НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

_____ Ігор БОЛБОТ
(підпис)

«___» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ Белла ГОЛУБ
(підпис)

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Кросплатформне програмне забезпечення онлайн банкінгу для приватного користувача»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ (підпис) **Ганна ВАЙГАНГ**

Керівник бакалаврської кваліфікаційної роботи
науковий ступінь та вчене звання _____ (підпис) **Віктор КІРІЧЕНКО**

Виконав _____ (підпис) **Ярослав ЯКОВЧЕНКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

ЯКОВЧЕНКО ЯРОСЛАВУ ОЛЕКСАНДРОВИЧУ

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Кросплатформне програмне забезпечення онлайн банкінгу для приватного користувач»

затверджена наказом від «19» грудня 2025 р. № 3196 «С»
Термін подання завершеної роботи на кафедру «___» _____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):
_____
Перелік питань, що підлягають дослідженню:
1.Аналіз та ТЗ. 2. Проєктування архітектури. 3. Розробка фронтенду(React Tailwind). 4.Розробка бекенду(Python(FastAPI). 5.Реалізація переказу функціоналу переказукоштів.
6.Реалізація функціоналу кредитування. 7. Тестування та розгортання
Перелік графічних документів (за необхідності).
_____

Дата видачі завдання «___» _____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ **Віктор КІРІЧЕНКО**
(підпис)
науковий ступінь та вчене звання

Виконав _____ **Ярослав ЯКОВЧЕНКО**
(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬВСТУП

API — Application Programming Interface – програмний інтерфейс застосунку

ACID — Atomicity, Consistency, Isolation, Durability – властивості транзакцій

БД — База даних

ER-діаграма — Entity-Relationship diagram – діаграма «сутність-зв'язок»

FastAPI — Веб-фреймворк для побудови API на Python

FinTech — Financial Technology – фінансові технології

HTTP — Hypertext Transfer Protocol

IBAN — International Bank Account Number

JWT — JSON Web Token – стандарт передачі авторизаційних даних

KYC — Know Your Customer – ідентифікація клієнта

ORM — Object-Relational Mapping – об'єктно-реляційне відображення

PCI DSS — Payment Card Industry Data Security Standard

PostgreSQL — Об'єктно-реляційна система управління базами даних

React.js — JavaScript-бібліотека для побудови інтерфейсів

REST — Representational State Transfer – архітектурний стиль API

SPA — Single Page Application – односторінковий застосунок

SQL — Structured Query Language

СУБД — Система управління базами даних

TLS — Transport Layer Security – протокол захищеної передачі даних

UML — Unified Modeling Language – уніфікована мова моделювання

URL — Uniform Resource Locator

UUID — Universally Unique Identifier

Сучасна банківська індустрія зазнає кардинальних трансформацій під впливом цифровізації. Клієнти все частіше надають перевагу дистанційному управлінню фінансами через мобільні та веб-застосунки, що стимулює фінансові установи активно розвивати цифровий банкінг. За даними НБУ, частка безготівкових платежів в Україні щороку зростає на 15–20%, а попит на зручні FinTech-рішення неухильно підвищується.

Актуальність роботи обумовлена нагальною потребою у розробці надійного, безпечного та зручного веб-застосунку для управління банківськими рахунками, що відповідає вимогам сучасного ринку, міжнародним стандартам безпеки (PCI DSS, ISO 27001) та банківського законодавства України. Особливої ваги набуває рішення, що може функціонувати на різноманітних платформах і пристроях без необхідності встановлення програмного забезпечення.

Метою роботи є проєктування та розробка кросплатформного програмного забезпечення онлайн банкінгу для приватного користувача – системи VAULT BANK, що забезпечує повний спектр базових банківських операцій через веб-інтерфейс.

Для досягнення мети вирішуються такі завдання:

- аналіз предметної області онлайн-банкінгу та існуючих аналогічних рішень;
- формалізація вимог і проєктування архітектури системи;
- розробка логічної та фізичної моделей бази даних;
- реалізація клієнтської частини на React.js з адаптивним дизайном;
- реалізація серверної частини на FastAPI з REST API та JWT-автентифікацією;
- інтеграція з PostgreSQL та Telegram Bot API;
- тестування та розгортання системи на віддаленому хостингу.

Об'єктом дослідження є процеси автоматизації банківського обслуговування приватних клієнтів засобами веб-технологій.

Предметом дослідження є методи та інструменти розробки кросплатформних FinTech-застосунків із забезпеченням безпеки фінансових транзакцій.

У роботі використовуються такі методи та технології: компонентна розробка інтерфейсу (React.js), асинхронне програмування (Python async/await), об'єктно-реляційне відображення (SQLAlchemy ORM), REST-архітектура, JWT-автентифікація, UML-моделювання.

Практичне значення роботи полягає у створенні повнофункціонального банківського застосунку, розгорнутого на публічному хостингу за адресою https://vaulty-bank-production-a78b.up.railway.app/, придатного для демонстрації та подальшого розвитку.

Структура записки. Пояснювальна записка містить 60 сторінок основного тексту, 4 розділи, 15 рисунків, 10 таблиць, список із 20 використаних джерел та 3 додатки. Перший розділ присвячений аналізу предметної області та постановці завдання. Другий розділ описує проєктування інформаційного забезпечення. Третій розділ містить опис архітектури та реалізації програмного забезпечення. Четвертий розділ охоплює тестування та розгортання системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної галузі онлайн-банкінгу

Онлайн-банкінг є одним із ключових напрямів сучасних фінансових технологій, що забезпечує клієнтам банку цілодобовий доступ до управління рахунками, виконання платежів та отримання фінансових послуг через мережу Інтернет. Стрімкий розвиток мобільних технологій та загальна цифровізація фінансового сектору зумовлюють зростання попиту на якісні та безпечні банківські застосунки.

Сучасні системи онлайн-банкінгу можна класифікувати за рядом ознак: за платформою (веб, мобільний, десктоп), за цільовою аудиторією (роздрібний, корпоративний, інвестиційний банкінг), за рівнем функціональності (базовий, розширений з кредитними інструментами), за архітектурним рішенням (монолітні, мікросервісні, гібридні).

Характерними рисами сучасних FinTech-рішень є: компонентна архітектура з чітким розмежуванням відповідальностей; безстанова (stateless) авторизація на основі JWT-токенів; ACID-транзакції для забезпечення цілісності фінансових даних; адаптивний дизайн для роботи на різноманітних пристроях; сповіщення в реальному часі про фінансові операції; відповідність міжнародним стандартам безпеки.

Система VAULT BANK розробляється як навчальна FinTech-платформа класу Internet Banking для фізичних осіб, що охоплює базові операції: управління рахунками, переказ коштів, оформлення кредитів, перегляд транзакційної історії.

1.2 Моделювання предметної області

Для формалізації предметної області використано апарат UML-моделювання, зокрема діаграми прецедентів, послідовності та активності.

1.2.1 Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) визначає межі системи та описує взаємодію акторів із функціональними можливостями. Основними акторами системи є Клієнт (фізична особа) та Адміністратор системи.

Клієнт має доступ до таких прецедентів: реєстрація облікового запису, авторизація, перегляд балансу, переказ коштів, оформлення кредиту, дострокове погашення кредиту, перегляд історії операцій, редагування профілю, звернення до підтримки. Адміністратор здійснює моніторинг транзакцій та управління обліковими записами.

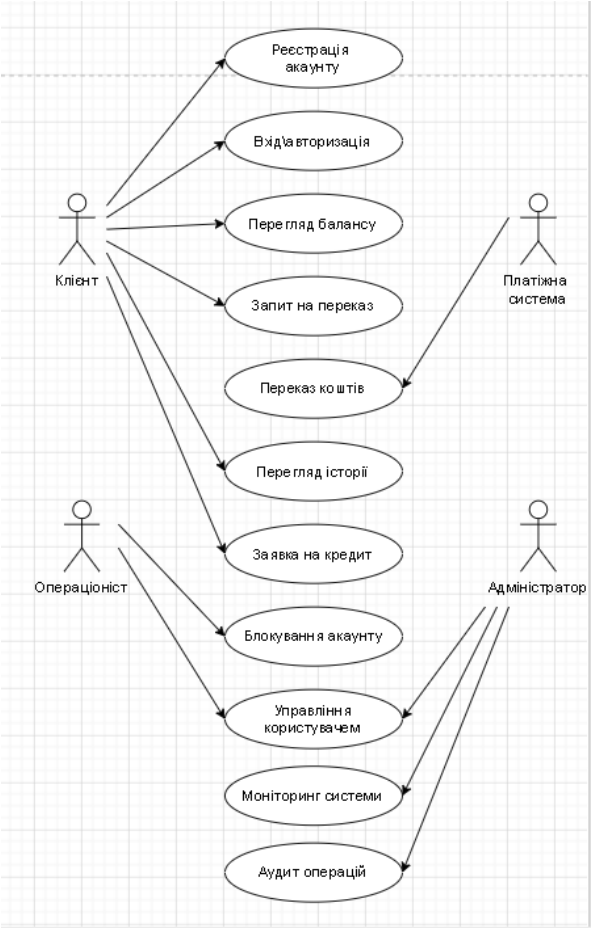


Рис. 1 Діаграма прецедентів системи VAULT BANK

1.2.2 Діаграма послідовності

Діаграма послідовності ілюструє взаємодію об'єктів системи в часі для ключових сценаріїв. Розглянемо сценарій «Переказ коштів»: Клієнт вводить реквізити отримувача та суму у формі. Компонент TransferForm передає дані до фронтенд-валідації. При успішній валідації генерується POST-запит до ендпоінту /api/transfer. API Gateway перевіряє JWT-токен та маршрутизує запит до TransactionService. Сервіс перевіряє достатність коштів, оновлює баланси обох рахунків у межах однієї транзакції, зберігає запис у таблиці bank_transaction та надсилає Telegram-сповіщення.

