МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ АУДІО СИГНАЛІВ ЗАСОБАМИ DJANGO ТА REACT.JS»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
К.О. Терещенко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Терещенку Костянтину Олеговичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка автоматизованої системи розпізнавання аудіо сигналів засобами Django та React.js

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

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

7. Дата видачі завдання 03.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	17.05.2023	
2.	Збір вихідних даних.	07.06.2023	
3.	Обробка методичних та теоретичних	28.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	30.08.2023	
5.	Розробка третього розділу.	08.11.2023	
6.	Оформлення та нормоконтроль	20.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	15.12.2023	

Студент _____ К.О. Терещенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка автоматизованої системи розпізнавання аудіо сигналів засобами Django та React.js»: 49 с., 19 рис., 13 джерел, 4 додатки.

API, DJANGO, FRAMEWORK, JS, MVC, PYTHON, REACT, UML.

Об’єкт дослідження – процес розробки вебзастосунку розпізнавання аудіо сигналів засобами Django та React.js.

Мета роботи – розробити автоматизовану систему розпізнавання аудіо сигналів.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

Сучасний світ вкрай динамічний і насичений інформацією, і однією з найважливіших її складових є аудіо сигнали. Завдяки швидкому розвитку технологій та зростанню інтересу до обробки аудіо даних, автоматизовані системи розпізнавання аудіо сигналів стають надзвичайно актуальними. Ці системи мають великий потенціал у різних сферах, включаючи медицину, безпеку, розваги, освіту і багато інших

Основним завданням системи розпізнавання аудіо є перетворення звукових сигналів у цифрову інформацію, яку можна аналізувати і обробляти за допомогою комп’ютерних алгоритмів. Такі системи використовуються для вирішення різних завдань, включаючи: розпізнавання мови, аналіз емоцій, музичний аналіз, тощо.

Таким чином, за результатами роботи створено зручну та ефективну систему розпізнавання аудіо сигналів засобами Django та React.js.

SUMMARY

Master’s qualifying paper «Development of an Automated Audio Signal Recognition System Using Django and React.js»: 49 p., 19 figures, 13 references, 4 supplements.

API, DJANGO, FRAMEWORK, JS, MVC, PYTHON, REACT, UML.

The object of the study is the process of developing a web application for audio signal recognition using Django and React.js.

The aim of the study is to develop an automated audio signal recognition system.

The methods of research are modeling, design, software, analytical.

The modern world is extremely dynamic and full of information, and one of its most important components is audio signals. Due to the rapid development of technology and the growing interest in audio data processing, automated audio signal recognition systems are becoming extremely relevant. These systems have great potential in various fields, including medicine, security, entertainment, education, and many others.

The main task of an audio recognition system is to convert audio signals into digital information that can be analyzed and processed using computer algorithms. Such systems are used to solve various tasks, including: speech recognition, emotion analysis, music analysis, etc.

Thus, based on the results of the work, a convenient and efficient audio signal recognition system was created using Django and React.js.

ЗМІСТ

ВСТУП

Автоматизовані системи розпізнавання аудіо сигналів можуть революціонізувати багато галузей, допомагаючи виявляти та аналізувати інформацію, яку раніше було б важко або неможливо обробити вручну. Завдяки швидкому розвитку штучного інтелекту, нейронних мереж і обчислювальної потужності, такі системи стають все більш точними та потужними.

Крім того, автоматизовані системи розпізнавання аудіо можуть сприяти вирішенню великої кількості соціальних, медичних, технічних та наукових завдань. Наприклад, вони можуть використовуватися для ранньої діагностики певних медичних захворювань на підставі змін у звучанні голосу пацієнта, для виявлення звуків літаючих об’єктів на аеродромах або для створення інтерактивних освітніх ігор для дітей.

В заключення, розробка автоматизованих систем розпізнавання аудіо сигналів має великий потенціал у різних галузях і може допомогти вирішувати різноманітні завдання, сприяючи розвитку сучасних технологій та покращенню якості життя людей. Ця актуальність надає стимул для подальших досліджень і розвитку цієї області.

Виходячи з цього, було вирішено створити систему, котра би мала зрозумілий інтерфейс та надавала необхідний функціонал користувачам.

Актуальність дослідження: актуальність теми зумовлена необхідністю створення простого та зрозумілого продукту для розпізнавання аудіо сигналів.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити систему розпізнавання аудіо сигналів.

Задачі:

– сформувати вимоги до системи;
– спроєктувати та побудувати архітектуру системи;
– реалізувати систему розпізнавання аудіо сигналів;
– протестувати роботу системи.

Об’єкт – процес розробки вебзастосунку розпізнавання аудіо сигналів засобами Django та React.js.

Предмет – фреймворки веброзробки.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду інструментів розробки та опису системи.

У другому розділі розглянуто етапи проєктування системи, наведено детальний опис прецедентів.

Третій розділ присвячено реалізації та тестуванню роботи системи, наведено детальне керівництво користувача, яке описує процес роботи з системою розпізнавання аудіо сигналів.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Система – вебдодаток створений на основі Django та React.

Django – це відкрите програмне забезпечення, яке надає розробникам вебдодатків повний набір інструментів для швидкої розробки безпечних та масштабованих вебдодатків.

React.js – це бібліотека JavaScript, яка використовується для розробки інтерактивних вебінтерфейсів.

ДВІ – Діаграма Варіантів Використання чи Use Case Diagram.

ДД – Діаграма Діяльності.

ДК – Діаграма Класів.

ДП – Діаграма Послідовності.

ДР – Діаграма Розгортання.

Користувач – людина, котра взаємодіє з системою.

1.1.2 Технічні терміни

API – це набір чітко визначених методів для взаємодії різних компонентів.

DB – база даних.

MVC – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати функціональні можливості розпізнавання аудіо сигналів.

Експлуатаційне призначення системи: система може експлуатуватися користувачем системи.

Мета створення системи – розробка системи для розпізнавання аудіо.

1.2.2 Загальні функціональні можливості системи

Система має надавати користувачам такі можливості:

– завантаження/зміна/видалення аудіофайлу;
– вибір/зміна системи розпізнавання;
– перегляд/завантаження/генерація результату;
– вибір типу файлу для збереження результату.

1.3 Нефункціональні вимоги

Інтерфейс користувача. Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

Підтримка браузерів. Система повинна працювати для наступних браузерів останніх версій:

– Mozilla Firefox;
– Google Chrome;
– Safari;
– Opera.

Вимоги до продуктивності. Система повинна відображати будь-яку сторінку не довше, ніж за 2 секунди.

Система повинна відправляти повідомлення не довше, ніж за 5 секунд.

Вимоги до безпеки. Система не повинна дозволяти завантажувати дані, які можуть бути використані, як експлойти.

Система не повинна надавати доступ до серверної частини.

1.4 Опис предметної області

Предметною областю є розробка системи розпізнавання аудіо сигналів. Даний застосунок повинен надавати користувачам можливість завантажити аудіофайл для розпізнавання. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері та перейти на нього. Система відображає інтерфейс для взаємодії з аудіо.

Після входу до системи користувач має можливість взаємодіяти з такими розділами як: завантаження аудіо, генерація результата, вибір системи розпізнавання, завантаження результату.

Процес створення генерації результата проходить наступним чином. Користувач натискає на кнопку «Завантажити файл», система відображає файли користувача. Після того як користувач обрав файл, він обирає систему розпізнавання з запропонованих і натискає на кнопку «Розпізнати». Система обробляє результат та відображає відповідь користувачу.

У разі помилки, система сповістить про це користувача.

1.5 Опис системи

Зростаючі потреби в автоматизації і обробці даних в сучасному світі призвели до розвитку різноманітних систем розпізнавання сигналів. Однією зі

значущих галузей є розробка автоматизованих систем розпізнавання аудіо сигналів. Ця технологія дозволяє комп’ютерам аналізувати та інтерпретувати звукові сигнали, що відкриває безліч можливостей в різних сферах, включаючи медицину, безпеку, медіа, автомобільну промисловість та інше.

Розробка автоматизованих систем розпізнавання аудіо сигналів стикається зі своїми викликами, такими як нестабільність вхідних даних, складність обробки аудіо інформації, а також проблеми з етикою та конфіденційністю даних. Проте, ці виклики переважаються безліччю переваг, які можуть принести автоматизовані системи розпізнавання аудіо.

Завдяки швидкому розвитку машинного навчання і штучного інтелекту, ми можемо реалізувати багато інноваційних проєктів і покращити якість життя людей. Із зростанням обсягу даних та обчислювальних можливостей ми можемо очікувати подальшого розвитку цієї галузі та більше застосувань в найрізноманітніших галузях.

Зважаючи на вищесказане, було розроблено подібну систему.

Можливості системи:

– завантаження аудіо;
– обробка аудіо;
– розпізнавання аудіо;
– генерація результату;
– вибір системи розпізнавання;
– завантаження результату.

1.6 Огляд інструментів розробки

1.6.1 Django

Django – це відкрите програмне забезпечення, яке надає розробникам вебдодатків повний набір інструментів для швидкої розробки безпечних та

масштабованих вебдодатків. Django базується на мові програмування Python і використовує модель Model-View-Controller (MVC), яка дозволяє розробникам легко розділяти логіку додатку, представлення та роботу з базою даних [5].

Django надає широкий набір функціональності, яка забезпечує зручну роботу з базами даних, URL-ми, формами, аутентифікацією та авторизацією, а також безпекою. Django також підтримує створення API та роботу з різними форматами даних, такими як JSON та XML.

Однією з головних переваг Django є зручність роботи з базою даних. Django використовує ORM (Object-Relational Mapping), який дозволяє розробникам працювати з базою даних на рівні об’єктів, замість написання SQL-запитів. Це спрощує роботу з базою даних та зменшує кількість помилок, що можуть виникнути під час роботи з базою даних [12].

Django також має потужний фреймворк для роботи з URL-адресами та відображенням. Він дозволяє легко налаштовувати URL-адреси та створювати відображення для різних типів запитів HTTP, включаючи GET, POST, PUT та DELETE. Django також надає розширену підтримку для роботи з формами, що дозволяє розробникам легко створювати та обробляти форми на стороні сервера [6].

Для забезпечення безпеки, Django має вбудований механізм аутентифікації та авторизації. Він також має вбудований захист від атак XSS та CSRF. Django також дозволяє налаштовувати рівень доступу до окремих сторінок та функцій додатку для різних користувачів, що забезпечує високий рівень безпеки додатку.

Django також дозволяє створювати API, що дозволяє інтегрувати додаток з іншими сервісами та додатками. Django підтримує різні формати даних, такі як JSON та XML, що дозволяє передавати дані у форматі, який зручний для користувачів [13].

Однією з головних переваг Django є велика кількість сторонніх бібліотек та модулів, які розроблені для Django, що значно спрощує розробку вебдодатків. Такі бібліотеки, як Django REST Framework, дозволяють швидко створювати API для додатку, а бібліотека Django-allauth надає зручний механізм для авторизації користувачів через соціальні мережі.

1.6.2 React.js

React – це бібліотека JavaScript, яка використовується для розробки інтерактивних вебінтерфейсів. Вона була розроблена Facebook і стала дуже популярною серед розробників вебдодатків і сайтів. Однією з головних особливостей React є використання компонентного підходу до побудови інтерфейсу.

Розглянемо, що включають основні концепції React [9].

Компоненти: React дозволяє створювати компоненти, які є незалежними від інших частин додатка і містять у собі власний стан та логіку. Компоненти можуть бути повторно використовані та комбіновані, що робить код більш структурованим і підтримуваним.

Віртуальний DOM: React використовує віртуальний об’єкт моделі документа (DOM), щоб ефективно оновлювати інтерфейс, уникати зайвих маніпуляцій з реальним DOM і покращити продуктивність додатка.

JSX: React дозволяє писати JSX (JavaScript XML) – розширену версію JavaScript, яка дозволяє описувати структуру інтерфейсу, схожу на HTML, безпосередньо в коді JavaScript.

Односторонній потік даних: React пропагує зміни даних через компоненти за допомогою одностороннього потоку даних, що допомагає підтримувати консистентність даних та робить додаток більш передбачуваним.

React став основою для багатьох інших бібліотек і фреймворків, таких як Redux, MobX, Next.js, і багатьох інших. Він використовується для розробки вебдодатків і односторінкових додатків (SPA), а також для створення мобільних додатків за допомогою React Native.

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

Уніфікована мова моделювання (UML) є потужним інструментом для розробки систем і визначення їх архітектури. Вона надає стандартний набір символів і нотацію для створення моделей, які спрощують процес розробки та розуміння системи.

Використання UML під час розробки системи має безліч переваг. Розглянемо їх детальніше.

Візуалізація архітектури: UML дозволяє створювати діаграми, які відображають архітектурну структуру системи, включаючи класи, компоненти, модулі, інтерфейси та їх зв’язки. Це спрощує візуалізацію системи і розуміння її складових.

Аналіз та проєктування системи: UML надає різні види діаграм, такі як діаграми класів, діаграми послідовностей, діаграми діяльності та інші, які допомагають при аналізі і проєктуванні системи. Вони дозволяють розробникам розробляти та вдосконалювати дизайн системи перед фактичною реалізацією.

Узгодженість і спільне розуміння: використання стандартної нотації UML сприяє спільному розумінню всіх учасників проєкту. Всі можуть подивитися на діаграми і зрозуміти структуру та логіку системи.

Документація: UML може бути використаний для створення документації проєкту. Діаграми UML можуть бути включені у технічну документацію, що полегшує комунікацію з клієнтами, менеджментом і іншими зацікавленими сторонами.

Підтримка рефакторингу: UML може бути використаний для ідентифікації можливостей рефакторингу в системі, що дозволяє покращити якість коду та підтримувати систему в актуальному стані.

Спадковість та перевикористання: UML дозволяє моделювати

спадковість та перевикористання класів і компонентів, що сприяє ефективному використанню наявних ресурсів та коду.

Автоматизація генерації коду: деякі інтегровані середовища розробки (IDE) підтримують автоматичну генерацію коду з UML-моделей. Це може значно спростити розробку, зменшити кількість помилок та зекономити час.

Спрощення тестування і валідації: моделі UML можуть служити як основа для створення тестових сценаріїв та валідації системи перед реалізацією.

Загалом, використання UML під час розробки системи сприяє покращенню якості та продуктивності процесу розробки. Вона допомагає зменшити ризики та вартість проєкту, а також забезпечує більшу передбачуваність у виконанні завдань.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є однією з ключових діаграм UML і використовується для моделювання функціональності системи з точки зору її користувачів і зовнішніх агентів. Ця діаграма допомагає ідентифікувати варіанти використання системи і їх взаємозв’язки. Давайте розглянемо основні елементи та приклад створення діаграми варіантів використання.

Розглянемо елементи діаграми варіантів використання.

Актори (Actors): актори представляють зовнішніх користувачів або системи, які спілкуються з системою. Вони зображуються у вигляді піктограм, зазвичай з лівого боку діаграми. Прикладами акторів можуть бути користувачі, інші системи, зовнішні сервіси тощо.

Варіанти використання (Use Cases): варіанти використання відображають функціональність системи з точки зору користувачів або акторів. Вони представляють конкретні можливості, які користувачі можуть використовувати в системі. Вони зображуються у вигляді овалів, розташованих всередині

коробки, що представляє систему.

Взаємозв'язки (Associations): взаємозв'язки відображають, як актори взаємодіють з варіантами використання. Зазвичай вони зображаються стрілками, які з'єднують акторів і варіанти використання. Взаємозв'язки можуть бути однонаправленими або двонаправленими.

На рисунку 2.1 представлена діаграма варіантів використання.

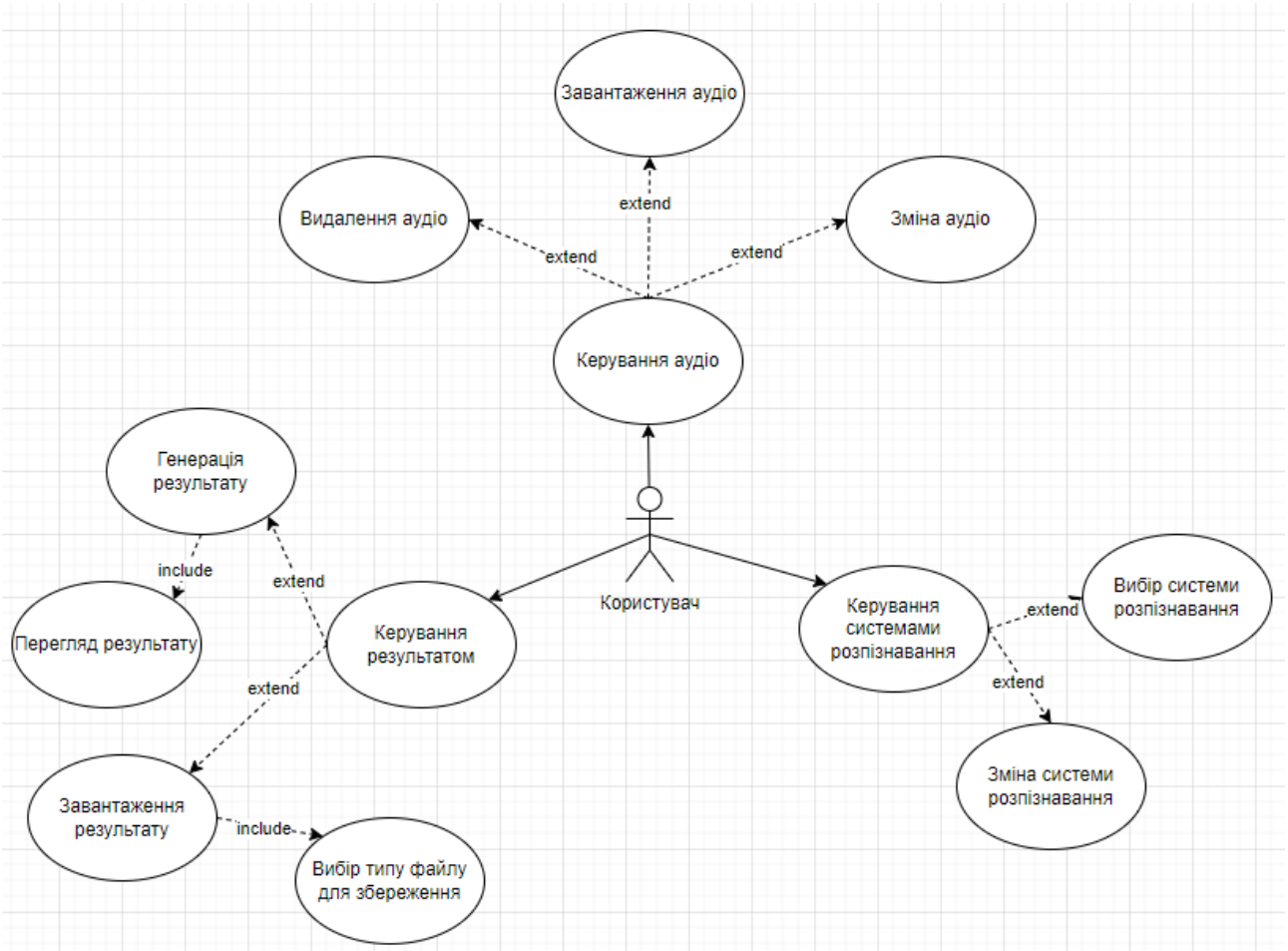

Рисунок 2.1 – Діаграма варіантів використання

На діаграмі представлено актора «Користувач», який відображає функціональні можливості взаємодії з системою.

Виділено один основний варіант використання – «Генерація результату». Після того, як користувач натискає на кнопку «Розпізнати», система надсилає аудіофайл на сервер та обробляє його. Після обробки та збереження результату до локальної БД, система повертає результат розпізнавання аудіо та відображає

його користувачу. У разі помилки, система надсилає повідомляє користувача у вигляді відповідного повідомлення.

Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант використання відповідає послідовності дій для того, щоб користувач міг отримати певний результат.

2.2.1 Опис варіантів використання

Прецедент «Керування аудіо»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з аудіофайлами.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи завантажує, змінює або видаляє аудіофайл. Система оновлює інформацію про поточний аудіофайл.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці системи системи.

Виняткова ситуація 1: файл занадто великий – система відображає відповідне повідомлення. Користувач може обрати новий файл.

Виняткова ситуація 2: файлів з необхідним розширенням не знайдено на пристрої користувача – система відображає відповідне повідомлення.

Прецедент «Завантаження аудіо»

Призначення: даний варіант використання надає можливість користувачу завантажити аудіофайл.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Завантажити файл», система відображає файли користувача. Після вибору, система відображає інформацію про поточний файл.

Передумова: перед початком виконання даного варіанта використання

користувач повинен знаходитися на сторінці системи системи.

Виняткова ситуація 1: файл занадто великий – система відображає відповідне повідомлення. Користувач може обрати новий файл.

Виняткова ситуація 2: файлів з необхідним розширенням не знайдено на пристрої користувача – система відображає відповідне повідомлення.

Прецедент «Зміна аудіо»

Призначення: даний варіант використання надає можливість користувачу змінити аудіофайл.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на назву файлу в полі завантаження файлу, система відображає файли користувача. Після вибору нового файлу, система оновлює інформацію про поточний файл.

Передумова: перед початком виконання даного варіанта використання користувач повинен завантажити аудіофайл.

Виняткова ситуація 1: файл занадто великий – система відображає відповідне повідомлення. Користувач може обрати новий файл.

Виняткова ситуація 2: файлів з необхідним розширенням не знайдено на пристрої користувача – система відображає відповідне повідомлення.

Прецедент «Видалення аудіо»

Призначення: даний варіант використання надає можливість користувачу видалити поточний аудіофайл.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Видалити файл», система оновлює інформацію про поточний файл.

Передумова: перед початком виконання даного варіанта використання користувач повинен завантажити аудіофайл.

Прецедент «Керування системами розпізнавання»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з системами розпізнавання.

Основний потік подій: даний варіант використання починає виконуватися,

коли користувач системи обирає або змінює систему розпізнавання. Система оновлює інформацію.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці системи системи.

Прецедент «Вибір системи розпізнавання»

Призначення: даний варіант використання надає можливість користувачу вибрати систему розпізнавання.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на селект «Система розпізнавання» та обирає один із пунктів, система запам’ятовує інформацію про поточну систему розпізнавання.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці системи системи.

Прецедент «Зміна системи розпізнавання»

Призначення: даний варіант використання надає можливість користувачу змінити систему розпізнавання.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на селект «Система розпізнавання» та обирає пункт, відмінний від поточного, система оновлює інформацію про поточну систему розпізнавання.

Передумова: перед початком виконання даного варіанта використання користувач повинен обрати один з запропонованих пунктів системи розпізнавання.

Прецедент «Керування результатом»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з результатом розпізнавання аудіофайлу.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи генерує або завантажує результат розпізнавання аудіофайлу. Система відображає відповідний результат або інтерфейс взаємодії.

Передумова: перед початком виконання даного варіанта використання

користувач повинен завантажити аудіофайл та обрати систему розпізнавання.

Прецедент «Генерація результату»

Призначення: даний варіант використання надає можливість користувачу згенерувати результат розпізнавання аудіофайлу.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Розпізнати», система надсилає аудіофайл на сервер та обробляє його. Після обробки та збереження результату до локальної БД, система повертає результат розпізнавання аудіо та відображає його користувачу.

Передумова: перед початком виконання даного варіанта використання користувач повинен завантажити аудіофайл та обрати систему розпізнавання.

Виняткова ситуація 1: сервер не відповідає – система відображає відповідне повідомлення. Користувач може спробувати згенерувати результат пізніше.

Виняткова ситуація 2: не вдалось розпізнати файл – система відображає відповідне повідомлення. Користувач може обрати інший файл або систему розпізнавання.

Прецедент «Перегляд результату»

Призначення: даний варіант використання надає можливість користувачу переглядати результат розпізнавання аудіофайлу.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи отримує відповіть від серверу про результат розпізнавання аудіофайлу. Система відображає результат у відповідному полі.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати прецедент «Генерація результату».

Прецедент «Завантаження результату»

Призначення: даний варіант використання надає можливість користувачу переглядати завантажити результат розпізнавання аудіофайлу.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Завантажити результат». Система

формує файл та зберігає його на пристрій користувача.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати прецедент «Генерація результату».

Прецедент «Вибір типу файлу для збереження»

Призначення: даний варіант використання надає можливість користувачу вибрати розширення файлу для завантаження результату.

Основний потік подій: даний варіант використання починає виконуватися, коли авторизований користувач натискає на селект «Розширення файлу» та обирає один із пунктів, система запам’ятовує інформацію про поточне розширення файлу.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати прецедент «Генерація результату».

2.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) є однією з ключових діаграм UML і використовується для моделювання послідовності операцій або дій, що відбуваються в системі або процесі. Діаграма діяльності допомагає візуалізувати робочі процеси та послідовність дій у системі чи бізнес-процесі.

Діаграми діяльності часто використовуються в сферах проєктування програмного забезпечення, управління бізнес-процесами, аналізу робочих процесів і т.д.

Діаграми діяльності допомагають зрозуміти послідовність подій у процесі, спростити аналіз та оптимізацію цих процесів, а також служать для документування інструкцій або алгоритмів. Вони можуть бути створені вручну або з використанням спеціалізованих програм для моделювання діаграм.

Наведемо діаграму діяльності, що описує модель поведінки варіанта використання «Генерація результату». Діаграма представлена на рисунках 2.2 – 2.3.

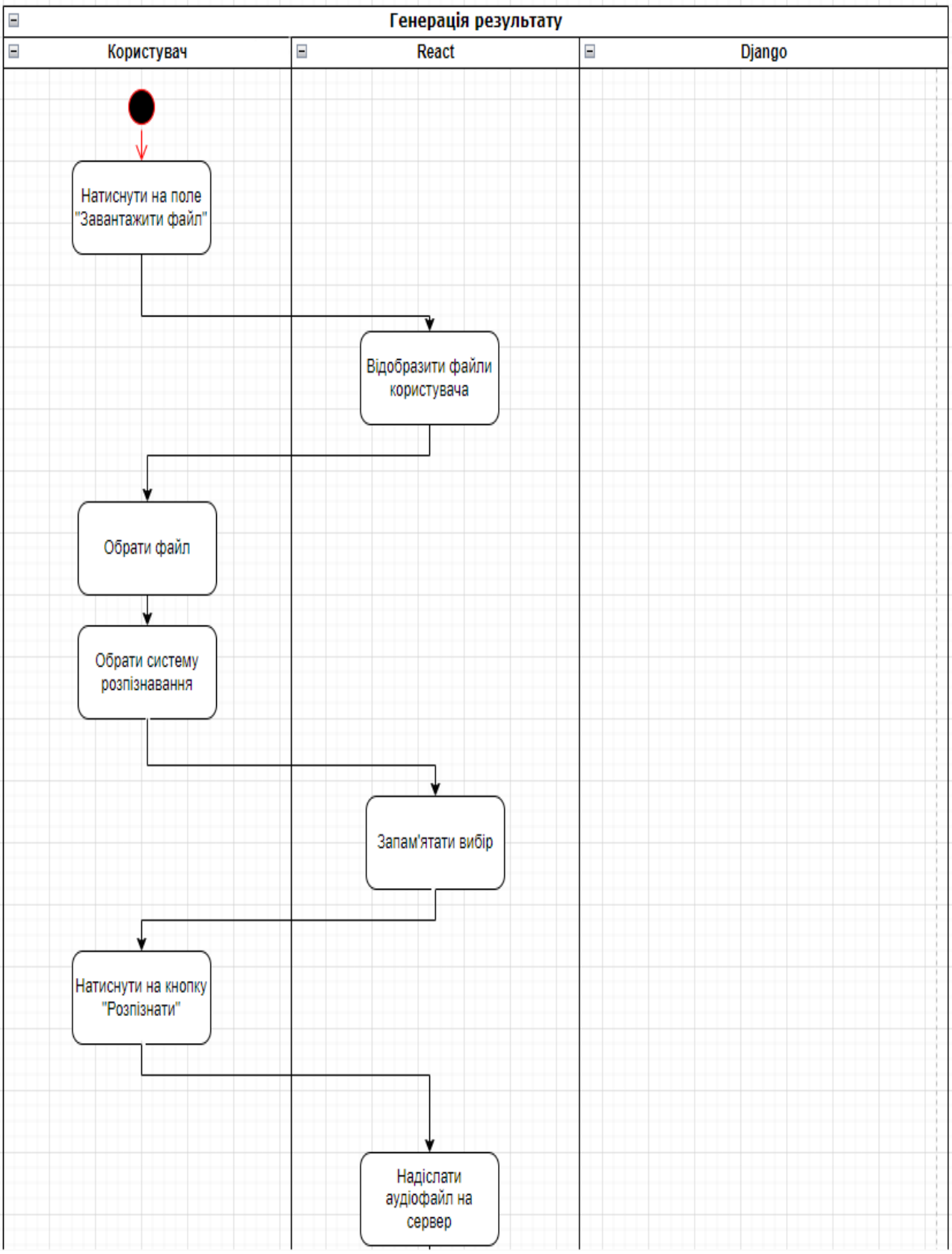

Рисунок 2.2 – Діаграма діяльності

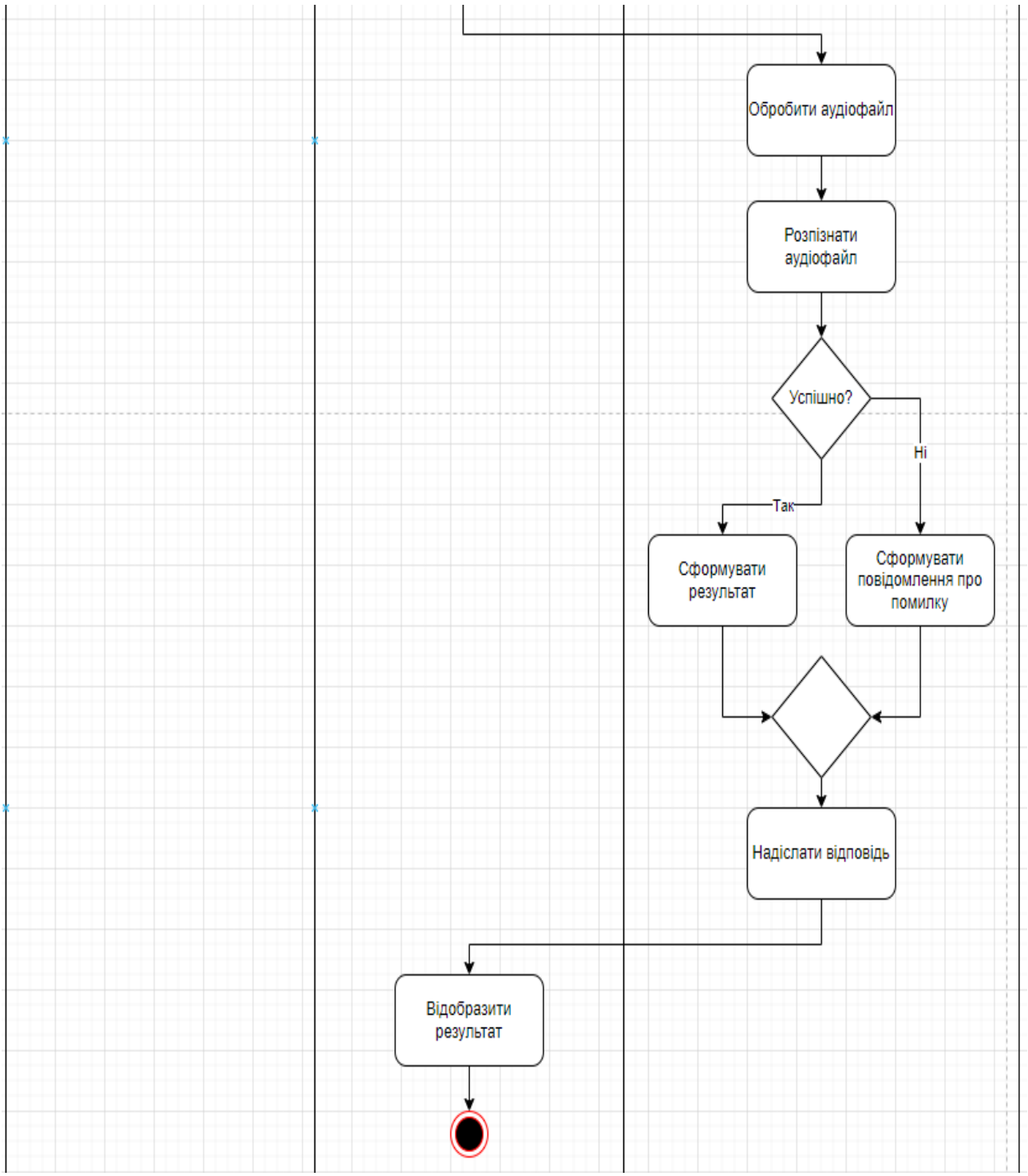

Рисунок 2.3 – Діаграма діяльності

Розглянемо елементи діаграми діяльності.

Дії (Actions): дії представляють окремі кроки чи дії, які виконуються в рамках діаграми діяльності. Дії зазвичай зображуються у вигляді прямокутників і містять назви.

Стан (State): стани показують певні умови або стани системи під час

виконання діаграми діяльності. Вони зображуються у вигляді кругів і зазвичай розміщені біля дій, які переходять у певні стани.

Вибори (Decisions): вибори використовуються для умовного розгалуження в діаграмі діяльності. Вони зазвичай мають умови і дві або більше гілки для подальших дій.

Витоки (Flows): витоки вказують послідовність переходу між діями та станами. Вони зображуються у вигляді стрілок, які показують напрямок послідовності виконання дій.

2.4 Діаграма послідовності

Діаграма послідовності (Sequence diagram) – це графічний інструмент для моделювання взаємодії між об’єктами або компонентами системи в конкретних послідовностях подій або дій. Цей вид діаграми широко використовується в області проєктування програмного забезпечення та аналізу систем для візуалізації, як об’єкти спілкуються між собою у певному порядку під час виконання певного сценарію або функціональної послідовності.

Розглянемо основні елементи діаграми послідовності.

Об’єкти (або учасники): це об’єкти або компоненти системи, які беруть участь у взаємодії. Об’єкти зазвичай позначаються великими прямокутниками з іменем.

Лінії життя: це вертикальні лінії, які виходять від об’єкта і показують «життя» об’єкта впродовж часу взаємодії. Лінії життя позначають, коли об’єкт активний та коли неактивний.

Повідомлення: це стрілки, які вказують на передачу повідомлень між об’єктами в певний момент часу. Повідомлення можуть бути синхронними (блокуючими) або асинхронними (не блокуючими).

Згортки (іноді називають фреймами): згортки використовуються для вираження альтернативних послідовностей або умов у діаграмі. Вони

допомагають спростити діаграму, коли є багато можливих шляхів взаємодії.

Загальні структури: діаграми послідовності також можуть містити загальні структури, які вказують на використання зовнішніх систем або компонентів.

Діаграми послідовності допомагають розуміти, як система взаємодіє з іншими об’єктами і які повідомлення обмінюються між ними в конкретних сценаріях. Вони часто використовуються для аналізу та документування системних взаємодій, а також для виявлення потенційних проблем у дизайні системи.

На рисунку 2.4 описана діаграма послідовності прецедента «Генерація результату».

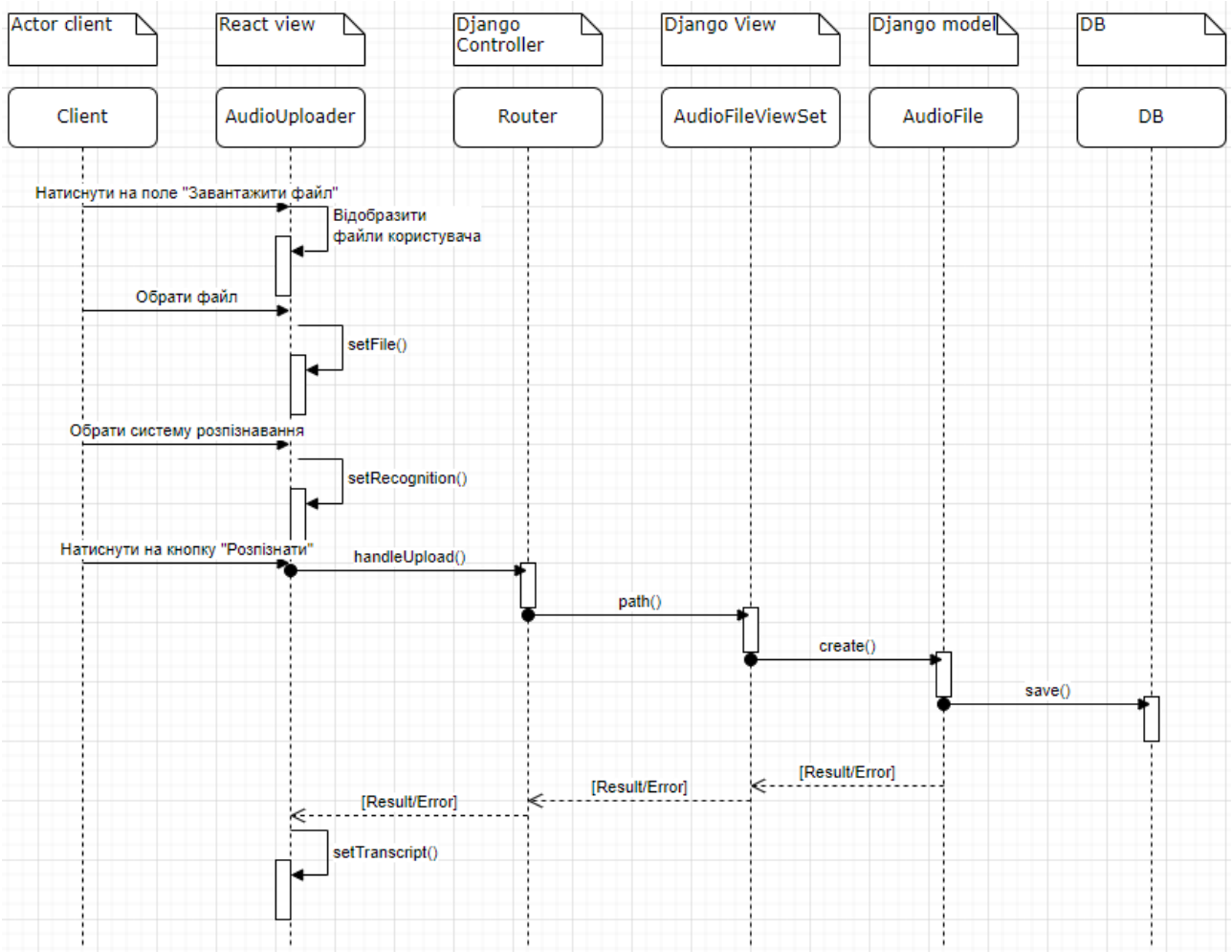

Рисунок 2.4 – Діаграма послідовності

2.5 Діаграма класів

Діаграма класів (Class diagram) – це один із основних видів діаграм UML (Unified Modeling Language), який використовується для моделювання структури класів і взаємозв'язків між ними в об'єктно-орієнтованому програмуванні.

На рисунку 2.5 наведено діаграму класів для системи розпізнавання аудіо сигналів, створеного засобами Django та React.js.

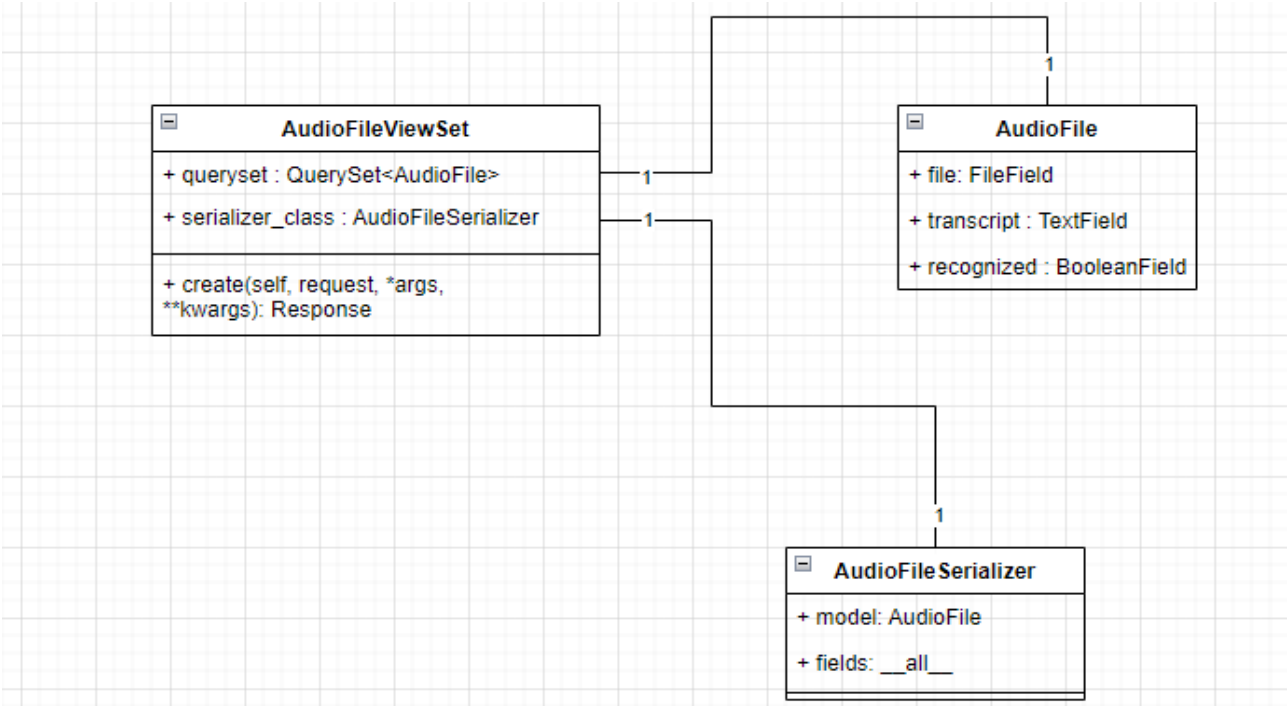

Рисунок 2.5 – Діаграма класів

Розглянемо основні елементи діаграми класів.

Класи: класи представляють абстракції або шаблони для створення об'єктів. Вони зображуються у верхній частині діаграми із вказанням імені класу.

Атрибути: атрибути представляють характеристики класу, тобто дані, які зберігаються в об'єктах цього класу. Вони зображуються під іменем класу та мають тип даних.

Методи: методи вказують на функціональність класу, тобто операції, які можна виконати над об'єктами цього класу. Вони також зображаються під

іменем класу та мають вказані параметри та тип повернення.

Взаємозв'язки: взаємозв'язки показують зв'язки між різними класами. Вони можуть вказувати асоціації, композиції, агрегації та інші типи взаємодій.

Стереотипи та інші анотації: діаграми класів можуть також містити стереотипи та інші анотації, які допомагають уточнити або розширити інформацію про класи та їх взаємозв'язки.

Діаграми класів допомагають розуміти структуру програми, визначити взаємозв'язки між класами та забезпечити основу для подальшого проєктування та реалізації системи. Вони є важливим інструментом для аналізу та проєктування об'єктно-орієнтованих систем.

2.6 Діаграма розгортання

Діаграма розгортання (Deployment diagram) – це один із видів діаграм UML (Unified Modeling Language), який використовується для моделювання фізичної архітектури системи. Ця діаграма дозволяє візуалізувати розташування фізичних об'єктів (які можуть бути апаратними серверами, програмними компонентами, мережевими вузлами і т.д.) та їх взаємозв'язки в рамках системи.

Розглянемо основні елементи діаграми розгортання.

Вузли (Nodes): вузли представляють фізичні об'єкти, такі як сервери, комп'ютери, сенсори, друкувальники тощо. Кожен вузол зазвичай має ім'я та може бути зображений у вигляді круга або еліпса.

Артефакти (Artifacts): артефакти – це програмні компоненти, які розгортані на фізичних вузлах. Вони позначаються у вигляді прямокутників або інших геометричних фігур та зв'язуються з вузлами за допомогою ліній.

Зв'язки (Connections): зв'язки між вузлами та артефактами показують способи взаємодії між ними. Це може бути мережеві з'єднання, протоколи комунікації, канали передачі даних і т.д.

Ресурси (Resources): ресурси представляють різні об'єкти, які можуть бути

спільно використані різними вузлами. Вони допомагають моделювати загальні ресурси, такі як бази даних, сховища або інші об’єкти, до яких можна звертатися з різних вузлів.

Діаграми розгортання допомагають інженерам систем та розробникам візуалізувати фізичну структуру системи, розташування її компонентів і взаємозв’язки між ними. Це важливо для розробки, впровадження і управління розгортанням складних систем, особливо для розподілених програмних рішень та мережевих архітектур.

На рисунку 2.6 наведено діаграму розгортання для системи розпізнавання аудіо сигналів, створеного засобами Django та React.js.

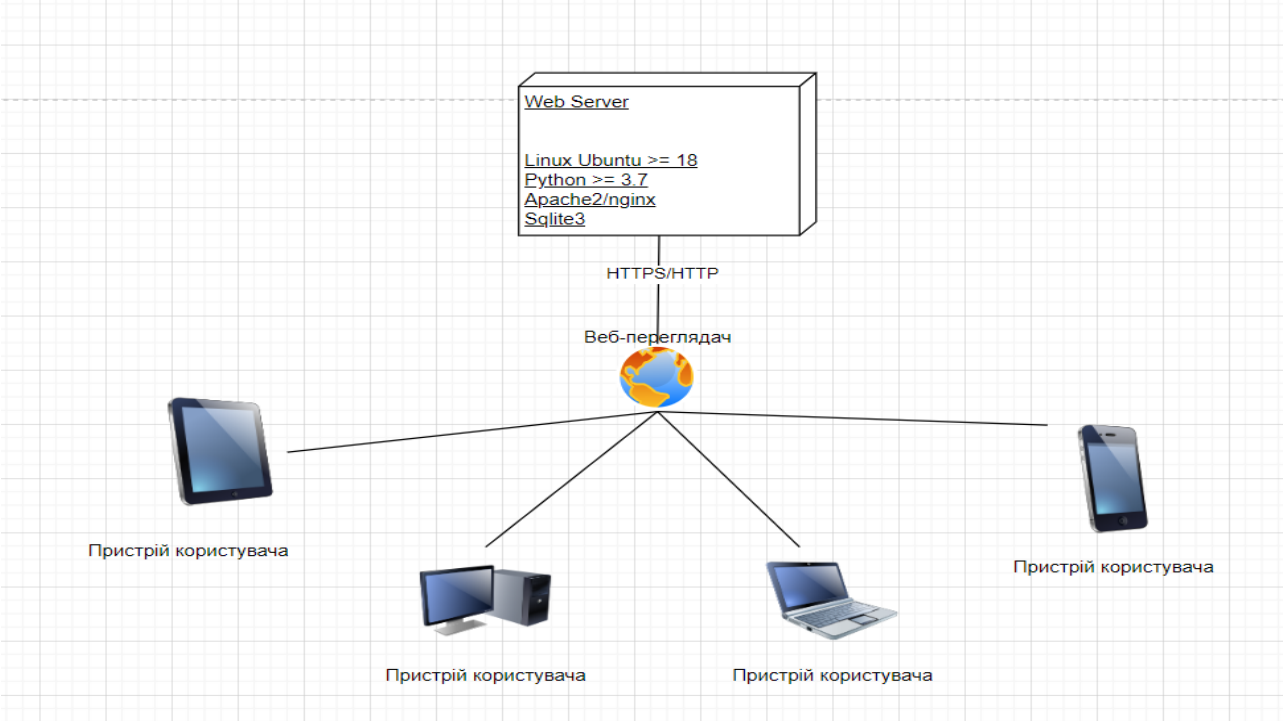

Рисунок 2.6 – Діаграма розгортання

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації був використаний python-фреймворк Django та js-бібліотеку React.js.

Sqlite3 – це вбудована реляційна база даних, яка зберігає дані у локальних файлових базах даних. Вона досить легка у використанні, не потребує окремого сервера баз даних та надає можливість прямого доступу до бази даних з допомогою SQL-запитів [11].

3.2 React-компонент

React-компонент – це фундаментальний елемент у React, бібліотеці для створення інтерфейсів користувача. Компоненти в React використовуються для розділення користувацького інтерфейсу на невеликі і незалежні частини, що полегшує управління та розробку [9].

Розглянемо основні риси React-компонента [2].

Стан (State): багато React-компонентів мають стан, який є внутрішнім об’єктом, що описує, як компонент «відчуває» себе в даний момент часу. Зміни стану викликають перерендеринг компонента.

Властивості (Props): компоненти можуть приймати властивості (props), які їм передаються від батьківських компонентів. Вони визначають параметри, які можна налаштовувати при використанні компонента.

Методи життєвого циклу (Lifecycle methods): компоненти в React мають методи життєвого циклу, такі як componentDidMount, componentDidUpdate, і componentWillUnmount, які викликаються в різних етапах життєвого циклу компонента.

JSX (JavaScript XML): React використовує JSX, розширення JavaScript, що дозволяє описувати структуру інтерфейсу користувача у вигляді тегів, схожих на HTML або XML.

Приклад такого компоненту наведено на рисунку 3.1, а код компоненту проєкту наведено в Додатку А.

```jsx
import React, { Component } from 'react';

class MyComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      // стан компонента
    };
  }

  componentDidMount() {
    // логіка, яка виконується після монтування компонента
  }

  render() {
    return (
      <div>
        {/* JSX, що відображається компонентом */}
        <h1>{this.props.title}</h1>
        <p>{this.state.someData}</p>
      </div>
    );
  }
}

export default MyComponent;
```

Рисунок 3.1 – Приклад компоненту

3.3 Django-модель

У Django моделі використовуються для визначення структури даних і взаємодії з базою даних. Модель Django є класом Python, який наслідується від базового класу *django.db.models.Model*. Кожне поле моделі представляє стовпці бази даних [7].

Приклад моделі наведено на рисунку 3.2, а код моделей проєкту наведено в Додатку Б.

```
class AudioFile(models.Model):
    file = models.FileField(upload_to='audio/')
    transcript = models.TextField(blank=True)
    recognized = models.BooleanField(default=False)
```

Рисунок 3.2 – Приклад моделі

3.4 Django-представлення

У Django представлення (views) відповідають за обробку HTTP-запитів і повернення HTTP-відповідей. Вони містять бізнес-логіку вебдодатку і забезпечують взаємодію між моделями (дані) та шаблонами (вигляд) [4, 8].

Приклад представлення наведено на рисунку 3.3.

```
class AudioFileViewSet(viewsets.ModelViewSet):
    queryset = AudioFile.objects.all()
    serializer_class = AudioFileSerializer

    def convert_to_wav(self, file_path):
        audio = AudioSegment.from_file(file_path)
        return audio.export(format="wav")

    def create(self, request, *args, **kwargs):
        file = request.data.get('file', None)
        recognizer = None

        if file is None:
            return Response({'file': 'This field is required.'}, status=status.HTTP_400_BAD_REQUEST)

        recognition_method = request.data.get('recognition_method', 'google')  # Метод розпізнавання

        try:
            # Конвертація аудіофайлу у формат WAV
            wav_file = self.convert_to_wav(file)
        except Exception as e:
            return Response({'error': f'Error converting audio to WAV: {e}'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

Рисунок 3.3 – Приклад представлення

Код представлень проєкту наведено в Додатку В.

3.5 Основні класи та функції системи

Так як основним прецедентом системи є «Генерація результату», то основними класами та функціями системи будуть ті, що надають можливість завантажувати, обробляти та надсилати результат.

Головними класами основного бізнес-процесу є:

– AudioUploader – відповідає за дії пов’язані з Django API, а саме містить логіку по завантаженню файлу та відображенню результату;
– AudioFileViewSet – обробляє та відображає дані, отримані від методів AudioUploader. Містить логіку щодо зчитування файлу, визначення системи розпізнавання, обробку та повернення результату.

Детально ознайомитися з кодом проєкту можна у додатку Г.

3.6 Тестування проєкту

Unit-тест. Основна ідея unit-тестування полягає в тому, щоб перевірити, чи працюють окремі компоненти програми (наприклад, функції, методи чи класи) так, як очікується, і чи вони генерують правильні результати при різних вхідних умовах.

Unit тести дозволяють розробникам переконатися, що функції або методи працюють правильно до того, як вони будуть інтегровані в більші структури програми. Це допомагає виявляти та виправляти помилки на ранніх етапах розробки, покращує стабільність програми та полегшує роботу в команді розробників [10].

Приклад такого тестування наведено на рисунку 3.4.

```
const mockAxios = new MockAdapter(axios);

describe('AudioUploader Component', () => {
  beforeEach(() => {
    mockAxios.reset();
  });

  it('renders without crashing', () => {
    render(<AudioUploader />);
  });

  it('handles file change correctly', async () => {
    const { container } = render(<AudioUploader />);
    const fileInput = container.querySelector('input[type="file"]');
    const testFile = new File(['test content'], 'test.mp3', { type: 'audio/mp3' });

    fireEvent.change(fileInput, { target: { files: [testFile] } });

    await waitFor(() => {
      expect(screen.getByText('Файл: test.mp3')).toBeInTheDocument();
    });
  });
```

Рисунок 3.4 – Unit-тест

Integration-тест. Вид тестування програмного забезпечення, який спрямований на перевірку взаємодії між різними компонентами, модулями або системами для впевненості в їхній правильній роботі як єдиної сукупності. Основна мета – переконатися, що окремі частини програми правильно працюють разом, коли їх об’єднують у цілісну систему [1, 3].

На рисунку 3.5 наведено приклад integration-тестування AudioFileViewSet.

```
class AudioFileIntegrationTest(APITestCase):
    def setUp(self):
        self.file = SimpleUploadedFile("test.mp3", b"file_content", content_type="audio/mp3")

    def test_create_audio_file_with_google_recognition(self):
        with patch('recognition.views.sr.Recognizer') as mock_recognizer:
            mock_recognizer.return_value.record.return_value = "mock_transcript"

            data = {
                'file': self.file,
                'recognition_method': 'google',
            }

            response = self.client.post('/api/audiofiles/', data, format='multipart')

            self.assertEqual(response.status_code, status.HTTP_201_CREATED)
            self.assertEqual(AudioFile.objects.count(), 1)
            self.assertEqual(AudioFile.objects.first().transcript, 'mock_transcript')
```

Рисунок 3.5 – Integration-тест

3.7 Керівництво користувача

3.7.1 Рівень підготовки користувача

Користувач повинен мати базові навички роботи з ПК, та з web-браузером. Знайомство з Керівництвом користувача.

3.7.2 Підготовка до роботи

Запуск системи. Доступ до сайту здійснюється через мережу Інтернет за допомогою звичайного web-браузера. Адреса сайту в мережі Інтернет: https://audio.loc. Для коректної роботи клієнтської частини повинен використовуватися браузер Google Chrome, Mozilla Firefox, Opera, Safari.

При вході на Сайт користувач потрапляє на сторінку для завантаження аудіо. На Сайті розрізняються наступні групи користувачів: користувач – особа, що має взаємодіє з системою.

3.7.3 Загальний інтерфейс

При вході на сайт, система відображає інтерфейс взаємодії. Якщо користувач ще не виконував дії по розпізнаванню аудіо, то система відображає повідомлення «Жодного файлу ще не було розпізнано» (див. рис. 3.6).

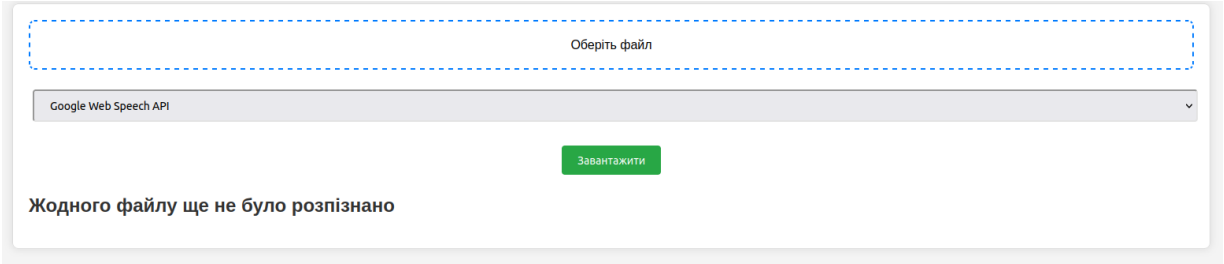

Рисунок 3.6 – Форма входу до системи

3.7.4 Елементи інтерфейсу

Інтерфейс можна умовно поділити на 4 частини:

– поле для завантаження файлу;

– вибір системи розпізнавання;

– виконання дії;

– відображення результату.

Розглянемо взаємодію з кожним елементом окремо.

Поле для завантаження файлу відповідає за завантаження файлу з пристрою користувача та підтримує формати mp3 і wav. Якщо користувач ще не обирав файл, то поле буде відображати напис «Оберіть файл» (рис. 3.7). При натисканні на дане поле система відкриває файли з пристрою користувача (рис. 3.8). Після того, як користувач обрав файл, система відображає його назву та розширення (рис. 3.9).

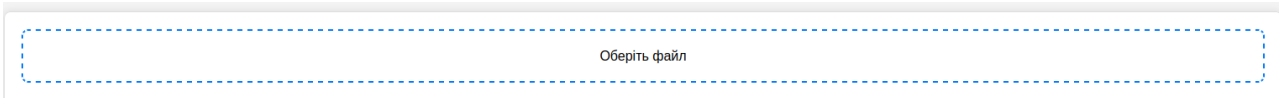

Рисунок 3.7 – Поле для завантаження файлу

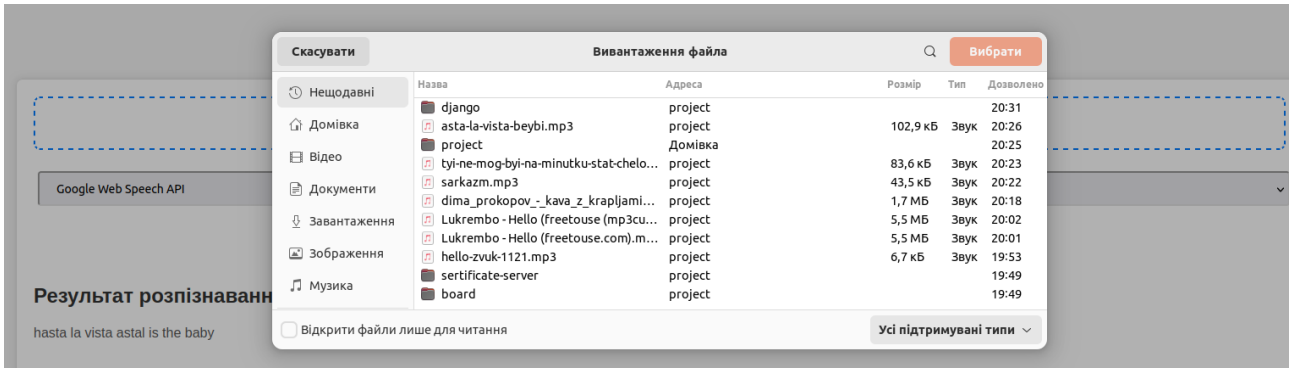

Рисунок 3.8 – Файли користувача

Рисунок 3.9 – Завантажений файл

Вибір системи розпізнавання відповідає за алгоритм (бібліотеку), який буде використовувати наша система для розпізнавання аудіо. За замовчуванням встановлено значення «Google Web Speech API» (рис. 3.10). Для зміни системи розпізнавання користувач повинен натискати на поле списку та вибирати потрібну систему (рис. 3.11).

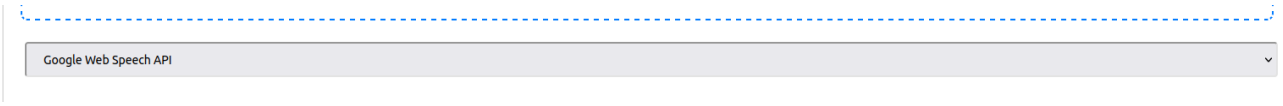

Рисунок 3.10 – Система за замовчуванням

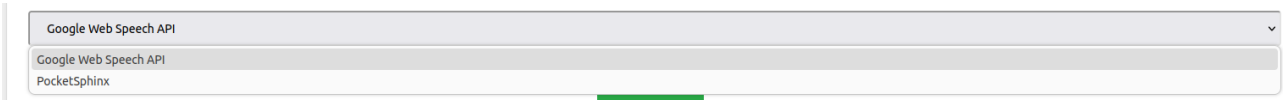

Рисунок 3.11 – Вибір системи

Після завантаження файлу та вибору системи розпізнавання, користувач повинен натиснути на кнопку «Завантажити», система надсилає запит на сервер та відображає результат (рис. 3.12, 3.13).

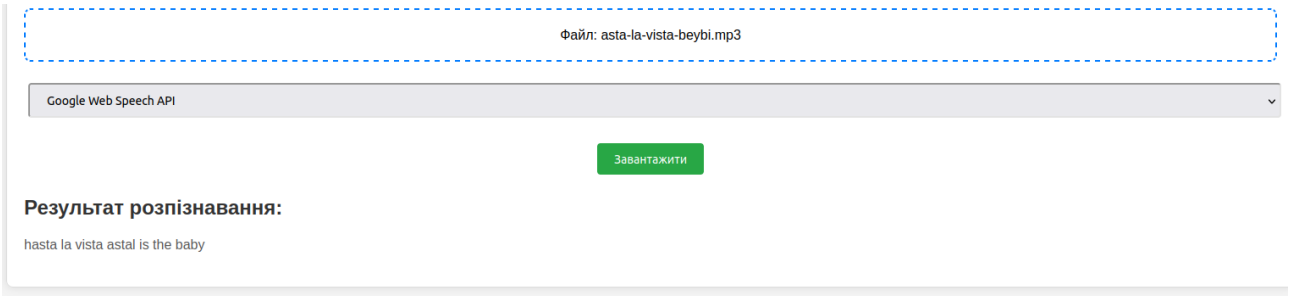

Рисунок 3.12 – Результат розпізнавання Google Web Speech API

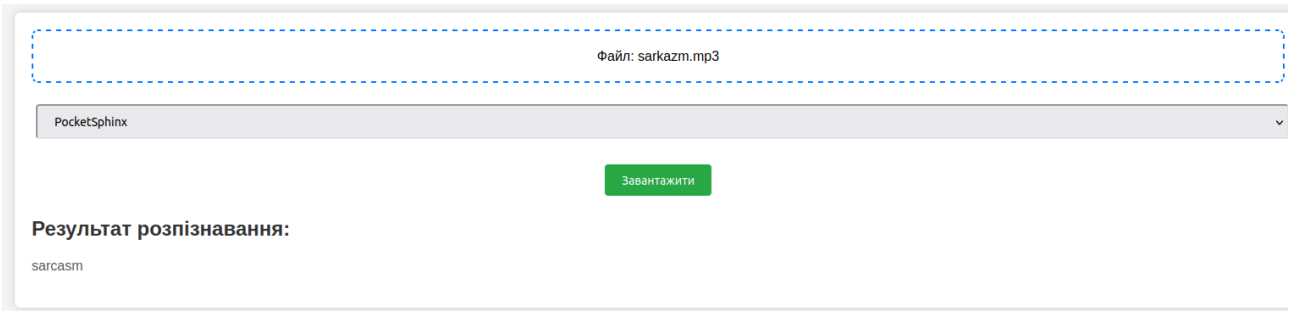

Рисунок 3.13 – Результат розпізнавання PocketSphinx

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку автоматизованої системи розпізнавання аудіо сигналів. Для створення цієї системи було обрано фреймворк Django зі сторони сервера, та React.js зі сторони клієнта, за їх можливості у сфері створення подібних систем.

У відповідності з метою кваліфікаційної роботи було розроблено автоматизовану систему розпізнавання аудіо із застосуванням наступних технологій:

- Django для реалізації серверної частини;
- React.js для реалізації клієнтської частини;
- SQLite для зберігання даних.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- сформовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність)), також проведено огляд предметної області та інструментів розробки;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності, класів та розгортання; надано детальний опис прецедентів);
- реалізовано систему розпізнавання аудіо (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проєкту);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mele A. Django 4 By Example: Build powerful and reliable Python web applications from scratch. Birmingham : Packt Publishing, 2022. 766 p.
2. Barklund M., Mardan A. React Quickly. Manning, 2023. 456 p.
3. Web Development with Django: Learn to build modern web applications with a Python-based framework / B. Shaw, S. Badhwar and otc. Birmingham : Packt Publishing, 2021. 826 p.
4. Django Developer Documentation. URL: https://docs.djangoproject.com/ (дата звернення: 11.09.2023).
5. Matthes E. Python Crash Course, 3rd Edition: A Hands-On, Project-Based Introduction to Programming. No Starch Press, 2023. 552 p.
6. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. Sebastopol : O'Reilly Media, 2022. 1012 p.
7. Dinder M. Becoming an Enterprise Django Developer: Discover best practices, tooling, and solutions for writing and organizing Django applications in production. Birmingham : Packt Publishing, 2022. 526 p.
8. Robbins P. Python Programming for Beginners: The Complete Guide to Mastering Python in 7 Days with Hands-On Exercises – Top Secret Coding Tips to Get an Unfair Advantage and Land Your Dream Job. Independently published, 2023. 114 p.
9. Raymond L. Reason, React, Respond: The LangChain Developer's Handbook Reason, React, Respond: The LangChain Developer's Handbook. Independently published, 2023. 128 p.
10. Roberts A. React JS: A Beginner's Guide to Mastering React JS for Front-End Development. Independently published, 2023. 187 p.
11. SQLite Developer Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 12.09.2023).
12. Siahaan V., Rismon Hasiholan Sianipar. SQLite for Data Analysis And

Visualization With Python GUI. Independently published, 2022. 412 p.

13. Vincent W. S. Django for Beginners: Build websites with Python and Django. WelcomeToCode, 2020. 221 p.

ДОДАТОК А

Компонент

```
import React, { useState } from 'react';
import axios from 'axios';
import './App.css';

function AudioUploader() {
  const [file, setFile] = useState(null);
  const [transcript, setTranscript] = useState('');
  const [recognitionMethod, setRecognitionMethod] = useState('google');
  const [fileName, setFileName] = useState('');

  const handleFileChange = (e) => {
    const selectedFile = e.target.files[0];
    setFile(selectedFile);
    setFileName(selectedFile.name); // Оновлення імені файлу
  };

  const handleMethodChange = (e) => {
    setRecognitionMethod(e.target.value);
  };

  const handleUpload = async () => {
    const formData = new FormData();
    formData.append('file', file);
    formData.append('recognition_method', recognitionMethod);

    try {
```

```
      const response = await axios.post('http://localhost:8000/api/audiofiles/', formData,
{
        headers: {
          'Content-Type': 'multipart/form-data',
        },
      });

      setTranscript(response.data.transcript);
    } catch (error) {
      console.error('Error uploading file:', error);
    }
  };

  return (
    <div className="container">
      <div          className="section          input-section"          onClick={()          =>
document.getElementById('fileInput').click()}>
        {!fileName && <label htmlFor="fileInput">Оберіть файл</label>}
        <input          type="file"          id="fileInput"          accept=".mp3,          .wav"
onChange={handleFileChange} />
        {fileName && <span>Файл: {fileName}</span>}
      </div>

      <div className="section">
        <select value={recognitionMethod} onChange={handleMethodChange}>
          <option value="google">Google Web Speech API</option>
          <option value="pocketsphinx">PocketSphinx</option>
        </select>
      </div>
```

```
      <div className="section">
        <button onClick={handleUpload}>Завантажити</button>
      </div>

      {transcript && (
        <div className="section result">
          <h2>Результат розпізнавання:</h2>
          <p>{transcript}</p>
        </div>
      )}

      {!transcript && (
        <div className="section result">
          <h2>Жодного файлу ще не було розпізнано</h2>
        </div>
      )}
    </div>
  );
}

export default AudioUploader;
```

ДОДАТОК Б

Моделі

```
from django.db import models


class AudioFile(models.Model):
    file = models.FileField(upload_to='audio/')
    transcript = models.TextField(blank=True)
    recognized = models.BooleanField(default=False)

    def save(self, *args, **kwargs):
        if self.file:
            self.file._committed = True
        super().save(*args, **kwargs)
```

ДОДАТОК В

Представлення

```
from rest_framework import viewsets, status
from rest_framework.response import Response
from .models import AudioFile
from .serializers import AudioFileSerializer
import speech_recognition as sr
from pocketsphinx import LiveSpeech
from pydub import AudioSegment


class AudioFileViewSet(viewsets.ModelViewSet):
    queryset = AudioFile.objects.all()
    serializer_class = AudioFileSerializer

    def convert_to_wav(self, file_path):
        audio = AudioSegment.from_file(file_path)
        return audio.export(format="wav")

    def create(self, request, *args, **kwargs):
        file = request.data.get('file', None)
        recognizer = None

        if file is None:
            return Response({'file': 'This field is required.'},
status=status.HTTP_400_BAD_REQUEST)

        recognition_method = request.data.get('recognition_method', 'google')  # Метод
```

розпізнавання

```
        try:
            # Конвертація аудіофайлу у формат WAV
            wav_file = self.convert_to_wav(file)
        except Exception as e:
            return Response({'error': f'Error converting audio to WAV: {e}'},
status=status.HTTP_500_INTERNAL_SERVER_ERROR)

        audio_file = AudioFile(file=wav_file)

        if recognition_method == 'google':
            recognizer = sr.Recognizer()
        elif recognition_method == 'pocketsphinx':
            recognizer = LiveSpeech()
        else:
            return Response({'recognition_method': 'Invalid recognition method.'},
status=status.HTTP_400_BAD_REQUEST)

        try:
            with sr.AudioFile(wav_file) as source:
                if recognition_method == 'google':
                    audio = recognizer.record(source)
                    transcript = recognizer.recognize_google(audio)
                elif recognition_method == 'pocketsphinx':
                    transcript = ""
                    for phrase in recognizer:
                        transcript += str(phrase) + " "

                audio_file.transcript = transcript
```

```
            audio_file.recognized = True
            audio_file.save()
            serializer = AudioFileSerializer(audio_file)
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        except sr.UnknownValueError:
            return Response({'error': 'Speech recognition could not understand audio.'},
status=status.HTTP_400_BAD_REQUEST)
        except sr.RequestError as e:
            return Response({'error': f'Speech recognition error: {e}'},
status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

ДОДАТОК Г

Посилання на Git

https://bitbucket.org/s_var_og/audio/src/master/