ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка інтелектуального сервісу для генерації анотацій до аудіофайлів»

(тема кваліфікаційної роботи українською мовою)

«Development of an intelligent service for generating annotations for audio files»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
ТОЛМАЧЕВСЬКИЙ Олександр Олександрович
(прізвище, ім'я, по-батькові здобувача)

Керівник к.т.н., доцент Перелигін Б.В. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент кафедри інженерії ПЗ Національного Університету "Одеська політехніка", Зіноватна С. Л.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ___ від ___ _____ 2025 р.

Завідувачка кафедри

_____ КАЗАКОВА Надія
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № ___ від _____ 2025 р.

Оцінка _____ / _____ / _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ КОПИЧЕНКО Іван
(підпис) (прізвище,ім'я)

Одеса 2025

АНОТАЦІЯ

Мета роботи полягає у розробці системи автоматичного розпізнавання мовлення з використанням сучасних методів машинного навчання. Для створення моделі застосовано фреймворк PyTorch, бібліотеку NeMo для роботи з акустичними моделями, а також Python як основну мову програмування. Для демонстрації роботи моделі було розроблено веб-сервіс за допомогою Flask для бекенду та React для фронтенду. Додатково використано мовну модель, побудовану з використанням KenLM для покращення якості розпізнавання.

У результаті виконання роботи створено ефективну систему, яка може бути використана для автоматичної транскрипції аудіо, створення голосових помічників та інших додатків. Рішення відзначається високою гнучкістю та потенціалом для подальшого розвитку на основі сучасніших архітектур.

Ключові слова: розпізнавання мовлення, PyTorch, NeMo, Flask, React, KenLM, автоматична транскрипція.

ABSTRACT

The aim of this work is to develop a speech recognition system using modern machine learning methods. The model was created using the PyTorch framework, the NeMo library for working with acoustic models, and Python as the main programming language. A web service was developed to demonstrate the model's capabilities, using Flask for the backend and React for the frontend. Additionally, a language model built with KenLM was implemented to improve recognition quality.

As a result of this work, an efficient system was created that can be used for automatic audio transcription, development of voice assistants, and other applications. The solution is characterized by high flexibility and potential for further development based on more modern architectures.

Keywords: speech recognition, PyTorch, NeMo, Flask, React, KenLM, automatic transcription.

ЗМІСТ

ВСТУП

Останні роки характеризуються стрімким розвитком та широкомасштабним впровадженням систем розпізнавання мовлення, що зумовлено зростанням їхньої доступності, точності та ефективності. Цей розвиток безпосередньо пов’язаний із експоненційним збільшенням обсягів аудіоданих, що створюються щодня у найрізноманітніших сферах – від освіти й медіа до бізнесу та повсякденного спілкування.

Мовна інформація є цінним джерелом даних, на основі яких можуть вирішуватись складні завдання – такі як тематичне моделювання, аналіз емоцій, інформаційний пошук, класифікація контенту тощо. Це створює високу потребу в автоматизованих, інтелектуальних підходах до обробки аудіоінформації, зокрема в інструментах, здатних трансформувати мовлення в текст та формувати його змістовне узагальнення у вигляді анотації.

Цифрова обробка аудіофайлів та генерація коротких описів до них стають важливою частиною взаємодії людини з інформаційним середовищем. Технології розпізнавання мовлення дедалі частіше застосовуються у повсякденному житті: голосові команди керують смартфонами, автомобілями та системами «розумного дому», а віртуальні асистенти – Siri, Alexa, Google Assistant – здатні інтерпретувати природну мову й виконувати різноманітні дії.

Особливої популярності набуває автоматична транскрипція аудіо – як ефективний спосіб швидко отримувати текстові версії лекцій, інтерв’ю, нарад чи подкастів. У поєднанні з технологіями обробки природної мови (NLP) це відкриває нові можливості для генерації змістовних, коротких анотацій, які значно спрощують доступ до інформації.

Таким чином, розробка сервісів, здатних об’єднати обробку аудіо, розпізнавання мовлення та автоматичне створення анотацій, є актуальним і перспективним напрямом, що відповідає сучасним потребам інформаційного суспільства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Сучасні підходи до обробки аудіо інформації

Обробка аудіоінформації є важливою складовою сучасних інформаційних систем. Завдяки стрімкому розвитку штучного інтелекту, машинного навчання та обчислювальних потужностей, з'явилася можливість автоматизовано аналізувати звукові дані, розпізнавати мову, класифікувати звуки та витягати з аудіофайлів смислову інформацію.

Ключовими етапами обробки аудіо є: попередня фільтрація (шумозаглушення, нормалізація), перетворення в спектрограму, фреймінг, витяг ознак (наприклад, MFCC – Mel-Frequency Cepstral Coefficients), а також подальший аналіз на основі цих ознак.

У випадку роботи з мовленням, особливу роль відіграють алгоритми автоматичного розпізнавання мови (ASR – Automatic Speech Recognition), які дають змогу трансформувати мовлення в текст для подальшої обробки.

Сучасні системи ASR базуються переважно на глибоких нейронних мережах – рекурентних (RNN), трансформерах (Transformer), або їх комбінаціях (наприклад, модель Wav2Vec 2.0 від Facebook AI, Whisper від OpenAI). Вони забезпечують високу точність розпізнавання навіть у складних акустичних умовах.

Після транскрипції тексту з аудіо постає задача виділення ключових думок та узагальнення інформації. Тут застосовуються методи обробки природної мови (NLP), зокрема:

– extractive summarization – виділення ключових речень з тексту;

– abstractive summarization – формування нового короткого змісту на основі розуміння контексту.

Для цього використовують моделі, які тренуються на великих корпусах текстів [1].

1.2 Існуючі рішення та сервіси автоматичної генерації анотацій

На ринку існують як комерційні, так і відкриті рішення для генерації анотацій. Розглянемо найбільш популярні сервіси транскрипції з можливістю створення коротких резюме.

Першим проаналізуємо платформу Descript (рис. 1.1).

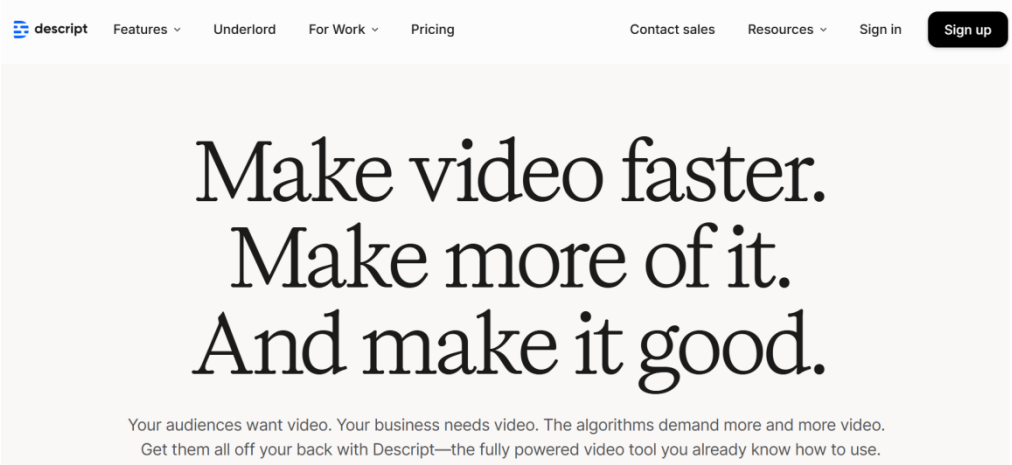

Рисунок 1.1 – Платформа Descript

Descript – це хмарна платформа для обробки аудіо та відео, яка поєднує можливості редагування, транскрипції, колаборації та штучного інтелекту. Вона особливо популярна серед подкастерів, журналістів, маркетологів і відеоредакторів завдяки унікальному підходу: редагування мультимедійного контенту відбувається через редагування транскрипованого тексту [2].

Ключовими можливостями платформи є наступні:

– автоматична транскрипція аудіо та відео з високою точністю (підтримка кількох мов, зокрема англійської);

– текстове редагування аудіо/відео: видалення або заміна фрагментів аудіо шляхом редагування тексту транскрипції;

– overdub – функція генерації синтетичного голосу для заміни окремих слів або створення нових фраз голосом користувача;

– інтегроване відеоредагування;

– спільна робота над проєктами;

– субтитри та титри для відео;

– публікація та експорт контенту в різних форматах.

Переваги платформи Descript:

– інтуїтивний інтерфейс – платформа орієнтована на користувачів без технічного досвіду;

– інноваційна функція редагування через текст – значно пришвидшує процес монтажу;

– якісна транскрипція англійською мовою з можливістю ручного доопрацювання;

– overdub – унікальний інструмент, що дозволяє швидко виправити або додати фрагменти мовлення;

– хмарна синхронізація та колаборація – зручно для командної роботи;

– автоматичне видалення слів-паразитів («uh», «um», тощо).

Недоліки платформи Descript:

– обмежена підтримка мов – платформа орієнтована переважно на англомовний ринок (українська мова офіційно не підтримується);

– платна модель – повний функціонал доступний лише у підписці (безкоштовна версія має суттєві обмеження);

– потребує хорошого інтернет-з’єднання – через хмарний характер платформи;

– обмежена точність при поганій якості запису – шум, акценти або технічні проблеми знижують якість транскрипції;

– overdub вимагає навчання моделі на вашому голосі, що займає час і доступне лише в платних тарифах.

