Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка програмного забезпечення з використанням фреймворку Dspy для інтелектуальної системи підготовки до співбесід в ІТ-галузі»

Виконав: студент IV курсу, групи СП-41
спеціальності 121
«Інженерія програмного забезпечення»
(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) Стоянов Ю.М. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Петрик М.Р.
(підпис) (прізвище та ініціали)
« 6 » квітня 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Бурилу Вадиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення з використанням фреймворку Dspy для інтелектуальної системи підготовки до співбесід в ІТ-галузі

Керівник роботи Стоянов Юрій Миколайович, кандидат технічних наук, доцент кафедри ПІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 06 » квітня 2026 року №

2. Термін подання студентом завершеної роботи 22.06.2026

3. Вихідні дані до роботи вирішено проблему процесу підготовки фахівців ІТ-галузі до технічних співбесід

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
Аналіз предметної області та постановка задачі
Теоретичні основи та обґрунтування вибору технологій
Проєктування та реалізація програмної системи
Безпека життєдіяльності та основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність, мета, задачі дослідження
3. Існуючі технології реалізації подібних систем.
4. Функціональні та нефункціональні вимоги .5. Загальна архітектура системи.
6. Варіанти використання. 7. Компоненти програми для налаштування параметрів системи.
8. Програмні засоби та технології. 9. Інтерфейси реалізації застосунку.
10. Тестування. 11. Висновки по роботі. 12. Слайди презентації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			
Нормоконтроль			

7. Дата видачі завдання ___6 квітня_____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням кваліфікаційної роботи	13.04.2026	виконано
2	Збір та аналіз інформації за темою дослідження (огляд	13.04.2026 -	виконано
	фреймворку DSPy, методів автоматичного навчання та	15.04.2026	виконано
	оптимізації промптів, сучасних систем підготовки до		
	співбесід)		
3	Формування структури пояснювальної записки	16.04.2026	виконано
4	Розробка технічного завдання, проектування архітектури	17.04.2026 -	виконано
	бази даних та вибір методів реалізації інтелектуальних	21.04.2026	виконано
	модулів на базі DSPy		
5	Реалізація програмного забезпечення інтелектуальної	21.04.2026 -	виконано
	системи підготовки до співбесід та інтеграція з LLM	30.04.2026	виконано
6	Тестування функціоналу системи, оцінка точності	30.04.2026 -	виконано
	генерації та валідація відповідей	1.05.2026	виконано
7	Написання розділів пояснювальної записки	1.05.2026 -4.05.2026	виконано
8	Написання розділу 4: «Безпека життєдіяльності та	5.05.2026 -	виконано
	основи охорони праці»	6.05.2026	виконано
9	Оформлення висновків, списку використаних джерел,	7.05.2026	виконано
	додатків		
10	Перевірка роботи керівником, внесення правок		
11	Нормоконтроль		
12	Перевірка кваліфікаційної роботи на плагіат		
13	Попередній захист кваліфікаційної роботи		
14	Захист кваліфікаційної роботи		

Студент _____ Бурило В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бурило В. В. Розробка програмного забезпечення з використанням фреймворку Dspy для інтелектуальної системи підготовки до співбесід в IT-галузі, ТНТУ ім. Івана Пулюя, Тернопіль 2026.

Пояснювальна записка містить: 68 сторінок, 27 рисунків, 33 джерела та 3 додатки.

Об'єкт дослідження: процес автоматизованого оцінювання професійних навичок кандидатів та проведення тренувальних технічних співбесід в IT-галузі з використанням великих мовних моделей.

Мета: розробити інтелектуальну програмну систему підготовки до співбесід, яка за допомогою фреймворку DSPy забезпечує автоматичне вилучення та агрегацію вимог до вакансій, проведення адаптивного інтерв'ю та формування об'єктивного звіту про відповідність кандидата вимогам на основі аналізу історії чату.

Традиційні тренажери для IT-співбесід часто використовують статичні сценарії, що не враховують вимоги конкретних вакансій та обмежують об'єктивність оцінювання знань. У роботі розроблено інтелектуальну систему адаптивної підготовки до інтерв'ю. За допомогою фреймворку DSPy реалізовано модулі вилучення й агрегації вимог із вакансій порталу DOU, діалоговий агент на основі логіки Chain of Thought та модуль оцінювання відповідей за правилом білого списку (Whitelist Rule). Впроваджено двосторонню голосову взаємодію через WebSockets з інтеграцією сервісів розпізнавання та синтезу мовлення OpenAI. Вебдодаток побудовано на базі React, Vite та TailwindCSS, а серверна частина функціонує під керуванням FastAPI із базою даних PostgreSQL.

Ключові слова: інтелектуальна система, підготовка до співбесід, фреймворк DSPy, великі мовні моделі, аналіз вимог вакансії, оцінювання знань, WebSockets, розпізнавання мовлення, FastAPI, React.

ABSTRACT

Burylo, V. V. Software Development Using the DSPy Framework for an Intelligent System of Preparation for IT Industry Interviews, Ivan Pul'uj Ternopil National Technical University, Ternopil 2026.

The explanatory note contains: 68 pages, 27 figures, 33 references, and 3 appendices.

Research object: the process of automated evaluation of candidates' professional skills and conducting mock technical interviews in the IT sector using large language models.

Objective: to develop an intelligent software system for interview preparation that leverages the DSPy framework to automatically extract and aggregate job requirements, conduct adaptive mock interviews, and generate objective candidate fit assessments based on chat history analysis.

Traditional IT interview simulators often rely on static scenarios, ignoring specific vacancy requirements and limiting the objectivity of skills assessment. This work develops an intelligent system for adaptive interview preparation. Utilizing the DSPy framework, modules for extracting and aggregating requirements from DOU portal vacancies, a dialogue agent based on Chain of Thought logic, and an evaluation module using the Whitelist Rule are implemented. Two-way voice interaction is deployed via WebSockets integrating OpenAI speech recognition and synthesis services. The web application is built on React, Vite, and TailwindCSS, with the backend running FastAPI and PostgreSQL.

Keywords: intelligent system, interview preparation, DSPy framework, large language models, job requirements analysis, knowledge evaluation, WebSockets, speech recognition, FastAPI, React.

ПЕРЕЛІК СКОРОЧЕНЬ

AI (Artificial Intelligence) — Штучний інтелект.

API (Application Programming Interface) — Прикладний програмний інтерфейс.

CoT (Chain of Thought) — Метод послідовних міркувань великих мовних моделей.

DSPy (Declarative Self-improving Language Programs in Python) — Фреймворк для декларативного програмування та автоматичної оптимізації конвеєрів великих мовних моделей.

JSON (JavaScript Object Notation) — Текстовий формат обміну даними.

LLM (Large Language Model) — Велика мовна модель.

NLP (Natural Language Processing) — Обробка природної мови.

ORM (Object-Relational Mapping) — Об'єктно-реляційне відображення даних.

RAG (Retrieval-Augmented Generation) — Генерація відповідей великими мовними моделями із залученням додаткового контексту з зовнішніх джерел.

SQL (Structured Query Language) — Мова структурованих запитів до реляційних баз даних.

STT (Speech-to-Text) — Технологія перетворення усного мовлення на текстовий формат.

TS (TypeScript) — Мова програмування зі строгою типізацією на базі JavaScript.

TTS (Text-to-Speech) — Технологія синтезу усного мовлення за текстовим шаблоном.

UI (User Interface) — Користувацький інтерфейс.

UX (User Experience) — Досвід користувача від взаємодії з інтерфейсом.

WS (WebSocket) — Протокол повнодуплексного зв'язку та передачі даних у реальному часі.

ЗМІСТ

ВСТУП

Процес працевлаштування у галузі інформаційних технологій вимагає від кандидатів високого рівня технічної підготовки та відповідності динамічним вимогам ринку праці. Етап технічного інтерв'ю є визначальним для оцінювання кваліфікації спеціаліста, проте його проходження часто супроводжується значним рівнем стресу та браком практичного досвіду самопрезентації. Наявні засоби підготовки до співбесід не забезпечують достатньої адаптивності та реалістичності процесу оцінювання знань.

Існуючі програмні рішення для підготовки до співбесід пропонують статичні запитання або шаблонні сценарії, які не враховують вимог конкретних вакансій та обмежують об'єктивність оцінювання знань. Натомість реальні технічні інтерв'ю вимагають гнучкого діалогу та всебічної перевірки кваліфікації. Застосування великих мовних моделей дозволяє автоматизувати генерування запитань у реальному часі. Використання LLM дає змогу створювати сценарії спілкування, що адаптуються до відповідей користувача. Проте пряма взаємодія з моделями призводить до нестабільності відповідей та труднощів у керуванні поведінкою агента. Для вирішення цих проблем застосовано сучасні інструменти декларативного програмування та оптимізації конвеєрів.

З огляду на це, розробка програмного забезпечення з використанням фреймворку DSPy для підготовки до співбесід у галузі IT є актуальним завданням, що дозволить підвищити рівень готовності спеціалістів до технічних інтерв'ю.

Метою кваліфікаційної роботи є розробка програмного забезпечення з використанням фреймворку DSPy для інтелектуальної системи підготовки до співбесід у галузі інформаційних технологій, яка забезпечує автоматичне вилучення вимог до вакансій, ведення діалогу та формування оцінки відповідей кандидата.

Для досягнення поставленої мети вирішено такі завдання: проведено аналіз існуючих програмних систем підготовки кандидатів; досліджено

принципи роботи великих мовних моделей та методи автоматичної оптимізації текстових підказок; спроектовано архітектуру інтелектуальної системи підготовки до співбесід; розроблено алгоритми автоматичного вилучення вимог із вакансій; реалізовано діалоговий агент на основі фреймворку DSPy та логіки міркувань CoT; створено серверне програмне забезпечення для обміну даними у реальному часі через WS; розроблено вебдодаток користувача для взаємодії з агентом; проведено тестування розробленого програмного забезпечення та оцінено точність аналізу відповідей.

Об’єкт дослідження: процес автоматизованого проведення тренувальних співбесід та оцінювання відповідей кандидатів на технічні запитання у галузі інформаційних технологій.

Предмет дослідження: методи побудови, оптимізації та інтеграції конвеєрів великих мовних моделей з використанням фреймворку DSPy для генерації адаптивних запитань та аналізу відповідей під час підготовки до співбесід.

Методи дослідження. Проектування архітектурних рішень розроблюваної системи здійснювалося із застосуванням системно-аналітичних підходів. Застосовано методи декларативного програмування для налаштування поведінки мовних моделей та концепції об’єктно-орієнтованого проектування для реалізації серверної і клієнтської частин. Оцінювання відповідей кандидатів виконано з використанням методів математичної статистики.

Практичне значення одержаних результатів. Створене програмне забезпечення є готовим до використання практичним інструментом для самостійної підготовки кандидатів до технічних інтерв’ю. Система автоматично формує індивідуальні сценарії співбесід на основі описів вакансій, проводить тестування у режимі діалогу та надає об’єктивну оцінку знань із рекомендаціями для виправлення помилок.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Цей розділ присвячено комплексному аналізу предметної області, що є фундаментом для подальшого проектування інтелектуальної системи. У межах розділу детально розглянуто специфіку процесу технічних співбесід в IT-галузі, проведено огляд наявних аналогічних рішень, а також визначено теоретичні та технологічні засади, необхідні для ефективної реалізації поставленої задачі.

1.1 Аналіз процесу проведення технічних співбесід в IT-галузі

Процес працевлаштування в IT-галузі вимагає від кандидатів підтвердження високого рівня технічної підготовки, що здійснюється через проходження технічних співбесід. Цей етап є визначальним для оцінювання практичних навичок, теоретичних знань та вміння вирішувати нестандартні інженерні задачі під тиском часу. Для роботодавців проведення інтерв'ю є ресурсомістким процесом, що відволікає провідних спеціалістів від розробки програмного забезпечення, тоді як кандидати потребують інструментів для самостійного тестування знань перед зустріччю з інтерв'юером.

Наявні підходи до підготовки кандидатів мають значні недоліки. Самостійне опрацювання теоретичної літератури та розв'язання алгоритмічних завдань на спеціалізованих вебплатформах дозволяє засвоїти базові концепції, проте не формує навичок ведення живого технічного діалогу. Тренувальні співбесіди із залученням менторів є ефективними, проте їх важко масштабувати через високу вартість послуг та складність узгодження графіків. Більшість доступних автоматизованих систем пропонують лише фіксовані тестові запитання, що унеможливлює перевірку глибини розуміння матеріалу та вміння будувати логічні міркування.

Розвиток технологій штучного інтелекту, зокрема поява великих мовних моделей, відкриває нові можливості для автоматизації процесу підготовки до співбесід. Використання LLM дає змогу будувати діалогові системи, які адаптуються до рівня знань кандидата, генерують запитання у реальному часі та

аналізують відповіді користувача. Проте пряма інтеграція LLM супроводжується нестабільністю відповідей, схильністю до галюцинацій та складністю контролю за структурою діалогу. Спроби вирішити ці проблеми шляхом конструювання текстових підказок не забезпечують стабільної якості та потребують коригування при зміні версій базової моделі.

Для подолання вказаних обмежень доцільно використовувати фреймворк DSPy, який пропонує декларативний підхід до програмування та оптимізації конвеєрів на базі великих мовних моделей. Замість ручного підбору підказок цей інструмент дозволяє структурувати взаємодію з моделями у вигляді модулів із чітко визначеними інтерфейсами [1]. Завдяки автоматичній оптимізації параметрів конвеєра, система забезпечує стабільність відповідей та дотримання правил ведення інтерв'ю. Застосування методу CoT у межах модулів DSPy дозволяє моделювати процес послідовних міркувань, що підвищує об'єктивність та обґрунтованість оцінювання відповідей [5].

Отже, наявна практика підготовки фахівців до технічних співбесід в IT-галузі потребує модернізації через впровадження автоматизованих інтелектуальних систем. Створення спеціалізованого програмного забезпечення на основі фреймворку DSPy дозволить автоматизувати аналіз вимог вакансій, організувати динамічний діалог із кандидатом та забезпечити об'єктивне оцінювання його відповідей у реальному часі. Впровадження такого підходу підвищить ефективність підготовки розробників та знизить часові витрати технічних спеціалістів на проведення первинного оцінювання знань.

1.2 Огляд аналогів

Еволюційний розвиток технологій підготовки фахівців до співбесід демонструє перехід від статичних довідників до адаптивних систем моделювання діалогу. Спочатку кандидати використовували друковані матеріали та збірники теоретичних запитань без зворотного зв'язку. Наступним кроком стало впровадження вебплатформ для перевірки навичок написання коду, які оцінювали лише функціональну коректність рішень за допомогою

модульних тестів. Згодом з'явилися сервіси парних тренувань за участю реальних менторів, проте такі рішення мають високу вартість та обмежене масштабування. Сучасні системи базуються на використанні великих мовних моделей, що дозволяють вести діалог, адаптувати розмову до рівня кандидата та надавати розгорнутий аналіз відповідей [12].

Для наочного представлення відмінностей між поколіннями засобів підготовки розроблено структурну схему еволюції, яку наведено на рисунку 1.1.

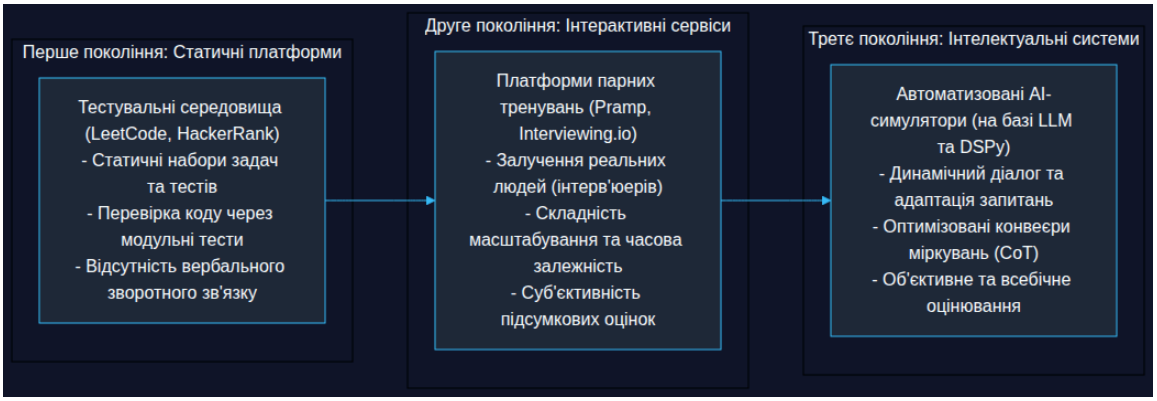


Рисунок 1.1 – Розвиток платформ для тренування перед технічними інтерв'ю

Архітектура сучасних систем моделювання співбесід складається з конвеєра взаємопов'язаних технологічних компонентів. Процес обробки відповіді кандидата починається з запису аудіосигналу та його передачі до модуля STT для перетворення мовлення на текстовий формат [6]. Текст надходить до ядра на базі LLM, де з використанням фреймворку DSPy відбувається семантичний аналіз відповіді та оцінювання теоретичних знань. Застосування методу CoT дозволяє системі будувати міркування для формування об'єктивного оціночного висновку. Після оцінювання система генерує наступне запитання, яке модуль TTS синтезує у голосове мовлення для відтворення в UI кандидата.

Попри переваги, практична реалізація таких діалогових систем має значні технічні обмеження. Головним бар'єром є затримка обробки запитів, оскільки послідовне виконання транскрипції, генерації відповіді моделлю та синтезу

мовлення потребує часу, що порушує динаміку розмови. Додатковою проблемою є збереження контексту тривалого діалогу, оскільки великий обсяг відповідей кандидата може перевищити ліміт контекстного вікна моделі, призводячи до втрати логіки бесіди. Нестабільність відповідей та схильність базових LLM до галюцинацій вимагають впровадження інструментів контролю діалогу, що обґрунтовує використання декларативного програмування для оптимізації взаємодії.

