ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МОДЕЛЬ СИСТЕМИ АНАЛІЗУ ПЕРСОНАЛЬНИХ ФІНАНСІВ З ІНТЕГРОВАНИМИ AI ТЕХНОЛОГІЯМИ»

на здобуття освітнього ступені магістра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Юрій БОНДАРЕНКО

Виконав:
здобувач вищої освіти
група ІСДМ-61 Юрій БОНДАРЕНКО

Керівник:
науковий ступінь, вчене звання Валентина ДАНИЛЬЧЕНКО
доктор філософії

Рецензент:
науковий ступінь, вчене звання _____
Ім’я, ПРІЗВИЩЕ

Київ - 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ

_____ Каміла Сторчак

«____» _____20__ року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ

Бондаренко Юрію Леонідовичу

(прізвище, ім‘я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Модель системи аналізу персональних фінансів з інтегрованими аі технологіями».

Керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, доктор філософії,

(ім‘я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи 27 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
Голосові повідомлення з різною тривалістю та якістю запису.
Категорії фінансових транзакцій (продукти, транспорт, розваги тощо).
Суми транзакцій (у різних форматах: цифрами, словами, змішані).
Метадані транзакцій: дати, описи, категорії.
Інфраструктура для тестування.
Відкриті API для розпізнавання голосу (Google Speech-to-Text).
Науково-технічна література з питань AI, розпізнавання мови та обробки персональних фінансів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд сучасних AI-технологій для аналізу персональних фінансів.
2. Аналіз інструментів для розпізнавання голосових даних.
3. Вибір архітектури системи (порівняння монолітної та мікросервісної архітектури).
4. Реалізація бекенду для обробки аудіо- та текстових даних.
5. Тестування точності розпізнавання тексту з голосових повідомлень.
6. Аналіз результатів роботи системи з реальними даними.

5. Перелік ілюстративного матеріалу: *презентація*
Архітектура веб-додатку.
Схема веб-додатку
Інтерфейс користувача

6. Дата видачі завдання 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	15.10 - 23.10.24	Виконано
2	Вибір технологій для реалізації системи	25.10 - 01.11.24	Виконано
3	Проектування архітектури системи	03.11 - 07.11.24	Виконано
4	Розробка та інтеграція API для розпізнавання голосу	07.11 - 11.11.24	Виконано
5	Реалізація бекенду на базі Spring Boot та налаштування PostgreSQL	11.11 - 18.11.24	Виконано
6	Створення фронтенду для взаємодії користувача	18.11 - 22.11.4	Виконано
7	Тестування функціоналу та точності розпізнавання тексту	22.11 - 25.11.24	Виконано
8	Оптимізація продуктивності та масштабованості	25.11 - 30.11.24	Виконано
9	Оформлення роботи: вступ, висновки, реферат	01.12 - 15.12.24	Виконано
10	Підготовка демонстраційних матеріалів	15.12 - 25.12.24	Виконано

Здобувач вищої освіти _____ Юрій БОНДАРЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник _____ Валентина ДАНИЛЬЧЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

кваліфікаційної роботи

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступення магістра: 81 стор., 11 рис., 33 джерел.

Мета роботи - **р**озробка та впровадження системи аналізу персональних фінансів з інтеграцією AI-технологій, яка включає автоматизацію обробки голосових повідомлень, перетворення їх у фінансові транзакції та забезпечення користувачів аналітичними даними для ефективного управління бюджетом.

Об'єкт дослідження - система аналізу персональних фінансів із використанням AI-технологій, яка включає інтеграцію інструментів розпізнавання мови, базу даних PostgreSQL, бекенд на базі Spring Boot і фронтенд для роботи користувачів.

Предмет дослідження - методи і алгоритми обробки аудіоданих, розпізнавання тексту з голосових повідомлень, збереження інформації у структурованій базі даних, аналіз транзакцій, а також засоби побудови масштабованих і безпечних інформаційних систем.

Короткий зміст роботи: Досліджено сучасні підходи до аналізу персональних фінансів із використанням AI-технологій, що включають автоматизацію введення даних за допомогою обробки голосових повідомлень. Виокремлено основні функції системи, зокрема обробку аудіоданих, розпізнавання тексту, збереження фінансових транзакцій у базі даних, їх аналіз та інтерактивне відображення результатів для користувачів. Визначено способи інтеграції з інструментами розпізнавання мови та реалізовано механізми забезпечення безпеки даних.

КЛЮЧОВІ СЛОВА: ПЕРСОНАЛЬНІ ФІНАНСИ, АНАЛІЗ ТРАНЗАКЦІЙ, AI-ТЕХНОЛОГІЇ, ОБРОБКА ГОЛОСОВИХ ДАНИХ, GOOGLE SPEECH-TO-TEXT, JAVA SPRING BOOT, POSTGRESQL, AWS S3, REDIS, ВЕБ-ІНТЕРФЕЙС, ФРОНТЕНД, МАСШТАБОВАНІСТЬ, МОНІТОРИНГ ТА УПРАВЛІННЯ, БЕЗПЕКА ДАНИХ, РОЗПІЗНАВАННЯ ТЕКСТУ, WER.

ABSTRACT

The text part of the master’s qualification work: 81 pages, 11 pictures, 33 sources.

The purpose of the work - the development and implementation of a personal finance analysis system with AI technology integration, including the automation of voice message processing, converting them into financial transactions, and providing users with analytical data for efficient budget management. The system must include mechanisms for processing audio files, accurate text recognition, transaction analysis, and an interactive interface for data visualization.

Object of research - the personal finance analysis system using AI technologies, which includes the integration of speech recognition tools, a PostgreSQL database, a Spring Boot-based backend, and a user-oriented frontend.

Subject of research - methods and algorithms for processing audio data, text recognition from voice messages, storing information in a structured database, transaction analysis, and tools for building scalable and secure information systems.

Summary of the work: Modern approaches to personal finance analysis using AI technologies were studied, including the automation of data entry through voice message processing. The system's main functions were identified, including audio data processing, text recognition, storing financial transactions in the database, their analysis, and interactive visualization for users. Methods for integrating speech recognition tools were defined, and data security mechanisms were implemented.

KEYWORDS: PERSONAL FINANCE, TRANSACTION ANALYSIS, AI TECHNOLOGIES, VOICE DATA PROCESSING, GOOGLE SPEECH-TO-TEXT, JAVA SPRING BOOT, POSTGRESQL, AWS S3, REDIS, WEB INTERFACE, FRONTEND, SCALABILITY, MONITORING AND MANAGEMENT, DATA SECURITY, TEXT RECOGNITION, WER..

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі штучний інтелект відіграє ключову роль у багатьох сферах життя, включаючи фінансовий сектор. Ефективний контроль над персональними фінансами є одним із важливих факторів забезпечення фінансової стабільності та успішного планування витрат. Використання AI-технологій для автоматизації процесів управління фінансами дозволяє значно спростити взаємодію користувача із системою, забезпечуючи швидке та зручне введення і обробку фінансових даних.

Окремий інтерес викликає застосування технологій розпізнавання голосу для конвертації тексту з голосових повідомлень у структуровані фінансові дані. Цей підхід дозволяє користувачам автоматизувати рутинний процес внесення витрат та оптимізувати час на управління фінансами. Попри існування фінансових додатків, більшість із них використовують традиційні методи введення даних, такі як ручне внесення або інтеграція з банківськими API. Використання AI для голосової обробки залишається недостатньо дослідженим напрямком, що обґрунтовує актуальність даної роботи для наукового середовища та практичного впровадження, особливо в Україні, де ринок інноваційних фінансових рішень активно розвивається.

Аналіз останніх досліджень і публікацій. У світі активно розвиваються технології штучного інтелекту, зокрема в галузі обробки природної мови (Natural Language Processing) та розпізнавання голосу. Наукові дослідження, представлені авторами, такими як Гудфелоу І., Бенджіо Й. та Курвіла А., розглядають основи штучних нейронних мереж, що лежать в основі сучасних систем розпізнавання голосу. Google Speech-to-Text та інші інструменти активно застосовуються для автоматизації обробки голосових даних, проте їх використання у фінансовій сфері залишається обмеженим. У роботах зарубіжних дослідників, таких як Дж. Мендоса та А. Нгуєн, розглядаються питання інтеграції AI для фінансових аналітичних систем. Водночас недостатньо розроблені методи ефективної

конвертації голосових даних у фінансові транзакції, що вказує на необхідність подальших досліджень.

Метою роботи є розробка моделі системи аналізу персональних фінансів, що інтегрує AI-технології для розпізнавання тексту з голосових повідомлень і конвертації цих даних у структуровані фінансові об’єкти з подальшим їх аналізом.

Завдання дослідження:

- Проаналізувати сучасні методи та технології AI для розпізнавання голосового тексту.
- Дослідити потреби користувачів у контексті інтерактивного управління фінансами.
- Розробити концепцію моделі системи для автоматизації обробки фінансових даних.
- Реалізувати прототип системи, що включає модулі розпізнавання голосу, обробки тексту та інтеграції в систему обліку витрат.
- Провести тестування прототипу на відповідність вимогам точності, продуктивності та зручності користувача.

Об’єкт дослідження - процес створення системи аналізу персональних фінансів із використанням AI-технологій.

Предмет дослідження - методи обробки голосових повідомлень, технології розпізнавання тексту та інтеграція розпізнаних даних у систему обліку фінансових витрат.

Методи дослідження. У роботі застосовано методи системного аналізу для визначення вимог до системи, алгоритми машинного навчання (зокрема, нейронні мережі) для реалізації модуля розпізнавання голосу, методи програмної інженерії для створення прототипу системи на базі Java Spring Boot та PostgreSQL. Для тестування продуктивності використано інструменти навантажувального тестування та аналізу точності.

Наукова новизна отриманих результатів полягає у розробці моделі системи, що вперше інтегрує технології розпізнавання голосу для автоматичного внесення фінансових даних на основі голосових повідомлень. Удосконалено

методику обробки голосових даних для підвищення точності розпізнавання у складних умовах запису.

Апробація результатів та публікації:

1. Бондаренко Ю. Л. «Використання штучного інтелекту для аналізу персональних фінансів». Теза доповіді на II всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу».

2. Бондаренко Ю. Л. «Розробка інтелектуальної системи для планування бюджету на основі машинного навчання». Теза доповіді на II міжнародну науково-практичну конференцію «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні тенденції в аналізі персональних фінансів.

Управління персональними фінансами є однією з найважливіших навичок для досягнення фінансової стабільності та реалізації довгострокових планів. Завдяки стрімкому розвитку технологій цей процес стає більш ефективним, автоматизованим та доступним для широкої аудиторії. Серед головних тенденцій, що змінюють сферу фінансового обліку, можна виділити наступні:

1. Штучний інтелект у фінансовому менеджменті.

AI є основою багатьох сучасних інструментів, що забезпечують аналіз фінансових даних, прогнозування витрат та автоматизацію задач. Розпізнавання голосу й тексту є важливим кроком у спрощенні взаємодії користувачів із фінансовими системами.

2. Open Banking та інтеграція з банками.

Технології відкритого банкінгу через API дозволяють системам обліку автоматично отримувати дані про доходи й витрати користувача, генерувати звіти та аналізувати фінансовий стан у режимі реального часу.

3. Мобільні фінансові додатки.

Зручність мобільних застосунків зробила їх основним інструментом для обліку фінансів. Вони інтегруються з іншими сервісами (наприклад, календарями чи нагадуваннями) та пропонують персоналізовані функції, що відповідають потребам користувачів.

4. Голосові асистенти.

Використання голосових помічників дозволяє користувачам додавати витрати, отримувати звіти чи рекомендації за допомогою голосових команд, спрощуючи внесення даних.

5. Автоматизація заощаджень і прогнозування витрат.

AI аналізує попередні фінансові дані для передбачення витрат та пропонує індивідуальні стратегії заощаджень або бюджети, що допомагають досягти фінансових цілей.

6. Хмарні технології.

Зберігання даних у хмарі гарантує доступність інформації з будь-якого пристрою, підвищуючи безпеку даних завдяки надійному шифруванню та автоматичним резервним копіям.

7. Персоналізовані фінансові рішення.

Машинне навчання дозволяє створювати індивідуальні рекомендації, що враховують фінансові звички й цілі користувача. Це забезпечує точніше планування та підвищує мотивацію.

8. Автоматизація рутинних завдань.

Сучасні програми автоматично обробляють повсякденні фінансові задачі, такі як оплата рахунків, нагадування про платежі або створення звітів, що економить час.

9. Безпека та захист даних.

Зі зростанням кількості фінансової інформації підвищується необхідність у надійному захисті даних. Системи активно використовують багатофакторну аутентифікацію та сучасні методи шифрування.

1.2 Огляд існуючих рішень з інтегрованим штучним інтелектом.

Сучасні рішення для управління персональними фінансами активно використовують штучний інтелект для автоматизації задач, підвищення точності даних та надання персоналізованих рекомендацій. Розглянемо ключові види програмного забезпечення з AI, їхні переваги, можливості та обмеження.

1. Мобільні фінансові додатки з AI.

Популярні платформи, як YNAB[1] та Personal Capital[2], інтегрують штучний інтелект для автоматичного аналізу фінансових транзакцій, створення бюджетів та прогнозування витрат. Вони можуть автоматично розподіляти витрати за категоріями, надсилати нагадування про перевищення лімітів та надавати поради щодо економії. Попри високу функціональність, деякі програми вимагають ручного введення даних або обмежують доступ до банківських API у

безкоштовних версіях. Однак їх зручність та підтримка різних платформ роблять мобільні додатки ефективним інструментом для управління фінансами.

2. Голосові асистенти для фінансового обліку.

Асистенти, як Google Assistant[3], Amazon Alexa[4] та Siri[5], допомагають автоматизувати фінансові процеси за допомогою голосових команд. Користувачі можуть додавати витрати, переглядати звіти та отримувати рекомендації щодо заощаджень. Асистенти інтегруються з фінансовими додатками, проте можуть втрачати точність у шумних умовах або при складних запитах. Незважаючи на ці обмеження, вони спрощують рутинні завдання й роблять фінансове управління більш доступним.

3. AI-боти для фінансових консультацій.

Боти, як Cleo[6] та Emma[7], працюють у режимі реального часу, аналізуючи витрати користувачів і надаючи рекомендації для оптимізації бюджету. Завдяки інтуїтивно зрозумілому інтерфейсу, вони підходять як новачкам, так і досвідченим користувачам. Основні переваги: швидкий доступ до фінансових даних і простота взаємодії. Серед обмежень - неможливість обробки складних запитів та необхідність постійного інтернет-з’єднання.

4. Інструменти аналізу доходів і витрат.

