволяє ефективно обробляти запити, виконувати бізнес-логіку, взаємодіяти з базами даних та забезпечувати безпеку обробки конфіденційної фінансової інформації. Python є ідеальним вибором для таких задач завдяки великій кількості бібліотек для роботи з фінансами, машинним навчанням та аналітикою [6].

Python — високорівнева мова програмування, яка вирізняється простим синтаксисом і широкими можливостями для створення серверних застосунків. Вона підтримує як синхронну, так і асинхронну модель обробки запитів, що дає змогу ефективно працювати з великими обсягами даних у фінансових системах.

Django — це фреймворк на Python, який надає повний набір інструментів для веб-розробки [7]: маршрутизацію, шаблони, ORM, аутентифікацію, безпеку тощо. Він дозволяє швидко створювати MVP і масштабувати продукт у майбутньому без втрати продуктивності.

Фронтенд реалізується з використанням HTML, CSS та JavaScript, що забезпечує зручне, адаптивне й естетично привабливе середовище для користувача. Завдяки використанню JavaScript-фреймворків, таких як Vue.js,

досягається миттєве оновлення інтерфейсу в реальному часі без перезавантаження сторінки, що є критично важливим для фінансових застосунків.

HTML визначає структуру вебсторінки та формує логічне дерево елементів: форми, кнопки, поля введення, текстові блоки. CSS забезпечує стилізацію інтерфейсу — кольори, шрифти, відступи, адаптивність тощо.

JavaScript дозволяє реалізувати інтерактивність, реакцію на дії користувача, динамічну зміну даних, надсилання запитів без оновлення сторінки.

Vue.js базується на компонентній архітектурі: кожен елемент, як-от таблиця витрат чи графік, є окремим модулем з власною логікою. Фреймворк забезпечує двобічне зв’язування даних та автоматичне оновлення інтерфейсу при зміні стану.

Для зберігання даних використовується реляційна база даних PostgreSQL. Вона забезпечує високу продуктивність, надійність і масштабованість, що робить її оптимальним вибором для фінансових застосунків, де важливі точність і транзакційна цілісність. Django ORM або SQLAlchemy забезпечують ефективний зв’язок між об’єктами Python і таблицями бази даних.

PostgreSQL підтримує складні транзакції, агрегації, індексацію, [8] збережені процедури. Вона здатна обробляти великі обсяги фінансових даних і виконувати складні SQL-запити. Django ORM дозволяє описувати структуру таблиць у вигляді Python-класів, автоматично генеруючи необхідні SQL-запити. SQLAlchemy надає додаткову гнучкість у роботі з базами даних, особливо корисну при реалізації складних умов і приєднань.

Інтеграція з банками, платіжними системами або біржами здійснюється через REST API [9]. У цьому проєкті реалізовано взаємодію з Monobank API, що дозволяє в реальному часі отримувати інформацію про транзакції, баланс та витрати користувача. Зовнішні запити до API виконуються асинхронно з використанням aiohttp або httpx, що дозволяє обробляти запити швидше й без навантаження на основний потік.

Monobank API дає доступ до даних клієнта — балансу, історії витрат, категорій покупок. Це дозволяє створювати актуальні дашборди та аналітичні модулі. aiohttp і httpx дають змогу виконувати асинхронні HTTP-запити, паралельно звертаючись до різних джерел даних, що особливо важливо при великій кількості одночасних запитів.

Окрему роль у системі відіграє модуль аналітики. Він обробляє великі обсяги фінансових даних, аналізує поведінку користувача, формує персоналізовані поради. Для реалізації використовуються інструменти Python: pandas, scikit-learn та TensorFlow.

Pandas дозволяє зручно обробляти структуровані табличні дані: фільтрувати, агрегувати, будувати графіки. scikit-learn надає алгоритми для машинного навчання — класифікації, кластеризації, прогнозування. TensorFlow використовується для створення глибоких нейронних мереж, які можуть аналізувати тренди витрат, прогнозувати фінансову поведінку.

Уся передача даних між клієнтом і сервером здійснюється по захищених каналах (HTTPS, SSL/TLS), що запобігає перехопленню інформації. Реалізовано сучасні механізми авторизації (OAuth 2.0, JWT), що забезпечують доступ лише для авторизованих користувачів.

HTTPS та SSL/TLS шифрують трафік, гарантуючи конфіденційність переданих даних. OAuth 2.0 дозволяє авторизувати користувача без передачі пароля — через доступні токени. JWT — це токени доступу, що передаються між клієнтом і сервером. Вони шифруються та мають обмежений строк дії, що підвищує безпеку.

Підсумовуючи, архітектура побудована на сучасному технологічному стеку: HTML, CSS, JavaScript і Vue.js — для клієнтської частини; Python, Django і Django ORM — для серверної логіки; PostgreSQL — для надійного зберігання фінансових даних; Monobank API — для інтеграції з банківськими сервісами. Ця багаторівнева система є масштабованою, безпечною та повністю адаптованою до потреб сучасних фінансових сервісів.

## 2.2 Огляд технологій інтеграції з банківськими та фінансовими API

У сучасному цифровому середовищі інтеграція з банківськими та фінансовими API стала ключовим елементом розвитку фінансових технологій. Ці API дозволяють стороннім додаткам взаємодіяти з банківськими системами, отримувати доступ до фінансових даних користувачів, ініціювати платежі та формувати нові сервіси на базі відкритих фінансових інфраструктур. Використання API надає можливість створювати персоналізовані фінансові продукти, покращувати користувацький досвід, а також автоматизувати обробку транзакцій і ведення обліку.

На прикладі Monobank API можна побачити, як український банк відкрив інтерфейс для доступу до даних клієнта. Через API розробники отримують можливість зчитувати інформацію про рахунок, історію транзакцій, поточний баланс, а також налаштовувати вебхуки для отримання сповіщень у реальному часі про зміну стану рахунку. Така відкритість дозволяє інтегрувати банківські сервіси до мобільних застосунків, бухгалтерських систем або чат-ботів, які, у свою чергу, можуть надавати додаткову цінність кінцевому користувачеві [9].

Іншим прикладом успішної реалізації є платформа Plaid, яка забезпечує стандартизований доступ до банківських даних у США, Канаді та Європі. Через API Plaid можна отримати інформацію про рахунки користувачів, їхні транзакції, аналітику витрат, кредити, а також здійснювати фінансову ідентифікацію. Компанії, що використовують Plaid, можуть об'єднувати дані з кількох банків, зводити фінансові звіти, створювати динамічні фінансові плани, що відкриває нові можливості для фінансового консалтингу.

Інтеграція з API стає особливо актуальною в умовах поширення концепції open banking, що активно просувається в Європейському Союзі на основі директиви PSD2. Згідно з цим регламентом, банки зобов'язані надавати доступ до рахунків користувача третім сторонам — так званим TPP (Third Party Providers) — за згодою клієнта. Це стимулює конкуренцію на ринку фінансових послуг і створює нові ніші для фінтех-стартапів. Наприклад, компанії можуть створювати

сервіси для порівняння банківських тарифів, автоматизації сплати рахунків, обліку витрат чи кредитного скорингу.

Технологічно API зазвичай базуються на REST-архітектурі з використанням HTTP-протоколу. Запити й відповіді формуються у форматі JSON, що робить інтеграцію швидкою й доступною для більшості мов програмування. Для забезпечення безпеки обміну даними використовуються сучасні протоколи шифрування, такі як TLS 1.3, а також стандарти авторизації — насамперед OAuth 2.0. Цей протокол дозволяє отримувати доступ до захищених ресурсів без необхідності зберігання паролів, через видачу тимчасових токенів доступу, які мають чітко визначений обсяг прав (scopes) і термін дії.

JSON — це формат обміну даними, що підтримує вкладену структуру і легко читається людиною. Він має вирішальне значення для сучасної інтеграції банківських сервісів, оскільки дозволяє точно структурувати інформацію про транзакції, рахунки, валюту, користувачів тощо. JSON є мовонезалежним і активно підтримується практично всіма мовами розробки, включно з Python, Java, C# тощо [11].

Серед інших методів забезпечення безпеки варто згадати підписування запитів, використання JWT-токенів, IP-фільтрацію та контроль частоти запитів (rate limiting). Банки, як правило, реалізують sandbox-середовище — тестову інфраструктуру, де розробники можуть перевірити інтеграцію без ризику порушити роботу реального рахунку чи надіслати небажану транзакцію. Це дозволяє покращити якість коду, провести валідацію формату запитів та перевірити обробку помилок без шкоди для користувача.

JWT-токен призначений для передачі підписаних «заявок» (claims) між службами (як зовнішніми, так і внутрішніми для застосунку/сайту). «Заявки» — частина інформації, яку інші можуть переглядати та/або перевіряти, але не змінювати. Щоб ідентифікувати/здійснити автентифікацію користувача у вашому (веб/мобільному) застосунку, необхідно розмістити стандартизований токен у заголовок або url сторінки (або кінцеву точку API). Таким чином можна

засвідчити, що користувач авторизувався та може переглядати бажаний контент [12].

На українському ринку спостерігається активне впровадження фінансових API не лише банками, а й іншими фінансовими інститутами. Наприклад, у 2021 році компанія Fintech Band, яка стоїть за Monobank, запровадила нову версію API з можливістю доступу до рахунків ФОПів. Це дозволяє бухгалтерським сервісам, таким як BookKeeper чи Finmap, автоматично зчитувати надходження, формувати звіти та спрощувати податкову звітність. Таким чином, інтеграція з API трансформує класичну банківську інфраструктуру в модульну цифрову платформу, здатну адаптуватися до потреб сучасного бізнесу.

На міжнародному рівні великі банки також активно відкривають власні API-платформи. Наприклад, HSBC, Barclays і Deutsche Bank мають окремі портали для розробників, де публікують документацію, приклади коду та ключі доступу до sandbox-оточення. Деякі з них пропонують SDK та готові бібліотеки для Java, Python чи Node.js, що суттєво спрощує розробку. Водночас, із відкриттям інтерфейсів постає питання про регуляторну відповідальність, тому доступ до API суворо контролюється через механізми сертифікації, логування та ревізії.

Ще одним напрямком використання фінансових API є побудова мультибанкінгових платформ. Такі сервіси дозволяють об'єднувати інформацію з кількох банків в одному інтерфейсі. В Україні прикладом може слугувати сервіс Neobank від Concord Bank або «Опенбанкінг хаб» від компанії Corezoid, які дозволяють здійснювати контроль витрат, перекази між рахунками та навіть ініціювати платежі через API. Це дає змогу спростити роботу як для фізичних осіб, так і для малого бізнесу, що використовує кілька рахунків у різних банках.

Таким чином, фінансові API виступають основою нового цифрового підходу до управління фінансами. Вони сприяють відкритості ринку, підвищують прозорість транзакцій, зменшують операційні витрати та створюють додану вартість для кінцевого користувача. У майбутньому очікується подальший розвиток API у напрямку стандартизації, підвищення безпеки та

розширення функціональності, зокрема через використання штучного інтелекту, машинного навчання та смарт-контрактів.

## 2.3 Порівняння протоколів та способів передавання фінансових даних

Передача фінансових даних вимагає використання протоколів, що гарантують безпеку, ефективність, швидкість та відповідність вимогам фінансових установ. Розглянемо основні протоколи, що використовуються для передачі фінансової інформації, з аналізом їх переваг, недоліків і специфічних сфер застосування.

HTTP (Hypertext Transfer Protocol) є основним протоколом для передачі даних через Інтернет, а HTTPS — його захищена версія, яка забезпечує шифрування за допомогою SSL/TLS. REST (Representational State Transfer) є стилем архітектури, що дозволяє розробникам будувати веб-сервіси через HTTP-запити. REST API використовує стандартні HTTP методи: GET, POST, PUT, DELETE для взаємодії з даними.

REST API здійснює запити через стандартні HTTP методи. Для фінансових операцій використовуються:

— GET для отримання інформації, наприклад, балансу або історії транзакцій;

— POST для надсилання нових даних, таких як перекази або створення рахунків;

— PUT/PATCH для оновлення існуючих даних;

— DELETE для видалення даних, наприклад, закриття рахунку.

Безпека забезпечується через HTTPS, з авторизацією через OAuth 2.0, а для аутентифікації використовуються JWT (JSON Web Tokens) або API-ключі.

REST API має низку переваг, зокрема легкість впровадження та підтримки, масову підтримку в багатьох мовах програмування, простоту інтеграції з іншими вебсервісами та велику документацію та підтримку спільноти. Однак він має й недоліки, зокрема, не підтримує двосторонній зв'язок і є менш ефективним для

високочастотних операцій, таких як торгівля криптовалютами або валютними парами. Крім того, можуть виникати затримки при великих навантаженнях.