Схему взаємодії компонентів та основні технологічні обмеження системи відображено на рисунку 1.2.

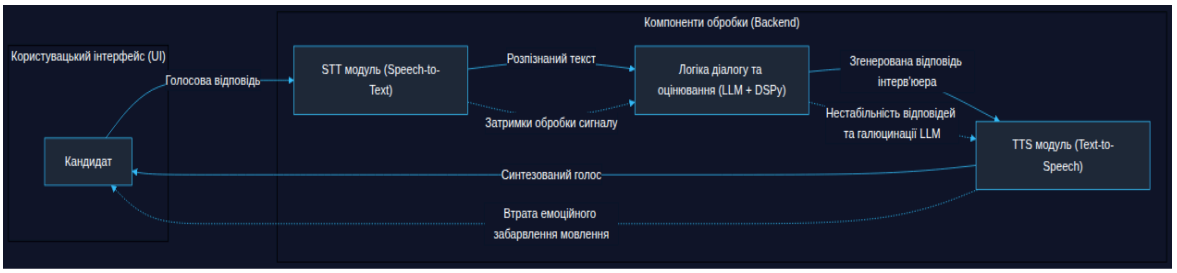


Рисунок 1.2 – Взаємодія компонентів та обмеження системи

Розробка систем підготовки до співбесід в Україні має власні особливості та лінгвістичні виклики. Головною перепоною є обмежена кількість відкритих україномовних наборів даних для навчання моделей розпізнавання та синтезу мовлення, адаптованих до термінології ІТ-галузі. Професійний сленг містить багато англомовних запозичень, що ускладнює роботу стандартних модулів розпізнавання та потребує адаптації моделей. Крім того, розгортання систем потребує забезпечення конфіденційності персональних даних кандидатів. Для вирішення цих проблем українські розробники застосовують локальні моделі та спеціалізовані методи оптимізації, створюючи основу для розвитку інтелектуальних рішень, або ж використовують готові сервіси для перетворення тексту на голос і у зворотню сторону, як от: OpenAI Whisper та OpenAI TTS, саме ці сервіси було використано для опрацювання людського голосу.

1.3 Дослідження методів оптимізації та побудови інтелектуальних систем

Проектування ефективного програмного забезпечення для моделювання технічних співбесід вимагає відмови від ручного налаштування підказок для великих мовних моделей. Традиційні підходи з жорстким кодуванням інструкцій часто призводять до нестабільності відповідей, що ускладнює їх автоматичну обробку. Нова парадигма розробки полягає в переході до використання декларативного програмування за допомогою фреймворку DSPy. Цей інструмент відокремлює логіку взаємодії від конкретної моделі, визначаючи лише сигнатури вхідних та вихідних даних, що забезпечує автоматичне формування оптимальних інструкцій на основі метрик якості. Процес оптимізації та компіляції цих модулів наведено на рисунку 1.3.

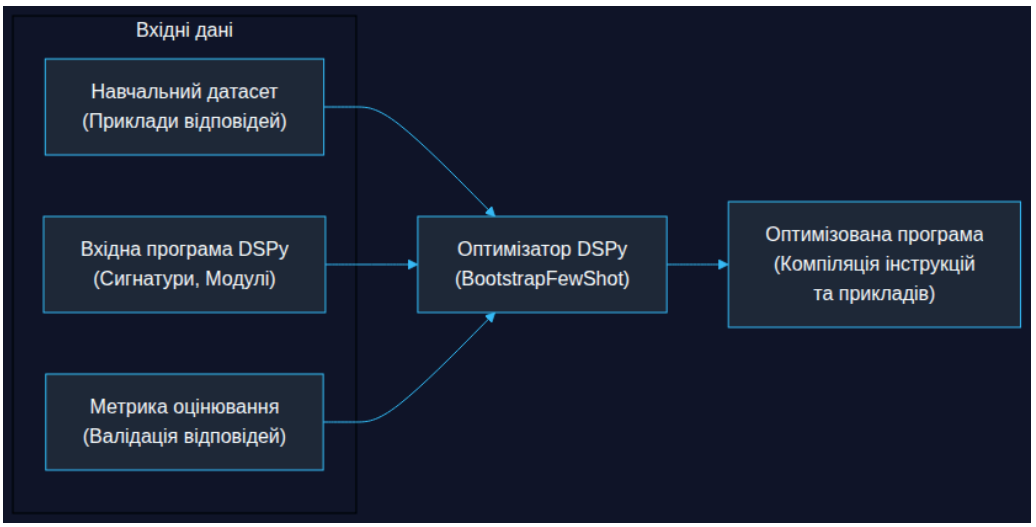


Рисунок 1.3 – Схема процесу оптимізації та компіляції модулів DSPy

Більшість існуючих систем підготовки до співбесід використовують комерційні API для прямого доступу до моделей компанії OpenAI. Використання таких сервісів зумовлює значні фінансові витрати при великій кількості запитів і регулярному тестуванні. Головним обмеженням цього підходу є висока вартість обробки токенів та неможливість прямого налаштування внутрішніх параметрів моделей. Крім того, часті оновлення

комерційних моделей призводять до непередбачуваних змін у якості генерації, через що раніше вручну налаштовані підказки втрачають свою ефективність.

Рішенням цієї проблеми є залучення фреймворку DSPy для автоматичної оптимізації взаємодії з моделями компанії OpenAI. Замість ручного підбору інструкцій, цей інструмент дозволяє автоматично компілювати оптимальні підказки, мінімізуючи довжину запитів та знижуючи загальну вартість використання API. Такий підхід забезпечує стабільність структури відповідей незалежно від змін у версіях моделей провайдера. Це дає змогу розробникам будувати економічно ефективну та стійку інфраструктуру, використовуючи високопродуктивні комерційні моделі без ризику погіршення якості обслуговування.

Синтез відкритих моделей та засобів декларативного програмування утворює оптимальну основу для розробки інтелектуального середовища. У роботі прийнято рішення побудувати систему у вигляді конвеєра, де кожен етап описується окремим модулем DSPy. Процес розпочинається з аналізу вимог вакансії та резюме кандидата. Далі генератор запитань будує діалог, адаптуючи складність питань, а для перевірки знань розробник застосовує модуль оцінювання, який використовує метод CoT для побудови обґрунтування оцінки. Система підлягає автоматичній компіляції за допомогою оптимізаторів фреймворку DSPy (рис. 1.4).

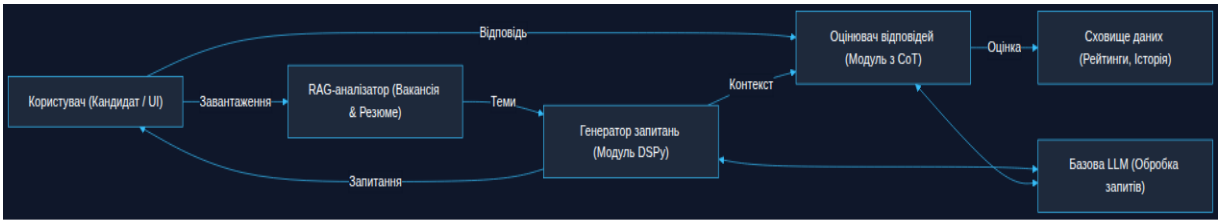


Рисунок 1.4 – Архітектурна схема взаємодії компонентів інтелектуальної системи

1.4 Огляд технологій організації потокової передачі та синтезу мовлення у вебсистемах

Організація двосторонньої мовленнєвої взаємодії у реальному часі між користувачем та інтелектуальним агентом у вебсистемах вимагає вирішення складної архітектурної задачі. Традиційні схеми передачі даних, засновані на періодичних запитах прикладного рівня та відправленні готових звукових файлів, спричиняють значні часові затримки. Вони порушують природну динаміку діалогу, роблячи систему незручною для симуляції співбесіди. Для подолання цих інфраструктурних обмежень застосовують технології потокового оброблення аудіосигналів на основі постійного двостороннього з'єднання. Використання протоколу WS дозволяє встановити стійкий канал зв'язку між клієнтом та сервером, забезпечуючи передачу пакетів даних із мінімальними накладними витратами без повторного встановлення з'єднання.

Базовим інструментом взаємодії з боку користувача виступає вебдодаток, розроблений на основі бібліотеки React та мови TS для реалізації динамічного UI [10]. Для захоплення голосового сигналу з мікрофона персонального комп'ютера застосовують стандартизовані інтерфейси веббраузера, зокрема прикладний інтерфейс запису медіафайлів [11]. Програмний код клієнтської частини здійснює безперервне зчитування аудіоданих із сенсора, сегментує отриманий потік на короткі фрагменти та конвертує їх у формат base64. Перетворення у текстове представлення дозволяє передавати бінарні дані в повідомленнях протоколу WS без ризику спотворення структури пакетів. Це забезпечує ізоляцію апаратної специфіки запису звуку від логіки обробки на сервері, гарантуючи кросплатформність та швидку доставку сигналу.

Отримані сервером текстові фрагменти у форматі base64 проходять декодування у бінарний вигляд для подальшої обробки. Для мінімізації навантаження на дискову підсистему та уникнення тривалих операцій введення-виведення, аудіодані накопичуються в оперативній пам'яті сервера за допомогою буферів потокового введення-виведення. Після завершення репліки

кандидата звукові дані передаються до інтегрованого сервісу Whisper-1 від OpenAI за допомогою інтерфейсу API. Технологія STT забезпечує точне перетворення усного мовлення на текстовий формат, розпізнаючи специфічну технічну термінологію ІТ-галузі та особливості вимови користувача. Швидка транскрипція в оперативній пам’яті без збереження проміжних файлів зменшує загальну затримку та готує дані для логічного аналізу.

Серверне програмне забезпечення системи реалізоване мовою Python із використанням асинхронного вебфреймворку FastAPI та сервера Uvicorn [7]. Вибір цієї технологічної платформи зумовлений потребою в ефективній обробці багатьох одночасних мережевих з’єднань. Завдяки використанню асинхронного циклу подій бібліотеки asyncio, сервер не блокує обчислювальні ресурси під час очікування аудіоданих від клієнтів [13]. Асинхронна архітектура дозволяє паралельно зчитувати аудіопотік із сокета WS, надсилати запити до сервісів OpenAI, обробляти результати у конвеєрі DSPy та повертати синтезовані голосові відповіді користувачу. Це забезпечує масштабування системи при мінімальних вимогах до апаратного забезпечення сервера.

Після успішного отримання тексту репліки користувача активується інтелектуальний конвеєр на базі фреймворку DSPy. Цей інструмент оптимізує взаємодію з мовними моделями та застосовує метод CoT для детального аналізу відповіді кандидата відповідно до вимог вакансії. Згенерована текстова відповідь мовної моделі передається до сервісу синтезу мовлення OpenAI TTS із використанням моделі gpt-4o-mini-tts [4]. Замість очікування повного синтезу тексту, сервер застосовує потокову генерацію звуку у вигляді послідовних бінарних пакетів. Кожен аудіопакет негайно конвертується у формат base64 і надсилається назад клієнту через з’єднання WS, що дозволяє розпочати відтворення голосу ще до завершення синтезу всієї фрази.

Клієнтський додаток на базі React та TS приймає вхідні повідомлення з аудіофрагментами через сокети WS. Аудіомодуль у складі UI здійснює миттєве декодування текстових рядків base64 назад у двійкові масиви безпосередньо у браузері. Для відтворення звуку без затримок розробник застосував прикладний інтерфейс Web Audio API [9]. Цей інструмент дозволяє динамічно створювати

буфери аудіо та планувати їх послідовне програвання на часовій шкалі з точністю до мілісекунди, формуючи ефект безперервного мовлення. Поєднання асинхронної обробки на сервері, потокового синтезу мовлення TTS та буферизації на стороні веббраузера забезпечує мінімальну затримку системи, наближаючи віртуальну співбесіду до реального інтерв'ю.

1.5 Постановка задачі на розробку програмного забезпечення

Результати дослідження предметної області дозволили визначити архітектурні орієнтири для майбутнього програмного продукту. У межах роботи створено інтелектуальне середовище тренування перед технічними співбесідами в IT-секторі з симуляцією діалогу в реальному часі. Функціонування системи спрямоване на підвищення готовності кандидатів за допомогою інтерактивного голосового діалогу та автоматизованої оцінки відповідей під вимоги конкретних вакансій.

Клієнтську частину системи побудовано у формі вебдодатка на базі React та TS. У вебдодатку забезпечено захоплення голосового сигналу з мікрофона користувача, його оперативну буферизацію та передачу бінарних даних на сервер без затримок. Для обміну інформаційними пакетами налаштовано стабільне повнодуплексне з'єднання за протоколом WS. Інтерфейс надає засоби введення пошукового запиту, вибору вакансій та запуску тестування, а відтворення відповідей асистента забезпечує інструмент Web Audio API.

Серверний додаток функціонує під управлінням мови Python з використанням FastAPI та Uvicorn, що гарантує високу швидкість обробки запитів. Звукові репліки кандидата проходять автоматичну транскрипцію в текстовий вигляд через Whisper-1 API, причому для мінімізації навантаження накопичення звуку відбувається в оперативній пам'яті сервера. Для озвучення відповідей інтегровано хмарний сервіс OpenAI TTS, який надсилає синтезовані голосові пакети назад клієнту у вигляді base64-рядків.

Управління штучним інтелектом здійснює конвеєр, спроектований за допомогою фреймворку DSPy. Цей ланцюжок поєднує модуль аналізу вимог

вакансій на базі RAG, генератор адаптивних запитань, оцінювач відповідей з CoT та інструмент оптимізації інструкцій. У системі застосовано автоматичну компіляцію промптів для LLM, що знижує витрати на запити та підвищує точність аналізу. Працездатність та швидкість відповіді конвеєра перевірено за допомогою автоматичних тестів на базі pytest.

Збереження результатів тестування реалізовано за допомогою реляційної бази даних та технології ORM. Такий архітектурний підхід гарантує високу надійність та структурну цілісність інформації. Усі звернення до сховища виконано асинхронно через мову SQL для запобігання блокуванню основного потоку сервера. Крім того, це суттєво підвищує загальну експлуатаційну продуктивність програмного комплексу. Програмний код також містить модуль логування для збору даних про тривалість розмов, фіксації помилок та збереження текстових транскриптів співбесід у форматі JSON з метою аудиту роботи системи. Детальне протоколювання подій є невід'ємною складовою створення безпечних інформаційних систем.

Для забезпечення повноцінного функціоналу в системі реалізовано механізми автоматичного збору та аналізу інформації про вакансії. У програмі спроектовано модуль парсингу даних з вебсайтів пошуку роботи (зокрема, DOU) за заданим користувачем пошуковим запитом. Отримані вакансії зберігаються в реляційній базі даних і використовуються для формування бази знань RAG. Інтеграція такої концептуальної бази знань дозволяє значно оптимізувати подальшу машинну обробку даних. Процес тестування включає генерацію адаптивних запитань на основі вимог знайдених вакансій, а за результатами симуляції співбесіди забезпечено розрахунок підсумкових оцінок із визначенням сильних та слабких сторін відповідей кандидата щодо кожної вакансії. Формування подібних аналітичних результатів сприяє максимальній об'єктивізації процесу професійного відбору.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У даному розділі описано етап проектування та безпосередньої розробки програмного забезпечення. Зокрема, висвітлено питання побудови архітектури комплексу, принципи роботи конвеєра інтелектуальних агентів, а також методи реалізації серверної логіки, інтеграції з хмарними сервісами та створення підсистеми оцінювання результатів.

2.1 Проектування загальної архітектури програмного та апаратного комплексу

Проектування архітектури програмного та апаратного комплексу є визначальним етапом створення інтелектуальних систем автоматизації IT-співбесід. У межах роботи прийнято рішення побудувати систему за триланковим клієнт-серверним архітектурним шаблоном, що складається з клієнтської частини, серверного додатка та обчислювального вузла інтелектуальних агентів. Головною метою розробки є забезпечення інтерактивного та стабільного двостороннього зв'язку між кандидатом і мовними моделями з мінімальним рівнем затримки мережі. Така конфігурація дозволяє симулювати умови реальної технічної співбесіди, адаптуючи поведінку віртуального інтерв'юера до реакцій кандидата. Розподіл системи на ізольовані модулі забезпечує високу відмовостійкість, оскільки збої під час обробки звукового сигналу чи виконання запитів до LLM локалізуються без зупинки інших процесів. Програмна архітектура орієнтована на можливість подальшого горизонтального масштабування, що дає змогу розподіляти обчислювальні навантаження між незалежними сервісами під час інтенсивного використання платформи.

Клієнтську частину реалізовано у вигляді односторінкового вебдодатка на базі React та TS з інтеграцією Tailwind CSS для побудови адаптивного інтерфейсу користувача [19]. Вебдодаток виконує функції відображення графічного UI, взаємодії з апаратними засобами користувача для запису звуку через Web Audio API, а також керування станом сесії підготовки до співбесіди.

Для оптимізації обміну інформацією між клієнтом і сервером застосовано протокол WS, який гарантує передачу аудіоданих у режимі реального часу без потреби постійного створення нових з'єднань. Використання TS на клієнтському рівні забезпечує строгу типізацію компонентів, що мінімізує виникнення помилок при обробці відповідей від сервера. Управління локальним станом додатка реалізовано за допомогою спеціальних користувацьких хуків, які ізолюють бізнес-логіку взаємодії від компонентів відображення. Клієнтська частина також відповідає за попередню фільтрацію вхідних даних та динамічну візуалізацію отриманих аналітичних звітів з оцінками навичок.