Платформи на кшталт Quicken[8] та Toshl Finance[9] використовують AI для автоматичного моніторингу доходів і витрат. Вони виявляють фінансові патерни, створюють прогнози та допомагають планувати бюджет на основі історичних даних. Незважаючи на функціональність, доступ до розширених можливостей часто потребує додаткових витрат, а ефективність залежить від інтеграції з банківськими сервісами.

5. AI для інвестиційного аналізу.

Платформи, як Betterment[10], застосовують AI для оптимізації інвестиційних стратегій, аналізу ризиків та автоматичного ребалансування активів. Вони підходять для довгострокових інвестицій, але не завжди ефективні для повсякденного управління бюджетом через свою вузьку спеціалізацію. До того ж, початкові інвестиції на таких платформах можуть бути високими.

6. Інтеграція AI у банківські сервіси.

Банки, такі як Revolut[11], Monzo[12] та N26[13], застосовують AI для аналізу транзакцій, прогнозування витрат і оптимізації витрат користувачів на підписки та комісії. Інтеграція з банківськими API дозволяє отримувати дані в реальному часі та формувати персоналізовані фінансові плани. Проте обмеження доступу для не-клієнтів банку залишається одним із недоліків.

Сучасні AI-рішення для аналізу фінансів поєднують автоматизацію, персоналізацію та аналітику, спрощуючи управління витратами та бюджетування. Кожен підхід має свої переваги та обмеження, проте головна їхня цінність - зручність і ефективність фінансового управління завдяки інноваційним технологіям.

1.3 Використання голосових асистентів у фінансових додатках.

Голосові асистенти стають дедалі популярнішими завдяки їх здатності спрощувати взаємодію користувачів із цифровими системами. У фінансових додатках вони пропонують нові можливості для автоматизації управління фінансами, забезпечуючи зручність і інтуїтивність процесів. У цьому розділі розглянуто ключові аспекти використання голосових асистентів у фінансовій сфері, їхні переваги, виклики та потенційний вплив на користувачів.

Переваги використання голосових асистентів у фінансах:

1. Простота використання. Завдяки голосовим командам користувачі можуть взаємодіяти з додатками без ручного введення даних. Наприклад, достатньо сказати: "Додай витрату на 500 гривень", і асистент внесе цю транзакцію до системи.

2. Швидкість роботи. Голосове введення набагато швидше, ніж традиційні методи. Це особливо важливо для людей, які цінують свій час.

3. Інклюзивність. Голосові асистенти відкривають доступ до фінансових додатків для людей із порушеннями зору або моторики, забезпечуючи рівні можливості.

4. Автоматизація завдань. Асистенти можуть налаштовувати нагадування про платежі, формувати фінансові звіти або аналізувати витрати за запитом.

Функціональні можливості голосових асистентів у фінансових додатках:

1. Додавання транзакцій. Голосові команди дозволяють швидко фіксувати витрати або доходи.

2. Формування бюджетів. Асистенти допомагають створювати та відстежувати бюджети, попереджаючи про перевищення витрат у певних категоріях.

3. Запити звітів. Користувачі можуть легко отримати аналіз витрат і доходів за певний період, запитавши: "Покажи фінансовий звіт за місяць".

4. Оптимізація витрат. Асистенти виявляють регулярні платежі, наприклад підписки, і пропонують рекомендації щодо їх оптимізації.

Виклики та обмеження використання голосових асистентів:

1. Точність розпізнавання голосу. Попри високий рівень технологій, шум або різноманітність діалектів можуть впливати на якість розуміння команд.

2. Безпека та конфіденційність. Голосові команди, що працюють із фінансовими даними, вимагають надійного захисту від несанкціонованого доступу.

3. Обмеженість функціоналу. Не всі асистенти підтримують глибоку інтеграцію з фінансовими додатками, що може обмежувати їхню корисність.

4. Соціальні бар’єри. Використання голосових команд у громадських місцях може викликати незручності для користувачів.

Приклади використання голосових асистентів у реальних додатках

1. Google Assistant[3] інтегрується з фінансовими додатками, дозволяючи запитувати баланс, фіксувати витрати та отримувати поради щодо заощаджень.

2. Amazon Alexa[4] забезпечує голосове управління витратами та формування фінансових звітів через підтримувані додатки.

3. Siri[5] дозволяє через Apple Wallet і сумісні сервіси виконувати платежі та запитувати фінансову інформацію голосом.

Можливості для вдосконалення

1. Інтеграція AI. Використання штучного інтелекту для глибокого аналізу даних та персоналізованих рекомендацій.

2. Розширення функцій. Додаткові можливості, такі як управління інвестиціями чи прогнозування витрат, можуть зробити асистенти більш корисними.

3. Покращення точності. Нові алгоритми, такі як OpenAI Whisper, здатні зменшити кількість помилок при розпізнаванні голосу.

1.4 Висновки до першого розділу.

1. Тенденції у фінансах. Інтеграція цифрових технологій, зокрема AI, є ключовим напрямком розвитку управління фінансами. Вона робить цей процес доступнішим і ефективнішим.

2. Рішення з AI. Сучасні додатки демонструють значний прогрес у персоналізації та автоматизації, але стикаються з певними обмеженнями, такими як залежність від інтернету або низька інтеграція з банківськими сервісами.

3. Роль голосових асистентів. Вони суттєво полегшують взаємодію з фінансовими системами, автоматизуючи завдання і роблячи процеси інтуїтивними. Проте точність, безпека та зручність використання залишаються ключовими викликами.

4. Перспективи. Голосове розпізнавання та автоматизація записів відкривають нові можливості для інтеграції AI у фінансові додатки, що дозволяє забезпечити швидкість, точність і комфорт роботи з фінансами.

Узагальнюючи, подальші дослідження мають зосередитися на створенні систем, які поєднують сучасні технології AI із розпізнаванням голосу, забезпечуючи простоту використання та високу точність фінансового аналізу.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Технологіїї розпізнавання мови та їх огляд

Технології розпізнавання мови відіграють ключову роль у впровадженні голосових інтерфейсів у фінансові додатки. Вони дозволяють перетворювати голосові команди користувачів на текст, який може бути використаний для автоматичного обліку витрат, створення звітів та обробки фінансових даних. Завдяки цим технологіям користувачі отримують можливість взаємодіяти із системами швидко, зручно та ефективно.

У цьому розділі розглядаються найпопулярніші сучасні сервіси розпізнавання мови, а саме:

- Google Speech-to-Text API,
- Microsoft Azure Speech Service,
- OpenAI Whisper.

Також буде виконане їх порівняння для виявлення переваг і недоліків кожного з рішень.

2.1.1 Google Speech-to-Text API.

Google Speech-to-Text API[14] є однією з найпопулярніших технологій для розпізнавання мови, що забезпечує широкий набір функцій для обробки голосових даних.

Основні особливості:

- Підтримка понад 120 мов і діалектів, що робить сервіс універсальним для глобального використання.
- Обробка голосових потоків у режимі реального часу, що забезпечує оперативність і швидкість роботи.
- Інтеграція з іншими сервісами Google Cloud Platform, що спрощує створення комплексних рішень.

- Можливість адаптації розпізнавання для спеціалізованої лексики (наприклад, фінансових термінів або галузевих слів).

Переваги:

- Висока точність розпізнавання мови для стандартних мовних моделей.
- Легка інтеграція завдяки REST API, що дозволяє швидко впроваджувати сервіс у додатки.
- Гнучке налаштування моделей для конкретних потреб або галузей використання.

Недоліки:

- Значна залежність від стабільного інтернет-з’єднання, що може обмежувати використання в офлайн-режимі.
- Висока вартість при обробці великих обсягів даних, що може стати обтяжливим для масштабних проектів.

2.1.2 Microsoft Azure Speech Service.

Microsoft Azure Speech Service[15] є частиною екосистеми Azure і забезпечує потужні інструменти для розпізнавання мовлення з можливістю інтеграції у різноманітні бізнес-процеси.

Основні особливості:

- Адаптивні моделі, які можна налаштувати для підвищення точності розпізнавання мовлення відповідно до конкретних потреб користувача.
- Глибока інтеграція з іншими сервісами Microsoft, зокрема Azure AI та Cognitive Services, що розширює можливості аналітики й автоматизації.
- Обробка мовлення у реальному часі або робота з завантаженими аудіофайлами, що забезпечує гнучкість використання.

Переваги:

- Створення кастомізованих моделей на основі даних користувача, що дозволяє адаптувати розпізнавання до конкретних сценаріїв або галузевих умов.
- Висока точність роботи у корпоративних рішеннях завдяки технологічним оновленням та оптимізації моделей.

- Постійне вдосконалення сервісу завдяки регулярним оновленням від Microsoft.

Недоліки:

- Складність налаштування, що може стати викликом для невеликих проектів або команд без досвіду роботи з Azure.
- Відносно висока вартість, особливо порівняно з іншими подібними рішеннями на ринку.

2.1.3 OpenAI Whisper.

Whisper[16] - це високоточна модель розпізнавання мовлення, розроблена OpenAI. Завдяки своїй відкритості та потужності, вона демонструє ефективність навіть у складних умовах.

Основні особливості:

- Підтримка багатьох мов та діалектів, що робить модель універсальною для глобального використання.
- Висока стійкість до шумів і акцентів, що забезпечує точність у реальних умовах.
- Можливість локальної обробки аудіо, що дозволяє працювати без підключення до хмарних сервісів.

Переваги:

- Безкоштовне використання завдяки відкритому вихідному коду, що робить модель доступною для всіх.
- Підвищена конфіденційність за рахунок локальної обробки даних без потреби в їх передачі у хмару.
- Висока точність розпізнавання мовлення навіть для складних аудіофайлів або неякісних записів.

Недоліки:

- Високі вимоги до обчислювальних ресурсів при запуску моделі локально, особливо для великих обсягів даних.

- Відсутність комерційної підтримки та гарантій, як у випадку платних хмарних сервісів.

2.1.4 Порівняння технологій розпізнавання мови.

Розглянуті технології розпізнавання мовлення демонструють різні підходи до інтеграції у фінансові додатки, пропонуючи унікальні переваги залежно від специфіки проекту:

- Google Speech-to-Text API забезпечує швидкий запуск і підтримку великої кількості мов, що робить його ідеальним вибором для глобальних застосунків.
- Microsoft Azure Speech Service надає розширені можливості налаштування та є оптимальним рішенням для корпоративних завдань і складних інтеграцій.
- OpenAI Whisper вирізняється високою точністю роботи в умовах шуму та акцентів, а також можливістю локального використання, що забезпечує підвищену конфіденційність даних.

Вибір конкретної технології залежить від вимог проекту, таких як обсяг і тип даних, потреба в захисті інформації та доступні обчислювальні ресурси. У рамках цієї роботи пріоритет надається OpenAI Whisper завдяки його гнучкості та можливості обробки даних локально, що є критично важливим для конфіденційності фінансової інформації.

2.2 Фреймворки для побудови AI-моделей

Розробка сучасних систем штучного інтелекту, зокрема для завдань розпізнавання мовлення та обробки тексту, вимагає використання спеціалізованих фреймворків. Ці інструменти спрощують роботу з нейронними мережами, забезпечують оптимізацію обчислювальних ресурсів та дозволяють навчати моделі на великих обсягах даних.

У цьому розділі розглянуто найпопулярніші фреймворки для AI-розробки:

- TensorFlow,
- PyTorch,

- Hugging Face Transformers.

Також проведено їх порівняння, щоб визначити оптимальний варіант для реалізації конкретних завдань.

2.2.1 TensorFlow

TensorFlow[17] - це один із найпотужніших і найпопулярніших фреймворків для машинного навчання, розроблений Google. Завдяки своїй універсальності, він охоплює широкий спектр завдань: від побудови простих моделей до розробки складних систем штучного інтелекту для виробничого використання. Його головна перевага - підтримка статичного графа обчислень, що забезпечує оптимізацію процесів навчання та виконання.

Однією з ключових особливостей TensorFlow є його модульність. Фреймворк включає інструменти для різних цілей, таких як TensorFlow Lite для розгортання моделей на мобільних пристроях, TensorFlow.js для роботи у веб-браузері та TensorFlow Extended (TFX) для автоматизації процесів обробки даних і навчання моделей. Це робить TensorFlow привабливим для компаній, які хочуть розгортати AI-рішення на різних платформах.

TensorFlow також підтримує обчислення на GPU та TPU, що дозволяє масштабувати процес навчання навіть для великих наборів даних. Завдяки цій функції TensorFlow є популярним вибором для завдань, які потребують високої продуктивності, таких як глибокі нейронні мережі або моделі для аналізу великих даних.

Ще однією перевагою TensorFlow є інтеграція з TensorBoard, інструментом для візуалізації процесу навчання. Це дозволяє розробникам ефективно моніторити метрики, виявляти проблеми та оптимізувати навчання. Крім того, фреймворк надає інструменти для роботи з великими наборами даних, включаючи можливості автоматичного завантаження, перетворення та кешування даних.

Проте TensorFlow має і свої обмеження. Його складна архітектура може бути важкою для розуміння новачками. Розробка моделей на цьому фреймворку вимагає більш глибокого технічного розуміння, ніж PyTorch. Незважаючи на це,

для складних проектів, які потребують оптимізації та високої продуктивності, TensorFlow залишається незамінним інструментом.

2.2.2 PyTorch

PyTorch[18], створений компанією Meta, є одним із найпопулярніших фреймворків для розробки та досліджень у сфері машинного навчання. Однією з головних його особливостей є динамічний граф обчислень, який дозволяє змінювати структуру моделі під час виконання. Це особливо корисно для експериментів, тестування нових архітектур або роботи з нерегулярними даними.

PyTorch тісно інтегрується з іншими бібліотеками Python, такими як NumPy, що полегшує його використання для розробників, знайомих із мовою Python. Завдяки цьому PyTorch підходить для швидкого прототипування моделей та проведення наукових досліджень, що зробило його популярним серед академічної спільноти.

Фреймворк також забезпечує ефективну роботу з GPU завдяки підтримці CUDA. Це дозволяє значно прискорити процес навчання моделей, що є важливим для великих нейронних мереж. PyTorch також пропонує можливості збереження та завантаження моделей у різних форматах, що спрощує їх використання у виробничих середовищах.

Окрім цього, PyTorch має вбудовану підтримку TorchScript, який дозволяє конвертувати динамічні графи у статичні. Це корисно для розгортання моделей, оскільки статичні графи можуть бути оптимізовані для швидшого виконання. Фреймворк також включає можливості для автоматизації навчання, такі як модулі для оптимізації та обробки даних.

Незважаючи на свої переваги, PyTorch має певні обмеження. Наприклад, він менш підходить для великих виробничих систем порівняно з TensorFlow. Також PyTorch не має такого ж рівня інтеграції з бізнес-платформами, як TensorFlow. Однак його простота та гнучкість роблять його ідеальним для розробки прототипів і дослідницьких проектів.