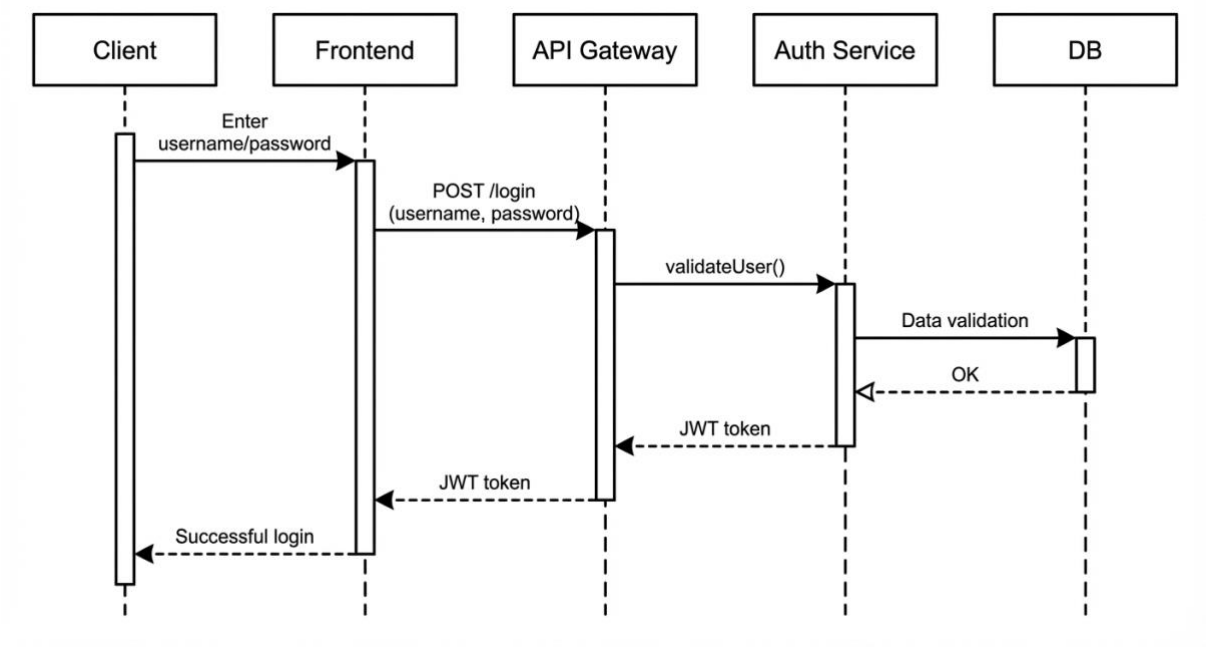


Рис. 2 Діаграма послідовності для сценарію «Переказ коштів»

1.2.3 Діаграма активності

Діаграма активності описує потік управління для алгоритмів авторизації та переказу коштів. Алгоритм авторизації починається з введення логіну та пароля, виконання фронтенд-валідації, відправки POST-запиту, перевірки даних на сервері (bcrypt-хешування пароля), генерації JWT-токена у разі успіху або повернення помилки 401 у разі невдачі.

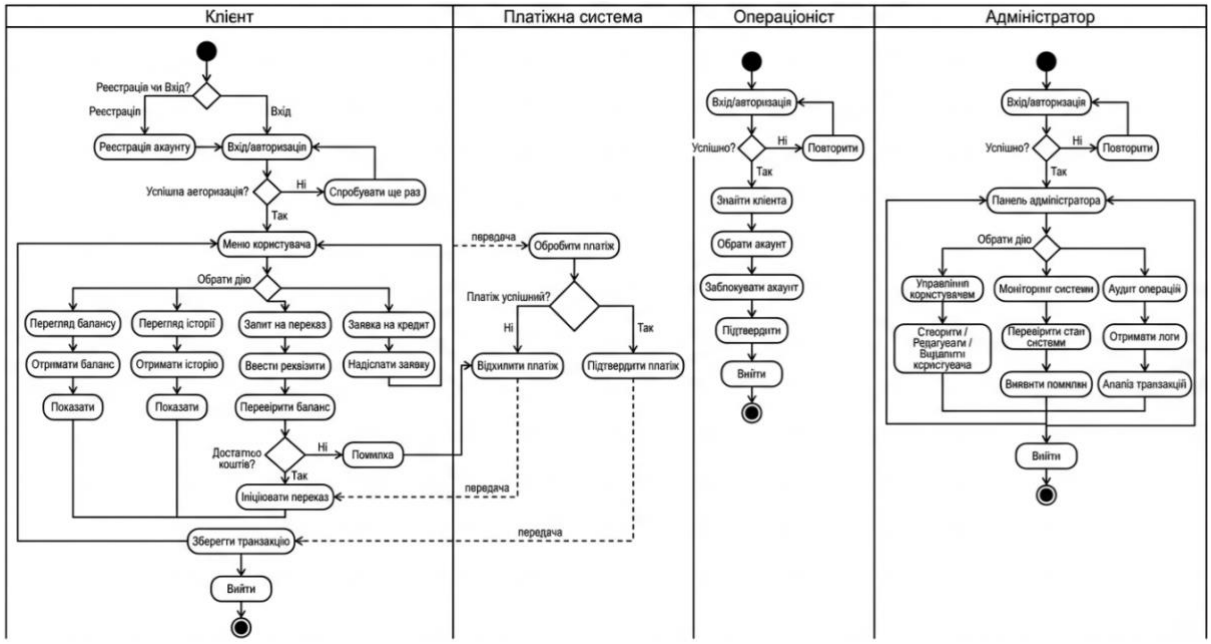


Рис. 3 Діаграма активності алгоритму авторизації

1.3 Огляд існуючих аналогічних систем

Аналіз ринку виявив декілька категорій конкурентних рішень у сегменті онлайн-банкінгу для фізичних осіб.

Система	**Переваги**	**Недоліки**
Приват24(ПриватБанк)	Широкий функціонал, велика база користувачів, інтеграція з державними сервісами	Складний інтерфейс для нових користувачів, залежність від інфраструктури одного банку
Monobank	Простий та інтуїтивний інтерфейс, швидкість операцій, cashback-програма	Обмежений функціонал для складних операцій, відсутність веб-версії з повним функціоналом

NEOBANK(ПУМБ)	Мультивалютність, зручна карта дебетова, кешбек	Менша мережа банкоматів, обмежена кредитна лінійка
Revolut	Мультивалютність, низькі комісії, криптовалюта	Не регульований НБУ повністю, обмеження для українських карток

На підставі проведеного аналізу визначено конкурентні переваги системи VAULT BANK: відкрита архітектура для навчальних цілей, відсутність залежності від конкретного банку, можливість локального розгортання, інтеграція з Telegram Bot API для сповіщень у реальному часі.

1.4 Постановка завдання

На підставі аналізу предметної області сформовано технічне завдання на розробку системи VAULT BANK.

1.4.1 Загальні відомості

Система призначена для надання клієнтам цифрового доступу до фінансових послуг. Мета: автоматизація базових банківських операцій та надання зручного цифрового інтерфейсу для приватних клієнтів.

1.4.2 Функціональні вимоги

Модуль авторизації та реєстрації:

- реєстрація нового акаунту з автоматичним створенням дебетового рахунку та картки;
- авторизація за логіном та паролем із JWT-токеном;
- автоматичне блокування акаунту після невдалих спроб входу;
- відновлення доступу.

Модуль управління рахунками:

- перегляд поточного балансу та номера картки;

– перегляд та фільтрація історії операцій (останні 50 транзакцій);
– копіювання номера картки одним кліком.

Модуль транзакцій:

– переказ коштів за логіном або номером картки отримувача;
– перевірка достатності балансу перед переказом;
– підтвердження операцій та Telegram-сповіщення.

Кредитний модуль:

– оформлення кредиту із зазначенням суми та терміну;
– перегляд таймера зворотного відліку до кінця терміну кредиту;
– дострокове погашення кредиту.

Модуль підтримки:

– звернення до служби підтримки через форму;
– редагування профілю (email, телефон, пароль);
– перемикання між темною та світлою темою інтерфейсу.

1.4.3 Нефункціональні вимоги

Характеристика	**Вимога**	**Метрика**
Доступність	99.9% uptime	Не більше 8.7 год простою на рік
Продуктивність	Відповідь API	< 200 мс для 95% запитів
Масштабованість	До 10 000 одночасних користувачів	Горизонтальне масштабування
Безпека	JWT + bcrypt-хешування	Аудит всіх операцій
Сумісність	Кросбраузерність	Chrome 90+, Firefox 88+, Safari 14+

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Абстракції предметної області

Відповідно до методології об'єктно-орієнтованого аналізу та проєктування (ООАП) виокремлено три категорії класів: сутнісні, граничні та класи керування.

2.1.1 Основні сутнісні класи

Нижче наведено основні абстракції предметної області системи VAULT BANK.

Абстракція	**Ключові властивості**	**Обов'язки**
User(Користувач)	userId, username, email, phone, passwordHash, status, createdAt	Зберігати дані клієнта; керувати станом акаунту
Account(Рахунок)	accountId, userId, IBAN, currency, balance, accountType, status	Забезпечувати цілісність балансу; формувати виписки
Transaction(Транзакція)	transactionId, fromAccountId, toAccountId, amount, type, status, timestamp	Незмінний запис операції; підтримувати ACID-властивості
Loan(Кредит)	loanId, userId, amount, term, startDate, endDate, status	Управляти кредитним циклом; розраховувати таймер погашення

AuditLog(Аудит)	logId, userId, action, entityId, ipAddress, timestamp	Незмінний журнал дій; забезпечення безпеки

2.1.2 Граничні та керуючі класи

Клас	**Тип**	**Призначення**
LoginForm	Граничний (UI)	Форма авторизації; фронтенд-валідація
TransferForm	Граничний (UI)	Форма переказу коштів
AccountDashboard	Граничний (UI Screen)	Головний екран з балансом та операціями
APIGateway	Граничний (System Interface)	Єдина точка входу; маршрутизація; rate limiting
AuthService	Керуючий	Автентифікація; JWT; управління сесіями
TransactionService	Керуючий	Обробка переказів; перевірка балансу; ACID
LoanService	Керуючий	Управління кредитами; розрахунок таймера
UserService	Керуючий	Реєстрація; управління профілем

2.1.3 Діаграма класів

На основі виокремлених абстракцій побудовано діаграму класів, що відображає асоціації між сутностями системи. Зв'язки між класами:

– User 1 — * Account: один користувач може мати декілька рахунків (композиція);
– Account 1 — * Transaction: один рахунок бере участь у багатьох транзакціях (асоціація);
– User 1 — * Loan: один користувач може мати декілька кредитів (асоціація);
– User 1 — * AuditLog: всі дії фіксуються в журналі (залежність).
–

