Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Тренажер на основі штучного інтелекту для підготовки до проходження співбесід
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5

Нікіта Ремешевський
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект

(повна назва освітньої програми)
Керівник доц. Олександра Вітько
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ремешевському Нікіті Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Тренажер на основі штучного інтелекту для підготовки до проходження співбесід

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.

3. Вихідні дані до роботи Python, FastAPI, Hugging Face Transformers, датасет «Confident/Unconfident Facial Expression», модель ResNet18, Google Cloud Storage, Gemini, Firestore, HTML, CSS, JavaScript

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичні дослідження та обґрунтування вибору технологій
3) Проєктування і реалізація вебзастосунку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі і постановка задачі	19.05.2025 – 22.05.2025	виконано
3	Теоретичні дослідження та обґрунтування	22.05.2025 – 24.05.2025	виконано
	вибору технологій		
4	Проєктування і реалізація вебзастосунку	24.05.2025 – 31.05.2025	виконано
5	Написання пояснювальної записки	31.05.2025 – 02.06.2025	виконано
6	Перевірка на академічний плагіат	09.06.2025 – 10.06.2025	виконано
7	Нормоконтроль	10.06.2025 – 12.06.2025	виконано
8	Підготовка презентації та доповіді	12.06.2025 – 14.06.2025	виконано
9	Підготовка до захисту	14.06.2025 – 17.06.2025	виконано
10	Захист перед ЕК	18.06.2025	

Дата видачі завдання 19 травня 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олександра Вітько
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 83 с., 33 рис., 4 табл., 3 дод., 30 джерел.

ВЕБЗАСТОСУНОК, ГЕНЕРАТИВНИЙ ШТУЧНИЙ ІНТЕЛЕКТ, МУЛЬТИМОДАЛЬНИЙ АНАЛІЗ, НЕЙРОМЕРЕЖІ, ПРОХОДЖЕННЯ ІНТЕРВ'Ю, РОЗПІЗНАВАННЯ ГОЛОСУ, RESNET.

Об'єкт дослідження – автоматизований процес підготовки кандидатів до технічних співбесід у вебсередовищі, що поєднує інструменти інтерв'ювання, аналізу та зворотного зв'язку.

Предмет дослідження – вебзастосунок із інтегрованими ML-модулями, який автоматизує формування запитань, запис і оцінку відповідей кандидата за допомогою аналізу тексту, аудіо та відео.

Мета роботи – розробка прототипу вебплатформи для тренування до співбесід, що дозволяє користувачам у зручному інтерфейсі практикувати відповіді на типові запитання, отримувати персоналізований зворотний зв'язок і аналіз емоційного стану.

Методи дослідження включали огляд і синтез наукових публікацій у галузі HR-технологій і веброзробки, експериментальну інтеграцію передових ML-моделей та практичну реалізацію клієнт-серверної архітектури, а також використання хмарних сервісів.

В результаті було розроблено вебзастосунок «AI Interview Trainer» для симуляції технічних співбесід, який генерує питання за допомогою великої мовної моделі, записує й аналізує відповідь користувача через автоматичну транскрипцію й емоційний аналіз аудіо та відео, а також візуалізує результати та зберігає історію сесій у хмарі. Запропоноване рішення поєднує чистий фронтенд без додаткових фреймворків, надійну серверну логіку й сучасні ML-алгоритми, що забезпечує персоналізований зворотний зв'язок і дає основу для подальшого розвитку HR-інструментів.

ABSTRACT

Bachelor’s thesis contains: 83 pp., 33 fig., 4 tabl., 3 ann., 30 references.

GENERATIVE ARTIFICIAL INTELLIGENCE, INTERVIEW SIMULATION, MULTIMODAL ANALYSIS, NEURAL NETWORKS, RESNET, SPEECH RECOGNITION, WEB APPLICATION.

Object of study – an automated process of preparing candidates for technical interviews in a web environment that combines interviewing, analysis, and feedback tools.

Subject of study – a web application with integrated ML modules that automates question generation, recording, and assessment of candidate responses through text, audio, and video analysis.

Purpose – to develop a prototype web platform for interview training that enables users to practice answers to typical questions in an intuitive interface, receive personalized feedback, and analyze emotional state.

Research methods were based on reviewing and synthesizing scientific publications in HR technologies and web development, experimentally integrating state-of-the-art ML models, practically implementing a client-server architecture, and leveraging cloud services for real-time data storage and processing.

The project resulted in the development of «AI Interview Trainer», a web application for simulating technical interviews. It generates questions using a large language model, captures and analyzes user responses via automated transcription and emotional analysis of audio and video, visualizes results with interactive charts, and stores session history in the cloud. The solution combines a lightweight front end without extra frameworks, robust server-side logic, and modern ML algorithms to deliver personalized feedback and lay the groundwork for future HR-tech innovations.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

Adam – Adaptive Moment Estimation – адаптивне оцінювання моментів;

API – Application Programming Interface – інтерфейс програмування застосунків;

CSS – Cascading Style Sheets – мова стилів;

GCS – Google Cloud Storage – хмарне сховище Google;

HTML – Hypertext Markup Language – мова гіпертекстової розмітки;

IDE – Integrated Development Environment – інтегроване середовище розробки;

JS – JavaScript – мова програмування JavaScript;

JSON – JavaScript Object Notation – нотація об’єктів JavaScript;

LLM – Large Language Model – велика мовна модель;

NLP – Natural Language Processing – обробка природної мови;

ReLU – Rectified Linear Unit (activation function) – функція активації «випрямлена лінійна одиниця» ;

RGB – Red Green Blue – червоний, зелений, синій (кольорова модель);

URL – Uniform Resource Locator – уніфікований локатор ресурсів;

WAV – Waveform Audio File Format – формат аудіофайлів у вигляді хвильової форми.

ВСТУП

Сучасний ринок праці зазнав суттєвих змін протягом останніх років, про що свідчить зростання конкуренції та активне впровадження нових технологій у процес підбору кадрів. За даними «LinkedIn Talent Trends», за п’ять останніх років кількість вакансій із високими кваліфікаційними вимогами збільшилася приблизно на 25 %, що підтверджує більша зацікавленість HR-менеджерів у володінні сучасними технологіями та інноваційними підходами [1]. Водночас можна помітити, що попри доступні навчальні ресурси, багато кандидатів відчувають невпевненість у власних відповідях, не відповідають очікуванням роботодавців, а також мають недостатні комунікаційні навички, що зменшує їхні шанси на успішне працевлаштування.

Враховуючи це, розвиток штучного інтелекту надає нові можливості для вирішення цих проблем. Завдяки постійному розвитку алгоритмів машинного навчання, обробки природної мови та комп’ютерного зору у нас є можливість створювати інноваційні чи вдосконалювати вже існуючі системи для аналізу різних сигналів, жестів, міміки які супроводжують кандидата під час співбесіди. Після появи великих мовних моделей багато аспектів нашого життя зазнало радикальних змін. Велика частина компаній почала використовувати їх у різних галузях в тому числі і у процесі підбору персоналу. У цьому контексті особливо цікавою є робота «Can ChatGPT Challenge the Scientific Impact of Published Research, Particularly in the Context of Industry 4.0 and Smart Manufacturing?», яка досліджує можливості ChatGPT як інструмента аналізу та оцінки наукового внеску в галузі індустрії 4.0, що чудово демонструє перспективи використання подібних моделей у різних прикладних сценаріях [2].

Якщо казати точніше, то ШІ використовують для аналізу резюме, автоматизованого спілкування через чат-боти та відстеження залученості команди. Це дає змогу помітно скоротити рутинне навантаження на HR-

відділи та збільшити ефективність найму. Загалом, тренди показують, що ШІ стає невід’ємним інструментом у сучасному HR-середовищі.

Запропонована технологія буде корисним інструментом як для індивідуальних тренувань, так і для впровадження в корпоративні програми навчання. Це допоможе як пришвидшити процеси добору персоналу для HR-відділів, так і покращити їх якість в умовах постійних змін в сучасній економіці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Сфера підготовки до співбесід охоплює різноманітні прийоми, методи та засоби, розроблені, щоб допомогти претендентам здолати співбесіду з найкращим результатом. В епоху сьогодення цей процес набуває все більшої технологічної складності: стають популярними онлайн-симулятори, імітаційні співбесіди та застосування штучного інтелекту для оцінки відповідей [3].

Процес співбесіди є ключовим етапом у процесі найму, на якому роботодавці оцінюють:

– професійні знання та компетенції кандидата;

– його комунікативні навички та впевненість під час розмови;

– стресостійкість і вміння відповідати на складні запитання;

– невербальні сигнали (міміка, жести чи вираз обличчя).

Традиційні методи підготовки кандидатів це штудіювання книг, самостійне відпрацювання перед дзеркалом, перегляд навчальних відео та участь у пробних співбесідах. Проте такі підходи не завжди дають по-справжньому неупереджену оцінку й не враховують унікальні характеристики кожного кандидата. Залучення штучного інтелекту до підготовки до співбесіди здатне значно збільшити ефективність підготовки: автоматизований аналіз відповідей допомагає виявляти прогалини в знаннях, а персоналізовані поради допомагають покращити саме ті навички, в яких він почуває себе невпевнено.

Також, аналізуючи жести, міміку й тон голосу разом із тим, що говорить кандидат, ми отримуємо повне уявлення про особливості його манери поведінки і відповіді на питання. Це дає змогу оцінити не лише фахові навички і знання, але й комунікативну впевненість, здатність контролювати емоції та вміння подати себе. З іншого боку у розвитку

технологій штучного інтелекту виникають і певні етичні дилеми. Наприклад, застосування систем, які надають готові відповіді під час співбесіди, однозначно буде розцінене як «нечесне». Це питання вже викликає суперечки серед експертів та дослідників і вимагає розробки чітких правил і стандартів використання ШІ у процесі відбору персоналу [4].

1.2 Аналіз існуючих рішень

1.2.1 Вебзастосунок для допомоги з підготовкою до інтерв'ю Big Interview

Big Interview – це вебсервіс для відпрацювання навичок проходження співбесід [5]. У ньому є відеотренінги, навчальні матеріали й інструменти на базі штучного інтелекту, що допомагають покращити вміння відповідати на питання під час інтерв'ю.

Сервіс пропонує велику базу типових запитань, розділених за категоріями: (наприклад, «Розкажіть про себе»), технічні (за напрямком вашої спеціалізації), керівні та поведінкові. Є можливість записати відповідь через вебкамеру, після чого алгоритми аналізують темп мовлення, жести, зоровий контакт і ключові слова, щоб дати рекомендації щодо поліпшення і самовдосконалення.

Також Big Interview дозволяє пройти пробне інтерв'ю в реальному часі, як із віртуальним інтерв'юером, так і з реальними людьми. У бібліотеці навчальних матеріалів можна знайти відеоуроки, поради щодо співбесід та приклади успішних відповідей. Крім того, сервіс пропонує інтерактивні симуляції співбесід, де запитання адаптуються залежно від відповідей користувача.

Платформа корисна для студентів і випускників, які шукають першу роботу, досвідчених фахівців, які хочуть підтягнути свої навички для зміни

посади. А керівники й топменеджери на ній зможуть відпрацювати відповіді на запитання для високого рівня інтерв'ю.

Серед основних переваг Big Interview можна виділити велику базу запитань, можливість аналізу відповідей за допомогою штучного інтелекту, тренування у форматі відео та гнучку систему навчання. Водночас мінусами є платний доступ до більшості функцій, наявність лише англійської мови, дещо застарілий інтерфейс, з яким можна ознайомитися на рисунку 1.1, та дуже часті неточності в оцінюванні невербальних сигналів.

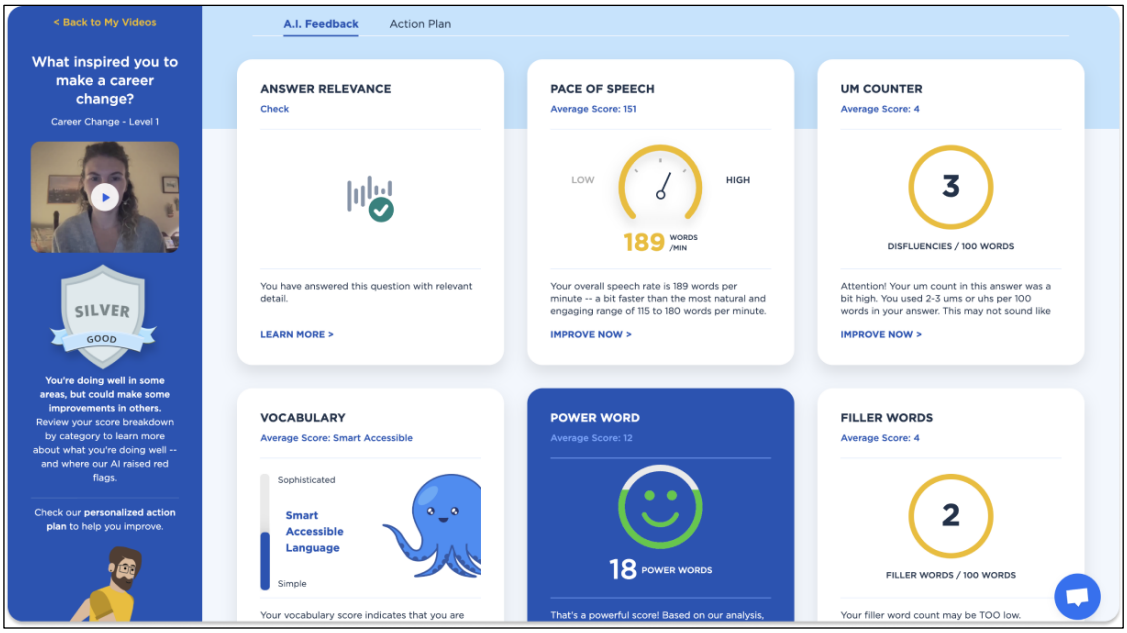


Рисунок 1.1 – Інтерфейс Big Interview

Платформа пропонує безкоштовну пробну версію, однак для повноцінної роботи доведеться оформити преміум-підписку, яка коштує приблизно 300 доларів для необмеженого використання, або ж 39 доларів за місяць. Окрім цього, Big Interview пропонує корпоративні пакети для бізнесу та освітніх закладів, а також регулярно оновлює базу запитань відповідно до актуальних вимог ринку праці, що допомагає користувачам залишатися конкурентоспроможними.

1.2.2 Вебзастосунок для допомоги з проходженням інтерв'ю наживо ParakeetAI

ParakeetAI – сучасний вебзастосунок для допомоги під час співбесід із застосуванням ШІ [6]. Платформа автоматично формує відповіді на питання інтерв'юера на основі передової моделі GPT-4o, що гарантує високий рівень точності та контекстної відповідності.

Інструмент без проблем інтегрується з будь-якими сервісами відеоконференцій (Zoom, Google Meet, Microsoft Teams тощо), дозволяючи користуватися підказками у звичному середовищі. ParakeetAI працює з 52 мовами, тож можна вести інтерв'ю рідною мовою, а всі дані залишаються захищеними й невидимими для рекрутера. Інтерфейс застосунку продемонстровано на рисунку 1.2.

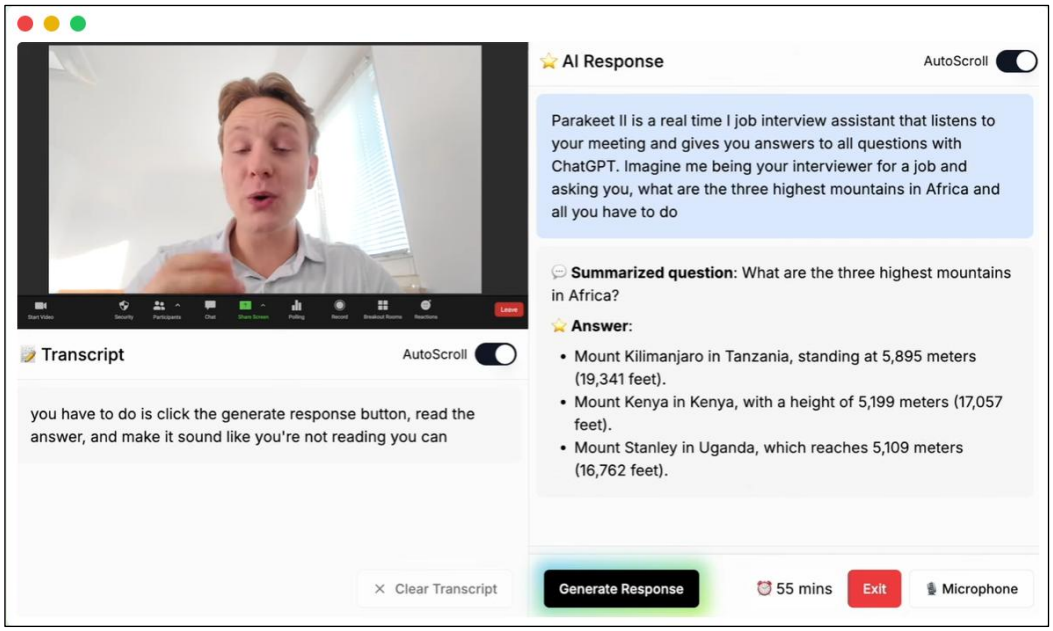


Рисунок 1.2 – Інтерфейс вебзастосунку Parakeet

Щоб розпочати роботу, потрібно просто завантажити своє резюме – це дає змогу ParakeetAI формувати відповіді з урахуванням вашого досвіду та фахових компетенцій. Після кожної сесії система підготує детальний звіт

із аналітикою вашої презентації та рекомендаціями з покращення навичок інтерв'ю.