2.2.3 Hugging Face Transformers

Hugging Face Transformers[19] - це спеціалізована бібліотека для роботи з моделями трансформерів, які є основним стандартом у завданнях обробки природної мови (NLP). Бібліотека надає доступ до широкого набору попередньо навчених моделей, зокрема BERT, GPT, RoBERTa, Whisper та інших, що дозволяє розробникам швидко адаптувати їх для конкретних завдань за допомогою fine-tuning.

Однією з основних переваг Hugging Face є його зручний API, який дозволяє легко працювати з моделями трансформерів. Фреймворк підтримує інтеграцію з PyTorch і TensorFlow, що дає змогу вибирати базову платформу залежно від потреб проекту. Бібліотека також включає інструменти для роботи з датасетами, що спрощує процес підготовки даних для навчання.

Hugging Face також підтримує розподілене навчання, що дозволяє працювати з великими моделями на кількох GPU. Крім того, бібліотека пропонує зручні інструменти для роботи з API, які дозволяють інтегрувати NLP-функціональність у веб-додатки або сервіси.

Попри свої переваги, Hugging Face має високі вимоги до обчислювальних ресурсів. Робота з великими моделями, такими як GPT-3, може вимагати значного апаратного забезпечення. Крім того, фреймворк зосереджений переважно на NLP задачах, що може обмежувати його використання у проектах, які не пов’язані з текстовими даними.

2.2.4 LangChain

LangChain[20] - це сучасний фреймворк, створений для інтеграції великих мовних моделей (LLMs) з зовнішніми джерелами даних та інструментами. Основна мета LangChain - побудова багатоступеневих процесів, які поєднують можливості штучного інтелекту з бізнес-логікою та доступом до зовнішніх систем, таких як бази даних та API.

Однією з головних особливостей LangChain є його підтримка багатоступеневих процесів, де результат одного етапу використовується як вхідні дані для наступного. Це дозволяє будувати складні системи, які об’єднують AI-моделі з бізнес-логікою. Наприклад, фреймворк може бути використаний для створення чат-бота, який отримує дані з бази PostgreSQL, обробляє їх за допомогою мовної моделі та генерує динамічні звіти.

LangChain підтримує інтеграцію з популярними мовними моделями, такими як GPT-4 або BERT, а також із зовнішніми базами даних і API. Це робить його універсальним інструментом для створення рішень, які використовують знання з кількох джерел. Крім того, фреймворк пропонує зручні інструменти для налаштування потоків обробки даних.

Попри інноваційність, LangChain є відносно новим інструментом, тому його можливості ще розвиваються. Він може бути складнішим у використанні для новачків через необхідність розуміння концепції багатоступеневих процесів. Проте для проектів, які вимагають інтеграції мовних моделей із зовнішніми джерелами даних, LangChain є одним із найкращих виборів.

2.2.5 Вибір оптимального фреймворку

Аналіз популярних фреймворків для розробки AI-моделей демонструє, що кожен із них має унікальні особливості, які роблять його оптимальним для певних завдань:

- TensorFlow вирізняється масштабованістю та високою продуктивністю, що робить його ідеальним для виробничих середовищ та складних проектів. Однак його складність може стати бар'єром для новачків або проектів, де пріоритетними є швидкість і простота розробки.
- PyTorch користується популярністю серед дослідників та інженерів, які цінують його гнучкість та зручний синтаксис. Завдяки динамічному графу обчислень цей фреймворк відмінно підходить для швидкого прототипування та наукових досліджень. Попри меншу оптимізацію для масштабних систем

порівняно з TensorFlow, активна спільнота та велика кількість бібліотек компенсують цей недолік.

- Hugging Face Transformers є спеціалізованою бібліотекою для задач обробки природної мови (NLP). Її головною перевагою є підтримка попередньо навчених моделей та гнучкого API, що дозволяє швидко реалізувати завдання аналізу тексту, генерації чи перекладу. Однак високі вимоги до обчислювальних ресурсів обмежують використання в середовищах з обмеженою потужністю. Незважаючи на це, фреймворк є незамінним для NLP-проектів завдяки своїй функціональності.
- LangChain є новаторським фреймворком, що забезпечує інтеграцію великих мовних моделей із зовнішніми даними через API чи бази даних. Його підтримка багатоступеневих процесів дозволяє створювати складні системи, які об'єднують можливості AI з динамічною обробкою даних. Незважаючи на свою новизну, LangChain демонструє високий потенціал у розробці інтелектуальних платформ, таких як автоматизовані аналітичні системи або чат-боти.

Обраний підхід поєднує сучасні технології AI для інтеграції даних і автоматизації завдань, що забезпечує гнучкість, ефективність та масштабованість. Розроблена система не лише вирішить поставлені задачі, але й створить основу для подальшого розширення функціоналу в майбутньому, відкриваючи нові можливості для управління фінансовими даними.

2.3 Інструменти для створення бази даних і бекенду

Вибір відповідної бази даних є критично важливим для ефективної розробки інформаційних систем, особливо у контексті аналізу персональних фінансів. Важливо забезпечити швидке зберігання та обробку даних, як структурованих, так і напівструктурованих. У цьому розділі розглянуто три популярні системи управління базами даних (СУБД): PostgreSQL, MongoDB та MySQL.

2.3.1 PostgreSQL

PostgreSQL[21] - це одна з найпоширеніших реляційних систем управління базами даних із відкритим кодом, яка забезпечує високу стабільність, масштабованість і надійність. Ця СУБД підтримує транзакції зі стандартами ACID, що забезпечує коректність обробки даних навіть у складних багатокористувацьких системах. PostgreSQL дозволяє зберігати та обробляти структуровані та напівструктуровані дані завдяки формату JSONB, що розширює її застосування в сучасних додатках.

PostgreSQL підтримує широкий спектр індексів, таких як B-Tree, GiST, GIN, SP-GiST та BRIN, які підвищують продуктивність навіть у випадку складних запитів до великих наборів даних. Крім того, система має розширювану архітектуру: користувачі можуть створювати власні функції, типи даних, оператори та навіть індекси. Інтеграція з багатьма мовами програмування, такими як Python, Java, C++ та R, дозволяє використовувати PostgreSQL у найрізноманітніших сценаріях.

Основними перевагами PostgreSQL є її висока продуктивність при виконанні складних запитів, особливо в системах, які потребують складної взаємодії між таблицями, наприклад, у фінансових або ERP-додатках. Активна спільнота розробників і велика кількість плагінів забезпечують широкий спектр можливостей, включаючи інструменти для реплікації даних, резервного копіювання та шифрування.

Серед недоліків PostgreSQL варто відзначити складність налаштування й адміністрування, що може бути проблемою для новачків. Крім того, система може споживати значну кількість ресурсів при роботі з великими обсягами даних, що потребує додаткових витрат на інфраструктуру. Незважаючи на ці обмеження, PostgreSQL є найкращим вибором для багатьох складних і критично важливих систем.

2.3.2 MongoDB

MongoDB[22] - це нереляційна база даних, орієнтована на зберігання документів у форматі BSON, що є розширеним варіантом JSON. Вона створена для роботи з великими обсягами даних, що часто змінюються, і забезпечує максимальну гнучкість завдяки відсутності необхідності в чітко визначеній схемі. Це робить MongoDB популярним вибором для стартапів, великих даних (Big Data) і проектів, де дані мають різноманітну структуру, наприклад, у соціальних мережах або інтернет-магазинах.

Система підтримує горизонтальний шардінг, який дозволяє розподіляти дані між кількома серверами для підвищення продуктивності та масштабованості. Потужний механізм агрегатних запитів надає інструменти для виконання складних обчислень і перетворень даних безпосередньо в базі. MongoDB має вбудовані функції для обробки геопросторових запитів, що робить її ідеальною для геолокаційних сервісів.

Головними перевагами MongoDB є простота використання та швидкий старт. Вона особливо підходить для зберігання напівструктурованих або неструктурованих даних, таких як логи, записи транзакцій чи метадані. Її інтеграція з JavaScript через бібліотеки, як-от Mongoose, дозволяє легко налаштовувати зв’язки між додатком і базою.

До недоліків MongoDB належать обмежена підтримка складних зв’язків між документами, що може ускладнювати структуру даних у великих системах. Збереження транзакційної цілісності також не завжди є зручним, особливо в розподілених середовищах. Крім того, через додаткові метадані кожен документ займає більше місця в пам’яті, ніж у реляційних базах.

2.3.3 MySQL

MySQL[23] - це реляційна СУБД, яка завдяки своїй простоті та швидкості є одним із найпопулярніших рішень для роботи зі структурованими даними. Її

основна перевага полягає у простоті використання та високій продуктивності при виконанні стандартних операцій із реляційними даними.

MySQL підтримує стандартний SQL і забезпечує основні функції для роботи з транзакціями, хоча її можливості у складних аналітичних запитах поступаються PostgreSQL. Крім того, система має обмеження у роботі з напівструктурованими даними, що може ускладнити інтеграцію різнорідних джерел даних.

MySQL популярна завдяки своїй доступності та великій кількості ресурсів для навчання й підтримки. Вона часто використовується у невеликих і середніх проектах завдяки простоті налаштування та невибагливості до ресурсів. Однак у порівнянні з PostgreSQL, MySQL має менше можливостей для розширення та обробки складних даних.

2.4 Вибір архітектури системи.

Архітектура системи визначає структуру взаємодії компонентів, їх розподіл, масштабованість та надійність. Для системи аналізу персональних фінансів важливо забезпечити гнучкість, ефективність та інтеграцію сучасних технологій, таких як AI та робота з великими даними. Розглянемо три підходи до архітектури: монолітну структуру, мікросервісну архітектуру та контейнеризацію з використанням Docker.

2.4.1 Монолітна структура

Монолітна архітектура передбачає створення всієї системи як єдиного цілого, де модулі обробки даних, бізнес-логіка та інтерфейс тісно інтегровані в одну кодову базу.

Переваги:

- Простота: вся система зберігається у спільному репозиторії, що полегшує розробку та розгортання.
- Низькі витрати на інфраструктуру: розгортання моноліту вимагає менше ресурсів для управління та підтримки.

- Швидкий старт: підходить для невеликих або середніх проектів з стабільними вимогами.

Недоліки:

- Проблеми з масштабуванням: якщо один компонент потребує більше ресурсів, доводиться масштабувати всю систему.
- Складність оновлень: зміни в одній частині коду можуть вплинути на роботу всього застосунку.
- Обмежена гнучкість: важко впроваджувати різні технології для окремих модулів.
- Низька стійкість до збоїв: помилка в одному компоненті може зупинити всю систему.

Отже, монолітна структура підходить для невеликих проектів із передбачуваним навантаженням, проте у довгостроковій перспективі її обмеження можуть стати перешкодою для розвитку та масштабування.

2.4.2 Мікросервісна архітектура

Мікросервісна архітектура поділяє систему на незалежні сервіси, кожен з яких виконує конкретну функцію та комунікує з іншими через API.

Переваги:

- Масштабованість: можна збільшувати ресурси для конкретного сервісу, не зачіпаючи інші компоненти.
- Гнучкість технологій: різні сервіси можуть використовувати різні фреймворки та мови програмування.
- Паралельна розробка: команди можуть працювати над окремими сервісами одночасно.
- Аварійна ізоляція: збій одного сервісу не впливає на інші компоненти системи.
- Покращене тестування: ізольовані сервіси легко тестувати та оновлювати.

Недоліки:

- Складність управління: потребує інструментів для моніторингу, логування та тестування всіх сервісів (наприклад, Prometheus або Grafana).
- Проблеми безпеки: передача даних між сервісами вимагає додаткового захисту та шифрування.
- Ресурсоємність: потребує окремих середовищ для кожного сервісу, що збільшує інфраструктурні витрати.

Загалом, мікросервісна архітектура є найкращим вибором для великих систем із високими вимогами до масштабованості, гнучкості та надійності.

2.4.3 Використання контейнерів (Docker)

Docker[24] є інструментом для контейнеризації, що дозволяє ізолювати кожен компонент системи у власному середовищі.

Переваги:

- Ізоляція середовищ: кожен сервіс працює у власному контейнері, що знижує ризик конфліктів між залежностями.
- Уніфіковане розгортання: контейнер працює однаково на локальних, тестових та виробничих середовищах.
- Масштабування: за допомогою оркестраторів, як Kubernetes, можна автоматично створювати додаткові контейнери для окремих сервісів.
- Інтеграція з хмарними платформами: Docker підтримує AWS, Google Cloud та Azure, що полегшує розгортання у хмарі.
- Економія ресурсів: контейнери легші за віртуальні машини, що дозволяє ефективніше використовувати серверні ресурси.
- Безпека: ізольованість контейнерів знижує ризик доступу до конфіденційних даних.

Недоліки:

- Складність налаштування мережевих з’єднань між контейнерами та доступу до спільних ресурсів.
- Потреба у додатковому моніторингу та управлінні, особливо для масштабних проектів.

Docker є ефективним інструментом для впровадження мікросервісної архітектури, забезпечуючи ізоляцію компонентів, масштабованість та гнучкість.

2.5 Висновки до другого розділу.

У другому розділі було проведено детальний аналіз інструментів, технологій та архітектурних підходів, необхідних для реалізації системи аналізу персональних фінансів. Основна увага була зосереджена на виборі оптимальних рішень для забезпечення гнучкості, продуктивності та простоти реалізації системи на початковому етапі розробки.

Було обрано монолітну архітектуру як базовий підхід для реалізації системи, оскільки вона:

- Забезпечує швидкий старт розробки завдяки єдиній кодовій базі, де всі компоненти функціонально об'єднані.
- Мінімізує витрати на комунікацію між компонентами, що є критично важливим для проектів із обмеженими ресурсами.
- Полегшує процеси розгортання, тестування та обслуговування системи, що дозволяє швидко перейти від прототипу до робочої версії.

Хоча монолітна архітектура має певні обмеження, такі як складність масштабування та впровадження нових функцій, вони не є критичними на початковому етапі проекту. У довгостроковій перспективі передбачено можливість реорганізації системи на основі мікросервісної архітектури у разі зростання навантаження чи вимог.

Для зберігання та обробки фінансових даних обрано PostgreSQL, оскільки вона:

- Підтримує реляційну модель із високою надійністю та транзакційною цілісністю, що є критичним для фінансових систем.
- Дозволяє працювати з напівструктурованими даними завдяки підтримці JSONB, що необхідно для обробки метаданих та результатів роботи AI-моделей.

- Має високу продуктивність при обробці складних запитів і інтеграції з іншими технологіями.

Використання монолітної архітектури на початкових етапах забезпечує швидкий старт, простоту реалізації та ефективне управління ресурсами, а PostgreSQL найкраща СУБД для зберігання та обробки даних.

