Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

перший (бакалаврський)
(рівень вищої освіти)

Створення додатоку для ведення фінансової операції компанії
(тема)

Виконав:
студент 4 курсу, групи ТРІТЗР-20-1

Рижко Б.Р.

Спеціальності 172 Телекомунікації та радіотехніка

Тип програми Освітньо-професійна

Освітня програма Інтелектуальні технології засобів радіоелектроніки

Керівник

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 172 Телекомунікації та радіотехніка
Тип програми Освітньо-професійна
Освітня програма Інтелектуальні технології засобів радіоелектроніки
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІИТАР_____
(підпис)
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Рижко Богдану Руслановичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Створення додатоку для ведення фінансової операції компанії
Затверджена наказом по університету від 20.05.2024 №479Ст
2. Термін подання студентом роботи до екзаменаційної комісії 21.06.2024 р.
3. Вихідні дані до роботи Мова програмування - TypeScript
БД – PostgreSQL
Операційна система - Linux
Backend - NodeJS, Express
Frontend - Flutter

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз предметної області дослідження
4.3 Аналіз літератури та вимог технічного завдання
4.4 Розробка програмного модуля
4.5 Розробка структури бази даних
4.6 Висновки
4.7 Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – --- с. формату А4

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Вступ	29.04 – 30.04.24	Виконано
2	Аналіз предметної області дослідження	01.05 – 05.05.24	Виконано
3	Аналіз літератури та вимог технічного завдання	8.05 – 12.05.24	Виконано
4	Розробка програмного модуля	15.05 – 19.05.24	Виконано
5	Розробка структури бази даних	24.05 – 14.06.24	Виконано
6	Висновки	15.06 – 16.06.24	Виконано
7	Додатки	17.06 – 18.06.24	Виконано
8	Подача на перевірку	21.06 – 25.06.24	Виконано
9	Захист дипломної роботи	26.06.2024	Виконано

Дата видачі завдання 21.03.2024 р.

Студент _____ Рижко Б.Р. (підпис)

Керівник роботи _____ доц. Бронніков А. І.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«07» червня 2024 р. Рижко Б.Р

РЕФЕРАТ

Пояснювальна записка: 56 с., 7 табл., 24 рис., 2 дод., 21 джерел.

БАЗА ДАНИХ, LINUX, POSTGRESQL.

Мета роботи – розробити програмний засіб, що буде проводити облік фінансових операцій з використанням штучного інтелекту;

Об’єкт розробки – запис фінансових операцій;

Предмет розробки – підсистема, що реалізує керування фінансовою частиною розумного виробництва.

Звіти допомагають інвесторам, кредиторам, регуляторам, менеджменту компанії та іншим зацікавленим сторонам оцінити фінансовий стан компанії, її прибутковість та ризики. Фінансова звітність також є інструментом для внутрішнього управління, який допомагає компаніям приймати рішення та планувати свою діяльність.

Додаток має на собі мету покращити та оптимізувати всі дані, розмістивши їх всіх в одному місці.

ABSTRACT

Explanatory note: 56 p., 7 tables, 24figs., 2 appendices, 21 sources.

DATABASE, LINUX, POSTGRESQL.

The purpose of the work is to develop a software tool that will record financial transactions using artificial intelligence;

The object of development is the record of financial transactions;

The subject of development is a subsystem that implements the management of the financial part of smart production.

Reports help investors, creditors, regulators, company management, and other stakeholders assess a company's financial health, profitability, and risks. Financial reporting is also a tool for internal management that helps companies make decisions and plan their operations.

The app aims to improve and optimize all the data by putting it all in one place.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект;

MVC – модель–вигляд–контролер.

ВСТУП

Темою даної дипломної роботи є створення додатоку для ведення фінансової операції компанії. Фінансова операції компанії – це набір фінансових операції і документів, які розкривають фінансовий стан і фінансові результати діяльності підприємства протягом певного періоду. Фінансова операції є важливим інструментом для оцінки фінансового здоров'я та роботи компанії і включає наступні ключові елементи:

– баланс (Баланс стану). Баланс показує фінансовий стан компанії в певний момент часу. Він розкриває активи (всі ресурси компанії, такі як гроші, майно, заборгованості) та пасиви (фінансові зобов'язання та власний капітал) компанії;

– звіт про прибуток і збитки (Прибутковий звіт). Цей звіт відображає доходи, витрати та прибуток (або збиток) компанії протягом певного періоду, зазвичай за рік. Він включає чистий прибуток (прибуток після оподаткування) та показники, які описують прибутковість компанії;

– звіт про зміни в капіталі (Звіт про рухи капіталу). Цей звіт показує, як змінювався власний капітал компанії протягом певного періоду через різні операції, такі як видача дивідендів, придбання чи продаж акцій, а також прибуток або збиток;

– звіт про готовність до платежу (Готівковий звіт). Цей звіт показує рух готівки в компанії та її платіжну здатність, включаючи готівку, що надходить та виходить з компанії.

Ці звіти допомагають інвесторам, кредиторам, регуляторам, менеджменту компанії та іншим зацікавленим сторонам оцінити фінансовий стан компанії, її прибутковість та ризики. Фінансова звітність також є інструментом для внутрішнього управління, який допомагає компаніям приймати рішення та планувати свою діяльність.

Мета роботи – розробити програмний засіб, що буде проводити облік фінансових операцій з використанням штучного інтелекту;

Об’єкт розробки – керування розумним виробництвом;

Предмет розробки – підсистема, що реалізує керування фінансовою частиною розумного виробництва.

1 АНАЛІЗ ЛІТЕРАТУРИ ТА ВИМОГ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Загальна характеристика системи

Темою цього курсового проєкту є створення додатоку для ведення фінансової операції компанії.

Для того, щоб створити додаток, потрібно для початку вибрати технології, мову програмування та метод її створення. Потрібно обрати мову програмування база даних, програми для backend та frontend.

1.2 Вибір мови програмування

Мова програмування – це формальний засіб для написання програм, які виконують певні завдання на комп'ютері. Це мова, яку розуміє комп'ютер і виконує інструкції, надані програмістом. Мови програмування використовуються для створення програмного забезпечення, веб-сайтів, мобільних додатків, вбудованих систем, і багатьох інших застосувань [4].

Існує багато різних мов програмування, кожна з яких має свої особливості і призначення. Розглянемо декілька популярних мов програмування:

Java:

– висока переносимість завдяки віртуальній машині Java (JVM);

– в розробці мобільних додатків, веб-серверів, корпоративного програмування.

Python:

– простий для вивчення і читання, має багато бібліотек;

– використовується в науковому обчисленні, веб-розробці, штучному інтелекті.

C++:

– висока продуктивність і можливість роботи з пам'яттю;

– застосовується в розробці вбудованих систем, ігор, операційних систем.

JavaScript:

– основна мова для фронтенд розробки веб-сайтів;

– використовується для створення веб-інтерфейсів та додатків.

Ruby:

– простий і елегантний синтаксис, акцент на продуктивності розробника;

– використовується в веб-розробці, зокрема з Ruby on Rails.

PHP:

– використовується для розробки веб-сайтів і взаємодії з базами даних;

– популярний для створення динамічних веб-сторінок.

C#:

– розробка на платформі Microsoft, включаючи Windows і ігрові консолі;

– використовується в розробці десктопних додатків і ігор.

Swift:

– розробка мобільних додатків для iOS та macOS;

– висока швидкодія і безпека.

Go (Golang):

– спрямований на швидкодію і простоту синтаксису;

– часто використовується для створення веб-серверів і мікрослужб.

Kotlin:

– офіційна мова для розробки на платформі Android;

– має багато спільних рис з Java, але додає нові можливості.

Кожна мова програмування має свій синтаксис і правила, які програмісти використовують для написання програм. Вибір мови програмування залежить від конкретних вимог проекту, особистих вподобань програміста та інших факторів.

JavaScript може бути гарним вибором для створення додатку для ведення фінансової операції компанії. Ось кілька переваг JavaScript для такого додатку:

– веб-орієнтований підхід. JavaScript – це мова, яка добре підходить для створення веб-додатків, і фінансова операції часто потребує віддаленого доступу через веб-інтерфейс. JavaScript може використовуватися для створення динамічних та інтерактивних веб-сторінок, які дозволять користувачам зручно працювати з фінансовими даними;

– багато бібліотек і фреймворків, в екосистемі JavaScript існують численні бібліотеки і фреймворки, які полегшують розробку веб-додатків. Наприклад, React, Angular, або Vue.js можуть спростити створення веб-інтерфейсу для фінансового додатку;

– зручність інтеграції, JavaScript добре інтегрується з багатьма серверними технологіями, такими як Node.js, що дозволяє легко спілкуватися з сервером та базами даних. Це важливо для роботи з фінансовими даними, які зазвичай зберігаються в базах даних;

– крос-платформеність, JavaScript може використовуватися для створення веб-додатків, які працюють на різних платформах та пристроях без значних змін в коді, що полегшує розповсюдження та доступність додатку;

– безкоштовність, JavaScript – це вільна та відкрита мова програмування, тобто ви можете використовувати її безкоштовно, що може бути важливо для стартапів та невеликих компаній з обмеженим бюджетом [5].

TypeScript – це мова програмування, яка є суперсетом мови JavaScript. Іншими словами, TypeScript розширює можливості JavaScript, додаючи до нього статичну типізацію та деякі інші функції. Основні риси TypeScript включають:

– статична типізація, TypeScript дозволяє вказувати типи для змінних, функцій та об'єктів. Це допомагає виявити помилки на етапі розробки, забезпечуючи більшу надійність та зручність у великих проектах;

– класи та модулі, TypeScript підтримує об'єктно-орієнтований підхід до програмування, включаючи підтримку класів, інтерфейсів та модулів. Це полегшує організацію та структурування коду;

– сумісність з JavaScript, TypeScript використовує ту ж синтаксичну основу, що і JavaScript, і може включати існуючий JavaScript-код без змін. Крім того, TypeScript генерує JavaScript-код, який можна виконувати в браузерах або на серверах Node.js;

– інструменти розробки, TypeScript має багато інструментів розробки, включаючи редактори, які підтримують автодоповнення, перевірку типів та інші корисні функції;

– спільнота та екосистема, TypeScript користується популярністю в розробницькій спільноті, і в ній існують численні бібліотеки та фреймворки для різних сфер розробки, включаючи веб-розробку, мобільну розробку та багато інших;

– підтримка ES6 та нові функції, TypeScript підтримує стандарт ECMAScript 6 (ES6) та нові функції JavaScript, що дозволяє використовувати останні досягнення мови.

TypeScript допомагає збільшити якість та безпеку коду, особливо в великих проектах, де може бути багато команд та розробників. Вона стає все більш популярною в розробницькому співтоваристві і знаходить застосування у різних галузях розробки програмного забезпечення. Через збільшення безпеки, є актуальним використання цієї мови у даному додатку [6].

1.3 Вибір програми для БД

База даних – це організована колекція даних, яка зберігається і управляється з метою забезпечення доступу до цих даних і їх подальшого використання. Бази даних використовуються для зберігання, організації, оновлення та аналізу інформації. Вони можуть містити різні типи даних, такі як текст, числа, зображення, відео та інші формати.