Оплата здійснюється без фіксованої підписки: ви купуєте кредити, де 1 кредит прирівнюється до однієї години співбесіди, а пакети можна підбирати відповідно до власних потреб.

Окрім базового функціоналу, є модуль AI Apply Agent – він автоматично відповідає на скринінгові запитання та надсилає заявки від вашого імені, що суттєво спрощує пошук роботи.

Але використання таких систем викликає багато етичних питань. З одного боку, це інструмент, який допомагає кандидатам краще підготуватися та впевненіше відповідати на запитання. Але з іншого боку, його використання порушує абсолютно всі принципи чесності та прозорості у відборі кандидатів.

1.2.3 Платформа імітації питань на співбесідах Interviewsby.ai

Interviewsby.ai – це сучасний онлайн-сервіс, що ґрунтується на потужностях штучного інтелекту та автоматизує підготовку до співбесід [7]. Платформа самостійно генерує запитання для тренувальних інтерв'ю, аналізуючи ключові вимоги та обов'язки з опису вакансії, тож практичні вправи максимально наближені до реальних кейсів роботодавця. Від початкового етапу – вибору однієї з готових ролей – до завантаження власного опису посади, система адаптивно підбирає рівень складності питань, охоплюючи технічні, поведінкові та ситуаційні аспекти. Завдяки інтеграції з ChatGPT користувачі отримують не лише швидкий зворотний зв'язок щодо своїх відповідей, але й глибинний аналіз вербальних та невербальних сигналів, а також зразки «ідеальних» відповідей із роз'ясненнями, чому вони вважаються оптимальними. Це дозволяє не тільки відточити власні навички, а й знизити стрес та підвищити впевненість

перед справжньою співбесідою. Інтерфейс сервісу продемонстровано на рисунку 1.3.

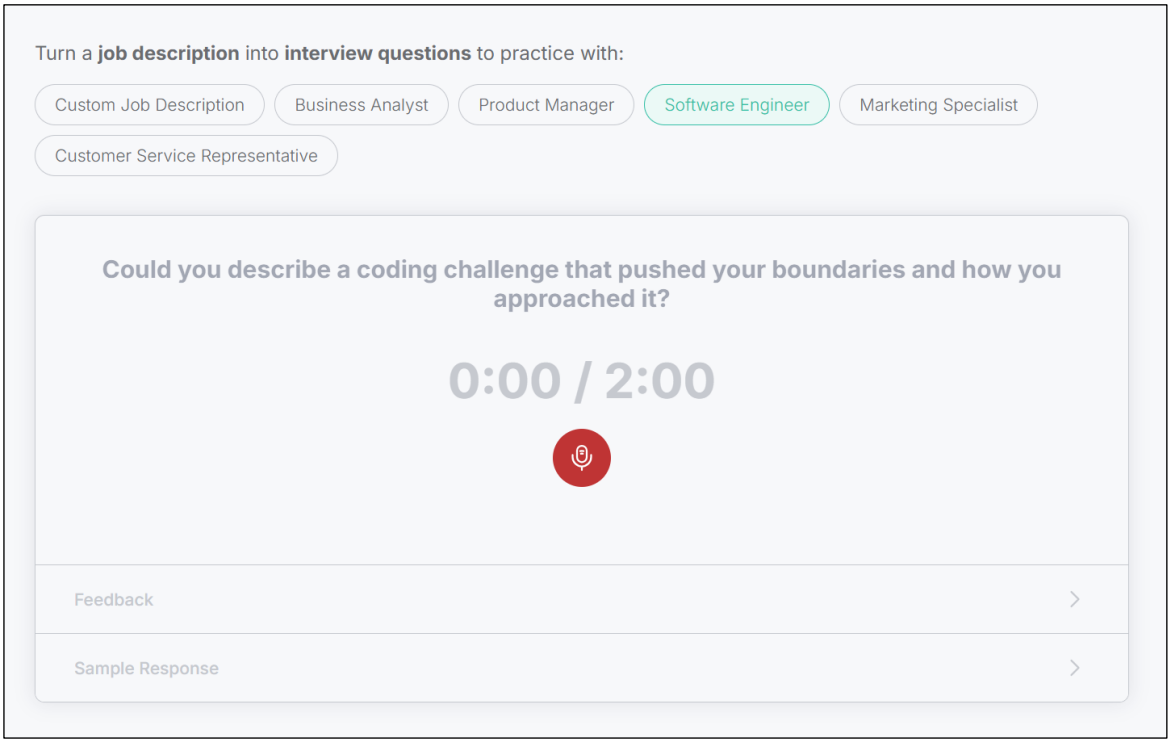


Рисунок 1.3 – Інтерфейс вебплатформи Interviewsby.ai

Родзинка платформи – це величезний пул із понад 20 000 запитань, розділених за індустріями й рівнями посад, що дозволяє відпрацювати найрізноманітніші сценарії інтерв'ю. AI-алгоритми аналізують не лише текст відповіді, а й інтонацію та темп мовлення, надаючи рекомендації щодо покращення відповідей.

Водночас тут нема відеоаналізу, тож оцінка міміки й жестів відбувається лише опосередковано – це зменшує точність перевірки невербальної комунікації. До того ж результативність тренування сильно залежить від того, наскільки чітко ви описали вакансію: неточний опис може породжувати нерелевантні запитання. Сервіс працює за схемою freemium: базовий функціонал безкоштовний, але повний набір запитань із

розгорнутим фідбеком доступний лише в платній підписці, що може стати бар’єром для деяких користувачів.

На відміну від Interviewsby.ai, розроблений тренажер зможе забезпечити повноцінний мультимодальний аналіз. За допомогою комп’ютерного зору, алгоритмів розпізнавання емоцій і тонових змін голосу він оцінює вербальні, невербальні та емоційні сигнали в комплексі.

1.3 Постановка задачі

Мета проєкту – розробити «розумний» тренажер для підготовки до співбесід, який поєднає методи машинного навчання, NLP (Natural Language Processing) і комп’ютерного зору. Система самостійно генеруватиме питання на основі опису вакансії та уподобань, оцінюватиме вербальні відповіді з погляду лексичної коректності та логічної послідовності й аналізуватиме невербальні сигнали – інтонацію, міміку й жести – аби видавати практичні поради для вдосконалення комунікативних навичок і підвищення впевненості під час інтерв’ю.

1.3.1 Визначення функціональних вимог застосунку

Перед початком сесії кандидат обирає професійну галузь або конкретну роль із довгого списку доступних спеціальностей. На основі цього вибору застосунок формує набір запитань, характерних для обраної позиції, включаючи питання про типові завдання, компетенції та очікування роботодавців у відповідній сфері.

Застосунок повинен надавати користувачеві можливість обирати рівень складності запитань, від початкового до просунутого, при цьому автоматично коригуючи формулювання та глибину тематики відповідно до обраного рівня. Кандидат в інтерфейсі вибирає бажаний рівень складності,

і тренажер формує запитання відповідно до цих налаштувань, забезпечуючи поступове ускладнення практики в процесі підготовки.

Для повноцінного аналізу відповідей застосунок повинен підтримувати одночасний запис відео та аудіо, що дозволить обробляти невербальні сигнали, міміку й інтонацію голосу кандидата. Після завершення відповіді система автоматично зберігає медіа-файли та готує їх до подальшої обробки алгоритмами комп’ютерного зору і обробки природної мови.

Користувач завжди повинен мати доступ до «еталонних» відповідей – зразків, створених за допомогою генеративного штучного інтелекту. Після кожної сесії тренажер пропонує перелік рекомендованих варіантів формулювання, що відповідають найкращим практикам, та пояснює, чому саме ці відповіді демонструють необхідні навички й компетенції.

Налаштування генерації запитань повинні бути параметризовані: користувач може вказати теми, на яких слід зробити акцент або додати власні критерії оцінки, наприклад, увагу до структури відповіді або використання термінології. Ці параметри дозволять адаптувати тренажер під індивідуальні потреби, створюючи максимально правдиві сценарії підготовки.

Після кожної відповіді тренажер проводить аналіз і видає розгорнутий фідбек: оцінює логіку й змістовність відповіді, відзначає сильні та слабкі сторони мовлення, інтонації та невербальних реакцій. Такий зворотний зв’язок допомагає кандидату зрозуміти, над чим варто попрацювати, та спрямовує подальшу підготовку.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

Перш за все, перед реалізацією застосунку були проведені теоретичні дослідження для ознайомлення і вибору найоптимальніших технологій, методів і моделей для реалізації поставленої задачі.

2.1 Теорія аналізу впевненості кандидата

2.1.1 Визначення «впевненості» в мовленні та міміці

У науковій літературі «впевненість» у мовленні і міміці розглядається як інтегральний показник внутрішнього стану та одночасно сигнал для співрозмовника. Впевнена промова вирізняється контрольованим темпом, чіткою артикуляцією та стабільною інтонацією без запинок, що створює враження компетентності й надійності [8]. Цьому відповідають акустичні параметри голосу – рівномірність енергії та низька варіабельність частоти, які можна оцінити за допомогою спектрального аналізу та MFCC-ознак у системах розпізнавання промови [9].

Невербальні сигнали, зокрема міміка, доповнюють вербальне повідомлення, причому мікровирази обличчя можуть видавати справжні емоції навіть за намаганням їх приховати. У тренажері на основі штучного інтелекту «впевненість» розуміється як поєднання голосових характеристик і виразу обличчя, що відображають самоконтроль і психологічний комфорт. Реалізація цієї ідеї передбачає аналіз акустичних параметрів та застосування згорткових нейронних мереж для розпізнавання міміки, що дозволяє в режимі реального часу оцінити готовність кандидата і надати йому зворотний зв’язок щодо невербальних і голосових сигналів. Крім того, для підвищення точності оцінки враховується контекстна адаптація моделі до індивідуальних особливостей мовлення та міміки користувача.

2.1.2 Підходи до обробки аудіо- та відеосигналів

У сучасній теорії обробки аудіосигналів першим етапом вважають приведення запису до єдиних умов: усунення фонового шуму та нормалізацію рівня гучності, що створює більш стабільні вхідні дані для подальших розрахунків. Після цього виконують перетворення сигналу у частотно-часову область – найчастіше через короткочасне перетворення Фур'є (STFT) або виділення мел-частотних кепстральних коефіцієнтів (MFCC). Саме ці коефіцієнти використовують як вхідні карти ознак для згорткових нейронних мереж (CNN), які навчаються розпізнавати характерні візерунки спектра, пов'язані з різними емоційними станами мовця чи його динамікою інтонації [9]. Паралельно до CNN можуть застосовуватися рекурентні або трансформерні архітектури для моделювання часових залежностей у сигналі: вони фіксують, як змінюється інтенсивність та тембр голосу в часі, що особливо важливо для визначення плавності та впевненості мовлення.

У частині відеоаналізу ключовим етапом є виділення просторових ознак із окремих кадрів з допомогою згорткових мереж, натренованих на великих наборах зображень. Спершу система обробляє кожен кадр індивідуально, витягуючи зі зміни текстур обличчя, контурів та ключових точок міміки високорівневі ознаки, які служать основою для подальшої інтерпретації експресії. Щоб врахувати часові аспекти, ці просторові векторні представлення можуть об'єднуватися в послідовності й аналізуватися за допомогою простих рекурентних або 1D-згорткових мереж, що дозволяють фіксувати динаміку рухів: підйоми брів, зміни куточків рота тощо. Іншим варіантом є використання технік агрегації – наприклад, середнього чи максимального поєднання ознак із сусідніх кадрів, що допомагає зменшити вплив шумів та раптових артефактів відеозапису.

Важливим аспектом є застосування перенесеного навчання: мережа, попередньо натренована на масштабних задачах класифікації облич чи загальних об’єктів, виступає джерелом «загальних» просторових патернів. Під час адаптації до завдання розпізнавання емоцій чи мікровиразів ці шаблони уточнюються на цільових даних, що дозволяє системі швидко освоювати специфіку людської міміки без потреби в надмірних обсягах розмічених відеозаписів. За рахунок такого підходу обробка відео стає не лише точнішою в плані виявлення тонких емоційних нюансів, але й ефективнішою з точки зору обчислювальних ресурсів, оскільки більша частина вагів використовується з уже навчених шарів, а донавчання відбувається тільки в кількох останніх рівнях мережі.

2.2 Вибір моделей машинного навчання

2.2.1 Дослідження і вибір архітектури аудіомоделі

У сучасних інтерактивних системах оцінка впевненості мовця виступає не просто додатковим елементом аналітики, а ключовим компонентом, який дозволяє користувачу отримувати зворотний зв’язок у реальному часі та коригувати власну поведінку під час спілкування. Інтегрування модуля аналізу впевненості відкриває можливість не лише відображати фактичні результати відповіді, а й прогнозувати тенденції змін емоційного стану, що особливо важливо для тренажерів і систем самовдосконалення.

З технічної точки зору завдання оцінки впевненості можна розглядати як поєднання класифікації емоцій та регресійного перетворення в єдину шкалу: спочатку голосовий сигнал розпізнається за набором базових емоцій, а потім відносні ваги позитивних і негативних складових перетворюються в плавну метричну оцінку. Навчання мережі виконано на корпусах

RAVDESS [10], SAVEE [11], TESS [12] та URDU [13], що дозволяє досягти високої точності розпізнавання базових емоційних класів.

Загалом, однією з найбільш ефективних архітектур для такого поетапного підходу є трансформерна модель Wav2Vec 2.0, що в режимі self-supervised навчання здатна будувати універсальні спектрально-часові репрезентації з необмежених обсягів некерованих даних, а потім адаптуватися під конкретні класи емоцій із мінімальним обсягом розмічених прикладів [14].

У підготовці вхідних даних насамперед проводиться уніфікація аудіопотоку: звук нормалізується за гучністю, видаляються фонові шуми спеціальними спектральними фільтрами, а паузи та тихі ділянки відсікаються для уникнення артефактів спектральної статистики. Після цього застосовується ковзне вікно з функцією Хеммінга для формування короткочасних фрагментів, на яких обчислюються мел-спектрограми або безпосередні тензорні представлення через спеціалізований feature-extractor. Такий підхід забезпечує сталість вхідного формату та зберігає часовий контекст інтонаційних зсувів, критично важливий для емоційного розпізнавання.

На рівні виходу трансформера використовується простий лінійний шар, що перетворює вектор логітів емоцій у скалярну оцінку впевненості: позитивні ймовірності («радість», «нейтральність») підсумовуються, від них віднімаються негативні («страх», «засмученість»), і результат проходить через сигмоїдальну функцію, щоб потрапити до інтервалу (0,1). Для згладжування часової кривої ґрунтовність оцінок підвищується за допомогою ковзного аналізу з перекриттям фрагментів та зваженого середнього, що зменшує вплив раптових сплесків шуму або мікрофонних артефактів. Така поєднана архітектура гарантує одночасно високу точність, стійкість до варіативності даних і прозорість інтерпретації кінцевої метрики впевненості.

2.2.2 Обґрунтування вибору ResNet18 для відеоаналізу

ResNet18 (Residual Network із 18 шарами) – це одна з найпопулярніших архітектур згорткових нейронних мереж, запропонована командою Microsoft Research у 2015 році. Головна її особливість – залишкові (residual) блоки, які через «короткі» (identity) зв’язки допомагають долати проблему зникнення градієнта та стабілізувати навчання навіть у глибоких мережах. Кожен резидуальний блок складається з двох послідовних згорткових шарів (3×3) із пакетним нормуванням (BatchNorm) та активацією ReLU, а також з identity-доріжки, яка додає вхід блоку до його виходу. Архітектуру цієї мережі продемонстровано на рисунку 3.10.