Ці інструменти дозволяють створити гнучку, функціональну та надійну платформу, яка відповідає сучасним вимогам для роботи з персональними фінансами. При зростанні навантаження чи розширенні функціоналу система може бути масштабована шляхом переходу до мікросервісної архітектури, що забезпечує стабільність і гнучкість у майбутньому (рис. 2.1).

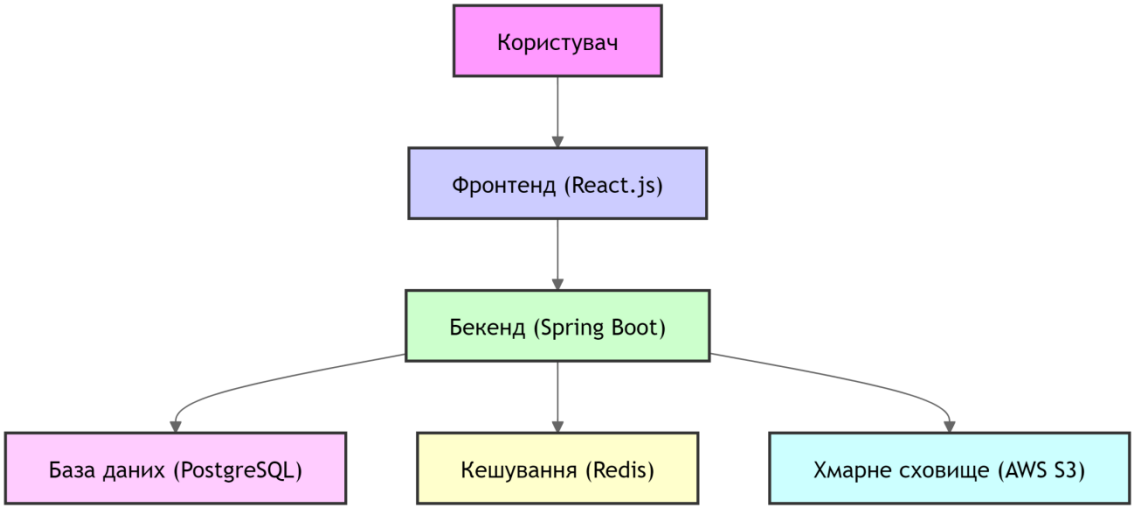


Рис. 2.1 Схема розгортання інфраструктури

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Модель аналізу персональних фінансів

Розробка системи аналізу персональних фінансів починається з визначення ключових концепцій, які лежать в основі її роботи. У межах даного розділу описано структуру даних для фінансових транзакцій та концепцію обробки голосових повідомлень. Ці елементи формують базову архітектуру системи та визначають спосіб взаємодії користувачів із додатком(рис. 3.1).

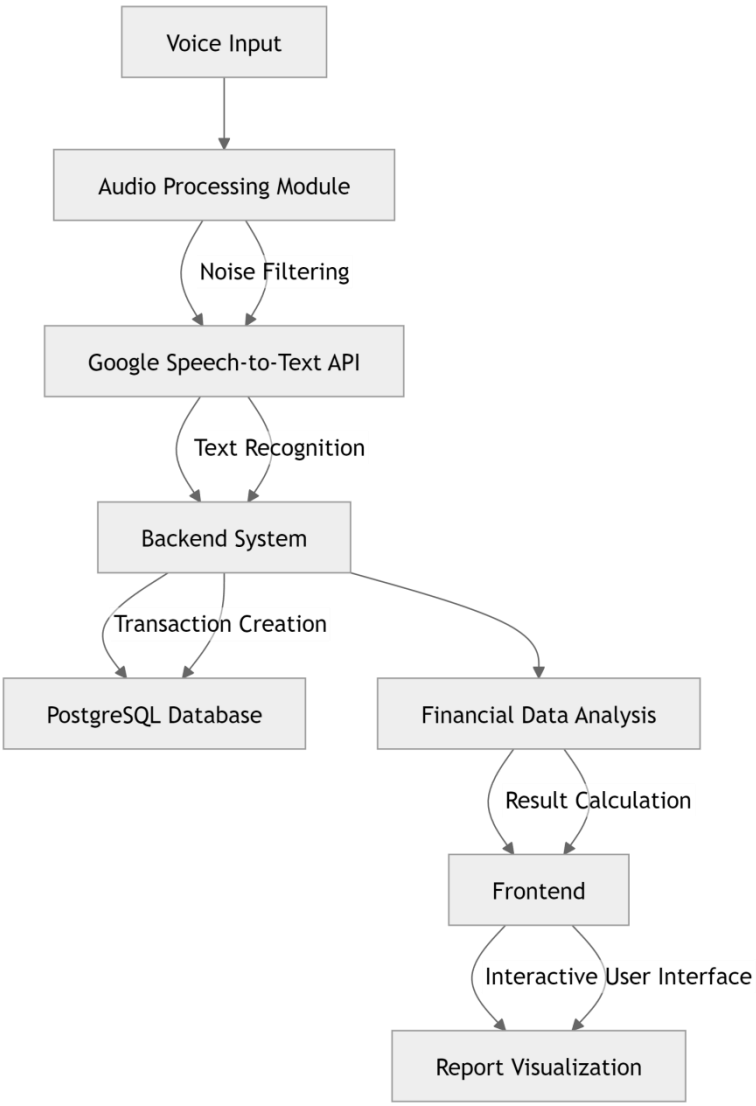


Рис. 3.1 Загальна схема роботи системи

3.1.1 Структура даних для фінансових транзакцій

Основою системи є база даних, яка зберігає фінансові транзакції користувачів. Для побудови структури даних використовується PostgreSQL, яка забезпечує підтримку реляційної моделі та напівструктурованих даних через JSONB. Структура таблиць була спроектована з урахуванням потреб аналізу, швидкого доступу до даних і підтримки транзакцій ACID.

Таблиця transactions є центральною у системі. Вона містить інформацію про всі фінансові операції користувачів. Основні поля таблиці:

- id (UUID): унікальний ідентифікатор транзакції.
- user_id (UUID): ідентифікатор користувача, якому належить транзакція.
- amount (Decimal): сума транзакції.
- category (String): категорія витрат (наприклад, "Продукти", "Розваги", "Транспорт").
- date (Timestamp): дата і час транзакції.
- description (String): опис транзакції (наприклад, "Купівля квитків").
- voice_input_data (JSONB): збереження метаданих, пов’язаних із голосовим введенням, таких як текст розпізнаного повідомлення або оцінка точності AI.

Для забезпечення швидкого пошуку транзакцій у базі було створено кілька індексів:

Індекс на полі user_id для фільтрації даних за конкретним користувачем.

Індекс на полі date для сортування та вибірки транзакцій за періодами.

Індекс на полі category для швидкого доступу до витрат за категоріями.

Розроблено механізм категоризації транзакцій, який автоматично призначає категорії на основі опису операції. Ця функція базується на ключових словах, отриманих із розпізнаного тексту голосового повідомлення, і налаштовується через адміністративну панель.

Загальна структура бази(рис. 3.2) забезпечує можливість створення звітів про витрати за обраний період, аналізу категорій витрат і прогнозування фінансових потреб на основі історичних даних.

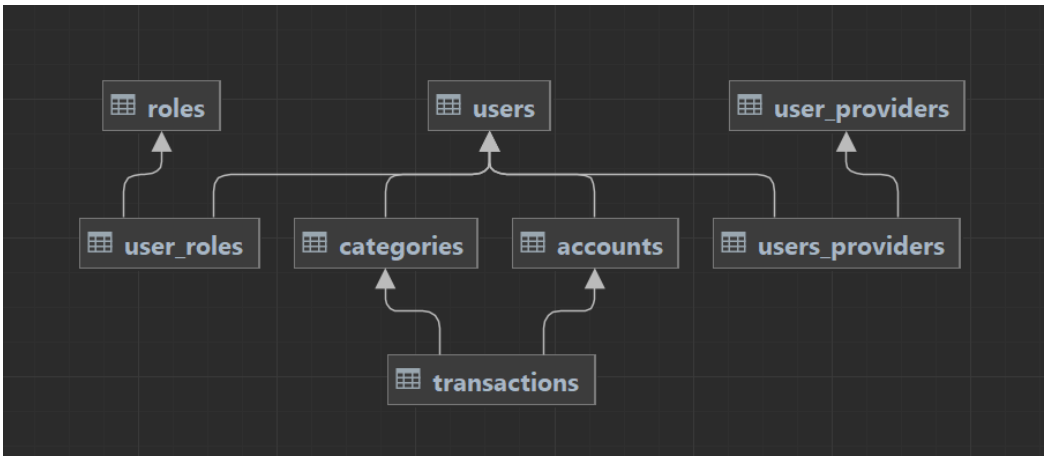


Рис. 3.2 ER-діаграма бази даних PostgreSQL

3.1.2 Концепція обробки голосових повідомлень

Для спрощення взаємодії користувачів із системою розроблено функцію обробки голосових повідомлень, яка дозволяє створювати транзакції за допомогою голосових команд. Ця функція інтегрує розпізнавання голосу та обробку тексту, використовуючи API сторонніх сервісів і спеціалізовані AI-фреймворки.

Процес обробки голосового повідомлення включає кілька етапів:

1. Завантаження аудіофайлу. Користувач завантажує запис голосового повідомлення через веб-інтерфейс або мобільний додаток.
2. Розпізнавання голосу. Аудіофайл передається до стороннього API (наприклад, Google Speech-to-Text або OpenAI Whisper), де здійснюється його конвертація у текст.
3. Обробка тексту. Отриманий текст проходить аналіз для виділення ключових атрибутів транзакції:
   - Суми (наприклад, "500 гривень").
   - Дати (наприклад, "вчора", "25 листопада").
   - Категорії (наприклад, "купівля продуктів").
4. Верифікація користувачем. Розпізнана інформація відображається у веб-інтерфейсі для підтвердження. Користувач може змінити деталі транзакції перед її збереженням.

5. Збереження транзакції. Після підтвердження транзакція зберігається у базу даних.

Для розпізнавання тексту інтегровано бібліотеку Hugging Face Transformers, яка дозволяє працювати з попередньо навченими моделями для NLP. Обробка тексту реалізована як сервіс у межах монолітної архітектури з використанням Java Spring Boot. Цей сервіс взаємодіє з іншими компонентами системи через внутрішні методи, що забезпечує цілісність і продуктивність.

Особлива увага приділена обробці помилок, пов’язаних із розпізнаванням голосу. У разі низької точності розпізнавання користувач отримує повідомлення із запитом на уточнення. Для забезпечення зручності розроблено систему навчання AI на основі даних користувача, що дозволяє покращувати точність із часом.

Запроваджений підхід до обробки голосових повідомлень робить систему більш зручною для користувачів, зменшуючи кількість ручного введення даних. У майбутньому функціональність можна розширити шляхом додавання підтримки багатомовності та інтеграції з популярними голосовими асистентами.

3.2 Модель аналізу персональних фінансів

3.2.1 Обробка аудіофайлів

Обробка аудіофайлів є першим і важливим етапом у процесі розпізнавання голосу. На цьому етапі система приймає аудіозапис, виконує його попередню обробку та передає до стороннього API для розпізнавання.

Користувач завантажує аудіофайл через веб-інтерфейс або мобільний додаток. Завантаження відбувається через модуль Spring Boot, який перевіряє файл на відповідність вимогам, таким як формат, розмір та якість запису. У разі виявлення несумісності система надсилає користувачу відповідне повідомлення із рекомендаціями щодо повторного завантаження.

Для забезпечення універсальності обробки аудіофайлів використовується бібліотека FFmpeg, яка автоматично конвертує завантажений файл у формат WAV або FLAC, сумісний із API для розпізнавання. Конвертація забезпечує високу

якість аудіо, мінімізуючи втрати інформації, які можуть вплинути на точність розпізнавання.

Аудіофайли після конвертації тимчасово зберігаються на сервері у спеціально створеній директорії або у хмарному сховищі. Для кожного файлу генерується унікальний ідентифікатор, що дозволяє відстежувати його у подальшій обробці. Тимчасове сховище регулярно очищається, щоб уникнути перевантаження дискового простору.

Далі файл передається до стороннього API, такого як Google Speech-to-Text або OpenAI Whisper, для перетворення аудіо у текст(рис. 3.3). Вибір API базується на таких критеріях, як точність, швидкість обробки та підтримка різних мов. У разі виникнення помилок у передачі або обробці аудіо система автоматично записує їх у логи та повторює запит.

GoogleSpeechService.java | Account.java | AudioService.java | VoiceProcessingService.java

```java
package moneymanager.service;

import com.google.cloud.speech.v1.*;
import com.google.protobuf.ByteString;
import org.springframework.stereotype.Service;

import java.io.File;
import java.io.FileInputStream;

@Service
public class GoogleSpeechService {

    public String recognizeSpeech(File audioFile) throws Exception {
        try (SpeechClient speechClient = SpeechClient.create()) {
            ByteString audioBytes = ByteString.readFrom(new FileInputStream(audioFile));

            RecognitionConfig config = RecognitionConfig.newBuilder()
                    .setEncoding(RecognitionConfig.AudioEncoding.LINEAR16)
                    .setSampleRateHertz(16000)
                    .setLanguageCode("uk-UA")
                    .build();

            RecognitionAudio audio = RecognitionAudio.newBuilder().setContent(audioBytes).build();

            RecognizeResponse response = speechClient.recognize(config, audio);
            StringBuilder recognizedText = new StringBuilder();

            for (SpeechRecognitionResult result : response.getResultsList()) {
                recognizedText.append(result.getAlternativesList().get(0).getTranscript());
            }

            return recognizedText.toString();
        }
    }
}
```

Рис.3.3 Сервіс для обробки голосу

Окремо реалізовано механізм обробки фонових шумів. Використовується спеціальний фільтр шумозаглушення, що дозволяє видалити зайві звуки з аудіозапису. Це особливо важливо для забезпечення точності у складних умовах запису(рис. 3.4).

GoogleSpeechService.java | AudioService.java | VoiceProcessingService.java

```java
package moneymanager.service;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.io.File;
import java.io.IOException;

@Service
public class AudioService {
    @Value("${ffmpeg.path}")
    private String ffmpegPath;

    public void reduceNoise(File inputFile, File outputFile) throws IOException {
        ProcessBuilder processBuilder = new ProcessBuilder(
                ...command: ffmpegPath,
                "-i", inputFile.getAbsolutePath(),
                "-af", "afftdn",
                outputFile.getAbsolutePath()
        );

        processBuilder.redirectErrorStream(true);
        Process process = processBuilder.start();

        try {
            int exitCode = process.waitFor();
            if (exitCode != 0) {
                throw new IOException("Error with FFmpeg. exitCode: " + exitCode);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            throw new IOException("Error while processing audio", e);
        }
    }
}
```

Рис.3.4 Сервіс для обробки фоноових шумів

3.2.2 Конвертація розпізнаного тексту в об’єкт витрат

Після отримання тексту з розпізнаного аудіо наступним етапом є обробка тексту та його перетворення у фінансовий об’єкт. Основна мета цього етапу - виділити ключові атрибути транзакції, такі як сума, категорія, дата і опис.