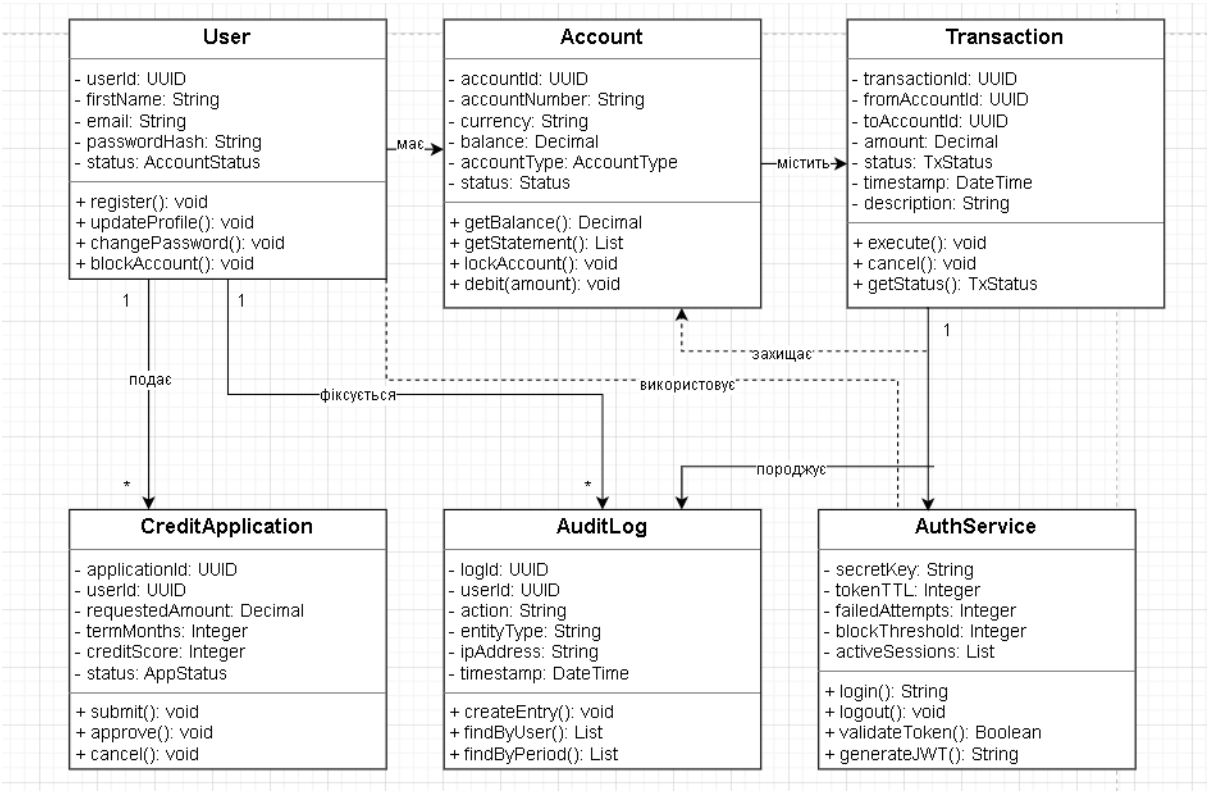


Рис. 4 Діаграма класів системи VAULT BANK

2.1.4 Кооперації

На основі діаграми класів побудовано три кооперації, що описують ключові бізнес-сценарії: «Заявка на кредит» (LoanService, Account, Loan, User), «Блокування акаунту» (AuthService, User, AuditLog) та «Переказ коштів» (TransactionService, Account, Transaction).

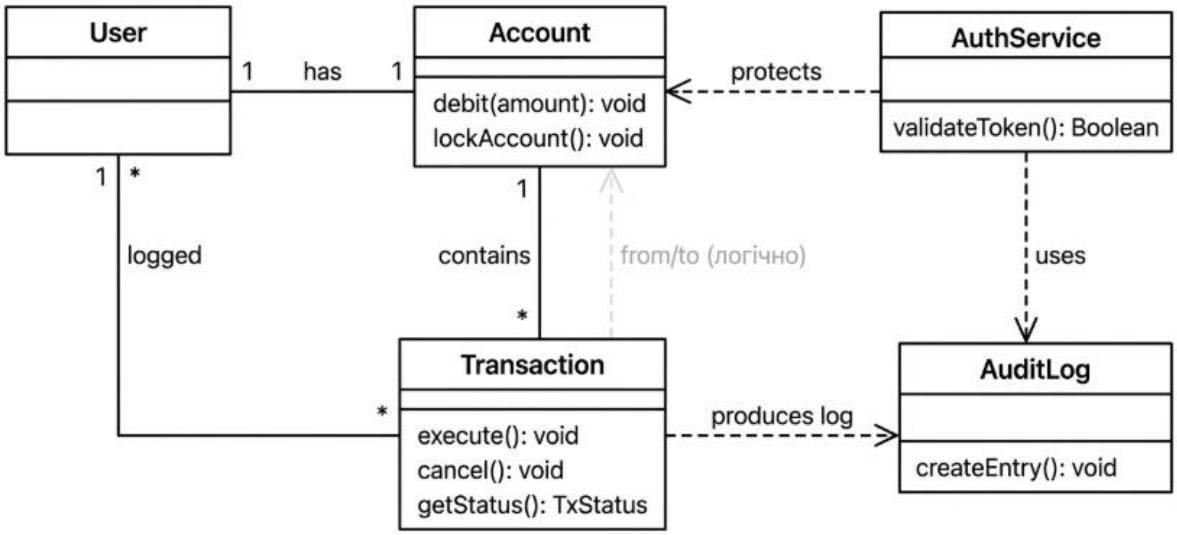


Рис. 5 Кооперація «Переказ коштів»

2.2 Логічна модель даних

Логічна модель даних описує структуру інформаційного забезпечення системи у вигляді ER-діаграми (сутність-зв'язок). Основні сутності та їх зв'язки:

Сутність	**Тип**	**Основні атрибути**
users	Сильна	id (PK), username, email, phone, password_hash, created_at
accounts	Сильна	id (PK), user_id (FK), card_number, balance, currency
bank_transaction	Асоціативна	id (PK), from_account_id (FK), to_account_id (FK), amount, type, created_at
loans	Сильна	id (PK), user_id (FK), amount, term_months,

		start_date, end_date, status
support_requests	Сильна	id (PK), user_id (FK), message, created_at

Тип зв'язків: між users і accounts – «один до багатьох» (1:N); між accounts і bank_transaction – «один до багатьох» (1:N) для обох рахунків (відправник і отримувач); між users і loans – «один до багатьох» (1:N).

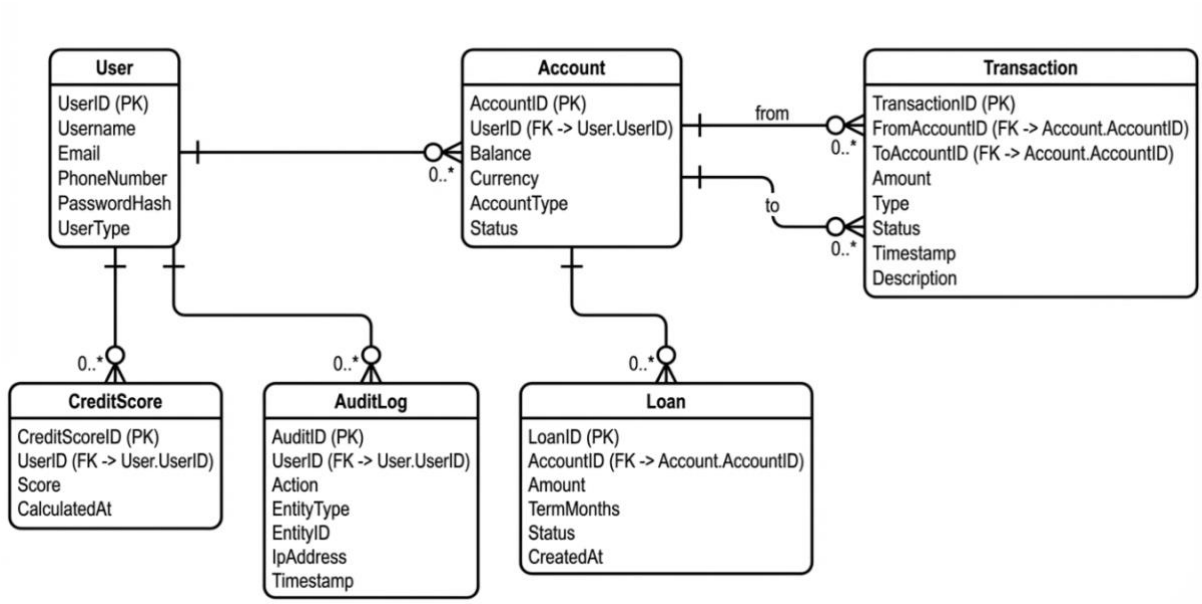


Рис. 6 ER-діаграма бази даних системи VAULT BANK

2.3 Вибір системи управління базами даних

Для реалізації інформаційного забезпечення системи VAULT BANK обрано PostgreSQL версії 14+. Обґрунтування вибору здійснено за трьома критичними для FinTech-систем групами критеріїв.

Надійність і відповідність стандартам. PostgreSQL підтримує повну модель ACID-транзакцій, що є фундаментальною вимогою для фінансових систем, де некоректне оновлення балансів може призвести до критичних втрат. СУБД відповідає вимогам банківського законодавства та міжнародним стандартам безпеки.

Масштабованість і продуктивність. PostgreSQL добре інтегрується з мікросервісною архітектурою, підтримує реплікацію та контейнеризацію через Docker, що дозволяє ефективно масштабувати систему під навантаженням.

Гнучкість, безпека та інтеграція. СУБД підтримує рольову модель доступу, TLS-шифрування, тип NUMERIC(19,2) для точних фінансових розрахунків, TIMESTAMPTZ для коректної роботи таймера кредиту, а також легко інтегрується з ORM SQLAlchemy та фреймворком FastAPI.

2.4 Фізична структура бази даних

Перехід від логічної до фізичної моделі здійснюється з використанням мови SQL. Основні таблиці бази даних наведено нижче.

Таблиця users зберігає дані облікових записів користувачів. Первинним ключем є поле id типу UUID. Поле password_hash зберігає хеш пароля, згенерований алгоритмом bcrypt. Поле created_at фіксує дату реєстрації з часовим поясом.

Таблиця accounts містить інформацію про банківські рахунки. Поле balance типу NUMERIC(19,2) забезпечує точність фінансових розрахунків без похибок округлення. Зовнішній ключ user_id пов'язує рахунок із власником.

Таблиця bank_transaction є центральним журналом фінансових операцій. Обидва поля from_account_id та to_account_id є зовнішніми ключами до таблиці accounts. Поле type може набувати значень 'transfer', 'deposit', 'withdrawal'. Записи у цій таблиці є незмінними після створення.

Таблиця loans зберігає інформацію про кредитні договори. Поля start_date та end_date типу TIMESTAMPTZ використовуються для розрахунку таймера зворотного відліку на клієнтській частині.

Рис. 7 Схема фізичної структури бази даних

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Система VAULT BANK реалізована за клієнт-серверною архітектурою з чітким розподілом відповідальностей між трьома шарами.

3.1.1 Клієнтська частина (Frontend)

Клієнтська частина реалізована як SPA (Single Page Application) на React.js 18+. Виконує такі функції: відображення інтерфейсу користувача, надсилання REST API запитів до сервера, управління локальним станом застосунку (баланс, кредит, тема), валідація форм на стороні клієнта.