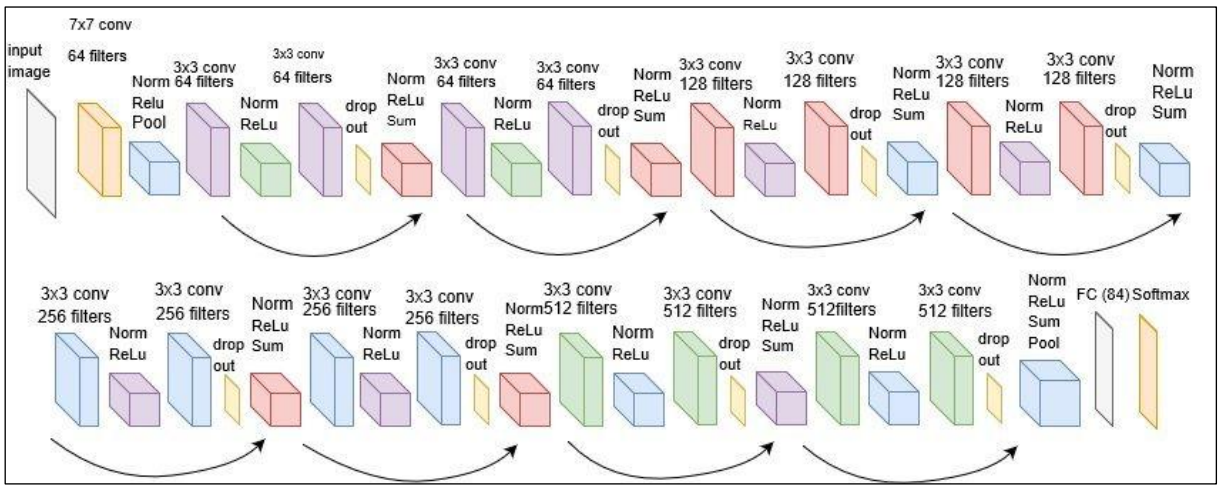


Рисунок 2.1 – Архітектура мережі ResNet18 [15]

При виборі ResNet18 для реалізації завдання відіграли роль кілька чинників. Архітектура містить близько 11 мільйонів параметрів, що значно менше, ніж у глибших версіях, тому донавчання відбувається швидше і з меншими вимогами до відеопам’яті – важливий аспект для обробки відеопотоку в реальному часі. Одночасно модель забезпечує достатньо глибоке представлення просторових ознак, здатне виявляти контури,

текстури й ключові точки міміки. Попереднє навчання на великому колекціонованому наборі зображень дало змогу сформувати універсальні фільтри для детектування базових патернів, що потім уточнюються під час адаптації до специфіки власного датасету.

ResNet18 також продемонструвала ефективність на невеликих зображеннях (32×32 пікселі), тому обробка облич розміром близько 48×48 пікселів відбувається з мінімальними втратами в контрасті та деталях. Це забезпечує високу точність розпізнавання експресії навіть при помірній роздільній здатності вхідних кадрів. Для донавчання перші шари мережі «заморожуються» на етапі фіксованих фільтрів, а класифікаційний блок перенавчається на відеозаписах з різними виразами обличчя. Після досягнення стабільної якості на валідації глибші шари розморожуються з поступовим зниженням темпу навчання, що дозволяє фільтрам адаптуватися до тонких емоційних нюансів.

У процесі тренування було застосовано балансування класів під час формування мініпакетів, регуляризацію через відсів нейронів та ранню зупинку на основі метрик на валідаційній вибірці. Завдяки цьому ResNet18 із донавчанням показала високу здатність вловлювати мікродеталі міміки, зберігаючи при цьому обчислювальну ефективність, необхідну для роботи в режимі реального часу.

2.3 Середовище та засоби розробки

2.3.1 Середовище програмування PyCharm

PyCharm є інтегрованим середовищем розробки (IDE) компанії JetBrains, орієнтованим суто на мову програмування Python і вибраним для реалізації AI-тренажера завдяки широкому спектру можливостей та глибинній інтеграції з екосистемою Python [16].

IDE оснащене контекстно-залежними підказками коду та автодоповненням (type hints, аналіз імпортів, структурна навігація), що помітно прискорює написання програмних модулів і знижує кількість синтаксичних і логічних помилок. Вбудовані засоби рефакторингу дозволяють оперативно перейменовувати елементи коду, виносити методи й класи та змінювати організацію проєкту без ризику пошкодити його структуру. Для діагностики й тестування передбачено потужний дебагер із підтримкою точок зупину, моніторингу стеку викликів і змінних, що дуже зручно використовувати під час навчання моделей. А також середовище розробки має вбудовану інтеграцію з профайлером і популярними тестовими фреймворками.

PyCharm також забезпечує зручну роботу з реляційними базами даних: можна підключатися до різних СУБД через вбудовані редактори SQL, автоматично генерувати ORM-моделі та переглядати результати запитів безпосередньо в IDE. Повноцінна підтримка вебстеків у базовій конфігурації включає шаблони Django, плагіни для Flask, керування середовищами через virtualenv або Conda, контейнеризацію проєктів у Docker та можливість віддаленого розгортання. Під час розробки, особливо зручно було працювати з фреймворком FastAPI саме у PyCharm [17].

Серед недоліків можна зазначити підвищене навантаження на оперативну пам’ять і процесор, особливо під час індексації великомасштабних проєктів. До того ж у середовища розробки існують дві редакції: безкоштовна Community та розширена Professional із більшою підтримкою вебфреймворків, віддаленого дебагу й поглибленого SQL-інструментарію. І загалом, інтерфейс і велика кількість налаштувань можуть ускладнити адаптацію новачків, створюючи додаткові перешкоди на початкових етапах знайомства з IDE. Крім того, PyCharm пропонує вбудовану інтеграцію з системами відстеження версій (наприклад, Git), що спрощує колаборацію та управління змінами в командних проєктах.

2.3.2 Мова програмування Python

Python – високорівнева мова програмування універсального призначення, що вирізняється простотою та читабельністю синтаксису, що в свій час робить її ідеальною для швидкої розробки складних застосунків [18].

По-перше, зрозуміла відступна структура та мінімальна кількість синтаксичних конструкцій дозволяють розробнику зосередитися на логіці програми, уникаючи ручного керування пам’яттю або надмірної шаблонності коду [19]. По-друге, насичена екосистема – від бібліотек NumPy і Pandas для обробки даних до TensorFlow, PyTorch і Scikit-learn для машинного навчання, OpenCV для комп’ютерного зору та SpaCy чи NLTK для NLP – дає змогу швидко інтегрувати готові модулі замість реалізації базових алгоритмів власноруч [20]. По-третє, велика спільнота розробників гарантує постійне оновлення інструментів, обмін досвідом та велику кількість документації і прикладів.

Серед обмежень варто відзначити інтерпретований характер мови, що зазвичай забезпечує нижчу швидкодію порівняно з компільованими рішеннями (наприклад, на C++ або Java). Наявність GIL (Global Interpreter Lock) ускладнює масштабування багатопотокових обчислень у Python, хоча це можна частково обійти за допомогою багатопроцесорності або асинхронних підходів. Динамічна типізація іноді призводить до появи помилок лише під час виконання, якщо не застосовувати статичний аналіз чи type hints. Крім того, використання великих наукових блоків іноді доводиться використовувати сторонні обгортки чи надбудови.

У рамках розробки AI-тренажера Python слугуватиме основою для об’єднання модулів обробки природної мови, аудіо- та відеопотоків із вебінтерфейсом. Лаконічність синтаксису спростить імплементацію і підтримку ключових алгоритмів, а вбудована можливість створення

асинхронного рішення забезпечить високу продуктивність та відмовостійкість при паралельній обробці запитів під час аналізу.

2.4 Фреймворки та бібліотеки

У розробці програмного забезпечення, особливо у галузі штучного інтелекту, фреймворки та бібліотеки відіграють надзвичайно важливу роль. Вони дозволяють суттєво прискорити процес створення, перевірки й удосконалення рішень, адже розробнику немає потреби власноруч втілювати базові алгоритми або архітектури – усе це вже втілено у формі зручних інструментів, які необхідно лише правильно скласти докупи. Зокрема, у сферах машинного навчання, комп'ютерного зору та обробки аудіо, ці інструменти перетворилися на стандарт. Під час розробки цього проекту буде використано чимало бібліотек і фреймворків. Буде доцыльно ознайомитися з основними із них.

TensorFlow – один із найпопулярніших фреймворків для машинного та глибинного навчання, розроблений компанією Google [21]. Він дозволяє створювати та тренувати складні нейронні мережі, працювати з великими обсягами даних і ефективно розгортати моделі як на сервері, так і на локальних машинах. TensorFlow підтримує як високорівневі API (через Keras), так і низькорівневий контроль над мережею. Його перевагою є масштабованість – одну й ту саму модель можна запустити на CPU, GPU або TPU за допомогою однієї строки у коді.

Keras – високорівнева бібліотека, яка працює поверх TensorFlow і спрощує створення нейронних мереж. Як вказано на головній сторінці бібліотеки – це API для глибокого навчання, створене для людей, а не для машин. Keras зосереджується на швидкості налагодження, елегантності та лаконічності коду, зручності обслуговування та розгортання [22]. Keras ідеально підходить для швидкого прототипування, а також підтримує

основні типи шарів, втрат, оптимізаторів та метрик, що значно полегшує експериментування та налагодження моделей.

Scikit-learn – універсальна бібліотека для машинного навчання, яка містить широкий спектр алгоритмів класифікації, кластеризації, регресії та зниження розмірності [23]. У роботі sklearn буде використовуватися здебільшого для попередньої обробки даних (нормалізації, масштабування, крос-валідації, тощо).

OpenCV (Open Source Computer Vision Library) – одна з найпотужніших бібліотек для обробки зображень та відео. У межах тренажера вона буде використовуватися для аналізу невербальних сигналів користувача: міміки, рухів голови, погляду, загальної емоційної реакції. Бібліотека має широкий набір функцій для роботи з камерою, відеопотоками, об’єктами на зображенні, фільтрами й трекінгом [24].

Librosa – спеціалізована бібліотека для аналізу аудіо сигналів, особливо добре працює з музичними та мовними даними [25]. У контексті цього застосунку вона буде застосована для попередньої обробки аудіо відповідей користувача і виділення ознак. Ці параметри є критично важливими для подальшої класифікації або оцінки якості мовлення.

Для реалізації серверної частини обрано FastAPI через асинхронну архітектуру цього фреймворку на основі ASGI, що забезпечує високоефективну обробку мультимодальних запитів у реальному часі. Використання аннотацій типів Python дозволяє чітко описувати параметри запитів і формат відповідей, підвищуючи читабельність коду та мінімізуючи ризик помилок типізації. Механізм вбудованої залежності (Dependency Injection) сприяє модульності та гнучкості конфігурації компонентів – наприклад, обробки файлів, взаємодії з базою даних чи викликів зовнішніх сервісів. Висока продуктивність досягається завдяки легковаговому ASGI-серверу Uvicorn і асинхронній основі Starlette, що дозволяє масштабувати систему та обслуговувати велику кількість одночасних з’єднань. У сукупності всіх цих переваг FastAPI створює надійну й гнучку

платформу для побудови тренажера співбесід із гарантованою швидкістю реакції та стійкістю до навантажень в розумних межах.

Використання цих інструментів забезпечує ґрунтовну технічну основу для реалізації інтелектуального тренажера співбесід. Вони дозволяють ефективно працювати з мультимодальними даними – як звуковими, так і візуальними – і створювати адаптивне та персоналізоване середовище підготовки. Завдяки гнучкості і різноманітності вищезгаданих бібліотек Python, можливо досягти високого рівня точності аналізу та якості зворотного зв’язку не жертвуючи при цьому швидкодією системи.

2.5 Набори даних та зовнішні сервіси

2.5.1 Використання Vertex AI Api

У роботі в якості ключового зовнішнього сервісу для генерації запитань і глибинного аналізу відповідей було обрано платформу Vertex AI від Google, а саме модель генеративного штучного інтелекту Gemini-2.5-Flash. Вона дозволяє працювати з великим об’ємом контексту – до мільйона токенів одночасно і завдяки оптимізаціям «Flash» демонструє низькі затримки при запитах [26], що дуже важливо для будь-якого вебзастосунку. Інтеграція у тренажері здійснюється через офіційний Python-клієнт google-cloud-aiplatform, що надає зручний SDK (Software Development Kit) який дозволяє генерувати відповіді за допомогою кількох рядків коду.

Застосування Gemini-2.5-Flash у поєднанні з Vertex AI API забезпечує низку переваг, які були враховані під час проєктування рішення кваліфікаційної роботи. По-перше, хмарна інфраструктура повністю керується Google і автоматично масштабується під навантаження, усуваючи потребу у налаштуванні власних серверів або в розгортанні кластерів для обробки запитів. По-друге, текстовий режим взаємодії у тренажері забезпечує високу точність аналізу: модель добре розпізнає нюанси

формулювань а також структурних і логічних зв’язків у відповідях кандидата, а зворотний зв’язок, в свою чергу, подається у вигляді детальних та варіативних текстових рекомендацій. Такий підхід дозволяє адаптувати підказки до індивідуального стилю спілкування, водночас спрощуючи інтеграцію компонентів і знижуючи складність реалізації. І нарешті, Flash-версія Gemini-2.5 відзначається оптимізованою архітектурою і застосуванням знань дистиляції, завдяки чому витрати на обчислення і токени залишаються помірними навіть при інтенсивному навантаженні сотень чи тисяч користувачів [27].

2.5.2 Використання хмарного сервісу для зберігання даних Google Cloude Storage

Для зберігання аудіо- та відеофайлів упродовж роботи застосунку обрано GCS (Google Cloud Storage). Його масштабованість дозволяє не турбуватися про обсяг локального дискового простору, а розміщення даних у дата-центрах Google гарантує безперервний доступ до файлів і зводить до мінімуму ризик їхньої втрати. GCS пропонує декілька класів зберігання – від економних варіантів для рідкісного доступу, до високошвидкісних і надійних, які, хоч і дорожчі, найкраще підходять для системи, що розроблюється. Для інтеграції використовується офіційна бібліотека google.cloud для Python, завдяки якій завантаження та вивантаження файлів реалізуються всього кількома рядками коду. Крім того, шифрування даних виконується на стороні сервісу, а автентифікація й авторизація здійснюються виключно за допомогою OAuth-токенів. При цьому можна налаштувати політики lifecycle для автоматичного переміщення старих файлів у холодні класи та їх видалення за розкладом. Доступ до бакетів контролюється за допомогою ролей IAM, що дозволяє делегувати обмежені права різним компонентам системи.

2.5.3 Використання хмарного сервісу Cloud Firestore для зберігання транскрипції і аналізу відповідей кандидата

Загалом Firestore є документно-орієнтованою NoSQL-базою даних із підтримкою realtime-підписок. Кожен результат аналізу й обчислень моделей зберігається у власному документі, а документи об’єднуються за сесіями (записами інтерв’ю). Завдяки механізму оновлень у реальному часі клієнтський інтерфейс отримує часткові результати аналізу одразу після їх генерації, що звільняє користувача від необхідності чекати на підсумковий звіт і дозволяє поступово ознайомлюватися з даними. Firestore також пропонує гнучкі можливості фільтрації та запитів, що спрощує пошук необхідної інформації. Окрім того, клієнтська бібліотека автоматично кешує отримані дані, тож навіть за відсутності інтернет-з’єднання користувач може переглядати раніше завантажені результати, а після відновлення зв’язку відбудеться їхня синхронізація. У підсумку Firestore є зручною базою для зберігання транскрипцій, аналітики та метаданих, яка легко інтегрується з фронтендом, не потребує налаштування власних серверів і миттєво масштабується відповідно до навантаження.

2.5.4 Використання бібліотеки SpeechRecognition

Для розшифрування відповіді користувача і транскрипції її у текст було вирішено використати бібліотеку SpeechRecognition, а саме метод recognize_google. який розглядається як інтерфейс до хмарного сервісу Google Speech Recognition API [28]. Саме ця зв’язка обрана з кількох причин.

По-перше, SpeechRecognition надає зручний Python-інтерфейс до різних сервісів розпізнавання мовлення (буде використовуватися саме online-версія), а метод recognize_google відправляє аудіо на сервер Google і повертає текстовий результат, без потреби розгортати власні моделі чи

налаштовувати громіздку інфраструктуру. Це суттєво скорочує час розробки базового модуля транскрипції відповідей кандидата.

По-друге, Google Speech Recognition демонструє високу якість розпізнавання навіть за наявності фонового шуму та підтримує десятки мов і діалектів, що критично важливо в мультикультурному середовищі потенційних користувачів тренажера. Наявність готового хмарного рішення дозволить отримати точні результати без інвестицій у підготовку великих мовних корпусів і тренування власних моделей.

По-третє, використання цього сервісу є безкоштовним у межах «первинного» ліміту запитів Google, що робить його оптимальним для етапу розробки й тестування.

Недоліки цього підходу полягають у необхідності стабільного інтернет-з’єднання та потенційних ризиках з точки зору конфіденційності: аудіо дані користувача передаються на віддалений сервер, тому в умовах відкритого середовища потрібно буде дотримуватися вимог політик обробки й захисту персональних даних, або використовувати власні моделі розпізнавання. Втім, для прототипу розробленого в рамках кваліфікаційної роботи обране рішення буде найоптимальнішим, оскільки воно поєднує в собі швидку інтеграцію з уже готовими хмарними сервісами, високу точність розпізнавання мови та мінімальні витрати на розгортання й підтримку.

2.5.5 Використання набору даних Confident Unconfident Facial Expression

У рамках кваліфікаційної роботи для навчання та валідації модуля аналізу невербальних сигналів обрано відкритий датасет «Confident/Unconfident Facial Expression», доступний на платформі Hugging Face [29]. Цей набір містить тисячі зображень облич, розмічених у

дві категорії: «Confident» та «Unconfident», завдяки чому ідеально підходить для задач бінарної класифікації.

По-перше, він уже структурований та попередньо відфільтрований – зображення підібрані з різних джерел та мають однорідний розмір (48x48 пікселів) і якість, що значно спрощує етап попередньої обробки. По-друге, класи «Confident» і «Unconfident» становлять приблизно рівні за обсягом підмножини, що дозволяє уникнути серйозного дисбалансу класів під час тренування моделі.