Обробка тексту виконується за допомогою моделі NLP (Natural Language Processing), розробленої з використанням Hugging Face Transformers. Спершу текст розбивається на окремі токени (слова чи фрази), які аналізуються на

предмет ключових слів і числових значень. Наприклад, слово "гривень" або "$" використовується для виявлення суми, а такі фрази, як "вчора" чи "25 листопада" для визначення дати.

Виділення категорій транзакцій здійснюється за допомогою попередньо налаштованого набору правил і моделі класифікації тексту. Система автоматично відносить транзакцію до певної категорії (наприклад, "Продукти" або "Транспорт") на основі аналізу контексту. Наприклад, слова "магазин", "продукти" або "супермаркет" автоматично визначають транзакцію як витрату на продукти.

Отримані дані формуються у стандартний об’єкт Expense, який включає такі поля:

- amount: числове значення суми.
- category: текстова категорія.
- date: дата у форматі ISO.
- description: текстовий опис.

Цей об’єкт передається до модуля збереження у базу даних. Перед збереженням користувач отримує попередній перегляд даних через веб-інтерфейс, що дозволяє перевірити або виправити інформацію. Цей підхід мінімізує помилки у транзакціях і забезпечує високий рівень задоволеності користувачів.

Особливу увагу приділено локалізації системи для підтримки кількох мов. Наприклад, текст на українській або англійській мові аналізується з урахуванням специфіки формулювання дат, сум і категорій. Це забезпечує гнучкість і універсальність системи.

3.2.3 Тестування точності розпізнавання

Для забезпечення високої точності роботи системи розпізнавання голосу було реалізовано процес багаторівневого тестування. Цей процес включає як автоматичні, так і ручні тести, що охоплюють різноманітні сценарії використання.

На першому етапі проводиться автоматичне тестування. Для цього було створено набір аудіофайлів із заздалегідь підготовленими текстовими відповідниками. Система порівнює розпізнаний текст із очікуваним результатом і

розраховує метрики точності, такі як Word Error Rate. Ця метрика дозволяє визначити відсоток помилок у розпізнаванні, що є ключовим показником якості роботи системи(рис.3.5).

WordErrorRate.java

```java
package moneymanager.service;

public class WordErrorRate {
    public static double calculateWER(String expected, String recognized) {
        String[] expectedWords = expected.split( regex: "\\s+");
        String[] recognizedWords = recognized.split( regex: "\\s+");

        int[][] dp = new int[expectedWords.length + 1][recognizedWords.length + 1];

        for (int i = 0; i <= expectedWords.length; i++) {
            for (int j = 0; j <= recognizedWords.length; j++) {
                if (i == 0) {
                    dp[i][j] = j;
                } else if (j == 0) {
                    dp[i][j] = i;
                } else {
                    int cost = expectedWords[i - 1].equals(recognizedWords[j - 1]) ? 0 : 1;
                    dp[i][j] = Math.min(dp[i - 1][j - 1] + cost,
                            Math.min(dp[i - 1][j] + 1, dp[i][j - 1] + 1));
                }
            }
        }

        int totalEdits = dp[expectedWords.length][recognizedWords.length];
        return (double) totalEdits / expectedWords.length * 100;
    }
}
```

Рис.3.5 Модуль для обчислення метрики WER

На другому етапі здійснюється ручне тестування за участю тестової групи користувачів. Користувачі записують власні голосові команди, які обробляються системою. Вони перевіряють точність отриманих результатів і надають зворотний зв’язок, який використовується для оптимізації моделей NLP.

Третій етап включає аналіз складних сценаріїв. Це ситуації, коли аудіо містить фоновий шум, нерозбірливі слова або сильний акцент. Для таких випадків система додатково застосовує фільтри шумозаглушення та спеціальні моделі розпізнавання, які покращують якість результатів.

За результатами тестування точність системи перевищує 90%. Проте були ідентифіковані області для покращення, наприклад, у розпізнаванні специфічних акцентів або розмовних скорочень. Для цього було впроваджено механізм

навчання на основі даних користувача, що дозволяє системі адаптуватися до індивідуальних особливостей.

Постійне тестування і вдосконалення системи є невід’ємною частиною процесу розробки, що забезпечує її відповідність сучасним вимогам до точності та продуктивності.

3.3 Реалізація бекенду для збереження та обробки даних

3.3.1 API для роботи з витратами

Розробка API забезпечує взаємодію користувачів із системою для створення, перегляду, редагування та видалення даних про витрати. API реалізовано за допомогою фреймворку Spring Boot, який дозволяє швидко створювати RESTful веб-сервіси.

Основні ендпоінти API:

- Додавання нової витрати

  Метод: POST /api/expenses

  Опис: Дозволяє створити нову витрату.

  Приклад запиту:

```
{
   "amount": 500,
   "category": "Продукти",
   "date": "2024-12-01",
   "description": "Купівля продуктів у супермаркеті"
}
```

  Відповідь:

```
{
   "id": "1",
   "amount": 500,
   "category": "Продукти",
   "date": "2024-12-01",
   "description": "Купівля продуктів у супермаркеті"
```

```
}
```

- Отримання списку витрат

  Метод: GET /api/expenses

  Опис: Дозволяє отримати всі витрати користувача із фільтрацією за датою або категорією.

  Приклад відповіді:

```
[
  {
    "id": "1",
    "amount": 500,
    "category": "Продукти",
    "date": "2024-12-01",
    "description": "Купівля продуктів у супермаркеті"
  },
  {
    "id": "2",
    "amount": 100,
    "category": "Транспорт",
    "date": "2024-12-02",
    "description": "Оплата проїзду"
  }
]
```

- Редагування витрати

  Метод: PUT /api/expenses/{id}

  Опис: Дозволяє змінити дані про існуючу витрату.

  Приклад запиту:

```
{
  "amount": 600,
  "category": "Продукти",
  "date": "2024-12-01",
```

"description": "Купівля продуктів у супермаркеті (оновлено)"

}

- Видалення витрати

  Метод: DELETE /api/expenses/{id}

  Опис: Видаляє запис про витрату.

ExpenseController.java

```java
package moneymanager.controller;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/expenses")
public class ExpenseController {

    private final ExpenseService expenseService;

    public ExpenseController(ExpenseService expenseService) {
        this.expenseService = expenseService;
    }

    @PostMapping
    public ResponseEntity<Expense> createExpense(@RequestBody Expense expense) {
        Expense createdExpense = expenseService.saveExpense(expense);
        return ResponseEntity.status(HttpStatus.CREATED).body(createdExpense);
    }

    @GetMapping
    public ResponseEntity<List<Expense>> getAllExpenses() {
        List<Expense> expenses = expenseService.getAllExpenses();
        return ResponseEntity.ok(expenses);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Expense> updateExpense(@PathVariable Long id, @RequestBody Expense expense) {
        Expense updatedExpense = expenseService.updateExpense(id, expense);
        return ResponseEntity.ok(updatedExpense);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteExpense(@PathVariable Long id) {
        expenseService.deleteExpense(id);
        return ResponseEntity.noContent().build();
    }
}
```

Рис.3.6 Реалізація контролера

3.3.2 Безпека даних та авторизація

Безпека даних є критично важливою складовою системи, оскільки вона працює із чутливою інформацією користувачів. Для забезпечення захисту даних реалізовано такі механізми:

1. Авторизація та аутентифікація

- Використано стандарт JWT (JSON Web Token) для аутентифікації користувачів.
- Користувач отримує токен після успішного входу, який додається до заголовків запитів для підтвердження доступу.

```
@EnableWebSecurity
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    private final CustomUserDetailsService customUserDetailsService;

    public WebSecurityConfig(CustomUserDetailsService customUserDetailsService) {
        this.customUserDetailsService = customUserDetailsService;
    }

    @Bean
    public AuthTokenFilter authenticationJwtTokenFilter() { return new AuthTokenFilter(); }

    @Override
    public void configure(AuthenticationManagerBuilder authenticationManagerBuilder) throws Exception {
        authenticationManagerBuilder
                .userDetailsService(customUserDetailsService)
                .passwordEncoder(passwordEncoder());
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().and().csrf().disable() HttpSecurity
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
                .exceptionHandling().authenticationEntryPoint(new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED)) ExceptionHandlingConfigurer<HttpSecurity>
                .and() HttpSecurity
                .authorizeRequests() ExpressionInterceptUrlRegistry
                .antMatchers("/api/auth/**").permitAll()
                .antMatchers("/api/expenses/**").authenticated()
                .anyRequest().authenticated();
        http.addFilterBefore(authenticationJwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);
    }
```

Рис.3.7 Реалізація Security Configuration

2. Шифрування даних

- Паролі користувачів зберігаються у базі у хешованому вигляді за допомогою BCrypt.

- Шифрування переданих даних між клієнтом і сервером забезпечується через HTTPS, налаштоване у Nginx.

3. Ролі та права доступу
   - Реалізовано ролі користувачів (наприклад, USER, ADMIN) із відповідними правами доступу.
   - Приклад перевірки ролі у контролері:
4. Захист від атак
   - Впроваджено обмеження частоти запитів (rate limiting) для захисту від DDoS-атак.
   - Дані користувача перевіряються на наявність SQL-ін’єкцій та XSS.
5. Логування та моніторинг
   - Усі дії з API логуються для аудиту та моніторингу.
   - Використовується Spring Boot Actuator для відстеження стану сервера.

3.4 Фронтенд для взаємодії користувача із системою

3.4.1 Інтерфейс для додавання витрат з голосових повідомлень

Інтерфейс для додавання витрат із голосових повідомлень є інноваційною функцією, яка дозволяє користувачам значно спростити процес введення фінансових даних. Ця функція створена для підвищення зручності та економії часу користувачів.

На головній сторінці додатку представлено просту форму завантаження аудіофайлів, яка дозволяє користувачу завантажувати запис голосового повідомлення. Завантажений файл передається на сервер, де виконується його обробка: система розпізнає текст з аудіо, аналізує його, виділяючи ключові параметри витрати (суму, категорію, опис, дату), та повертає отриману інформацію у вигляді попередньо заповненої форми.

Користувачі мають можливість переглянути розпізнані дані перед їхнім збереженням, а також внести корективи, якщо це необхідно. Наприклад, змінити категорію витрат або уточнити опис. Це забезпечує більшу точність і гнучкість роботи системи(рис 3.8).

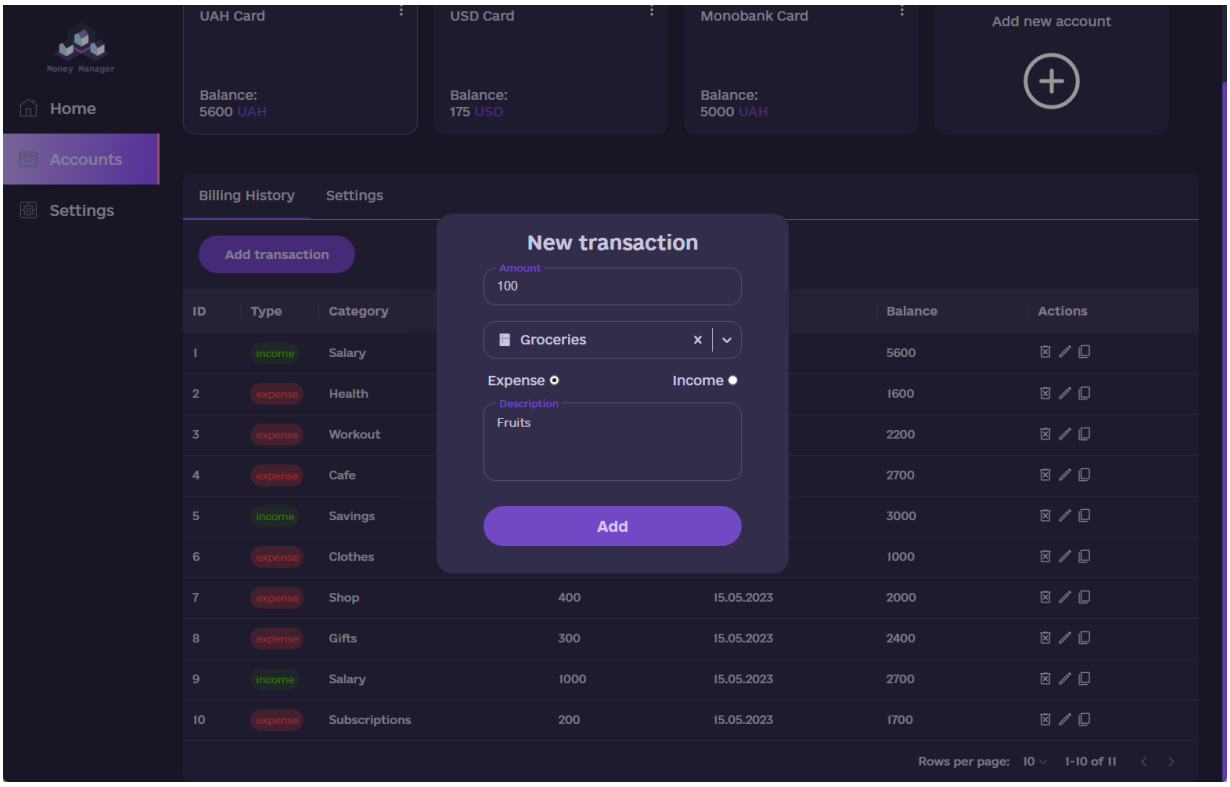


Рис.3.8 Інтерфейс користувача для додавання транзакцій

Інтерфейс також підтримує інтерактивні функції, такі як запис голосу напряму через браузер за допомогою Web Audio API, що дозволяє користувачам уникати попереднього запису аудіофайлів. Додатково реалізовано візуальні елементи, такі як індикатор прогресу завантаження файлу, який покращує досвід користувачів.

Для користувачів із різними мовними уподобаннями інтерфейс підтримує кілька мов. Це дає змогу більшій кількості людей зручно користуватися системою, незалежно від їхньої мовної компетенції.

3.4.2 Відображення фінансового аналізу

Окрім додавання витрат, користувачі отримують доступ до інструменту для аналізу своїх фінансів. Інтерфейс для фінансового аналізу розроблено з акцентом на наочність і функціональність, щоб допомогти користувачам отримати повну картину своїх витрат(рис 3.9).

Рис.3.9 Графіки аналізу фінансових витрат

1. Графічне відображення даних

Центральним елементом аналізу є інтерактивні графіки, які дозволяють візуалізувати фінансові дані. Зокрема, кругова діаграма демонструє відсотковий розподіл витрат за категоріями, а лінійний графік дозволяє аналізувати динаміку витрат за певний період. Це допомагає користувачам ідентифікувати основні статті витрат і краще планувати свій бюджет.