Descript – це потужна, зручна і технологічно просунута платформа для обробки аудіо- та відеоматеріалів, яка демонструє інноваційний підхід до редагування через текст. Вона ідеально підходить для англомовного

контенту, однак має певні обмеження для користувачів, які працюють з іншими мовами або без доступу до платної версії.

Наступним роглянемо сервіс Otter.ai (рис. 1.2).

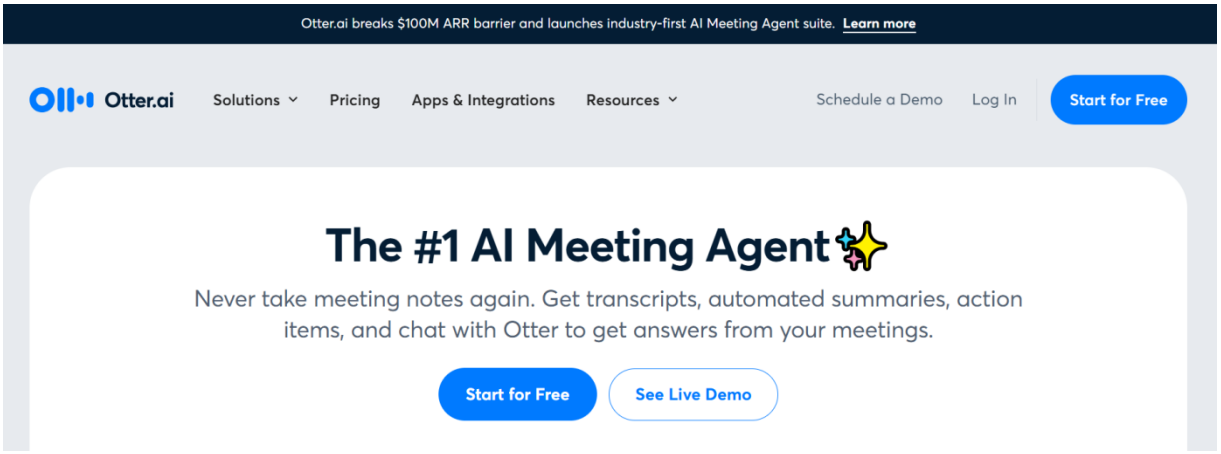

Рисунок 1.2 – Сервіс Otter.ai

Otter.ai – це вебсервіс і мобільний додаток для автоматичної транскрипції мовлення в текст у режимі реального часу або з попередньо записаних файлів. Його часто використовують у бізнесі, освіті, журналістиці та для особистих нотаток [3].

Основні можливості сервісу:

– автоматична транскрипція в реальному часі;
– підтримка завантаження аудіо/відеофайлів;
– визначення спікерів та маркування ролей;
– синхронізація з Zoom, Google Meet, Microsoft Teams;
– автоматичне збереження ключових фраз;
– експорт у форматах txt, docx, pdf;
– наявні мобільні додатки (iOS/Android).

Переваги сервісу:

– висока точність транскрипції англійською мовою;
– підтримка реального часу;

– дружній та простий інтерфейс;

– інтеграція з відеоконференціями;

– безкоштовний тариф із базовим функціоналом.

Недоліки сервісу:

– обмеження за кількістю хвилин у безкоштовному тарифі;

– платна модель для розширених функцій;

– підтримка лише деяких мов (українська – відсутня);

– немає функцій редагування контенту або створення анотацій.

Ще один аналог – це Whisper (OpenAI) [4].

Whisper – це відкритий фреймворк для автоматичного розпізнавання мовлення (ASR), розроблений компанією OpenAI. Модель доступна як open-source і підтримує багатомовне розпізнавання, включно з українською.

Можливості системи:

– підтримка понад 50 мов;

– висока точність навіть при наявності шуму;

– працює офлайн (можна запускати на локальному сервері);

– розпізнає тайм-коди, багатомовність, акценти;

– доступ до коду та можливість адаптації під проєкт;

– можна використовувати для побудови кастомних сервісів.

Переваги системи:

– безкоштовна та відкрита модель;

– підтримка української мови;

– надійна якість навіть при фоновому шумі;

– гнучкість – інтегрується в будь-яку систему.

Недоліки системи:

– відсутній готовий інтерфейс – потрібна технічна реалізація;

– вимоги до ресурсів (особливо при використанні великих моделей);

– відсутність функцій генерації анотацій «з коробки» – лише розпізнавання;

– не має хмарного api від openai – потрібно розгортати самостійно або через сторонні сервіси.

Whisper AI – це нейронна мережа, розроблена та відкрита для використання, яка досягає рівня стійкості та точності розпізнавання мови на рівні людського. Це автоматична система розпізнавання мови, навчена на 680 000 годин мультитаскових даних різних мов, зібраних з інтернету. Використання такого широкого та різноманітного набору даних дозволяє покращити стійкість до акцентів, фонового шуму та технічної лексики, а також здійснювати транскрипцію кількома мовами та переклад англійською.

Архітектура Whisper є простий енд-то-енд підхід, реалізований у вигляді кодера-декодера Transformer. Вхідний аудіофайл розбивається на ділянки по 30 секунд, конвертується в логарифмічну крейду-спектрограму та подається на вхід кодеру. Декодер навчений передбачати відповідний текст, а також виконувати додаткові завдання, такі як ідентифікація мови, тимчасові мітки на рівні фраз, транскрипція мультитональної мови та переклад мови на англійську.

Одна з ключових особливостей Whisper полягає у використанні широкого та різноманітного набору даних, що включає не лише англійську, а й інші мови.

В таблиці 1.1 надається порівняльна характеристика аналогів, що розглядалися.

Таблиця 1.1 – Порівняльна характеристика систем

Характеристика	Descript	Otter.ai	Whisper (OpenAI)
Автоматична транскрипція	+	+	+
Підтримка української мови	–	–	+
Можливість офлайн-роботи	–	–	+
Генерація анотацій / короткого змісту	+ (частково)	–	–

Продовження таблиці 1.1

Характеристика	Descript	Otter.ai	Whisper (OpenAI)
Редагування через текст	+	–	–
Підтримка багатьох мов	–	частково	+
Простота використання (готовий інтерфейс)	+	+	–
Можливість кастомізації під власні задачі	–	–	+
Інтеграція з іншими сервісами (Zoom, API тощо)	+	+	– (потрібна реалізація)

1.3 Постановка задачі

У рамках роботи передбачається розробка інтелектуального сервісу, що дозволяє автоматично генерувати анотації до аудіофайлів на основі їхнього змісту. Сервіс повинен бути здатним:

– приймати на вхід аудіофайл;

– обробляти звукові дані та трансформувати їх у текстову форму;

– аналізувати зміст отриманого тексту;

– створювати короткий, зрозумілий виклад основних ідей – анотацію;

– забезпечувати зручний доступ до результатів через інтерфейс.

Таким чином, система має поєднувати методи обробки аудіо, автоматичного розпізнавання мовлення та узагальнення текстів.

1.4 Вимоги до сервісу

Розглянемо функціональні вимоги:

– завантаження аудіофайлу користувачем;

– автоматичне розпізнавання мовлення;

– генерація анотації з тексту транскрипції;

– виведення результату в текстовій формі;

– можливість експорту результату.

Пердставимо функціональні вимоги у вигляді діаграми варіантів використання (рис. 1.3).

Рисунок 1.3 – Діаграма варіантів використання

Основні сценарії використання.

1. Завантажити аудіофайл.

Актор: Користувач.

Опис: Користувач завантажує аудіофайл у систему для подальшої обробки.

Передумова: Користувач авторизований.

Результат: файл успішно збережено на сервері або в хмарному сховищі системи.

2. Транскрибувати мовлення.

Актор: Користувач.

Опис: система автоматично розпізнає мовлення на аудіофайлі та перетворює його в текст.

Передумова: аудіофайл вже завантажено.

Результат: Користувач отримує текстову транскрипцію аудіофайлу.

3. Згенерувати анотацію.

Актор: Користувач.

Опис: система аналізує транскрипт та формує коротку текстову анотацію – суть змісту.

Передумова: транскрипція завершена.

Результат: Користувач бачить згенеровану анотацію.

4. Завантажити результат.

Актор: Користувач.

Опис: Користувач може завантажити результати – транскрипт і/або анотацію – у вигляді файлу.

Передумова: дані вже згенеровані.

Результат: файл збережено локально.

5. Переглянути історію.

Актор: Користувач.

Опис: Користувач переглядає історію попередніх обробок аудіофайлів.

Передумова: Користувач авторизований.

Результат: відображено список попередніх завантажень із транскрипціями/анотаціями.

6. Реєстрація/Вхід.

Актор: Користувач, Адміністратор.

Опис: Користувач проходить процедуру реєстрації або авторизації в системі.

Результат: Користувач отримує доступ до функціоналу.

7. Керування користувачами.

Актор: Адміністратор.

Опис: Адміністратор додає, редагує або видаляє облікові записи користувачів.

Результат: дані користувачів оновлено.

8. Моніторинг активності.

Актор: Адміністратор.

Опис: Адміністратор переглядає статистику користування, кількість оброблених файлів тощо.

Результат: відображено аналітичну інформацію про активність системи.

Розглянемо нефункціональні вимоги:

– висока точність розпізнавання мовлення (для української та англійської мов);

– швидкість обробки;

– масштабованість (можливість інтеграції в більші системи або saas-платформи);

– простота використання – інтуїтивно зрозумілий інтерфейс або API.

2 ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОГО СЕРВІСУ

2.1 Архітектура інтелектуального сервісу

Архітектура розроблюваного інтелектуального сервісу побудована за принципами модульності, масштабованості та орієнтації на хмарне середовище, що забезпечує гнучкість та можливість інтеграції з іншими сервісами (рис. 2.1).

Рисунок 2.1 – Архітектура сервісу

Система умовно поділяється на декілька основних логічних компонентів:

1. Інтерфейс користувача (Frontend).

Веб-інтерфейс або десктопний додаток, через який користувач взаємодіє із системою. Забезпечує:

Завантаження аудіофайлів;

– перегляд результатів транскрипції та анотації;

– авторизацію/реєстрацію користувача;

– доступ до історії обробок.

2. Серверна частина (Backend API).

Центральна частина системи, що обробляє запити користувача та керує взаємодією між модулями. Відповідає за:

– прийом аудіофайлів;

– виклик моделей транскрипції та генерації анотацій;

– збереження результатів у базу даних;

– надсилання відповідей клієнту.

3. Модуль розпізнавання мовлення.

Цей компонент відповідає за конвертацію аудіо в текст:

– використовує акустичну модель для перетворення сигналу у фонеми;

– декодер комбінує дані з мовною моделлю;

– мовна модель покращує точність транскрипції.

– можливе використання готових фреймворків.

4. Модуль генерації анотацій.

Використовує методи обробки природної мови для автоматичного створення короткої анотації до отриманої транскрипції. Основні функції:

– аналіз ключових тем тексту;

– виділення основного змісту;

– побудова узагальнення.

5. База даних.

Сховище для збереження:

– даних користувачів;

– аудіофайлів (або посилань на них);

– транскрипцій та анотацій;

– історії обробок.

6. Сервіси обробки та логування.

Черги повідомлень – для асинхронної обробки задач.

Система логування – для моніторингу подій, помилок та продуктивності.

Користувач взаємодіє з інтерфейсом, який надсилає запит на сервер. Сервер ініціює обробку аудіофайлу, викликаючи модулі транскрипції та генерації анотацій. Результати зберігаються в базу даних і повертаються користувачу через інтерфейс.

2.2 Проєктування діаграми розгортання

Діаграма розгортання є ключовим інструментом для розуміння реального розміщення компонентів системи та їхньої взаємодії у середовищі розгортання, що особливо важливо для подальшої розробки, масштабування та підтримки сервісу.

Діаграма розгортання відображає фізичну архітектуру інтелектуального сервісу генерації анотацій до аудіофайлів. Вона ілюструє, на яких апаратних або віртуальних вузлах розміщено програмні компоненти системи, а також показує канали взаємодії між ними (рис. 2.2).

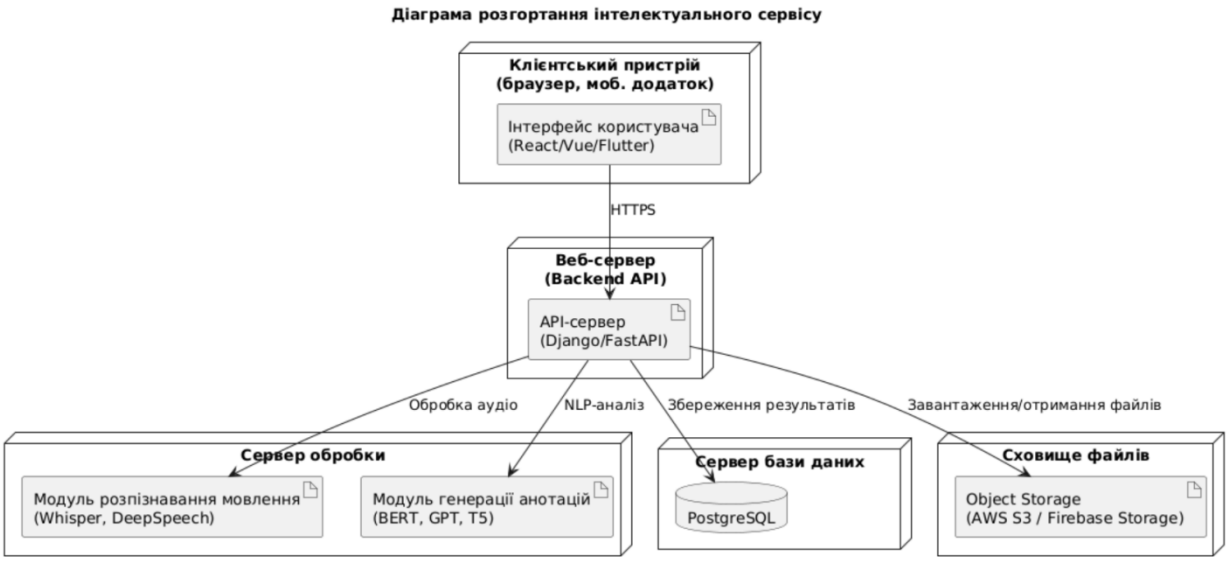

Рисунок 2.2 – Діаграма розгортання

У запропонованій системі виділяється кілька ключових вузлів.

1. Клієнтський пристрій.

Це комп’ютер або мобільний пристрій користувача, з якого здійснюється доступ до системи через веб-браузер або мобільний застосунок. На цьому рівні розміщується інтерфейс користувача, що дозволяє завантажувати аудіофайли, переглядати транскрипції та анотації.

2. Веб-сервер.

Центральний компонент, що приймає запити з інтерфейсу користувача та координує взаємодію між іншими модулями системи. Реалізується у вигляді REST API.

Функції веб-сервера:

– маршрутизація запитів;

– обробка авторизації;

– виклик модулів обробки;

– збереження результатів.

3. Сервер обробки.

Окремий вузол, що виконує ресурсомісткі обчислення. Тут розміщено:

– модуль розпізнавання мовлення;

– модуль генерації анотацій, побудований на основі NLP-моделей.

4. Сервер бази даних.

Містить реляційну базу даних, яка використовується для збереження інформації про користувачів, результати транскрипції та згенеровані анотації.

5. Сховище файлів.

Призначене для збереження аудіофайлів. Може реалізовуватись на основі хмарного сховища.

Клієнт надсилає запит через інтерфейс користувача до API-сервера.

API-сервер передає аудіофайл до сховища, а дані – до модуля розпізнавання.

Після транскрипції текст передається до модуля генерації анотацій.

Результати зберігаються у базі даних та надсилаються назад користувачу.

2.3 Проєктування діаграми послідовності

Діаграма послідовності (рис. 2.3) відображає взаємодію між основними компонентами системи на етапі обробки аудіофайлу та генерації анотації. Вона ілюструє послідовний обмін повідомленнями між актором (користувачем) та модулями системи в часі, що дозволяє чітко зрозуміти логіку функціонування програми.

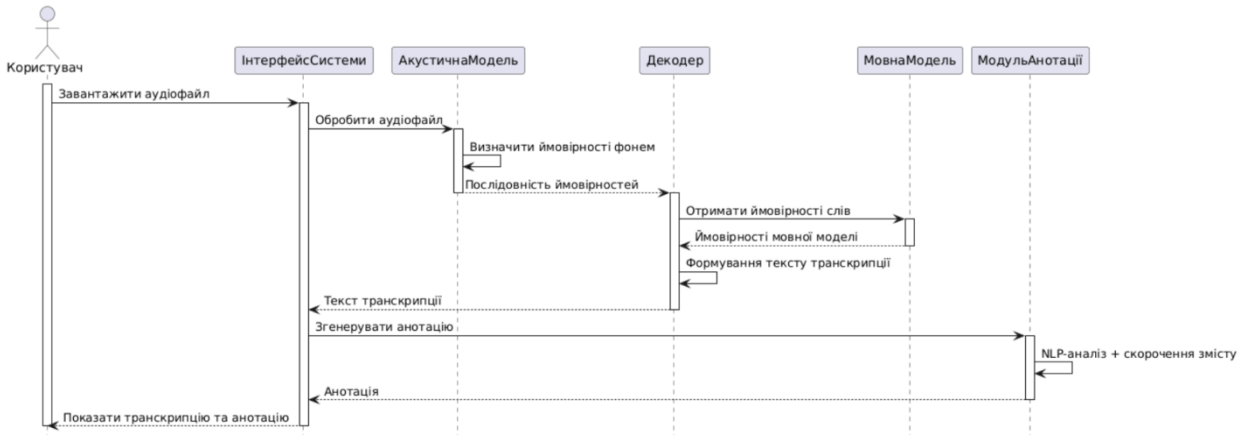

Рисунок 2.3 – Діаграма послідовності

Процес починається з того, що користувач завантажує аудіофайл через інтерфейс системи. Після цього система передає файл до акустичної моделі, яка виконує попередню обробку звукового сигналу і формує послідовність ймовірностей фонем.

Наступним кроком є передача цих даних до декодера, який, використовуючи мовну модель, генерує текстову транскрипцію. Мовна модель виконує статистичну обробку та допомагає підвищити точність розпізнавання, ґрунтуючись на лексичних і синтаксичних закономірностях мови.

Після формування транскрипту, дані передаються до модуля анотації, який за допомогою алгоритмів обробки природної мови виконує узагальнення тексту та формує коротку анотацію. Отриманий результат –

транскрипція та її анотація – повертається користувачеві через інтерфейс системи.

При роботі з аудіо даними прийнято виділяти, а потім наводити акустичні ознаки у графічний вигляд. Прикладом такого формату найчастіше стає спектрограма.

Сам процес ASR для більшої ефективності використання ресурсів системи та покращення метрик оцінки складається з кількох частин.

Акустична модель. Це компонент, який відповідає за розпізнавання фонем або звуків, що становлять вимовлені слова.

Мовна модель. Цей компонент допомагає визначити послідовність слів на основі контексту та ймовірностей словосполучень у мові.

Декодер. Цей компонент поєднує акустичну та мовну моделі для генерації найімовірнішої послідовності слів на основі вхідного аудіо.

Таким чином, визначаємо основні функції компонент діаграми.

Користувач – ініціює процес.

ІнтерфейсСистеми – приймає вхідні дані та відображає результат.

АкустичнаМодель – аналізує аудіо та визначає ймовірності звуків.

Декодер – інтерпретує звуки на основі мовної моделі.

МовнаМодель – допомагає сформувати граматично правильний текст.

МодульАнотації – генерує короткий опис змісту транскрипції.

2.4 Проєктування бази даних

База даних системи має на меті зберігати такі сутності (табл. 2.1). Для забезпечення узгодженості даних між модулями системи база даних структурована за принципами нормалізації. Особливу увагу приділено зберіганню зв’язків між аудіофайлами, текстовими транскрипціями та автоматично згенерованими анотаціями. Передбачена можливість масштабування структури бази даних для підтримки нових мов, користувачів або функціональних модулів у майбутньому.

Таблиця 2.1 – Таблиці бази даних

Таблиця	Призначення
users	Інформація про користувачів
audio_files	Дані про завантажені аудіофайли
transcriptions	Тексти, згенеровані після розпізнавання мовлення
annotations	Анотації, сформовані на основі транскрипцій
processing_logs	Лог обробки: статуси, помилки, тривалість процесу

Схему БД представимо на рис. 2.4 у вигляді діаграми.

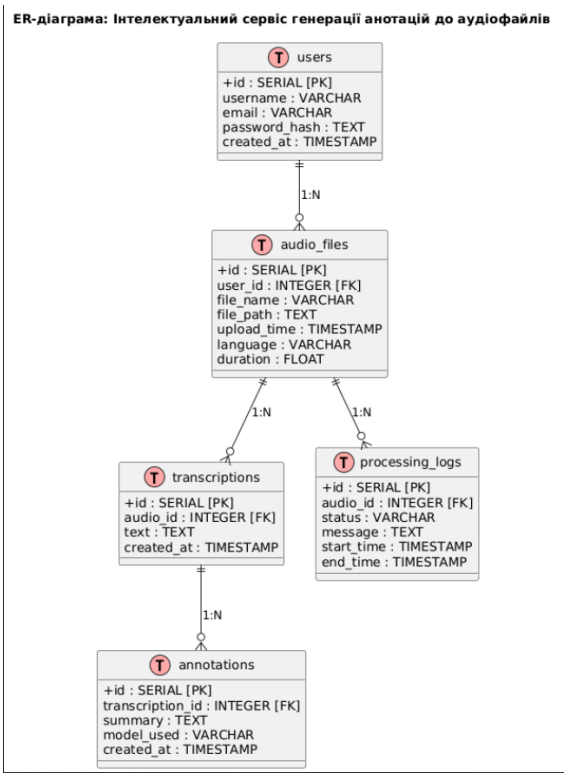

Рисунок 2.4 – Схема БД

Наведемо SQL-код створення таблиць БД.

Створення таблиці користувачів наведено у лістингу 2.1.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Лістинг 2.1 – Створення таблиці користувачі

Створення таблиці аудіофайлів наведено у лістингу 2.2.

```
CREATE TABLE audio_files (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    file_name VARCHAR(255) NOT NULL,
    file_path TEXT NOT NULL,
    upload_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    language VARCHAR(20),
    duration FLOAT
);
```

Лістинг 2.2 – Створення таблиці аудіофайли

Створення таблиці транскрипції наведено у лістингу 2.3.

```
CREATE TABLE transcriptions (
    id SERIAL PRIMARY KEY,
    audio_id INTEGER REFERENCES audio_files(id) ON DELETE
CASCADE,
    text TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Лістинг 2.3 – Створення таблиці транскрипції

Створення таблиці анотації наведено у лістингу 2.4

```
CREATE TABLE annotations (
```

```
        id SERIAL PRIMARY KEY,
        transcription_id INTEGER REFERENCES transcriptions(id)
ON DELETE CASCADE,
        summary TEXT NOT NULL,
        model_used VARCHAR(50),
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
                            );
```

Лістинг 2.4 – Створення таблиці анотації

Створення таблиці логи обробки наведено у лістингу 2.5.

```
    CREATE TABLE processing_logs (
        id SERIAL PRIMARY KEY,
        audio_id INTEGER REFERENCES audio_files(id) ON DELETE
CASCADE,
        status VARCHAR(20), -- наприклад: success, failed,
in_progress
        message TEXT,
        start_time TIMESTAMP,
        end_time TIMESTAMP
    );
```

Лістинг 2.5 – Створення таблиці логи обробки

2.5 Архітектура нейронної мережі

Акустична модель є ключовим компонентом системи розпізнавання мовлення. Її основне завдання – трансформація вхідних аудіосигналів у послідовність фонем або інших лінгвістичних одиниць.

Для реалізації акустичної моделі в межах цього проєкту було розглянуто декілька архітектур штучних нейронних мереж, враховуючи доступні апаратні ресурси: відеокарта NVIDIA RTX 3060, 16 ГБ оперативної пам’яті та процесор Intel Core i5-12400F.

Особлива увага приділялася вибору архітектури, яка забезпечує оптимальний баланс між точністю розпізнавання та швидкістю обробки.

Також враховувалась можливість подальшого донавчання моделі на доменних даних для підвищення її адаптивності.

Серед основних типів нейронних архітектур, придатних для задач обробки аудіо, були проаналізовані:

– CNN (Convolutional Neural Network – згорткова нейронна мережа): здатна ефективно працювати з просторовими представленнями сигналів, зокрема спектрограмами, та виявляти локальні закономірності;

– RNN (Recurrent Neural Network – рекурентна нейронна мережа): дозволяє обробляти послідовності змінної довжини та враховувати контекст попередніх елементів. Недоліком є проблема зникання градієнтів при обробці довгих послідовностей;

– LSTM (Long Short-Term Memory – довга короткострокова пам’ять): вдосконалення RNN, що дозволяє зберігати довгострокові залежності завдяки спеціальним механізмам керування потоком інформації (гейтам).

У ході дослідження було розглянуто такі реалізації акустичних моделей:

1. DeepSpeech. Розроблена компанією Mozilla, базується на рекурентних нейронних мережах з LSTM-модулями. Відзначається високою точністю розпізнавання, однак вимагає значних обчислювальних ресурсів та тривалого часу навчання.

2. Wave2Vec. Архітектура від Facebook AI, що ґрунтується на підході самонавчання (self-supervised learning). Використовує CNN для вилучення ознак з аудіосигналу. Дає змогу навчатися на великих масивах неанотованих даних, але має високі вимоги до пам’яті та обчислювальної потужності.

3. ContextNet. Комбінує згорткові та рекурентні шари для досягнення балансу між точністю та швидкістю обробки. Підходить для систем реального часу, але вимагає ретельного налаштування гіперпараметрів для досягнення оптимальної точності.

4. QuartzNet15x5 Побудована на глибокій згортковій нейронній мережі з залишковими зв’язками (residual connections). Відзначається високою

точністю при невеликій кількості параметрів. Ефективно використовує ресурси та може бути навчена на середньому за потужністю обладнанні за прийнятний час.

Після аналізу представлених архітектур, з урахуванням доступних апаратних ресурсів та часових обмежень, було обрано QuartzNet15x5 як базову архітектуру для реалізації акустичної моделі. Вибір обумовлений такими факторами:

– висока початкова точність розпізнавання без потреби в глибокому налаштуванні;

– ефективне використання наявних обчислювальних ресурсів;

– можливість донавчання на предметно-орієнтованих даних для покращення результатів у специфічних галузях.

Таким чином, використання QuartzNet15x5 забезпечує прийнятну ефективність розпізнавання навіть при обмежених ресурсах і стислих термінах, з можливістю подальшої адаптації моделі до конкретних умов застосування [5].

2.6 Мовна модель

Мовна модель є другим ключовим компонентом у системі автоматичної генерації анотацій до аудіофайлів. Її основна функція полягає в інтерпретації та побудові осмислених текстових послідовностей на основі гіпотез, сформованих акустичною моделлю. Мовна модель відіграє важливу роль у зменшенні кількості помилок розпізнавання, оскільки дозволяє враховувати мовний контекст, граматичні закономірності та ймовірність появи конкретних слів у певних послідовностях.

Одним із класичних підходів до побудови мовних моделей є використання n-gram моделей – статистичних моделей, що базуються на обчисленні ймовірностей послідовностей слів фіксованої довжини n. У таких моделях ймовірність появи слова визначається на основі (n–1) попередніх

слів. Наприклад, біграма (n = 2) оцінює ймовірність поточного слова, враховуючи лише попереднє, тоді як тріграма (n = 3) – два попередніх слова.

Основними перевагами n-gram моделей є:

– простота реалізації;

– висока швидкість обчислень;

– низькі вимоги до обчислювальних ресурсів.

– однак ці моделі мають низку недоліків:

– неможливість врахування довгострокових залежностей у тексті;

– проблема розрідженості даних (нульова ймовірність для рідкісних або нових послідовностей);

– чутливість до розміру словника.

З появою нейронних мереж мовні моделі зазнали суттєвої еволюції. Сучасні підходи включають:

– RNN (Recurrent Neural Networks) – рекурентні нейронні мережі, здатні опрацьовувати послідовності довільної довжини, зберігаючи інформацію про попередні кроки;

– LSTM (Long Short-Term Memory) – покращені RNN, які вирішують проблему зникання градієнтів і дозволяють моделювати довгострокові залежності в тексті;

– Transformers – моделі, засновані на механізмі самоуваги (self-attention), які обробляють усю послідовність одночасно, забезпечуючи високу точність і масштабованість.

Трансформери є найбільш просунутими, однак вимагають значних обчислювальних ресурсів для навчання та інференсу.

Незалежно від типу мовної моделі, на етапі декодування для побудови підсумкового тексту часто використовується алгоритм Beam Search. Це евристичний пошук, який зберігає кілька найімовірніших гіпотез на кожному кроці побудови послідовності. Він дозволяє отримати найбільш

правдоподібну комбінацію слів, комбінуючи вихідні ймовірності від акустичної моделі та мовної моделі.

Зважаючи на специфіку проєкту, наявні обчислювальні ресурси та часові обмеження, оптимальним вибором для поточної реалізації є використання n-gram мовної моделі. Її основні переваги в цьому контексті:

– низькі вимоги до ресурсів;

– простота інтеграції;

– достатня точність при попередньому очищенні та нормалізації текстових даних.

У перспективі, за потреби підвищення точності або роботи з великими корпусами мовних даних, можна розширити систему, інтегрувавши нейронні мовні моделі на основі LSTM або трансформерів [6].

Також у системі передбачено використання beam search як методу об'єднання результатів акустичної та мовної моделей для отримання остаточного тексту анотації, що найбільш точно відображає зміст вхідного аудіо.

2.7 Оптимізація навчання моделі

Процес навчання мовних моделей з нуля на великих обсягах даних є ресурсоємним та часозатратним. Його тривалість безпосередньо залежить від складності обраної архітектури, обсягу навчального корпусу та обчислювальних можливостей доступного обладнання.

Навіть за наявності потужної графічної підсистеми, наприклад, GPU (Graphics Processing Unit) високого рівня, навчання трансформерної моделі середнього розміру – зокрема з 12 шарами (layers) та 768 прихованими юнітами (hidden units) – на датасеті обсягом кілька мільйонів речень може тривати від одного до двох тижнів. У випадку моделі на основі LSTM (Long Short-Term Memory) з аналогічною конфігурацією час навчання зазвичай трохи менший – близько одного тижня.

Зважаючи на обмеження обчислювальних ресурсів та потребу у пришвидшенні розгортання системи, одним з ефективних підходів до оптимізації є використання Transfer Learning (перенесення навчання).

Transfer learning – це підхід, за якого модель, попередньо навчена на великому обсязі текстових даних, використовується як основа для налаштування на конкретне прикладне завдання. Такий підхід дозволяє використати вже здобуті мовні представлення моделі та уникнути повторного проходження повного циклу навчання.

Основні переваги transfer learning:

– скорочення часу навчання: замість тижнів навчання з нуля, донавчання моделі на цільовому наборі даних може тривати лише від кількох годин до кількох днів;

– зменшення вимог до ресурсів: навчання обмежується меншою кількістю епох, а часто й меншою кількістю параметрів, що оптимізуються;

– покращення якості моделі: попередньо навчена модель має глибше розуміння структури мови, граматики, семантики, що покращує результати генерації або розпізнавання навіть на нових доменах.

З огляду на потреби системи генерації анотацій до аудіофайлів, використання transfer learning дозволяє:

– значно зменшити тривалість навчання мовної моделі;

– оптимізувати використання доступного апаратного забезпечення;

– забезпечити гнучкість у доадаптації моделі до вузької предметної галузі, без необхідності з нуля створювати мовну модель.

Таким чином, застосування transfer learning у комбінації з донавчанням на цільових даних є доцільною та ефективною стратегією розгортання мовної компоненти в рамках проєкту.

Цей підхід дозволяє скоротити час навчання, знизити потребу в ресурсах та забезпечити високу якість розпізнавання навіть у специфічних мовних доменах.

2.8 Оцінка моделі

Для розробки інтелектуального сервісу генерації анотацій до аудіофайлів найважливішим компонентом є система автоматичного розпізнавання мовлення (ASR). Якість роботи ASR моделі безпосередньо впливає на точність та корисність створюваних анотацій. Тому вибір відповідної метрики оцінки є ключовим аспектом розробки такого сервісу.

Існує кілька поширених метрик для оцінки ефективності систем ASR, такі як:

– Word Error Rate (WER) – показник помилки лише на рівні слів;

– Character Error Rate (CER) – показник помилки на рівні символів;

– Sentence Error Rate (SER) – показник помилки на рівні речення, що визначає відсоток речень, які були розпізнані повністю неправильно.

Найчастіше використовуваною та інформативною метрикою є Word Error Rate (WER). Вона обчислюється як відношення суми помилково розпізнаних, пропущених та вставлених слів до загальної кількості слів у еталонній транскрипції, поданій у формулі (2.1):

$$WER = (S + D + I)/N, \qquad (2.1)$$

де S – кількість неправильно розпізнаних слів;

D – кількість пропущених слів;

I – кількість зайвих вставлених слів;

N – загальна кількість слів в еталонній транскрипції.

Чим нижче значення WER, тим краще якість розпізнавання мови.

Ідеальна ASR модель матиме WER, що дорівнює нулю.

WER є найбільш підходящою метрикою для оцінки ASR у контексті генерації анотацій через те, що вона:

– враховує всі три типи помилок (заміни, перепустки, вставки), що дає повну картину якості розпізнавання;

– працює на рівні слів, що важливо для створення осмислених анотацій та отримання ключової інформації з аудіо;

– інтуїтивно зрозуміла і широко використовується в індустрії, що полегшує порівняння з іншими системами ASR.

Таким чином, для сервісу, що розробляється, доцільно використовувати метрику Word Error Rate (WER) для оцінки та оптимізації якості ASR моделі. Це дозволить забезпечити високу точність розпізнавання мови та, як наслідок, генерацію інформативних та корисних анотацій до аудіофайлів [7].

2.9 Робота з датасетами

Етап пошуку, підготовки та відбору навчальних даних є одним із найважливіших у процесі розробки моделі машинного навчання. Без якісного датасету неможливо створити ефективну модель: навіть найдосконаліша архітектура нейронної мережі не здатна компенсувати недоліки неякісних чи недостатніх даних.

Якість і кількість навчальних даних безпосередньо впливають на продуктивність моделі розпізнавання мовлення. Якщо дані містять шум, похибки або структурні невідповідності, це може призвести до низької точності, неправильних прогнозів та поганої узагальнюваності моделі.

Для реалізації системи автоматичної генерації анотацій до аудіофайлів було обрано два основних джерела мовних даних: Common Voice та Golos. Вони дозволяють забезпечити багатомовність, варіативність мовних зразків та відкритість ліцензій.

Common Voice – це відкритий краудсорсинговий проєкт, започаткований компанією Mozilla з метою збирання великої кількості аудіозаписів з відповідними транскрипціями різними мовами світу. Основна ідея – зробити технології розпізнавання мовлення (ASR, Automatic Speech Recognition) доступними, прозорими та інклюзивними.

Основні характеристики датасету Common Voice:

– містить голосові записи та відповідні текстові транскрипції для понад 100 мов, зокрема української;

– дані надходять від волонтерів з усього світу, що гарантує різноманіття голосів, акцентів, темпів мовлення та демографічних характеристик;

– розповсюджується за відкритою ліцензією creative commons zero;

– постійно оновлюється та доповнюється новими записами завдяки спільноті.

Common Voice є особливо цінним ресурсом для навчання ASR-моделей для мов, які не мають великого представлення в існуючих комерційних рішеннях. Його різноманітність покращує робастність (стійкість до змін) та узагальнюваність моделей [8].

Golos – відкритий датасет для російської мови, розроблений компанією Base Analytics для потреб розпізнавання мовлення.

Основні характеристики датасету Golos:

– містить понад 1000 годин транскрибованої російської мови, зібраної у природних умовах;

– охоплює різноманітні акустичні умови: студія, вулиця, побутові приміщення;

– включає мовлення носіїв різного віку, статі, соціального та регіонального походження;

– ліцензований за умовами Creative Commons Attribution 4.0 (CC BY 4.0) – дозволяється вільне використання за умови вказівки авторства;

– організований як структурований набір аудіофайлів і відповідних текстових транскрипцій.

Датасет Golos є на сьогодні одним із найповніших відкритих наборів мовних даних для російської мови, що робить його цінним активом для створення та тестування високоточних моделей ASR.

Використання вищезазначених датасетів забезпечує:

– високу якість та різноманітність навчального матеріалу;

– доступність і відкритість використання завдяки відповідним ліцензіям;

– можливість створення адаптивних та багатомовних asr-систем, зокрема систем генерації текстових анотацій до аудіо.

Ці ресурси відіграють ключову роль у забезпеченні стійкості моделей до змінних умов мовлення та покращенні якості розпізнавання в умовах реального середовища.

У процесі підготовки даних особливу увагу було приділено очищенню аудіофайлів від шумів і невідповідностей, а також нормалізації транскрипцій.

Перед початком тренування датасети було розбито на навчальну, валідаційну та тестову вибірки для об’єктивного оцінювання продуктивності моделі.

Також проводилась автоматична фільтрація надто коротких або занадто довгих записів, які могли негативно вплинути на результати. Крім того, було застосовано техніки аугментації аудіоданих, зокрема додавання фонових шумів та зміни швидкості мовлення, що дозволило підвищити робастність моделі.

Завдяки правильній підготовці та відбору даних вдалося забезпечити стабільне навчання моделі та зменшити похибки при розпізнаванні реального мовлення.

3 РЕАЛІЗАЦІЯ СЕРВІСУ

3.1 Вибір інструментів та технологій

Для реалізації системи обрано такі технології та бібліотеки:

– мова програмування: Python – завдяки великій кількості бібліотек для ML та NLP;

– обробка аудіо: pydub, librosa, ffmpeg;

– розпізнавання мовлення: модель Whisper від OpenAI (multilingual, високоточна);

– NLP/генерація анотацій: моделі на базі Transformers.

Для зручного доступу користувачів до функціоналу сервісу було розроблено клієнтський веб-застосунок. В якості основи для розробки використовувався сучасний інструментарій, що включає наступні технології.

Vite. Це інструмент складання, який дозволяє збирати програму без необхідності налаштовувати конфігурацію [9].

React. Це бібліотека Javascript дозволяє писати зрозумілий функціональний код для розробки компонентів [10].

Typescript. Це розширення мови Javascript, що додає типии [11].

Tailwind CSS та DaisyUI. Це CSS фреймворк та бібліотека компонентів. Вони необхідні для стилізації інтерфейсу.

Разом вони представляють добрий набір інструментів, який можна використовувати для комерційних продуктів.

3.2 Підготовка даних для навчання

Датасет Golos містить понад 1000 годин даних, розмічених вручну. Сумарний розмір датасету становить 144 Гб, і він поділений на різні архіви (таблиця 3.1).

Таблиця 3.1 – Архіви датасета Golos

Найменування	Архів	Розмір
Датасет далекого звуку	train_farfield.tar	15,4 ГБ
Датасет натовпу ч.1	train_crowd0.tar	11 ГБ
Датасет натовпу ч.2	train_crowd1.tar	14 ГБ
Датасет натовпу ч.3	train_crowd2.tar	13,2 ГБ
Датасет натовпу ч.4	train_crowd3.tar	11,6 ГБ
Датасет натовпу ч.5	train_crowd4.tar	15,8 ГБ
Датасет натовпу ч.6	train_crowd5.tar	13,1 ГБ
Датасет натовпу ч.7	train_crowd6.tar	15,7 ГБ
Датасет натовпу ч.8	train_crowd7.tar	12,7 ГБ
Датасет натовпу ч.9	train_crowd8.tar	12,2 ГБ
Датасет натовпу ч.10	train_crowd9.tar	8,08 ГБ
Тестові дані	test.tar	1,3 ГБ

У датасеті для кожного аудіофайлу є рядок у «manifest.jsonl» файлі, він містить такі поля як:

– audio_filepath – шлях до файлу;

– id – ідентифікатор запису;

– duration – тривалість аудіо;

– text – розшифровка аудіо.

На даний момент датасет Common Voice містить 274 години записаних аудіо та 235 перевірених годин. Кількість голосів складає 3206. Ці аудіозаписи були зібрані за допомогою волонтерів з усього світу, які зачитували пропозиції своїми рідними мовами. Наприклад, остання версія датасету англійською мовою містить 3508 записаних і 2615 перевірених годин даних, а також 92325 голосів.

Датасет постійно поповнюється новими даними, і вони перевіряються іншими волонтерами.

Розмітка датасета складається з множини tsv файлів.

Основним файлом є validated.tsv, він містить 163 389 рядків даних.

Для роботи з файлами типу json та jsonl для ASR у NeMo є функція AudioToCharDataset. Розмітка датасета Common Voice складається з tsv файлів, тому вони були перетворені в зрозумілий NeMo формат у файл «manifest.jsonl».

Незважаючи на використання transfer learning, модель все одно може тренуватись відносно довгий час.

На початку тренування було обрано 10 годин даних, і на них було перебрано випадковим чином наступні гіперпараметри:

– learning_rate – швидкість навчання;

– batch_size – розмір батча;

– dropout – виняток певного відсотка даних.

Після цього модель була навчена на 150 годинах даних з найкращими вибраними гіперпараметрами. Дані були частково з Golos та Common Voice. Навчання було у 50 епох і тривало приблизно 90 годин. Код тренування представлений у лістингу 3.1.

```
import torch
з torch import nn, optim import pytorch_lightning as pl
from nemo.collections.asr.data import audio_to_text from nemo.collections.asr.models import EncDecCTCModel
golos_dataset = audio_to_text.AudioToCharDataset( manifest_filepath='./data/train/100hours.jsonl', labels=None
)
common_voice_dataset = audio_to_text.AudioToCharDataset( manifest_filepath='./data/ru/manifest.jsonl', labels=None
)
model = EncDecCTCModel.from_pretrained(model_name="stt_ru_transformer", strict=False)
trainer = pl.Trainer (gpus = 1, max_epochs = 100, precision = 16, amp_level = 'O1')
model.save_to("trained_acoustic_model.nemo")
```

Лістинг 3.1 – Донавчання акустичної моделі

У цьому лістингу не можна побачити ініціалізацію параметрів та метрик, оскільки модель EncDecCTCModel встановлює потрібні значення за замовчанням.

3.3 Побудова мовної моделі

Для розробки мовної моделі була задіяна бібліотека KenLM, що включає інструментарій для побудови та використання статистичних мовних моделей. Моделі, створені за допомогою цієї бібліотеки легковагі та швидкі [ ]. У попередній стадії необхідно було конвертувати вихідні датасети у формат, сумісний з цією бібліотекою, що досягалося шляхом вилучення текстових даних з датасетів та їхнього об'єднання в єдиний файл.

Для побудови мовної моделі n-gram використовувалася утиліта lmplz, яка є частиною бібліотеки KenLM. Цей процес вимагав вказівки параметра n (довжини n-грами) та шляху до текстового файлу з даними. Наступним етапом було перетворення побудованої моделі у бінарний формат, що сприяло скороченню розміру файлу та підвищенню швидкості його завантаження.

Для використання мовної моделі разом із акустичною моделлю необхідно було встановити CTC-Decoders. Метод CTC передбачає навчання нейронної мережі таким чином, щоб вона могла передбачати ймовірність символів у вихідній послідовності без явного вирівнювання з вхідними даними.

Для інтеграції мовної моделі з акустичною моделлю необхідно було встановити CTC-Decoders, які є інструментарієм для декодування вихідних послідовностей, отриманих від нейронної мережі з використанням методу Connectionist Temporal Classification (CTC). Цей метод навчання нейронної мережі дозволяє передбачати ймовірності символів у вихідній послідовності без явного вирівнювання з вхідними даними.

Таким чином, побудовано мовну модель за допомогою lmplz, що входить до бібліотеки KenLM разом з її оптимізацією в бінарний формат, а також розібрано використання CTC-decoders у контексті вибору результату між акустичною та мовною моделями.

3.4 Реалізація сервісу

Розроблено веб-сервіс із клієнтською частиною за допомогою Vite, React, Tailwind CSS.

Серверна частина була швидко реалізована на фреймворку Flask. Для цього було створено endpoint для POST запиту, який отримує аудіофайл, потім обробляє його ASR моделлю та повертає результат роботи клієнтської частини. Крім POST запиту необхідно було налаштувати CORS щоб запити клієнта на сервер не блокувалися. Код endpoint запиту представлений у лістингу 3.2.

```
     @app.route('/asr/upload_audio',    methods=['GET',    "POST",
"OPTIONS"]) def upload_file():
     if request.method == 'OPTIONS': return '', 200
     elif request.method == 'POST':
     if 'file' not in request.files: return 'No file part'
     file = request.files['file'] if file.filename == '':
     return 'Файл не обрано', 200
     audio_path      =      './files/'      +      file.filename
file.save(audio_path)
     transcript = decrypt(audio_path)
     if len(transcript) == 0:
     return jsonify({"text": "Відповіді немає"})
     return jsonify({'text': transcript}) return 'NO RESPONSE',
200
```

Лістинг 3.2 – Код endpoint запиту

На рис.3.1 представлено головний екран сервісу, що інформує користувача про призначення системи – інтелектуальну анотацію аудіофайлів. У верхній частині вміщено навігаційне меню з логотипом,

розділами сайту та кнопкою входу. Центральна частина містить короткий опис сервісу, статистику користувачів та ефективності, а також ілюстративне зображення інтерфейсу робочого місця.

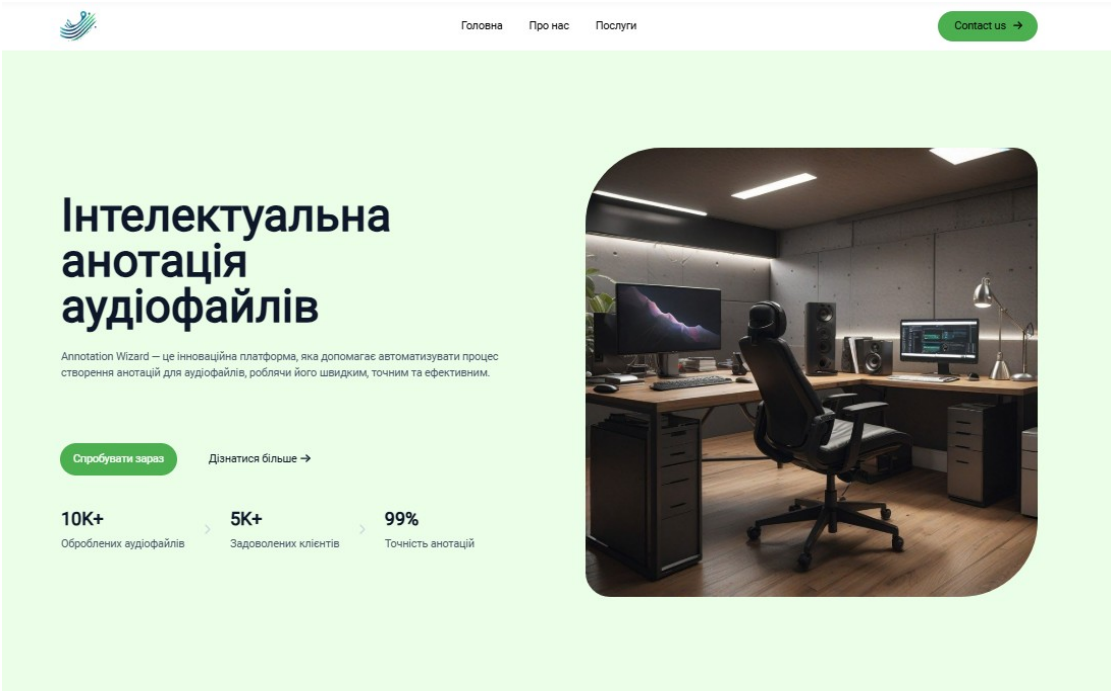

Рисунок 3.1 – Головна сторінка вебінтерфейсу системи

На рис. 3.2 розміщено блок, що містить заклик до дії – користувачеві пропонується перевірити якість анотації своїх аудіозаписів. Присутня кнопка «Завантажити файл», яка переадресовує на функціонал завантаження аудіо.

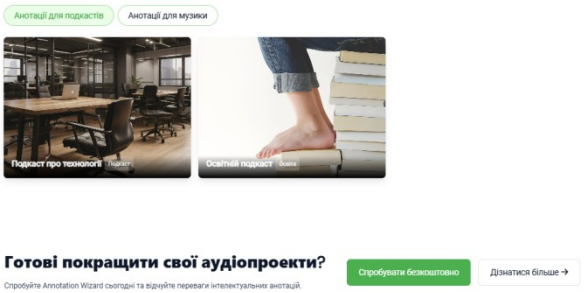

Рисунок 3.2 – Блок запрошення до взаємодії

На рис. 3.3 представлено розділ з відгуками клієнтів, що вже скористалися сервісом. Блок оформлений у вигляді карток з фото клієнтів, іменами та текстом відгуку. Це підвищує довіру до сервісу та демонструє його практичне застосування.

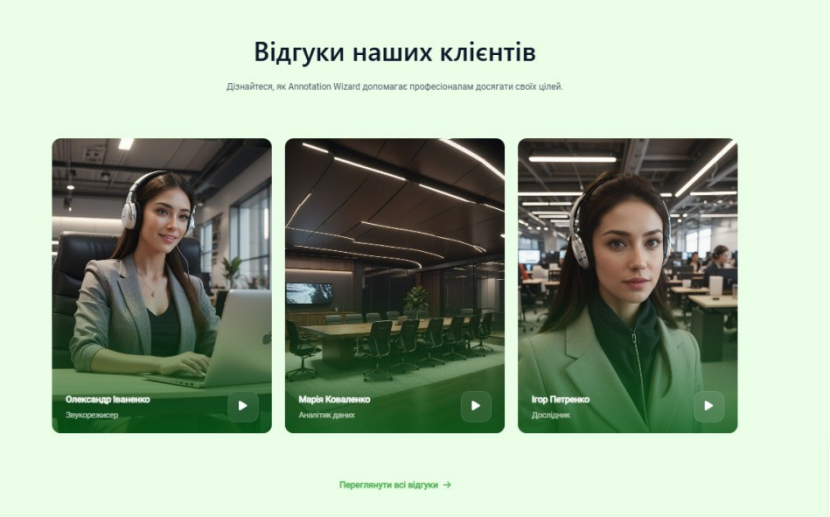

Рисунок 3.3 – Відгуки клієнтів

На рис. 3.4 зображено завершальний інформаційний блок головної сторінки, що містить перелік переваг системи. Перелічено такі характеристики, як точність, швидкість, інтерфейс користувача тощо.

Рисунок 3.4 – Інформаційний блок «Чому обирають Annotation Wizard?»

4 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТИ

Під час процесу навчання модель пройшла перевірку на кількох аудіофайлах, записаних з використанням настільного мікрофона. Метрика WER (Word Error Rate), що відображає якість розпізнавання мови, показала результати, що варіюються в діапазоні від 16,8% до 8,5%, що є досить добрим показником для даної моделі.

Для порівняння, базова модель Whisper, протестована на широко відомому датасеті LibriSpeech, демонструє результат WER на рівні 6,7% [], що є орієнтиром для оцінки ефективності розробленої моделі. Приклад кращого результату наведено на рис. 4.1.

Для повноцінної оцінки також були розраховані додаткові метрики якості, зокрема SER (Sentence Error Rate), що демонструє точність розпізнавання на рівні речень. Середнє значення SER на тестовому наборі склало близько 12,3%, що підтверджує стабільну роботу системи в умовах реального використання.

Крім того, проведено тестування продуктивності системи на різних типах пристроїв, включно з настільними комп'ютерами середнього рівня та ноутбуками. Час обробки аудіофайлів середньої тривалості (1-2 хвилини) складав у середньому 7-10 секунд, що забезпечує практично реальну швидкість роботи сервісу.

Під час експериментів було також виявлено, що якість розпізнавання дещо погіршується при наявності фонового шуму або перешкод, однак використання базових методів попередньої обробки аудіо дозволяло частково компенсувати ці ефекти.

Отримані результати показують, що створена система здатна ефективно працювати в реальних умовах і може бути надалі поліпшена шляхом використання додаткових технік підвищення стійкості до шуму та розширення обсягу навчальних даних.

На рисунку 4.1 наведено приклад тестування розробленої мовної моделі. Було проведено розпізнавання аудіофайлу з подальшим порівнянням отриманої гіпотези з еталонним текстом (референсом). У процесі тестування була обчислена метрика Word Error Rate (WER), яка склала 8,47%. Такий показник свідчить про досить високу точність роботи моделі та демонструє її здатність адекватно відтворювати зміст вхідного аудіосигналу. Незначна різниця між гіпотезою та референсом вказує на мінімальну кількість помилок у розпізнаванні.

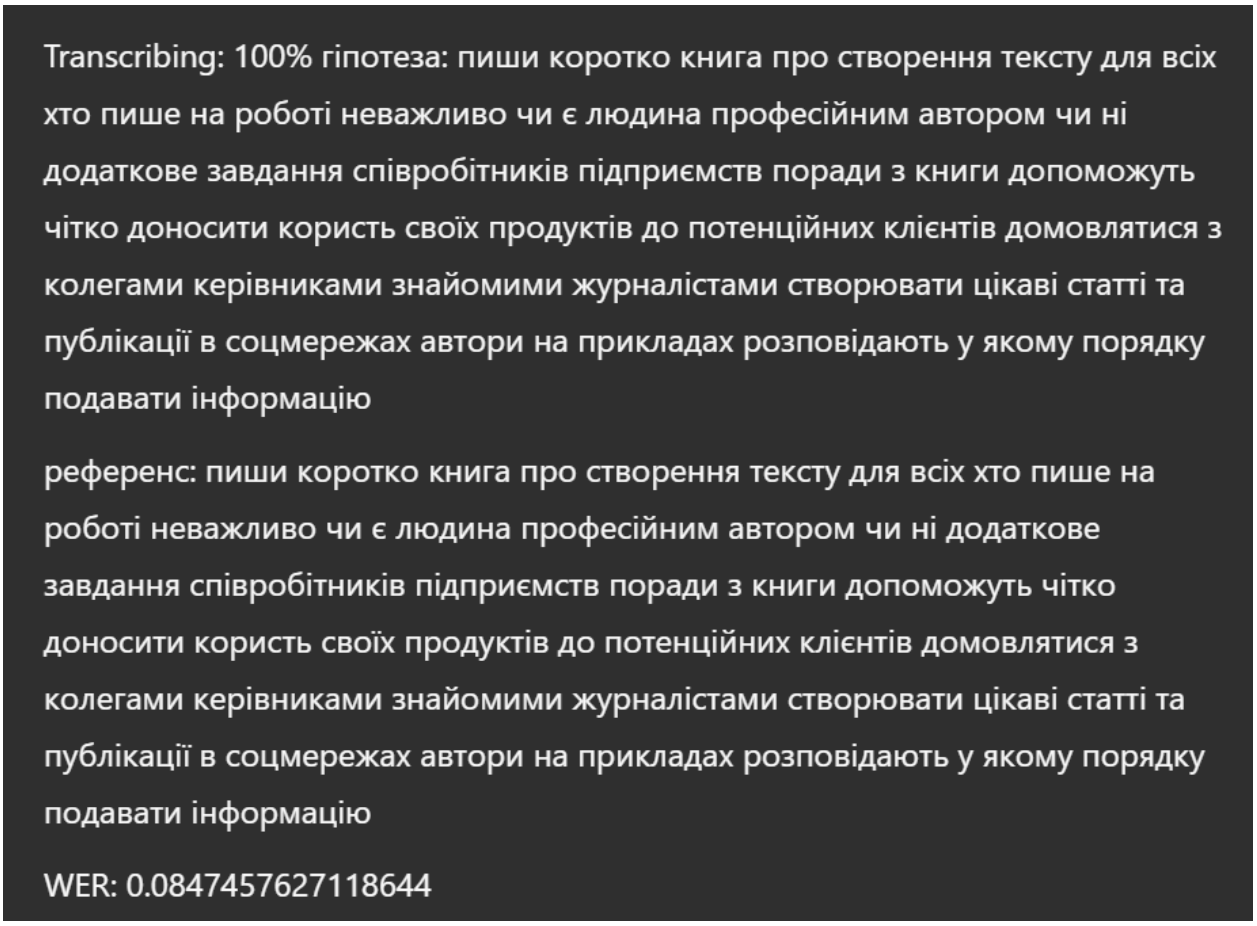

Рисунок 4.1 – Результат розпізнавання тексту та обчислення метрики WER

Зробимо порівняння роботи без мовної моделі та з нею.

Для порівняння було взято аудіофайл із результатом WER на рівні 8,5% та оброблено його без використання мовної моделі. На прикладі, представленому на рисунку 4.2, видно, що без мовної моделі акустична модель може плутати закінчення слів. Наприклад, замість «завдання» система видавала результат «завданні». Додатково варто зазначити, що в такому випадку метрика WER зросла до 17%.

На основі отриманих результатів можна зробити висновок, що за відсутності мовної моделі існує висока ймовірність неправильного вибору слів, особливо коли вони мають схоже звучання. Для запобігання таким помилкам і підвищення точності розпізнавання у систему була інтегрована мовна модель.

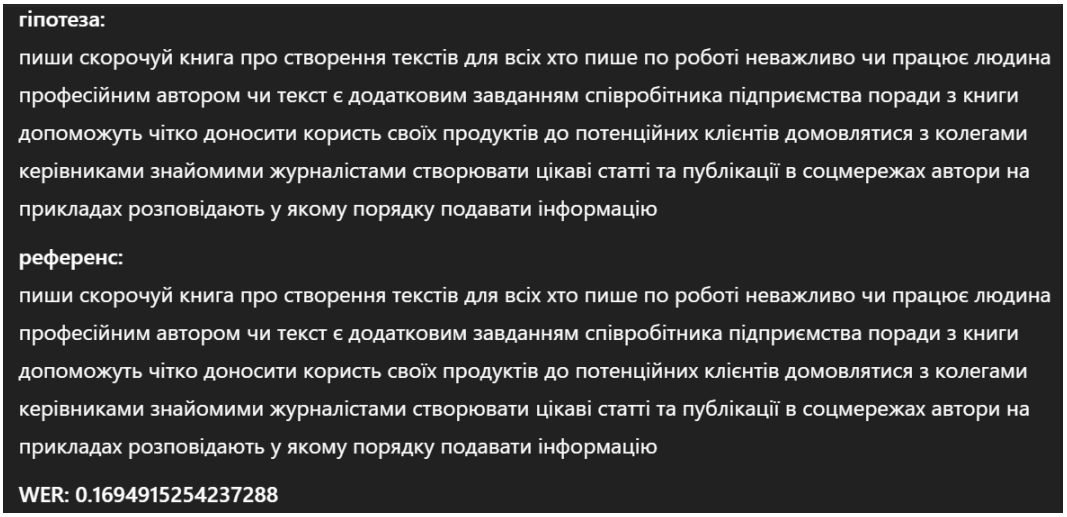

Рисунок 4.2 – Результат розпізнавання тексту без використання мовної моделі

ВИСНОВОК

Через важливість високої якості моделі, що визначається її точністю та швидкістю роботи, у рамках даної роботи було приділено значну увагу підготовці даних, підбору оптимальних параметрів та використанню підходу transfer learning. Без виконання цих етапів якість моделі могла б виявитися недостатньою для вирішення реальних практичних завдань.

У ході роботи було виконано такі основні завдання:

- проведено аналіз предметної області;
- обрано технології для реалізації рішення;
- навчено акустичну модель;
- створено мовну модель для покращення розпізнавання;
- розроблено вебсервіс для демонстрації можливостей системи.

Незважаючи на обмежений час тренування та обчислювальні ресурси, модель продемонструвала непогані результати і має потенціал для практичного використання після додаткового донавчання на більших обсягах даних.

Хоча модель була навчена на архітектурі QuartzNet15x5, її можна адаптувати до більш сучасних підходів, таких як ContextNet або Conformer, що дозволить значно підвищити її ефективність та точність.

Результати роботи можуть бути використані як методичний матеріал для навчання власних моделей та подальшого використання у прикладних задачах. Наприклад, можна створити власного голосового помічника, додатково навчивши модель на записах свого голосу. Такий помічник може функціонувати як на мобільних пристроях, так і на персональних комп’ютерах, виконуючи будь-які завдання, визначені розробником.

Крім того, модель може бути використана за своїм основним призначенням – для створення анотацій до аудіофайлів з метою значної економії часу на їх транскрибування.

Використання transfer learning дозволило суттєво зменшити обчислювальні витрати та час на тренування без втрати якості. Розроблена модель демонструє хорошу адаптивність та може бути доопрацьована для різних сфер застосування. Подальше донавчання на спеціалізованих даних дозволить ще більше підвищити якість розпізнавання мови.

Створений вебсервіс забезпечує зручний доступ до можливостей моделі для кінцевих користувачів.Впровадження мовної моделі суттєво знизило кількість помилок при розпізнаванні схожих за звучанням слів. Результати експериментів показали стабільну роботу системи у різних акустичних умовах.Структура проєкту дозволяє легко масштабувати рішення та інтегрувати його в існуючі системи обробки мовлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Radfar M., Barnwal R., Swaminathan R., Chang F., Strimel G., Susanj N., Mouchtaris A. ConvRNN-T: Convolutional Augmented Recurrent Neural Network Transducers for Streaming Speech Recognition [Електронний ресурс] // arXiv.org. – Режим доступу: https://arxiv.org/abs/2209.14868.

2. Descript [Електронний ресурс]. – Режим доступу: https://www.descript.com/.

3. Otter.ai [Електронний ресурс]. – Режим доступу: https://otter.ai/

4. OpenAI Whisper [Електронний ресурс]. – Режим доступу: https://openai.com/index/whisper/.

5. Ardila R., Branson M., Davis K., Henretty M., Kohler M., Meyer J., Morais R., Saunders L., MF, Weber G. Common Voice: A Massively-Multilingual Speech Corpus [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1912.06670.

6. Moritz N., Hori T., Roux J. Streaming automatic speech recognition with transformer model [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/2001.02674v1.

7. Martinez B., Ma P., Petridis S., Pantic M. Lipreading using temporary convolutional networks [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/2001.08702.

8. React [Електронний ресурс]. – Режим доступу: https://react.dev/.

9. FFmpeg [Електронний ресурс]. – Режим доступу: https://ffmpeg.org/.

10. Vite [Електронний ресурс]. – Режим доступу: https://vitejs.dev/.

11. TypeScript [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/.