Серверний додаток побудовано на базі асинхронного вебфреймворку FastAPI мовою Python під управлінням ASGI-сервера Uvicorn. Цей модуль виступає центральним маршрутизатором і шлюзу доступу до сховища та зовнішніх обчислювальних платформ. Сервер забезпечує обробку аудіосигналів, конвертацію форматів, а також координацію викликів зовнішніх інструментів STT та TTS. Усі транскрипції та синтезовані відповіді передаються через сервіси Whisper та OpenAI TTS, що дозволяє організувати голосовий діалог. Збереження результатів симуляції співбесід, транскриптів та оцінок реалізовано у базі даних PostgreSQL. Взаємодію зі сховищем реалізовано асинхронно через ORM SQLAlchemy за допомогою драйвера asyncpg та інструменту Alembic, що виключає блокування потоків сервера при високій частоті SQL запитів [8]. Загальну схему архітектури представлено на рис. 2.1.

Ядром інтелектуальної обробки є обчислювальний конвеєр на базі фреймворку DSPy, який замінює ручне конструювання промптів декларативним програмуванням модулів машинного навчання [18]. Конвеєр об'єднує класи InterviewAgent, AssessmentAgent та RequirementsExtractorAgent, які наслідують dspy.Module. RequirementsExtractorAgent аналізує вакансії для виділення ключових вимог, які зберігаються у базі знань. Під час сесії InterviewAgent використовує підхід RAG для вибірки вимог і динамічної генерації запитань кандидату з урахуванням історії діалогу за принципом CoT. Після симуляції AssessmentAgent здійснює оцінку відповідей користувача на відповідність вакансії, формуючи звіт у форматі JSON. Для підвищення точності відповідей

конвеєр оптимізовано за допомогою алгоритмів підбору прикладів для автоматичного конструювання найкращих інструкції для LLM. Детальну схему організації конвеєра та потоків даних зображено на рис. 2.2.

Взаємодію компонентів під час робочого сеансу підготовки побудовано на послідовному обміні інформаційними повідомленнями згідно з архітектурними зв'язками (рис. 2.1, рис. 2.2). Початковим етапом є парсинг вакансій з DOU через модуль збору вимог, після чого дані записуються в PostgreSQL. Після ініціалізації сесії користувачем клієнтський додаток встановлює з'єднання через WS з FastAPI. Кандидат надсилає аудіовідповідь, яка обробляється сервером та передається до STT для транскрибування. Отриманий текст надходить у DSPy конвеєр, де InterviewAgent здійснює пошук схожих вимог у базі знань за схемою RAG та генерує наступне запитання на базі LLM. Згенерована відповідь озвучується за допомогою TTS і надсилається користувачу у формі звукового потоку. Паралельно з цим AssessmentAgent накопичує текстові транскрипти та проводить аналіз відповідності навичок кандидата з фіксацією результатів у базі даних. Зазначений підхід дозволяє створювати гнучкі сценарії підготовки відповідно до актуальних вимог ринку праці.

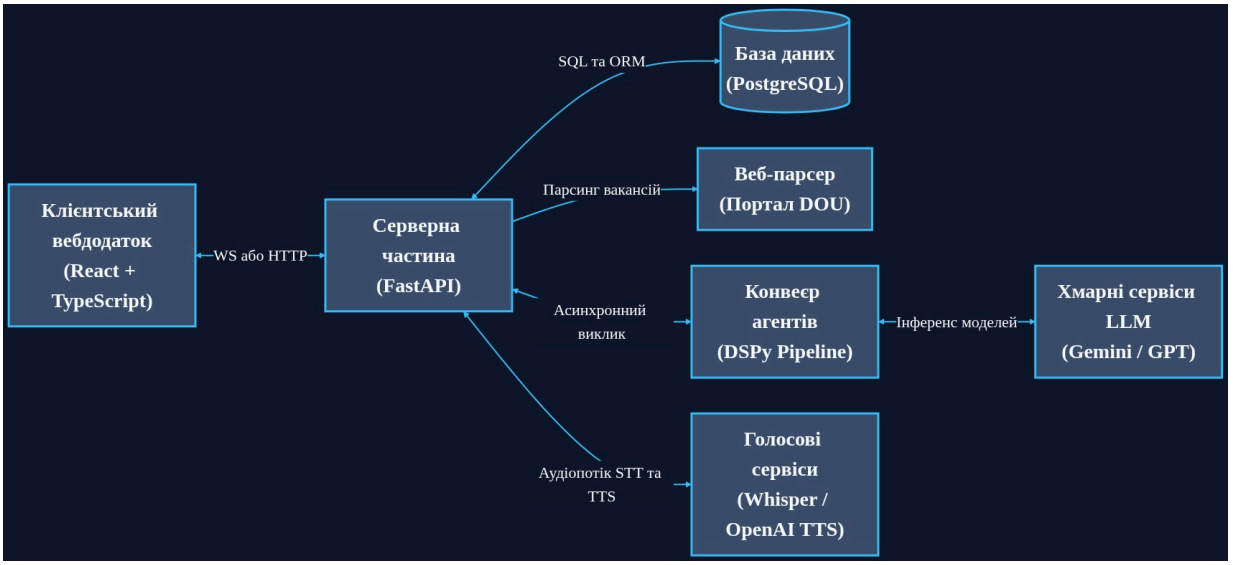


Рисунок 2.1 – Схема загальної архітектури програмно-апаратного комплексу

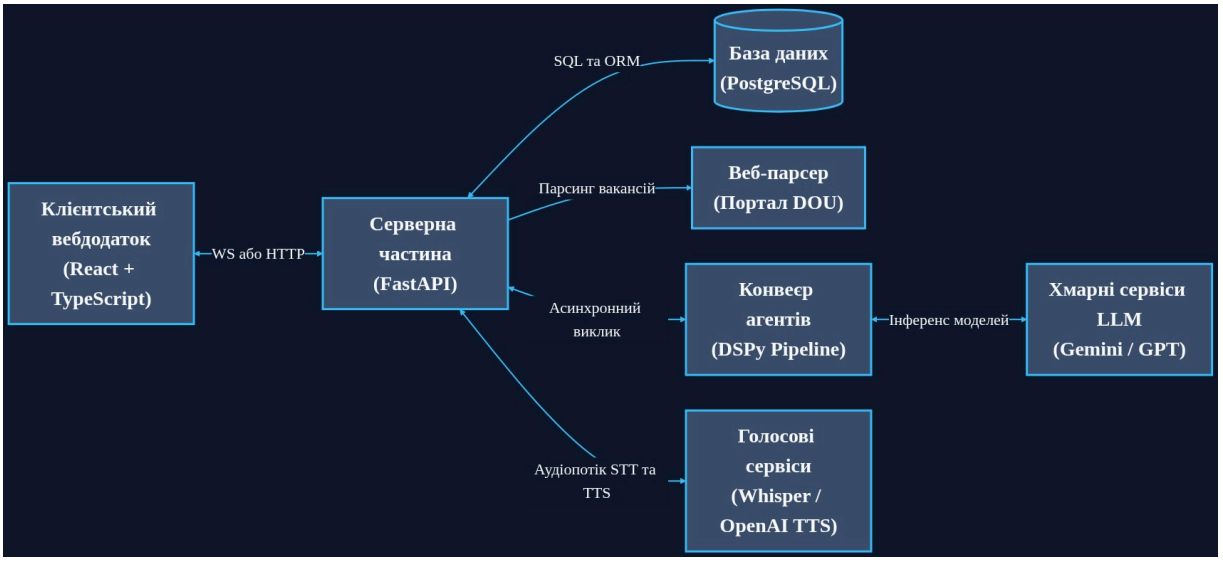


Рисунок 2.2 – Схема взаємодії та потоків даних конвеєра інтелектуальних агентів DSPy

2.2 Розробка та оптимізація конвеєра інтелектуальних агентів

Створення інтелектуальної системи підготовки до співбесід базується на застосуванні фреймворку DSPy для побудови конвеєрів великих мовних моделей. Замість ручного проектування промптів, розробник використовує декларативну парадигму програмування інтелектуальних агентів. Кожен агент розглядається як окремий програмний модуль, інтерфейс і поведінка якого визначаються строгою сигнатурою. Основу системи складають три взаємопов'язані модулі: екстрактор вимог вакансій, діалоговий агент та сервіс оцінювання відповідей. Це дозволяє повністю відокремити бізнес-логіку системи від конкретної великої мовної моделі та автоматизувати оптимізацію інструкцій при зміні цільових вимог.

При розробці модуля діалогового агента InterviewAgent першочерговим кроком стало проектування декларативного опису інтерфейсу взаємодії. Для цього розроблено клас InterviewSignature, який наслідує базовий dspy.Signature. Ця сигнатура визначає межі для вхідних полів та структури вихідних даних, що очікуються від великої мовної моделі. На відміну від звичайних промптів, такий

опис інтегрується безпосередньо у програмний код, що забезпечує строгу типізацію та спрощує обробку вихідних значень. У межах сигнатури вхідними параметрами виступають назва вакансії, вимоги, історія сесії та остання репліка кандидата, а вихідними — логічне обґрунтування дій, прапорець завершення діалогу та текст відповіді.

Агент проведення інтерв'ю реалізує складний алгоритм ведення діалогу з кандидатом. Головним завданням є перевірка відповідності навичок здобувача вимогам вакансії без повторення запитань чи ігнорування попередніх реплік. Для цього впроваджено семантичну дедуплікацію на основі бази знань. Процес починається з аудиту історії сесії та зчитування наявного контексту. Система використовує парасолькове правило: при наявності глибоких знань у певній темі, детальні запитання щодо дрібних інструментів цієї ж категорії можуть бути пропущені. Водночас агент перевіряє відповіді на валідність: якщо кандидат перевів тему, InterviewAgent повторює запит у простій формі. Алгоритм діалогу наведено на рис. 2.3.

Рисунок 2.3 – Схема алгоритму виконання діалогового кроку InterviewAgent

Після кожної репліки кандидата система здійснює аналіз покриття вимог вакансії. Якщо всі ключові навички з консолідованого списку були обговорені та оцінені, InterviewAgent встановлює прапорець завершення діалогу в значення істини, що сигналізує серверній частині про необхідність зупинки сесії. Оцінювання результатів здійснюється в асинхронному режимі після завершення діалогу. Модуль оцінювання аналізує транскрипт бесіди, зіставляючи відповіді з вимогами. Кожна вимога отримує статус відповідності за трьома рівнями:

успішно пройдено, не пройдено або частково підтверджено. На основі оцінок будується звіт у форматі JSON, який зберігається у PostgreSQL та передається клієнту через вебсокети для побудови графічної карти навичок користувача. Схему конвеєра оцінювання зображено на рис. 2.4.

Рисунок 2.4 – Схема конвеєра оцінювання та збереження результатів

Окремим важливим етапом розробки є оптимізація та компіляція створених DSPy-агентів. Традиційне використання штучного інтелекту часто супроводжується проблемою нестабільності вихідних даних та чутливості до найменших змін у тексті запиту. У цій роботі застосовано автоматичне навчання промптів за допомогою оптимізатора BootstrapFewShot. Процес компіляції використовує набір навчальних прикладів, який містить типові запитання та очікувані відповіді. Оптимізатор оцінює роботу програми за допомогою метрики якості й автоматично підбирає найкращі демонстраційні приклади для промптів, що мінімізує використання токенів та підвищує стабільність структури й унікальність відповідей. Схему процесу компіляції зображено на рис. 2.5.

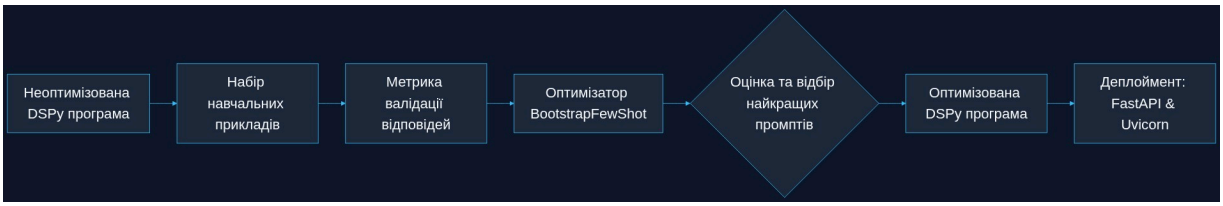


Рисунок 2.5 – Схема процесу компіляції та оптимізації конвеєра DSPy

2.3 Реалізація серверної частини та алгоритмів обробки звуку

Серверна частина розробленої системи функціонує під управлінням асинхронного вебфреймворку FastAPI на базі мови Python та сервера Uvicorn. Розробник обрав ці інструменти завдяки високій швидкодії, надійності та простоті інтеграції з конвеєрами машинного навчання. Головним архітектурним рішенням став перехід від багатопотокового підходу до асинхронної обробки мережевих подій. Використання циклу подій дозволяє обслуговувати з'єднання без блокування ресурсів. Під час ініціалізації додатка система конфігурує параметри дозволених джерел для взаємодії з клієнтським інтерфейсом, а також запускає життєвий цикл, який перевіряє стан підключення до бази даних та працездатність менеджера історії діалогів.

Для організації зв'язку між кандидатом та інтелектуальним агентом у реальному часі платформа підтримує роботу за протоколом WS на маршруті “/speech/stream”. Клієнтська частина ініціює сеанс, надсилаючи стартове повідомлення у форматі JSON, яке система валідує відповідно до схеми даних. Цей пакет містить ідентифікатор сесії, прапорець активації синтезу мовлення та параметри аудіозапису. Після підтвердження готовності з боку бекенду клієнтський додаток починає зчитування голосового сигналу користувача за допомогою Web Audio API. Клієнтський додаток кодує та передає отримані бінарні фрагменти через WS-з'єднання у вигляді текстових пакетів із base64-кодуванням.

На стороні обчислювального вузла асинхронне завдання приймає вхідні повідомлення та накопичує їх в оперативній пам'яті як байтовий буфер. Такий підхід виключає запис тимчасових файлів на диск, прискорюючи обробку інформації. Спеціальні службові фрейми узгоджують процес передачі даних. Коли користувач завершує чергову репліку, клієнтський додаток автоматично надсилає фрейм закінчення запису. Це повідомлення сигналізує обробнику про потребу переходу до етапу розпізнавання мовлення.

Після отримання відповідного сигналу асинхронний обробник об'єднує накопичені чанки в єдиний аудіопотік. Далі обробник викликає функцію

транскрибування на базі моделі Whisper STT, яка перетворює звуковий сигнал на текстовий рядок. Розпізнаний текст надходить клієнту через WS для відображення на екрані. Одночасно інформація передається до конвеєра DSPy, де InterviewAgent визначає наступне запитання. Застосування асинхронного генератора дозволяє передавати результати роботи великої мовної моделі покроково, без очікування завершення формування всієї відповіді.

Потокова передача відповіді кандидату складається з двох паралельних процесів: трансляції текстових токенів та генерації голосового супроводу. Система транслює окремі токени рішення чи відповіді клієнту у форматі JSON. Для розділення типів інформації платформа використовує систему міток подій, завдяки чому клієнт відрізняє проміжні міркування моделі від фінальної відповіді. Трансляція текстових фрагментів відбувається безпосередньо у процесі їх генерації великою мовною моделлю, що створює ефект живого спілкування. Якщо користувач активував режим озвучування, накопичений текст паралельно надходить до сервісу OpenAI TTS. Сервіс синтезу мовлення підтримує гнучке налаштування параметрів темпу та вибору конкретного голосового профілю через конфігураційний файл. Бекенд кодує звукові фрагменти в base64 та передає їх клієнту через WS-з'єднання. На стороні фронтенду черга аудіоконтексту Web Audio API послідовно відтворює отримані звукові пакети без затримок чи переривань. Це дозволяє відтворювати голос асистента з мінімальною затримкою. У разі перевищення ліміту очікування відповіді від зовнішнього API система автоматично перемикається на резервний сервер. При виникненні непередбачуваних помилок під час обробки звуку чи виклику зовнішніх сервісів система відправляє користувачу повідомлення про збій та закриває з'єднання. Система реєструє будь-які аномалії в роботі мережевих сокетів у журналі логування з детальним описом стеку викликів. Після завершення симуляції діалогу система записує зібрані аналітичні метрики в реляційну базу даних для подальшої обробки модулем оцінювання. Для оптимізації швидкодії клієнтська частина платформи додатково кешує отримані текстові транскрипти у локальному сховищі браузера.

Загальну послідовність процесів та потоків даних під час асинхронної обробки мовного сигналу зображено на рис. 2.6. Схема показує взаємодію між клієнтом, сервером FastAPI, сервісами Whisper STT, модулями DSPy та синтезатором OpenAI TTS.