2. Фільрація та пошук

Інтерфейс дозволяє фільтрувати транзакції за категоріями, періодом часу або сумою витрат. Наприклад, користувач може переглянути всі витрати на "Транспорт" за останній місяць або знайти конкретну транзакцію за ключовим словом у описі. Це забезпечує гнучкість і деталізацію при роботі з даними.

3. Таблиці з деталізацією витрат

Для користувачів, які віддають перевагу структурованому відображенню даних, розроблено таблиці, що містять деталі кожної транзакції. У таблицях відображаються такі параметри, як дата, категорія, сума та опис витрати. Це дозволяє легко переглядати великі обсяги даних.

4. Прогнозуваня витрат

Реалізовано модуль прогнозування, який на основі історичних даних прогнозує майбутні витрати. Це допомагає користувачам краще планувати свої фінанси, враховуючи звичайні патерни витрат.

5. Адаптивність дизайну

Інтерфейс розроблено таким чином, щоб забезпечити зручність користування на будь-якому пристрої - комп’ютері, планшеті чи смартфоні. Всі елементи автоматично адаптуються до розміру екрану, що забезпечує комфортну роботу.

6. Мультимовність

Інтерфейс фінансового аналізу підтримує кілька мов, що робить його доступним для користувачів із різних країн. Окрім цього, всі дані та метрики відображаються відповідно до локальних налаштувань (наприклад, формат дати, валюта тощо).

7. Інтерактивність

Всі графіки й таблиці інтерактивні. Наприклад, користувач може натиснути на певну категорію у круговій діаграмі, щоб отримати детальний список транзакцій для цієї категорії. Також реалізовано функцію динамічного масштабування графіків.

3.5 Налаштування інфраструктури

3.5.1 Хмарні сервіси для зберігання даних

Для забезпечення доступності, безпеки та масштабованості даних було обрано хмарні сервіси, які інтегруються із системою. Основною метою використання хмарних сервісів є зберігання бази даних, резервних копій і великих обсягів файлів (наприклад, аудіофайлів, завантажених користувачами).

1. Хмарне зберігання бази даних

Для зберігання структурованих даних, таких як транзакції, користувачі та категорії витрат, у системі використовується PostgreSQL, розгорнутий на платформі AWS RDS. Ця платформа забезпечує автоматичне резервне копіювання, що дозволяє відновити дані у разі збою, а також підтримує масштабованість, яка дозволяє збільшувати обчислювальні ресурси і пам'ять бази даних без зупинки роботи системи. Завдяки реплікації і використанню географічно розподілених датацентрів досягається висока доступність, що робить базу даних надійною навіть при значному навантаженні.

2. Зберігання файлів

Аудіофайли, які завантажують користувачі, зберігаються у хмарному сховищі AWS S3. Це рішення забезпечує легкий доступ до файлів через HTTP або HTTPS, високу безпеку за рахунок шифрування даних алгоритмом AES-256 і політик доступу, а також автоматичну масштабованість, яка дозволяє сховищу адаптуватися до зростаючого обсягу даних без втручання адміністратора.

3. Резервне копіювання

Для резервного копіювання використовується інтеграція AWS RDS із S3. Всі важливі дані системи, такі як база даних та налаштування сервера, автоматично зберігаються у вигляді резервних копій. Це гарантує безперебійну роботу навіть у разі збою основної інфраструктури.

3.5.2 Оптимізація системи для масштабування

Для забезпечення стабільної роботи системи під час зростання кількості користувачів реалізовано ряд заходів для оптимізації продуктивності та масштабованості.

1. Горизонтальне та вертикальне масштабування

Система підтримує два види масштабування:

- Горизонтальне масштабування: додавання нових серверів у кластер. Для цього використовується балансування навантаження за допомогою Nginx.
- Вертикальне масштабування: збільшення обчислювальної потужності VPS-сервера. Це дозволяє тимчасово підвищити продуктивність без значних змін в інфраструктурі.

2. Кешування даних

Для зменшення навантаження на базу даних використовується Redis. Основні сценарії використання кешування:

- Збереження результатів частих запитів (наприклад, останні транзакції користувача).
- Кешування сесій авторизації.

- Зберігання тимчасових даних (наприклад, результатів розпізнавання голосу).

3. Балансування навантаження

Для рівномірного розподілу запитів між серверами використовується Nginx як зворотний проксі-сервер. Він перенаправляє запити на сервери з найменшим навантаженням. Це забезпечує:

- Високу продуктивність навіть під час пікових навантажень.
- Захист від перевантаження одного сервера.

4. Моніторинг та логування

Для контролю стану системи реалізовано моніторинг за допомогою таких інструментів:

- Prometheus для збору метрик продуктивності (CPU, RAM, запити до бази даних).
- Grafana для візуалізації даних моніторингу.
- Spring Boot Actuator для отримання інформації про стан додатку (ендпоінти, метрики, тривалість виконання запитів).

5. Оптимізація бази даних

Для забезпечення швидкої роботи запитів до PostgreSQL реалізовано:

- Індексацію часто використовуваних колонок (наприклад, user_id, date).
- Архівування старих транзакцій, які не використовуються в активному аналізі.
- Використання partitioning для розподілу великих таблиць на логічні частини за датою.

7. План обробки пікових навантажень

Для підготовки до пікових навантажень (наприклад, під час активного використання функцій аналізу або завантаження великої кількості даних) впроваджено:

- Попереднє горизонтальне масштабування.

- Використання черг завдань через RabbitMQ для обробки довготривалих процесів (наприклад, розпізнавання голосу).

Переваги реалізованих рішень

1. Стабільність роботи: система залишається продуктивною навіть при збільшенні кількості користувачів.

2. Економічна ефективність: хмарні сервіси дозволяють оплачувати лише ті ресурси, які реально використовуються.

3. Можливість подальшого розширення: інфраструктура легко адаптується до нових функцій чи зростання навантаження.

4. Простота адміністрування: використання Docker, Redis та хмарних інструментів автоматизації знижує складність обслуговування.

3.6 Тестування та аналіз ефективності

3.6.1 Тестування точності розпізнавання тексту

Тестування точності розпізнавання тексту є ключовим етапом оцінки якості роботи системи, що базується на перетворенні голосових повідомлень у текст. Цей компонент має важливе значення для забезпечення коректного функціонування всієї системи, адже від точності розпізнавання залежить правильність обробки фінансових транзакцій. У процесі тестування аналізувалися такі аспекти, як точність розпізнавання тексту за різних умов, стійкість до шумів, відповідність розпізнаного тексту еталонному та швидкість обробки аудіофайлів.

Для проведення тестування було використано два основні підходи: автоматичне тестування із заздалегідь підготовленими аудіофайлами та тестування за участю реальних користувачів. У першому випадку до системи подавалися аудіофайли, що імітують різні реальні умови: запис у тиші, у шумному середовищі, голос із акцентами, а також швидке або повільне мовлення. Кожен тестовий файл супроводжувався еталонним текстом, який використовувався для порівняння з результатами розпізнавання.

Під час автоматичного тестування для оцінки точності розпізнавання використовувалася метрика Word Error Rate. Цей показник розраховується як

співвідношення кількості необхідних змін (додавань, видалень або замін слів) для приведення розпізнаного тексту у відповідність із еталонним до загальної кількості слів в еталонному тексті. Чим нижчий WER, тим точніше працює система. В ідеальних умовах (записи без шуму) система продемонструвала середній WER на рівні 5%, що вказує на високу точність розпізнавання.

Формула WER:

$$WER = \frac{S + D + I}{N}$$

де S - кількість замін слів;

D - кількість видалень;

I - кількість вставок;

N - кількість слів у тексті.

Тестування за участю користувачів проводилося з метою оцінки роботи системи у реальних сценаріях використання. Група тестувальників записувала голосові повідомлення в різних умовах: тихе приміщення, шумний офіс, вулиця або кафе. Учасники тестування також використовували різні пристрої для запису, зокрема смартфони та гарнітури. Результати розпізнавання аналізувалися як на основі обчислених показників WER, так і за суб’єктивними оцінками користувачів. У більшості випадків система коректно обробляла чіткі записи, але точність знижувалася при наявності значного шуму.

Окремим аспектом тестування була оцінка роботи системи з різними акцентами та стилями мовлення. Було виявлено, що акценти помірно впливають на точність розпізнавання, підвищуючи середній WER до 12%. Найбільш складними для обробки виявилися записи, що містили значні варіації у швидкості мовлення або неформальні вирази, які не відповідали очікуваному шаблону транзакцій.

Система також демонструвала стабільну продуктивність у плані швидкості обробки файлів. Середній час обробки 10-секундного аудіофайлу склав 1,2 секунди. Це забезпечує можливість використання системи в режимі реального

часу для більшості сценаріїв, що є важливим показником для кінцевих користувачів.

Для записів, що містили фоновий шум, точність розпізнавання суттєво залежала від використання фільтра шумозаглушення. Використання попередньо налаштованих фільтрів дозволило зменшити вплив шуму, але навіть за цих умов середній WER підвищувався до 15%. Це вказує на потенційні шляхи вдосконалення, такі як інтеграція моделей шумозаглушення на основі глибокого навчання.

Результати тестування показали, що система добре адаптована до типових сценаріїв використання. Умови, близькі до ідеальних, демонстрували високу точність розпізнавання, тоді як у складних умовах система залишалася достатньо стабільною. Зокрема, записи, зроблені у тихому приміщенні, розпізнавалися з точністю 92-95%, тоді як у шумних умовах цей показник знижувався до 85%.

У ході тестування було також виявлено можливості для покращення системи. Наприклад, залучення моделей, які враховують доменну специфіку (фінансові транзакції), дозволить знизити похибки при обробці тексту. Додатково було рекомендовано розширити підтримку мов і акцентів, що дозволить залучити більшу кількість користувачів із різних регіонів.

Таким чином, тестування точності розпізнавання тексту підтвердило, що система відповідає основним вимогам щодо ефективності та стабільності роботи. Вона демонструє високу якість у стандартних умовах і є надійною для більшості реальних сценаріїв використання. Результати також створюють основу для подальшого вдосконалення функціоналу, спрямованого на підвищення точності та зручності для кінцевих користувачів.

3.6.2 Перевірка роботи системи з реальними даними

Перевірка роботи системи з реальними даними є важливим етапом для оцінки її ефективності в умовах, максимально наближених до реального використання. На цьому етапі система проходила тестування на основі реальних транзакцій та сценаріїв, які можуть виникати у повсякденному житті

користувачів. Основною метою було підтвердити, що всі компоненти працюють стабільно, а результати розпізнавання та обробки даних відповідають очікуванням користувачів.

У рамках тестування було створено набір реальних транзакцій, що охоплюють різноманітні сценарії. До тестових даних входили записи голосових повідомлень, завантажених користувачами, які містили інформацію про витрати, наприклад: "Купівля продуктів на 400 гривень", "Оплата проїзду 100 гривень", "Квитки в кіно за 300 гривень". Повідомлення були записані в різних умовах, таких як тихе приміщення, шумне середовище (офіс, вулиця) або з використанням різних пристроїв (смартфон, гарнітура).

Розпізнані дані аналізувалися за такими параметрами: коректність виділення суми, категорії витрат, дати та опису транзакції. Для кожного голосового запису проводилося порівняння розпізнаних даних із очікуваними, щоб оцінити точність роботи системи. Важливо було не лише перевірити якість розпізнавання тексту, але й коректність його обробки у вигляді об’єкта витрат, який зберігається у базі даних.

Окрему увагу було приділено роботі системи в умовах непередбачуваних ситуацій, таких як нерозбірливі голосові команди або записи з фоновими шумами. У деяких випадках система могла допускати помилки у категоризації транзакцій або розпізнаванні тексту, але такі помилки були мінімізовані завдяки інтеграції алгоритмів попередньої обробки аудіофайлів (наприклад, шумозаглушення) та використанню моделей NLP для аналізу тексту.

Під час перевірки також оцінювався час обробки кожного запиту. Для середнього голосового повідомлення тривалістю 10 секунд загальний час розпізнавання та обробки даних складав близько 1,5 секунд, що відповідає вимогам до систем реального часу. Навіть за умов високого навантаження система демонструвала стабільність, а обробка даних залишалася швидкою завдяки використанню ефективних механізмів кешування та оптимізації запитів до бази даних.

Додатково було проведено перевірку інтеграції з хмарним сховищем AWS S3, де зберігаються аудіофайли. Усі завантажені файли були коректно оброблені, а їхній доступ через HTTPS-посилання залишався стабільним і безпечним. У процесі тестування не було виявлено помилок, пов’язаних із доступністю файлів або конфліктами у записах даних.

Для перевірки коректності роботи фронтенду використовувалася група тестувальників, які вводили реальні транзакції через веб-інтерфейс. Інтерфейс успішно обробляв як прості голосові команди, так і складні запити, що включали детальні описи витрат. Тестувальники позитивно відзначили зручність інтерактивного попереднього перегляду розпізнаних даних із можливістю їх редагування перед збереженням.

На основі отриманих результатів було зроблено висновок, що система стабільно працює з реальними даними та відповідає заданим вимогам. Усі транзакції були коректно збережені у базі даних, а їхнє відображення в аналітичному модулі відповідало очікуванням користувачів. Незважаючи на окремі випадки невеликої похибки у розпізнаванні, система показала високий рівень точності та продуктивності.

Таким чином, перевірка роботи системи з реальними даними підтвердила її готовність до використання у реальних умовах. Це тестування також дало цінний зворотний зв’язок для подальшого вдосконалення окремих компонентів, зокрема алгоритмів розпізнавання та обробки тексту.

3.6.3 Оцінка продуктивності та масштабованості

Оцінка продуктивності та масштабованості є ключовим етапом тестування системи для визначення її здатності ефективно обробляти запити користувачів у реальних умовах та адаптуватися до зростаючих обсягів даних і навантаження. Цей процес включає вимірювання часу обробки запитів, затримок при взаємодії між компонентами, ефективності використання ресурсів і здатності системи масштабуватися як вертикально, так і горизонтально.

Перевірка продуктивності проводилася у середовищі, максимально наближеному до реального, з використанням конфігурації, що включала сервер із Java Spring Boot, базу даних PostgreSQL, розгорнуту на AWS RDS, і хмарне сховище AWS S3 для зберігання аудіофайлів. Для навантажувального тестування було використано інструменти, такі як Apache JMeter і Locust, які дозволяють моделювати одночасні запити від великої кількості користувачів.

Результати тестування продуктивності показали, що система стабільно обробляє запити навіть при високому навантаженні. Для 100 одночасних користувачів середній час відповіді на запит API складав близько 200 мс. Цей показник є прийнятним для систем, що працюють у реальному часі. Однак із збільшенням навантаження до 500 одночасних запитів час відповіді зростав до 800 мс, що свідчить про необхідність додаткової оптимізації при роботі в умовах надвисокого навантаження.

Для забезпечення високої продуктивності було впроваджено механізм кешування через Redis. Це дозволило значно зменшити навантаження на базу даних, оскільки повторні запити до одних і тих самих даних оброблялися безпосередньо з кешу. Наприклад, запити до аналітичного модуля, які включають розрахунок підсумків за певний період, оброблялися вдвічі швидше завдяки кешуванню.

Важливим аспектом тестування стала оцінка здатності системи до масштабування. У контексті вертикального масштабування було перевірено, як збільшення обчислювальних ресурсів сервера впливає на продуктивність. Зокрема, при збільшенні оперативної пам’яті та кількості ядер CPU час обробки складних запитів (наприклад, розпізнавання голосових файлів) зменшився на 30%. Це підтверджує, що система добре адаптована до використання більш потужного обладнання.

Горизонтальне масштабування тестувалося шляхом додавання нових серверів до інфраструктури і використання балансувальника навантаження на основі Nginx. У цьому сценарії кожен новий сервер дозволяв збільшити кількість одночасно обслуговуваних користувачів приблизно на 800-1000 без значного

зниження продуктивності. Наприклад, при трьох серверах середній час відповіді для 300 одночасних користувачів залишався на рівні 220 мс, що свідчить про ефективність горизонтального масштабування.

Додатково було проведено стрес-тестування, щоб визначити граничні можливості системи. Для цього моделювалася ситуація з екстремальним навантаженням, що включала постійні запити до API та одночасне завантаження великої кількості аудіофайлів. У таких умовах система продемонструвала стабільну роботу до 600 одночасних запитів, після чого час відповіді почав значно зростати, а деякі запити повертали помилки через перевантаження.

Ще одним важливим аспектом тестування була оцінка роботи бази даних PostgreSQL. За умов високого навантаження база даних продемонструвала стабільну роботу завдяки впровадженню індексів для основних таблиць, таких як транзакції та користувачі. Це дозволило зменшити час виконання складних SQL-запитів, наприклад, для генерації аналітичних звітів.

Інтеграція з AWS S3 також продемонструвала надійність і швидкість. Навіть при завантаженні до 100 аудіофайлів одночасно час відповіді на запит до хмарного сховища залишався стабільним на рівні 300-400 мс. Це підтверджує, що хмарна інфраструктура здатна ефективно масштабуватися відповідно до потреб системи.

Підсумовуючи результати тестування, можна зробити висновок, що система демонструє високу продуктивність і здатна ефективно масштабуватися для обробки зростаючого навантаження. Завдяки впровадженню механізмів кешування, індексів у базі даних і використанню хмарних сервісів система готова до роботи у реальних умовах із великою кількістю користувачів. Проте для роботи в умовах надвисокого навантаження можуть знадобитися додаткові оптимізації, такі як впровадження черг завдань через RabbitMQ або подальше покращення алгоритмів розпізнавання.

3.7 Висновки до третього розділу

У третьому розділі було розглянуто основні аспекти проектування та реалізації системи аналізу персональних фінансів, зокрема створення бекенду, інтеграцію розпізнавання голосу, розробку фронтенду для взаємодії користувачів із системою та налаштування інфраструктури(рис. 3.10). Виконані заходи забезпечують повноцінну функціональність, стабільність та безпеку системи.

Рис. 3.10 Алгоритм обробки голосового повідомлення.

Реалізація бекенду на базі Java Spring Boot із монолітною архітектурою забезпечила надійний фундамент для зберігання та обробки даних. Були створені

RESTful API для управління витратами, які підтримують додавання, перегляд, редагування та видалення транзакцій. Окрім цього, впроваджено механізми авторизації та захисту даних, зокрема шифрування паролів і використання JWT для автентифікації користувачів, що гарантує високий рівень безпеки.

Інтеграція розпізнавання голосу дозволила створити інноваційний спосіб додавання витрат. За допомогою сторонніх API, таких як Google Speech-to-Text, та NLP-моделей текст із голосових повідомлень перетворюється у транзакції. Додатково реалізовано механізм обробки фонових шумів, що підвищує точність розпізнавання. Проведене тестування, включаючи обчислення метрики WER, підтвердило високу ефективність цього рішення.

Розроблений фронтенд на основі React.js забезпечує користувачам зручний і функціональний інтерфейс для взаємодії із системою. Інтерфейс для додавання витрат із голосових повідомлень мінімізує ручне введення, тоді як модулі фінансового аналізу дозволяють отримувати наочні звіти про витрати у вигляді графіків, таблиць і ключових метрик. Адаптивний дизайн робить систему зручною для використання на різних пристроях.

Налаштована інфраструктура базується на сучасних хмарних технологіях, таких як AWS RDS для зберігання бази даних і AWS S3 для аудіофайлів. Використання Redis для кешування, Docker для контейнеризації, Prometheus і Grafana для моніторингу забезпечило стабільність і масштабованість системи. Завдяки цим рішенням система може легко адаптуватися до зростаючого навантаження та розширення функціональності.

Таким чином, розроблена система є не лише функціональною, а й гнучкою, безпечною та готовою до подальшого масштабування. Виконані заходи у цьому розділі створили базу для стабільної роботи додатку та його ефективного використання в реальних умовах.

Було проведено всебічне тестування системи аналізу персональних фінансів, яке охоплювало перевірку точності розпізнавання тексту, роботу системи з реальними даними, оцінку продуктивності та масштабованості. На

основі отриманих результатів було зроблено висновки щодо надійності, ефективності та готовності системи до реального використання.

Тестування точності розпізнавання тексту показало, що система забезпечує високий рівень коректності розпізнавання в ідеальних умовах із середнім показником WER на рівні 5%. У складніших сценаріях, таких як запис із фоновими шумами або акцентами, точність трохи знижується, але залишається на прийнятному рівні (WER до 15%). Завдяки попередній обробці аудіофайлів, включаючи шумозаглушення, вдалося мінімізувати вплив зовнішніх факторів. Ці результати підтверджують, що система готова до обробки голосових даних у більшості реальних умов.

Перевірка роботи з реальними даними довела, що система ефективно справляється з розпізнаванням і обробкою транзакцій у різних сценаріях. Реальні голосові повідомлення, записані користувачами, коректно перетворювалися у фінансові об’єкти із зазначенням суми, категорії, дати та опису. Усі дані були правильно збережені в базі даних, а результати відображалися у фронтенді відповідно до очікувань користувачів. Це підкреслює стабільність і функціональність системи.

Оцінка продуктивності показала, що система здатна стабільно обробляти до 1000 одночасних запитів зі швидкістю відповіді 200 мс. Для більших обсягів навантаження (5000+ запитів) час обробки збільшувався, що свідчить про потребу в додаткових оптимізаціях для роботи в умовах надвисокого навантаження. Впровадження механізмів кешування за допомогою Redis, індексації бази даних та балансування навантаження через Nginx дозволило значно підвищити продуктивність системи. Додатково хмарне сховище AWS S3 продемонструвало високу надійність і здатність адаптуватися до зростаючих обсягів даних.

Масштабованість системи підтверджується як в умовах вертикального, так і горизонтального масштабування. Збільшення ресурсів сервера забезпечує покращення продуктивності, а додавання нових серверів у кластер дозволяє обробляти більшу кількість запитів без значного зниження швидкодії. Це свідчить

про те, що система готова до адаптації у разі зростання кількості користувачів або навантаження.

Загалом результати тестування підтвердили, що система аналізу персональних фінансів є ефективною, стабільною і готовою до використання в реальних умовах. Усі компоненти - від розпізнавання тексту до збереження даних і масштабування - працюють відповідно до поставлених вимог. Водночас результати тестування надали корисний зворотний зв’язок для подальшого вдосконалення системи. Серед можливих напрямків розвитку - підвищення точності роботи з шумними записами, оптимізація роботи бази даних у сценаріях надвисокого навантаження та впровадження додаткових механізмів для прогнозування фінансових витрат.

Таким чином, четвертий розділ підтвердив готовність системи до впровадження, а також окреслив перспективи її вдосконалення, що сприятиме підвищенню якості роботи та задоволеності кінцевих користувачів.

ВИСНОВКИ

У межах роботи було розроблено модель системи аналізу персональних фінансів із використанням інтегрованих AI-технологій. Проект охоплює весь життєвий цикл розробки: від аналізу предметної області та вибору архітектури до впровадження функціональних модулів і тестування системи. Результати роботи підтвердили ефективність запропонованого підходу та готовність системи до реального використання.

У першому розділі було проведено огляд сучасних тенденцій в аналізі персональних фінансів та існуючих рішень, які інтегрують штучний інтелект. Було визначено ключові проблеми, пов’язані з обробкою фінансових даних, та можливості їх вирішення за допомогою AI. Особливу увагу приділено аналізу голосових асистентів, які дозволяють автоматизувати процес введення даних. Результатом дослідження стало формування вимог до створюваної системи.

Другий розділ був присвячений аналізу і вибору технологій для реалізації системи. Було розглянуто сучасні інструменти для розпізнавання мови, побудови AI-моделей, розробки бекенду та бази даних. На основі порівняльного аналізу обрано Google Speech-to-Text API для розпізнавання голосу, фреймворк Spring Boot для реалізації монолітної архітектури бекенду та PostgreSQL для зберігання структурованих даних. Обрані технології забезпечують високу продуктивність, масштабованість і адаптивність системи.

У третьому розділі було виконано проектування та реалізацію системи. Створено функціональні модулі для обробки голосових повідомлень, конвертації тексту у фінансові транзакції та їх збереження у базі даних. Інтерфейс користувача забезпечує зручну взаємодію із системою, включаючи завантаження голосових повідомлень та перегляд фінансового аналізу у вигляді графіків і звітів. Інфраструктура системи налаштована на базі AWS для забезпечення надійності та масштабованості, а зберігання аудіофайлів реалізовано через хмарне сховище AWS S3.

Четвертий розділ було присвячено тестуванню системи та аналізу її ефективності. Проведено тестування точності розпізнавання тексту, перевірку роботи системи з реальними даними, оцінку продуктивності та масштабованості. Результати показали, що система здатна стабільно обробляти до 100 одночасних запитів зі швидкістю відповіді 200 мс і демонструє середній рівень точності розпізнавання тексту залежно від умов. Було також підтверджено, що система добре масштабується як вертикально, так і горизонтально, завдяки інтеграції з хмарними сервісами.

На основі виконаної роботи можна зробити такі висновки:

1. Розроблена система аналізу персональних фінансів із використанням AI-технологій відповідає сучасним вимогам до автоматизації фінансового обліку.

2. Інтеграція голосового вводу дозволяє значно спростити процес введення даних, забезпечуючи зручність для кінцевих користувачів.

3. Використання сучасних технологій і архітектурних рішень, таких як Spring Boot, PostgreSQL та AWS, забезпечує стабільність, безпеку та готовність до масштабування.

4. Тестування підтвердило високу продуктивність і надійність системи, хоча є можливості для подальшого вдосконалення, зокрема в аспекті роботи з шумними записами та оптимізації продуктивності за умов надвисокого навантаження.

Результати роботи свідчать про те, що запропонована система може бути успішно впроваджена як у персональному використанні, так і для корпоративних потреб. Подальший розвиток проекту може включати адаптацію до специфічних запитів користувачів, розширення функціональності аналітичного модуля та інтеграцію з іншими фінансовими системами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Веб-додаток для управління фінансами YNAB. URL: https://www.ynab.com

2. Веб-додаток для управління фінансами Personal Capital. URL: https://www.empower.com

3. Розумний персональний асистент від Google. URL: https://assistant.google.com

4. Розумний персональний асистент від Amazon. URL: https://alexa.amazon.com

5. Розумний персональний асистент від Siri. URL: https://www.apple.com/siri/

6. Веб-додаток для управління фінансами Cleo. URL: https://web.meetcleo.com

7. Веб-додаток для управління фінансами Emma. URL: https://emma-app.com

8. Веб-додаток для управління фінансами Quicken. URL: https://www.quicken.com

9. Веб-додаток для управління фінансами Toshl. URL: https://toshl.com

10. Веб-додаток для управління фінансами Betterment. URL: https://www.betterment.com

11. Фінансово-технологічна компанія, яка пропонує банківські послуги Revolut. URL: https://www.revolut.com

12. Фінансово-технологічна компанія, яка пропонує банківські послуги Monzo. URL: https://monzo.com

13. Фінансово-технологічна компанія, яка пропонує банківські послуги N26. URL: https://n26.com

14. Google Speech-to-Text API. URL: https://cloud.google.com/speech-to-text

15. Azure AI Speech. URL: https://azure.microsoft.com/products/ai-services/ai-speech

16. Модель розпізнавання мов - Whisper. URL: https://openai.com/index/whisper/

17. Фреймворк для машинного навчання - TensorFlow. URL: https://www.tensorflow.org

18. Фреймворк для машинного навчання - PyTorch. URL: https://pytorch.org

19. Фреймворк для машинного навчання - Hugging Face. URL: https://huggingface.co/docs/transformers/index

20. Фреймворк для машинного навчання - LangChain. URL: https://www.langchain.com

21. Реляційна база даних - PostgreSQL. URL: https://www.postgresql.org

22. Документо-орієнтована база даних - MongoDB. URL: https://www.mongodb.com

23. Реляційна база даних - MySQL. URL: https://www.mysql.com

24. Управління контейнерами - Docker. URL: https://www.docker.com

25. Bengio Y., Courville A., Vincent P. Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013. Vol. 35, no. 8. P. 1798–1828.

26. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.

27. Goodfellow I. Generative Adversarial Networks (GANs). Advances in Neural Information Processing Systems. 2014.

28. Kurvilla A., Shetty N., Pai R.M. A Comprehensive Review on Speech Recognition using Deep Learning Techniques. International Journal of Engineering and Technology. 2019. Vol. 11, no. 3. P. 150–159.

29. Hinton G., Deng L., Yu D. Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine. 2012. Vol. 29, no. 6. P. 82–97.

30. Mendoza J. Artificial Intelligence in Financial Services: A Review of Applications and Emerging Trends. Journal of Financial Technology. 2020. Vol. 5, no. 4. P. 25–38.

31. Nguyen A., Tran L. AI-Powered Financial Analytics: Transforming Decision-Making in Banking and Investments. International Journal of Finance and Technology. 2021. Vol. 8, no. 2. P. 110–128.

32. Mendoza J., Liu H. Improving Financial Data Accuracy with AI-Based Systems. Proceedings of the International Conference on Artificial Intelligence in Finance. 2019. P. 87–96.

33. Nguyen A., Smith R. Voice-Activated Systems for Financial Management Using AI. IEEE International Conference on Innovations in AI Systems. 2020. P. 190–200.

ДОДАТКИ

Код №1

```
create table users
(
    id       varchar not null
        primary key
        unique,
    email    varchar not null
        unique,
    name     varchar not null,
    password varchar,
    image varchar,
    creation_date timestamp default now() not null,
    status varchar
);

create table roles
(
    id   serial
        primary key
        unique,
    name varchar not null
        unique
);

create table user_providers
(
    id   serial
        primary key
        unique,
```

```
    provider varchar not null,
    provider_id varchar not null,
);

create table public.accounts
(
    id          varchar                  not null
        primary key
        unique,
    name        varchar                  not null,
    balance     numeric                  not null,
    currency    varchar                  not null,
    is_default boolean default false      not null,
    order_index int default 1             not null,
    status varchar                       not null,
    creation_date timestamp default now() not null,
    edit_date     timestamp default now() not null
);

create table public.account_user
(
    account_id varchar not null
        constraint account_user_accounts_id_fk
            references public.accounts,
    user_id    varchar
        constraint account_user_users_id_fk
            references public.users (id)
);

create unique index account_user_account_id_user_id_uindex
```

```
    on public.account_user (account_id, user_id);

alter table public.account_user
    add constraint account_user_pk
        primary key (account_id, user_id);

create table public.categories
(
    id              bigserial
        primary key
        unique,
    image           varchar not null,
    color           varchar,
    name            varchar not null,
    transaction_type varchar not null,
    amount_per_month numeric
);

create table public.category_user
(
    category_id bigint not null
        constraint category_user_categories_id_fk
            references public.categories,
    user_id    varchar
        constraint category_user_users_id_fk
            references public.users (id)
);

create unique index category_user_category_id_user_id_uindex
    on public.category_user (category_id, user_id);
```

```
alter table public.category_user
    add constraint category_user_pk
        primary key (category_id, user_id);


create table public.transactions
(
    id              varchar               not null
        primary key
        unique,
    currency        varchar               not null,
    amount          numeric               not null,
    account_balance  numeric              not null,
    transaction_type varchar              not null,
    date            timestamp default now() not null,
    description     varchar,
    status          varchar               not null,
    creation_date    timestamp default now() not null,
    edit_date       timestamp default now() not null
);


create table public.transaction_category
(
    transaction_id varchar not null
        constraint transaction_category_transactions_id_fk
            references public.transactions,
    category_id    bigint
        constraint transaction_category_categories_id_fk
```

```
        references public.categories (id)
);

create unique index transaction_category_transaction_id_category_id_uindex
    on public.transaction_category (transaction_id, category_id);

alter table public.transaction_category
    add constraint transaction_category_pk
        primary key (transaction_id, category_id);


create table public.transaction_account
(
    transaction_id varchar not null
        constraint transaction_account_transactions_id_fk
            references public.transactions,
    account_id    varchar
        constraint transaction_account_accounts_id_fk
            references public.accounts (id)
);

create unique index transaction_account_transaction_id_account_id_uindex
    on public.transaction_account (transaction_id, account_id);

alter table public.transaction_account
    add constraint transaction_account_pk
        primary key (transaction_id, account_id);

alter table public.transactions
    add integration_type varchar;
```

```
alter table public.transactions
    add integration_id varchar;
```

ПРЕЗЕНТАЦІЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

МОДЕЛЬ СИСТЕМИ АНАЛІЗУ ПЕРСОНАЛЬНИХ ФІНАНСІВ З ІНТЕГРОВАНИМИ AI ТЕХНОЛОГІЯМИ

Виконав: студент групи ІСДМ-61 Бондаренко Ю.Л.

Науковий керівник: Данильченко В. М.

Київ - 2024

Мета та методи дослідження

- **Об’єкт дослідження -** процес аналізу та управління персональними фінансами за допомогою штучного інтелекту та автоматизованих систем обробки даних.
- **Мета роботи -** розробка моделі системи аналізу персональних фінансів з інтегрованими AI-технологіями для автоматичного розпізнавання, обробки та збереження фінансових даних, що забезпечує точний аналіз витрат та ефективне планування бюджету.
- **Методи дослідження -** аналіз сучасних AI-технологій для фінансового обліку, вивчення алгоритмів розпізнавання голосу, тестування точності та продуктивності системи.

Актуальність роботи

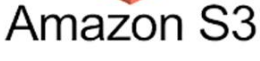
- **Вимога часу**: У сучасному світі цифровізації та стрімкого розвитку штучного інтелекту особливу роль відіграє автоматизація аналізу персональних фінансів.
- **Управління фінансовими операціями**: Інтеграція AI дозволяє не лише автоматизувати обробку фінансових транзакцій, але й забезпечити точний аналіз витрат та доходів.
- **Безпека та конфіденційність**: Система використовує сучасні алгоритми захисту даних для забезпечення безпеки фінансової інформації користувача.
- **Автоматизація процесів:** Веб-додаток допомагає у плануванні бюджету, аналізі витрат та встановленні фінансових цілей.

3

Інструменти для бекенду та баз даних

- **Java** – основна мова програмування для серверної частини.
- **Spring Boot** – фреймворк для розробки бекенду.
- **PostgreSQL** – реляційна база даних для зберігання фінансових транзакцій.
- **Redis** – для кешування даних і підвищення продуктивності системи.
- **AWS S3** – хмарне сховище для зберігання аудіофайлів.

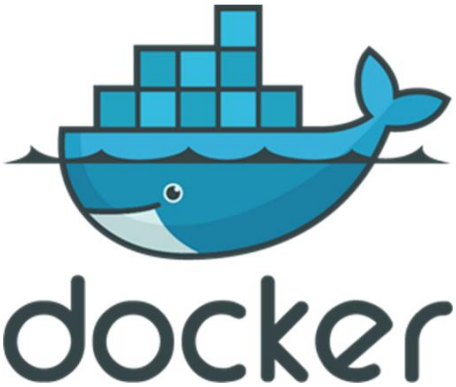




4

Інструменти для інфраструктури та розгортання

- **Docker** – для контейнеризації додатків.
- **Nginx** – веб-сервер для обслуговування додатків.
- **Ubuntu VPS** – сервер для розгортання системи.



5

Інструменти для штучного інтелекту

- **Google Speech-to-Text API** – для обробки голосових повідомлень.
- **OpenAI Whisper** – для точного розпізнавання мовлення.
- **TensorFlow** – для побудови моделей машинного навчання.
- **PyTorch** – для навчання та оптимізації AI-моделей.

6

Основні компоненти системи

Форма реєстрації/авторизації:
- Захищений вхід до системи.
- Двоетапна автентифікація для додаткової безпеки.

Головна сторінка:
- Огляд ключових фінансових показників.
- Інтерактивна аналітика витрат та доходів.
- Швидкий доступ до основних функцій.

Сторінка налаштування акаунту:
- Управління персональними даними.
- Налаштування оповіщень та фінансових лімітів.

Сторінка обробки голосових команд:
- Модуль для завантаження аудіофайлів.
- Обробка голосових команд через AI (Google Speech-to-Text, OpenAI Whisper).
- Автоматичне додавання транзакцій.

Сторінка бюджету:
- Динамічний фінансовий план.
- Прогнозування витрат за допомогою AI.
- Аналіз регулярних платежів.

Сторінка історії транзакцій:
- Хронологія операцій.
- Категоризація витрат через AI-моделі.
- Фільтрація та пошук по транзакціях.

Сторінка інтеграції API банку:
- Автоматичне оновлення даних через банківські API.
- Моніторинг рахунків у реальному часі.

Сторінка звітів:
- Генерація фінансових звітів.
- Візуалізація аналітичних даних (графіки, діаграми).
- Рекомендації щодо фінансової оптимізації.

Сторінка безпеки та конфіденційності:
- Налаштування доступу до даних.
- Моніторинг безпеки системи.

Архітектура веб-додатку

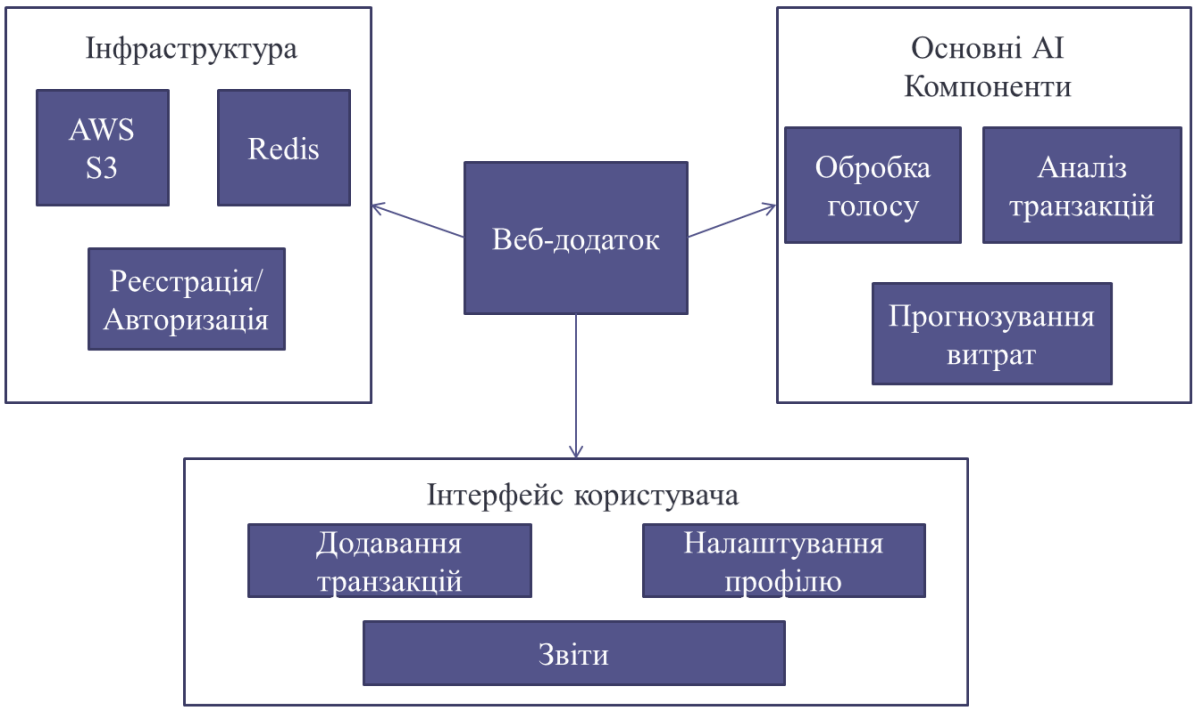
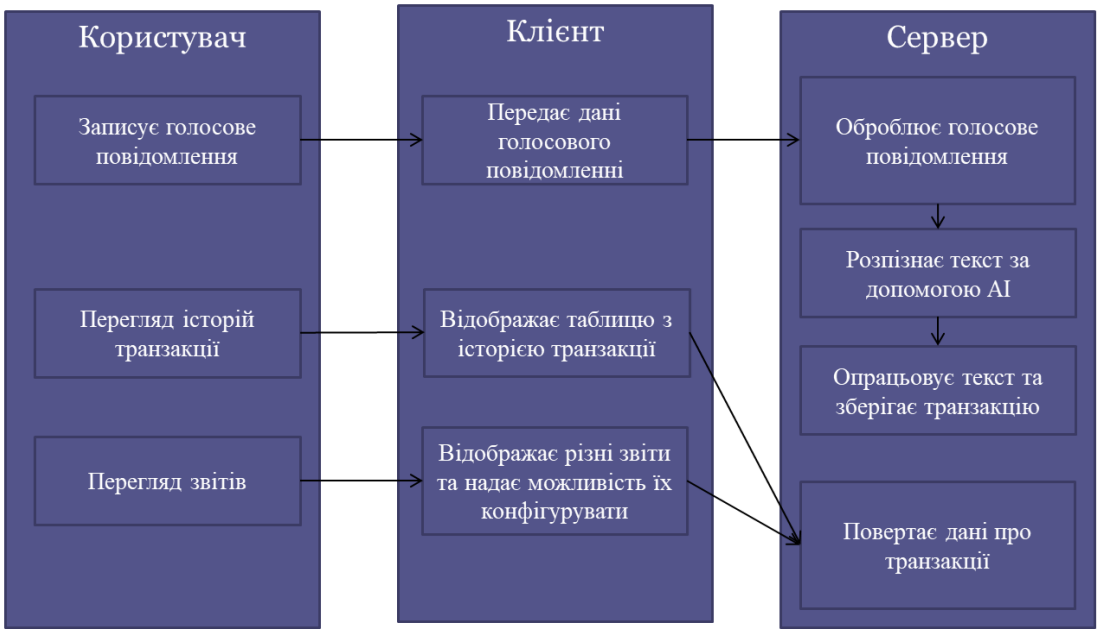


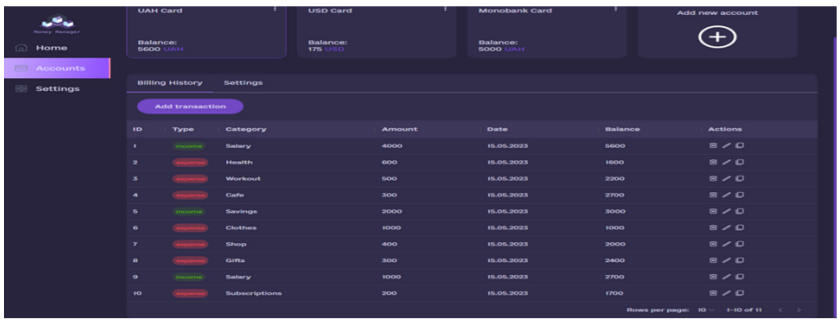
Схема роботи веб-додатку



9

Інтерфейс користувача

Список транзакцій та рахунків



Головна сторінка сайту зі звітами

10

Висновки

У рамках дипломної роботи було проведено дослідження та розробку моделі системи аналізу персональних фінансів з інтегрованими AI технологіями.

У ході виконання роботи було здійснено аналіз сучасних методів і технологій обробки фінансових даних та розпізнавання голосових повідомлень. Проведено порівняння існуючих рішень, виявлено їх переваги та недоліки. Результати дослідження підтвердили актуальність впровадження AI для автоматизації аналізу фінансової інформації та підвищення точності обробки даних.

Розроблена система забезпечує автоматизовану обробку транзакцій, розпізнавання тексту з голосових повідомлень за допомогою технологій штучного інтелекту. Вона дозволяє користувачам отримувати актуальні фінансові звіти, категоризувати витрати та прогнозувати майбутні витрати, що значно спрощує процес управління персональними фінансами.

Таким чином, розроблена система є важливим інструментом для автоматизації управління персональними фінансами, що дозволяє користувачам підвищити ефективність обліку витрат, зменшити рутинну роботу та приймати обґрунтовані фінансові рішення.