3.1.2 Серверна частина (Backend)

Серверна частина реалізована на FastAPI (Python). Відповідає за: обробку HTTP-запитів від клієнта, валідацію вхідних даних через Pydantic-схеми, реалізацію бізнес-логіки (кредитування, перекази, автентифікація), взаємодію з базою даних через SQLAlchemy ORM, надсилання сповіщень через Telegram Bot API.

3.1.3 Діаграма пакетів

Діаграма пакетів відображає архітектуру системи: клієнтська частина містить пакети ClientGUI (взаємодія з користувачем) та ClientNetwork (мережева комунікація). Серверна частина включає ServerBusinessLogic (основна логіка), RequestDB (робота з БД), ServerNetwork (прийом і відправка даних) та Utils (спільні функції).

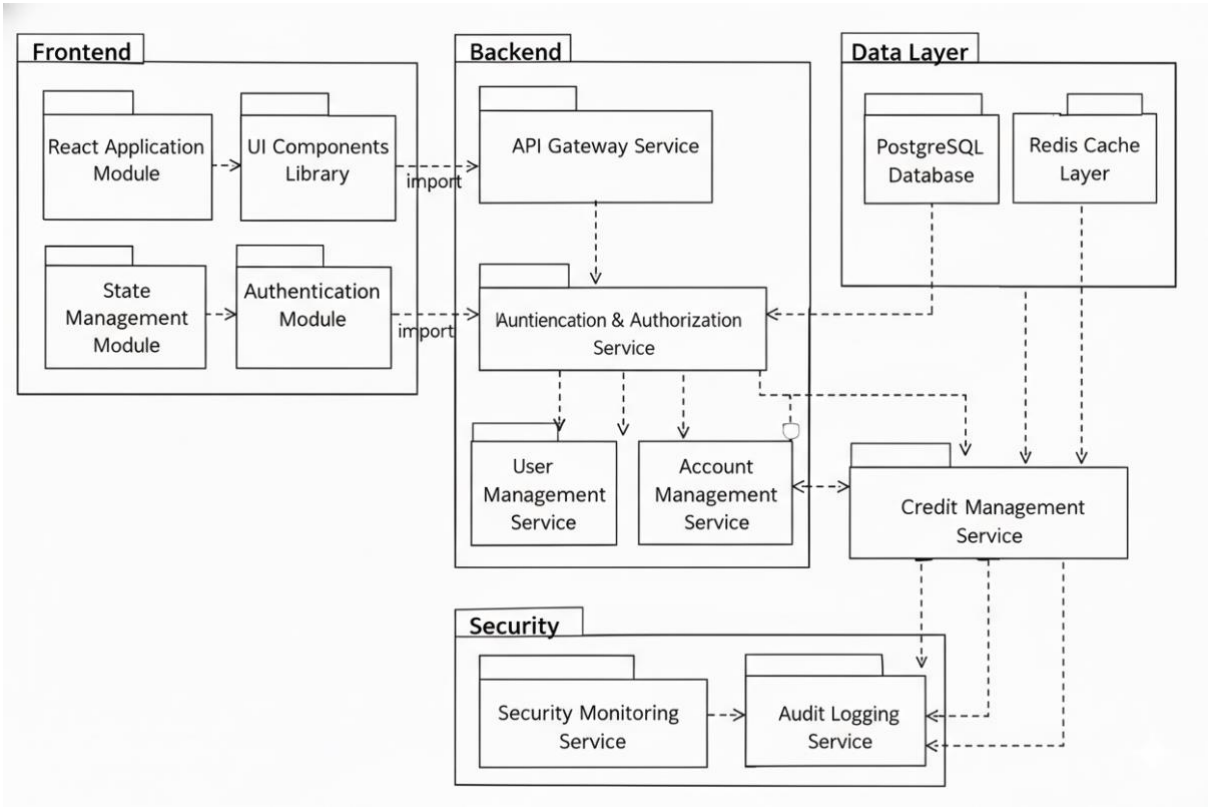
1. 

Рис. 8 Діаграма пакетів системи VAULT BANK

3.2 Структура та взаємодія об'єктів

Структура програмного забезпечення описується через співвідношення класів і компонентів. Клас AuthService описує логіку автентифікації як абстракцію в коді. Компонент auth-service – це вже цілий мікросервіс на FastAPI з власними маршрутами та залежностями, готовий до запуску в Docker-контейнері.

Між поняттями клас, компонент, артефакт та вузол існує чітка вертикаль абстракції: клас → компонент → артефакт → вузол. Клас живе в пам'яті розробника та IDE, компонент – у структурі проєкту, артефакт – на файловій системі, вузол – у реальній інфраструктурі.

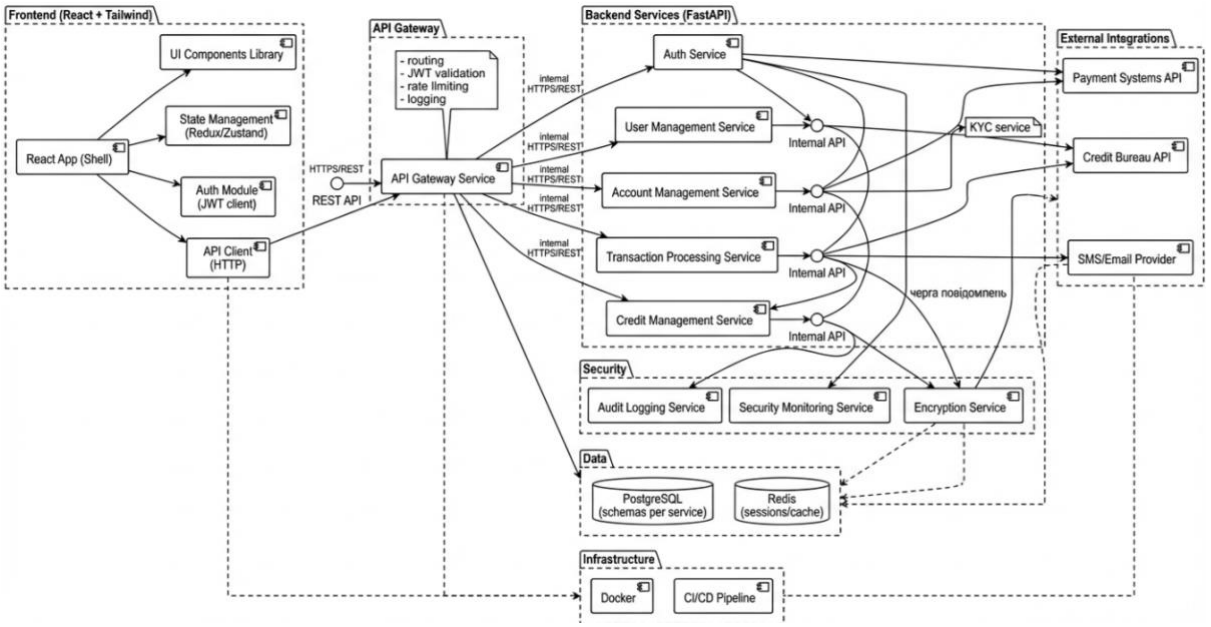


Рис. 9 Діаграма компонентів системи VAULT BANK

3.3 Організаційна та компонентна структура системи

Файлова структура проєкту організована відповідно до принципу розділення відповідальностей. Бекенд (Python/FastAPI) містить такі модулі: main.py (точка входу, ініціалізація FastAPI), auth.py (маршрути авторизації та реєстрації), transactions.py (маршрути переказів), loans.py (маршрути кредитного модуля), models.py (ORM-моделі SQLAlchemy), schemas.py (Pydantic-схеми валідації), database.py (підключення до PostgreSQL), telegram_bot.py (відправка Telegram-сповіщень).

Фронтенд (React.js) побудований за компонентною архітектурою: App.jsx (кореневий компонент, управління станом автентифікації), Auth.jsx (компонент авторизації та реєстрації), Dashboard.jsx (головний екран після входу), Transfer.jsx (форма переказу коштів), Loan.jsx (кредитний модуль з таймером), Support.jsx (форма звернення до підтримки), Profile.jsx (редагування профілю).

3.3.1 Топологія розгортання

Діаграма розміщення описує фізичну топологію системи. Клієнтський браузер взаємодіє з фронтендом (React.js SPA, розгорнутий на Railway).

Фронтенд звертається до бекенду (FastAPI, Railway). Бекенд взаємодіє з PostgreSQL (Railway managed DB) та зовнішнім Telegram Bot API.

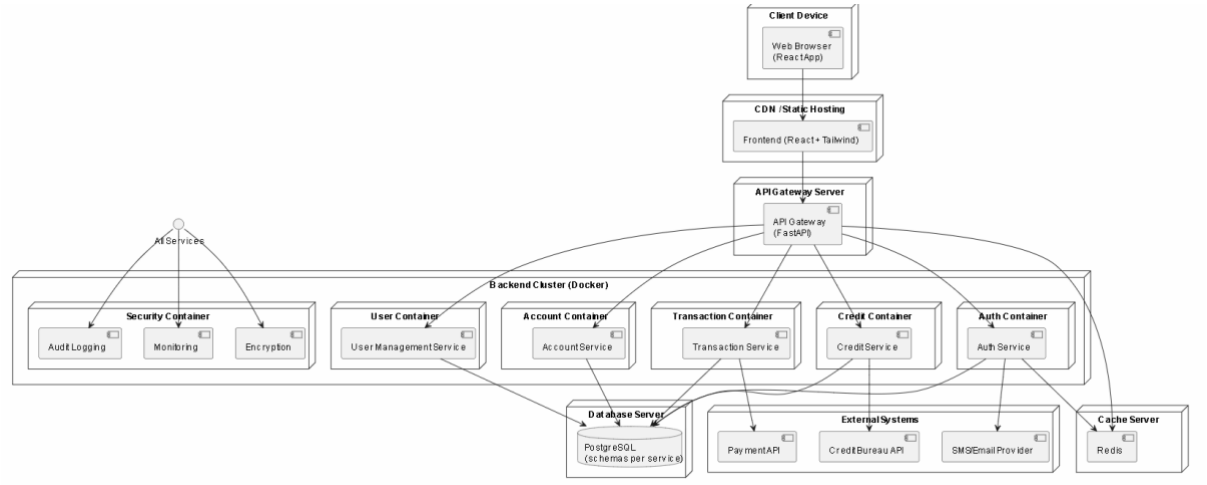


Рис. 10 Діаграма розміщення системи VAULT BANK

3.4 Вибір інструментарію розробки

Вибір інструментальних засобів обумовлений специфікою FinTech-застосунку та технічними вимогами до системи.