Бази даних використовуються в різних галузях, включаючи бізнес, науку, освіту, медицину та багато інших. Вони можуть бути створені та управлятися за допомогою спеціалізованих систем управління базами даних (СУБД), таких як MySQL, PostgreSQL, Microsoft SQL Server, Oracle та інші.

Основними функціями бази даних є збереження даних, забезпечення доступу до них, забезпечення цілісності даних (тобто захист від помилок і несанкціонованого доступу), підтримка конкурентного одночасного доступу до даних різними користувачами та надання можливості проведення операцій з даними, таких як додавання, видалення, оновлення та пошук. Вони можуть бути реляційними, нереляційними або гібридними, залежно від того, як дані організовані та зв'язані між собою. В реляційних базах даних дані зберігаються у вигляді таблиць з рядками і стовпцями, і між таблицями встановлюються відносини на основі ключів. Нереляційні бази даних використовують інші способи для зберігання та організації даних, такі як документи, ключ-значення, стовпчасті сховища тощо.

Важливим аспектом баз даних є їхній дизайн, який визначає, як дані будуть структуровані та як будуть встановлені зв'язки між ними. Добре спроектована база даних може сприяти ефективному використанню даних та полегшити роботу з ними, в той час як поганий дизайн може призвести до проблем з продуктивністю та надійністю системи.

Застосування баз даних різняться від простих систем збереження інформації до складних підприємницьких систем управління даними. Бази даних грають важливу роль в сучасному інформаційному суспільстві та великою мірою визначають можливості компаній та організацій з ефективного управління та аналізу даних [7].

Актуальним варіантом є PostgreSQL. Це об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із

початковим кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

Функції дозволяють виконувати деякий код безпосередньо сервером бази даних. Ці функції можуть бути написані на SQL, який має деякі примітивні програмні оператори, такі як галуження та цикли.

Backend і frontend є двома основними частинами багатьох веб-додатків та програмного забезпечення загалом. Frontend відповідає за відображення інтерфейсу для користувача, в той час як backend забезпечує функціональність та логіку, яка лежить в основі додатку. Обидва компоненти співпрацюють, щоб створити повноцінну програму або систему [17].

1.4 Backend

Backend (задній кінець) – це термін, який використовується в розробці програмного забезпечення для позначення частини програми або системи, яка відповідає за обробку даних, бізнес-логіку та внутрішню логістику. Backend забезпечує функціональність програми, яка не відображається для користувача, і працює у фоновому режимі [18].

Основні завдання backend включають наступне:

– збереження та управління даними. Backend відповідає за зберігання даних в базі даних або інших сховищах, а також за виконання операцій з цими даними, такими як додавання, видалення, оновлення та пошук;

– бізнес-логіка, Backend включає в себе код, який визначає логіку додатку або системи. Він виконує обчислення, обробку даних і приймає рішення відповідно до задач програми;

– взаємодія з базами даних, Backend взаємодіє з базами даних, виконуючи запити до них і отримуючи необхідну інформацію для подальшої обробки;

– забезпечення безпеки, Backend відповідає за забезпечення безпеки системи, включаючи аутентифікацію, авторизацію та захист від зловмисних атак;

– взаємодія з frontend, Backend надає API (інтерфейс програмування додатків), який дозволяє взаємодіяти з frontend (користувацьким інтерфейсом) та іншими частинами системи.

Як backend використав Node.js та Express.js як програмний каркас. Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

– асинхронна одно-нитева модель виконання запитів;

– неблокуючий ввід/вивід;

– система модулів CommonJS;

– рушій JavaScript Google V8.

Для керування модулями використовується пакетний менеджер npm (node package manager). Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

В платформі використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх

викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу нитей (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe) [8].

Express.js, або просто Express – програмний каркас розробки серверної частини вебзастосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT. Він спроєктований для створення вебзастосунків і API. Де-факто є стандартним каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але має велику кількість плаґінів, що підключаються.

Express є бекендом для програмного стека MEAN, разом з базою даних MongoDB і каркасом AngularJS для фронтенду [9, 10, 11].

1.5 Frontend

Frontend (передній кінець) – це частина програмного забезпечення або веб-додатку, яка відповідає за відображення користувацького інтерфейсу та взаємодію з користувачем. Frontend обумовлює те, як інформація та функціональність відображаються на екрані комп'ютера, смартфона або іншого пристрою, що використовується користувачем [12].

Основні завдання frontend включають наступне:

– відображення інтерфейсу, Frontend відповідає за розміщення та стилізацію елементів на веб-сторінці або в додатку, включаючи тексти, зображення, кнопки, форми та інші компоненти;

– взаємодія з користувачем, Frontend надає способи взаємодії користувача з додатком, такі як кнопки, поля для введення, меню та інші елементи, які дозволяють користувачу взаємодіяти з програмою;

– обробка подій, Frontend включає в себе JavaScript-код, який дозволяє обробляти події, такі як кліки, наведення миші, введення даних та інші дії користувача;

– завантаження та відображення даних, Frontend здатний відправляти запити на сервер для отримання даних, які потім можуть бути відображені на сторінці.

Обрав Flutter як фронтенд. Flutter – це програмний каркас із відкритим кодом для створення додатків для платформ Android та iOS, а також на вебі, розроблений компанією Google. Він є основним способом створення додатків для Google Fuchsia. Весь графічний інтерфейс Google Fuchsia створено за допомогою Flutter.

Flutter складається з:

– Flutter рушій – програмний рушій для рендерингу, написаний в основному на C++ з використанням графічної бібліотеки Google Skia. Він також використовує SDK платформ Android або iOS;

– базової бібліотеки (Foundation library) — бібліотека складається з класів та функцій (написані на Dart), які використовують для побудови Flutter програм, для взаємодії із Flutter рушієм;

– віджетів. Дизайн інтерфейсу користувача у Flutter будують з віджетів.

Віджет у Flutter являє собою незмінний об'єкт, який описує частину інтерфейсу користувача. Вся графіка, текст, фігури та анімації створюють за допомогою віджетів [19].

1.6 Аналіз аналогічних продуктів

Є багато іноземних додатків для обліку фінансів. Частіше за все, ці додатки створені для фізичних осіб, а не для компаній. Ось деякий список із самих популярних:

– Monefy, контроль витрат та відстеження розтрат, програма з найпростішим інтерфейсом. Після запуску на головному екрані з'являється велике коло, де будуть відображатися доходи та витрати. Останні класифікуються за категоріями, достатньо вибрати відповідний значок (одяг, їжа, ресторани, таксі, спорт, подарунки, тварини) та вказати суму.Вводити всі дані доведеться вручну. Великий значок плюса в нижній частині екрана вказує на прибуток, мінус — на витрати. Є можливість вибрати спосіб надходження коштів: кеш чи електронний переклад. Валюту вибираєте самостійно. Доступні гривні, долари, євро, шилінги, фунти стерлінгів, рупії, дирхами та ще десятки варіантів;

– Money Flow, програма для фінансового контролю доступна на IPhone, у Google Play Market є програми з такою назвою. Вони відрізняються за інтерфейсом і мовою – доступна тільки англійська. Відразу після запуску у програмі пропонується створити демо рахунок. Це свого роду інструкція, яка дозволить ознайомитись з основним функціоналом та грамотно виконати налаштування. У програмі відображається загальна статистика надходжень та витрат за місяць, рік та весь період використання ПЗ, а також інформація по днях. У рядку звітів можна побачити кругову діаграму із зазначенням відсоткової та точної кількості витрат за кожною категорією. Серед інших опцій це можливість точно планувати бюджет та ставити цілі, функція додавання фото до кожної операції (можна сфотографувати чек або іншу інформацію) та функція повтору, щоб не вводити щомісячні платежі вручну.

1.7 Економічна виправданість

Для того, щоб заробляти на своєму продукту, розробники додають платні функції у свій продукт, або рекламу, яку можна відключити за одноразову плату або щомісячну підписку. Частіше усього з відключенням реклами вам дають розширений функціонал програми, так званий Pro режим.

У програмі Monefy розробники ПЗ пропонують платну передплату. Серед її особливостей:

– можливість створення власних категорій та вибору іконок до них;

– відсутність рекламних оголошень;

– синхронізація з кількома пристроями.

А ось вже у програмі Money Flow основна частина функцій є платною і їх рентабельність залишається на розсуд користувача [14,15].

2 АЛГОРИТМ РОБОТИ ПРОГРАМИ

2.1 Блок схема

БД складається з 4 таблиць – користувачі, рахунки, витрати та доходи. Є можливість зарєструватися та залогінитись, це реалізовано за допомогою токена, генерація токена відбувається за допомогою бібліотеки jsonwebtoken. Пароль у бд зберігається захешованим, хешування відбувається за допомогою бібліотеки bcrypt.

Після входу в застосунок користувач може побачити 3 екрани: сторінку з транзакціями, сторінку з графіками і розрахунки.

Спочатку користувач потрапляє на сторінку з рахунками його компанії, тут можна ними керувати, до кожного рахунку прив'язується історія транзакцій. Транзакції діляться на 2 сторінки – доходи і витрати, тут є можливість створити нову транзакцію, вибрати спосіб, як поділитися з якоюсь транзакцією або видалити транзакцію. На другій сторінці показуються доходи і витрати у вигляді графіків. На третій сторінці проводяться розрахунки з управління фінансами, де є можливість запланувати доходи і розтрати.

Блок-схема – це графічне представлення алгоритму або процесу, яке використовує блоки для представлення кроків або дій та стрілки для показу зв'язків між ними. Блок-схеми використовуються для візуального подання порядку виконання завдань чи послідовності подій у програмуванні, інженерії, бізнес-процесах та інших областях. Вигляд алгоритму приведено на рис. 2.1, у вигляди блок-схеми.