По-третє, датасет охоплює різноманітність вікових груп, статі та етнічних ознак, що підвищує стійкість моделі до варіацій у зовнішньому вигляді кандидатів. Це продемонстровано на рисунку 2.1, де представлені кілька прикладів зображень з датасету.

Рисунок 2.2 – Приклад зображень у датасеті «Confident/Unconfident Facial Expression»

У межах застосунку будуть додатково поділятися вихідні відеокадри на окремі рамки (frames) і для кожної рамки спершу буде знаходитися і виділятися обличчя, якщо воно є. І лише після цього виконуватиметься класифікація зображення за допомогою попередньо натренованої мережі. І

використання саме цього датасету дасть змогу швидко отримати реалістичні результати оцінки «рівня впевненості» у відповіді кандидата, не витрачаючи час на власний збір та маркування великої кількості прикладів.

У підсумку, «Confident/Unconfident Facial Expression» забезпечує високу якість навчальних даних для завдання розпізнавання невербальної комунікації, має відкриту ліцензію й широко використовується в дослідницьких проєктах, що робить його найкращим існуючим вибором для прототипу нашого тренажера.

2.6 Визначення загальної архітектури проєкту

У майбутньому проєкті буде побудовано чотирирівневу архітектуру: клієнтський інтерфейс, серверний шар, аналітичні модулі й зовнішній NLP-сервіс. На боці клієнта через MediaRecorder API відбуватиметься запис аудіо й відео, а сервер на FastAPI прийматиме файли, зберігатиме їх у тимчасовій структурі та запускатиме обробку. Аудіомодуль здійснюватиме шумозаглушення, нормалізацію та розрізання потоку на сегменти перед перетворенням у тензори для аналізу емоцій, а відеомодуль – екстракцію кадрів із їх нормалізацією та подачею до глибокої нейронної мережі для виявлення міміки. Після локальних обчислень запит на генерацію запитань і семантичний аналіз відповідей спрямовуватиметься до сервісу Gemini (Vertex AI). Усі файли й проміжні дані зберігатимуться в Google Cloud Storage, а метадані й стани обробки фіксуватимуться в Firestore, що забезпечить масштабованість, стійкість і гнучкість системи. З діаграмою структури майбутнього застосунку можна ознайомитися на рисунку Б.1 у додатку Б.

3 ПРОЄКТУВАННЯ І РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

Після розгляду і вибору технологій наступає етап проєктування і розробки вебзастосунку.

3.1 Технічні вимоги до вебзастосунку

Загалом вебзастосунок має забезпечувати інтерактивну генерацію запитань, запис аудіо і відео, а також зберігання й обробку мультимедійних і текстових даних з аналізом у режимі реального часу. Усі компоненти (інтерфейс, моделі штучного інтелекту, бекенд) повинні бути модульними, масштабованими та стійкими до збоїв. Розглянемо кожен аспект цих вимог.

3.1.1 Технічні вимоги до моделей застосунку

У межах даного проєкту планується реалізація двох окремих модулів – для аналізу аудіо та відео. Основна реалізація виконується мовою Python із використанням бібліотек OpenCV, PyTorch, torchvision, librosa та transformers. Аудіофайли будуть оброблятися у форматах WAV з частотою дискретизації 16 кГц. Після попередньої обробки звукові дані будуть перетворюватися у формат, придатний для подачі на вхід моделі. Візуальний модуль буде працювати з окремими кадрами відео, які приводяться до стандартного розміру 48×48 пікселів та нормалізуються згідно з поширеними параметрами. Кожен із модулів розробляється з можливістю запуску як на GPU, так і на CPU, що дозволяє забезпечити гнучкість залежно від ресурсів, наявних у користувача. Інтерфейси модулів мають бути уніфікованими, щоб в майбутньому можна було змінювати або оновлювати модель без необхідності переписувати основну логіку роботи. Уся система буде будуватися з урахуванням принципів модульності: окремі

частини коду відповідають лише за свою функціональність, а також передбачено обробку помилок і базове логування, щоб полегшити налагодження і подальшу підтримку.

3.1.2 Технічні вимоги до серверної частини застосунку

Сервер буде створений за допомогою мови Python та фреймворку FastAPI, що дозволить обробляти запити асинхронно й ефективно взаємодіяти з клієнтом. Відео-відповіді користувача надходитимуть на бекенд, де їх автоматично зберігатимуться у структуру папок за поточною сесією, конвертуватимуться у формат MP4 і витягуватимуть звукову доріжку в WAV для подальшої обробки. Для отримання текстової версії відповіді аудіо передаватиметься на зовнішній транскрипційний сервіс, а отриманий текст зберігатиметься локально. Далі транскрипти надходитимуть у модуль оцінювання для перевірки змісту, а відео-фрагменти аналізуватимуться внутрішнім алгоритмом, який формуватиме криву впевненості на основі візуальних сигналів. Після завершення всіх операцій результати й файли завантажуватимуться в Google Cloud Storage, а ключові метадані фіксуватимуться в Firestore. Стан обробки кожного запиту зберігатиметься у внутрішньому сховищі статусів і буде доступний для моніторингу клієнтом. Усі параметри конфігурації та секретні ключі задаватимуться через змінні середовища, що забезпечить просте масштабування та безпечне оновлення налаштувань.

3.1.3 Технічні вимоги до інтерфейсу застосунку

Інтерфейс буде створено на чистих HTML5, CSS3 та JavaScript без використання зовнішніх фреймворків, щоб забезпечити сумісність із будь-яким браузером. Головна сторінка міститиме поле для вводу спеціалізації, вибір рівня складності та текстове поле для додаткових вказівок, а також

кнопки для початку інтерв’ю, перегляду фідбеку та історії сесій. Запис відповіді користувача реалізовано через MediaRecorder API, із наочними індикаторами запису й статусу аналізу в блоці з питаннями. Взаємодія з бекендом здійснюватиметься за допомогою Fetch API, а списки питань та статуси тимчасово кешуватимуться в локальному сховищі, щоб не втрачати дані при оновленні сторінки. Сторінка з фідбеком рендеритиметься за допомогою шаблонів Jinja2, де відобразяться оцінки та графіки з Chart.js, а історія інтерв’ю – у вигляді таблиці з пагінацією. Лаконічна кольорова гамма й легкі hover-ефекти підвищать зручність, а модульна структура JavaScript-файлів (отримання запитань, запис, оновлення статусу, вивід графіків) полегшить подальшу підтримку. Константи на початку скриптів (URL API, інтервали опитування) забезпечать швидку зміну налаштувань без пошуку по всьому коду. Додатково реалізовано адаптивне відображення для мобільних пристроїв, що гарантує коректну роботу інтерфейсу на різних екранах.

3.2 Структура проєкту

Структура проєкту буде організована для чіткого розмежування всіх компонентів системи і полегшення навігації під час розробки та підтримки. У кореневій папці Interview_Trainer розміщуватимуться піддиректорії для бекенду (FastAPI), фронтенду (статичні файли HTML/CSS/JS), аналітичних модулів (Python-скрипти для обробки аудіо/відео) та конфігураційних файлів. Окремий каталог відведено для тестів і документації, а також для шаблонів Jinja2 і ресурсів Chart.js. Завдяки такому поділу легко знайти потрібний компонент, додати нові модулі або внести зміни без ризику порушити загальну архітектуру. Крім того, базова структура вже передбачає можливість масштабування за рахунок виділення окремих сервісів у майбутніх оновленнях. Повна ієрархія файлів та папок відображена на рисунку 3.1.

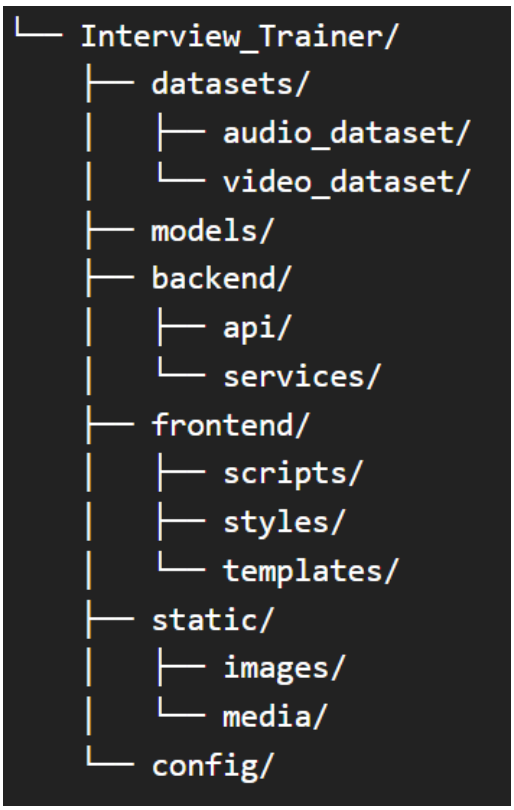


Рисунок 3.1 – Структура проєкту «Interview Trainer»

У загальній структурі буде 5 основних директорій:

– datasets буде використовуватись для зберігання даних, які потрібні для тренування і тестування моделей. А у внутрішніх папках audio_dataset і video_dataset міститимуться відповідні датасети, які будуть застосовані для навчання моделей;

– у models зберігатимуться машинні моделі, які будуть використовуватись для оцінювання відповідей, наприклад, моделі для розпізнавання емоцій, тональності або рівня впевненості;

– backend буде містити весь серверний код. У папці services буде логіка обробки, наприклад, сервіс для аналізу відповідей або генерації фідбеку. Окремо буде папка api, де знаходитимуться маршрути (ендпоінти) для обміну між фронтендом і бекендом;

– у frontend входитиме весь клієнтський інтерфейс. У scripts будуть JavaScript-файли, які керуватимуть логікою на клієнті. У styles будуть CSS-

файли, а в templates – HTML-шаблони, які, в подальшому будуть рендеритись через Jinja2;

– static буде окремою папкою для всіх статичних файлів, які не змінюються під час роботи, наприклад, іконки, картинки, медіафайли, стилі, скрипти, якщо не хочемо їх змішувати з робочими файлами;

– config буде зберігати конфігураційні файли, наприклад, налаштування API, шляхи до моделей, параметри таймерів або інші глобальні константи, які будуть використовуватись у скриптах і бекенді.

3.3 Реалізація моделі аналізу рівня впевненності у голосі

Отже, після всіх проведених досліджень і визначення вимог було розпочато створення першої моделі. Оскільки датасету з розміченими класами «впевненний» і «невпевненний» в аудіо-форматі не існує, то було прийнято рішення спершу навчити модель класифікувати базові емоції, для яких існує багато датасетів.

3.3.1 Попередня робота з датасетами

Усі такі датасети містять короткі (3–5 секунд) аудіозаписи, на яких звучать різні фрази з різними інтонаціями: сумні, щасливі, злі, нейтральні, огидні, налякані, здивовані чи спокійні. Як уже було зазначено, для створення базової моделі було використано 4 датасети з найпопулярнішими емоціями. Спочатку було завантажено всі аудіофайли та об’єднано їх в один DataFrame. Частина датасетів підтягується з Hugging Face, частина – з Kaggle. Одразу ж для кожного запису фіксується шлях до файлу, відповідна емоція та назва датасету, звідки він узятий. Наприкінці перед об’єднанням застосовується попередня нормалізація аудіо та валідація метаданих для забезпечення якості даних. Приклад коду для завантаження й підготовки одного з цих наборів можна подивитися в лістингу 3.1.

Лістинг 3.1 – Приклад завантаження і збереження датасету RAVDESS

```
def load_ravdess(root_dir: str) -> pd.DataFrame:
 code2emo = {
     1: "neutral", 2: "calm", 3: "happy", 4: "sad",
     5: "angry", 6: "fearful", 7: "disgust", 8:"surprised"
 }
 rows: List[Dict] = []
 ravdess_dir = os.path.join(root_dir, "ravdess")
 for dirpath, _, files in os.walk(ravdess_dir):
     for fn in files:
         if fn.lower().endswith('.wav'):
             parts = fn.split('-')
             emo_code = int(parts[2])
             rows.append({
                 'path': os.path.join(dirpath, fn),
                     'emotion': code2emo.get(emo_code,
'unknown'),
                 'dataset': 'ravdess'
             })
 return pd.DataFrame(rows)
```

Після цього усі датафрейми об’єднуються в один, при чому усі записи у ньому перемішуються. Кінцевий результат продемонстровано у таблиці 3.1.

Таблиця 3.1 – Приклад даних у датафреймі, які були отримані після виконання усіх операцій

	Path	Emotion	Dataset
0	datasets/audio_dataset/ravdess/Actor_01/01-01-01-01-01-01-01.wav	neutral	ravdess
1	datasets/audio_dataset/ravdess/Actor_01/01-01-01-02-01-01-01.wav	calm	ravdess
2	datasets/audio_dataset/savee/DC_angry.wav	angry	savee
3	datasets/audio_dataset/savee/DC_fearful.wav	fearful	savee
4	datasets/audio_dataset/tess/George/happy/03.wav	happy	tess

На завершальному етапі підготовки даних здійснюється поділ датасету на тренувальну та тестову вибірки. Для забезпечення якісного навчання моделі обрано співвідношення 80/20, що дозволяє залишити достатню кількість даних як для навчання, так і для перевірки точності моделі. Після цього дані, представлені у форматі таблиць бібліотеки pandas, конвертуються у словникову структуру, яка відповідає вимогам форматів, прийнятих у більшості фреймворків машинного навчання.

3.3.2 Конфігурація аудіомоделі та екстрактора ознак

На наступному етапі розглядається побудова нейронної мережі для класифікації емоцій за голосом, а також механізм перетворення сирого аудіо у внутрішні репрезентації. Основою, як вже було зазначено у другому розділі, обрано архітектуру Wav2Vec 2.0.

Для початку формується перелік класифікаційних міток та відповідний їм мапінг, що можна подивитися на лістингу 3.2.

Лістинг 3.2 – Формування міток і виконання відповідного їм мапінгу

```
emotions = ["angry", "disgust", "fearful", "happy",
"neutral", "sad", "surprised"]
label2id = {e: i for i, e in enumerate(emotions)}
id2label = {i: e for e, i in label2id.items()}
```

В підсумку просто надається числове значення кожній емоції. Це необхідно для того, щоб модель однозначно зіставляла кожну емоційну категорію з цілим індексом і навпаки. Далі задаються архітектурні гіперпараметри за допомогою Wav2Vec2Config. Ці налаштування дозволяють адаптувати глибину та ширину моделі до специфіки обраного корпусу аудіоданих. Ознайомитися з усіма параметрами можна на лістингу 3.3.

Лістинг 3.3 – Архітектурні параметри задані у Wav2Vec2Config

```
config = Wav2Vec2Config(
    hidden_size=768,
    num_attention_heads=12,
    num_hidden_layers=6,
    vocab_size=1,
    problem_type="single_label_classification",
    num_labels=len(emotions),
    id2label=id2label,
    label2id=label2id,
)
```

На деяких заданих параметрах варто акцентувати увагу:

– hidden_size=768, num_attention_heads=12, num_hidden_layers=6 – визначають розмірність прихованого простору та складність трансформера;

– vocab_size=1 – оскільки в завданні немає текстового виводу, обсяг мовного словника зменшується до одиниці;

– problem_type="single_label_classification", num_labels=7 – конфігурують модель для мультикласового завдання.

І тепер, вже на базі цієї конфігурації, ініціалізується Wav2Vec2ForSequenceClassification. Тобто спочатку трансформер аналізує аудіосигнал і витягує з нього ключові характеристики, а потім один або кілька щільних (лінійних) шарів на їх основі обчислюють набір чисел – ймовірностей кожної емоції.

І вже для підготовки вхідних даних використовується Wav2Vec2FeatureExtractor, наведений у лістингу 3.4.

Лістинг 3.4 – Параметри Wav2Vec2FeatureExtractor необхідні для підготовки вхідних данних

```
feature_extractor = Wav2Vec2FeatureExtractor(
    feature_size=1,
    sampling_rate=16000,
```

Продовження лістингу 3.4

```
    padding_value=0.0,
    do_normalize=True,
)
```

Він спочатку приводить звук до єдиної частоти дискретизації 16 кГц, що забезпечує сталість тимчасової розгортки. Далі виконується нормалізація рівня амплітуд, щоб різні файли з різною гучністю стали порівнювані. І вже наприкінці короткі записи доповнюються нулями до необхідної довжини, а маска attention_mask вказує моделі, де закінчується справжній звук і починається штучне заповнення.

3.3.3 Налаштування і процес тренування моделі для аналізу базових емоцій

Після цього починається безпосереднє навчання. Для автоматизації циклу тренування та валідації буде використано клас Trainer з бібліотеки Transformers [30]. У конфігурації задаються такі параметри:

– per_device_train_batch_size та per_device_eval_batch_size – розмір батчу для тренування і перевірки;

– evaluation_strategy та eval_steps – визначають частоту перевірки якості на валідаційній вибірці;

– save_steps та save_total_limit – регулюють збереження контрольних точок моделі;

– num_train_epochs, learning_rate, weight_decay – основні гіперпараметри оптимізації.

Під час тренування модель проходить через кілька ключових кроків: спочатку батч даних передається в мережу (forward pass), де обчислюються вихідні логіти; після цього обчислюється функція втрат CrossEntropy Loss для порівняння передбачених логітів із справжніми мітками, що дозволяє

оцінити якість поточного передбачення; потім відбувається backward pass – поширення похибки назад по шарах мережі та оновлення її ваг за допомогою оптимізатора AdamW. І нарешті здійснюється логування метрик та збереження чекпоінтів моделі (checkpointing).