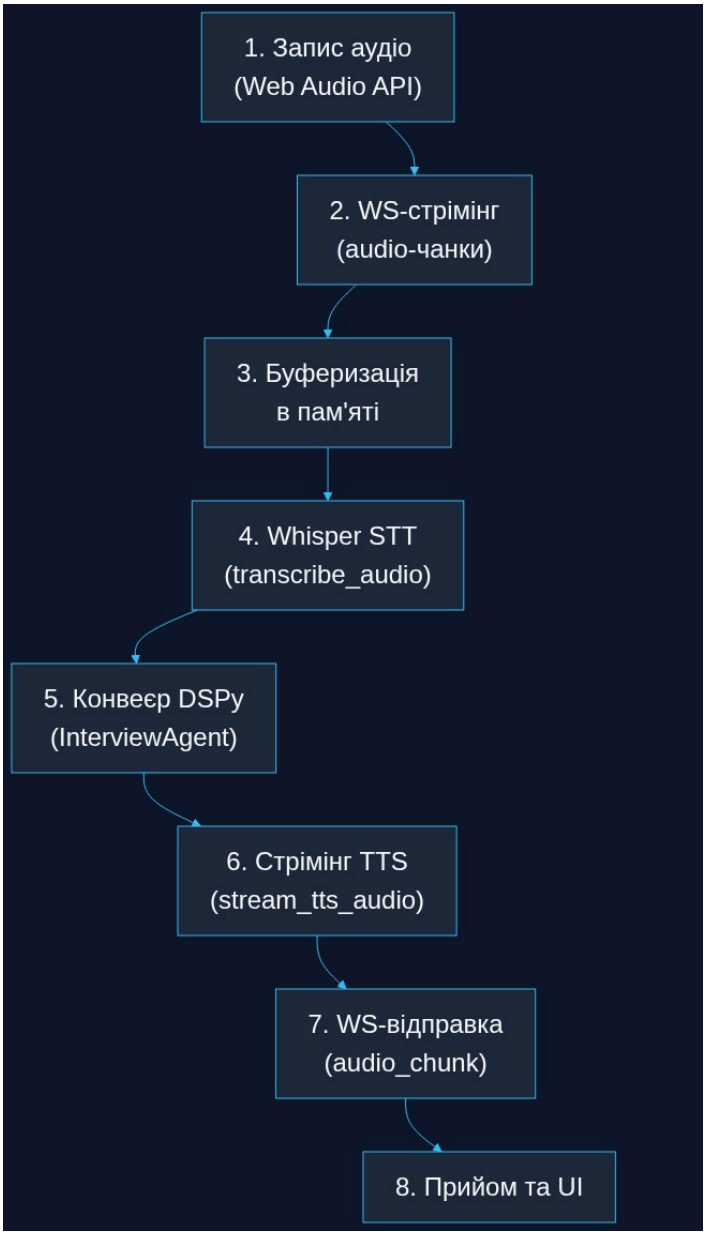


Рисунок 2.6 – Схема асинхронної обробки та потокової трансляції мовлення

2.4 Інтеграція хмарних сервісів для розпізнавання та синтезу мовлення

Інтеграція хмарних сервісів штучного інтелекту є завершальним етапом розробки системи підготовки до співбесід. Для забезпечення голосової взаємодії у реальному часі реалізовано зв'язок серверної частини FastAPI із зовнішніми хмарними службами розпізнавання та синтезу мовлення. Головним інструментом взаємодії між клієнтським додатком на базі React та сервером є протокол WS. Вибір цього протоколу зумовлений можливістю підтримувати постійне двостороннє з'єднання без повторного встановлення сеансу для кожної репліки кандидата. Це мінімізує мережеві затримки, які є критичними для систем реального часу. Авторизація у хмарних сервісах здійснюється на рівні серверної частини за допомогою унікальних ключів доступу, які зберігаються у захищених змінних оточення. Такий підхід гарантує безпеку конфіденційних даних та унеможливлює перехоплення ключів на стороні клієнта. Голосовий трафік шифрується на транспортному рівні під час передачі через глобальну мережу.

Процес взаємодії починається з обов'язкового етапу конфігурування сеансу. Після встановлення підключення за протоколом WS клієнт надсилає початковий пакет у форматі JSON. Повідомлення містить ідентифікатор сесії, параметри мови та ознаку активації зворотного синтезу мовлення. Отримавши ці дані, сервер перевіряє статус співбесіди та завантажує вимоги вакансії з бази даних через ORM. Для формування контексту використовується метод RAG, що дозволяє передавати агенту InterviewAgent актуальну інформацію для перевірки навичок кандидата. Сервер також встановлює параметри синтезу мовлення, визначаючи тип голосу та темп мовлення. Лише після успішного підтвердження конфігурації хмарним сервісом система переходить у режим очікування голосового потоку. Така послідовність гарантує, що агент не почне обробляти відповіді кандидата до повного завантаження правил та обмежень бази знань.

Основний механізм обміну даними побудовано на принципах асинхронного потокового передавання. Під час відповіді клієнтський додаток

записує голос користувача через Web Audio API, кодує звукові фрагменти у формат Base64 та надсилає їх на сервер. Сервер асинхронно приймає ці повідомлення та накопичує їх в оперативній пам'яті як байтовий буфер. Це виключає потребу запису тимчасових файлів на диск і прискорює обробку. Після отримання сигналу про завершення репліки сервер передає накопичений буфер до моделі Whisper STT для перетворення звуку на текст. Отриманий рядок надходить у конвеєр DSPy, де InterviewAgent на основі історії діалогу за принципом CoT генерує відповідь. Після формування відповіді запускається сервіс OpenAI TTS, який синтезує голосову відповідь та надсилає її клієнту у вигляді закодованих аудіочанків через з'єднання WS. Клієнт декодує ці фрагменти та відтворює звук, забезпечуючи природний темп спілкування з мінімальною затримкою.

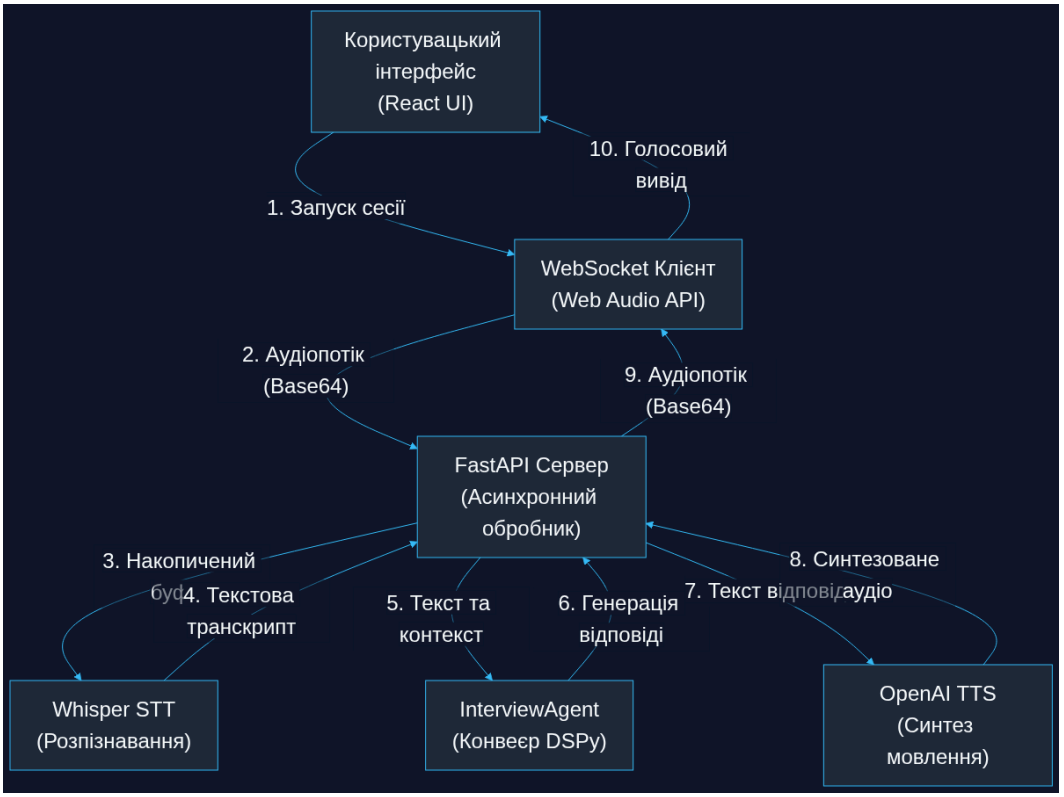


Рисунок 2.7 – Схема потокового обміну даними та інтеграції хмарних сервісів

2.5 Налаштування логіки діалогового агента

Налаштування логіки діалогового агента та розробка сигнатур фреймворку DSPy є визначальним етапом проектування системи. Традиційний підхід із ручним написанням промптів демонструє нестабільність вихідних форматів даних та високу чутливість до найменших змін у моделях. Для усунення цих недоліків розробник реалізує декларативну парадигму, де поведінка агента визначається структурованим інтерфейсом dspy.Signature. Це дозволяє повністю відокремити бізнес-логіку від конкретної LLM, автоматично валідувати типи даних і спростити інтеграцію з іншими сервісами.

Центральним елементом конвеєра є сигнатура InterviewSignature. Вхідні параметри включають назву вакансії, вимоги до кандидата, історію поточної сесії та останню репліку, отриману після Whisper STT. Вихідні дані містять міркування CoT, наступну репліку інтерв'юера та статус завершення сесії. Кожне поле сигнатури має детальний текстовий опис, що обмежує поведінку моделі. Перед кожною відповіддю агент виконує аналіз відповідності навичок вимогам, що мінімізує ризик виникнення галюцинацій та підвищує якість діалогу.

Для симуляції реального процесу спілкування з технічним спеціалістом IT-галузі розробник налаштовує поведінку віртуального інтерв'юера за допомогою інструкцій у сигнатурі. Агент веде розмову у професійному, об'єктивному та стриманому тоні. Алгоритм діалогу передбачає послідовну перевірку знань за кожною вимогою вакансії без повторення питань чи різких переходів. Для забезпечення сумісності з OpenAI TTS встановлено технічне обмеження: модель генерує виключно лінійний текст без таблиць, списків, спеціальних символів чи складного форматування, що гарантує плавне звукове відтворення.

Програмний механізм керування сеансом використовує динамічний аналіз стану діалогу. Агент зіставляє відповіді користувача з переліком вимог вакансії. Коли всі навички перевірено або кандидат самостійно прощається, модель встановлює статус завершення в істинне значення. Сервер FastAPI розпізнає

цей маркер, ініціює збереження даних у PostgreSQL, закриває WS-з'єднання та активує модуль оцінювання для формування JSON-звіту. Цей механізм автоматизує завершення інтерв'ю та оновлює графічний інтерфейс користувача з фінальними оцінками. Схему потоків зображено на рис. 2.8.

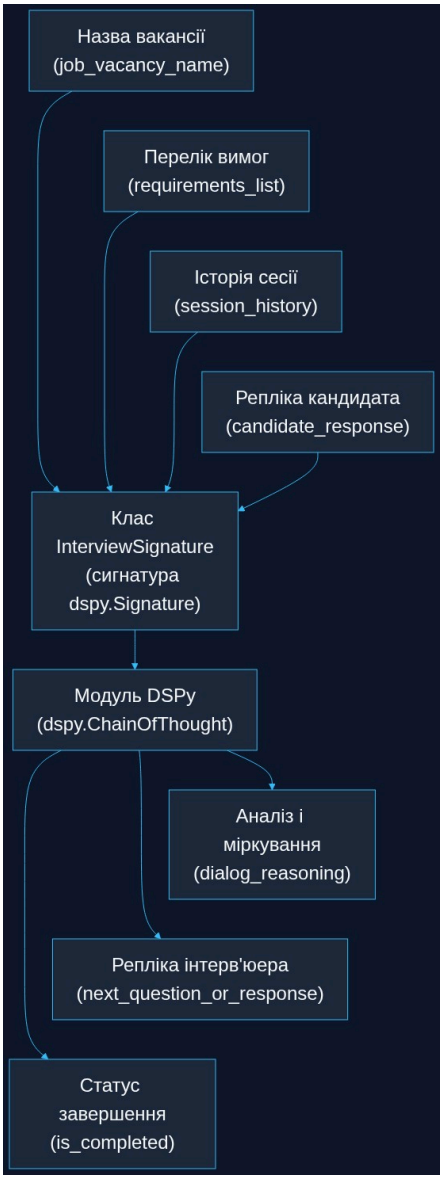


Рисунок 2.8 – Структура сигнатури та схема потоків даних діалогового агента

2.6 Створення підсистеми оцінювання та збереження результатів

Створення надійної підсистеми оцінювання та збереження результатів є обов'язковим етапом розробки інтелектуальної системи підготовки до співбесід. Головним завданням цього архітектурного модуля є всебічний аналіз відповідей кандидата, об'єктивне оцінювання рівня його технічних навичок згідно з обраною вакансією та надійне збереження сформованих звітів у базі даних. Окрім оцінки знань, модуль виконує функцію діагностики перебігу розмови, фіксуючи технічні параметри з'єднання, кількість використаних токенів і затримки відповідей великої мовної моделі. Такий аналіз дозволяє адміністраторам відстежувати ефективність роботи діалогового агента та превентивно виявляти збої у мережевому зв'язку або аномалії в обробці аудіопотоку. Крім того, накопичені підсистемою дані слугують основою для формування нових навчальних вибірок, які використовуються для оптимізації та компіляції інших агентів фреймворку DSPy. Усі процеси оцінювання та запису логів відбуваються в асинхронному режимі у фоні, що повністю унеможливлює негативний вплив на швидкість взаємодії під час поточної розмови та не створює додаткових затримок для користувача.

Створення даного модуля спирається на застосування асинхронних можливостей мови Python разом із механізмами розумного оброблення даних. Робота модуля активується автоматично після завершення сесії зв'язку та отримання відповідного сигналу від сервера FastAPI. Підсистема зчитує повний транскрипт діалогу з оперативної пам'яті та передає його до спеціалізованого агента оцінювання AssessmentAgent, побудованого на базі фреймворку DSPy. Цей агент працює на основі строго визначеної сигнатури, яка приймає на вхід історію розмови та перелік вимог до вакансії, що зберігаються в базі даних PostgreSQL. Процес перевірки містить етап логічного обґрунтування за технологією CoT, під час якого модель послідовно аналізує кожну репліку кандидата, зіставляє її з необхідними навичками та визначає рівень володіння технологіями. Для гарантування строгої структури вихідних даних агент використовує вбудовану валідацію за допомогою Pydantic, що дозволяє

уникнути помилок синтаксичного аналізу JSON. Завдяки такому підходу мінімізується суб'єктивність оцінок та запобігає виникненню галюцинацій штучного інтелекту, забезпечуючи відповідність висновків фактичним відповідям користувача.

На завершальному етапі роботи підсистема структурує отримані результати та здійснює їх персистентне збереження. Сформований звіт перетворюється на JSON-об'єкт, який містить детальні оцінки за кожною вимогою вакансії, цитати з відповідей кандидата, загальний висновок та рекомендації для покращення знань. Взаємодія з базою даних PostgreSQL реалізується за допомогою асинхронної ORM SQLAlchemy, що забезпечує високу ефективність обробки запитів. Збережена інформація практично миттєво передається до клієнтського застосунку на базі React за допомогою протоколів WebSocket або REST API. Це архітектурне рішення дозволяє динамічно оновлювати інтерфейс користувача в режимі реального часу та коректно відображати інтерактивну карту навичок.

Використання строго уніфікованого формату звітів значно спрощує експорт зібраної статистики до зовнішніх аналітичних систем та створює надійну базу для подальшого ретроспективного аналізу професійного прогресу кандидата. Завдяки комплексній автоматизації процесу оцінювання та збереженню детальної історії відповідей, рекрутери отримують об'єктивний інструмент для проведення швидкого скринінгу. Це, у свою чергу, суттєво оптимізує процеси найму в IT-галузі.

Варто також відзначити, що така архітектура забезпечує високий рівень масштабованості платформи в умовах постійного зростання кількості користувачів. Динамічний розподіл обчислювальних ресурсів сприяє максимально раціональному використанню наявних апаратних потужностей сервера. Завдяки цьому горизонтальне масштабування платформи відбувається автоматично і залишається абсолютно непомітним для кінцевого користувача. Невід'ємною складовою успішного функціонування агента є застосування вдосконалених методів препроцесингу вхідного тексту. Впроваджені алгоритми дозволяють системі з високою точністю розпізнавати складні семантичні

конструкції у неструктурованих відповідях кандидатів. Глибинна обробка зібраних метрик на серверному боці дозволяє не лише формувати базові профілі компетенцій, але й нівелювати вплив суб'єктивного людського фактора під час ухвалення рішень. Отриманий в результаті такого аналізу масив даних формує вичерпну репрезентативну вибірку професійних компетенцій. Ця вибірка слугує надійним статистичним базисом для прийняття обґрунтованих управлінських рішень щодо працевлаштування. Більше того, повне усунення когнітивних упереджень, які неминуче супроводжують традиційні формати співбесід, кардинально підвищує валідність кінцевих оцінок. Розроблений алгоритм оперує виключно кількісними метриками, що апріорі виключає будь-які прояви дискримінації. Такий стандартизований підхід гарантує максимально прозорі та рівні умови оцінювання для всіх претендентів, незалежно від поточного навантаження на систему чи спеціалістів відділу кадрів.

Крім того, генерування системних логів надає розробникам можливість систематично вдосконалювати діалогові сценарії та актуалізувати бази знань платформи. Щодо підсистеми системного логування, то вона додатково виконує функцію ключового інструменту для проведення глибокого ретроспективного аналізу можливих інцидентів. Усі агреговані журнали подій проходять обов'язкову процедуру строгого очищення перед їх передачею до аналітичного модуля. Для забезпечення комплексного захисту конфіденційної інформації на всіх рівнях архітектури застосовано сучасні протоколи наскрізного криптографічного шифрування. Розроблений регламент обслуговування передбачає створення не лише повних, але й регулярних інкрементальних знімків бази даних, що дозволяє швидко відновити систему після будь-яких апаратних збоїв. Такий комплексний підхід до збереження інформації мінімізує ймовірність безповоротної втрати критично важливих відомостей у разі виникнення непередбачуваних позаштатних ситуацій. Окрім цього, впроваджена підсистема моніторингу безперервно відстежує стан бази даних, завчасно сигналізуючи про потенційні відхилення від нормальних параметрів експлуатації. У підсумку, інтеграція всіх вищезазначених безпекових та архітектурних рішень формує надійний фундамент для безперебійної та

довгострокової роботи розробленого програмного комплексу. Послідовність подій цього процесу схематично зображено на рис. 2.9.