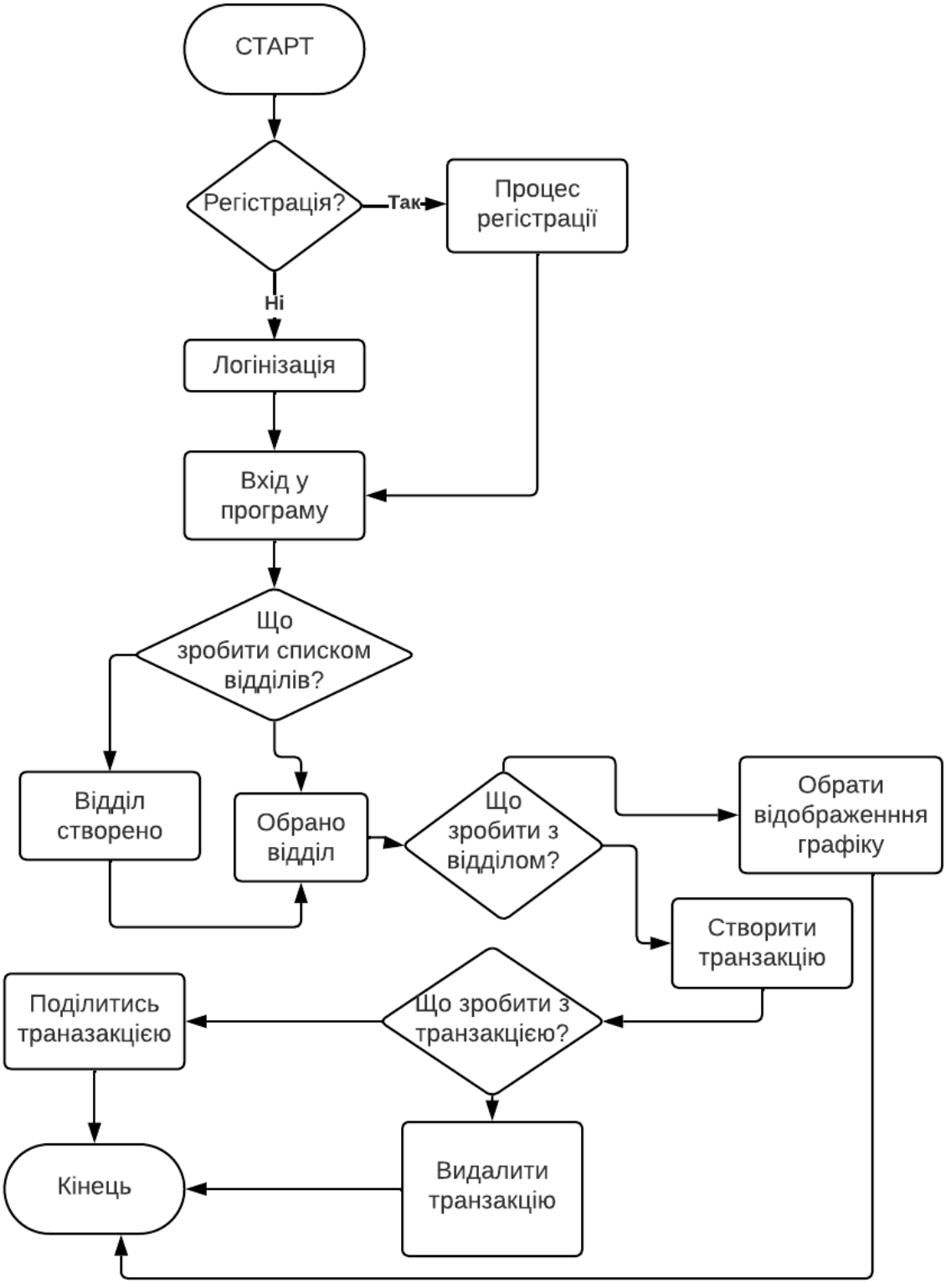


Рисунок 2.1 – Блок схема

2.2 Архитектура MVC

MVC - найперший із шаблонів проєктування архітектури програми. Як і всі, що з’явилися на його основі, він потрібен для організації коду, поділу його на функціональні блоки для різних завдань. У MVC є три складові, що відповідають за бізнес-логіку, графічний інтерфейс та дані програми:

Model – база даних та основна логіка програми для роботи з нею, обробки запитів, проведення обчислень тощо;

View – інтерфейс користувача, усі візуальні елементи, з якими буде взаємодіяти користувач;

Controller – прошарок між View і Model, що реагує на зміни.

Вони приведені на рисунку 2.2.

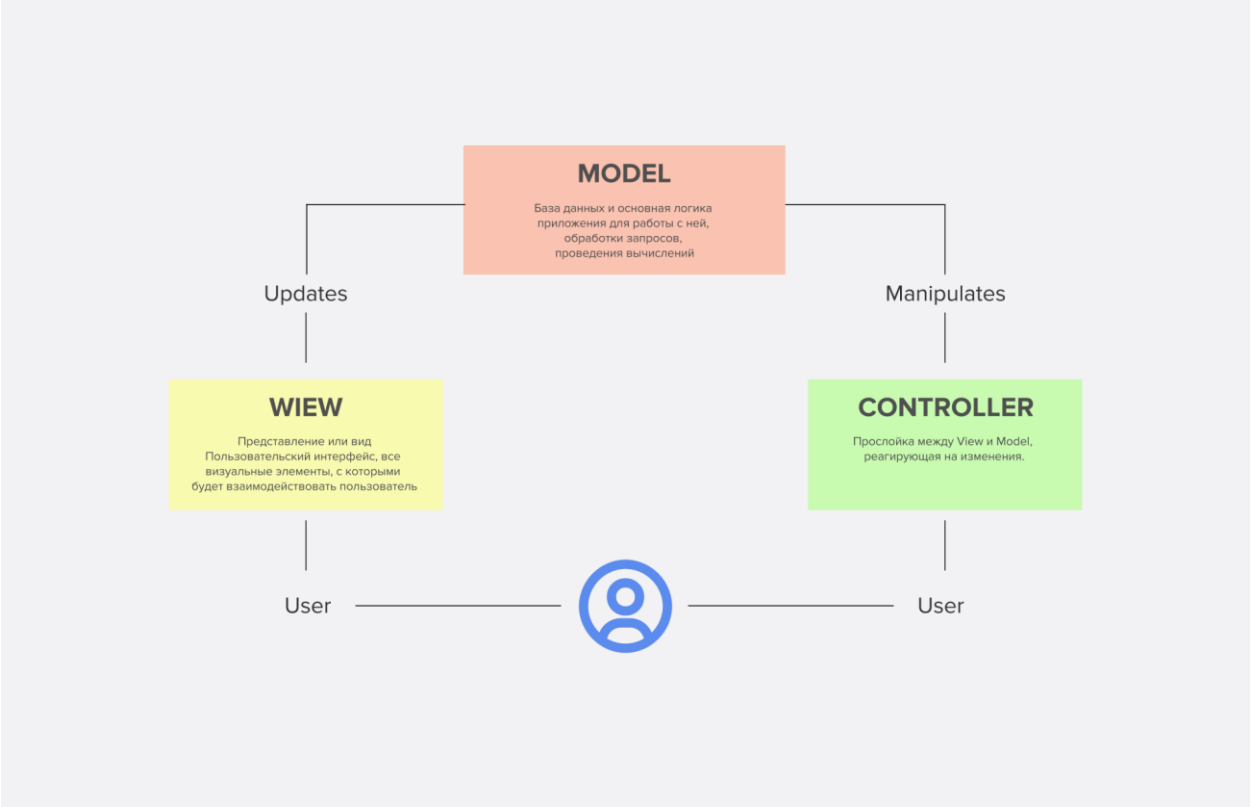


Рисунок 2.2 – Зв’язок трьох складових

У патерні MVC контролер і вид залежить від моделі, а вона від них ні. У контролері перебуває логіка отримання даних із моделі та передачі до виду. Controller передає запити, сформовані View Model, де зберігаються всі дані. Модель змінюється відповідно до запиту. Вид «дізнається» про те, як йому потрібно змінитись, оскільки підписаний на події моделі, і відображає нові дані.

У мене чітко виражені контр Controller (рис. 2.3) та Model (рис. 2.4) для взаємодії з базою даних.

Рисунок 2.3 – Controller

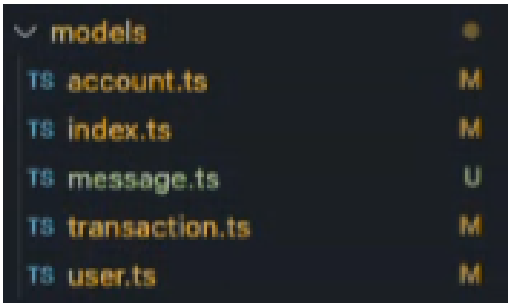


Рисунок 2.4 – Model

View – оборобчик який взаємодію з контролерами і моделями. Якщо зайти в app.ts то можна занйти цей код. Його приведено у додатку А [16].

2.3 Штучний Інтелект

Логіка роботи штучного інтелекту (ШІ) зазвичай ґрунтується на використанні алгоритмів і методів комп'ютерної науки, таких як машинне навчання та глибоке навчання. Він відноситься до галузі комп'ютерних наук, що ставить за мету створення систем, які можуть виконувати завдання, які зазвичай потребують людського інтелекту. Ці системи можуть включати різноманітні аспекти інтелектуальної діяльності, такі як розуміння мови, вирішення проблем, навчання, планування, сприйняття, розпізнавання образів і багато іншого. В нашому контексті він схожий на роботу головного мозку, апаратна та програмна структура теж нагадує мозок. Це математичний апарат з електронною частиною, яка може працювати з інформацією як людина

Основними аспектами логіки роботи ШІ є обробка даних, машинне навчання, глибоке навчання, планування і прийняття рішень та взаємодія з користувачем. Логіка роботи ШІ базується на використанні алгоритмів та методів комп'ютерної науки для досягнення певних цілей, таких як аналіз даних, розуміння мови, прийняття рішень та взаємодія з людьми.

Вже існують досить складні та «розумні» алгоритми, є різноманітні тести і часто не можливо відрізнити, з ким людина спілкується, якщо не бачить і не знає напевно, чи то людина, чи то штучний інтелект [17].

2.4 Ризики ШІ для людства

Штучний інтелект представляє як потенційні переваги, так і ризики для людства. Ризики пов'язаних з розвитком ШІ приведені у таблиці 2.1

Таблиця 2.1 – Ризики ШІ

Ризики	Опис
Втрата контролю	Одним з основних ризиків є втрата контролю над ШІ. З розвитком та самовдосконаленням штучного інтелекту може виникнути ситуація, коли людство втратить здатність контролювати дії ШІ. Це може призвести до непередбачуваних наслідків та великих проблем, оскільки ШІ може розпочати діяти в шкідливий спосіб без контролю або зрозуміння його створювачів.
Масова безробіття	Розвиток ШІ також може викликати масове безробіття через автоматизацію робочих процесів. ШІ може замінити людей у багатьох сферах, що може призвести до значного зменшення кількості робочих місць та виникнення соціальних проблем, пов'язаних з безробіттям.
Етичні питання	З розвитком ШІ виникають різноманітні етичні питання. Це охоплює такі аспекти, як використання автономних збройних систем, конфіденційність даних, а також вплив штучного інтелекту на суспільство та моральні цінності. Вирішення цих етичних питань вимагає уважного розгляду та розробки відповідних регулятивних механізмів.
Збільшення рівня нерівності	ШІ може також збільшити рівень нерівності у суспільстві. Відбувається нерівномірний доступ до ресурсів, технологій та можливостей, що може призвести до зростання рівня соціальної та економічної нерівності.

Продовження табл. 2.1

Ризики	Опис
Потенційна загроза для людства	Існує загроза, що ШІ може стати загрозою для самого людства. Це може статися через недбалість, помилки в програмному забезпеченні або навіть через намірені дії. Недостатня увага до ризиків, пов'язаних з розвитком ШІ, може призвести до непередбачуваних наслідків.
Втрата робочих місць у важливих сферах	Розвиток ШІ може призвести до втрати робочих місць у важливих сферах, таких як медицина, фінанси та інші. ШІ може здійснювати дії більш точно та ефективно, що може призвести до заміщення людської праці.
Залежність від технологій	З розвитком ШІ зростає загальна залежність суспільства від технологій. Це може зробити суспільство вразливим до відмови технологій або кібератак.