Рівень	**Технологія**	**Версія**	**Обґрунтування**
Frontend	React.js	18+	Компонентна архітектура; реактивне оновлення стану без перезавантаження сторінки
Стилізація	Tailwind CSS	3+	Утилітарний підхід; швидка побудова адаптивних інтерфейсів без

			окремих CSS-файлів
Типографіка	Oxanium / Space Mono	–	Геометричний моноширинний шрифт, характерний для фінансових продуктів
Backend	FastAPI (Python)	0.100+	Автоматична валідація через Pydantic; вбудована OpenAPI-документація; async/await
ORM	SQLAlchemy	2.0+	Зручна абстракція над SQL; підтримка міграцій через Alembic
База даних	PostgreSQL	14+	ACID-транзакції; NUMERIC(19,2) для точних фінансових розрахунків; TIMESTAMPTZ
Автентифікація	JWT (python-jose)	–	Stateless авторизація;

			відсутність необхідності зберігати сесії на сервері
Хешування	bcrypt (passlib)	–	Надійне хешування паролів; стійкість до brute-force атак
Сповіщення	Telegram Bot API	–	Миттєві push-сповіщення про операції в реальному часі
Розгортання	Railway	–	PaaS-платформа з підтримкою PostgreSQL; простота налаштування CI/CD

3.5 Алгоритмізація та програмування модулів

Нижче описано реалізацію ключових алгоритмів системи.

3.5.1 Алгоритм авторизації

Функція handleLogin виконується при натисканні кнопки «Увійти». Спочатку виконується клієнтська валідація – перевірка заповненості полів. При успішній валідації відправляється POST-запит на ендпоінт /api/auth/login. У разі отримання JWT-токена він зберігається в localStorage та виконується перехід на головний екран.

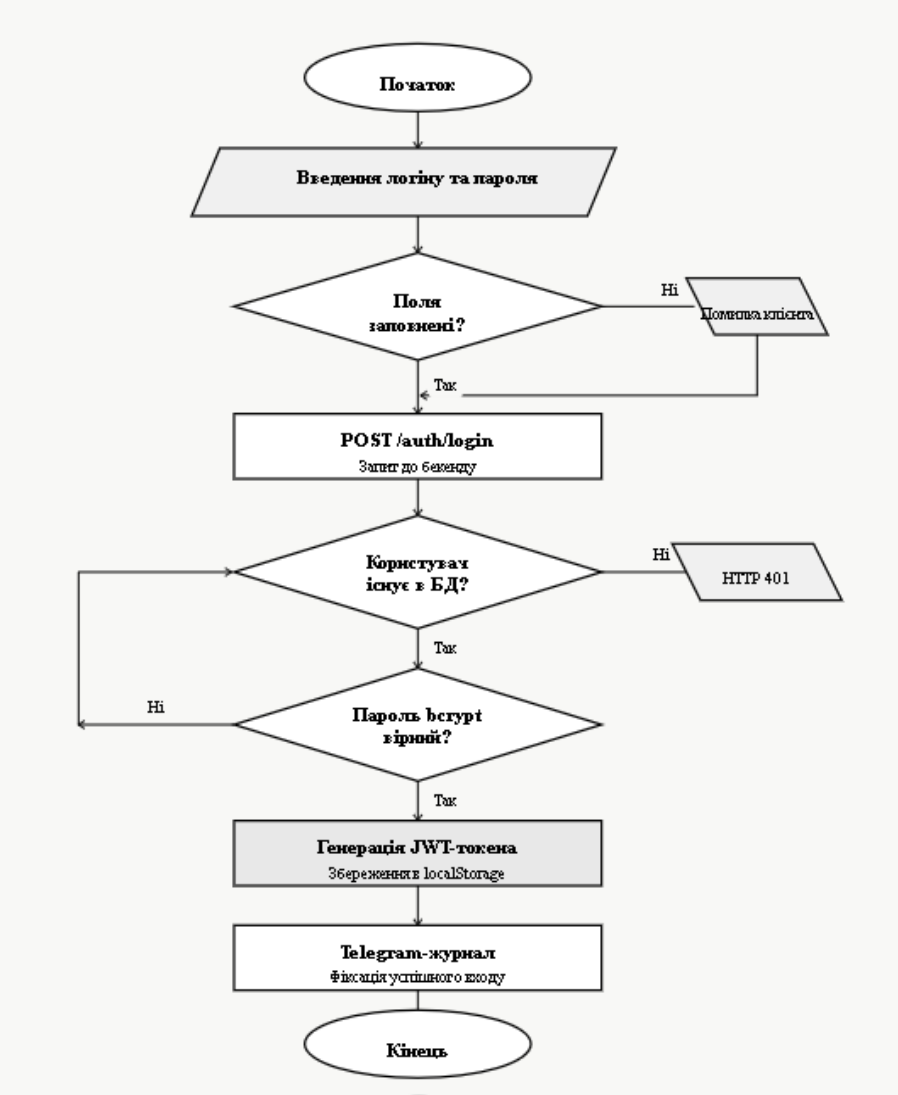


Рис. 11 Блок-схема алгоритму авторизації

3.5.2 Алгоритм переказу коштів

Алгоритм переказу реалізований у функції handleTransfer. Після авторизації користувач обирає режим пошуку отримувача (за логіном або номером картки). Система знаходить рахунок отримувача та перевіряє, що він відмінний від рахунку відправника. Далі перевіряється достатність коштів. При виконанні всіх умов баланси обох рахунків оновлюються в межах однієї SQL-транзакції, створюється запис у таблиці bank_transaction, і надсилається

Telegram-сповіщення.

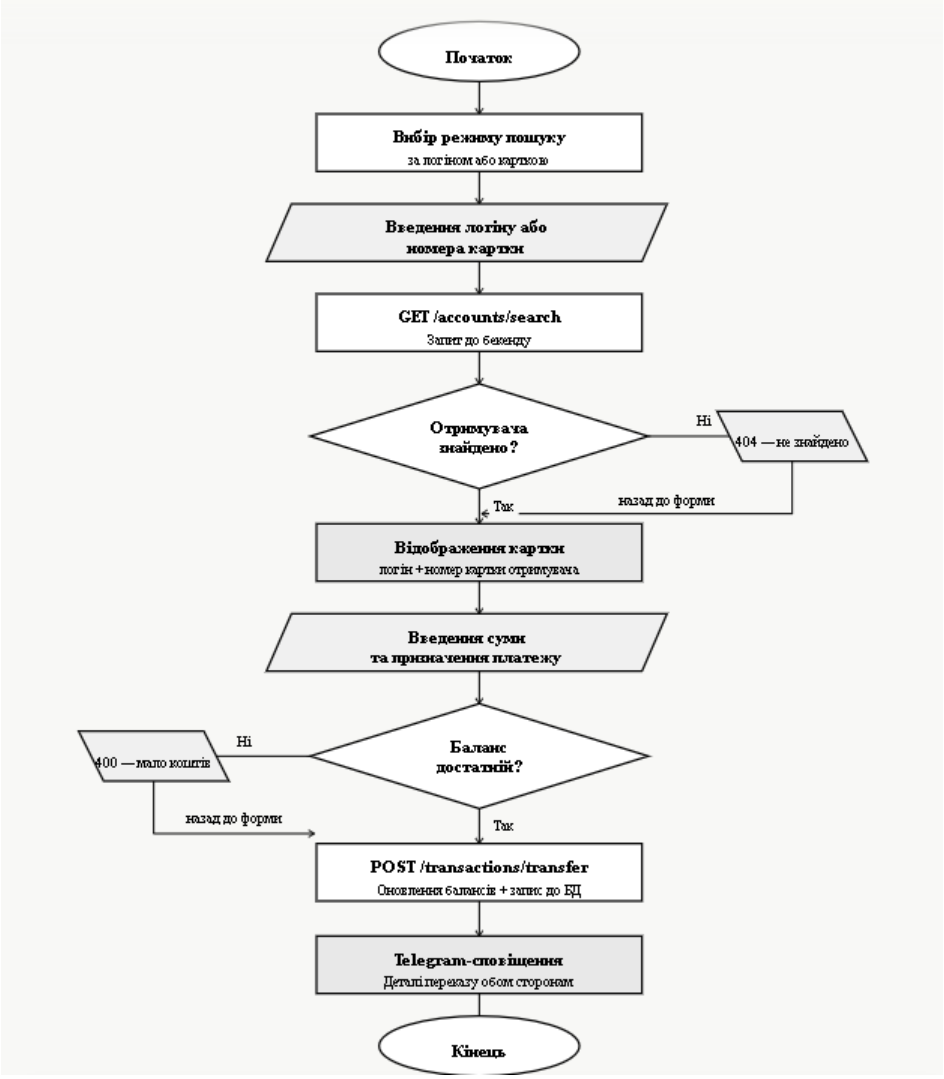


Рис. 12 Блок-схема алгоритму переказу коштів

3.5.3 Алгоритм кредитного модуля

Алгоритм оформлення кредиту включає такі кроки: перевірка наявності активного кредиту (система дозволяє лише один активний кредит), валідація суми та терміну, нарахування кредитних коштів на рахунок, збереження запису в таблиці loans із зазначенням start_date та end_date, запуск таймера зворотного відліку на фронтенді.

Таймер реалізований через хук useEffect, що кожну секунду обчислює різницю між поточним часом та end_date кредиту і оновлює відображення у форматі «ДД:ГГ:ХХ:СС».

3.5.4 Демонстрація інтерфейсу

Інтерфейс застосунку VAULT BANK реалізований у стилі dark fintech з власною дизайн-системою: жовтий акцент (#F5C400) для ключових елементів, червоний (#D62828) для помилок та попереджень, темний фон (#0A0A0A). Підтримується перемикання між темною та світлою темою зі збереженням вибору.