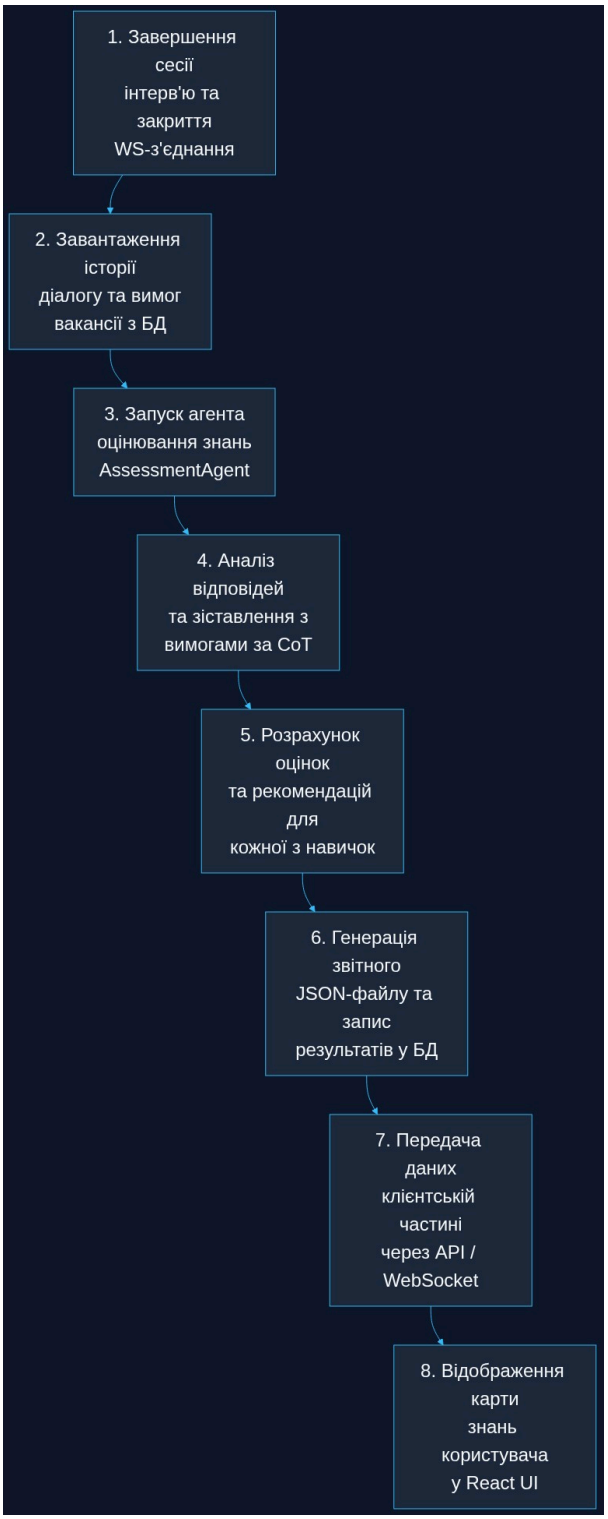


Рисунок 2.9 – Схема роботи підсистеми оцінювання та збереження результатів

3 ВПРОВАДЖЕННЯ ІНТЕЛЕКТУАЛЬНОГО ГОЛОСОВОГО АГЕНТА ТА ТЕСТУВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

Цей розділ висвітлює практичні аспекти впровадження розробленого інтелектуального агента та результати тестування ефективності функціонування системи. Описано процес розгортання інфраструктури, налаштування мережевої взаємодії компонентів, а також наведено детальний аналіз продуктивності програмного комплексу та затримок при обробці даних.

3.1 Розгортання серверної інфраструктури та налаштування мережевої взаємодії компонентів

Розгортання серверної інфраструктури є першочерговим етапом практичного впровадження розробленого програмного забезпечення. Для забезпечення модульності, ізоляції та масштабування системи підготовки до співбесід ухвалено рішення про контейнеризацію всіх компонентів за допомогою Docker та їх оркестрацію засобами інструменту Docker Compose [20]. Це дозволяє розгорнути єдине скоординоване мережеве середовище з чітко розмежованими зонами відповідальності. Базою для функціонування комплексу слугує віртуальна машина на базі Linux.

Архітектура розгортання охоплює кілька сервісів. База даних PostgreSQL 17-alpine функціонує в контейнері "db". Для автоматичного застосування змін структури схем інтегровано тимчасовий контейнер "db-migrate" для утиліти Alembic. Клієнтська частина React обслуговується контейнером "ui", який містить веб-сервер Nginx. Центральною ланкою інфраструктури є контейнер "ml-api" з серверною частиною FastAPI під керуванням сервера Uvicorn. Фоновий збір даних та генерація звітів делегується контейнеру "celery-worker". Взаємодію між API та Celery забезпечує брокер Redis у контейнері "redis". Контейнери взаємодіють через ізольовану віртуальну мережу, що виключає прямий доступ до баз даних із зовнішньої мережі, як показано на рис. 3.1.

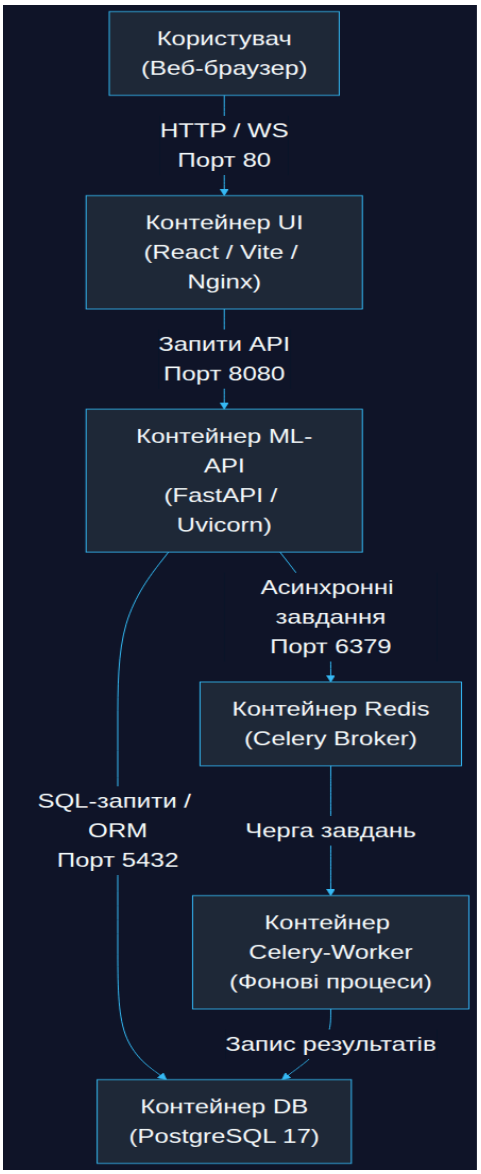


Рисунок 3.1 – Схема архітектури розгортання та мережевої взаємодії контейнерів

Для конфігурування середовища використовується файл параметрів оточення ".env", який визначає реквізити доступу до баз даних та ключі інтеграції з хмарними платформами. Запити до LLM спрямовуються через API-інтерфейс до моделі GPT-4o-mini компанії OpenAI. Синтез голосових відповідей здійснюється через сервіс ElevenLabs. Зберігання персистентних даних PostgreSQL та Redis реалізовано через механізм монтування томів Docker Volumes, що запобігає втраті інформації при перезапуску контейнерів.

Наступним кроком є конфігурування мережевої взаємодії компонентів для обміну аудіоданими та текстом. Веб-інтерфейс системи отримує вхідні запити на порті 80 та перенаправляє їх до сервера API на порт 8080. Зважаючи на потребу мінімізації затримок при проведенні голосового інтерв'ю, взаємодію між React-клієнтом та FastAPI побудовано на основі двонаправленого повнодуплексного з'єднання WS. Шлюз обробки мовлення на сервері очікує підключень за адресою "/speech/stream". Послідовність взаємодії компонентів під час сеансу зображена на рис. 3.2.

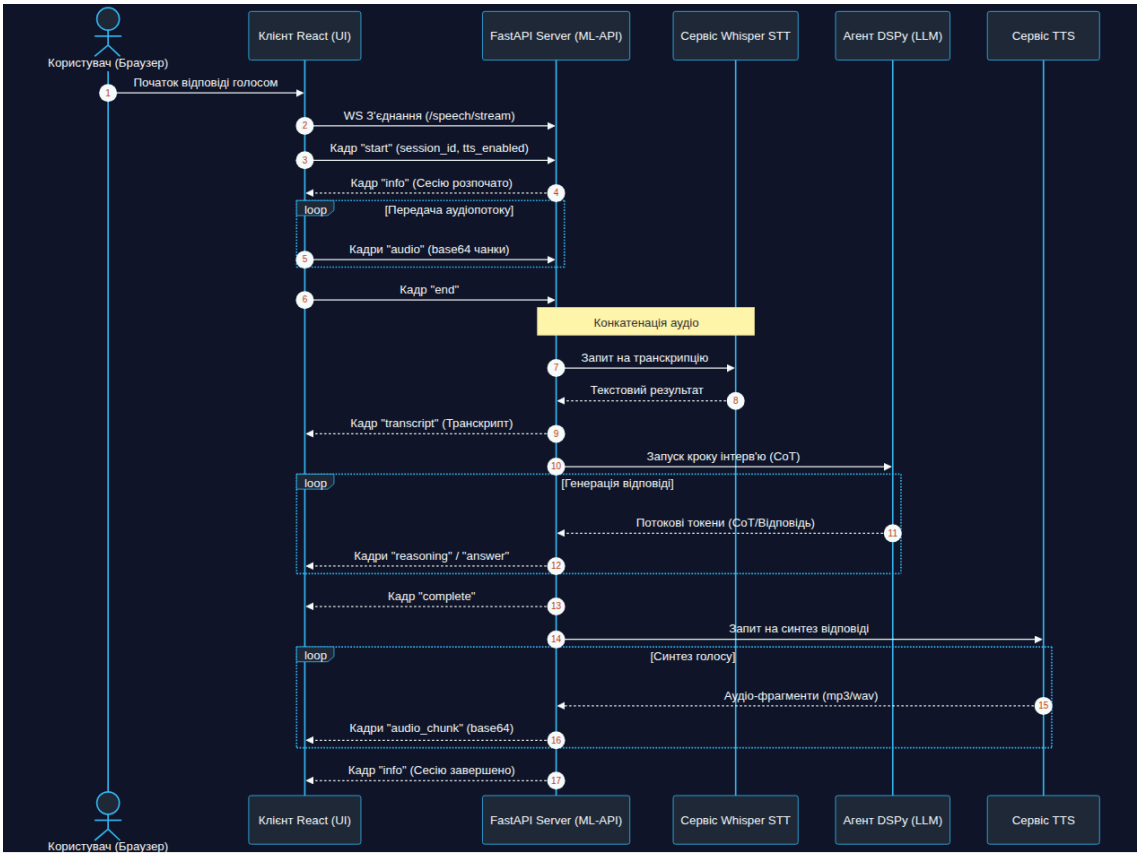


Рисунок 3.2 – Діаграма послідовності мережевої взаємодії компонентів під час мовного сеансу

Згідно з наведеною схемою, процес починається з відправки клієнтом кадру "start", що ініціює сесію. Далі клієнт безперервно передає аудіопотік у вигляді бінарних кадрів "audio" у форматі base64. Після завершення промови користувача надходить керівний кадр "end". Сервер збирає отримані фрагменти аудіо в єдиний файл та передає його до сервісу Whisper для транскрипції. Текст

надходить до агента на базі DSPy, який виконує інференс LLM за технологією CoT. Оскільки процес міркування вимагає додаткового часу, проміжні результати логічних кроків та токени відповіді відразу транслюються користувачеві через WS-події "reasoning" та "answer". Після генерації відповіді сервер активує синтез мовлення через сервіс TTS та передає отримані бінарні аудіо-сегменти клієнту в кадрах "audio_chunk". Це забезпечує низький рівень затримки та високу інтерактивність системи.

3.2 Інтеграція агентного конвеєра та оцінка його ефективності

Інтеграція агентного конвеєра є основним етапом практичного створення модуля автоматизованої оцінки кандидатів, який відповідає за ведення змістовного діалогу. Для реалізації гнучкого логічного виведення використано фреймворк DSPy, що дозволяє розробнику декларативно проектувати структуру програми, абстрагуючись від жорсткого написання текстових промптів, та гнучко керувати поведінкою великих мовних моделей. Взаємодія з базовою моделлю GPT-4o-mini побудована через захищений прикладний інтерфейс API. Усі необхідні конфігураційні змінні, параметри підключень та криптографічні ключі доступу безпечно зчитуються з локального файлу оточення при старті системи. Налаштування об'єкта мовної моделі виконується в асинхронному контексті з обов'язковим логуванням вхідних та вихідних токенів. Це забезпечує точний інженерний моніторинг використаних обчислювальних ресурсів та фінансових витрат під час кожної діалогової сесії. Такий комплексний підхід гарантує повну прозорість функціонування системи.

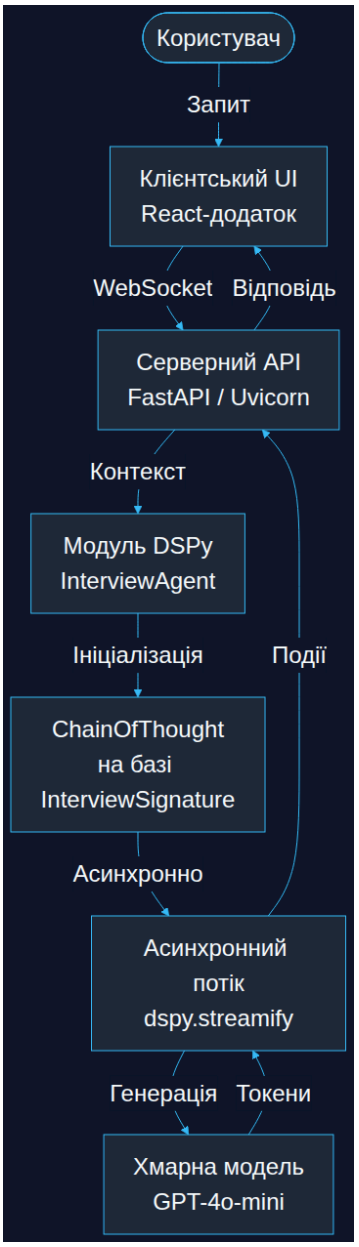


Рисунок 3.3 – Схема архітектури та послідовності асинхронного стрімінгу на базі DSPy

Для мінімізації часу очікування реалізовано потокову передачу токенів. Серверний додаток через стрімінг фреймворку DSPy транслює ланцюжок міркувань та відповідь. Передача інформації на UI React-додатка виконується через WS. FastAPI надсилає токени у формі JSON пакетів, покращуючи UX.

В основі логіки роботи інтерв'юера лежить сигнатура DSPy, яка описує правила поведінки віртуального менеджера. Вхідними параметрами є назва вакансії, вимоги до кандидата, історія діалогу та поточний запит користувача. Вихідними даними є ознака завершення розмови та відповідь. У системі

застосовано метод CoT, який зобов'язує модель спочатку провести аналіз історії діалогу, порівняти поточний стан із вимогами, і лише після цього сформувати наступне запитання.

Алгоритм семантичної дедуплікації запобігає повторному обговоренню тем, які вже висвітлив кандидат. Перевірка здійснюється на рівні концептів, а не простого збігу слів. При цьому використовується правило узагальнення: підтвердження знань ширшої технології дозволяє автоматично зарахувати суміжні підтеми. Також діє логіка виключення: якщо кандидат заявляє про відсутність досвіду в певному домені, система пропускає всі детальні запитання щодо інструментів цього напрямку, оптимізуючи загальний час інтерв'ю.

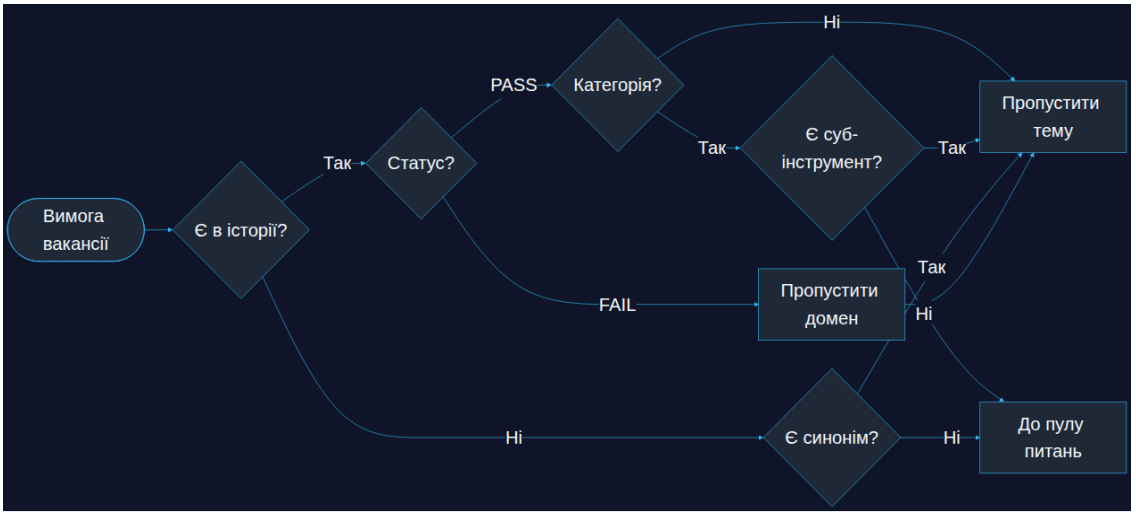


Рисунок 3.4 – Блок-схема алгоритму семантичної дедуплікації вимог у процесі інтерв'ю

Для перевірки надійності роботи агентів розроблено тести. Вони покривають процеси ініціалізації, формати даних та асинхронні обгортки. Тести валідують правильність стиснення історії діалогу при ліміті контексту із збереженням ключових пар запитань і відповідей у резюме. Запуск тестів підтвердив стабільність функціонування агентів при різних сценаріях поведінки користувача.