2.5 Транслювання TypeScript на JavaScript

Проєкт розроблено з використанням TypeScript. Але є головною задачею – налаштування транскрипції TypeScript на JavaScript. Приклад:

Щоб конвертувати код з typescript в js, потрібно його скомпілювати. Щоб встановити typescript глобально, скористайтеся командою sudo npm install -g typescript (понадобится ввести пароль администратора).

Далі створіть файл index.ts і додайте до нього наступний код typescript:

```
const message: string = 'Hello World';
console.log(message);
```

Щоб скомпілювати код в js, введіть команду tsc index.ts, а потім створіть js-файл index.js в тому ж каталозі:

```
var message = 'Hello World';
console.log(message);
node index.js // => Hello World!
```

Я використав більш надійну механіку, сама частина транслювання має таке древо (рисунок 2.5)

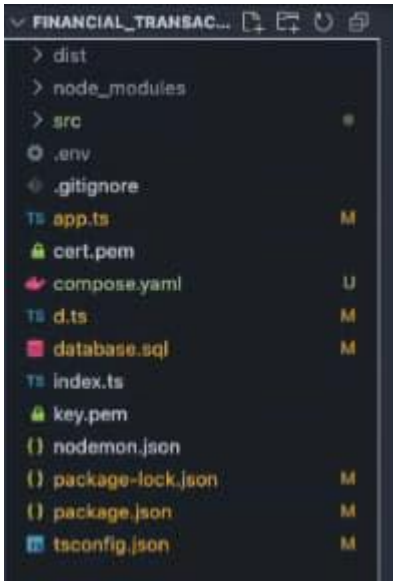


Рисунок 2.5 – Древо частини транслювання коду з TypeScript на JavaScript

3 РОЗРОБКА СТРУКТУРИ БАЗИ ДАНИХ

База даних складається з 4 таблиць – users, account, expenses та reveneus. Вони приведені на рисунку 3.1.

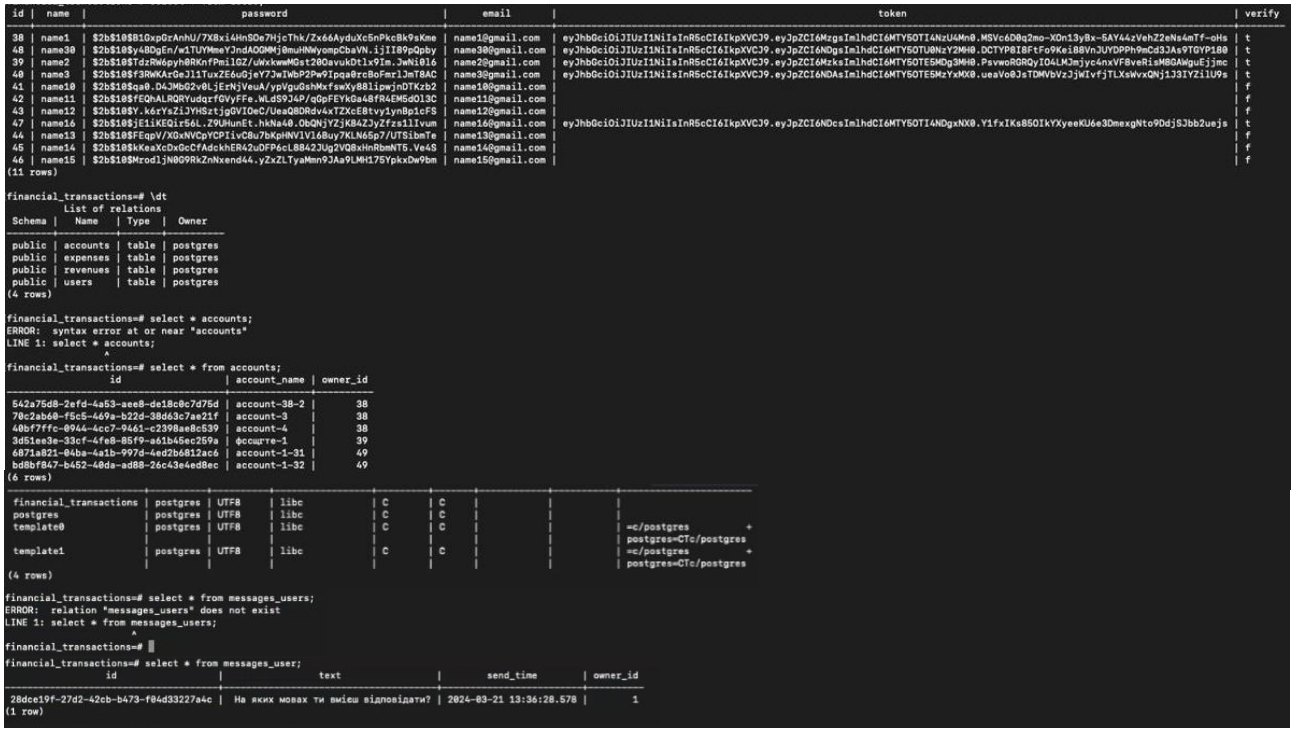


Рисунок 3.1 – Склад бази даних

Розберемо кожну таблицю окремо. Склад таблиці users у таблиці 3.1.

Таблиця 3.1 – Склад таблиці users

№	Неймінг	Тип	Ознака
1	id	integer	PK
2	name	varchar	-
3	password	varchar	-
4	email	varchar	-
5	token	varchar	-
6	verify	boolean	-

Опис кожного неймінгу більш конкретно:

– id – унікальний id, який генерує PostgreSQL, необхідний для впізнавання користувача і зв’язує таблицю users із accounts;

– name – ім’я користувача;

– password – пароль користувача, зберігається у хешованому форматі, для хешування використовую бібліотеку bcrypt;

– email – пошта користувача, додав валідацію цього поля за допомогою сталих виразів у JS;

– token – авторизаційний токен, складається з 3 частин: id користувача, секретне слово і термін життя токену, для спрощення роботи зробив усі токени без терміну життя (завжди валідні), генерую токен за допомогою бібіліотеки jsonwebtoken;

– verify – булеве значення, яке визначає чи пройшов верифікацію користувач чи ні.

З’єднана з таблицею account через значення id – ownder_id. Вид таблиці приведено на рисунку 3.2.

```
financial_transactions=# select * from users;
 id | name   |                           password                           |        email        |                                                   token                                                   | verify
----+--------+--------------------------------------------------------------+---------------------+-----------------------------------------------------------------------------------------------------------+--------
 49 | name31 | $2b$10$cI/wFb1N.GhYkz1Lx2lIhOQmg0IdEHZ9r7thqqs8dwfrAgpy7u6lm | name31@gnmail.com   | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDksImlhdCI6MTY5OTU0Nzg1OX0.YilJVKNiO4e6O0ozHwC7I-Az3aNR2b6c5JuPtUb20AM | t
 39 | name2  | $2b$10$TdzRW6pyh0RKnfPmilGZ/uWxkwwMGst20OavukDtlx9Im.JwNi0l6 | name2@gmail.com     | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MzksImlhdCI6MTY5OTYxOTc3OX0.kHedFHgEMI42e27e9AU9sbH8PyGDsvb1bdngBZxrEzY | t
 48 | name30 | $2b$10$y4BDgEn/w1TUYMmeYJndAOGMMj0muHNWyompCbaVN.ijII89pQpby | name30@gmail.com    |                                                                                                           | f
 38 | name1  | $2b$10$B1GxpGrAnhU/7X8xi4HnSOe7HjcThk/Zx66AyduXc5nPkcBk9sKme | name1@gmail.com     | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MzgsImlhdCI6MTY5OTk1MzkyMH0.7vrrbNa4-lMqpYuDaoih_KXb4CIO9oqKxaBXML_qL7g | t
 40 | name3  | $2b$10$f3RWKArGeJl1TuxZE6uGjeY7JwIWbP2Pw9Ipqa0rcBoFmrlJmT8AC | name3@gmail.com     | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDAsImlhdCI6MTY5OTE5MzYxMX0.ueaVo0JsTDMVbVzJjWIvfjTLXsWvxQNj1J3IYZilU9s | t
 41 | name10 | $2b$10$qa0.D4JMbG2v0LjErNjVeuA/ypVguGshMxfswXy88lipwjnDTKzb2 | name10@gmail.com    |                                                                                                           | f
 42 | name11 | $2b$10$fEQhALRQRYudqrfGVyFFe.WLdS9J4P/qGpFEYkGa48fR4EM5dOl3C | name11@gmail.com    |                                                                                                           | f
 43 | name12 | $2b$10$Y.k6rYsZiJYHSztjgGVIOeC/UeaQ8DRdv4xTZXcE8tvy1ynBp1cFS | name12@gmail.com    |                                                                                                           | f
 47 | name16 | $2b$10$jE1iKEQir56L.Z9UHunEt.hkNa40.ObQNjYZjK84ZJyZfzs1lIvum | name16@gmail.com    | eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDcsImlhdCI6MTY5OTI4NDgxNX0.Y1fxIKs85OIkYXyeeKU6e3DmexgNto9DdjSJbb2uejs | t
 44 | name13 | $2b$10$FEqpV/XGxNVCpYCPIivC8u7bKpHNVlVl6Buy7KLN65p7/UTSibmTe | name13@gmail.com    |                                                                                                           | f
 45 | name14 | $2b$10$kKeaXcDxGcCfAdckhER42uDFP6cL8842JUg2VQ8xHnRbmNT5.Ve4S | name14@gmail.com    |                                                                                                           | f
 46 | name15 | $2b$10$MrodljN0G9RkZnNxend44.yZxZLTyaMmn9JAa9LMH175YpkxDw9bm | name15@gmail.com    |                                                                                                           | f
(12 rows)
```

Рисунок 3.2 – Вид таблиці users

Склад таблиці accounts у таблиці 3.2.

Таблиця 3.2 – Склад таблиці accounts

№	Неймінг	Тип	Ознака
1	id	uuid	PK
2	account_name	varchar	-
3	owner_id	integer	PK

Опис кожного неймінгу більш конкретно:

– id – id рахунку, в PostgreSQL є вбудована функция UUID, яка генерує унікальну строку, це поле використовується для з’єднання з таблицями revenues та expenses;

– account_name – назва рахунку;

– owner_id – це ключ-посилання на власника цього рахунку.

З’єднана з таблицями revenues та expenses через значення id – account_id. Вид таблиці приведено на рисунку 3.3.

```
financial_transactions=# select * from accounts
financial_transactions-# ;
                  id                  | account_name | owner_id
--------------------------------------+--------------+----------
 542a75d8-2efd-4a53-aee8-de18c0c7d75d | account-38-2 |       38
 70c2ab60-f5c5-469a-b22d-38d63c7ae21f | account-3    |       38
 40bf7ffc-0944-4cc7-9461-c2398ae8c539 | account-4    |       38
 6871a821-04ba-4a1b-997d-4ed2b6812ac6 | account-1-31 |       49
 bd8bf847-b452-40da-ad88-26c43e4ed8ec | account-1-32 |       49
 47c088e6-7bec-4700-94e9-de4a76a16c6f | account-123  |       38
(6 rows)
```

Рисунок 3.3 – Вид таблиці accounts

Склад таблиці revenues у таблиці 3.3.