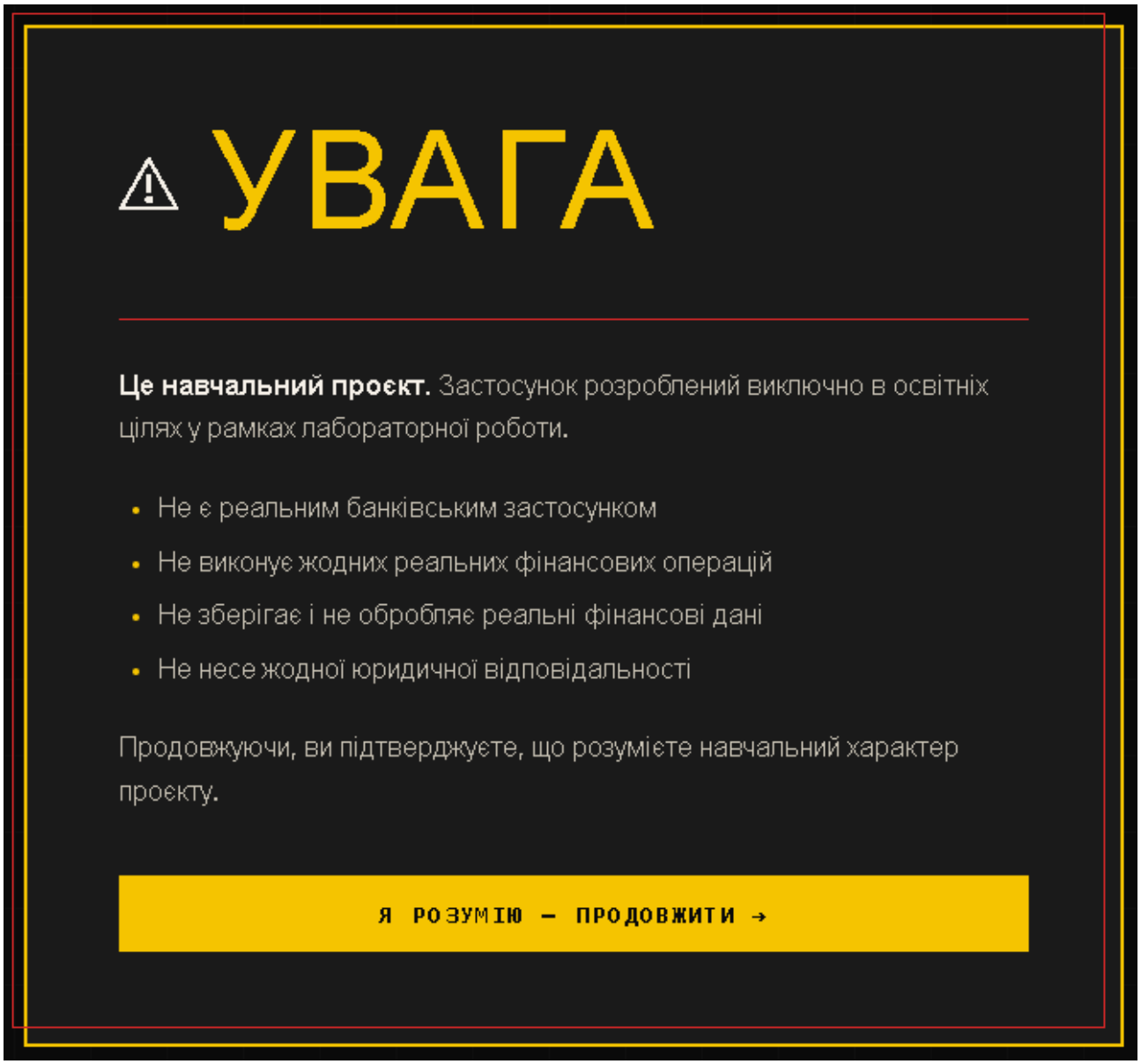


Рис. 13 Головний екран системи VAULT BANK

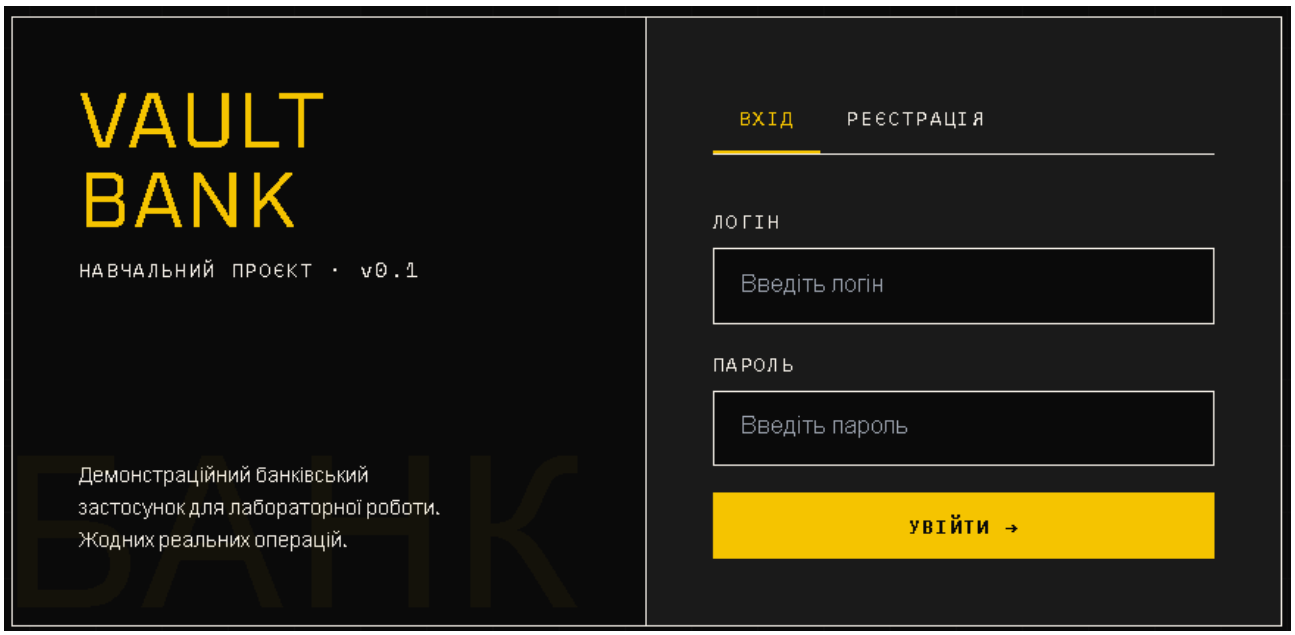


Рис. 14 Екран авторизації (вкладки «Вхід» та «Реєстрація»)

Рис. 15 Екран персонального акаунту користувача

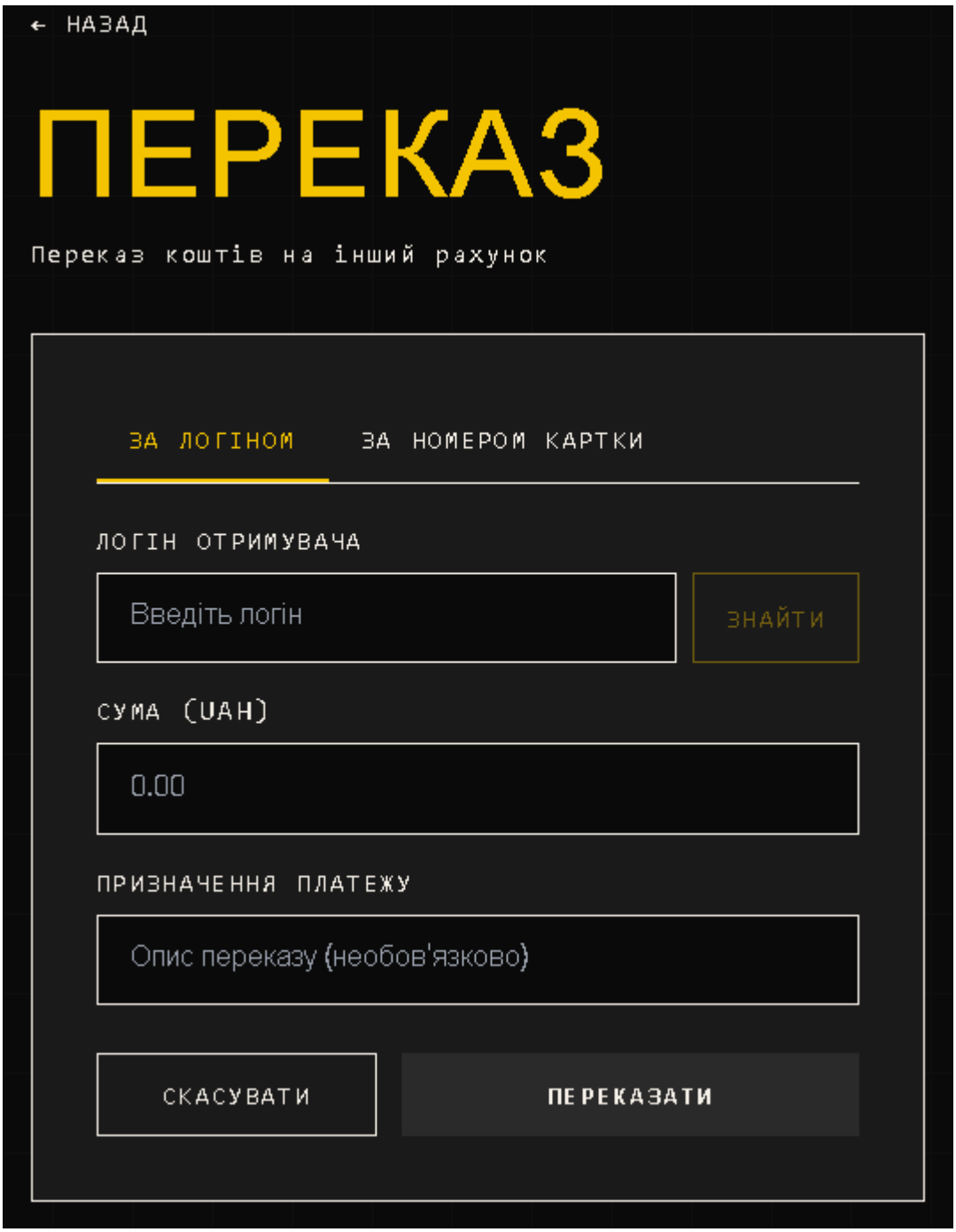


Рис. 16 Екран переказу коштів

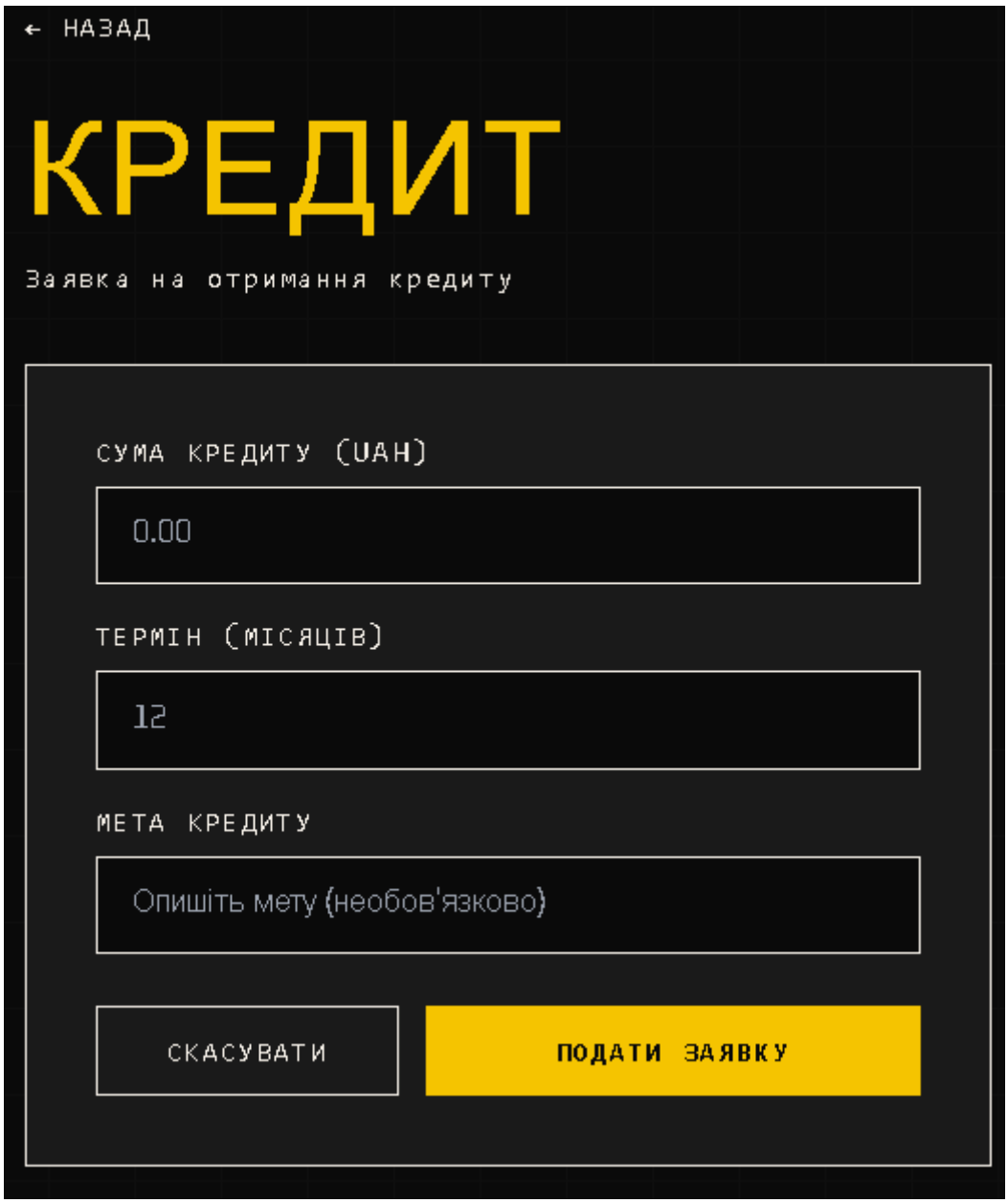


Рис. 17 Екран кредитування

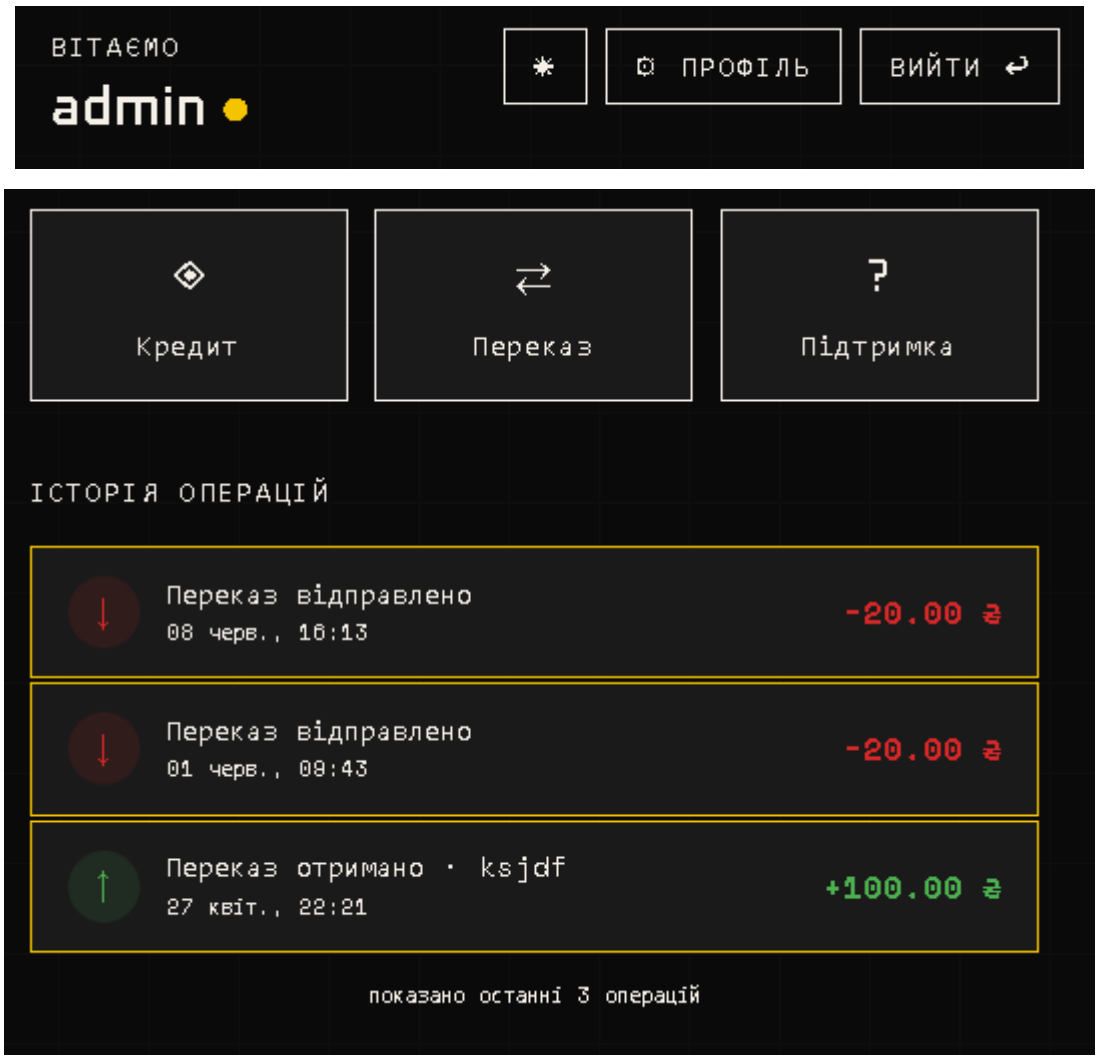


Рис. 18 Екран основних віджетів користувача

3.5.5 Кросплатформність системи

Кросплатформність системи VAULT BANK забезпечується веб-орієнтованою архітектурою на основі SPA (Single Page Application). Застосунок не потребує встановлення та однаково функціонує на будь-якому пристрої з сучасним браузером незалежно від операційної системи.

Адаптивність інтерфейсу реалізована засобами Tailwind CSS з використанням утилітарних класів для різних контрольних точок (breakpoints): sm (640px+), md (768px+), lg (1024px+). Завдяки цьому інтерфейс коректно відображається на

мобільних телефонах, планшетах та десктопних екранах без окремих версій застосунку.

Перевірку кросплатформності проведено на таких конфігураціях:

Платформа	Браузер	Версія	Результат
Windows 11	**Google Chrome**	**124**	**Пройдено**
Windows 11	**Microsoft Edge**	**124**	**Пройдено**
macOS Sonoma	**Safari**	**17**	**Пройдено**
Android 13	**Chrome Mobile**	**124**	**Пройдено**
iOS 17	**Safari Mobile**	**17**	**Пройдено**
Ubuntu 22.04	**Mozilla Firefox**	**125**	**Пройдено**

Усі функціональні модулі системи — авторизація, управління рахунками, переказ коштів, кредитний модуль з таймером, перегляд історії операцій — працюють коректно на всіх перевірених платформах. Перемикання між темною та світлою темою зі збереженням вибору також функціонує однаково на всіх пристроях.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Тестування системи

Тестування є невід'ємною складовою процесу розробки програмного забезпечення та спрямоване на виявлення дефектів і підтвердження відповідності системи встановленим вимогам.

4.1.1 Стратегія тестування

Для системи VAULT BANK застосовано комбінований підхід до тестування, що включає такі рівні:

- Модульне тестування (Unit Testing) – перевірка окремих функцій бізнес-логіки (валідація суми переказу, розрахунок таймера, хешування пароля);
- Інтеграційне тестування – перевірка взаємодії між компонентами (FastAPI ↔ PostgreSQL, Frontend ↔ API);
- Функціональне тестування – перевірка реалізації всіх функціональних вимог за сценаріями прецедентів;
- Тестування безпеки – перевірка захисту ендпоінтів (доступ без JWT-токена, SQL-ін'єкції, XSS).

4.1.2 Тест-кейси

№	Сценарій	Вхідні дані	Очікуваний результат	Результат
1	Авторизація (успішна)	Коректний логін та пароль	JWT-токен, перехід на Dashboard	Пройдено

2	Авторизація (невдала)	Неправильний пароль	Помилка 401, повідомлення про помилку	Пройдено
3	Переказ коштів (успішний)	Коректний логін отримувача, сума ≤ балансу	Баланси оновлено, запис у БД, Telegram-сповіщення	Пройдено
4	Переказ (недостатній баланс)	Сума > поточний баланс	Помилка 400, баланс незмінний	Пройдено
5	Оформлення кредиту	Сума 1000, термін 30 днів	Кредит нараховано, таймер запущено	Пройдено
6	Дострокове погашення	Активний кредит, баланс ≥ суми боргу	Кредит погашено, таймер скинуто	Пройдено
7	Захист ендпоінту	GET /api/profile без JWT	Помилка 401 Unauthorized	Пройдено
8	Реєстрація (дублікат)	Username, що вже існує	Помилка 400 з	Пройдено

			повідомлення м	

Всі тест-кейси пройдено успішно. Критичні уразливості не виявлено.

4.2 Вимоги до апаратного та програмного забезпечення

4.2.1 Вимоги до сервера

Компонент	**Мінімальні вимоги**	**Рекомендовані**
Процесор	Intel Core i3, 2 ядра, 2 ГГц	Intel Core i5+, 4 ядра
Оперативна пам'ять	4 ГБ	8 ГБ+
Дисковий простір	20 ГБ SSD	50 ГБ SSD
Мережа	100 Мбіт/с	1 Гбіт/с
ОС	Ubuntu 20.04 LTS або Docker-сумісна	Ubuntu 22.04 LTS

4.2.2 Програмні вимоги до сервера

Компонент	**Інструмент / Версія**	**Призначення**
Мова бекенду	Python 3.10+	Серверна логіка
Веб-фреймворк	FastAPI 0.100+	REST API
ORM	SQLAlchemy 2.0+	Робота з БД
База даних	PostgreSQL 14+	Збереження даних
HTTP клієнт	httpx	Telegram API
Автентифікація	python-jose, passlib	JWT та bcrypt

Фронтенд	React.js 18+ (Node.js 18+)	Інтерфейс
Стилізація	Tailwind CSS 3+	Оформлення UI

4.2.3 Вимоги до клієнта

Система є веб-орієнтованим SPA і не потребує встановлення на пристрої клієнта. Мінімальні вимоги:

– будь-який пристрій з доступом до мережі Інтернет (ПК, ноутбук, смартфон, планшет);
– один із підтримуваних браузерів: Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 14+, Opera 76+;
– роздільна здатність дисплею від 360×640 (підтримка мобільних пристроїв);
– швидкість Інтернет-з'єднання від 1 Мбіт/с.

4.3 Склад інсталяційного пакету та розгортання

4.3.1 Інсталяційний пакет

Для розгортання системи VAULT BANK підготовлено такий склад файлів:

Файл / Директорія	**Призначення**
backend/	Директорія серверної частини (Python/FastAPI)
backend/main.py	Точка входу FastAPI-застосунку
backend/requirements.txt	Список Python-залежностей
backend/models.py	ORM-моделі SQLAlchemy
backend/schemas.py	Pydantic-схеми валідації

backend/auth.py, transactions.py, loans.py	Маршрути API
frontend/	Директорія клієнтської частини (React.js)
frontend/package.json	Node.js залежності фронтенду
frontend/src/	Вихідний код React-компонентів
railway.toml / Procfile	Конфігурація розгортання на Railway

4.3.2 Розгортання на Railway

Система розгорнута на платформі Railway (PaaS) і доступна за адресою https://vaulty-bank-production-a78b.up.railway.app/. Розгортання здійснюється в три сервіси: PostgreSQL (managed database), Backend (Python FastAPI), Frontend (React.js SPA).

Процес розгортання включає: підключення репозиторію GitHub до Railway, налаштування змінних середовища (DATABASE_URL, JWT_SECRET, TELEGRAM_BOT_TOKEN, TELEGRAM_CHAT_ID), автоматичне збирання та запуск при кожному push до гілки main.

Рис. 16 Панель керування Railway з трьома розгорнутими сервісами

Система пройшла перевірку функціонування у production-середовищі. Всі функціональні вимоги виконано. Веб-застосунок доступний з будь-якого пристрою через мережу Інтернет.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі вирішено актуальне завдання розробки кросплатформного програмного забезпечення онлайн-банкінгу для приватного користувача – системи VAULT BANK.

У процесі виконання роботи досягнуто таких результатів:

- проведено аналіз предметної галузі онлайн-банкінгу, виявлено переваги та недоліки існуючих рішень (Приват24, Monobank, Revolut), визначено функціональні та нефункціональні вимоги до системи;
- побудовано комплекс UML-моделей предметної області: діаграми прецедентів, послідовності, активності, класів, кооперацій, пакетів, компонентів та розміщення;
- розроблено логічну та фізичну модель бази даних. Обрано PostgreSQL як СУБД, що відповідає вимогам ACID-транзакцій та точності фінансових розрахунків;
- реалізовано серверну частину на Python FastAPI з REST API, JWT-автентифікацією, bcrypt-хешуванням паролів та інтеграцією з Telegram Bot API для сповіщень;
- реалізовано клієнтську частину на React.js 18+ у стилі dark fintech з підтримкою темної та світлої теми, адаптивного дизайну для мобільних пристроїв;
- забезпечено кросплатформність системи засобами адаптивного дизайну на основі Tailwind CSS. Застосунок не потребує встановлення та коректно функціонує на Windows, macOS, Linux, Android та iOS у всіх сучасних браузерах без окремих версій для кожної платформи;
- система розгорнута на публічному хостингу Railway та доступна за адресою https://vaulty-bank-production-a78b.up.railway.app/;
- проведено комплексне тестування системи: функціональне, інтеграційне та безпекове. Всі 8 ключових тест-кейсів пройдено успішно.

Розроблена система забезпечує повний цикл базових банківських операцій: реєстрацію та авторизацію, управління рахунками, переказ коштів, оформлення та погашення кредитів, перегляд транзакційної історії. Впроваджена система сповіщень через Telegram Bot API забезпечує прозорість фінансових операцій у реальному часі.

Практичне значення роботи полягає у створенні функціонального FinTech-прототипу, що може бути використаний як основа для розробки комерційного банківського застосунку після проходження відповідних регуляторних процедур та аудиту безпеки.

Перспективи подальшого розвитку системи: впровадження двофакторної автентифікації (2FA), розширення кредитного модуля (кредитний скоринг, графік погашення), інтеграція з платіжними шлюзами, розробка мобільного застосунку (React Native), реалізація мікросервісної архітектури з Docker Compose.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. FastAPI Documentation. – URL: https://fastapi.tiangolo.com (дата звернення: 01.05.2026).

2. React Documentation. – URL: https://react.dev (дата звернення: 01.05.2026).

3. PostgreSQL Documentation. – URL: https://www.postgresql.org/docs (дата звернення: 01.05.2026).

4. SQLAlchemy Documentation. – URL: https://docs.sqlalchemy.org (дата звернення: 01.05.2026).

5. Tailwind CSS Documentation. – URL: https://tailwindcss.com/docs (дата звернення: 01.05.2026).

6. Python-jose: JOSE implementation in Python. – URL: https://github.com/mpdavis/python-jose (дата звернення: 01.05.2026).

7. Passlib – Password Hashing Library for Python. – URL: https://passlib.readthedocs.io (дата звернення: 01.05.2026).

8. Telegram Bot API Documentation. – URL: https://core.telegram.org/bots/api (дата звернення: 01.05.2026).

9. Railway Documentation. – URL: https://docs.railway.app (дата звернення: 01.05.2026).

10. Pydantic Documentation. – URL: https://docs.pydantic.dev (дата звернення: 01.05.2026).

11. OWASP Top Ten. – URL: https://owasp.org/www-project-top-ten (дата звернення: 01.05.2026).

12. JSON Web Token (JWT) RFC 7519. – URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 01.05.2026).

13. PCI DSS v4.0. – URL: https://www.pcisecuritystandards.org (дата звернення: 01.05.2026).

14. Закон України «Про платіжні послуги» від 30.06.2021 р. №1591-IX. – URL: https://zakon.rada.gov.ua (дата звернення: 01.05.2026).

15. Gamma E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley, 1994. – 395 p.

16. Kleppmann M. Designing Data-Intensive Applications. – O'Reilly, 2017. – 590 p.

17. Richardson C. Microservices Patterns. – Manning, 2018. – 520 p.

18. Alembic Documentation. – URL: https://alembic.sqlalchemy.org (дата звернення: 01.05.2026).

19. Claude (Anthropic). Anthropic Claude AI assistant [Електронний ресурс]. – Режим доступу: https://claude.ai – Дата звернення: травень 2026 р.

Додаток А

Лістинг основних програмних модулів

Нижче наведено лістинг ключових програмних модулів системи VAULT BANK.

Модуль авторизації (backend/auth.py) – основні функції:

```
@router.post("/login")async def login(data: LoginSchema, db: Session = Depends(get_db)):    user = db.query(User).filter(User.username == data.username).first()    if not user or not verify_password(data.password, user.password_hash):        raise HTTPException(status_code=401, detail="Invalid credentials")    token = create_access_token({"sub": str(user.id)})    await send_telegram(f"✅ Вхід: {user.username}")    return {"access_token": token, "token_type": "bearer"}@router.post("/register")async def register(data: RegisterSchema, db: Session = Depends(get_db)): if db.query(User).filter(User.username == data.username).first():        raise HTTPException(status_code=400, detail="Username already exists")    hashed = get_password_hash(data.password)    user = User(username=data.username, email=data.email, phone=data.phone, password_hash=hashed)    db.add(user)    db.flush()    card_number = generate_card_number()    account = Account(user_id=user.id, card_number=card_number, balance=0) db.add(account)    db.commit()    return {"message": "Registration successful"}
```

Модуль переказів (backend/transactions.py) – переказ коштів:

```
@router.post("/transfer")async def transfer(data: TransferSchema,                 current_user: User = Depends(get_current_user),                 db: Session = Depends(get_db)):    from_account = db.query(Account).filter( Account.user_id == current_user.id).first()    if data.search_by == "username":        recipient = db.query(User).filter( User.username == data.recipient).first()        to_account = db.query(Account).filter(            Account.user_id == recipient.id).first()    else:        to_account = db.query(Account).filter(            Account.card_number == data.recipient).first()    if from_account.id == to_account.id:        raise HTTPException(400, "Cannot transfer to yourself")    if from_account.balance < data.amount:        raise HTTPException(400, "Insufficient funds") from_account.balance -= data.amount    to_account.balance += data.amount    tx = BankTransaction(from_account_id=from_account.id,                         to_account_id=to_account.id, amount=data.amount, tx_type="transfer")    db.add(tx)    db.commit()    await send_telegram(f"💸 Переказ {data.amount} UAH")    return {"message": "Transfer successful"}
```

Додаток Б

Лістинг фронтенд-компонентів

Компонент авторизації (frontend/src/Auth.jsx) – handleLogin:

```
const handleLogin = async () => {  setLoginError("");  if (!loginUsername || !loginPassword) {
setLoginError("Заповніть всі поля");    return;  }  setLoginLoading(true);  try {    const data = await
api.post("/auth/login", {      username: loginUsername,      password: loginPassword    });
localStorage.setItem("token", data.access_token);    onLogin(data.access_token);  } catch (e) {
setLoginError(e.message);  } finally {    setLoginLoading(false);  }};
```

Компонент таймера кредиту:

```
useEffect(() => {  if (!loan || loan.status !== "active") return;  const interval = setInterval(() => {    const now = new
Date();    const end = new Date(loan.end_date);    const diff = end - now;    if (diff <= 0) {
setTimeLeft("00:00:00:00");      clearInterval(interval);      return;    }    const days = Math.floor(diff /
(1000*60*60*24));    const hours = Math.floor((diff % (1000*60*60*24)) / (1000*60*60));    const mins =
Math.floor((diff % (1000*60*60)) / (1000*60));    const secs = Math.floor((diff % (1000*60)) / 1000);
setTimeLeft(`${String(days).padStart(2,'0')}:${String(hours).padStart(2,'0')}:${String(mins).padStart(2,'0')}:${String(s
ecs).padStart(2,'0')}`);  }, 1000);  return () => clearInterval(interval);}, [loan]);
```

Додаток В

Декларація про академічну доброчесність та використання ШІ

Я, Яковченко Ярослав, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Кросплатформне програмне забезпечення онлайн банкінгу для приватного користувача» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що інструменти ШІ (Claude, ChatGPT) були використані для:

- стилістичного редагування та перевірки граматики;
- допомоги у написанні/відлагодженні програмного коду (генерація шаблонів компонентів, налагодження SQL-запитів).

«___»_____ 2026 р. _____Ярослав ЯКОВЧЕНКО