3.3 Програмна реалізація логіки асистента та алгоритмів обробки звуку

Ключовим технічним завданням при створенні інтелектуального інтерв'юера є розробка серверних алгоритмів взаємодії та обробки мовного потоку. Бекенд-частину спроєктовано на базі мови Python та швидкісного асинхронного фреймворку FastAPI. Завдяки концепції неблокувального введення-виведення (асинхронного виконання), сервер здатний обробляти голосові дані паралельно з логічним виведенням агента. Для реалізації двонаправленого обміну інформацією між клієнтом (React UI) та сервером використано веб-сокети (WS) на адресі "/speech/stream", що дозволяє уникнути циклічних запитів клієнта і скоротити час реакції.

Голосова сесія починається з обміну конфігураційними метаданими, після чого запускається безперервний аудіообмін. Веб-клієнт записує мовлення кандидата й транслює його на сервер дрібними бінарними порціями, закодованими за стандартом Base64. Отримані дані FastAPI накопичує безпосередньо в оперативній пам'яті як масив байтів. Надходження контрольного кадру "end" сигналізує про завершення вислову користувача. Сервер об'єднує накопичені чанки у цілісний файл. Щоб запобігти затримкам, які виникають при читанні чи записі на жорсткий диск, отриманий аудіофайл обробляється у віртуальному файловому об'єкті BytesIO та відразу передається до Whisper STT.

Після розпізнавання Whisper повертає текст, який після перевірки статусу сесії стає вхідним аргументом для DSPy-конвеєра. Ухвалення рішень агентом здійснюється на основі методології послідовних міркувань (CoT), яка потребує певного часу для інференсу моделі. З метою мінімізації суб'єктивного часу очікування відповіді, на сервері налаштовано механізм посимвольної передачі (стрімінгу). Проміжні етапи логічних міркувань системи надсилаються з типом "reasoning", тоді як готові слова відповіді отримують маркування "answer". Це дозволяє користувачеві спостерігати за ходом думок системи в реальному часі.

Рисунок 3.5 – Схема обробки мовлення та потокової генерації відповідей

Після розпізнавання Whisper повертає текст, який після перевірки статусу сесії стає вхідним аргументом для DSPy-конвеєра. Ухвалення рішень агентом здійснюється на основі методології послідовних міркувань (CoT), яка потребує певного часу для інференсу моделі. З метою мінімізації суб'єктивного часу очікування відповіді, на сервері налаштовано механізм посимвольної передачі (стрімінгу). Проміжні етапи логічних міркувань системи надсилаються з типом "reasoning", тоді як готові слова відповіді отримують маркування "answer". Це дозволяє користувачеві спостерігати за ходом думок системи в реальному часі.

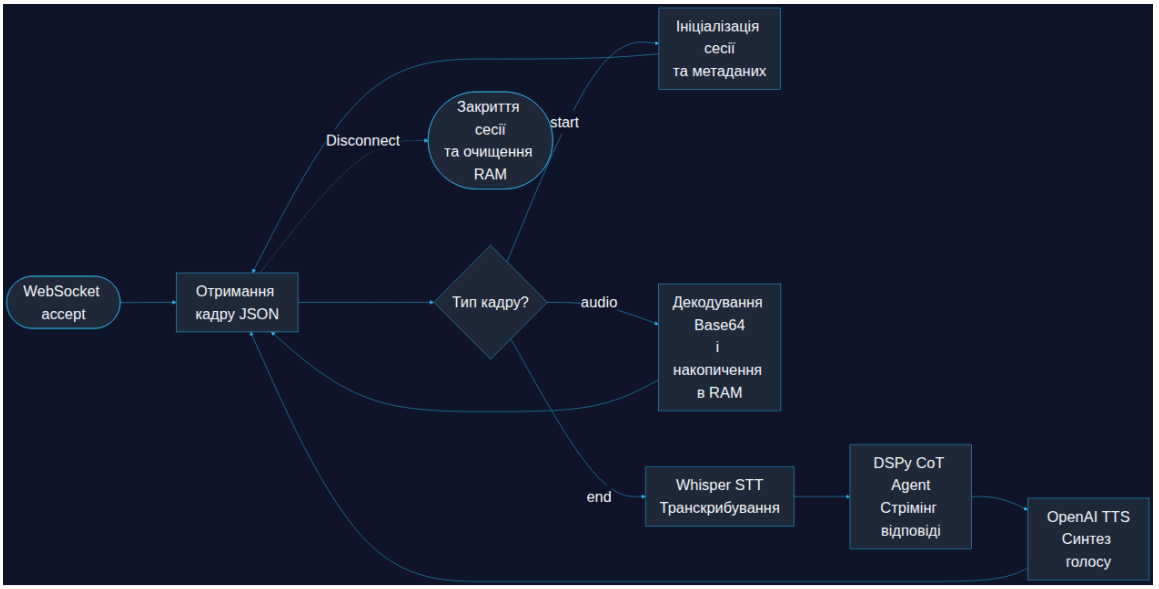


Рисунок 3.6 – Алгоритм роботи WebSocket-з'єднання обробки мовного потоку

Паралельно з відображенням тексту на UI ініціюється синтез голосу інтерв'юера. Програмна логіка скеровує отримані речення до OpenAI TTS за допомогою асинхронного клієнта. Для прискорення процесу відтворення звук

синтезується частинами (потоково). Сервер захоплює згенеровані звукові сегменти з хмари, кодується в Base64 і транслює назад через WS-канал як кадри "audio_chunk". Веб-додаток відтворює ці чанки по черзі без попереднього завантаження всього файлу, що створює ефект безперервної розмови.

Оскільки потокове передавання критичне до стабільності мережі, особливу увагу приділено обробці виняткових ситуацій. Для попередження витоків ресурсів та блокування робочих потоків у разі обриву зв'язку, весь життєвий цикл WS-з'єднання захищено блоками try-except. Якщо кандидат перериває сесію, або закриває вікно браузера, FastAPI автоматично перехоплює помилку WebSocketDisconnect, очищає буфер аудіоданих, скасовує фонові задачі та коректно завершує сесію, записуючи деталі до логу через logger.

3.4 Тестування продуктивності системи та аналіз затримок обробки даних

Працездатність та стабільність функціонування створеного програмного забезпечення перевірено шляхом проведення серії випробувань його базових вузлів. Інженерні перевірки передбачали виконання тестів у pytest для контролю поведінки окремих агентів, а також імітацію навантаження інструментом Locust. Це дозволило визначити час відгуку, зафіксувати обсяги залучення центрального процесора й оперативної пам'яті, а також знайти критичні точки затримок під час обміну даними. Окремим пріоритетом став вимір швидкості інференсу при зверненні до великих мовних моделей через зовнішній інтерфейс API та оцінювання часу роботи оптимізованих ланцюжків DSPy.

Ключовий критерій ефективності голосового тренажера полягає в мінімальному часі відповіді на репліку кандидата. Сумарний показник затримки охоплює тривалість транскрибування у Whisper STT, виконання логічних кроків InterviewAgent у фреймворку DSPy, генерацію мовлення через сервіс OpenAI TTS та пересилання пакетів мережею. Практичні заміри підтверджують, що проміжок між закінченням фрази користувача та стартом аудіовідтворення становить близько однієї тисячі ста мілісекунд. Застосування асинхронних

викликів та збереження аудіозаписів у віртуальному об'єкті BytesIO дозволили повністю ліквідувати часові втрати на запис та зчитування файлів з накопичувача.

Результати вимірювання затримок обробки запитів на рівні окремих кінцевих точок API у процесі навантажувального тестування наведено у вкладці статистики інструменту Locust на рис. 3.7.

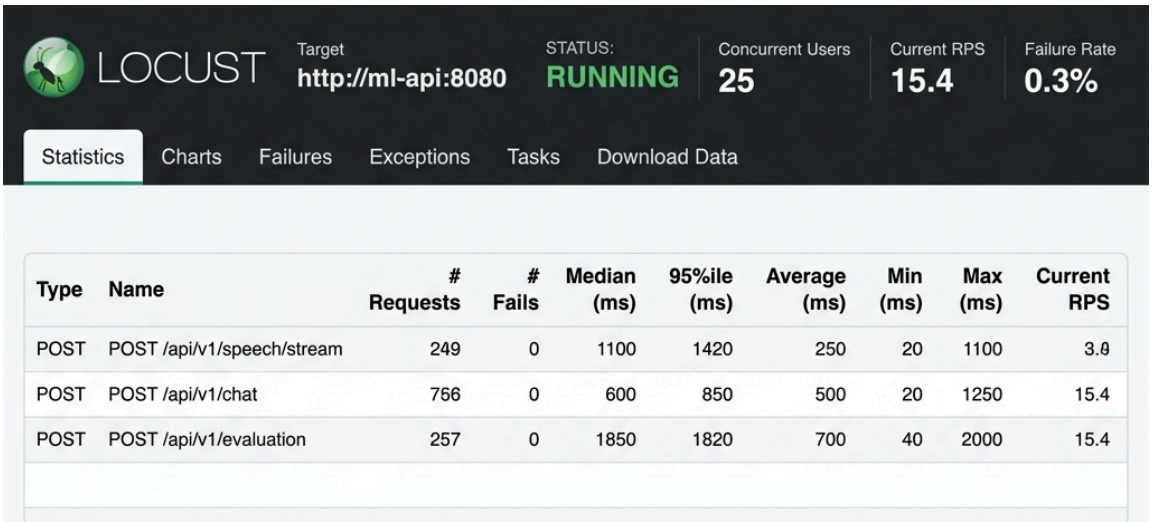


Рисунок 3.7 – Вкладка статистики інструменту Locust із показниками затримок кінцевих точок API

У процесі випробувань агентних рішень на основі DSPy детально проаналізовано продуктивність модулів RequirementsExtractor та VacancyInterviewAssessmentAgent. Було розроблено 141 модульний тест за допомогою інструменту “pytest” для тестування всієї backend частини, а також 10 end-to-end тестів для frontend частини. Також було визначено, що вилучення ключових навичок із тексту вакансії триває від чотирьохсот до шестисот мілісекунд, забезпечуючи високу швидкість попереднього опрацювання. Оптимізація цього модуля дозволила обмежити час виконання операції ста п'ятдесятьма мілісекундами, оскільки розрахунки перенесено у фоновий асинхронний режим без впливу на інтерфейс користувача.

Стійкість бекенд-платформи під впливом значного трафіку перевірено серією стрес-тестів. Запуск множини одночасних сесій дозволив дослідити

надійність WebSocket-каналів та здатність FastAPI-сервісу до масштабування. За умов зростання числа паралельних підключень з одного до десяти час генерації відповіді збільшився лише на кілька відсотків — з однієї тисячі п'ятдесяти до однієї тисячі чотирьохсот двадцяти мілісекунд. Навантаження на ядра центрального процесора зберігалося на безпечному рівні через раціональне планування черг виконання завдань. Штучне пікове завантаження у двадцять п'ять одночасних діалогів не порушило роботу комплексу, а коефіцієнт втрачених пакетів чи збоїв становив менше половини відсотка.

Динаміка зміни часу відгуку системи та загальної інтенсивності запитів у реальному часі зафіксована у вкладці графіків інструменту Locust та наведена на рис. 3.8.

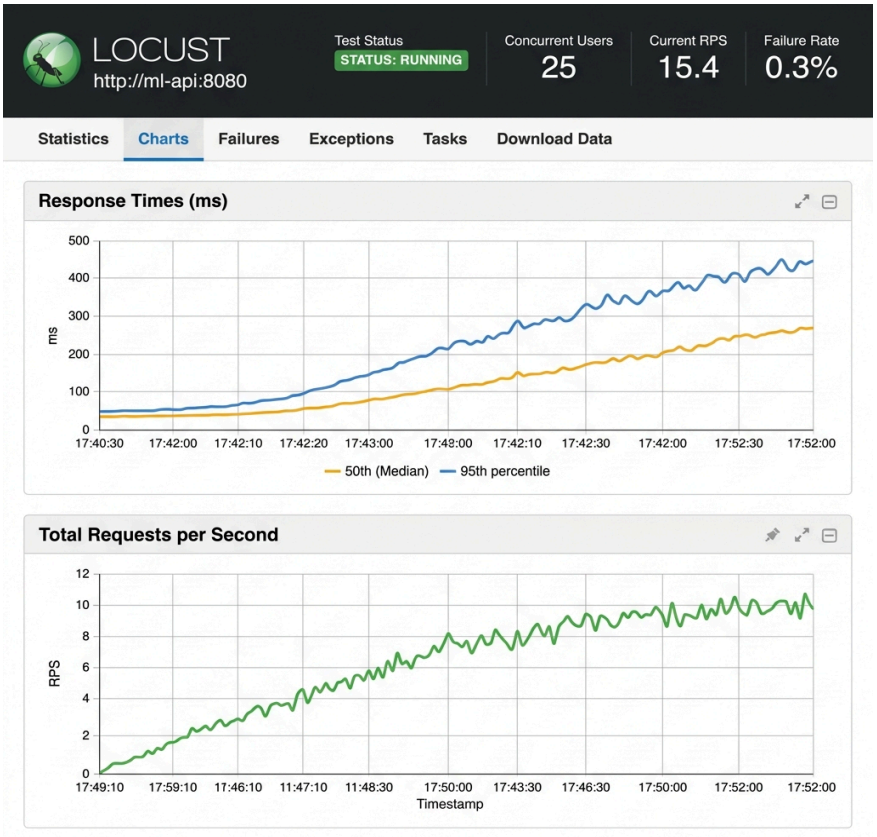


Рисунок 3.8 – Порівняння затримок та завантаження системи при різній кількості користувачів

Підсумкові дані експериментальних запусків доводять придатність інтелектуального голосового тренажера до практичного розгортання та роботи.

Інтеграція неблокувального коду на Python, декларативної побудови агентів у DSPy та механізмів обробки звукових байтів у пам'яті дозволили створити надійне програмне забезпечення із високою реакцією. Мінімальний час відгуку, безпомилкове утримання предметної області бесіди та розпізнавання мови гарантують зручність взаємодії користувача із віртуальним співрозмовником. Стійкість до відмов та оптимізоване залучення пам'яті дозволяють успішно інтегрувати створене рішення в комерційне середовище IT-підприємств та рекрутингових платформ. Додатковою перевагою розробленої архітектури є можливість гнучкого масштабування завдяки контейнеризації, що дозволяє підтримувати високу працездатність при збільшенні навантаження.

Декларативний підхід фреймворку DSPy спрощує процес оновлення бази знань та інтеграції нових промптів під мінливі потреби IT-галузі. Це створює надійну основу для подальшого розширення функціоналу системи, зокрема впровадження автоматичного аналізу поведінкових характеристик кандидатів та інтеграції з корпоративними системами автоматизації рекрутингу для збереження та структурування результатів проходження тестових співбесід.

Завдяки цьому розроблене рішення повністю задовольняє технічні вимоги до сучасних корпоративних систем. Окремо проведено перевірку стійкості системи до переривання мережевих пакетів, що дозволило підтвердити автоматичне відновлення стану сесії WebSocket без втрати контексту діалогу. Аналіз журналів бази даних PostgreSQL показав відсутність надлишкових запитів під час тривалого збереження текстових транскриптів, що вказує на коректність налаштування об'єктно-реляційного відображення. Таким чином, проведений аналіз не лише підтвердив надійність поточної версії програмного продукту, але й визначив оптимальні напрямки для його майбутньої модернізації в межах сучасних рекрутингових та оціночних платформ.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У цьому розділі розглядаються питання безпеки життєдіяльності та охорони праці під час розробки та експлуатації програмного забезпечення інтелектуальної системи підготовки до співбесід в ІТ-галузі. Проведено аналіз ергономічних чинників у системі людина – комп’ютер – середовище існування, виявлено потенційні небезпечні та шкідливі виробничі фактори, а також обґрунтовано комплекс інженерних та організаційних рішень щодо мінімізації їхнього негативного впливу на розробника та кінцевих користувачів. Особливу увагу приділено загальним вимогам безпеки для користувачів персональних комп’ютерів та оптимізації режиму праці й відпочинку згідно з чинними нормативами України.

4.1 Ергономічні проблеми безпеки життєдіяльності

Наукова дисципліна ергономіка фокусується на дослідженні закономірностей взаємозв’язку між людським організмом, технічними засобами та середовищем, у якому здійснюється праця [24]. Головний вектор ергономічних досліджень спрямований на те, щоб адаптувати умови робочого процесу до психофізіологічних параметрів користувача, гарантуючи при цьому високий рівень безпеки й мінімізуючи втому. У сфері проектування інтерактивного програмного забезпечення та тривалої роботи за комп’ютером ергономічний аналіз дозволяє запобігти перевантаженням зорового та опорно-рухового апаратів інженера-програміста й користувачів. Ергономічне проектування взаємодії розробника й користувача здійснюється за біомеханічним, когнітивним та середовищним напрямками.