Таблиця 3.3 – Склад таблиці revenues

№	Неймінг	Тип	Ознака
1	id	uuid	PK
2	amount	integer	-
3	transaction_date	TIMESTAMP	-
4	transaction_description	VARCHAR(200)	-
5	account_id	uuid	PK

Таблиці revenues та expenses мають однакову структуру, різниця між ними revenues це доходи, expenses це витрати.

– id – id транзакції, також генерується за допомогою функції UUID;

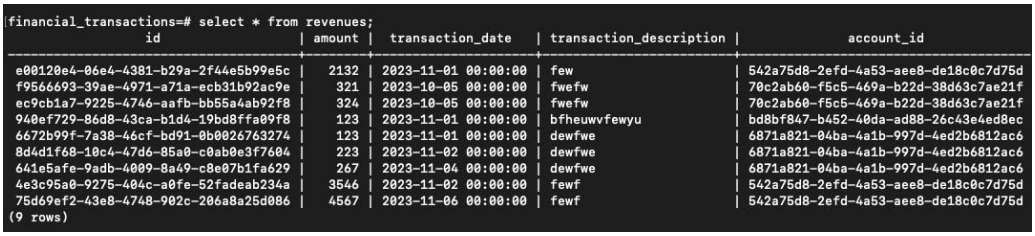
– amount – поле для зберігання суми транзакції;

– transaction_date – дата виконання транзакції;

– transaction_description – опис транзакції;

– account_id – поле, яке прив’язується до рахунку користувача.

Вид таблиці приведено на рисунку 3.4.



```
financial_transactions=# select * from revenues;
                  id                  | amount |  transaction_date   | transaction_description |              account_id
--------------------------------------+--------+---------------------+-------------------------+--------------------------------------
 e00120e4-06e4-4381-b29a-2f44e5b99e5c |   2132 | 2023-11-01 00:00:00 | few                     | 542a75d8-2efd-4a53-aee8-de18c0c7d75d
 f9566693-39ae-4971-a71a-ecb31b92ac9e |    321 | 2023-10-05 00:00:00 | fwefw                   | 70c2ab60-f5c5-469a-b22d-38d63c7ae21f
 ec9cb1a7-9225-4746-aafb-bb55a4ab92f8 |    324 | 2023-10-05 00:00:00 | fwefw                   | 70c2ab60-f5c5-469a-b22d-38d63c7ae21f
 940ef729-86d8-43ca-b1d4-19bd8ffa09f8 |    123 | 2023-11-01 00:00:00 | bfheuwvfewyu            | bd8bf847-b452-40da-ad88-26c43e4ed8ec
 6672b99f-7a38-46cf-bd91-0b0026763274 |    123 | 2023-11-01 00:00:00 | dewfwe                  | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
 8d4d1f68-10c4-47d6-85a0-c0ab0e3f7604 |    223 | 2023-11-02 00:00:00 | dewfwe                  | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
 641e5afe-9adb-4009-8a49-c8e07b1fa629 |    267 | 2023-11-04 00:00:00 | dewfwe                  | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
 4e3c95a0-9275-404c-a0fe-52fadeab234a |   3546 | 2023-11-02 00:00:00 | fewf                    | 542a75d8-2efd-4a53-aee8-de18c0c7d75d
 75d69ef2-43e8-4748-902c-206a8a25d086 |   4567 | 2023-11-06 00:00:00 | fewf                    | 542a75d8-2efd-4a53-aee8-de18c0c7d75d
(9 rows)
```

Рисунок 3.4 – Вид таблиці revenues

Склад таблиці expenses у таблиці 3.4.

Таблиця 3.4 – Склад таблиці expenses

№	Неймінг	Тип	Ознака
1	id	uuid	PK
2	amount	integer	-
3	transaction_date	TIMESTAMP	-
4	transaction_description	VARCHAR(200)	-
5	account_id	uuid	PK

Вид таблиці приведено на рисунку 3.5.

```
financial_transactions=# select * from expenses;
                  id                  | amount |  transaction_date   | transaction_description |              account_id
--------------------------------------+--------+---------------------+-------------------------+--------------------------------------
 728eb230-0805-4532-af80-3878bbb0c0af |    123 | 2023-11-01 00:00:00 | feww                    | 542a75d8-2efd-4a53-aee8-de18c0c7d75d
 693f4ff1-3072-4788-9fbe-0e0017411ac1 |    333 | 2023-11-01 00:00:00 | fewwfewfw               | 542a75d8-2efd-4a53-aee8-de18c0c7d75d
 e8144880-9b30-4b95-a9e4-426dbc93477e |    324 | 2023-10-05 00:00:00 | fwefw                   | 70c2ab60-f5c5-469a-b22d-38d63c7ae21f
 de12dd63-6f3d-436b-8f51-4c052d9aa35d |    424 | 2023-10-05 00:00:00 | fwefw                   | 70c2ab60-f5c5-469a-b22d-38d63c7ae21f
 806a555d-51a4-40ea-928b-327f1d26d069 |    267 | 2023-11-04 00:00:00 | dewfwe                  | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
 72628365-4e77-472c-93a1-ed2759f4e9e7 |    123 | 2023-11-08 00:00:00 | dewfwe                  | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
 02f2df1b-500b-4d9b-ac8b-fd08ffb22274 |    324 | 2023-11-08 00:00:00 | fewf                    | 6871a821-04ba-4a1b-997d-4ed2b6812ac6
(7 rows)
```

Рисунок 3.5 – Вид таблиці expenses

Склад таблиці messages_ai у таблиці 3.5.

Таблиця 3.5 – Склад таблиці

№	Неймінг	Тип	Ознака
1	id	uuid	PK
2	text	VARCHAR(10000)	-
3	send_time	TIMESTAMP	-
4	owner_id	integer	PK

Таблиці messages_ai та messages_user мають однаковий склад. Головна різниця у тому, що таблицю messages_ai заповнює штічний інтелект, а messages_user зопвнюється запросом користовуча.

– id – id транзакції, також генерується за допомогою функції UUID;

– text – поле самого запросу(messages_user) чи відповіді (messages_user);

– send_time – дата і час запиту;

– owner_id – це ключ-посилання на власнику.

Вид таблиці приведено на рисунку 3.6.

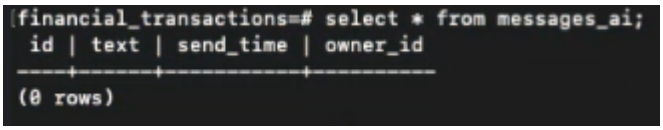


Рисунок 3.6 – Вид таблиці messages_ai

Склад таблиці messages_user у таблиці 3.6.

Таблиця 3.6 – Склад таблиці

№	Неймінг	Тип	Ознака
1	id	uuid	PK
2	text	VARCHAR(10000)	-
3	send_time	TIMESTAMP	-
4	owner_id	integer	PK

Вид таблиці з запитом від юзера приведено на рисунку 3.7.

Рисунок 3.7 – Вид таблиці messages_user

Токен – це строка, яка складається з 3 частин. Вони розділені точкою. В кожній з цих частин знаходиться інформація о користувачі. В першій частині зашифровано id користувача. В другій – секретне слово (secret key). В третій срок життя токену. В нашому проекті він вічний. Для хешування використовував бібліотеку bcrypt.

Хешування – це процес перетворення даних змінної довжини у фіксований рядок фіксованої довжини. Результат цього перетворення називається "хешем". Хеш-функції широко використовуються в криптографії, базах даних, структурах даних і в різних алгоритмах для забезпечення швидкого пошуку даних.

4 РОБОТА ПРОГРАМИ

Після запуску потрапляємо у вікно регістрації. Справа внизу можна вибрати вікно логінізації (рис. 4.1).

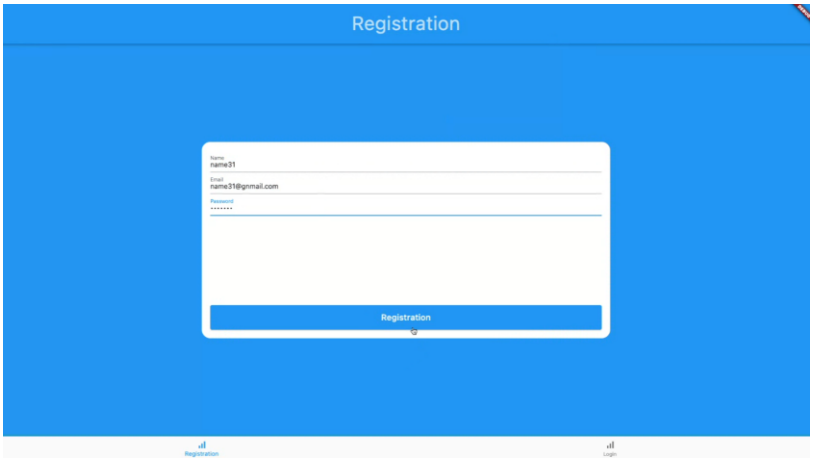


Рисунок 4.1 – Вікно регістрації

Після входу потрапляємо у меню, де можна створити або обрати сторінку з рахунками відділів (рис. 4.2).

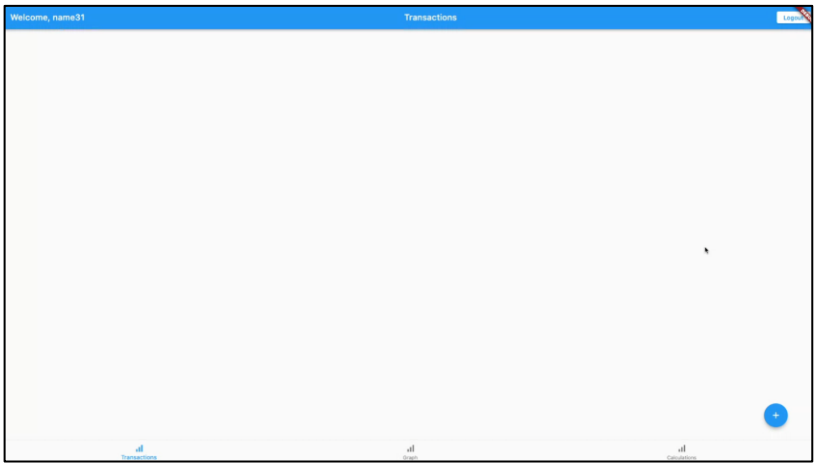


Рисунок 4.2 – Меню

Створимо відділ (рис. 4.3).

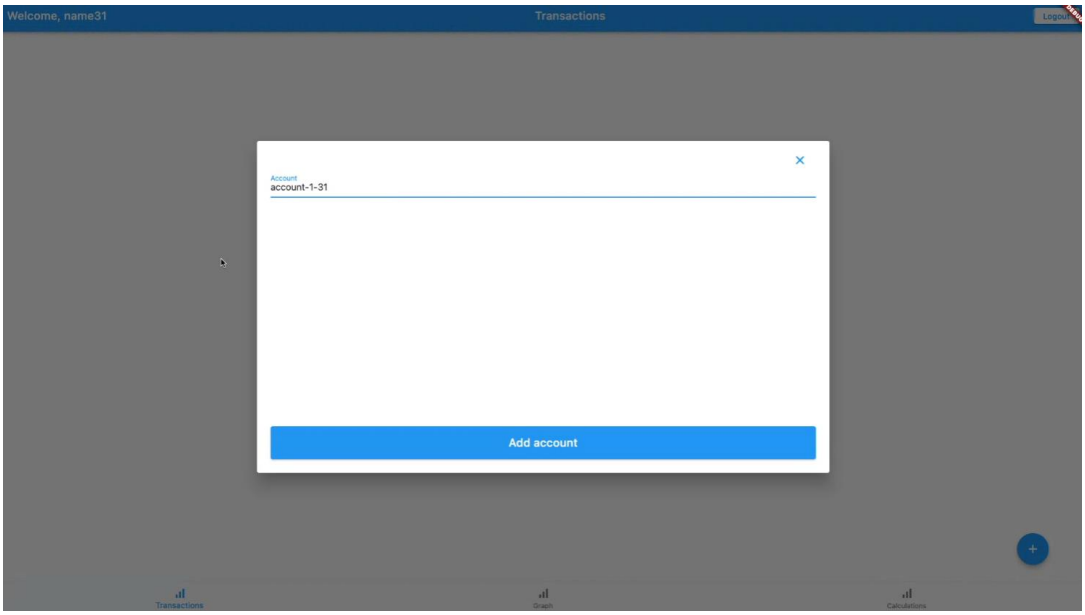


Рисунок 4.3 – Створення відділу

Створений відділ відображається зверху. Можна зробити велику кількість відділів (рис. 4.4).

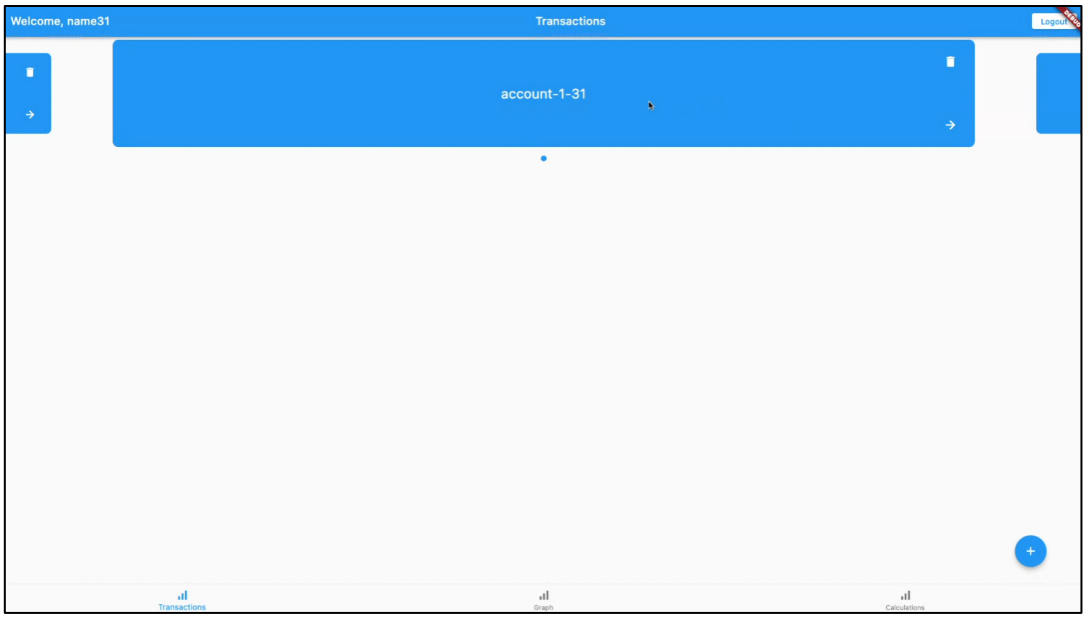


Рисунок 4.4 – Створений відділ

Заходимо у рахунки відділу. Зверху є шапка вибору надходжень та витрат. (рис. 4.5).

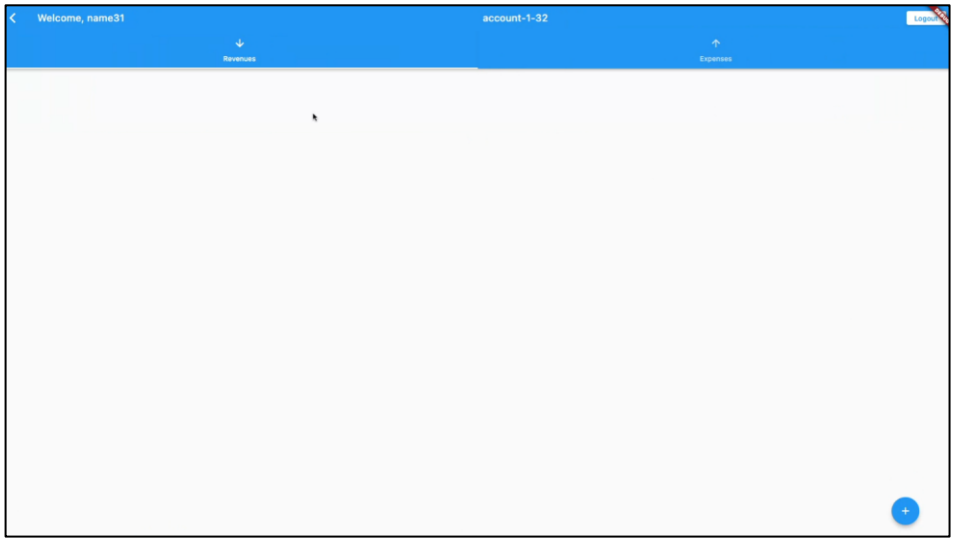


Рисунок 4.5 – Рахунки відділу

Створемо надходження (рис. 4.6).

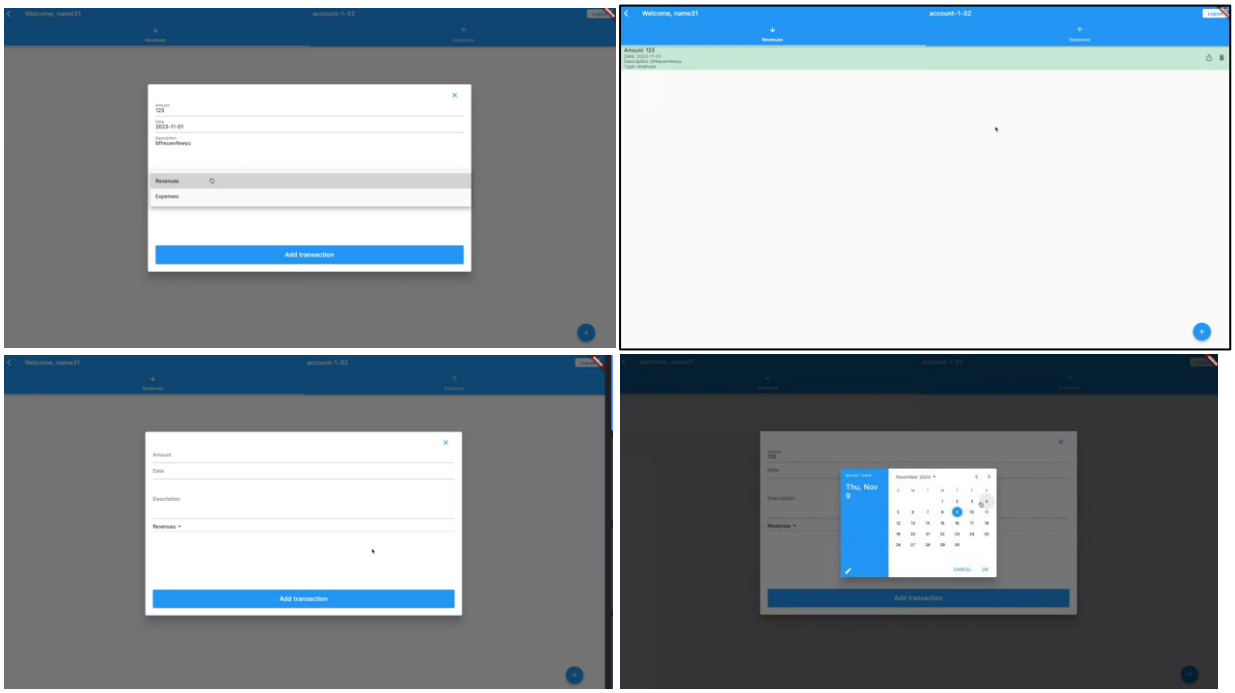


Рисунок 4.6 – Створення надходження

Створеним надходженням або тратою можна поділитись. Наприклад, у телеграмі (рис. 4.7).

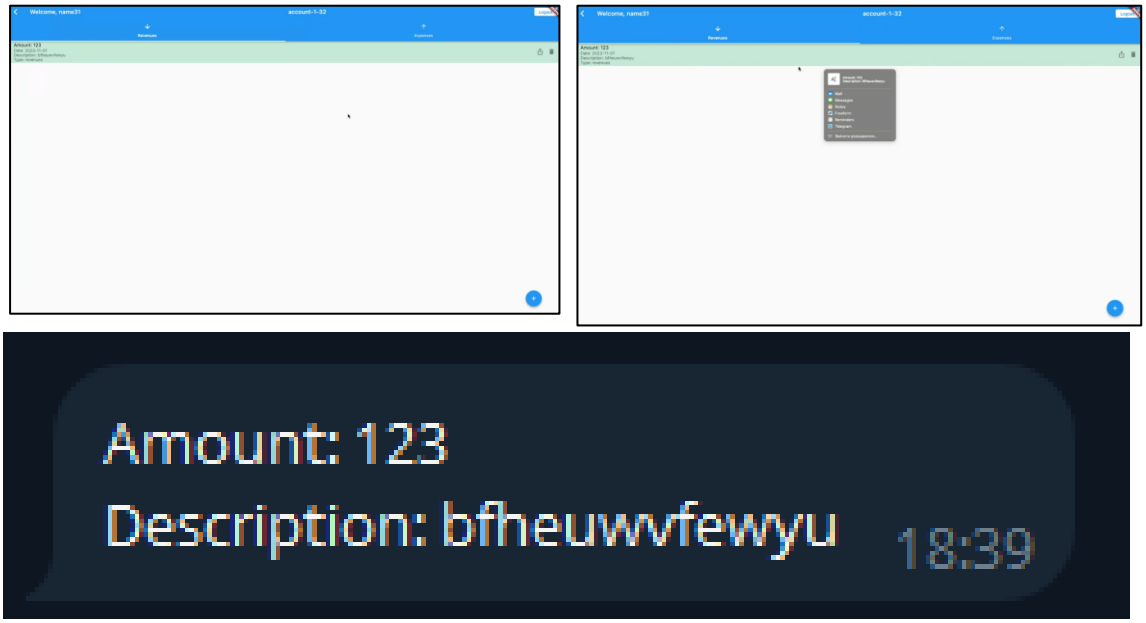


Рисунок 4.7 – Демонстрація роботи функції «поділитись»

Також його можна видалити (рис. 4.8).

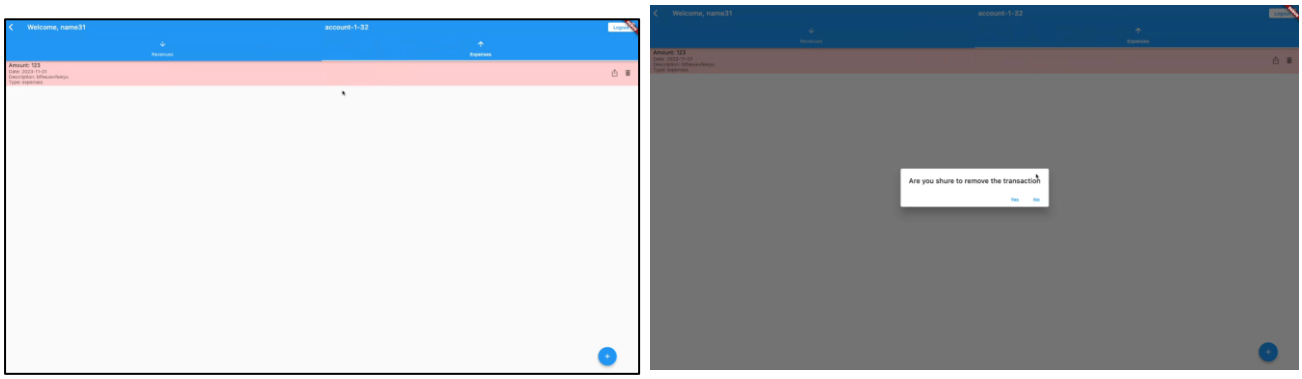


Рисунок 4.8 – Видалення

У головному меню є пункт Graph. В цьому пункті в графічному вигляді можливо подивитись на трати або надходження, їх сортировано по даті (рис. 4.9).

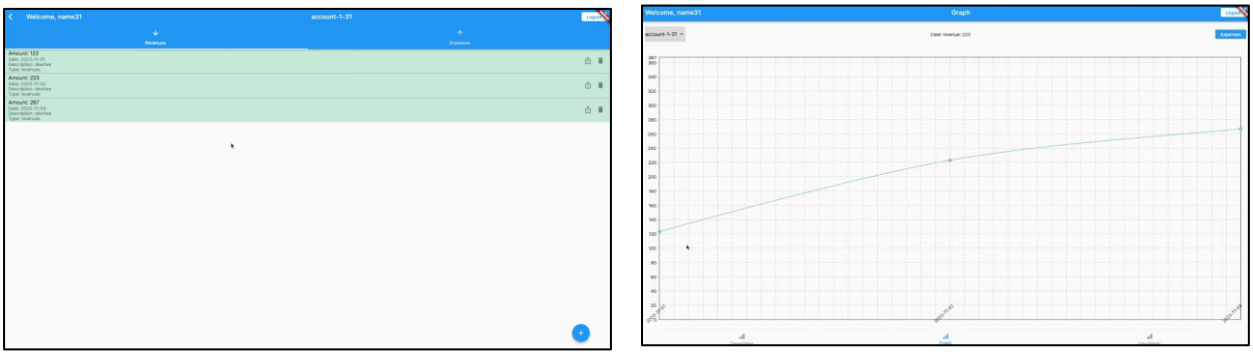


Рисунок 4.9 – Графік

Во вкладці chat ai при першому заході штучній інтелект пропонує шляхи для оптимізації (рис. 4.10).

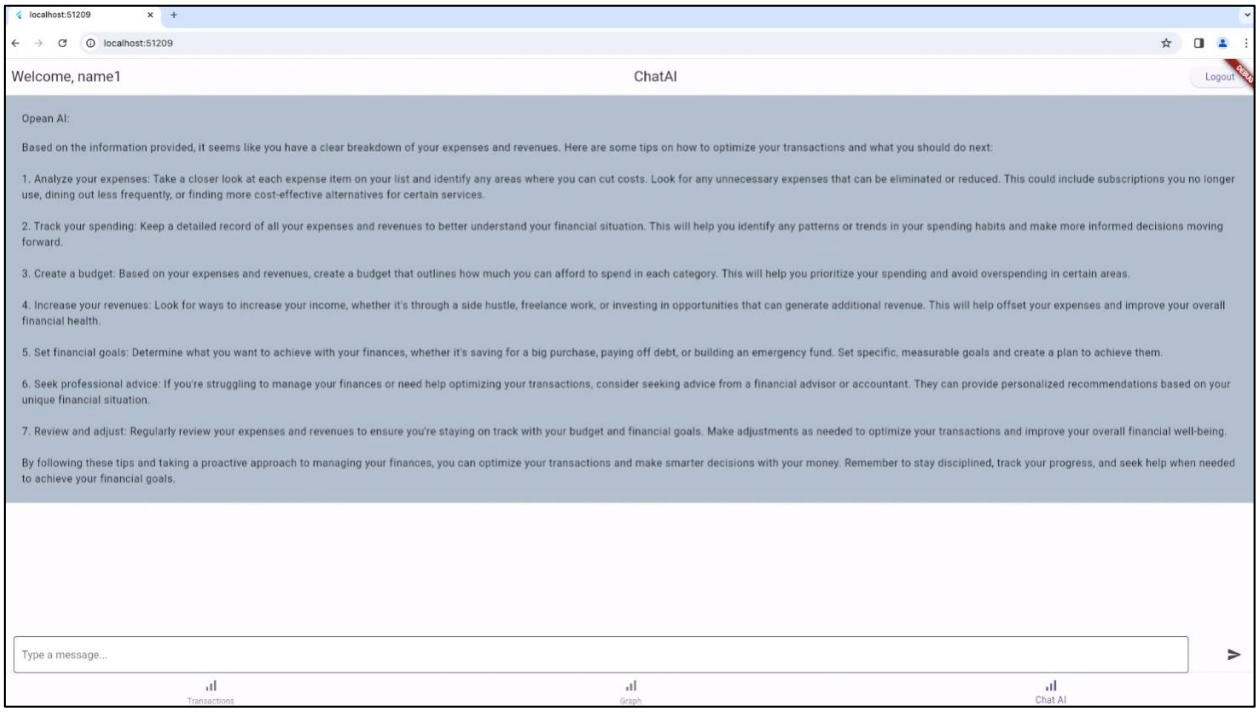


Рисунок 4.10 – Перша відповідь ШІ

Для штучного інтелекту формується такий запит:

```
final expenses = descriptions.expenses == ""
    ? "I don't have any expenses"
    : descriptions.expenses;
final revenues = descriptions.revenues == ""
    ? "I don't have any revenues"
    : descriptions.revenues;
final message =
    "I have a list of expenses: $expenses, also I have a list of revenues:
$revenues. Write me tips based on my transactions, how to optimize my
transactions and what should I do next?";
appBloc.chat
    .sendMessage(message: message, typeMessage: "sendMessage")
```

де expenses це витрати, revenues це надходження. На основі цих данний штучний інтелект формує запрос. Це повноціий штучній інтелоект, который може відповісти на будь-яке запитання. Це приведено на рис.4.11 та рис. 4.12.