Після експериментів з налаштуванням навчання були знайдені найоптимальніші гіперпараметри:

– швидкість навчання (learning rate): 5e-05;

– розмір пакету оцінки і навчання (Evaluate and Train Batch Size): 2;

– кроки накопичення градієнта (Gradient Accumulation Steps): 5;

– загальний розмір пакету навчання (Total Train Batch Size): 10;

– оптимізатор: Adam з параметрами: betas=(0.9, 0.999) та epsilon=1e-08;

– кількість епох: 8.

Загалом модель навчалася впродовж 13 епох, але найкращий результат був отриманий на 8-ій епосі. Фінальна точність класифікації дорівнює 91%, що є дуже непоганим результатом. На функцію втрат впродовж навчання можна подивитися на рисунку 3.2.

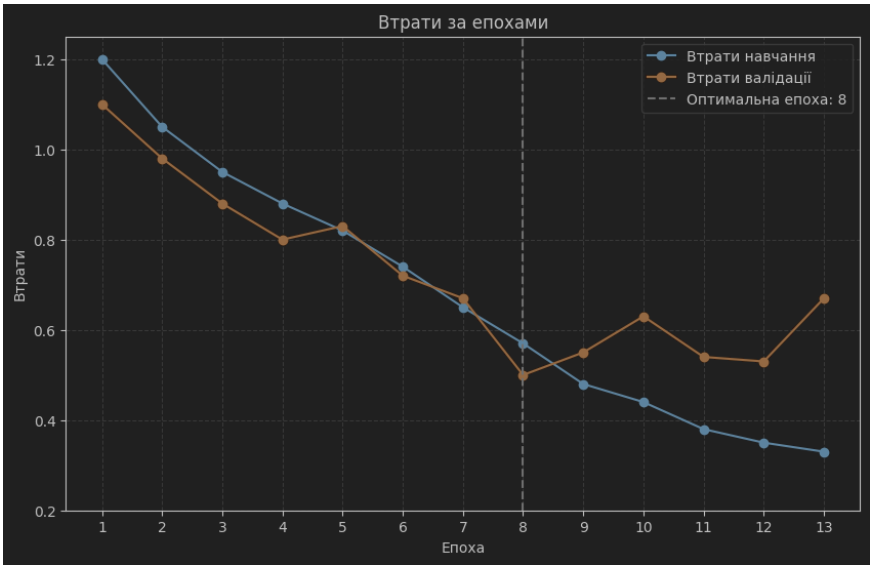


Рисунок 3.2 – Функція втрат на тренувальній і валідаційній виборках впродовж епох

А за зростанням точності можна спостерігати на рисунку 3.3.

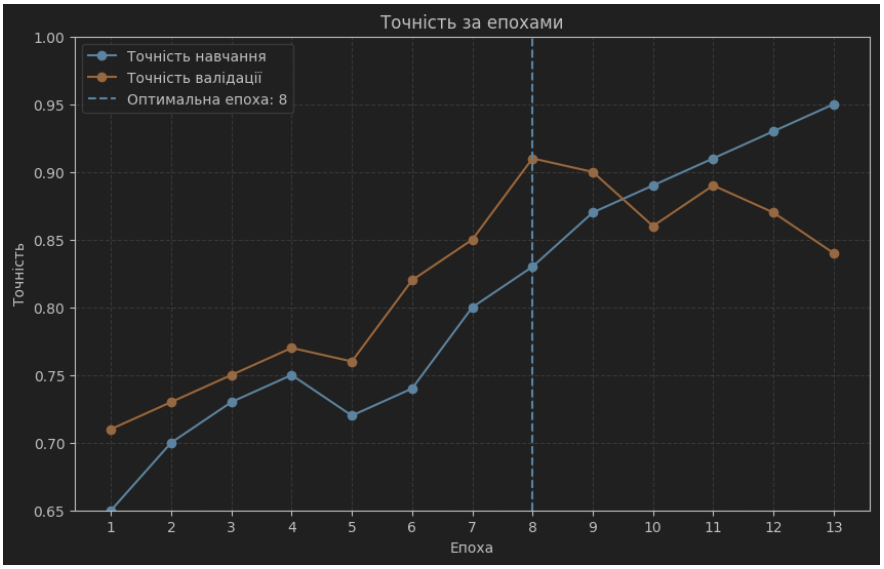


Рисунок 3.3 – Точність на тренувальній і валідаційній виборках впродовж епох

3.3.4 Аналіз роботи базової моделі в умовах інтерв'ю: оцінка рівня впевненості на прикладі відповідей

На цьому етапі було проведено прикладний аналіз роботи розробленої моделі в умовах симульованого інтерв'ю. Для цього було записано тестові аудіо-відповіді на типові запитання. Частина відповідей проговорювалася впевненим і позитивним тоном, інша – розгубленим та невпевненим. Також були створені записи, в яких кандидат демонструє невпевненість лише на початку, в середині або наприкінці відповіді. Після обробки аудіо модель формує ймовірнісний розподіл емоцій, що домінують у голосі, дозволяючи оцінити емоційний стан кандидата. Додатково результати аналізу візуалізуються у вигляді часових графіків емоційних змін протягом відповіді для спрощення інтерпретації та подальшого вдосконалення моделі. Отримані результати представлені в таблиці 3.2.

Таблиця 3.2 – Результати аналізу ймовірностей емоцій продовж відповіді на питання в інтерв'ю

Тип відповіді на записі	Ймовірності емоцій						
	Злість	Огида	Страх	Щастя	Нейтральність	Засмученість	Здивованість
Абсолютна впевненість	0.0012	0.0006	0.0015	0.9881	0.0017	0.0066	0.0004
Абсолютна невпевненість	0.0006	0.0008	0.0048	0.0023	0.0008	0.9903	0.0004
Приблизно рівна тривалість впевненості і невпевненості	0.0052	0.0055	0.1037	0.0193	0.0085	0.8545	0.0033
Невпевнений на початку	0.0032	0.0022	0.5175	0.0067	0.0154	0.0202	0.4348
Невпевнений в середині	0.0016	0.0007	0.016	0.0142	0.0061	0.003	0.9584
Невпевнений в кінці	0.0172	0.0075	0.4585	0.1252	0.1572	0.0259	0.2085

Помітно, що у випадках, коли кандидат демонструє чітко виражену впевненість або невпевненість, модель ефективно розпізнає емоційний стан і видає однозначні результати. Наприклад, на аудіозаписі з впевненою відповіддю найвищу ймовірність має емоція «щастя», тоді як на невпевненому відрізку – «засмученість». Водночас у складніших випадках, де впевненість змінюється протягом запису або виявляється змішана інтонація, моделі значно важче зробити однозначний висновок через перехідні стани та накладення різних емоційних сигналів.

Щоб краще проаналізувати динаміку емоційної реакції та врахувати такі перехідні моменти, аудіозапис буде поділятися на короткі сегменти тривалістю 2–3 секунди. Для кожного фрагмента автоматично обчислюватиметься домінуюча емоція разом із рівнем впевненості моделі в цьому висновку. Це дозволить відслідковувати зміну емоцій у реальному часі, виявляти інтервали невизначеності та будувати детальні часові криві емоційного стану. Результати такого покадрового аналізу наведено в таблиці 3.3.

Таблиця 3.3 – Результати аналізу кількості емоцій впродовж відповіді на питання в інтерв’ю

Тип відповіді на записі	Кількість емоцій						
	Злість	Огида	Страх	Щастя	Нейтральність	Засмученість	Здивованість
Абсолютна впевненість	2	0	4	10	8	1	0
Абсолютна невпевненість	0	0	9	1	1	22	0
Приблизно рівна тривалість впевненості і невпевненості	0	0	10	5	4	11	0
Невпевнений на початку	0	0	9	2	7	7	6
Невпевнений в середині	0	0	5	3	5	4	8
Невпевнений в кінці	0	0	9	3	4	4	5

Таким чином, було зроблено кілька важливих висновків. По-перше, емоції «злість» та «огида» практично не виявляються в жодному з типів відповідей, отже, доцільно виключити їх із подальшого аналізу. По-друге, детальний покадровий розбір дозволяє чітко відображати зміну емоцій протягом відповіді, що особливо важливо в контексті інтерв’ю, де рівень впевненості може змінюватися поступово або нерівномірно. По-третє, найчастіше домінують «нейтральність» та «засмученість» – особливо в разі абсолютної невпевненості та змінних станів, що вказує на потребу в додаткових практиках емоційного контролю під час підготовки до співбесіди. І, нарешті, такий підхід надає можливість не лише кількісно оцінити розподіл емоцій упродовж усієї відповіді, а й виявити критичні точки, в які кандидату слід звернути особливу увагу на власну інтонацію та невербальні сигнали. Зокрема, інтеграція цих даних із результатами зворотного зв’язку від реальних рекрутерів може покращити точність рекомендацій тренажера. А приклади такого розподілу можна побачити на рисунках 3.4–3.6.

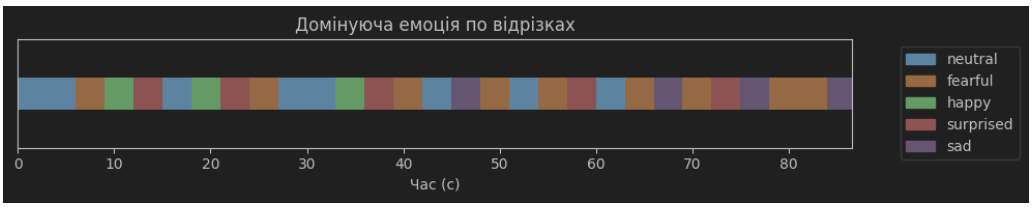


Рисунок 3.4 – Розподіл емоцій на аудіозаписі, де кандидат відповідав невпевнено у кінці запису

Рисунок 3.5 – Розподіл емоцій на аудіозаписі, де кандидат відповідав невпевнено у середині запису

Рисунок 3.6 – Розподіл емоцій на аудіозаписі, де кандидат відповідав невпевнено на початку запису

3.3.5 Донавчання моделі для лінійної класифікації емоцій у голосі

І заключний крок на цьому етапі – донавчання моделі для лінійної класифікації впевненості чи невпевненості впродовж відповіді на питання.

Після великої кількості експериментів, було знайдено найбільш оптимальні ваги для базових емоцій, які пов’язані з впевненістю чи невпевненістю, які можна переглянути у таблиці 3.4.

Таблиця 3.4 – Знайдені коефіцієнти впливу базових емоцій на рівень впевненості

Емоція	Позитивний коефіцієнт	Негативний коефіцієнт
Радість	0.68	0
Нейтральність	0.26	0
Здивованість	0.06	0
Засмученість	0	0.71
Страх	0	0.29

Залишається ключове питання: як донавчити модель так, щоб вона одразу видавала коректний рівень впевненості в інтервалі (0,1). Після експериментів із багатьма підходами було обрано метод, що показав стабільно хороші результати.

По-перше, умовно всі записи, де домінують емоції «радість» та «нейтральність», вважаються еталоном впевненості, а ті, де домінують «страх» та «розчарованість» – еталоном невпевненості. Для кожного такого фрагмента спочатку за формулою 3.1 обчислюється score, який відображає баланс «сигналів впевненості» й «сигналів невпевненості» на основі встановлених емпіричних ваг. Далі, відповідно до формули 3.2 цей score приводиться до інтервалу (0,1) за допомогою сигмоїдального перетворення – отримане значення sw використовується як sample weight в обчисленні loss-функції, тобто задає індивідуальну вагу кожного прикладу залежно від того, наскільки виразно він демонструє впевненість або невпевненість голосу.

$$\text{score} = \sum_{i \in E^+} w_i p_i - \sum_{j \in E^-} w_j p_j , \tag{3.1}$$

де E^+ – множина емоцій, що сигналізують про впевненість;

E^- – множина емоцій, що вказують на невпевненість;

w – це ваги позитивних і негативних емоцій відповідно;

p – це ймовірність емоції, отримана від мультикласового класифікатора емоцій.

$$sw = \sigma(\alpha \times score) = \frac{1}{1 + e^{-\alpha \times score}}, \quad (3.2)$$

де σ(x) – сигмоїдальна функція, яка гарантує, що $sw \in (0,1)$;

α – параметр steep, який керує «крутизною» переходу сигмоїдальної функції.

Після цього було інтегровано отримані проміжкові результати до функції втрат. Спочатку за формулою 3.3 для кожного екземпляру обчислюється бінарна крос-ентропійна функція втрат без редукції, для точного вимірювання розбіжності між прогнозом та міткою.

$$L_i = -\left[y_i \times \ln\sigma(l_i) + (1-y_i) \times \ln\ (1-\sigma(l_i))\right], \quad (3.3)$$

де L_i – функція втрат для і-го прикладу;

y_i – істинна позначка класу: 1 (впевнено) або 0 (невпевнено);

l_i – логіт, або вихід моделі до застосування сигмоїди;

$\sigma(l_i)$ – сигмоїдна функція, яка переводить логіт у ймовірність у межах (0, 1).

Тепер за формулою 3.4 отримана функція втрат помножується на попередньо обчислену вагу відповідного екземпляру даних, а отримані значення зважено усереднюються з урахуванням значущості кожного спостереження.

$$L_{batch} = \frac{\sum_i sw_i L_i}{\sum_i sw_i} \quad (3.4)$$

Після налаштування функції втрат таким чином, щоб «чіткі» приклади (з яскравим емоційним сигналом) більше впливали на оновлення ваг, а «неоднозначні» – менше, було розпочате навчання. При цьому в процесі тренування застосовувалися рання зупинка й автоматичне зниження темпу навчання за відсутності покращень на валідаційній вибірці. Для початку було завантажено вже навчену на базових емоціях модель і змінено її голову на одиночний логіт, що можна побачити в лістингу 3.5.

Лістинг 3.5 – Завантаження переднавченої моделі і заміна її голови

```
model = Wav2Vec2ForSequenceClassification.from_pretrained
("emotion_model/checkpoint-final")
model.classifier = nn.Sequential(
    nn.Dropout(0.1),
    nn.Linear(model.config.hidden_size, 1)
)
model.config.num_labels = 1
model.config.problem_type = "single_label_classification"
```

Після цього як і при навчанні базової моделі в результаті експериментів були визначені основні оптимальні гіперпараметри для навчання моделі:

– швидкість навчання (learning rate): 3e-05;

– розмір пакету навчання (Train Batch Size): 16;

– розмір пакету оцінки (Evaluate Batch Size): 16;

– регуляризація (weight_decay): 0.01;

– частота збереження (save_steps): 200;

– теплий старт (warmup_steps): 300;

– кількість епох: 17.

В результаті після всіх кроків було проведено навчання. За ходом навчання можна спостерігати на рисунках 3.7–3.8.

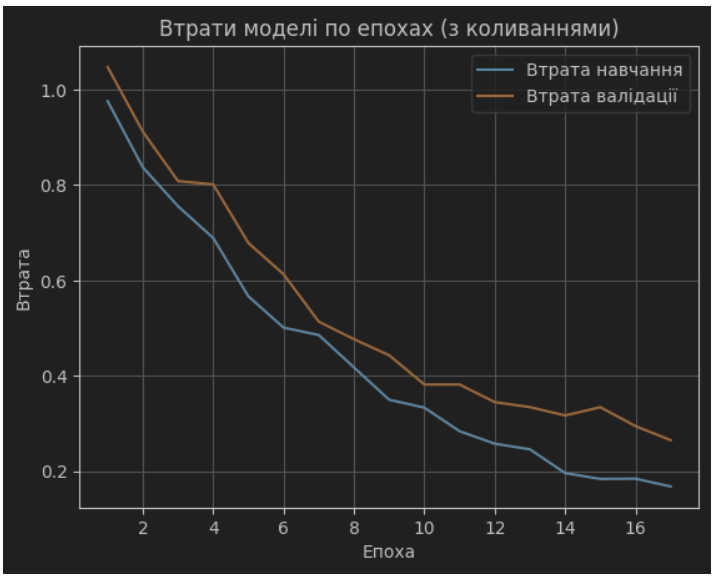


Рисунок 3.7 – Втрати на тренувальній і валідаційній виборах впродовж епох

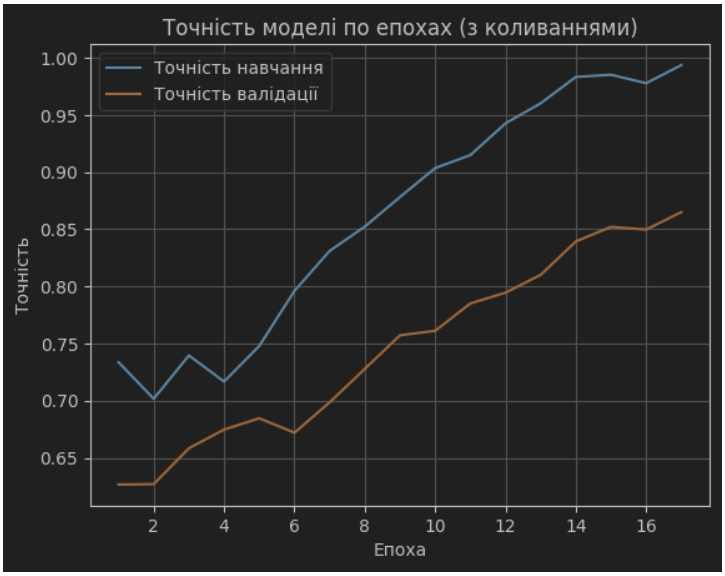


Рисунок 3.8 – Точність на тренувальній і валідаційній виборах впродовж епох

В результаті навчання була отримана точність 86% на валідаційній вибірці, що трохи гірше за модель, яка класифікувала базові емоції, та все ще доволі хороший результат.

3.3.6 Перевірка роботи аудіомоделі моделі на реальних прикладах

Після навчання моделі важливо перевірити її ефективність на реальних даних. Раніше для цього використовувалася модель для класифікації базових емоцій, де остаточний висновок вимагав залучення експерта для інтерпретації результатів: чи дійсно людина хвилюється, чи ні. Нова ж модель була донавчена спеціально для задачі оцінки впевненості кандидата, тому э можливість детально проаналізувати її поведінку на різних типах відповідей. Для прикладу було розглянуто запис із впевненою відповіддю кандидата (рисунок 3.9). На графіку показано зміну рівня впевненості протягом усієї відповіді.

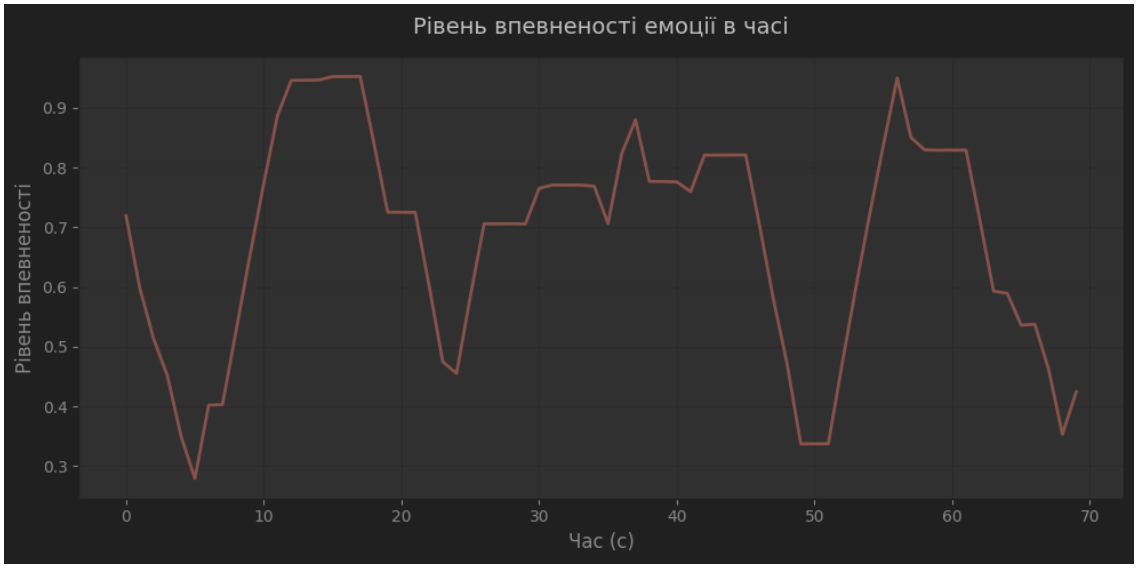


Рисунок 3.9 – Графік впевненості кандидата на записі із впевненою відповіддю

Як видно, модель досить точно відтворює коливання впевненості: хоча абсолютні значення можуть відрізнятися через природну невизначеність людської мови, загальна динаміка відповідає очікуванням. Цей приклад демонструє, що розроблений підхід дозволяє відстежувати рівень впевненості без постійного втручання експерта.

Більш детальний аналіз результатів із різними типами відповідей наведено в додатку А (рисунки А.1–А.5).

3.4 Реалізація моделі аналізу рівня впевненості на відео в ході інтерв'ю

Наступним завданням була реалізація моделі для аналізу впевненості, яку демонструє кандидат на відео (невербальні жести, міміка і т.д). В результаті досліджень у другому розділі для реалізації такої моделі було прийнято рішення використовувати переднавчену модель ResNet18 із донавчанням на обраному для цього датасеті.

3.4.1 Підготовка даних та навчання моделі класифікації рівня впевненості на відео

Перш за все було завантажено датасет і форматовано усі зображення у необхідний для навчання моделі формат. Для цього спочатку змінено колір зображення з сірого на RGB (Red, Green, Blue). Після чого перетворено розмір зображення з 48*48 пікселів на 224*224. Всі етапи перетворення можна подивитися на лістингу 3.6.

Лістинг 3.6 – Переформатування зображень датасету

```
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
img = self.transform(img)
```

Метод transform використовується для перетворення зображення на тензор та його нормалізації відповідно до стандартів ImageNet, на яких була навчена ResNet18. Для підвищення стійкості моделі додано базові аугментації – випадкове обрізання (RandomResizedCrop) і горизонтальне віддзеркалення (RandomHorizontalFlip). Після підготовки датасету його

було поділено на тренувальну й валідаційну вибірки у співвідношенні 80/20 із подальшим перемішуванням. Після цього було завантажено модель ResNet18 із попередньо навченими вагами та останній шар був змінений на лінійний із двома виходами – «впевнений» і «невпевнений».

Обрано функцію втрат CrossEntropyLoss, яка перетворює задачу класифікації на мінімізацію негативної лог-ймовірності правильного класу та забезпечує гладкий градієнт для ефективного навчання. Оптимізатор – Adam (Adaptive Moment Estimation) – адаптивно коригує швидкість навчання кожного параметра на основі першого та другого моментів градієнта, що пришвидшує збіжність і покращує стабільність моделі. Використано початковий темп навчання 1e-4 та batch_size=32, а також налаштовано раннє зупинення за відсутності покращень на валідації. Для додаткового контролю перенавчання застосовано регуляризацію Dropout і знижувальний шедулер темпу навчання, що допомогло зберегти стабільність процесу тренування. Після цього було запущено навчання, а його результати та динаміку втрат і точності можна проаналізувати на рисунках 3.10–3.11.

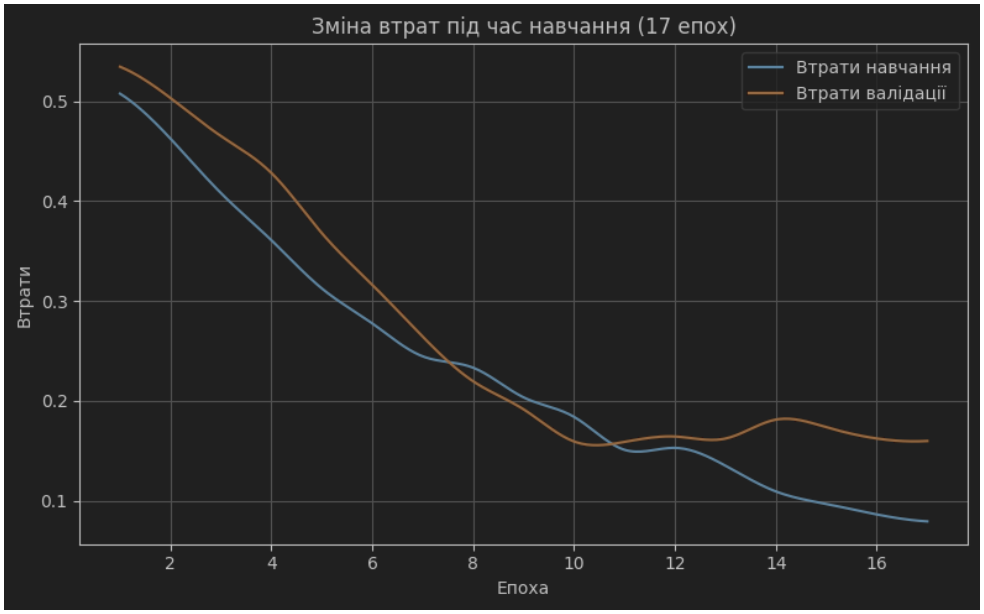


Рисунок 3.10 – Зменшення функції втрат впродовж навчання

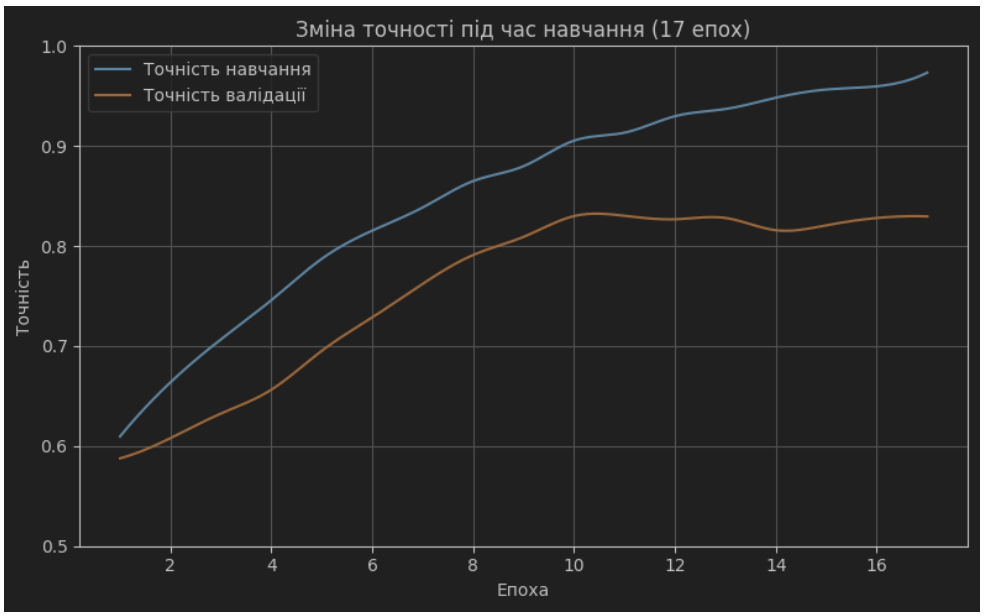


Рисунок 3.11 – Збільшення точності впродовж навчання

Після 17 епох навчання модель досягла максимальної точності 83,6 %, після чого подальше підвищення не спостерігалося.

3.4.2 Оцінка результатів роботи моделі на реальних прикладах

Далі було перевірено роботу моделі на реальних відео. Але спершу необхідно було підготувати вхідні дані. Оскільки мережа навчена працювати саме з обличчями, а не з цілими кадрами, спочатку відео було розбито на окремі кадри за допомогою OpenCV. Для балансу між точністю та швидкістю було обрано по три кадри з кожної секунди запису. Далі за допомогою методу face_cascade.detectMultiScale на цих кадрах було обрізано область обличчя і отримане зображення передано на вхід моделі, застосувавши до нього ті самі перетворення (масштабування, тензоризацію та нормалізацію), що й під час тренування.

На рисунку 3.12 наведено графік рівня впевненості, який модель видає для відео, де кандидат почувається впевнено протягом всієї відповіді. Інші

приклади аналізу записів відповідей продемонстровано у додатку А на рисунках А.6–А.9.

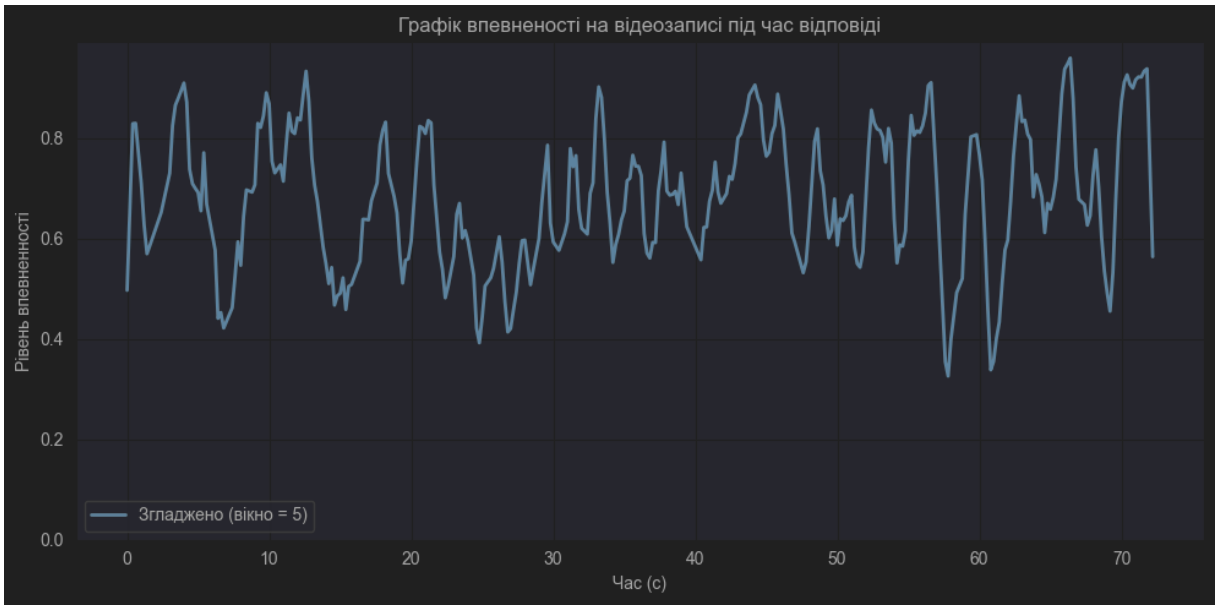


Рисунок 3.12 – Динаміка рівня впевненості на відео, де кандидат впевненно відповідає впродовж усього запису

Отже, модель успішно проаналізувала рівень впевненості на реальному прикладі, тож було прийнято рішення рухатися далі.

3.5 Реалізація серверної частини застосунку

Тепер, коли моделі навчені і готові до використання було розпочато написання серверної частини застосунку.

3.5.1 Визначення загальної архітектури бекенду застосунку

У центрі серверної частини буде знаходитися FastAPI-сервер, який відповідатиме за прийом запитів з фронтенда, делегування обробки відповідним модулям та формування фінального результату. Коли

користувач запитуватиме генерацію списку питань, FastAPI буде викликати функцію get_questions(), яка звертатиметься до зовнішнього API Gemini через пакет google.generativeai і буде повертати відформатований набір запитань. Для завантаження відповіді відео користувач буде надсилати WebM-файл на ендпоінт /upload_answer/{q_id}; цей запит буде оброблятися асинхронно за допомогою BackgroundTasks. У фоновому процесі функція process_upload послідовно конвертуватиме WebM у MP4 та WAV, буде виконувати транскрипцію аудіодоріжки через Google Speech API, оцінюватиме текстову відповідь за допомогою Gemini, а також буде проводити аналіз відео і аудіо за допомогою модулів predict_video та get_audio_confidence_curve.

Після завершення обчислень медіа-файли будуть завантажуватися у Google Cloud Storage за допомогою функції upload_to_gcs, а результати аналізу разом із метаданими фіксуватимуться у Firestore через save_metadata. Локальний словник status_store буде служити простим механізмом індикації прогресу обробки, до якого фронтенд звертаєтиметься через ендпоінт /status/{q_id}. У кінці, коли обробка повністю завершена, користувач отримуватиме сформовану сторінку з результатами на маршруті /feedback. А за маршрутом /history користувач зможе подивитися на історію записів своїх відповідей.

3.5.2 Створення і налаштування ендпоінтів застосунку

Після того як було визначено загальну архітектуру серверної частини застосунку було розпочато реалізацію першої частини – створення ендпоінтів. Загалом у застосунку їх буде 7:

– «GET /» – Повертатиме головну HTML-сторінку з інтерфейсом для запуску сесії (введення назви посади, складності тощо);

– «POST /get_questions» – Прийматиме форму з полями job_title, difficulty, additional_instructions, викликатиме get_questions() для звернення до Gemini, отримує три запитання та віддаватиме їх у JSON;

– «POST /upload_answer/{q_id}» – Прийматиме WebM-файл з відповіддю кандидата й метадані (job_title, question), одразу зберігатиме статус у status_store, відправлятиме process_upload у фон (BackgroundTasks) і буле повертати { "status": "processing" };

– «GET /status/{q_id}» – Перевірятиме вміст status_store[q_id] і буде повертати поля recorded, analyzed, video_analysis, audio_curve, щоб фронтенд міг показувати прогрес;

– «GET /feedback/{session_label}» – Збиратиме всі відповіді з Firestore для вказаної сесії, очищуватиме «raw» JSON відповіді від Gemini, і віддаватиме відформатовану HTML-сторінку з транскриптами, оцінками, графіками впевненості та посиланнями на збережені медіа;

– «GET /history» – Буде повертати перелік всіх сесій із пагінацією: для кожної – session_label, job_title, difficulty, кількість відповідей та дату створення;

– «GET /history/{session_label}» – Автоматично перенаправлятиме на сторінку /feedback/{session_label}.

Для легшого розуміння на рисунку 3.13 продемонстровано як ендпоінти виглядають у FastAPI docs.

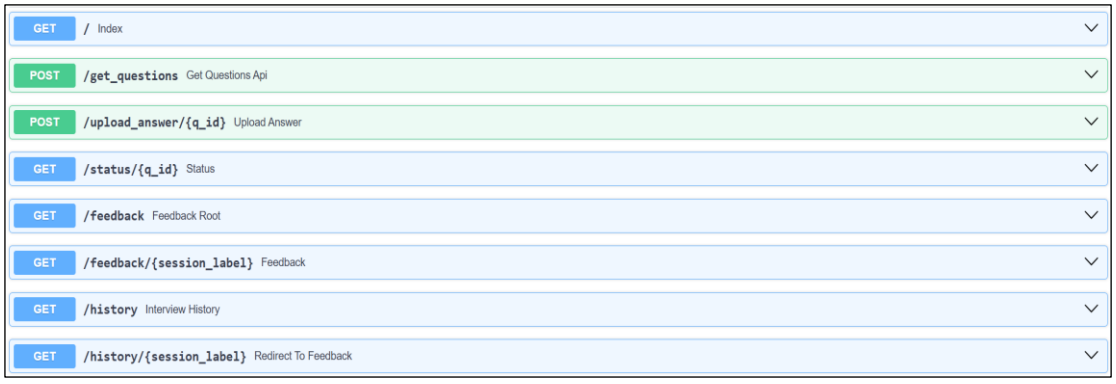


Рисунок 3.13 – Структура ендпоінтів застосунку у FastAPI docs

3.5.3 Використання Gemini для генерації та оцінки запитань

Першим кроком після створення та налаштування ендпоінтів є організація генерації питань, а також створення функції для подальшого аналізу текстової транскрипції відповіді користувача. Для реалізації обох задач використовувалась велика мовна модель Gemini. Взаємодія з нею реалізована через модуль question_creator.py, який спочатку налаштовує клієнтську бібліотеку google.generativeai за допомогою виклику genai.configure(api_key), що забезпечує безпечний доступ до API.

Функція get_questions формує контекстний запит на основі заданої професії, рівня складності та додаткових інструкцій (якщо вони вказані), після чого надсилає цей запит до моделі. Оскільки у відповідь Gemini повертає текстовий блок із нумерованими питаннями, для перетворення його у структурований список застосовується допоміжна функція question_splitter. Вона, використовуючи регулярні вирази, коректно розбиває текст на окремі питання, зберігаючи оригінальне форматування й запобігаючи втраті змісту навіть при складній синтаксичній побудові.

Для прикладу, під час генерації питань для позиції «Middle Data Scientist» із вказівкою «зосередитися на питаннях, пов’язаних із візуалізацією», модель сформувала такі запитання:

– Describe your process for selecting the most appropriate visualization type when working with a new dataset and a specific business objective. What common pitfalls related to misleading or biased visualizations do you actively try to avoid, especially when presenting findings to non-technical stakeholders?

– Tell me about a project where data visualization played a crucial role in uncovering a key insight or effectively communicating complex results. How did you ensure the visualization was both statistically accurate and easy for your intended audience to understand and act upon?

– Imagine you’ve created a detailed dashboard, but a key decision-maker is misinterpreting a specific chart or missing the main takeaway. How would you

approach explaining the visualization and the underlying data story to clarify the insight and build confidence in the findings?

Наступним логічним етапом є оцінювання відповіді користувача за допомогою функції evaluate_answer, яка приймає текст питання, назву посади та транскрибовану відповідь. Усередині функції формується промпт, який інструктує модель оцінити відповідь за кількома критеріями: точність, повнота, ясність і глибина, а також надати еталонну відповідь на питання. У відповідь Gemini повертає результат у форматі JSON, однак іноді ця відповідь приходить у вигляді текстової обгортки з елементами форматування. У таких випадках виконується попереднє очищення – видаляються зайві символи, після чого за допомогою регулярних виразів з тексту витягується чистий JSON-об’єкт. Лише після цього з отриманих даних формується структурований словник, який містить числові оцінки, текстові поради для покращення відповіді і приклад зразкової відповіді.

Ось приклад такої оцінки: при відповіді на питання «As a junior analyst, you might be asked to present your findings to people who don't have a technical background. How would you ensure they understand your analysis and its implications?» модель видала такий аналіз:

«The candidate identifies several crucial techniques for presenting data to non-technical audiences, such as focusing on the 'so what', using simple visuals, avoiding jargon, and engaging the audience. This shows a good grasp of the necessary principles. To improve, concentrate on structuring your points clearly and practicing articulation to ensure your message is easily understood and impactful».

А приклад зразкової відповіді виглядає так:

«I would start by framing the problem and its importance in everyday terms, avoiding technical jargon and focusing on the business context. To make complex data more accessible, I would use clear visuals such as simple charts or infographics, accompanied by plain-language captions that highlight the key takeaway. When explaining my methods, I would use analogies or relatable

examples – like comparing a data trend to a rising sales curve – to help listeners grasp abstract concepts. I would summarize the main findings first, then walk through the supporting evidence step by step, pausing to check for questions or confusion. Throughout the presentation, I would emphasize the real-world implications of the analysis, such as potential cost savings or growth opportunities, rather than dwelling on statistical details. After the talk, I would provide a succinct one-page summary that stakeholders can refer to, ensuring they have a clear record of the recommendations. Finally, I would invite ongoing feedback and offer to clarify any points, reinforcing that open dialogue is welcome and that I’m available to support their decision-making process.»

Узагальнюючи, можна сказати, що ця частина системи виконує одразу дві ключові функції: забезпечує гнучке й релевантне формування питань залежно від позиції кандидата та контексту інтерв’ю, а також дозволяє швидко і якісно оцінити його відповідь за допомогою інтеграції з потужною мовною моделлю. Цей модуль фактично задає інтелектуальний каркас усієї оцінки, на який надалі нашаровуються результати аудіо- та відеоаналізу.

Готовий фундамент інтелектуальної взаємодії з кандидатом створює умови для переходу до наступного етапу – обробки відповіді, агрегування всіх результатів та інтеграції з хмарними сервісами, що і буде розглянуто в наступному розділі.

3.5.4 Обробка відповіді та інтеграція з хмарними сервісами

У момент отримання від клієнта відеофайлу у форматі WebM сервер запускає єдиний конвеєр обробки відповідей, реалізований у методі process_upload. Спершу оригінальний WebM зберігається локально і конвертується за допомогою ffmpeg у формат MP4, який краще підходить для подальшого відтворення та аналізу, а також у WAV – для зручнішої обробки аудіо. Далі звуковий файл передається до сервісу Google Speech API через функцію transform_to_speech, де виконується автоматична

транскрипція мовлення в текст. Отриманий результат надсилається в модуль evaluate_answer, який за допомогою моделі Gemini формує структуровану відповідь у вигляді оцінок і рекомендацій.

Паралельно з цим відбувається аналіз рівня впевненості в голосі. Модуль get_audio_confidence_curve розбиває аудіосигнал на короткі фрагменти, визначає для кожного ймовірні емоційні стани, а потім з урахуванням ваг емоційних категорій формує плавну криву рівня впевненості.

У цей же час здійснюється обробка відео: модуль predict_video вибірково відбирає кадри через рівні інтервали часу, передає їх у попередньо донавчену нейронну мережу ResNet18, яка класифікує емоційний чи поведінковий стан кандидата. На основі цих прогнозів формуються часові мітки з відповідними оцінками.

Після завершення обох гілок аналізу формується єдиний словник, що містить аудіо- й відеоряди впевненості, середні значення показників та загальний висновок про рівень впевненості кандидата. Крім того, реалізовано централізоване логування та моніторинг виконання конвеєра для швидкого виявлення збоїв і відстеження продуктивності.

Після обробки всі артефакти і результати передаються до хмарного середовища. За допомогою клієнта Google Cloud Storage, обгорнутого в утиліту upload_to_gcs, отримані медіафайли зберігаються у відповідному бакеті, після чого сервер формує публічні URL для кожного ресурсу. Ці посилання разом із транскриптами, результатами оцінки від Gemini, часовими рядами впевненості та супутніми метаданими передаються до функції save_metadata, яка створює документ у Firestore за шляхом sessions/{session_label}/responses.

Таким чином, у Google Cloud зберігаються лише сирі відео- та аудіофайли, тоді як у Firestore – готові результати аналізу в зручному структурованому форматі. Цю структуру можна побачити на рисунку 3.14.

Рисунок 3.14 – Структура збережених результатів аналізу у Firestore

Завдяки такій інтеграції всі дані надійно зберігаються в хмарі, забезпечуючи швидкий та уніфікований доступ до результатів аналізу на стороні клієнта або для наступних етапів обробки.

3.5.5 Безпека, масштабування та висновок

У завершальній частині опису серверної архітектури було приділено особливу увагу захисту даних та готовності системи до зростання навантаження. Щоб уникнути небажаного доступу, взаємодія з Google Cloud Storage і Firestore відбувається через спеціальні сервісні облікові записи з мінімально необхідними правами, а налаштування CORS у тестовому середовищі потім звужується до конкретних довірених доменів у продакшені. FastAPI плюс фонова обробка через BackgroundTasks забезпечують неблокуюче виконання складних обчислень і водночас

дозволяють легко масштабувати рішення горизонтально, наприклад, розгортаючи кілька реплік у Kubernetes або іншому середовищі оркестрації.

Таким чином, було побудовано чіткий і прозорий pipeline, що простежується від моменту надходження HTTP-запиту до формування повноцінного фідбеку для користувача.

Взаємодія з великими мовними моделями Gemini у поєднанні з кастомними ML-модулями для відео- та аудіоаналізу дає змогу отримувати високоякісні рекомендації й об’єктивні оцінки відповіді. При цьому медіафайли зберігаються у GCS як «сирі» об’єкти, а у Firestore акумулюються вже оброблені дані – транскрипти, часові ряди впевненості, бали й текстові поради.

Така архітектура поєднує надійність і гнучкість, дозволяючи оперативно реагувати на зростання навантаження й безболісно інтегрувати нові модулі без суттєвого втручання в базовий код. У підсумку реалізована серверна частина забезпечує безпечний, масштабований і адаптивний тренажер для інтелектуального інтерв’ювання кандидатів.

3.6 Розробка клієнтської архітектури та реалізація інтерфейсу користувача

3.6.1 Архітектурна організація та структура файлів

Клієнтська частина організована в рамках невеликого, але водночас гнучкого набору файлів. Файл index.html виступає відправною точкою – він містить статичну розмітку і підключає стилі (style.css) та бізнес-логіку (app.js). При побудові інтерфейсу ми спираємося на семантичні елементи HTML5: input для введення професії, combobox для вибору рівня складності та textarea для додаткових інструкцій. Завдяки чіткому розподілу відповідальностей – розмітка в HTML, стилі в CSS та динамічна поведінка в JavaScript – код легко підтримувати і розширювати.

Каскадні стилі у файлі style.css забезпечують чистий та консистентний вигляд, відокремлюючи візуальне оформлення від логіки. Тут визначені базові правила для контейнера, таблиць, кнопок і круглих індикаторів статусу, що дозволяє швидко змінювати дизайн без правки Js.

Нарешті, app.js організовує зв’язок із бекендом: усі запити формуються методом fetch, отримані дані кешуються в localStorage, а при завантаженні сторінки відновлюють останній стан інтерфейсу.

3.6.2 Інтерактивне формування запитань і запис відповідей

Після введення назви професії та натискання кнопки «Get Questions» скрипт надсилає форму на ендпоінт /get_questions і отримує масив рядків із текстом запитань. Ці рядки динамічно перетворюються на табличні рядки або картки – кожне питання супроводжується двома індикаторами статусу: «recorded» і «analyzed». При натисканні на кнопку відповіді у відповідному рядку відбувається запит дозволу на доступ до камери та мікрофона, а потім запускається MediaRecorder. Під час запису індикатор змінює свій колір на жовтий, а після зупинки автоматично перетворюється на зелений і ініціює відправку отриманого WebM-файла на бекенд.

Ключова перевага такої реалізації – відсутність сторонніх бібліотек для запису медіа: стандартний API браузера дозволяє гнучко керувати розміром фрагментів, типом контенту та тривалістю запису. Паралельно з відправкою файлу у фоновому режимі запускається функція опитування {/status/{q_id}}, яка кожні кілька секунд перевіряє готовність результатів і візуально оновлює індикатор «analyzed». На випадок помилки при завантаженні чи обробці даних користувачу виводиться зрозуміле повідомлення з рекомендацією перевірити з’єднання та права доступу.

Інтерфейс головної сторінки застосунку продемонстровано на рисунку 3.15.

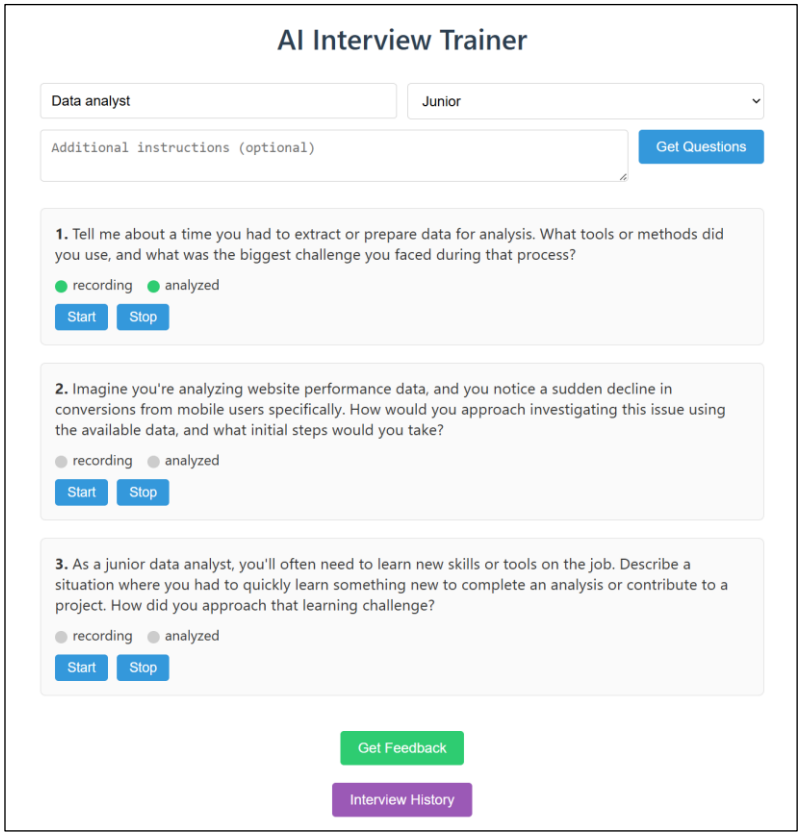


Рисунок 3.15 – Інтерфейс головної сторінки вебзастосунку

3.6.3 Аналіз результатів, історія інтерв'ю та візуалізація даних

Як тільки бекенд завершує обробку відео, аудіо та ML-аналізу і повертає клієнту analyzed: true, відповідний індикатор миттєво змінює свій колір на зелений, а користувач отримує доступ до докладного фідбеку.

Сторінка feedback.html підключає Chart.js, що дозволяє побудувати дві лінійні діаграми: одна відображає динаміку рівня впевненості в голосі з аудіокривої (audio_curve.times та audio_curve.confidences), інша – рівень впевненості на відео (video_analysis.timestamps і video_analysis.confidence_scores). Поряд із графіками вгорі сторінки розміщуються компактні картки з метриками точності, повноти, ясності і глибини, а під ними у відокремленому блоці виводяться текстові коментарі та поради від Gemini. Картку з аналізом транскрипції відповіді користувача

продемонстровано на рисунку 3.16, а графік впевненості у голосі (майже однаковий з відео) зображено на рисунку 3.17.

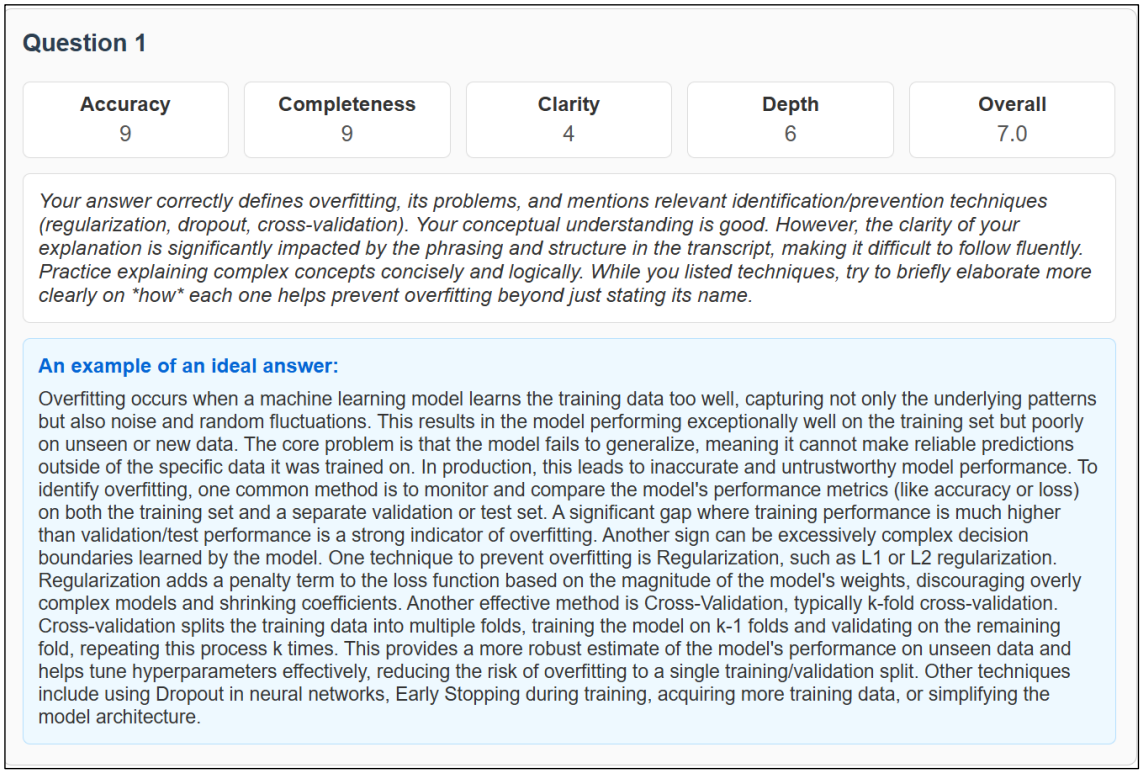


Рисунок 3.16 – Картка аналізу транскрипції відповіді кандидата

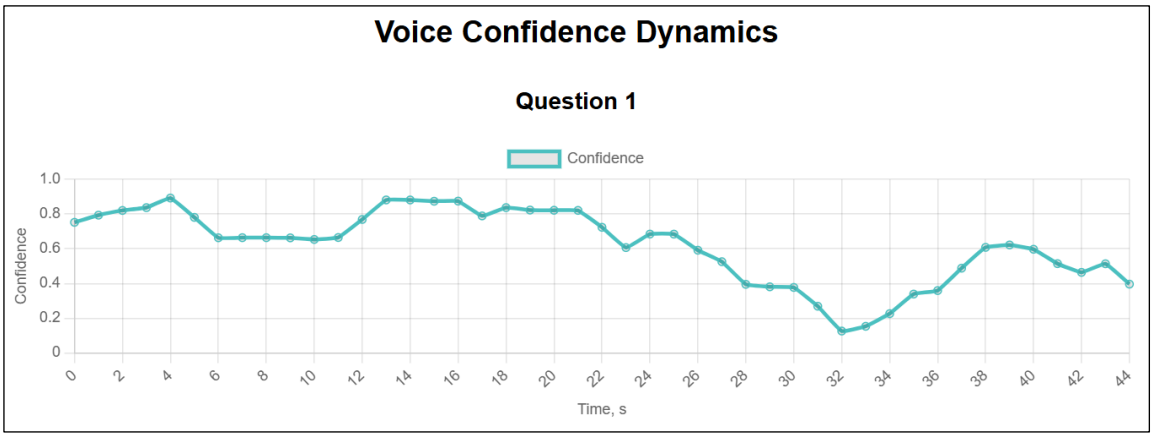
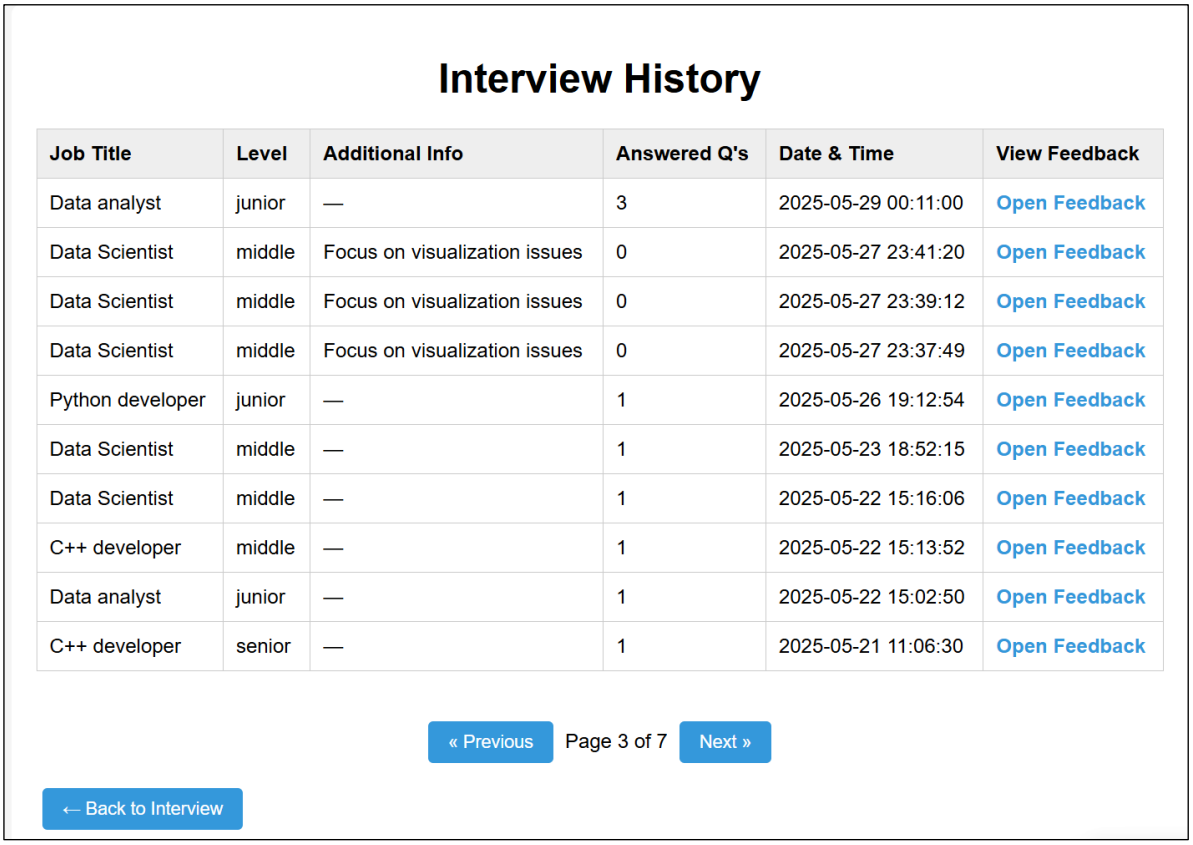


Рисунок 3.17 – Графік впевненості голосу кандидата

Крім фідбеку за поточну відповідь, користувач може звернутися до історії всіх проведених інтерв’ю. Сторінка history.html запитує у бекенда

перелік сесій із Firestore та формує табличний звіт, де для кожної сесії наведені назва позиції, рівень складності, кількість опрацьованих запитань і час створення. Для таблиці реалізована проста пагінація, що дозволяє перегортати довгий список історії, а в останньому стовпчику кожного рядка розміщено посилання «Open Feedback», яке веде безпосередньо на докладний звіт конкретної сесії. Для зручності користувача є опція вибору кількості записів на одній сторінці, що допомагає швидко орієнтуватися в обсязі даних. Можна подивитися на інтерфейс історії інтерв'ю на рисунку 3.18.



Interview History

Job Title	Level	Additional Info	Answered Q's	Date & Time	View Feedback
Data analyst	junior	—	3	2025-05-29 00:11:00	Open Feedback
Data Scientist	middle	Focus on visualization issues	0	2025-05-27 23:41:20	Open Feedback
Data Scientist	middle	Focus on visualization issues	0	2025-05-27 23:39:12	Open Feedback
Data Scientist	middle	Focus on visualization issues	0	2025-05-27 23:37:49	Open Feedback
Python developer	junior	—	1	2025-05-26 19:12:54	Open Feedback
Data Scientist	middle	—	1	2025-05-23 18:52:15	Open Feedback
Data Scientist	middle	—	1	2025-05-22 15:16:06	Open Feedback
C++ developer	middle	—	1	2025-05-22 15:13:52	Open Feedback
Data analyst	junior	—	1	2025-05-22 15:02:50	Open Feedback
C++ developer	senior	—	1	2025-05-21 11:06:30	Open Feedback

« Previous Page 3 of 7 Next »

← Back to Interview

Рисунок 3.18 – Інтерфейс сторінки з історіями пройдених інтерв'ю

У підсумку реалізована клієнтська частина поєднує простоту й ефективність: розподіл коду на верстку, стилі та скрипти забезпечує чітку і зручну модульність, а використання вбудованих браузерних API зводить залежності до мінімуму. Інтерактивна робота із запитаннями й відповідями,

побудована на динамічному рендерінгу карток та управлінні MediaRecorder, гарантує зручність запису та своєчасне оновлення статусів. Завершальні етапи – візуалізація результатів через Chart.js і модуль архівування з історією інтерв’ю – доповнюють користувацький шлях, дозволяючи миттєво оцінити продуктивність і повернутися до попередніх сесій. Крім того, дані на сторінці оновлюються без повного перезавантаження завдяки застосуванню AJAX-запитів, що покращує швидкодію й комфорт роботи. Загалом такий фронтенд працює швидко, гнучко й прозоро, що дозволяє зручно проходити тренувальне інтерв’ю в вебзастосунку.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи створено комплексний вебзастосунок для підготовки кандидатів до технічних інтерв'ю, який поєднує сучасні підходи машинного навчання, хмарні сервіси та зручний інтерфейс користувача.

Перш за все, успішно реалізовано дві ключові моделі аналізу впевненості кандидата: аудіомодель на основі аналізу базових емоцій, а також відеомодель, донавчену на ResNet18 для розпізнавання впевненості за виразом обличчя. Кожна з них пройшла етапи підготовки даних, тренування та валідації на реальних прикладах, що підтвердило їхню здатність оцінювати рівень упевненості з високою точністю у контексті інтерв'ю.

Другою важливою складовою є серверна частина, реалізована на FastAPI. Вона забезпечує стабільну маршрутизацію, інтеграцію з мовною моделлю Gemini для генерації та оцінки запитань, а також послідовний конвеєр обробки: конвертацію медіафайлів, транскрипцію через Google Speech API, аналіз даних, збереження сирих відео та аудіо у Google Cloud Storage і метаданих у Firestore.

Третім досягненням стало створення легкого, але функціонального клієнтського інтерфейсу, що не потребує великих фреймворків. Завдяки розділенню на HTML, CSS та JavaScript, використанню вбудованих API браузера для запису медіа й взаємодії з бекендом, а також Chart.js для візуалізації результатів, система забезпечує безшовний досвід: від генерації питань до перегляду інтерактивних графіків і історії інтерв'ю.

У ході роботи отримано нові прикладні результати: опис методики побудови кривої впевненості із поєднанням ваг позитивних і негативних емоцій, алгоритм обробки варіативних відповідей Gemini, а також практика інтеграції розподілених ML-сервісів із сучасними хмарними сховищами. Ці

напрацювання можуть бути корисні в подальших дослідженнях інтерпретуємості моделей та автоматизації оцінки як soft, так і tech skills.

Рекомендації для подальшого розвитку проєкту включають розширення функціоналу інтерфейсу шляхом додавання підтримки мобільних пристроїв, локалізації інтерфейсу та реалізації особистого кабінету користувача з можливістю реєстрації й управління профілем. Це дозволить забезпечити індивідуальний підхід до кожного кандидата, зберігати історію його результатів та поступово адаптувати параметри моделей під стиль спілкування конкретного користувача. Крім того, доцільно вдосконалити моделі аналізу, інтегрувавши багатоканальний розбір міміки та жестів, а також застосувавши контекстуальні NLP-методи для глибинного розуміння змісту відповідей. Оптимізація продуктивності може передбачати обробку потокового відео в режимі реального часу та розгортання мікросервісної архітектури для горизонтального масштабування.

Отримані результати можуть стати основою для практичних робіт із веброзробки та машинного навчання, а також слугувати фундаментом для створення корпоративних тренінгових платформ у сфері HR-технологій, де персоналізований підхід і аналітика на кожному етапі інтерв’ю є критично важливими.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Global talent trends. *Business Solutions on LinkedIn I LinkedIn*. URL: https://business.linkedin.com/talent-solutions/global-talent-trends (дата звернення: 19.05.2025).

2. Can ChatGPT Challenge the Scientific Impact of Published Research, Particularly in the Context of Industry 4.0 and Smart Manufacturing? / V. Terziyan та ін. *Procedia Computer Science*. 2024. Т. 232. С. 2540–2550. URL: https://doi.org/10.1016/j.procs.2024.02.072 (дата звернення: 19.05.2025).

3. Chamorro-Premuzic T. How to use GenAI to prepare for your next job interview. *Harvard Business Review*. URL: https://hbr.org/2024/01/how-to-use-genai-to-prepare-for-your-next-job-interview (дата звернення: 20.05.2025).

4. Department for Science, Innovation and Technology. Responsible AI in recruitment. *GOV.UK*. URL: https://www.gov.uk/government/publications/responsible-ai-in-recruitment-guide/responsible-ai-in-recruitment (дата звернення: 20.05.2025).

5. Job interview training platform. *#1 Job Interview Training Platform (1,000,000+ users)*. URL: https://www.biginterview.com/ (дата звернення: 20.05.2025).

6. ParakeetAI. *ParakeetAI*. URL: https://www.parakeet-ai.com/ (дата звернення: 20.05.2025).

7. Interviews by AI | AI-powered interview preparation. Interviews by AI | AI-powered Interview Preparation. URL: https://interviewsby.ai/ (дата звернення: 22.05.2025).

8. Build Your Public Speaking Confidence. *Nonprofit Communications Report*. 2018. Т. 17, № 1. С. 3. URL: https://doi.org/10.1002/npcr.31094 (дата звернення: 22.05.2025).

9. Confidence Based Acoustic Event Detection / X. Xia та ін. *ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal*

Processing (ICASSP), м. Calgary, AB, 15–20 квіт. 2018 р. 2018. URL: https://doi.org/10.1109/icassp.2018.8461845 (дата звернення: 23.05.2025).

10. Papers with Code - RAVDESS Dataset. *The latest in Machine Learning | Papers With Code*. URL: https://paperswithcode.com/dataset/ravdess (дата звернення: 23.05.2025).

11. Papers with Code - SAVEE Dataset. *The latest in Machine Learning | Papers With Code*. URL: https://paperswithcode.com/dataset/savee (дата звернення: 23.05.2025).

12. AbstractTTS/TESS · Datasets at Hugging Face. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/datasets/AbstractTTS/TESS (дата звернення: 23.05.2025).

13. ReySajju742/urdu-language-speech-dataset · Datasets at Hugging Face. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/datasets/ReySajju742/urdu-language-speech-dataset (дата звернення: 24.05.2025).

14. Wav2Vec2. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/docs/transformers/model_doc/wav2vec2 (да та звернення: 24.05.2025).

15. Bangla Handwritten Character Recognition using Convolutional Neural Network / M. M. Rahman та ін. *International Journal of Image, Graphics and Signal Processing*. 2015. Т. 7, № 8. С. 42–49. URL: https://doi.org/10.5815/ijigsp.2015.08.05 (дата звернення: 02.06.2025).

16. JetBrains. PyCharm: The only Python IDE you need. JetBrains. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 24.05.2025).

17. JetBrains. PyCharm Features - JetBrains python IDE. JetBrains. URL: https://www.jetbrains.com/pycharm/features/ (дата звернення: 24.05.2025).

18. The python tutorial. Python documentation. URL: https://docs.python.org/3/tutorial/index.html (дата звернення: 24.05.2025).

19. Ascher D., Lutz M. Learning Python, Second Edition. O'Reilly Media, Inc., 2003. 552 с.

20. Fluent Python: Clear, Concise, and Effective Programming / ред.: M. Blanchette, R. Roumeliotis. O'Reilly Media, 2015. 792 с.

21. TensorFlow. *TensorFlow*. URL: https://www.tensorflow.org/ (дата звернення: 25.04.2025).

22. Keras: Deep Learning for humans. *Keras: Deep Learning for humans*. URL: https://keras.io/ (дата звернення: 25.04.2025).

23. Scikit-learn: machine learning in Python. *scikit-learn: machine learning in Python – scikit-learn 0.16.1 documentation*. URL: https://scikit-learn.org/stable/ (дата звернення: 26.04.2025).

24. OpenCV: Introduction to OpenCV-Python Tutorials. *OpenCV documentation index*. URL: https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html (дата звернення: 26.04.2025).

25. Librosa – librosa 0.11.0 documentation. *Librosa*. URL: https://librosa.org/doc/latest/index.html (дата звернення: 26.04.2025).

26. Google cloud platform in action. Manning Publications, 2018. 632 с.

27. Lakshmanan V. Data Science on the Google Cloud Platform : Implementing End-To-End Real-Time Data Pipelines: From Ingest to Machine Learning. O'Reilly Media, Incorporated, 2022.

28. Speech-to-Text AI: speech recognition and transcription. *Google Cloud*. URL: https://cloud.google.com/speech-to-text?hl=en (дата звернення: 30.04.2025).

29. March18/FacialConfidence · datasets at hugging face. *Hugging Face – The AI community building the future*. URL: https://huggingface.co/datasets/march18/FacialConfidence (дата звернення: 01.05.2025).

30. Trainer. *Hugging Face – The AI community building the future*. URL: https://huggingface.co/docs/transformers/main_classes/trainer (дата звернення: 01.06.2025).