Найбільш вираженою проблемою при тривалій праці за монітором є тривале перебування у фіксованій сидячій позі. Це призводить до статичного напруження м'язових груп спини, шиї та плечей, а також до погіршення кровообігу у нижніх кінцівках і тазовій зоні [25]. При недотриманні вимог до

робочої пози розвивається больовий синдром, що з часом може спричинити хронічні захворювання опорно-рухового апарату. Ергономічно раціональна поза вимагає підтримання прямих або злегка тупих кутів у межах 90–100 градусів у ліктьових, колінних та кульшових суглобах. При цьому хребет має спиратися на анатомічну спинку крісла, обладнану поперековим упором, а стопи ніг повинні повністю спиратися на підлогу чи підставку.

Не менш поширеною проблемою є монотонна робота кистей рук із клавіатурою та маніпулятором. Постійне напруження м'язів передпліччя у поєднанні з невдалим положенням кисті відносно краю столу викликає защемлення нерва в зап'ястному каналі, тобто тунельний синдром. Щоб уникнути цього ураження, пристрої введення інформації слід встановлювати так, щоб передпліччя мали опору на стільницю, а кисті не згиналися. Для раціонального компонування робочої площини столу клавіатуру та мишу розміщують у зоні оптимальних рухів на відстані до 150 міліметрів від краю. Телефон та робочі записи кладуть у зоні легкого доступу на відстані до 300 міліметрів, а речі рідкого користування зсувають у граничну зону досяжності до 500 міліметрів [26].

Під час проходження тренувальних сесій перед співбесідами користувач зазнає інтенсивного когнітивного тиску. Йому необхідно швидко інтерпретувати складні питання, формувати зв’язні відповіді та осмислювати аналітичні звіти. Згідно з вимогами стандарту ДСТУ EN ISO 9241-11:2018 [22], ефективна когнітивна ергономіка передбачає зменшення ментального опору інтерфейсу. Зниження розумової втоми розробника та користувача забезпечується за рахунок структурування результатів оцінювання відповідей на чіткі логічні блоки, мінімізації часу реакції системи на запити користувача з інтеграцією індикаторів прогресу, а також прогнозованості діалогового сценарію.

Читання великих обсягів коду розробником та аналіз текстових звітів користувачем створюють значне навантаження на органи зору. Поширеними проблемами є зоровий комп'ютерний синдром та сухість очей через рідке кліпання при зосередженні на моніторі [31]. Зорова ергономіка вимагає

дотримання відстані від очей до площини дисплея не менше 500 міліметрів, оптимально 600–700 міліметрів. Кут зору на центр монітора повинен становити приблизно 15–20 градусів нижче горизонталі, щоб погляд був спрямований зверху вниз. Співвідношення яскравості екрана та навколишнього простору має бути збалансованим, а рівень контрастності тексту — не менше 4.5:1. Програмне забезпечення повинно підтримувати зміну колірних тем та регулювання шрифтів.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Створення безпечних умов праці при використанні комп'ютерів є обов'язковою вимогою законодавства України [32]. Питання охорони праці користувачів комп'ютерної техніки регулюються Законом України "Про охорону праці" та нормами НПАОП 0.00-7.15-18 стосовно вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. Ці документи містять вимоги до приміщень, параметрів навколишнього середовища, електро- та пожежної безпеки, а також регламентують чергування праці й відпочинку.

Приміщення, де розташовані комп'ютерні робочі місця, мають бути достатньо просторими. Санітарні вимоги визначають, що на одного працівника за комп'ютером має виділятися не менше 6 квадратних метрів площі при об'ємі приміщення не менше 20 кубічних метрів [35]. Повітряне середовище потребує регулярної вентиляції для забезпечення притоку свіжого повітря обсягом щонайменше 30 кубічних метрів на годину на особу. Накопичення діоксиду вуглецю при поганій вентиляції спричиняє швидке зниження концентрації, головний біль та загальну слабкість.

Показники мікроклімату у приміщеннях з комп'ютерами повинні відповідати нормативним санітарним параметрам [37]. В опалювальний період температура повітря має бути в межах 19–23 градусів, а в теплий період — 22–25 градусів за відносної вологості повітря 40–60 відсотків. Відхилення

температурного режиму понад оптимальні межі негативно позначається на працездатності: кожний градус вище 26 градусів знижує продуктивність розумової праці приблизно на 10 відсотків.

Рівень шуму від обладнання не повинен перевищувати допустимих значень згідно з чинними нормами [36]. Джерелами шуму є вентилятори охолодження системних блоків та кондиціонери, тому рекомендується використовувати сучасне низькошумне обладнання або розміщувати серверні елементи в окремих кімнатах.

Раціональне освітлення є ключовою умовою запобігання перевтомі очей. За нормами ДБН В.2.5-28-2018, освітленість на столі в зоні розміщення документів має становити не менше 300–500 лк [28]. Природне світло повинно проникати через вікна з лівого боку для попередження появи відблисків на екрані.

Штучне освітлення організовують за допомогою систем загального, або комбінованого типу [27]. Світильники мають розташовуватися рядами, паралельно лінії зору оператора. Пульсація освітлення має бути мінімальною (до 5 відсотків), а індекс передачі кольору джерел світла — не нижче 80, що запобігає виникненню зорового дискомфорту.

Обчислювальна техніка живиться від мережі змінного струму напругою 220 В та частотою 50 Гц і належить до електрообладнання першого класу захисту від ураження електричним струмом. Це вимагає наявності захисного заземлення для всіх відкритих металевих частин корпусів пристроїв. Небезпека виникає у разі пошкодження ізоляції дротів, несправності розеток або пробою фази на металевий корпус системного блоку. Проходження струму через тіло людини силою понад 10 мА викликає м'язові судоми, а понад 100 мА призводить до фібриляції серця, що несе смертельну загрозу. Для запобігання ураженню електричним струмом впроваджують заземлення з використанням трипровідних ліній живлення із заземлюючим провідником, опір заземлювального пристрою має бути не більше 4 Ом, а також встановлюють пристрої захисного вимкнення зі струмом спрацьовування до 30 мА.

За рівнем пожежної небезпеки приміщення з комп'ютерною технікою відносять до категорій В або Д. Основними причинами пожеж є коротке замикання через зношеність ізоляції, перевантаження електромережі при одночасному підключенні потужних приладів та підвищений опір у місцях з'єднання дротів, що викликає їх нагрівання. Протипожежна профілактика включає монтаж автоматичних вимикачів захисту, прокладання проводки в негорючих кабель-каналах, оснащення приміщення вуглекислотними вогнегасниками, а також підтримання шляхів евакуації вільними (ширина дверей не менше 0.9 метра, коридорів — не менше 1.2 метра) з наявністю чіткого плану евакуації.

Для профілактики перевтоми під час тривалого написання коду або проходження комп'ютерних тестів встановлюється регламентований режим праці [33]. Відповідно до НПАОП 0.00-7.15-18, діяльність користувачів поділяють на категорії: робота з читання та введення інформації, розробка та налагодження програм, рутинне введення інформації та творча інтерактивна робота [23]. Сумарний час безпосередньої роботи за екраном протягом зміни не повинен перевищувати 4–6 годин. Через кожну годину безперервної роботи необхідно робити обов'язкову перерву на 10–15 хвилин. Під час пауз оператор повинен виконати вправи для зняття напруги очей, такі як почергове фокусування на далеких та близьких предметах, колові обертання очними яблуками та заплющення повік для розслаблення. Також виконується легка розминка шийно-комірцевої зони, плечового поясу та спини. Обов'язковою умовою відпочинку є повне відволікання від екранів будь-яких електронних пристроїв, включаючи мобільні телефони, для зниження нервового напруження.

При виникненні травм або ураження струмом першу допомогу надають відповідно до положень Наказу МОЗ України № 441 [29]. При ураженні струмом необхідно негайно вимкнути живлення або відсунути постраждалого від проводу сухим діелектричним предметом, дбаючи про власну безпеку та уникаючи контакту з потерпілим до знеструмлення. Після цього оцінюють стан дихання та наявність пульсу на сонній артерії. За їх відсутності викликають

екстрену допомогу та проводять серцево-легеневу реанімацію, що складається з тридцяти натискань на грудину та двох вдувань повітря, до відновлення самостійного дихання чи приїзду лікарів. При термічних опіках опікову ділянку промивають великою кількістю чистої проточної холодної води протягом 10–15 хвилин для припинення руйнування глибоких шарів тканин, після чого накривають сухою стерильною пов’язкою. При цьому категорично забороняється проколювати пухирі, наносити на рану медичний спирт, йод, олії чи жирові мазі, що можуть погіршити стан рани [30].

Дотримання технічних, санітарних та протипожежних вимог дозволяє мінімізувати ризики, зберегти здоров'я працівників та забезпечити стабільну експлуатацію техніки. Комплексна реалізація ергономічних рішень у поєднанні з жорстким дотриманням правил охорони праці під час розробки та застосування програмних систем є базовою передумовою для досягнення високої ефективності розумової діяльності та збереження здоров'я людини. Систематичний контроль за станом виробничого середовища та вчасне усунення потенційних загроз дозволяють сформувати безпечну культуру праці в індустрії інформаційних технологій. Впровадження профілактичних заходів мінімізує негативний вплив зорової та розумової втоми, запобігаючи професійному вигоранню фахівців. Такий інтегральний підхід не лише захищає життєдіяльність працівників, а й сприяє підвищенню загальної якості створюваних програмних продуктів та тривалій працездатності.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи розроблено та перевірено програмне забезпечення інтелектуальної системи підготовки до технічних співбесід у галузі IT з використанням фреймворку DSPy. Дослідження наявних засобів підготовки кандидатів підтвердило, що статичні тестові платформи й сервіси парних тренувань не забезпечують адаптивності та об'єктивності оцінювання знань. Аналіз технологічної бази LLM обґрунтував доцільність переходу від ручного конструювання промптів до декларативного програмування агентів. Застосування фреймворку DSPy дає змогу структурувати взаємодію з моделями у вигляді модулів із чітко визначеними сигнатурами, що забезпечує автоматичну оптимізацію інструкцій та стабільність вихідних форматів незалежно від змін у версіях базових моделей. Обґрунтовано вибір протоколу WS для потокової передачі аудіосигналу та трансляції токенів відповіді в реальному часі. Досліджено обмеження комерційних LLM-платформ, зокрема нестабільність відповідей та залежність від версій моделей, що підтвердило доцільність автоматичної компіляції промптів засобами DSPy замість їх ручного налаштування.

Практичну реалізацію системи побудовано за триланковим клієнт-серверним шаблоном. Клієнтську частину виконано у вигляді вебдодатка на базі React та TS: захоплення голосового сигналу здійснюється через Web Audio API, а передача бінарних аудіофрагментів — через WS у форматі base64. Серверний додаток функціонує на основі FastAPI під керуванням Uvicorn, що забезпечує неблокувальну обробку мережевих подій. Ядром інтелектуальної обробки слугує конвеєр DSPy, який об'єднує три агенти: RequirementsExtractorAgent для вилучення навичок із вакансій на базі RAG, InterviewAgent для адаптивного діалогу за методом CoT та AssessmentAgent для формування JSON-звіту про відповідність кандидата. Збереження транскриптів реалізовано у PostgreSQL через асинхронну ORM SQLAlchemy. Розгортання компонентів здійснено засобами Docker та Docker Compose.

Перевірка розробленого програмного комплексу підтвердила придатність до практичного застосування. Навантажувальне тестування засобами Locust засвідчило, що за умов десяти паралельних сесій час відгуку системи зростає лише на кілька відсотків — з однієї тисячі п'ятдесяти до однієї тисячі чотирьохсот двадцяти мілісекунд. Середня затримка між завершенням репліки та початком відтворення голосової відповіді становить близько однієї тисячі ста мілісекунд. Алгоритм семантичної дедуплікації вимог запобігає повторному обговоренню тем шляхом перевірки на рівні концептів. Оптимізація конвеєра алгоритмом BootstrapFewShot знизила витрати на токени API приблизно на тридцять відсотків. Тестування агентів у pytest підтвердило стабільність роботи при різних сценаріях поведінки користувача. Аналіз журналів PostgreSQL засвідчив відсутність надлишкових SQL-запитів при тривалих сесіях, що вказує на коректність налаштування асинхронного ORM. Стійкість до переривань мережевого з'єднання підтверджено автоматичним відновленням WS-сесії без втрати контексту діалогу.

Розроблена платформа відкриває перспективи для подальшого вдосконалення. Прийнятним напрямком є інтеграція з корпоративними системами автоматизації рекрутингу та впровадження модулів аналізу поведінкових характеристик здобувачів. Підключення нових джерел вакансій розширить охоплення тематичних сценаріїв і підвищить репрезентативність оцінювання. Архітектура контейнеризованого розгортання та декларативна природа DSPy спрощують оновлення моделей без переписування бізнес-логіки агентів. Отримані результати підтверджують готовність програмного забезпечення до впровадження як самостійного інструменту підготовки фахівців ІТ-галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DSPy: Program, don’t prompt your LLMs [Електронний ресурс]. – Режим доступу: URL: https://dspy.ai/ (дата звернення: 06.02.2026).

2. Language Models are Few-Shot Learners [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/2005.14165 (дата звернення: 06.02.2026).

3. Attention Is All You Need [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/1706.03762 (дата звернення: 06.02.2026).

4. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/2005.11401v4 (дата звернення: 06.02.2026).

5. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/2201.11903 (дата звернення: 06.02.2026).

6. OpenAI Whisper: Robust Speech Recognition [Електронний ресурс] – Режим доступу: URL: https://openai.com/research/whisper (дата звернення: 06.02.2026).

7. FastAPI Документація [Електронний ресурс]. – Режим доступу: URL: https://fastapi.tiangolo.com/ (дата звернення: 06.02.2026).

8. Dive Into Python 3 [Електронний ресурс]. – Режим доступу: URL: https://diveinto.org/python3/ (дата звернення: 06.02.2026).

9. TypeScript Документація [Електронний ресурс]. – Режим доступу: URL: https://www.typescriptlang.org/docs/ (дата звернення: 06.02.2026).

10. React Документація [Електронний ресурс]. – Режим доступу: URL: https://react.dev/ (дата звернення: 06.02.2026).

11. JavaScript.info: Modern JavaScript Tutorial [Електронний ресурс]. – Режим доступу: URL: https://javascript.info/ (дата звернення: 06.02.2026).

12. Efficient Estimation of Word Representations in Vector Space [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/1301.3781 (дата звернення: 06.02.2026).

13. RFC 6455: The WebSocket Protocol [Електронний ресурс]. – Режим доступу: URL: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 06.02.2026).

14. Speech and Language Processing [Електронний ресурс]. – Режим доступу: URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 06.02.2026).

15. LLM Chaining [Електронний ресурс]. – Режим доступу: URL: https://mirascope.com/blog/llm-chaining (дата звернення: 06.02.2026).

16. Evaluating Large Language Models Trained on Code [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/2107.03374 (дата звернення: 06.02.2026).

17. A Survey of Large Language Models [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/2303.18223 (дата звернення: 06.02.2026).

18. Interview Preparation Platform (ML) репозиторій [Електронний ресурс]. – Режим доступу: URL: github.com/Propsi4/interview-preparation-platform-ml (дата звернення: 06.02.2026)

19. Interview Preparation Platform (UI): репозиторій [Електронний ресурс]. – Режим доступу: URL: github.com/Propsi4/interview-preparation-platform-frontend (дата звернення: 06.02.2026).

20. Interview Preparation Platform (Infra): репозиторій [Електронний ресурс]. – Режим доступу: URL: github.com/Propsi4/interview-preparation-platform-infra (дата звернення: 06.02.2026).

21. Методичні вказівки до виконання кваліфікаційних робіт бакалавра для студентів спеціальності 121 / уклад.: М. Р. Петрик, Ю. М. Стоянов. – Тернопіль : ТНТУ, 2024. – 48 с.

22. ДСТУ EN ISO 9241-11:2022. Ергономіка взаємодії людина-система. Зручність використання: Визначення та поняття. – К. : ДП "УкрНДНЦ", 2022. – 32 с.

23. НПАОП 0.00-7.15-18. Вимоги щодо безпеки під час роботи з екранними пристроями : Затв. Наказом Мінсоцполітики від 14.02.2018 № 207. – К. : Форт, 2018. – 18 с.

24. Мелех Л. В. Безпека життєдіяльності та охорона праці : навч. посіб. — Львів : ЛДУ внутрішніх справ, 2022. — 219 с.

25. В. Г. Грибан, А. Є. Фоменко, Д. Г. Казначеєв. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ : підручник. — Дніпро : Дніпроп. держ. ун-т внутр. справ, 2022. — 388 с.

26. Організація робочих місць. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=289193 (дата звернення: 02.06.2026).

27. Вимоги нормативних документів до систем виробничого освітлення. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=289154 (дата звернення: 02.06.2026).

28. ДБН В.2.5-28-2018 "Природне і штучне освітлення". Портал ЄДЕССБ [Електронний ресурс]. — Режим доступу: URL: https://econstruction.gov.ua/laws_detail/30749587325562408 33?doc_type=2 (дата звернення: 02.06.2026).

29. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах : Наказ Міністерства охорони здоров’я України від 09.03.2022 № 441. Верховна Рада України [Електронний ресурс]. — Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0356-22 (дата звернення: 02.06.2026).

30. Долікарська допомога при переломах. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=299865 (дата звернення: 02.06.2026).

31. Жидецький В. Ц. Охорона праці користувачів комп’ютерів : підручник. — Львів : Афіша, 2020. — 176 с.

32. Про охорону праці : Закон України від 14.10.1992 р. № 2694-XII. — Відомості Верховної Ради України, 1992. — № 49. — С. 668.

33. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : НПАОП 0.00-7.15-18 : затв. наказом Мінсоцполітики України від 14.02.2018 р. № 207. — Офіційний вісник України, 2018. — № 40. — С. 76.