WebSockets є протоколом для двостороннього обміну даними через постійне з'єднання, яке залишається відкритим між клієнтом та сервером. Це дозволяє здійснювати миттєвий обмін даними в реальному часі.

Після ініціалізації з'єднання через URL WebSocket, клієнт і сервер можуть обмінюватися даними в реальному часі без необхідності відновлювати з'єднання для кожного нового запиту.

WebSockets мають кілька переваг, зокрема забезпечення двостороннього зв'язку в реальному часі, що є особливо важливим для обміну цінами криптовалют або акціями. Вони також підходять для високочастотних операцій, де необхідна мінімальна затримка при передачі даних. Однак їх використання може бути складним в налаштуванні та підтримці, і вони не підходять для простих додатків, де немає потреби в постійному обміні даними. Вимога стабільного з'єднання може бути проблемою в умовах непостійних мереж.

SOAP — це протокол, заснований на XML, для обміну структурованими повідомленнями через Інтернет. SOAP активно використовується в корпоративних фінансових системах завдяки своїй стандартизації та здатності підтримувати складні транзакції та високу безпеку.

SOAP передає дані у форматі XML через HTTP або SMTP і забезпечує додаткову безпеку через WS-Security для підписування та шифрування повідомлень.

Переваги SOAP включають високу безпеку через WS-Security, підтримку цілісності транзакцій і надійності, а також стандартизований процес для обміну даними, що гарантує сумісність між різними системами. Проте SOAP є складнішим у порівнянні з REST API, має великі об'єми повідомлень, що можуть уповільнити обробку даних, і потребує підтримки XML, що може бути менш ефективним для деяких сучасних застосувань.

ISO 20022 є міжнародним стандартом для обміну фінансовими повідомленнями, який забезпечує єдину структуру для фінансових транзакцій і підтримує передачу даних у форматах XML або JSON.

Стандарт визначає детальну структуру повідомлень для всіх типів фінансових операцій, включаючи платіжні перекази, кредити, дебети, акредитиви тощо.

ISO 20022 має кілька переваг, зокрема, він є міжнародним стандартом, що забезпечує сумісність між різними фінансовими установами, а також підтримує високу деталізацію і забезпечує чіткість в обміні фінансовими даними. Крім того, він широко використовується в міжбанківських платіжних системах, таких як SWIFT та SEPA. Однак його впровадження може бути складним для невеликих або нових установ, а великі розміри повідомлень можуть сповільнити процеси.

MQ (черги повідомлень) забезпечує асинхронний обмін даними між різними сервісами або системами. Протоколи, такі як AMQP, Kafka, MQTT, забезпечують доставку повідомлень навіть у разі відключення деяких компонентів системи.

Дані передаються через чергу, і кожен компонент, який споживає повідомлення, може працювати асинхронно. Після того, як повідомлення доставлено, система може продовжити свою роботу.

Переваги MQ включають високу продуктивність і масштабованість, що робить його ідеальним для високонавантажених систем. Крім того, він забезпечує надійність доставки повідомлень. Однак його використання вимагає спеціалізованих знань і навичок для управління чергами, а також складності в налаштуванні та інтеграції.

REST API найкраще підходить для додатків, де не потрібно високочастотне оновлення даних. Його простота впровадження та підтримки робить його популярним серед фінансових сервісів для надання API доступу до даних. Також з переваг використання REST API можна відзначити масштабованість, незалежність від мови програмування.

WebSockets підходять для систем реального часу, таких як криптовалютні біржі або платформи для торгівлі акціями, де важлива мінімальна затримка при передачі даних.

SOAP краще застосовувати в умовах корпоративних фінансових систем, де необхідні високий рівень безпеки і транзакційної цілісності. ISO 20022 є стандартом для глобальних платіжних мереж, що забезпечує високу деталізацію даних і сумісність між різними фінансовими установами.

MQ ідеально підходить для обробки великих обсягів фінансових даних, наприклад, для систем, що обробляють транзакції в реальному часі або для обміну повідомленнями між різними мікросервісами. Використання брокера повідомлень дозволяє зменшити навантаження на основні компоненти системи, забезпечити надійність передачі даних та покращити масштабованість усієї архітектури.

Завдяки чергам повідомлень можна зменшити навантаження на основні сервіси та ефективно масштабувати систему в умовах зростання кількості користувачів або транзакцій.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН-СЕРВІСУ

У цьому розділі представлено процес програмної реалізації веб-ресурсу, створеного в рамках даного проєкту. Розглянуто ключові функціональні можливості системи, структуру її роботи та логіку взаємодії між окремими компонентами. Окрема увага приділяється обґрунтуванню вибору технологій, мов програмування, фреймворків і бібліотек, що були використані під час розробки.

Також детально описано принципи взаємодії між клієнтською та серверною частинами, а саме: обробку запитів користувача, передачу даних через API, зберігання та обробку інформації на сервері. Наведено приклади реалізації основних елементів користувацького інтерфейсу, що забезпечують інтуїтивну навігацію, адаптивність і зручність у користуванні веб-ресурсом.

Крім того, проаналізовано рішення, які сприяють підтримці стабільності та масштабованості системи, з урахуванням потенційного розширення функціоналу в майбутньому.

## 3.1 Обґрунтування вибору мов програмування, фреймворків, середовища розробки тощо

У процесі розробки вебзастосунку надзвичайно важливим є раціональний вибір технологічного стека. Коректно підібрана мова програмування, фреймворки та середовище розробки суттєво впливають на ефективність розробки, підтримку коду, масштабованість, безпеку та загальний життєвий цикл програмного продукту. У рамках даного проєкту були обрані наступні основні технології: Python з фреймворком Django для серверної частини, JavaScript як базова мова фронтенду, та Vue.js як сучасний реактивний JavaScript-фреймворк.

Фронтенд-розробка за останнє десятиліття зазнала суттєвих змін. Традиційний підхід з використанням "чистого" HTML, CSS і JavaScript на стороні клієнта витісняється новими архітектурними моделями. Зокрема, сьогодні домінують SPA (Single Page Applications) — односторінкові застосунки,

що не перезавантажують сторінку при переході між розділами, а динамічно оновлюють лише необхідні частини інтерфейсу. Для створення SPA необхідні потужні інструменти, які дозволяють ефективно працювати з компонентами інтерфейсу, синхронізувати дані, керувати станом, обробляти маршрути і організовувати зручну архітектуру.

Найбільш популярними інструментами для цього стали фреймворки та бібліотеки JavaScript: Vue.js**,** React**,** Angular **та** Svelte. Кожен з них має свою філософію, архітектурні рішення, особливості розробки та екосистему. При виборі інструменту для реалізації бакалаврського проєкту, важливо враховувати складність проєкту, досвід розробника, очікувану масштабованість, швидкість прототипування та документацію.

Vue.js позиціонується як прогресивний фреймворк, який дозволяє поступово розширювати функціональність — від простого шаблону в HTML-файлі до повноцінного SPA. Його головна перевага — збалансованість між простотою використання та можливістю масштабування.

У Vue основною одиницею побудови інтерфейсу є компонент. Компоненти — це самодостатні блоки, які містять шаблон (HTML), логіку (JavaScript/TypeScript) та стилі (CSS/SCSS). Це дозволяє легко масштабувати проєкт, повторно використовувати код, тестувати окремі частини.

Файл компоненту зазвичай має розширення .vue і складається з трьох секцій:

— <template> описує HTML-структуру компонента;

— <script> містить бізнес-логіку;

— <style> визначає стилізацію (локальну або глобальну).

Vue реалізує реактивність на рівні ядра. Будь-яка зміна стану автоматично оновлює DOM, що суттєво зменшує кількість ручного коду. Система реактивності базується на Proxies (в 3-й версії), що дозволяє робити глибоке відстеження змін навіть у вкладених об’єктах.

Іншим ключовим елементом Vue є двостороннє зв’язування даних (v-model), яке дозволяє зв’язувати поля форм з властивостями об’єкта без необхідності писати окремі обробники подій.

Vue підтримує декілька форматів написання компонентів:

— Options API (традиційний) — опис компоненту через об'єкт (data, methods, computed, watch);

— Composition API (сучасний, з 3-ї версії) — використання функцій ref, reactive, computed для гнучкішої організації логіки (особливо з TypeScript);

— Script Setup API — нова форма, що поєднує зручність;

— Окрім ядра, Vue має сильну екосистему:

— Vue Router — бібліотека для організації маршрутизації (вкладені маршрути, lazy loading, переходи між сторінками SPA);

— Pinia — офіційне сховище стану (наступник Vuex), що підтримує модульність, реактивність, інтеграцію з DevTools;

— Vite — сучасний інструмент для збирання, що забезпечує надзвичайно швидкий старт дев-сервера, гаряче перезавантаження (HMR), модульну структуру;

— Nuxt.js — фреймворк на базі Vue для створення SSR (Server Side Rendered) і JAMStack-застосунків.

Vue є повністю open-source і підтримується великою спільнотою, особливо в Азії (зокрема Китаї), де його використовують найбільше.

React був створений у Facebook у 2013 році і спочатку позиціонувався як бібліотека для побудови UI. Його основна особливість — використання JSX — синтаксису, який поєднує HTML і JavaScript. Компоненти в React — це функції або класи, які повертають фрагмент віртуального DOM.

React є вкрай потужним інструментом, проте має вищу криву навчання, оскільки вимагає розуміння:

— роботи з state/props;

— життєвого циклу компонентів;

— підходів до керування станом (наприклад, Redux, Zustand, Recoil);

— специфіки роботи з маршрутизацією (React Router);

— конфігураційних інструментів (Create React App, Next.js, Vite тощо).

React — максимально гнучкий, але при цьому вимагає більше зусиль на підтримку архітектури. Наприклад, в простому SPA потрібно самостійно підключати й налаштовувати багато речей, які у Vue доступні з коробки.

Angular — найпотужніший, але і найскладніший фреймворк серед усіх. Він був повністю переписаний у 2016 році (версія Angular 2+) і став повноцінним фреймворком із власним DI-контейнером, RxJS, CLI, власним синтаксисом шаблонів, модулями, сервісами та системою форм.

Angular ідеально підходить для enterprise-рішень, але для невеликого проєкту (наприклад, бакалаврської роботи) він надмірний, адже велика кількість файлів, конфігурацій, жорстка структура, висока крива навчання. Він вимагає обов'язкового використання TypeScript, знання реактивного програмування (RxJS) та налаштування шаблонів з директивами (*ngIf, *ngFor).

Svelte підходить до створення UI принципово інакше: замість того, щоб інтерпретувати компоненти у браузері (як Vue чи React), він компілює їх у високоефективний JavaScript під час збірки. Це дозволяє створювати дуже легкі та швидкі застосунки без віртуального DOM.

Svelte — це "write less, do more" фреймворк. Зміни у змінних автоматично оновлюють інтерфейс, без потреби у setState, useEffect, ref, computed тощо. Він дозволяє писати менше коду, досягати більшої продуктивності, але його екосистема ще не така зріла, як у Vue чи React, а підтримка TypeScript іноді вимагає додаткових налаштувань.

Vue.js виявився найоптимальнішим варіантом з кількох причин:

— оптимальний баланс між простотою та можливостями: Vue поєднує доступний поріг входу з гнучкістю масштабування проєктів.

— висока швидкість розробки, за рахунок шаблонів, реактивності, наявності офіційних інструментів можна реалізувати проєкт за короткий час.

— зрозуміла документація, адже офіційна документація Vue одна з найкращих серед JS-фреймворків.

— легка архітектурна організація, тому що компоненти, маршрути, стан — усе чітко розділено, логічно організовано.

— сучасна екосистема, використання Vite, Pinia, Vue Router дозволяє реалізовувати SPA з розширеною логікою, підтримкою API, фільтрацією, авторизацією, тощо.

— інтуїтивний синтаксис, адже розробка в Vue більш простіша у реалізації, ніж у React або Angular — багато речей очевидні або автоматизовані.

— гнучкість, адже можна використовувати Vue як окремий віджет у старих системах або будувати повноцінні програми.

— постійна підтримка TypeScript, сучасна розробка з <script setup lang="ts"> дозволяє отримати автодопомогу, рефакторинг і статичну перевірку типів.

Для глибшого розуміння вибору фреймворку для реалізації веб-застосунку було проведено порівняльний аналіз найпопулярніших сучасних інструментів. У таблиці 3.1 наведено основні характеристики за рядом технічних та практичних критеріїв, які є важливими при розробці інтерфейсів користувача. Також враховувались аспекти продуктивності, підтримки спільнотою та можливості інтеграції з іншими сервісами, що забезпечує масштабованість і гнучкість за стосунку. Обраний фреймворк продемонстрував найкращий баланс між зручністю використання, та актуальністю у професійному середовищі. Крім того, значну увагу приділено безпеці, зокрема підтримці сучасних механізмузахисту даних користувачів. Окремо було оцінено якість офіційної документації та наявність активної спільноти, що відіграє важливу роль у швидкому вирішенні технічних питань.

Таблиця 3.1 — Порівняльна таблиця frontend фреймворків

| Критерій | Vue.js | React | Angular | Svelte |
|---|---|---|---|---|
| Рік появи | 2014 | 2013 | 2010 | 2016 |
| Розробник | Evan You | Meta (Facebook) | Google | Rich Harris (Vercel) |
| Тип | Фреймворк | Бібліотека | Повноцінний фреймворк | Фреймворк-компілятор |

Продовження таблиці 3.1

| | | | | |
|---|---|---|---|---|
| Синтаксис | HTML + JS/TS | JSX | HTML + TS + шаблони | HTML + JS/TS |
| Підхід до DOM | Віртуальний DOM | Віртуальний DOM | Віртуальний DOM | Прямий DOM (компіляція) |
| Двостороннє зв’язування | Є | Тільки вручну | | Є |
| Реактивність | Вбудована | Через useState/useEffect | RxJS | Вбудована |
| Поріг входу | Низький | Середній | Високий | Низький |
| Архітектура | Гнучка (MVVM / SPA) | Гнучка | Жорстка (DI, модулярність) | Мінімалістична |
| TypeScript підтримка | Вбудована | Є | Обов’язкова | Часткова |
| Документація | Зрозумі ла | Є | Складна, об'ємна | Зручна, але менша |
| Продуктивність | Висока | Висока | Середня | Дуже висока |

З проведеного порівняння видно, що всі розглянуті фреймворки мають свої переваги та особливості. React вирізняється популярністю та потужною екосистемою, але потребує глибшого розуміння внутрішніх механізмів. Angular пропонує повноцінне архітектурне рішення, проте є складним для новачків. Svelte демонструє високу продуктивність і мінімальний розмір, але ще не має такої розвиненої спільноти. Vue.js, натомість, поєднує простоту, гнучкість, високу продуктивність і чудову документацію, що робить його зручним як для новачків, так і для досвідчених розробників.

Саме тому для реалізації вебзастосунку в межах цієї бакалаврської роботи було обрано Vue.js — як оптимальне рішення, що забезпечує ефективну розробку без надлишкової складності.

## 3.2 Реалізація клієнтської частини

Клієнтська частина веб-ресурсу реалізована з використанням бібліотеки Vue.js, яка забезпечує створення інтерактивного та динамічного інтерфейсу користувача. Реактивна модель Vue дозволяє ефективно синхронізувати дані з

інтерфейсом у режимі реального часу, а компонентно-орієнтована архітектура спрощує розробку, тестування та повторне використання окремих елементів UI.

Основними перевагами використання Vue.js є простота інтеграції, зрозумілий синтаксис шаблонів, підтримка односторінкових застосунків (SPA), а також наявність інструментів для керування станом (наприклад, Pinia або Vuex) і маршрутизації (Vue Router).

Інтерфейс поділено на логічно ізольовані компоненти, кожен з яких відповідає за окрему функціональність чи візуальний блок, що дозволяє масштабувати застосунок без втрати зручності у підтримці та розширенні коду.

Основними компонентами інтерфейсу є:

— сторінка реєстрації та входу користувача, яка реазілує логіку введення облікових даних, перевірку правильності введених значеннь;

— головна сторінка сайту, на якій міститься інформація про проект, а також фінансові новини;

— сторінка з банківською карточкою, на якій міститься інформація про доходи/витрати за поточний місяць, категорії витрат, теги, а також форму для введення підписок на сервіси;

— сторінка з аналітикою криптовалютного ринку, а саме, топ 5 найпопулярніших криптовалют, та їхній графічний аналіз за місяць;

Розглянемо детальніше компоненти інтерфейсу на сторінці, а саме: сторінку реєтрації, логіну, основну сторінку, сторінку з карточкою, сторінку з криптовалютою.

Сторінка для реєстрації містить 4 поля: ім’я, електронну пошту, пароль, а також підтвердження паролю. Ці 4 поля допомагають запобігти надсиланню некоректної інформації на сервер, задля збереження цих даних до бази даних користувачів, сторінка для реєстрації наведена на рисунку 3.1

Валідація даних реалізована без використання сторонніх бібліотек, шляхом програмної перевірки введених значень безпосередньо у методі registerUser, реалізація метода registerUser, наведено у лістингу 3.1:

Лістинг 3.1 — Реалізація методу для реєстрацію користувача

```
async registerUser() {
    if (this.password !== this.confirmPassword) {
      this.errorMessage = 'Passwords do not match.';
      return;
    }
    try {
      const data = await auth.register(this.name, this.username, this.password);
      localStorage.setItem('user-token', data.access);
      this.$router.push('/dashboard');
    } catch (err) {
      this.errorMessage = 'Registration failed. Please try again.';
    }
  }
```

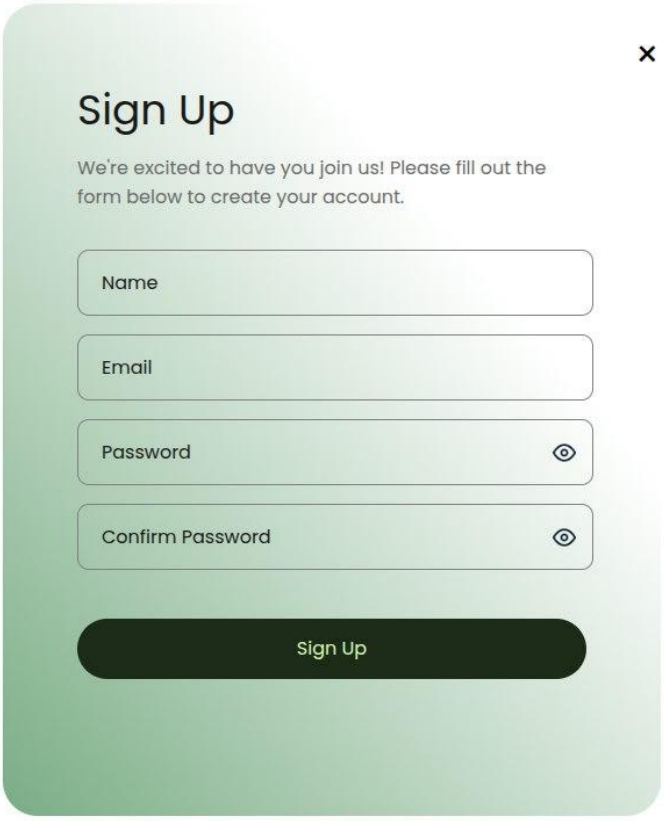


Рисунок 3.1 — Сторінка реєстрації користувача

Такий підхід дозволяє реалізувати базову логіку перевірки коректності даних на стороні клієнта з урахуванням специфіки бізнес-логіки веб-ресурсу. Зокрема, перевіряється відповідність пароля та його підтвердження, що забезпечує мінімальний захист від типових помилок користувача при реєстрації. У разі, якщо введені паролі не збігаються, користувачу виводиться відповідне

повідомлення про помилку.

Додатково застосовуються стандартні HTML-атрибути required, які активують базову перевірку обов’язковості полів засобами браузера. Такий підхід підвищує зручність користування інтерфейсом та забезпечує первинний рівень валідації без значного ускладнення коду.

У межах клієнтської частини також реалізовано форму авторизації (логіну) (рис.3.2), яка відповідає сучасним вимогам до користувацьких інтерфейсів і забезпечує базову валідацію даних без використання сторонніх бібліотек. Структура компонента побудована з використанням шаблонної системи Vue, де ключову роль відіграють дві змінні стану: username та password. Вони прив’язані до відповідних полів введення через директиву v-model, що забезпечує двосторонню зв’язність між даними та інтерфейсом

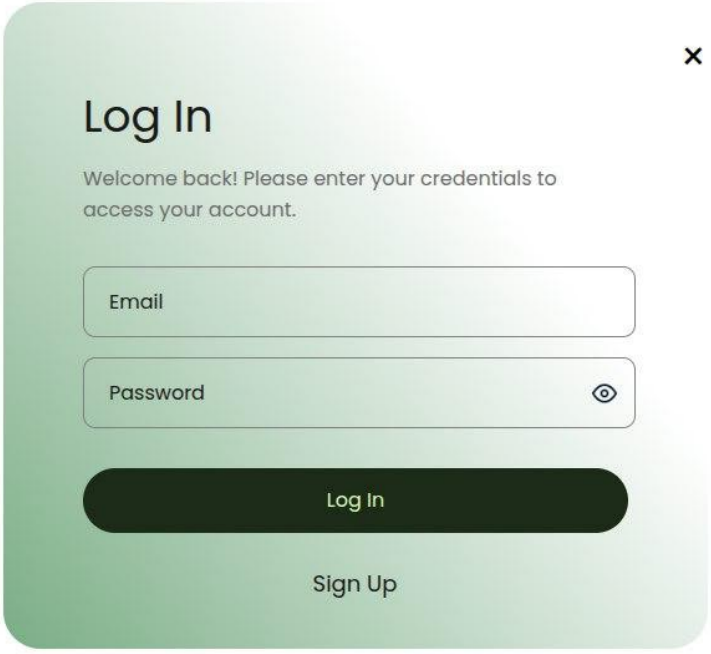


Рисунок 3.2 — Сторінка авторизації користувача

Процедура логіну реалізована у методі loginUser, який викликається після натискання кнопки «Log In». Метод працює асинхронно та ініціює запит до серверної частини через модуль auth, передаючи вказані облікові дані. Реалізація методу loginUser наведена в лістингу 3.2:

Лістинг 3.2 — Реалізація методу авторизації користувача

```
async loginUser() {
  try {
    const data = await auth.login(this.username, this.password);
    localStorage.setItem('user-token', data.access);
    this.$router.push('/home);
  } catch (err) {
    this.errorMessage = 'Invalid login or password.';
  }
}
```

У разі успішної авторизації, токен доступу зберігається у localStorage, після чого користувача перенаправляють на сторінку home. Така архітектура дозволяє підтримувати сесію користувача на клієнтській стороні та реалізовувати механізм захищеного доступу до внутрішніх розділів вебзастосунку.

Якщо авторизація завершується помилкою (наприклад, у випадку неправильного пароля або неіснуючого користувача), відповідне повідомлення відображається через змінну errorMessage, яка прив’язана до DOM-елемента за допомогою інтерполяції. Реалізація інтерполяції наведена в лістингу 3.3:

Лістинг 3.3 — Інтерполяція змінної

```
<p class="error" v-if="errorMessage">{{ errorMessage }}</p>
```

Це дозволяє оперативно інформувати користувача про результат взаємодії із сервером.

Візуальне оформлення логін-форми також було адаптовано під сучасні UI-практики, зокрема реалізовано можливість показу/приховування пароля за допомогою піктограми ока. Це реалізовано через умовну логіку в шаблоні та метод togglePassword. Реалізація методу togglePassword наведена в лістингу 3.4:

Лістинг 3.4 — Реалізація показу/приховування пароля

```
togglePassword() {
  this.showPassword = !this.showPassword;
}
```

Головна сторінка вебзастосунку виконує роль стартового інтерфейсу, з якого користувач починає взаємодію після успішної авторизації. Вона

реалізована з використанням Vue.js у форматі односторінкового компонента з шаблоном, стилями та логікою на стороні клієнта.

Структура сторінки побудована у відповідності до концепцій адаптивного дизайну — всі елементи компонуються у межах контейнера , який відповідає за основне компонування контенту та забезпечує гнучке масштабування на різних пристроях.

Однією з ключових функцій головної сторінки є динамічне завантаження та виведення стрічки новин. Це реалізовано через виклик API в методі життєвого циклу mounted(), який асинхронно витягує новини з бекенд-серверу. Реалізація методу mounted наведена в лістингу 3.5:

Лістинг 3.5 — Реалізація методу mounted

```
async mounted() {
  try {
    const response = await fetch(' http://0.0.0.0:8000/api/v1/news/');
    this.newsItems = await response.json();
  } catch (error) {
    console.error('Error fetching news:', error);
  }
}
```

Отримані новини зберігаються у масиві newsItems, який потім ітеративно відображається у шаблоні за допомогою конструкції v-for. Реалізація ітерації наведена в лістингу 3.6:

Лістинг 3.6 – Реалізація ітерації по масиму newsItems

```
<div class="news-section">
  <h2>Latest News</h2>
  <div v-for="item in newsItems" :key="item.id" class="news-item">
    <h3>{{ item.title }}</h3>
    <p>{{ item.description }}</p>
    <a :href="item.link" target="_blank">Read more</a>
  </div>
</div>
```

Такий підхід дозволяє підтримувати актуальність вмісту без потреби перезавантаження сторінки, що покращує загальний користувацький досвід.

Головна сторінка використовує сучасні підходи до UI-дизайну, зокрема:

— адаптивні стилі, що забезпечують коректне відображення як на

десктопах, так і на мобільних пристроях;

— модульну сітку, яка дозволяє компонувати новини, повідомлення або віджети в логічні блоки;

— анімації при завантаженні;

Окрім стрічки новин, на головній сторінці також реалізовано основну навігацію вебзастосунку. Вона включає верхній навігаційний блок із логотипом та кнопкою «Log out», яка дозволяє користувачу завершити сесію. Ця кнопка викликає метод logout, що очищає збережені дані користувача з localStorage та перенаправляє його на сторінку входу. Реалізація методу logout наведена в лістингу 3.7:

Лістинг 3.7 — Реалізація методу logout для виходу з облікового запису

```
logout() {
  localStorage.removeItem('user-token');
  this.$router.push('/login');
}
```

Такий підхід гарантує просту та безпечну реалізацію завершення сесії без необхідності звернення до серверу, що знижує затрати на обробку з боку бекенду.

На головній сторінці також реалізована кнопка «Read more», яка відкриває зовнішній ресурс, зокрема — репозиторій проєкту на платформі GitHub. Це дає можливість користувачам детальніше ознайомитися з кодом, структурою або технічною реалізацією відповідного проєкту, якщо вони зацікавились коротким описом, що представлений на сайті.

З технічної точки зору, функціональність реалізована за допомогою стандартного HTML-елемента <a> із вкладеною кнопкою, яка стилізована відповідно до дизайну інтерфейсу. Реалізація наведена в лістингу 3.8:

Лістинг 3.8 — Реалізація посилання та вкладеної кнопки

```
<a href="https://github.com/PetrikDima/banking-service" target="_blank" rel="noopener noreferrer">
  <button class="btn-primary">Read more &gt;</button>
</a>
```

Використання атрибутів target="_blank" і rel="noopener noreferrer" забезпечує відкриття посилання в новій вкладці та захищає від потенційних вразливостей, пов’язаних із маніпуляцією контекстом window.opener.

На рисунках 3.3 та 3.4 зображено вигляд основної сторінки програми.

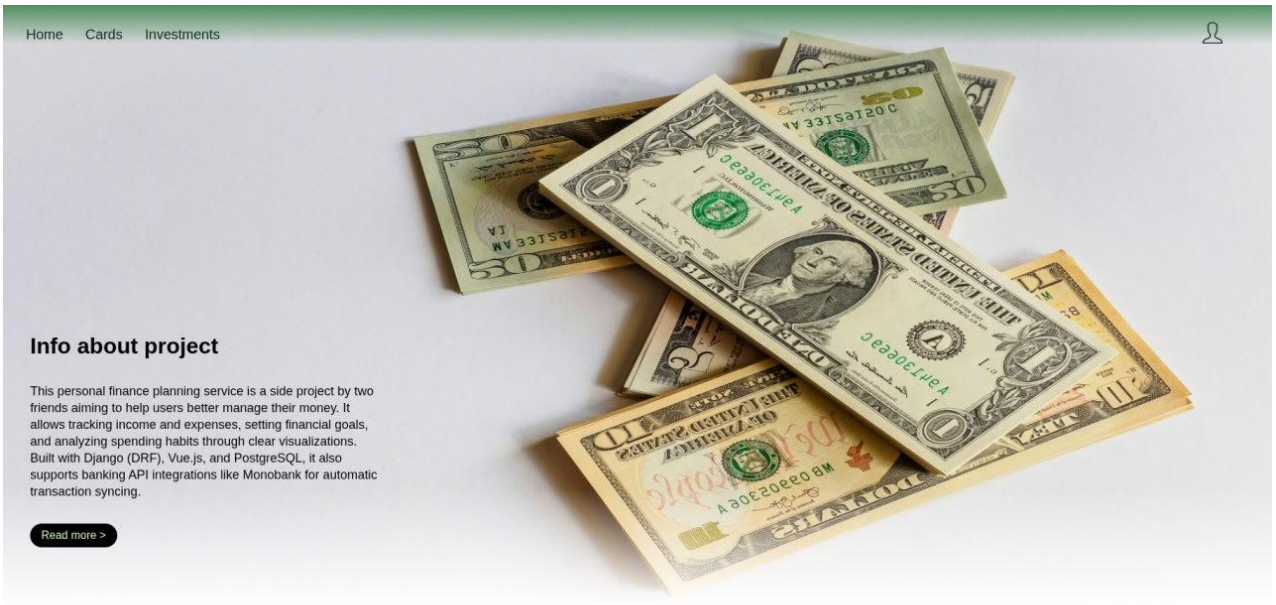


Рисунок 3.3 — Основна сторінка застосунку


Рисунок 3.4 — Продовження основної сторінки застосунку

При натисканні на картку новини користувач автоматично перенаправляється на сторінку з повним текстом відповідної публікації. Такий механізм забезпечує інтуїтивно зрозумілу навігацію та дозволяє зосередити увагу

на стислому огляді новин у головному інтерфейсі, водночас залишаючи можливість отримати детальну інформацію за потреби (рисунок 3.5).

У клієнтській частині проєкту реалізовано сторінку Cards (рисунки 3.6, 3.7), яка відповідає за відображення ключової інформації у вигляді окремих візуальних блоків — карточок. Даний компонент створений із застосуванням шаблонної системи Vue.js, що забезпечує ефективне управління станом і реактивне оновлення інтерфейсу користувача.


Рисунок 3.5 — Сайт yahoo з детальним описом фінансової новини

На рівні компоненту визначено набір властивостей (props), які відповідають за прийом вхідних даних для відображення. Дані про фінансові операції або інші сутності передаються зі сторінки в компонент Card у вигляді структурованих об’єктів, що сприяє повторному використанню одного й того ж компонента з різними наборами інформації.

Передача даних на сторінці відбувається після їх завантаження з серверної частини через асинхронний виклик API. Отримані дані зберігаються у локальному стані компонента, що дозволяє при зміні інформації миттєво

оновлювати інтерфейс без необхідності перезавантаження сторінки. Для зв’язку між властивостями компонента та полями шаблону використовується двостороння реактивність Vue, що забезпечує узгодженість даних між логікою та представленням.

Візуальне оформлення кожної карточки включає заголовок, опис та числові показники, які форматуються згідно з локальними стандартами валютного відображення. Такий підхід сприяє покращенню сприйняття інформації користувачем і відповідає сучасним вимогам до UI/UX.

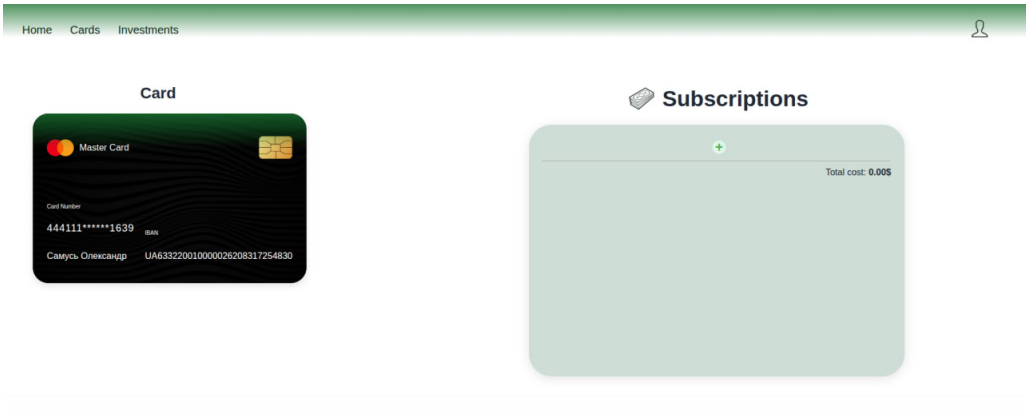


Рисунок 3.6 — Сторінка Cards


Рисунок 3.7 — Продовження сторінки Cards

Компонент Card сторінки Cards реалізує відображення інформаційної банківської картки, дані для якої динамічно отримуються з віддаленого серверу

після монтування компонента. Для цього використовується життєвий цикл onMounted() з асинхронною функцією, що виконує два HTTP-запити за допомогою бібліотеки axios.

У першому запиті здійснюється звернення до ендпоінту /api/v1/monobank/token/, де перевіряється наявність та коректність токена доступу (JWT_TOKEN), який зберігається у localStorage. У разі успішної авторизації виконується другий запит до /api/v1/monobank/personal-info/, який повертає персоналізовану інформацію користувача, зокрема перелік рахунків.

Серед отриманих рахунків здійснюється пошук першого рахунку з позитивним балансом. Якщо такий знайдено, з нього витягуються маскований номер картки, IBAN та ім’я власника. Ці значення прив’язуються до відповідних змінних реактивного стану, таких як cardNumber, cardHolder та cardIban, які відображаються безпосередньо в інтерфейсі картки на сторінці.

Таким чином, реалізація забезпечує персоналізований та актуальний вміст для кожного користувача, без необхідності ручного оновлення або статичних шаблонів. Нижче в лістингу 3.10 представлено фрагмент коду, що відповідає за цю логіку:

Лістинг 3.9 — Реалізація методу onMounted для звернення до ендпоінтів /api/v1/monobank/token, /api/v1/monobank/personal-info/

```
onMounted(async () => {
  try {
    const token = localStorage.getItem("user-token")
    const response = await axios.get("http://0.0.0.0:8001/api/v1/monobank/token/", {
      headers: { Authorization: `JWT_TOKEN ${token}` },
    })
    const response2 = await axios.get("http://0.0.0.0:8001/api/v1/monobank/personal-info/", {
      headers: { Authorization: `JWT_TOKEN ${token}` },
    })
    const accounts = response2.data.accounts
    const accountWithMoney = accounts.find(acc => typeof acc.balance === 'number' &&
acc.balance > 0)
    if (accountWithMoney) {
      cardNumber.value = accountWithMoney.maskedPan?.[0] || '**** **** **** ****'
      cardHolder.value = response2.data.name || 'Unknown'
      cardIban.value = accountWithMoney.iban || '**** **** **** ****'
    }
  } catch (error) {
```

```
    console.error("Помилка отримання даних картки:", error)
  }
})

<div class="layout-wrapper">
     <h2 class="cards-title">Card</h2>
     <div class="container">
       <header>
       <span class="logo">
         <img src="../assets/logo.png" alt=""/>
         <h5>Master Card</h5>
       </span>
         <img src="../assets/chip.png" alt="" class="chip"/>
       </header>
       <div class="card-details">
         <div class="name-number">
           <h6>Card Number</h6>
           <h5 class="number">{{ cardNumber }}</h5>
           <h5 class="name">{{ cardHolder }}</h5>
         </div>
         <div class="valid-date">
           <h6>IBAN</h6>
           <h5>{{ cardIban }}</h5>
         </div>
       </div>
     </div>
```

Сторінка управління підписками реалізована з використанням реактивних змінних, що керують як списком існуючих підписок, так і логікою додавання нових. Виведення наявних підписок відбувається за допомогою директиви v-for, яка динамічно створює DOM-елементи на основі вмісту масиву subscriptions. Кожна підписка представлена назвою, сумою щомісячної оплати та іконкою долара як візуальним індикатором фінансового значення. Реалізація підписок наведена в лістингу 3.10:

Лістинг 3.10 – Реалізація боксу для підписок

```
<div class="subscriptions-box">
 <div
   class="subscription-item"
   v-for="(item, index) in subscriptions"
   :key="index"
 >
   <span class="dollar-icon">$</span>
   <span>{{ item.title }}</span>
   <span class="price">{{ item.payment_sum }}$</span>
 </div>
```

```
</div>
```

Користувач має можливість додати нову підписку через інтерактивну форму, яка активується кнопкою з позначкою «+». Після натискання змінна showForm встановлюється в true, внаслідок чого форма стає видимою (рис. 3.8). Нові значення підписки вводяться через інпут-поля, прив’язані до властивостей об’єкта newSubscription за допомогою директиви v-model, що забезпечує двосторонню зв’язність між формою та даними. Реалізація додавання підписки наведена в лістингу 3.11:

Лістинг 3.11 — Реалізація добавлення підписок

```
<button class="add-subscription-button" @click="toggleForm">
  <span class="plus-icon">+</span>
</button>

<div v-if="showForm" class="add-subscription-form">
  <input type="text" v-model="newSubscription.title" placeholder="Назва підписки" />
  <input type="text" v-model="newSubscription.description" placeholder="Опис" />
  <input type="number" v-model="newSubscription.payment_sum" placeholder="Сума" step="0.01" />
  <button @click="submitSubscription">Додати</button>
</div>
```

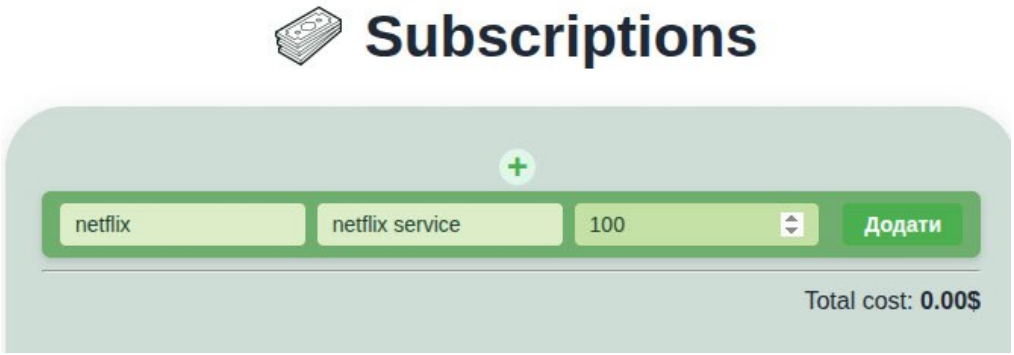


Рисунок 3.8 — Форма введення підписки

Після натискання кнопки «Додати» викликається метод submitSubscription, який додає новий об’єкт підписки до масиву subscriptions, що автоматично оновлює інтерфейс (рис. 3.9). Реалізація submitSubscription наведена в лістингу 3.12:

Лістинг 3.12 — Реалізація методу submitSubscription для добавлення підписки

```
const subscriptions = ref([])
const showForm = ref(false)
const newSubscription = ref({
  title: '',
  description: '',
  payment_sum: 0,
})
function toggleForm() {
  showForm.value = !showForm.value
}
function submitSubscription() {
  if (newSubscription.value.title && newSubscription.value.payment_sum > 0) {
    subscriptions.value.push({ ...newSubscription.value })
    newSubscription.value = { title: '', description: '', payment_sum: 0 }
    showForm.value = false
  }
}
```

Крім того, компонент обчислює загальну суму витрат по підписках за допомогою властивості computed, що постійно оновлюється в разі зміни вхідних даних. Реалізація методу computed, наведена в лістингу 3.13:

Лістинг 3.13 — Реалізація методу computed для обчислення суми витрат

```
const totalCost = computed(() =>
  subscriptions.value.reduce((sum, sub) => sum + Number(sub.payment_sum), 0)
)
```

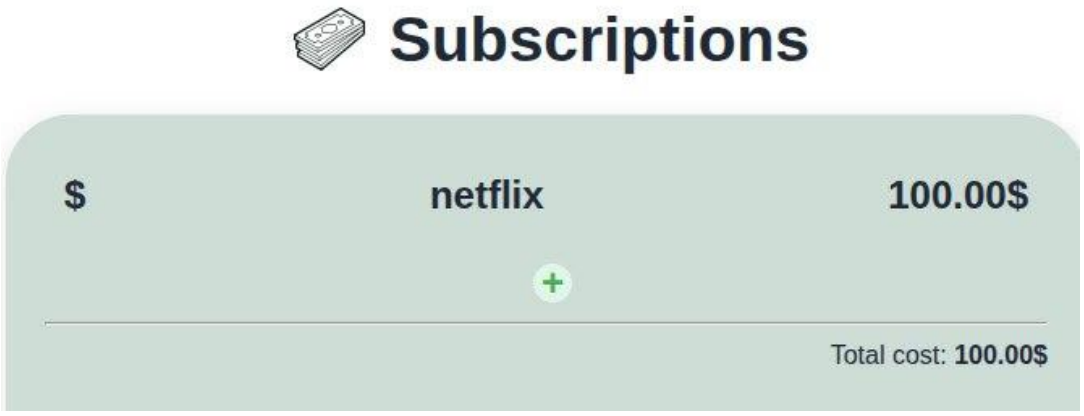


Рисунок 3.9 — Добавлена підписка Netflix

Таким чином, реалізація компоненту підписок забезпечує зручне додавання нових елементів, автоматичне оновлення інтерфейсу та динамічний підрахунок загальної вартості витрат.

Компонент Costs/Income реалізує змогу переглядати статистику витрат і доходів за поточний період за допомогою інтерактивного компонента, який реалізовано у вигляді перемикної картки. Початково активною є вкладка витрат, що задається змінною activeTab, ініціалізованою як ref('costs'). У шаблоні це контролюється через директиву v-if, яка відображає відповідний заголовок і стилі, залежно від вибраної вкладки. Наприклад, у заголовку картки виводиться динамічний текст через конструкцію. Реалізація перемикання наведена в лістингу 3.14:

Лістинг 3.14 — Реалізація перемикання компонентів Costs/Income

```
<span class="costs-title">
  {{ activeTab === 'costs' ? 'Costs' : 'Income' }}
</span>
```

Також використовується SVG-іконка, яка підставляється за допомогою тернарного оператора. Реалізація підставлення наведена в лістингу 3.15:

Лістинг 3.15 — Реалізація підставлення SVG-іконки

```
<img
  :src="activeTab === 'costs' ? iconCosts : iconIncome"
  class="w-[40px] h-[40px]"
/>
```

Перемикання між вкладками реалізовано через стрілки, які викликають метод switchTab(). Реалізація методу switchTab наведена в лістингу 3.16:

Лістинг 3.16 — Реалізація методу switchTab для перемикання компонентів

```
<div class="arrow left" @click="switchTab">
  <ChevronLeft />
</div>
<div class="arrow right" @click="switchTab">
  <ChevronRight />
</div>
function switchTab() {
  activeTab.value = activeTab.value === 'costs' ? 'income' : 'costs'
}
```

Сума доходів та витрат обчислюється після отримання всіх транзакцій з Monobank API. Для цього формується запит monobank/statement/{accountId}/{dateFrom}/{dateTo}/?category=true, результат

якого містить масив транзакцій. Їх обробка здійснюється функцією calculateTransactionsSum, яка проходиться по кожному елементу масиву transactions.value і класифікує значення як позитивні або негативні. Реалізація методу calculateTransactionsSum наведена в лістингу 3.17:

Лістинг 3.17 — Реалізація методу calculateTransactionsSum для обчислення доходів та витрат

```
function calculateTransactionsSum() {
  let totalNegative = 0
  let totalPositive = 0
  for (const item of transactions.value) {
    if (item.amount < 0) totalNegative += item.amount
    else totalPositive += item.amount
  }
  cardCosts.value = (totalNegative / 100).toFixed(2)
  cardIncome.value = (totalPositive / 100).toFixed(2)
}
```

Крім основної суми, користувач також бачить кількість активних категорій та тегів. Категорії формуються на основі об'єкта result, що генерується при обробці транзакцій у функції groupCategories(). Реалізація методу groupCategories наведена в лістингу 3.18:

Лістинг 3.18 — Реалізація методу groupCategories для групування категорій

```
function groupCategories(transactions) {
  const result = {}
  for (const item of transactions) {
    const category = item.category || 'Other'
    const total = result[category] || 0
    result[category] = total + item.amount
  }
  return result
}
```

Для кожного з типів вкладок застосовується фільтрація: для витрат – лише від’ємні числа, а для доходів – додатні. Реалізація фільтрації наведена в лістингу 3.19:

Лістинг 3.19 — Реалізація методу computed для фільтрації витрат та доходів

```
computed(() => {
```

```
  const result = {}
  for (const [category, total] of Object.entries(rawCategoriesMap.value)) {
    if (activeTab.value === 'costs' && total < 0) {
      result[category] = (total / 100).toFixed(2)
    } else if (activeTab.value === 'income' && total > 0) {
      result[category] = (total / 100).toFixed(2)
    }
  }
  return result
})
```

Компонент Costs/Income виглядає наступним чином (рис. 3.10, рис. 3.11)


Рисунок 3.10 — Компонент Costs з кількістю категорій та тегів

Залишаються категорії (рис. 3.12) – компонент для виведення списку повних категорій транзакцій. Цей компонент дозволяє користувачеві переглядати розподіл витрат або надходжень за категоріями у зручному вигляді. Нижче описується логіка його функціонування. Реалізація категорій витрат/доходів наведена в лістингу 3.20.

Лістинг 3.20 — Реалізація категорій витрат/доходів

```
<div class="categories-box">
  <h3>Categories</h3>
  <div class="categories-list">
    <div v-if="Object.keys(categoriesMap).length">
      <div style="margin-bottom: 10px" v-for="(amount, category) in categoriesMap" :key="category">
```

```
        <span>{{ category }}: {{ amount }} грн</span>
      </div>
    </div>
  </div>
</div>
```


Рисунок 3.11 — Компонент Income з категоріями та тегами

Категорії надходять у вигляді об’єкта rawCategoriesMap, де ключ – це назва категорії, а значення – масив транзакцій цієї категорії. Цей об’єкт отримується після запиту до бекенду і зберігається у реактивній змінній rawCategoriesMap.value.

Для подальшої роботи з категоріями визначене обчислюване властивість categoriesMap. Воно динамічно перетворює rawCategoriesMap залежно від вибраної користувачем вкладки: або витрати (costs), або доходи (income). Для кожної категорії відбувається проходження по транзакціях, де обчислюється сумарна сума операцій, ігноруючи тимчасово заблоковані (hold) транзакції.

Якщо активна вкладка — витрати, у categoriesMap потрапляють лише ті категорії, де загальна сума менша за нуль, тобто це категорії з від’ємним балансом. Якщо ж вкладка — доходи, враховуються категорії з додатнім балансом. Всі суми виводяться у гривнях, для цього значення діляться на 100 і округлюються до двох знаків після коми.

Таким чином, categoriesMap представляє вже відфільтрований та готовий для відображення у UI об’єкт із категоріями і їхніми підсумками відповідно до вибраної вкладки. Це забезпечує користувачу можливість швидко побачити сумарні витрати чи доходи по категоріях за заданий період. Реалізація методу categoriesMap наведена в лістингу 3.21.

Лістинг 3.21 — Реалізація методу categoriesMap для суми операцій

```
const rawCategoriesMap = ref({})
const categoriesMap = computed(() => {
  const result = {}
  if (!rawCategoriesMap.value || typeof rawCategoriesMap.value !== 'object') return result
  for (const category in rawCategoriesMap.value) {
    const transactions = rawCategoriesMap.value[category]
    const total = transactions.reduce((sum, tx) => {
      const amount = tx.operationAmount
      return sum + (tx.hold ? 0 : amount)
    }, 0)
    if (activeTab.value === 'costs' && total < 0) {
      result[category] = (total / 100).toFixed(2)
    } else if (activeTab.value === 'income' && total > 0) {
      result[category] = (total / 100).toFixed(2)
    }
  }
  return result
})
```

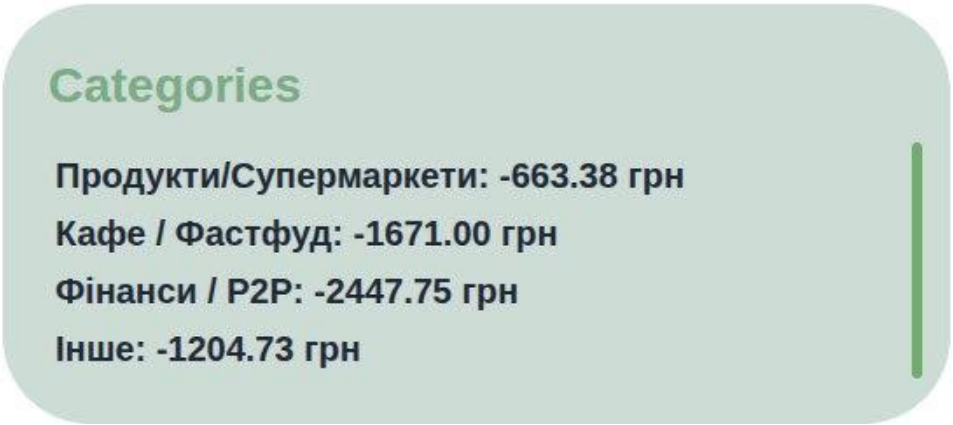


Рисунок 3.12 — Категорії витрат

Сторінка Investments є частиною веб-додатку, який дозволяє користувачу переглядати інформацію про п’ять основних криптовалют: Bitcoin, Ethereum, BNB, Solana та XRP. Кожна криптовалюта представлена окремим візуальним

компонентом з відповідною іконкою, що робить інтерфейс більш інтуїтивним і зручним для користувача. У коді кожна криптовалюта реалізована через компонент <Item>, що приймає такі параметри, як назва, торговий символ, іконка, а також властивість isSelected, яка визначає, чи обрана ця криптовалюта в даний момент. Реалізація компонентів <Item> наведена в лістингу 3.22.

Лістинг 3.22 — Реалізація компонентів <Item> для відображення іконок

```
<Item
  name="Bitcoin"
  trading_key="BTC"
  :img="bitcoinIcon"
  @select="handleCryptoSelect"
  :isSelected="selectedCrypto === 'BTC'"
/>
<Item
  name="Ethereum"
  trading_key="ETH"
  :img="ethereumIcon"
  @select="handleCryptoSelect"
  :isSelected="selectedCrypto === 'ETH'"
/>
<Item
  name="BNB"
  trading_key="BNB"
  :img="bnbIcon"
  @select="handleCryptoSelect"
  :isSelected="selectedCrypto === 'BNB'"
/>
<Item
  name="Solana"
  trading_key="SOL"
  :img="solanaIcon"
  @select="handleCryptoSelect"
  :isSelected="selectedCrypto === 'SOL'"
/>
<Item
  name="XRP"
  trading_key="XRP"
  :img="xrpIcon"
  @select="handleCryptoSelect"
  :isSelected="selectedCrypto === 'XRP'"
/>
```

Коли користувач обирає певну криптовалюту на сторінці, ця валюта позначається як активна, і в інтерфейсі відбувається оновлення стану, щоб відобразити вибір. Запускається асинхронна функція, яка надсилає запит на

бекенд для отримання торгових даних з різних бірж. Це дозволяє користувачу бачити інформацію в режимі реального часу. В процесі показується індикатор спіннера (рис. 3.13), щоб користувач розумів, що дані оновлюються. Реалізація методу handleCryptoSelect для завантаження спіннера, наведена в лістингу 3.23:

Лістинг 3.23 — Реалізація методу handleCryptoSelect для завантаження спіннера

```
<script setup>
import { ref } from 'vue'
import { getTradingByName, getPriceByName } from '@/api/investments.js'
const selectedCrypto = ref(null)
const exchangePrices = ref([])
const cryptoPrices = ref([])
const isLoading = ref(false)
async function handleCryptoSelect(tradingKey) {
  selectedCrypto.value = tradingKey
  isLoading.value = true
  try {
    const tradingData = await getTradingByName(tradingKey)
    exchangePrices.value = tradingData
    const priceData = await getPriceByName(tradingKey)
    cryptoPrices.value = priceData
  } catch (error) {
    console.error('Error loading crypto data:', error)
  } finally {
    isLoading.value = false
  }
}
</script>
<template>
  <div>
    <Item
      v-for="crypto in cryptos"
      :key="crypto.trading_key"
      :name="crypto.name"
      :trading_key="crypto.trading_key"
      :img="crypto.img"
      :isSelected="selectedCrypto === crypto.trading_key"
      @select="handleCryptoSelect(crypto.trading_key)"
    />
    <Spinner v-if="isLoading" />
      </div>
</template>
```

У цьому коді handleCryptoSelect відповідає за оновлення вибраної криптовалюти (selectedCrypto), запускає завантаження даних, а також керує

відображенням індикатора завантаження. Функції getTradingByName і getPriceByName асинхронно отримують відповідні дані з бекенду.

Метод getTradingByName звертається до ендпоінту /top_cryptos, де отримує повний перелік торгових показників для різних криптовалют. Після отримання відповіді він перевіряє, чи є у відповіді інформація саме про криптовалюту з переданим іменем. Якщо так — повертає цю інформацію, якщо ні — логічно повідомляє про відсутність даних.


Рисунок 3.13 — Спіннер завантаження аналізу криптовалюти

Метод getPriceByName аналогічно запитує дані з ендпоінту /monthly_prices. Він призначений для отримання історичних або поточних цінових даних за місяць для конкретної криптовалюти. Обидва методи мають обробку помилок, яка виводить у консоль деталі проблеми, якщо запит не вдався, що дозволяє зручно відслідковувати проблеми із зв’язком або сервером. Реалізація методів getTradingByName, getPriceByName наведена в лістингу 3.24.

Лістинг 3.24 — Реалізація методів getTradingByName, getPriceByName для звернення до ендпоінтів /api/v1/top_cryptos/, /api/v1/monthly_prices/

```
import axios from 'axios';
const API_URL = 'http://0.0.0.0:8002/api/v1';
export const investments = {
   async getTradingByName(name) {
     try {
       const response = await axios.get(${API_URL}/top_cryptos);
       const data = response.data;
       console.log("Trading data", data)
       if (data[name]) {
         return data[name];
       } else {
         console.warn('Crypto not found:', name);
         return null;
       }
```

```
    } catch (error) {
      console.error('Error getting crypto prices:', error.response?.data  error);
      throw error;
    }
  },
  async getPriceByName(name){
    try {
      const response = await axios.get(`${API_URL}/monthly_prices`)
      const data = response.data
      console.log("Fetched monthly prices:", data)
      if (data.hasOwnProperty(name)) {
        return data[name]
      } else {
        console.warn(`No data found for crypto: ${name}`)
        return null
      }
    } catch (error) {
      console.error('Error fetching crypto prices:', error.response?.data  error)
      throw error
    }
  }
}
```

Вигляд сторінки Investments показано на рисунках 3.14, 3.15:


Рисунок 3.14 — Топ 5 криптовалют за ринковою капіталізацією

У третьому розділі бакалаврської роботи було детально розглянуто процес програмної реалізації онлайн сервісу фінансового планування, який побудовано з використанням сучасних фронтенд-технологій, а саме Vue.js, JavaScript, HTML та CSS. Цей набір інструментів дозволив створити інтуїтивно зрозумілий та зручний для користувачів інтерфейс, що сприяє ефективній взаємодії з сервісом.

Розроблений вебзастосунок включає низку ключових сторінок, які забезпечують повний цикл користувацького досвіду від реєстрації до активного управління фінансами. Сторінка логіну та авторизації реалізує безпечний доступ до персональних даних, що є фундаментальним для збереження конфіденційності користувачів. Процес реєстрації забезпечує зручний старт для нових користувачів, надаючи можливість швидко зареєструватись і розпочати роботу з сервісом.

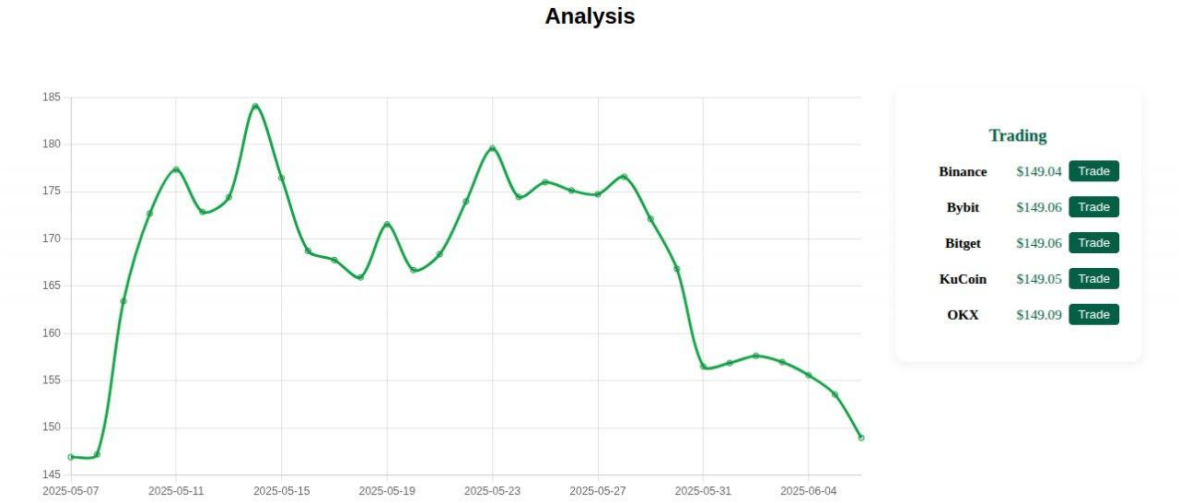


Рисунок 3.15 — Графічний аналіз за місяць криптовалюти по натисненню на іконку

Головна сторінка служить центром новин та актуальної інформації, що підтримує користувачів у курсі фінансових подій і тенденцій. Наявність функції логауту дозволяє забезпечити додатковий рівень безпеки, надаючи можливість коректного завершення сесії.

Окрему увагу приділено сторінці з карткою, де користувачі можуть детально переглядати свої доходи та витрати, що класифіковані за категоріями. Такий підхід дозволяє здійснювати більш глибокий аналіз особистих фінансів і сприяє формуванню усвідомленого фінансового планування.

Особливо важливою складовою сервісу є сторінка інвестицій, присвячена криптовалютам. В ній реалізовано аналіз топ-5 криптовалют за основними торговими показниками, що надає користувачам актуальні дані для прийняття

обґрунтованих інвестиційних рішень. Ця функціональність інтегрована з бекендом через асинхронні запити, що забезпечує своєчасне оновлення інформації та покращує користувацький досвід.

Загалом, застосування Vue.js і супутніх технологій дозволило створити масштабований, гнучкий та зручний у використанні сервіс, який може бути адаптований під потреби різних користувачів. Розроблена архітектура інтерфейсу забезпечує не лише базові функції фінансового планування, але й підтримує розвиток додаткових можливостей, що робить сервіс конкурентоспроможним та перспективним для подальшого вдосконалення.

Таким чином, реалізація програмного забезпечення онлайн сервісу фінансового планування є успішною і відповідає поставленим завданням щодо функціональності, зручності та безпеки користування. Отриманий результат відкриває широкі можливості для подальшого розвитку системи, впровадження нових аналітичних інструментів та інтеграції з іншими фінансовими сервісами. У майбутньому планується розширення функціоналу за рахунок додавання більш глибокої аналітики, автоматизованих рекомендацій щодо інвестицій і персоналізованих фінансових планів, що допоможе користувачам ефективніше керувати своїми фінансами. Також перспективним напрямком є інтеграція з додатковими банківськими API та сервісами, що дозволить зробити платформу більш універсальною та зручною для широкого кола користувачів, підвищуючи її конкурентоспроможність на ринку фінансових технологій.

# ВИСНОВКИ

Під час виконання бакалаврської роботи з розробки онлайн сервісу фінансового планування було реалізовано сучасний та комплексний веб-застосунок, що відповідає вимогам ефективного управління особистими фінансами в умовах цифрової економіки. Створена система включає низку важливих функціональних модулів, серед яких логін та авторизація користувачів із захищеним доступом, можливість реєстрації нових акаунтів, а також головна сторінка із відображенням актуальних фінансових новин, що дозволяє користувачам бути в курсі останніх тенденцій і подій на ринку.

Особливу увагу було приділено реалізації функціональних можливостей для управління доходами та витратами, де кожен користувач має змогу детально аналізувати свої фінансові операції по картках, розподіляти витрати за категоріями і отримувати візуалізовані звіти. Крім того, важливою складовою є сторінка інвестицій, на якій відображаються дані про топ-5 криптовалют із відповідними іконками, що забезпечує швидкий візуальний доступ до інформації. Завдяки впровадженню асинхронних запитів до бекенду, користувачі отримують оперативні та актуальні дані про ціни, торгові обсяги та інші ключові показники, які формуються на основі інформації з кількох бірж.

Для розробки інтерфейсу використовувалися сучасні веб-технології такі як Vue.js, JavaScript, HTML та CSS. Вони дозволили створити інтуїтивно зрозумілий, адаптивний та привабливий для користувача дизайн, який значно підвищує зручність роботи із сервісом на різних пристроях, включно з мобільними. Гнучка архітектура фронтенду у поєднанні з налаштованими API-запитами забезпечують швидкість роботи і стабільність системи навіть при великих обсягах даних.

Важливо відзначити, що особлива увага була приділена питанням безпеки, адже застосовані сучасні методи авторизації, захист даних користувачів, а також контроль доступу гарантують конфіденційність інформації і убезпечити систему від несанкціонованого доступу. Такий підхід забезпечує довіру користувачів до

платформи та створює надійне середовище для управління фінансами.

Реалізований онлайн сервіс фінансового планування успішно виконує основні завдання, поставлені у рамках роботи: автоматизує процес обліку фінансових операцій, дає змогу аналізувати структуру витрат і доходів, а також надає інструменти для ефективного інвестування. Відповідно, це сприяє підвищенню фінансової грамотності користувачів, полегшує прийняття обґрунтованих рішень щодо особистих фінансів та формує базу для відповідального фінансового планування.

Отримані результати відкривають широкі перспективи для подальшого розвитку системи. Можливе впровадження розширених аналітичних модулів із застосуванням штучного інтелекту, інтеграція з додатковими фінансовими сервісами та API банківських установ, а також розвиток мобільних додатків. Це дозволить не лише розширити функціонал, але й підвищити зручність і гнучкість використання платформи, адаптуючи її під індивідуальні потреби користувачів.

Таким чином, проведена робота підтверджує, що розробка онлайн сервісу фінансового планування з використанням сучасних веб-технологій є актуальною, технологічно досяжною і перспективною. Вона сприяє розвитку цифрової економіки та створенню умов для більш ефективного управління особистими фінансами, що є ключовим фактором підвищення якості життя у сучасному суспільстві. Отже, результати роботи можуть бути успішно використані як основа для подальших досліджень і проектів у сфері фінансових технологій.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналіз методів фінансового планування в умовах цифрової трансформації економіки. URL: https://economyandsociety.in.ua/journals/11_ukr/46.pdf (дата звернення: 24.03.2025).

2. Управління оборотним капіталом промислових підприємств | Економіка та суспільство. URL: https://economyandsociety.in.ua/index.php/journal/article/view/1234 (дата звернення: 05.05.2025).

3. Фінансовий менеджмент / Бланк І.А. — НТУУ «КПІ». URL: https://library.kpi.ua/uk/book/financial-management-blank (дата звернення: 27.04.2025).

4. Підходи до оптимізації особистих фінансів у сучасних умовах. Business Inform. URL: https://www.business-inform.net/article/view/5678 (дата звернення: 21.03.2025).

5. 10 застосунків і програм для роботи з грошима. Bazilik Media. URL: https://bazilik.media/10-zastosunkiv-i-prohram-dlia-roboty-z-hroshyma/ (дата звернення: 19.05.2025).

6. Python 3.13 documentation. Python Software Foundation. URL: https://docs.python.org (дата звернення: 30.05.2025).

7. Django documentation. Django Software Foundation. URL: https://docs.djangoproject.com (дата звернення: 29.05.2025).

8. PostgreSQL documentation. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs (дата звернення: 11.04.2025).

9. API Monobank – документація. Monobank. URL: https://api.monobank.ua/docs/ (дата звернення: 17.04.2025).

10. Як я створив фінансовий трекер з використанням Django та Vue. DOU.ua. URL: https://dou.ua/forums/topic/50364/ (дата звернення: 04.05.2025).

11. ECMA-404 The JSON Data Interchange Standard. Ecma International. URL: https://ecma-international.org/publications-and-standards/standards/ecma-404/ (дата звернення: 23.05.2025).

12. Як використовувати JSON Web Tokens (JWT) для автентифікації. DevZone. URL: https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi (дата звернення: 13.04.2025).

# ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«21» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання комплексної бакалаврської кваліфікаційної роботи
«Онлайн-сервіс фінансового планування. Частина 2. Клієнтський модуль»
08-54. КБКР.042.00.000 ПЗ

Науковий керівник: к.т.н., доц. каф. ОТ
_____Тарновський М.Г.

Студент групи 2КІ-21б
_____Самусь О.В.

м. Вінниця — 2025

1 Підстава для використання комплексної бакалаврської кваліфікаційної роботи (КБКР)

1.1 У сучасних умовах цифрової трансформації особливого значення набувають онлайн сервіси фінансового планування, які забезпечують користувачам зручний доступ до управління особистими коштами. Такі сервіси дають змогу контролювати поточні доходи та витрати, аналізувати фінансові звички, відстежувати динаміку змін по картках і категоріях, а також слідкувати за новинами та інвестиційною активністю. Інтеграція з API банків і використання сучасних вебтехнологій створюють основу для подальшого розширення функціоналу та підвищення фінансової грамотності користувачів.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета КБКР і призначення розробки

2.1 Мета роботи — розробка вебзастосунку для фінансового планування, який дозволяє користувачам ефективно керувати особистими фінансами шляхом відображення доходів і витрат, аналізу транзакцій по категоріях, доступу до фінансових новин, а також моніторингу інвестиційної активності, зокрема в сфері криптовалют.

2.2 Призначення розробки — створення клієнтської частини вебінтерфейсу онлайн-сервісу фінансового планування для надання користувачам зручного доступу до інформації про їхні доходи, витрати, фінансову аналітику, новини зі світу економіки та інвестиційні показники, зокрема дані про криптовалюти.

3 Вихідні дані для виконання КБКР

3.1 Проведення аналізу сучасних онлайн-сервісів для фінансового планування, принципів візуалізації даних, обробки транзакцій та роботи з API банків і криптовалютних бірж;

3.2 Проєктування архітектури вебзастосунку, розробка логіки взаємодії між компонентами інтерфейсу та користувачем;

3.3 Створення макетів і прототипів основних сторінок застосунку, включно з реєстрацією, авторизацією, фінансовою аналітикою, інвестиціями та новинами;

3.4 Розробка клієнтської частини застосунку з використанням HTML, CSS, JavaScript та фреймворку Vue.js;

3.5 Реалізація інтеграції з бекендом для отримання даних по транзакціях, категоріях витрат, новинах і криптовалютах;

4 Вимоги до виконання БКР

4.1 Проведено аналіз сучасних онлайн-сервісів для фінансового планування;

4.2 Розроблена архітектура вебзастосунку;

4.3 Розроблена програмна реалізація;

4.4 Розроблена пояснювальна записка, подані графічні матеріали та презентації;

5 Етапи КБКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

| № етапу | Назва етапу | Термін виконання | Очікувані результати |
|---|---|---|---|
| 1 | Аналіз сучасних сервісів фінансового планування | 21.03.25 – 29.03.25 | Розділ 1 |
| 2 | Архітектура сервісу фінансового планування | 30.03.25 – 09.04.25 | Розділ 2 |
| 3 | Програмна реалізація | 10.04.25 – 15.05.25 | Розділ 3 |
| 4 | Оформлення пояснювальної записки, графічного матеріалу та презентації | 10.04.25 – 13.05.25 | Пояснювальна записка, графічний матеріал та презентація |

6 Матеріали, що подаються до захисту КБКР

До захисту подаються: пояснювальна записка КБКР, графічні та ілюстративні матеріали, протокол попереднього захисту на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних

екзаменів, анотації до КБКР українською та іноземною мовами, довідка про відповідність оформлення вимогам.

7 Порядок контролю виконання та захисту КБКР

Контроль виконання етапів здійснюється науковим керівником згідно з встановленими термінами. Захист КБКР проходить на засіданні екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлення та порядок виконання КБКР

8.1 При оформленні КБКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») — кафедра обчислювальної техніки, ВНТУ 2023.

8.2 Порядок виконання КБКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

# ДОДАТОК Б

# ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: _____Онлайн-сервіс фінансового планування. Частина 2. Клієнтський модуль___

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___6___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Тарновський М.Г._к.т.н., доц. каф. ОТ___
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Самусь О.В._____
(підпис) (прізвище, ініціали)

# ДОДАТОК В

# ІЛЮСТРАТИВНА ЧАСТИНА

АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС МОНІТОРИНГУ СТАНУ ОТОЧУЮЧОГО СЕРЕДОВИЩА НА БАЗІ ІОТ ЗАСОБІВ. ЧАСТИНА 3. ВЕБЗАСТОСУНОК ДЛЯ ТРАНСЛЯЦІЇ ВІДЕОПОТОКУ

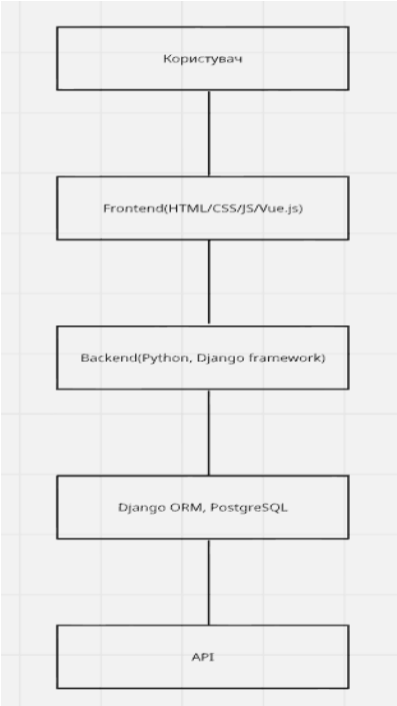


Рисунок В.1 — Схема обміну даними

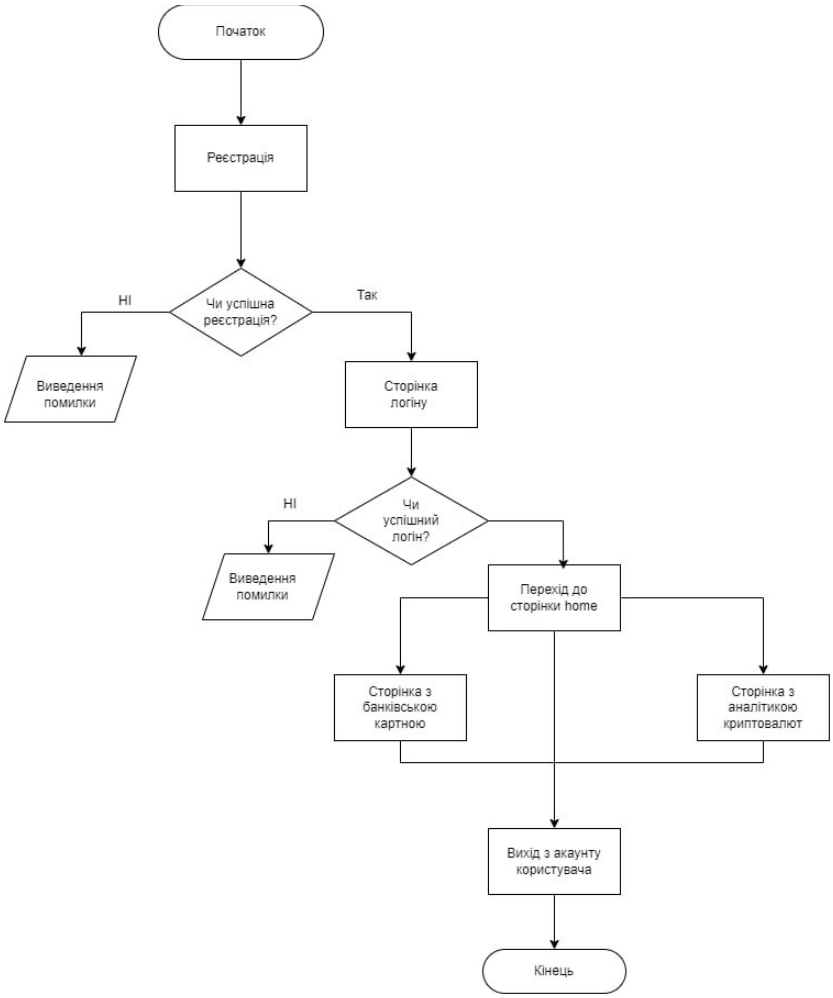


Рисунок В.2 — Алгоритм роботи застосунку

# ДОДАТОК Г

## Лістинг програмного коду файлу Register.vue

```
<template>
  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">
  <div class="login-wrapper">
    <div class="close-button" @click="closeForm">&times;</div>
    <div class="login-header">
      <h1>Sign Up</h1>
      <p>We're excited to have you join us! Please fill out the form below to create your account.</p>
    </div>
    <div class="input-form">
      <div class="input-group">
        <input v-model="username" type="text" placeholder="Name" required/>
      </div>
      <div class="input-group">
        <input v-model="email" type="email" placeholder="Email" required/>
      </div>
      <div class="input-group password-wrapper">
        <input :type="showPassword ? 'text' : 'password'" v-model="password" placeholder="Password" required/>
        <img
            v-if="showPassword"
            src="@/assets/icons/eye-off.png"
            alt="Hide Password"
            class="toggle-visibility"
            @click="togglePassword"
        />
        <img
            v-else
            src="@/assets/icons/eye-on.png"
            alt="Show Password"
            class="toggle-visibility"
            @click="togglePassword"
        />
      </div>
      <div class="input-group password-wrapper">
        <input :type="showConfirmPassword ? 'text' : 'password'" v-model="confirmPassword"
               placeholder="Confirm Password" required/>
        <img
            v-if="showConfirmPassword"
            src="@/assets/icons/eye-off.png"
            alt="Hide Confirm Password"
            class="toggle-visibility"
            @click="toggleConfirmPassword"
        />
        <img
            v-else
            src="@/assets/icons/eye-on.png"
            alt="Show Confirm Password"
            class="toggle-visibility"
            @click="toggleConfirmPassword"
        />
      </div>
    </div>
    <button class="login-button" @click="registerUser">Sign Up</button>
    <p v-if="errorMessage" class="error-text">{{ errorMessage }}</p>
  </div>
</template>
```

```
<script>
import {auth} from '../scripts/auth';
export default {
  data() {
    return {
      username: '',
      email: '',
      password: '',
      confirmPassword: '',
      showPassword: false,
      errorMessage: '',
      showConfirmPassword: false,
    };
  },
  methods: {
    async registerUser() {
      if (this.password !== this.confirmPassword) {
        this.errorMessage = 'Passwords do not match.';
        return;
      }
      try {
        const data = await auth.register(this.username, this.email, this.password);
        localStorage.setItem('user-token', data.access);
        this.$router.push('/login');
      } catch (err) {
        this.errorMessage = 'Registration failed. Please try again.';
      }
    },
    togglePassword() {
      this.showPassword = !this.showPassword;
    },
    toggleConfirmPassword() {
      this.showConfirmPassword = !this.showConfirmPassword;
    },
    closeForm() {
      this.$emit('close');
    }
  }
};
</script>
<style scoped>
body {
  font-family: 'Poppins', sans-serif;
}
.login-wrapper {
  width: 566px;
  height: 700px;
  border-radius: 30px;
  background: linear-gradient(230deg, #ffffff 30%, #7AAD85 100%);
  padding: 0;
  margin: 100px auto;
  position: relative;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
  overflow: hidden;
}
.login-header {
  position: absolute;
  top: 64px;
  left: 64px;
  width: 438px;
  height: 123px;
```

```
}
.login-header h1 {
  font-size: 36px;
  margin: 0 0 10px 0;
  color: #1a1a1a;
  font-weight: 400;
  font-family: 'Poppins', sans-serif;
}
.login-header p {
  color: #6c6c6c;
  font-size: 16px;
  margin: 0;
}
.input-form {
  position: absolute;
  top: 212px;
  left: 64px;
  width: 444px;
}
.input-group {
  margin-bottom: 16px;
}
.input-group input {
  width: 100%;
  padding: 15px 20px;
  border-radius: 10px;
  border: 1px solid #808080;
  font-size: 16px;
  box-sizing: border-box;
  background: transparent;
  font-family: 'Poppins', sans-serif;
  color: #1a1a1a;
}
.input-group input:focus {
  outline: none;
  border-color: #7AAD85;
  box-shadow: 0 0 0 2px rgba(122, 173, 133, 0.2);
}
.password-wrapper {
  position: relative;
}
.password-wrapper .toggle-visibility {
  position: absolute;
  right: 15px;
  top: 50%;
  transform: translateY(-50%);
  cursor: pointer;
}
.password-wrapper img {
  width: 20px;
  height: 20px;
  cursor: pointer;
}
.login-button {
  position: absolute;
  top: 530px;
  left: 64px;
  width: 438px;
  height: 52px;
  border-radius: 30px;
  background-color: #1c2b18;
  color: #DBFEB8;
```

```
  font-size: 16px;
  border: none;
  cursor: pointer;
  line-height: 52px;
  font-weight: 300;
  font-family: 'Poppins', sans-serif;
  text-align: center;
  padding: 0;
}
.error-text {
  position: absolute;
  top: 500px;
  left: 64px;
  color: red;
  font-size: 14px;
}
.close-button {
  position: absolute;
  top: 20px;
  right: 20px;
  width: 2px;
  height: 2px;
  font-size: 32px;
  text-align: center;
  border-radius: 50%;
  cursor: pointer;
  transition: background 0.2s;
  color: #000;
}
.close-button:hover {
  background: rgba(0, 0, 0, 0.1);
}
.input-group input::placeholder {
  color: #1a1a1a;
  opacity: 1;
}
</style>
```

## Investments.vue

```
<template>
  <div class="page-wrapper">
    <header class="header">
      <nav class="nav">
        <a href="/home">Home</a>
        <a href="/cards">Cards</a>
        <a href="/investments">Investments</a>
      </nav>
      <div class="header-buttons">
        <div class="account-menu-container">
          <button @click="toggleAccountMenu">
            <img src="../assets/icons/accout_icon.svg" alt="account" />
          </button>
          <div v-if="showAccountMenu" class="account-menu">
            <button class="logout-button" @click="logoutUser">
              <img src="../assets/icons/logout.svg" alt="logout icon" class="logout-icon" />
              Log Out
            </button>
          </div>
        </div>
      </div>
    </header>
```

```
    <main class="content">
      <section>
        <h2 style="text-align: center; font-size: 30px">Cryptocurrency</h2>
        <div class="grid">
          <Item
            name="Bitcoin"
            trading_key="BTC"
            :img="bitcoinIcon"
            @select="handleCryptoSelect"
            :isSelected="selectedCrypto === 'BTC'"
          />
          <Item
            name="Ethereum"
            trading_key="ETH"
            :img="ethereumIcon"
            @select="handleCryptoSelect"
            :isSelected="selectedCrypto === 'ETH'"
          />
          <Item
            name="BNB"
            trading_key="BNB"
            :img="bnbIcon"
            @select="handleCryptoSelect"
            :isSelected="selectedCrypto === 'BNB'"
          />
          <Item
            name="Solana"
            trading_key="SOL"
            :img="solanaIcon"
            @select="handleCryptoSelect"
            :isSelected="selectedCrypto === 'SOL'"
          />
          <Item
            name="XRP"
            trading_key="XRP"
            :img="xrpIcon"
            @select="handleCryptoSelect"
            :isSelected="selectedCrypto === 'XRP'"
          />
        </div>
        <div v-if="selectedCrypto">
          <h2 style="text-align: center">Analysis</h2>
          <div class="chart-section">
            <div v-if="isLoading" class="spinner-overlay">
              <div class="spinner-wrapper">
                <div class="spinner"></div>
                <p class="spinner-text">
                  Loading data<span class="dots"><span>.</span><span>.</span><span>.</span></span>
                </p>
              </div>
            </div>
            <div class="chart-and-markets" :style="{ opacity: isLoading ? 0.3 : 1, pointerEvents: isLoading ? 'none' : 'auto'
}">
              <div class="chart-container">
                <LineChart :data="crypto_prices" :label="selectedCrypto" />
              </div>
              <div class="market-prices">
                <h3>Trading</h3>
                <ul>
                  <li v-for="(exchange, name) in exchangePrices" :key="name">
                    <span class="exchange-name">{{ name }}</span>
                    <span class="exchange-price">${{ exchange.price.toFixed(2) }}</span>
```

```
                <a :href="exchange.trade_link" target="_blank" class="trade-button">Trade</a>
              </li>
            </ul>
          </div>
        </div>
      </div>
    </div>
   </section>
  </main>
  <footer class="footer">
   <p>Email: <a href="mailto:moneytalks_need_5@gmail.com">moneytalks_need_5@gmail.com</a></p>
   <div class="social-icons">
    <i class="fab fa-instagram"></i>
    <i class="fab fa-linkedin"></i>
   </div>
  </footer>
 </div>
</template>
<script setup>
import { ref } from 'vue'
import Item from './InvestmentsItem.vue'
import LineChart from './LineChart.vue'
import bitcoinIcon from '../assets/icons/bitcoin_icon.svg'
import ethereumIcon from '../assets/icons/ethereum_icon.svg'
import bnbIcon from '../assets/icons/bnb_icon.svg'
import solanaIcon from '../assets/icons/solana_icon.svg'
import xrpIcon from '../assets/icons/xrp_icon.svg'
import { investments } from "@/scripts/investments.js"
import axios from "axios";
const exchangePrices = ref({})
const selectedCrypto = ref('')
const crypto_prices = ref([])
const isLoading = ref(false)
const showAccountMenu = ref(false)
function toggleAccountMenu() {
 showAccountMenu.value = !showAccountMenu.value
}
async function logoutUser() {
 const refresh = localStorage.getItem('user-refresh');
 try {
  await axios.post('http://0.0.0.0:8000/api/logout/', { refresh }, {
   headers: {
    Authorization: `Bearer ${localStorage.getItem('user-token')}`
   }
  });
 } catch (err) {
  console.error('Logout error:', err);
 }
 localStorage.removeItem('user-token');
 localStorage.removeItem('user-refresh');
 router.push('/login');
}
async function handleCryptoSelect(symbol) {
 if (selectedCrypto.value === symbol) {
  selectedCrypto.value = ''
  return
 }
 isLoading.value = true
 selectedCrypto.value = symbol
 try {
  const result = await investments.getTradingByName(symbol)
  exchangePrices.value = result || {}
```

```
    const result1 = await investments.getPriceByName(symbol)
    crypto_prices.value = Array.isArray(result1) ? result1 : []
  } catch (error) {
    console.error('Error loading crypto data:', error)
  } finally {
    isLoading.value = false
  }
}
</script>

<style scoped>
body {
  font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
}
.page-wrapper {
  flex: 1;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  font-family: inherit;
}
a {
  font-family: "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
}
h2 {
  font-family: "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
  font-size: 24px;
}
.content {
  flex: 1;
}
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 40px 60px;
  max-width: 800px;
  margin: 50px auto;
  justify-items: center;
  align-items: center;
}
.grid > .item:nth-child(4) {
  grid-column: 1 / 2;
}
.grid > .item:nth-child(5) {
  grid-column: 3 / 4;
}
.footer {
  height: 60px;
  background: linear-gradient(to top, #1A7431CC 0%, #1A743100 100%);
  color: white;
  padding: 1rem 3rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
.footer p a {
  color: #dbfeb8;
  text-decoration: none;
  transition: color 0.2s;
}
.footer p a:hover {
  color: #d1fae5;
```

```
}
.social-icons {
  display: flex;
  gap: 1rem;
  font-size: 1.25rem;
}
.chart-section {
  max-width: 1200px;
  margin: 3rem auto;
  padding: 1rem;
  text-align: center;
  position: relative;
}
.chart-and-markets {
  display: flex;
  gap: 2rem;
  align-items: flex-start;
  justify-content: center;
  flex-wrap: wrap;
}
.chart-container {
  flex: 1 1 600px;
  z-index: 1;
}
.chart-description {
  background-color: #f3f4f6;
  padding: 1.5rem;
  border-radius: 12px;
  text-align: left;
  font-size: 1rem;
  line-height: 1.6;
  color: #111827;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.05);
  margin-top: 2rem;
}
.market-prices {
  flex: 0 0 220px;
  background-color: #ffffff;
  padding: 1.5rem;
  border-radius: 12px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.05);
  font-size: 0.95rem;
  z-index: 1;
}

.market-prices h3 {
  margin-bottom: 1rem;
  font-size: 1.15rem;
  color: #065f46;
  text-align: center;
}
.market-prices ul {
  list-style: none;
  padding: 0;
  margin: 0;
}
.market-prices li {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1rem;
  gap: 0.5rem;
```

```
}
.exchange-name {
  font-weight: 600;
  flex: 1;
}
.exchange-price {
  color: #065f46;
}
.trade-button {
  background-color: #065f46;
  color: white;
  padding: 0.25rem 0.6rem;
  border-radius: 4px;
  text-decoration: none;
  font-size: 0.85rem;
  transition: background-color 0.2s;
}
.trade-button:hover {
  background-color: #047857;
}
.spinner-overlay {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(255, 255, 255, 0.6);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 10;
}
.spinner-wrapper {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 0.75rem;
}
.spinner {
  border: 4px solid #d1fae5;
  border-top: 4px solid #065f46;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  animation: spin 0.8s linear infinite;
}

.spinner-text {
  font-size: 1rem;
  color: #065f46;
  font-weight: 500;
  text-align: center;
}
.dots span {
  opacity: 0;
  animation: dotsAnimation 1.5s infinite;
  display: inline-block;
}
.dots span:nth-child(1) {
  animation-delay: 0s;
}
.dots span:nth-child(2) {
```

```
  animation-delay: 0.2s;
}
.dots span:nth-child(3) {
  animation-delay: 0.4s;
}
@keyframes spin {
  to {
    transform: rotate(360deg);
  }
}
@keyframes dotsAnimation {
  0% {
    opacity: 0;
  }
  30% {
    opacity: 1;
  }
  100% {
    opacity: 0;
  }
}
</style>
```