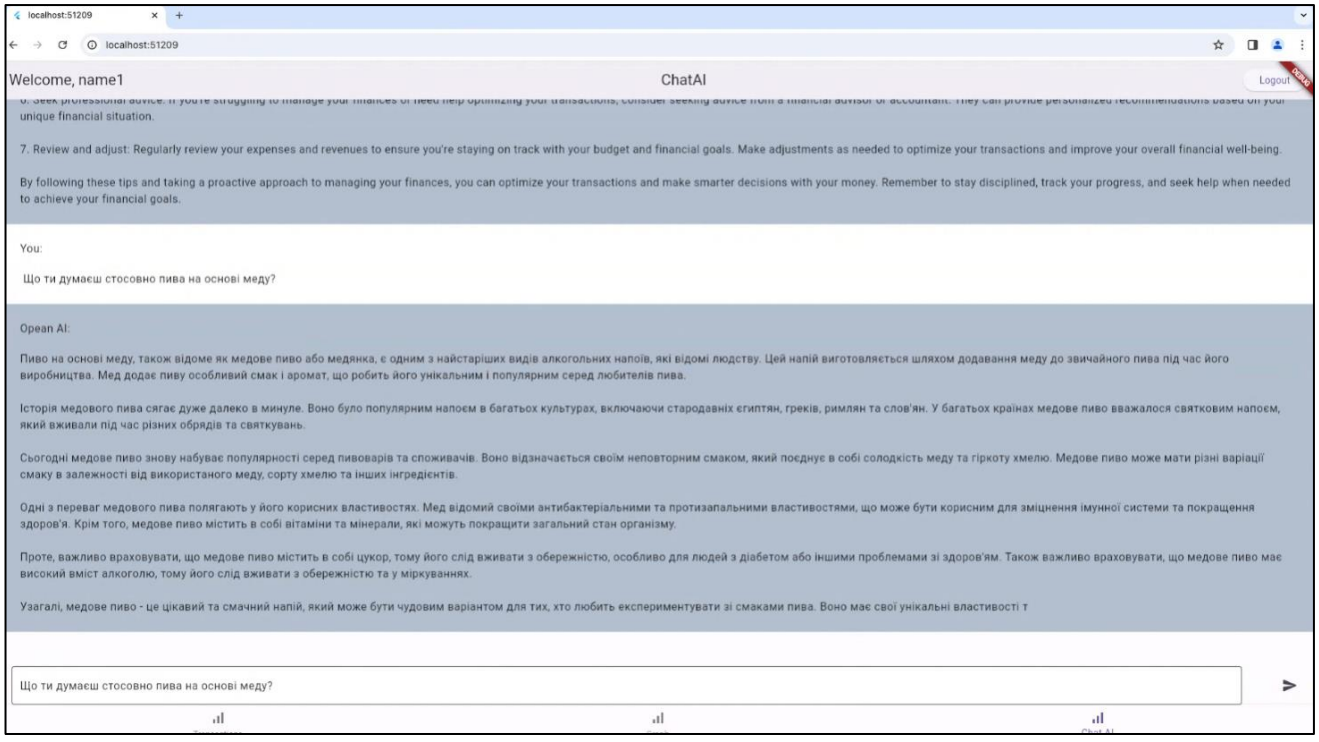


Рисунок 4.11 – Відповідь ШІ

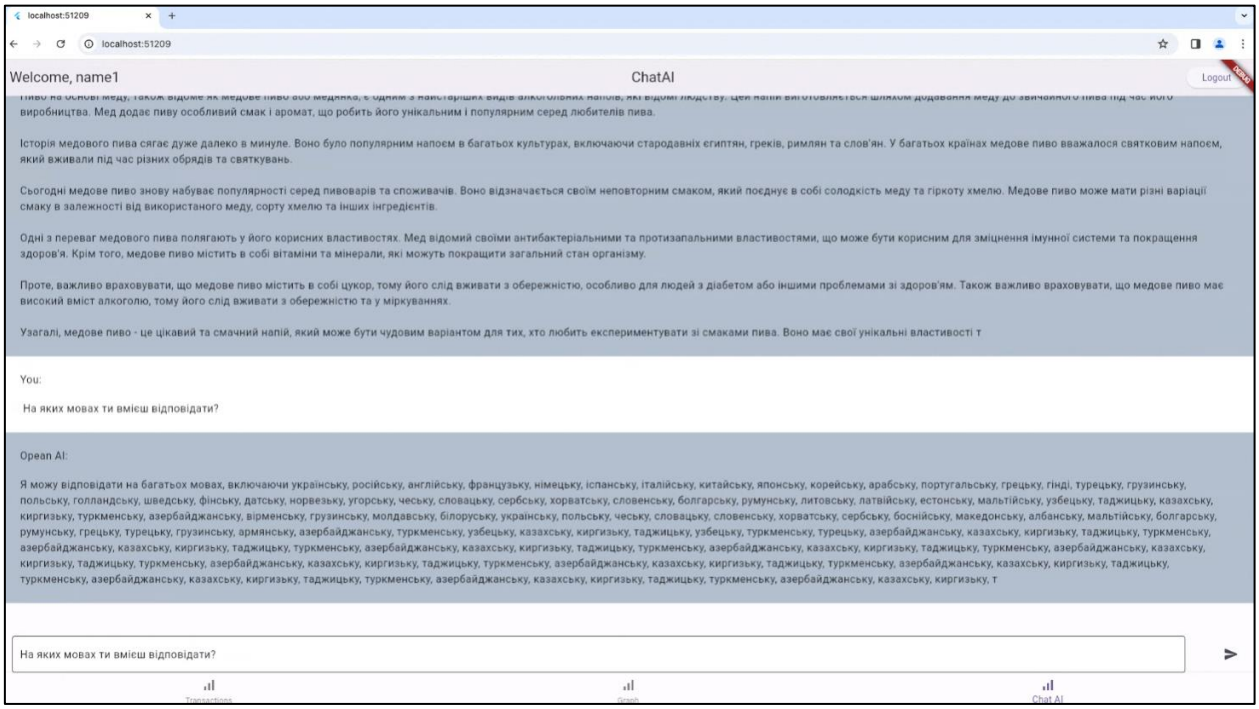


Рисунок 4.12 – Відповідь ШІ

Метод запиту має такий код:

```
Future<Message> sendMessage({
  required String message,
  required String typeMessage,
 }) async {
  try {
    String? token = await LocalStorageApi.getToken();
    dio.options.headers['Authorization'] = 'Bearer $token';

    final data = {
      "text": "Use maximum 10000 characters: $message",
    };

    Response<Map<String, dynamic>> response =
        await dio.post("$baseUrl/$typeMessage", data: data);
```

```
    if (response.data != null) {
      final responseFromAIData = response.data as Map<String, dynamic>;

      return Message(
        id: responseFromAIData["id"],
        text: responseFromAIData["text"],
        owner_id: responseFromAIData["owner_id"],
        send_time: responseFromAIData["send_time"],
        type: typeMessage == "writeQuestion" ? "userMessage" : "aiMessage",
      );
    } else {
      throw Exception('Failed to send message');
    }
  } catch (error) {
    debugPrint(error.toString());
    throw Exception('Failed to send message');
  }
}
```

5 ОХОРОНА ПРАЦІ

В сучасному світі, де технології та інформаційна безпека набувають все більшого значення, важливо не забувати про важливість охорони праці в процесі розробки та використання програмного забезпечення, зокрема додатків для фінансового обліку компаній. Запровадження ефективних заходів з охорони праці у процесі створення та експлуатації цього додатку є невід'ємною складовою забезпечення успішності його функціонування та збереження здоров'я й безпеки користувачів. ключові аспекти охорони праці, які необхідно врахувати у процесі розробки та використання додатку для ведення фінансової операції компанії це аналіз ризиків та визначення заходів безпеки, організація робочого місця та робочого процесу, навчання та інструктаж користувачів, моніторинг та аналіз безпеки та документування та аудит безпеки.

5.1 Аналіз ризиків та визначення заходів безпеки

Аналіз ризиків та визначення заходів безпеки є критичним етапом у забезпеченні безпечності та стабільності додатку для ведення фінансової операції компанії. Цей процес включає у себе ідентифікацію потенційних загроз безпеці даних, можливих вразливостей програмного забезпечення, а також ризиків, пов'язаних із здоров'ям користувачів. На основі отриманих результатів аналізу ризиків формулюються та впроваджуються відповідні заходи безпеки, які включають в себе розробку механізмів контролю доступу, використання шифрування даних, регулярні аудити безпеки та забезпечення належного навчання користувачів щодо правильного та безпечного використання додатку. Це дозволяє зменшити ризики виникнення інцидентів безпеки та забезпечує надійність та конфіденційність фінансової інформації компанії.

5.2 Організація робочого місця та робочого процесу

Організація робочого місця та робочого процесу включає в себе впровадження ергономічних принципів у дизайні інтерфейсу додатку для мінімізації ризику виникнення травм або струсу для користувачів. Це також охоплює забезпечення належної вентиляції та освітлення на робочому місці для підтримки здоров'я користувачів. Ефективна організація робочого процесу включає у себе розробку чітких та зрозумілих інструкцій щодо коректної роботи з додатком, а також створення умов для зручного доступу до необхідної інформації та інструментів. Це сприяє підвищенню продуктивності та забезпечує комфортні умови для користувачів під час роботи з додатком для ведення фінансової операції компанії.

5.3 Навчання та інструктаж користувачів

Навчання та інструктаж користувачів є важливою складовою безпечного та ефективного використання додатку для ведення фінансової операції компанії. Цей процес включає проведення обов'язкового навчання для користувачів з правильного та безпечного використання програми, викладення детальних інструкцій щодо процедур поводження з конфіденційною інформацією, а також надання рекомендацій з використання заходів безпеки даних. Постійна підтримка та надання доступу до ресурсів для користувачів також є важливою частиною цього процесу, що допомагає забезпечити їхню готовність та впевненість у роботі з додатком. Такий підхід сприяє зменшенню ризиків помилок та недорозумінь, а також забезпечує ефективне використання програмного забезпечення для досягнення поставлених цілей у фінансовому обліку компанії.

5.4 Моніторинг та аналіз безпеки

Моніторинг та аналіз безпеки є важливим етапом в забезпеченні надійності та безпеки додатку для ведення фінансової операції компанії. Цей процес включає в себе постійне відстеження подій безпеки та аналіз інцидентів з метою вчасного виявлення та усунення потенційних вразливостей. Регулярні аудити безпеки та оновлення заходів безпеки відповідно до нових загроз та вимог безпеки дозволяють забезпечити стійкість додатку до потенційних атак та забезпечити безпеку фінансової інформації компанії. Цей процес сприяє підвищенню рівня захищеності додатку та забезпечує впевненість користувачів у безпеці їхніх даних.

5.5 Документування та аудит безпеки

Цей процес включає оцінку ймовірності виникнення потенційних загроз та наслідків, які можуть випливати з цих загроз. Ймовірність враховує імовірність виникнення конкретної загрози, в той час як наслідки оцінюють масштаб шкоди, яку ця загроза може спричинити. Це дозволяє ідентифікувати найбільш критичні ризики та визначити пріоритетність заходів безпеки для їх запобігання або зменшення впливу. Врахування цих факторів допомагає забезпечити адекватний рівень захисту додатку та його користувачів від можливих загроз.

Ефективне впровадження вищезазначених заходів забезпечить стабільну та безпечну роботу додатку для ведення фінансової операції компанії, зменшуючи ризики виникнення проблем з безпекою даних та забезпечуючи здоров'я та безпеку користувачів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було успішно виконано поставлене завдання. У першому пункті обрано мову та технології, котрі використовувались під час виконання завдання. Також було приведено приклади різних технологій, і з них обрано найкращу для використання саме у цьому завдання, приведено короткі відомості про технології. У другому розділі приведено алгоритм роботи програми та створена блок схема. У третьому розділі продемонстровано вигляд бази даних, котра складається з чотирьох таблиць. В четвертому розділі розписано функціонал програми. Завдання виконано успішно.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

2. Невлюдов І. Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 171 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова. – Київ: НАУ, – 2016. – 320 с.

3. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 172 Телекомунікації та радіотехніка, освітньої програми «Інтелектуальні технології засобів радіоелектроніки» / Упоряд. Н. П. Демська, В. В. Євсєєв, О. М. Замірець, В. В. Невлюдова, Ю. М. Олександров. Харків : ХНУРЕ, 2022. 38 с.

4. ТОП10 мов програмування [Електроний ресурс]. – Режим доступу: http://apeps.kpi.ua/top10-mov-programuvania-2019 – 18.3.2024 р.

5. JavaScript [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/JavaScript – 18.3.2024 р.

6. TypeScript [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/TypeScript – 18.3.2024 р.

7. Що таке база даних? [Електроний ресурс]. – Режим доступу: https://apeps.kpi.ua/shco-take-basa-danykh – 18.3.2024 р.

8. Postgresql [Електроний ресурс]. – Режим доступу: https://www.postgresql.org/ – 18.3.2024 р.

9. Front end та back end [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Front_end_%D1%82%D0%B0_back_end – 18.3.2024 р.

10. Backend-розробка [Електроний ресурс]. – Режим доступу: https://mc.tod ay/uk/shho-take-backend-rozrobka/ – 18.3.2024 р.

11. Node.js [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Node.js – 18.3.2024 р.

12. Node.js [Електроний ресурс]. – Режим доступу: https://nodejs.org/en – 18.3.2024 р.

13. Express.js [Електроний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Express.js – 18.3.2024 р.

14. Express.js [Електроний ресурс]. – Режим доступу: https://expressjs.com/ – 18.3.2024 р.

15. front-end [Електроний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/how-to-front-end/ – 18.3.2024 р.

16. Flutter [Електроний ресурс]. – Режим доступу: https://flutter.dev/ – 18.3.2024 р.

17. Додатки фінансів [Електроний ресурс]. – Режим доступу: https://www.moyo.ua/ua/news/luchshie_prilozheniya_dlya_ucheta_finansov_v_2024_godu_9_interfeysov_dlya_tochnogo_planirovaniya_kazhdoy_kopeechki.html – 18.3.2024 р.

18. monefy.me [Електроний ресурс]. – Режим доступу: https://monefy.me/ – 18.3.2024 р.

19. ASP.NET MVC [Електроний ресурс]. – Режим доступу: https://learn.microsoft.com/ua-ua/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview – 18.3.2024 р.

20. Штучний інтелект [Електроний ресурс]. – Режим доступу: https://osvita.diia.gov.ua/courses/artificial-intelligence – 18.3.2024 р.

21. Про охорону праці : Закон України від 14.10.1992 р. No 2694-XII : станом на 1 жовт. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text –18.03.2024.