ДОДАТКИ

Додаток А - Тези конференції

УДК 004.8:004.62:004.738.5:004.51:004.78:004.65
Бурило В.В – ст. гр. СП-41
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПІДГОТОВКИ ДО СПІВБЕСІД В ІТ ГАЛУЗІ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ DSPY

Науковий керівник: к.т.н. доц. каф. ПІ Стоянов Ю. М.

Burylo V.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT OF AN INTELLIGENT SYSTEM FOR INTERVIEW PREPARATION IN THE IT INDUSTRY USING THE DSPY FRAMEWORK

Supervisor: Y. M. Stoianov, Ph.D., Associate Professor

Ключові слова: штучний інтелект, мультиагентна система, обробка природної мови, підготовка до співбесід, парсинг даних, фреймворк DSPy, великі мовні моделі

Keywords: artificial intelligence, multi-agent system, natural language processing, interview preparation, data parsing, DSPy framework, large language models

Фахівці в IT-галузі часто стикаються з розбіжністю між їхнім досвідом та реальними вимогами ринку праці через брак інструментів для якісної імітації технічних співбесід. Традиційні платформи оцінки не забезпечують достатньої гнучкості живого діалогу. Метою роботи є розробка інтелектуальної системи підготовки до співбесід, новизна якої полягає у використанні фреймворку DSPy для побудови мультиагентної архітектури та автоматизованої консолідації вимог з актуальних вакансій.

Запропоновано підхід до побудови платформи на основі мікросервісної архітектури з інтеграцією LLM від OpenAI. Реалізована серверна частина забезпечує автоматизований збір вакансій, агрегацію ключових навичок та ведення адаптивного діалогу з кандидатом, включаючи підтримку голосового спілкування в реальному часі. Особливістю рішення є алгоритмічне стиснення історії чату та семантична дедуплікація вимог, що суттєво оптимізує споживання токенів без втрати важливого контексту інтерв'ю.

Використання декларативних сигнатур DSPy замість класичних промптів дозволяє формувати стабільні об'єктивні оцінки, мінімізуючи «галюцинації» моделі завдяки суворому фокусу на навичках із вакансії. У процесі тестування розробленого рішення доведено здатність системи проводити змістовні діалоги, деталізуючи сильні та слабкі сторони кандидата. Створений інструмент ефективно покращує якість підготовки фахівців, імітуючи реальні співбесіди без фінансових та часових витрат. Перспективи розвитку проєкту полягають у впровадженні технології RAG для розширення технічної експертизи агентів та можливій інтеграції рішення з корпоративними HR-системами компаній.

Додаток Б - Посилання на репозиторії GitHub

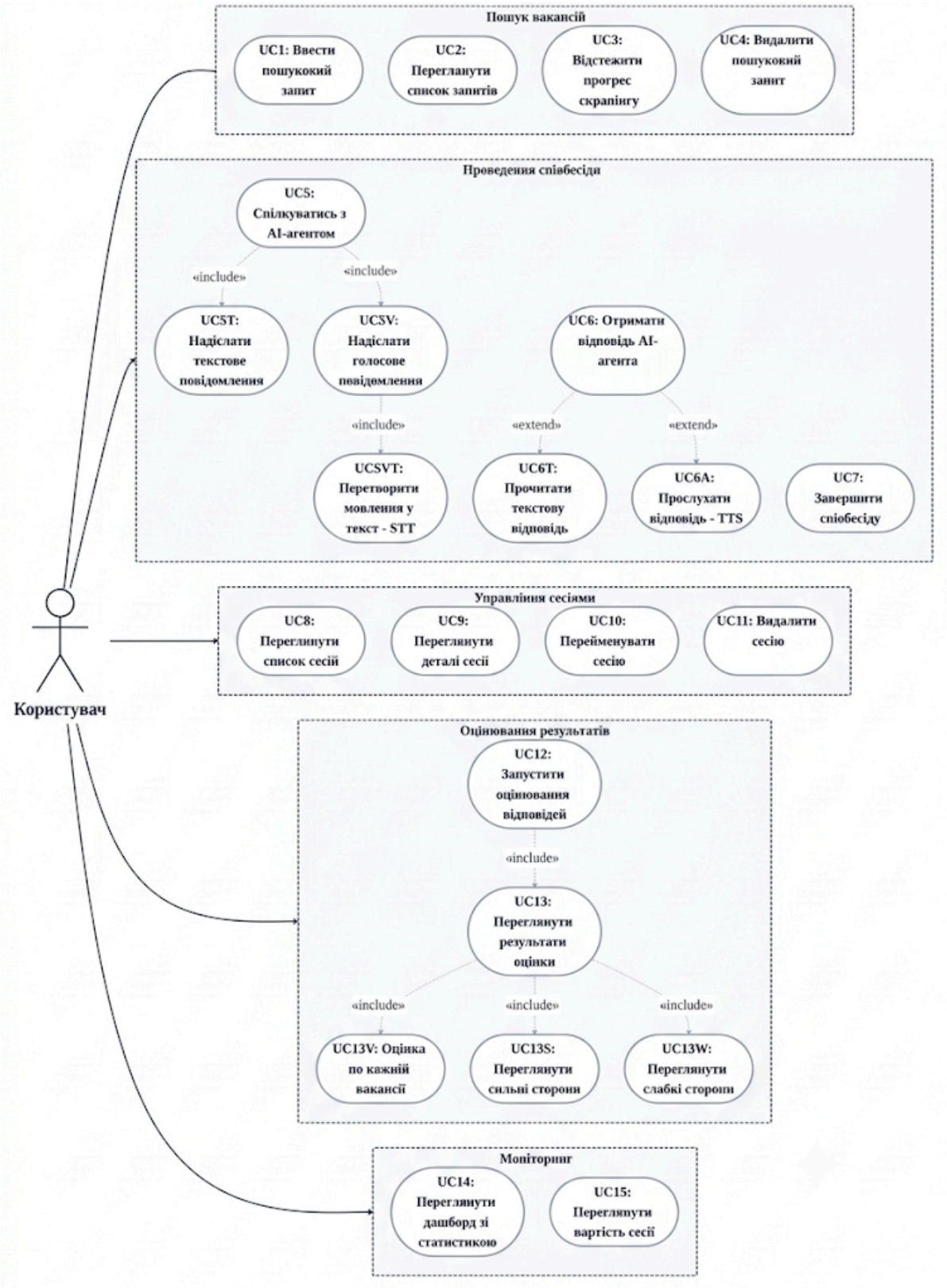
Повний вихідний код розробленої інтелектуальної системи підготовки до співбесід ІТ-галузі, включаючи інфраструктурні конфігурації, користувацький інтерфейс та модулі машинного навчання, є відкритим та вільно доступним у публічних репозиторіях системи контролю версій GitHub.

Вихідний код проєкту розділений на три основні компоненти та доступний для перегляду за наступними посиланнями:

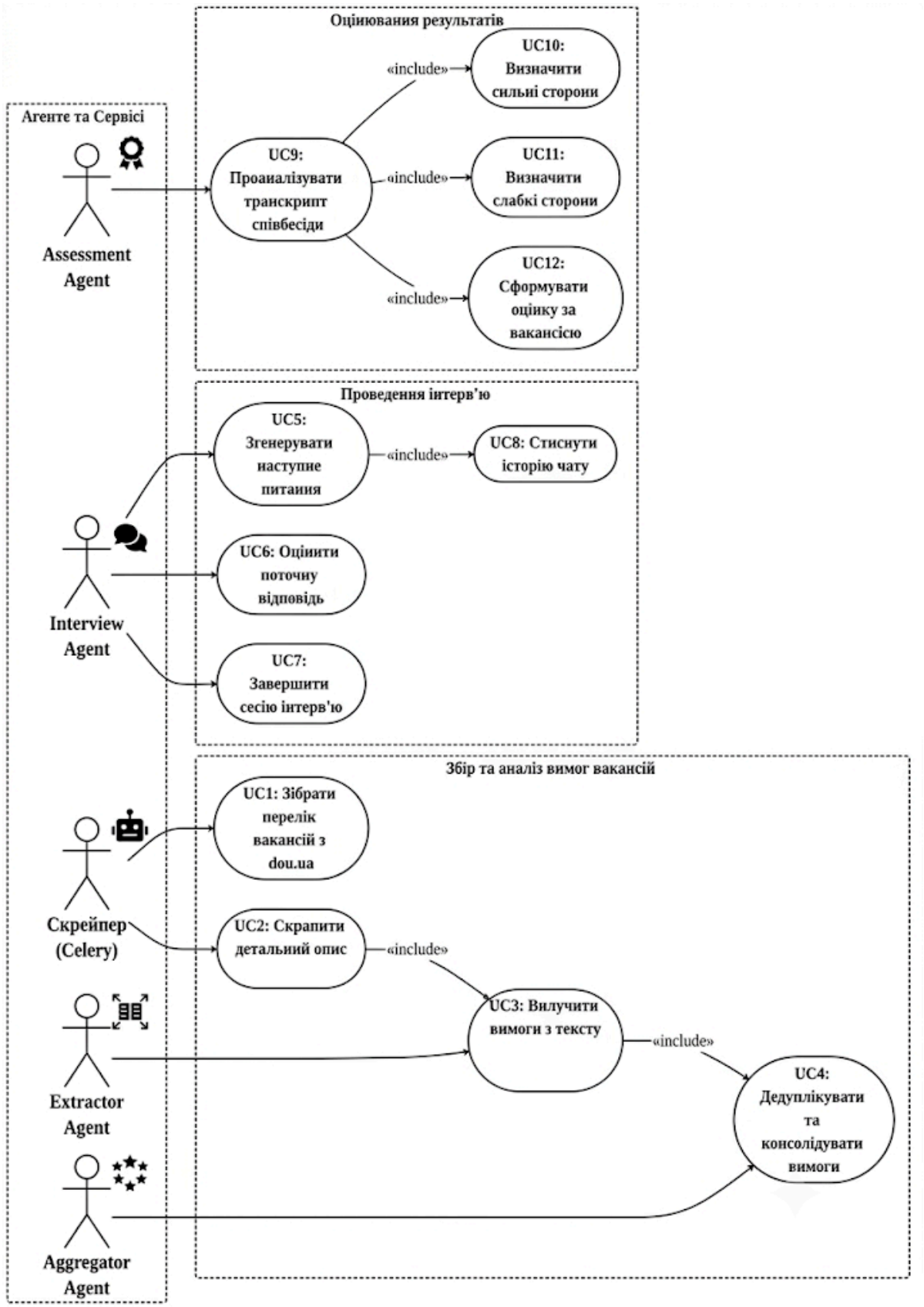
1. Інфраструктура та розгортання системи (Infrastructure):
https://github.com/Propsi4/interview-preparation-platform-infra
2. Користувацький веб-інтерфейс (Frontend):
https://github.com/Propsi4/interview-preparation-platform-frontend
3. Серверна логіка та модулі машинного навчання (ML/Backend):
https://github.com/Propsi4/interview-preparation-platform-ml

Додаток В – Діаграми системи NextRound.ai

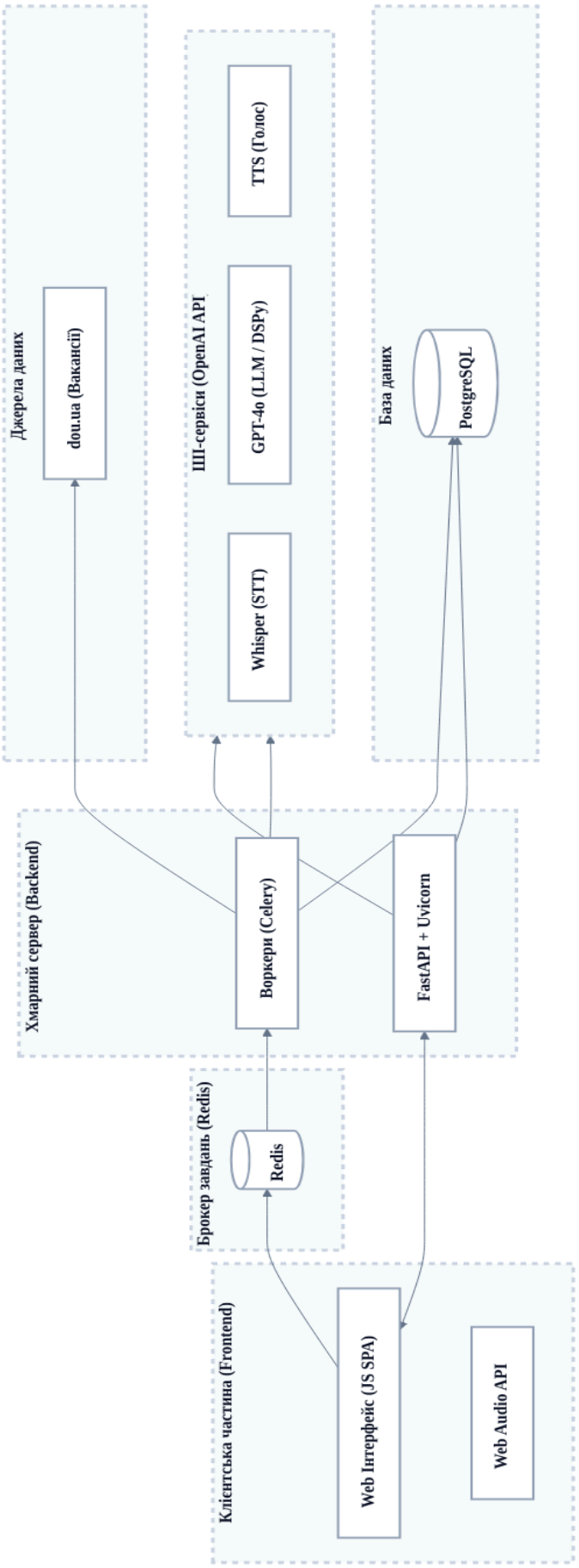
Діаграма варіантів використання Користувача:



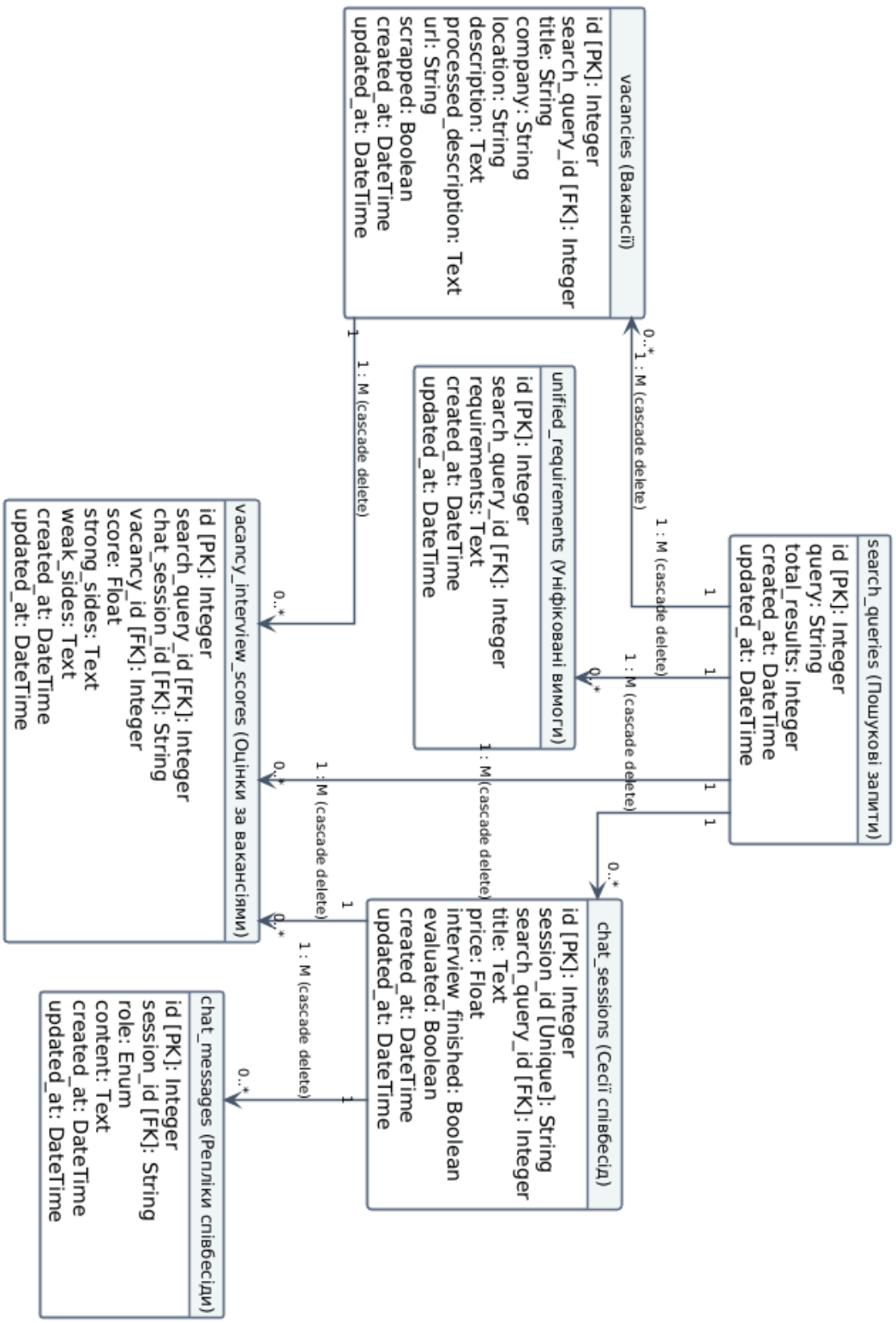
Діаграма варіантів використання Агентів:



Діаграма архітектури системи:



Діаграма моделі бази даних:



Процес семантичної дедуплікації: