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 рівня бакалавр
Бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення автоматичного Аналізу мовлення та адаптивного мовного навчання на основі методів обробки природної мови

Виконав: студент 4 курсу, групи СП-41
спеціальності 121
Інженерія програмного забезпечення
(шифр і назва спеціальності)

Апостол І.О
(підпис) (прізвище та ініціали)

Керівник Петрик М.Р.
(підпис) (прізвище та ініціали)

Нормоконтроль
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.
(підпис) (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)
« » 20___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 Апостолу Ігорю Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення автоматичного аналізу мовлення та адаптивного мовного навчання на основі методів обробки природної мови

Керівник роботи Петрик Михайло Романович, доктор фізико-математичних наук, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Потреба замовника, вимога замовника

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

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
	методів адаптивного навчання, систем інтервального	15.04.2026	
	повторення, LLM та RAG-архітектур для лінгвістичного		
	аналізу		
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	21.04.2026	
5	Реалізація програмного забезпечення адаптивного	21.04.2026 -	Виконано
	мовного навчання та аналізу мовлення	30.04.2026	
6	Тестування функціоналу системи, оцінка точності	30.04.2026 -	Виконано
	розпізнавання вимови та валідація відповідей AI-агента	1.05.2026	
7	Написання розділів пояснювальної записки (1–2 розділи)	1.05.2026 - 4.05.2026	Виконано
8	Написання розділу 3: «Безпека життєдіяльності та	5.05.2026 -	Виконано
	основи охорони праці»	6.05.2026	
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 _____ Апостол І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Апостол І. О. Розробка програмного забезпечення автоматичного аналізу мовлення та адаптивного мовного навчання на основі методів обробки природної мови, ТНТУ ім. Івана Пулюя, Тернопіль 2026.

Пояснювальна записка містить: 82 сторінок, 45 рисунків, 33 джерел та 2 додатків.

Об'єкт дослідження: процес проектування, розробки та інтеграції програмного забезпечення для гейміфікованого адаптивного навчання та автоматичного аналізу усного і текстового мовлення користувачів у режимі реального часу.

Мета: розробити комплексну інтелектуальну програмну платформу для вивчення мови, яка поєднує функції персонального AI-компаньйона, модулі автоматичного аналізу вимови та гейміфіковані алгоритми інтервального повторення для підвищення ефективності засвоєння знань.

Дослідження присвячено вирішенню проблеми персоналізації мовної освіти через впровадження інтелектуальних інструментів автоматизації фідбеку. Традиційні EdTech-рішення позбавлені гнучкого контекстуального аналізу розмовних навичок користувача, що обґрунтовує доцільність розгортання великих мовних моделей (LLM). У межах роботи розроблено кросплатформене ПЗ, яке містить гейміфікований блок із 12 інтерактивних ігор, систему накопичення досвіду (XP) та цифрову аналітику індивідуального прогресу. Особливу увагу приділено побудові дворівневого іспиту (написання та вокалізація термінів) із циклічним інтервальним розподілом за методикою Leitner. Технічний контур системи включає модулі STT/TTS для верифікації вимови, автономного AI-агента для виправлення помилок, а також панель адміністрування з RAG-архітектурою та доступом до веб-ресурсів для обробки користувацьких звернень і менеджменту знань.

Ключові слова: обробка природної мови, аналіз мовлення, адаптивне навчання, AI-компаньйон, великі мовні моделі (LLM), RAG-агент, гейміфікація, розпізнавання мовлення (STT), синтез мовлення (TTS), Python.

ABSTRACT

Apostol I. O. Development of Software for Automatic Speech Analysis and Adaptive Language Learning Based on Natural Language Processing Methods. - Manuscript. Bachelor's thesis in Software Engineering - Ternopil Ivan Puluj National Technical University, Ternopil, 2026.

The explanatory note contains: 82 pages, 45 figures, 33 sources, 2 appendices.

Object of research: the architecture and algorithms of interactive foreign language learning software environments based on real-time linguistic analysis of oral and textual data.

Objective: design and software implementation of a gamified learning ecosystem integrated with an AI mentor, speech recognition mechanisms, and dynamic vocabulary spaced repetition cycles.

The study addresses the issue of personalizing language education through the implementation of intelligent feedback automation tools. Traditional EdTech solutions lack flexible contextual analysis of the user's speaking skills, which justifies the feasibility of deploying Large Language Models (LLMs). Within the framework of this thesis, cross-platform software has been developed, which includes a gamified block of 12 interactive games, an experience point (XP) accumulation system, and digital analytics of individual progress. Special attention is paid to constructing a two-level examination system (spelling and vocalization of terms) with a cyclic spaced repetition distribution based on the Leitner system. The technical framework of the system includes STT/TTS modules for pronunciation verification, an autonomous AI agent for error correction, as well as an administrative panel based on RAG architecture with web-access capabilities for handling user inquiries and knowledge management.

Keywords: natural language processing, speech analysis, computer linguistics, AI agent, RAG technology, educational gamification, spaced repetition, speech recognition, Python.

ПЕРЕЛІК СКОРОЧЕНЬ

AI (Artificial Intelligence) - Штучний інтелект, комп’ютерні системи для імітації когнітивних функцій людини.

API (Application Programming Interface) - Програмний інтерфейс взаємодії між прикладними компонентами.

БД - База даних, впорядкована сукупність структурованої інформації в комп'ютерній системі.

JSON (JavaScript Object Notation) - Універсальний текстовий формат для обміну та структурування даних.

LLM (Large Language Model) - Велика мовна модель, глибока нейромережа для генерації та аналізу текстів.

NLP (Natural Language Processing) - Обробка природної мови, сфера на стику штучного інтелекту та лінгвістики.

RAG (Retrieval-Augmented Generation) - Методологія генерації відповідей ШІ з динамічним залученням зовнішніх баз знань.

STT (Speech-to-Text) - Програмна технологія транскрибації звукового сигналу людського мовлення в текст.

TTS (Text-to-Speech) - Інструмент штучного синтезування голосу на основі текстових вхідних файлів.

UI/UX (User Interface / User Experience) - Комплекс проектування візуальної форми та логіки взаємодії з користувачем.

UUID (Universally Unique Identifier) - Стандарт ідентифікації об'єктів у розподілених обчислювальних системах.

XP (Experience Points) - Одиниця виміру прогресу та ігрового досвіду користувача в освітньому процесі.

ЗМІСТ

ВСТУП

Проблема мовного бар'єра залишається однією з найбільших перешкод під час самостійного вивчення іноземної мови, оскільки більшість сучасних інтернет-ресурсів фокусуються лише на статичних вправах і не надають користувачам реальної розмовної практики. Вирішення цієї проблеми вбачається в інтеграції великих мовних моделей, які здатні виступати у ролі віртуальних текстово-голосових компаньйонів і забезпечувати миттєвий аналіз помилок у безстресовому для студента середовищі. З огляду на це, метою кваліфікаційної роботи є проєктування, програмна реалізація та хмарне розгортання інтелектуальної навчальної вебплатформи FLUEN AI, яка поєднує в собі розумного ШІ-співрозмовника, гейміфікований контур засвоєння лексики за алгоритмом Лейтнера та автоматизований кабінет вчителя на основі RAG-технології.

Для досягнення поставленої мети було сформульовано і послідовно вирішено низку інженерних завдань. Спочатку виконано системний аналіз предметної області та чинних мовних сервісів, після чого спроєктовано загальну архітектуру додатка та реляційну схему бази даних PostgreSQL із розширенням pgvector. Далі розроблено модулі автентифікації та особистого кабінету користувача з інтеграцією брокера черг Apache Kafka для асинхронного збору аналітики активності. На наступних етапах проведено інтеграцію ШІ-компаньйона на базі моделей GPT-4o та Whisper для ведення діалогів і генерації структурованого зворотного зв'язку, а також реалізовано кабінет ШІ-вчителя з динамічним доступом до веб-ресурсів. На завершення створено ігровий блок із дванадцяти мініігор, налаштовано логіку двоетапних іспитів, розроблено панель адміністратора з системою тикетів, здійснено контейнеризацію всієї системи через Docker для її деплою на сервіс Render, а також виконано комплексне модульне та наскрізне тестування створеного програмного продукту.

Об'єктом дослідження виступає процес автоматизації адаптивного навчання та виконання лінгвістичного аналізу мовлення на інтерактивних вебплатформах. Предметом дослідження визначено архітектурні підходи розробки мовних сервісів,

алгоритми штучного інтелекту, методи асинхронного збору інженерних метрик та ігрові механіки засвоєння лексики. Під час виконання роботи за основу було взято методи системного аналізу, принципи асинхронного програмування, інженерію підказок, технології векторного семантичного пошуку та методології гейміфікації освітнього процесу.

Практичне значення отриманих результатів полягає у створенні повністю готової до промислової експлуатації, контейнеризованої програмної системи, успішно розгорнутої у хмарі Render. Створена платформа забезпечує безперервну розмовну практику, автоматичний розбір граматичних конструкцій, гнучке управління векторною базу знань та CRM-аудит активності користувачів. Цей продукт розроблено як масштабоване EdTech-рішення, яке можна застосовувати як для індивідуальної самоосвіти, так і для інтеграції в діючу структуру мовних шкіл з метою модернізації та оптимізації навчання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перед тим як переходити до безпосереднього проєктування та програмної реалізації системи, потрібно детально дослідити теоретичні підвалини та специфіку обраної предметної області. У цьому розділі зосереджено увагу на вивченні ключових чинників, які заважають ефективному засвоєнню іноземної мови, а також аналізуються реальні запити потенційних користувачів платформи. Це дозволяє чітко обґрунтувати вибір майбутнього інженерного та технологічного інструментарію.

1.1 Проблематика подолання мовного бар'єра та аналіз потреб користувачів

Сучасний етап глобалізації та інтеграції інформаційних просторів висуває високі вимоги до рівня володіння іноземними мовами, серед яких англійська займає провідну роль інструменту міжнародної комунікації. Попри значну кількість теоретичних матеріалів, відкритих словників та інтерактивних платформ, більшість осіб, які самостійно вивчають мову, стикаються із серйозною деструктивною проблему, відомою як мовний бар'єр. Цей феномен має комплексну природу, що поєднує у собі психологічні, когнітивні та лінгвістичні чинники, які активно досліджуються в межах сучасних EdTech-методологій побудови адаптивних систем навчання та автоматизації розмовних навичок [1].

Основний парадокс мовного бар'єра полягає в тому, що користувач може володіти широким лексичним запасом і безпомилково виконувати складні письмові граматичні тести, проте опиняється у стані ступору при спробі побудувати спонтанне усне висловлювання. Психологічна складова проблеми зумовлена страхом зробити помилку, почути критику або виглядати некомпетентно перед співрозмовником. З когнітивного погляду проблема полягає у відсутності автоматизму формування фраз, коли людина витрачає занадто багато

ментальних ресурсів на внутрішній переклад слів з рідної мови на іноземну, що порушує природний темп мовлення та викликає затримки.

Аналіз професійних потреб сучасних користувачів освітніх платформ показує, що стандартні підходи комп'ютерного навчання (EdTech) не вирішують зазначену проблему. Поточні ринкові рішення пропонують переважно пасивні методи взаємодії, такі як механічний вибір варіантів відповідей, заповнення пропущених літер у словах або складання речень із готових статичних блоків. Такий підхід тренує виключно пасивні навички розпізнавання тексту, але не стимулює розвиток активного продуктивного мовлення. Для подолання комунікативного страху та автоматизації розмовних навичок користувачам необхідне середовище, яке відповідає набору критичних вимог.

По-перше, виникає потреба у створенні безпечного, психологічно комфортного простору для тренувань. Віртуальний співрозмовник, на відміну від реального репетитора чи носія мови, не виносить суб'єктивних оцінок, що дозволяє користувачеві повністю позбутися сорому та страху критики. По-друге, існує гостра потреба у безперервному та моментальному зворотному зв'язку. Програма повинна не просто підтримувати розмову, а паралельно виконувати роль експертного аналітика, який виявляє лексичні збої, граматичні помилки та делікатно вказує на них користувачеві після завершення діалогу.

По-третє, аналіз користувацьких запитів підтверджує необхідність гейміфікації процесу вивчення слів. Механічне зазубрювання застаріло, оскільки воно швидко викликає ментальну втому та призводить до втрати мотивації. Студенти потребують різноманітності інтерактивних механік, де засвоєння нової лексики відбувається через ігрові сесії, що супроводжуються змагальними елементами, накопиченням балів досвіду та наочною візуалізацією особистого прогресу.

Таким чином, розробка програмного забезпечення, що поєднує функції інтелектуального розмовного партнера з інструментами автоматичного аналізу помилок та ігровими методами вивчення слів, є відповіддю на реальні незадоволені потреби сучасних користувачів. Реалізація такої системи дозволить ефективно

трансформувати пасивні лінгвістичні знання студентів в активні комунікаційні навички, суттєво прискорюючи процес руйнування мовного бар'єра.

1.2 Принципи побудови інтерактивних ШІ-компаньйонів у лінгвістичній сфері

Щоб створити ефективну та архітектурно збалансовану вебплатформу для подолання мовного бар'єра, необхідно чітко визначити принципи функціонування розмовних агентів штучного інтелекту та виокремити ключові модулі їхньої екосистеми. На відміну від класичних детермінованих чат-ботів, які працюють за жорстко прописаними скриптами, інтелектуальний лінгвістичний компаньйон повинен динамічно реагувати на непередбачувані репліки користувача, забезпечувати асинхронну обробку голосового та текстового потоків, виконувати миттєву експертизу граматичних помилок і підтримувати природний темп ведення бесіди без критичних затримок з боку сервера [2, 31].

Головним інженерним принципом проєктування такої системи є розділення загальної архітектури на три взаємопов'язані функціональні рівні, що працюють у послідовному конвеєрному режимі. Перший рівень відповідає за аудіо-лінгвістичну взаємодію з користувачем. Його завдання полягає у захопленні звукового сигналу з мікрофона, його фільтрації та перетворенні на текстовий рядок за допомогою алгоритмів розпізнавання мовлення Speech-to-Text, а на фінальній стадії діалогу цей же рівень забезпечує зворотне перетворення згенерованого тексту в аудіопотік через модулі Text-to-Speech [3]. Другий уровень фокусується на лінгвістичному та контекстному аналізі отриманого тексту. Він виступає в ролі експертного фільтра, який сканує структуру речень, виявляє лексичні, морфологічні чи синтаксичні відхилення від мовної норми та формує структурований масив аналітики. Третій рівень координує бізнес-логіку та пам'ять агента. Цей модуль на базі великих мовних моделей Large Language Models аналізує зміст бесіди, утримує фокус на заданій темі та адаптує складність використовуваної лексики под поточний рівень студента [4, 33].

Для детального розуміння того, як саме ці архітектурні рівні взаємодіють між собою під час сесії спілкування, було розроблено структурну схему, яку наведено на рисунку 1.1.

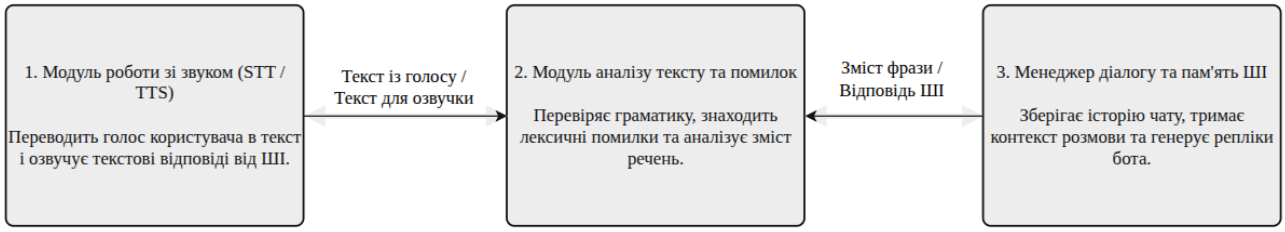

Рисунок 1.1 – Схема взаємодії рівнів обробки даних у ШІ-компаньйоні

Наступним базовим принципом є забезпечення нелінійної гнучкості комунікативного процесу. Навчальний ШІ-агент не повинен обмежувати хід розмови суворими статичними шаблонами. Якщо користувач раптово змінює вектор обговорення або ставить абстрактне питання, система за допомогою правильно налаштованого системного інструктажу (System Prompt) повинна підтримати діалог, дати релевантну відповідь, а потім плавно та делікатно повернути користувача до початкового навчального сценарію [5]. Важливим аспектом гейміфікації та ергономіки є принцип невтручання у безпосередній момент мовлення. Програма фіксує всі помилки асинхронно і непомітно для студента, акумулюючи їх у базі даних, а розгорнутий аналітичний звіт виводиться на екран лише після явного завершення поточної діалогової сесії, що дозволяє уникнути виникнення додаткового психологічного стресу та мовного ступору.

З технічного боку критично важливим є дотримання часових лімітів очікування відповідей. Тривалі затримки при генерації тексту руйнують динаміку живого спілкування, зводячи нанівець ефективність подолання комунікативного бар'єра. Побудова системи повинна базуватися на використанні оптимізованих асинхронних шлюзів і методів потокової передачі даних (Streaming API) [6], завдяки чому процеси розпізнавання, аналізу та генерації тексту відбуваються за мінімальний проміжок часу.

Таким чином, практична реалізація ШІ-компаньйона на основі вищезазначених принципів дозволяє розгорнути для студента повноцінне інтерактивне середовище, доступне в режимі двадцять чотири на сім. Віртуальний співрозмовник повністю нівелює потребу в первинному залученні реального репетитора, ефективно знімає психологічний острах критики та готує людину до природної комунікації в реальному мовному середовищі.

1.3 Математичне обґрунтування алгоритмів інтервального повторення слів

Ефективність вивчення нових слів на розробленій вебплатформі напряму залежить від того, за якими правилами система пропонує користувачеві повторювати пройдений матеріал. Якщо просто зазубрювати великі списки слів, людська пам'ять швидко їх забуде. Це природний процес, який у психології описується кривою забування [7]. Щоб вирішити цю проблему і допомогти студенту запам'ятати лексику надовго, в основу програми покладено математичний метод інтервальних повторень [8]. Його суть полягає в тому, що система підраховує оптимальний час для повторення кожного слова індивідуально для кожного користувача.

Для реалізації цієї логіки використовується адаптована система Лейтнера [9]. Математична логіка цього підходу базується на розподілі всіх слів за умовними рівнями або віртуальними скриньками. Коли користувач додає нове слово для вивчення, воно попадає на перший рівень, де його потрібно повторювати щодня. Якщо під час гри у модулі користувач дає правильну відповідь, слово переходить на наступний рівень, а час до його нової появи збільшується за певним інтервалом. Якщо ж користувач робить помилку, слово миттєво штрафується та повертається на самий початок.

Для наочного розуміння того, як математично розраховуються ці інтервали та як слова рухаються між рівнями, розроблено блок-схему, яку наведено на рисунку 1.2.

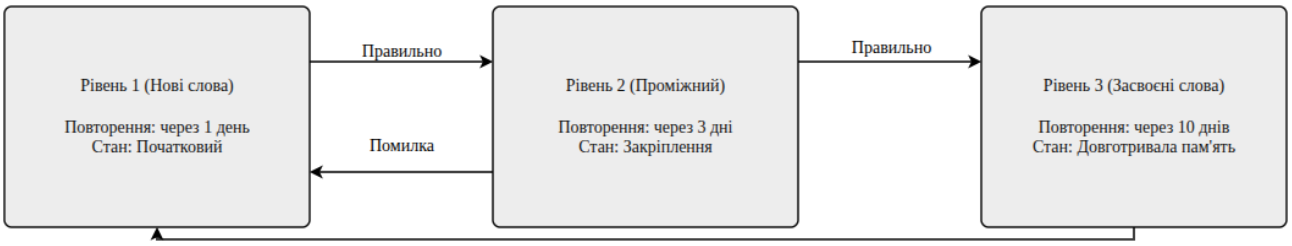

Рисунок 1.2 – Математична логіка алгоритму повторення слів за системою Лейтнера

З боку написання коду на сервері, для кожного слова у базі даних створюється спеціальний запис, де зберігається його поточний рівень та точний час останнього повторення. Програма постійно аналізує ці дані за допомогою спеціальних математичних формул, які розраховують коефіцієнт забування [10]. Серверна частина додатка автоматично підбирає зі словника саме ті лексеми, термін пам'яті яких добігає кінця, і виводить їх у ігровий інтерфейс користувача.

У межах проєкту цю модель налаштовано так, щоб вона працювала синхронно з 12 навчальними іграми. Коли студент успішно проходить ігрові квести чи мінітести, скрипт на бекенді змінює часову мітку для перевірених слів. Замість того, щоб щодня показувати користувачеві одні й ті самі прості слова, програма відкладає їх на 3 чи 10 днів, звільняючи місце для нової та більш складної лексики. Якщо ж фіксується помилка під час гри, база даних автоматично реєструє це, скидає прогрес слова і воно знову з'явиться на екрані вже наступного дня.

Використання такої математичної моделі дає можливість значно зекономити час студента. Замість хаотичного перегляду всього словника, система сама підказує, що саме починає забуватися. Це допомагає швидко перевести вивчені слова з короткочасної пам'яті у довготривалу, створюючи надійний запас слів для подальшого спілкування з ШІ-компаньйоном.

1.4 Методологія гейміфікації навчального процесу та механіки оцінювання

Ефективність самостійного вивчення іноземної мови через вебплатформу залежить не тільки від розумних алгоритмів ШІ, а й від того, наскільки цікаво побудований сам процес. Якщо інтерфейс пропонує лише сухі таблиці та монотонні тексти, користувач швидко втрачає інтерес і припиняє заняття. Щоб підтримувати постійне бажання вчитися, у проєкті використовуються принципи гейміфікації, тобто впровадження ігрових механік у звичайний навчальний процес [11]. Це дозволяє перетворити рутинну перевірку знань на захоплюючу гру.

Головна ідея гейміфікації на платформі полягає у створенні замкнутого ігрового циклу, де кожна дія користувача має чітку та швидку винагороду [12]. Для цього було розроблено ігровий контур, який складається з 12 різних мініігор для вивчення слів, системи накопичення балів досвіду (XP) та обмеженої кількості спроб для складання іспитів. Коли студент успішно проходить ігрові рівні, сервер нараховує йому бали, які відображаються у профілі на графіках статистики. Водночас для перевірки розмовних навичок у чаті з ШІ-компаньйоном діє система двоетапних іспитів, де у користувача є лише три спроби, що додає ігрового азарту та стимулює відповідальніше ставитися до тестів.

Для наочного представлення того, як побудовані ігрові механіки системи та як вони взаємодіють із контуром оцінювання, розроблено структурну схему, яку наведено на рисунку 1.3.

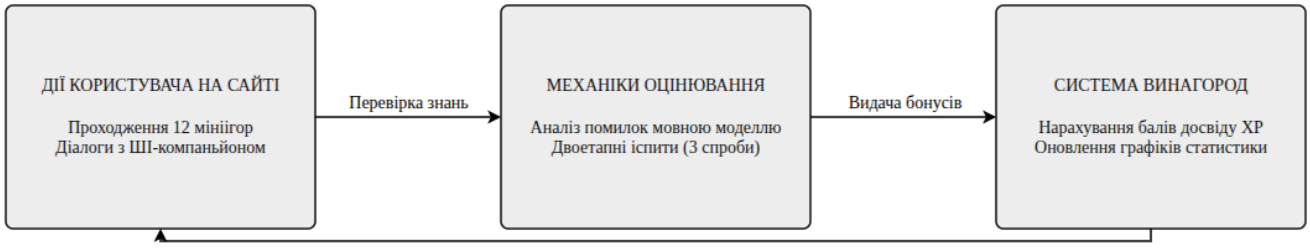

Рисунок 1.3 – Схема взаємодії ігрових механік та системи оцінювання платформи

З інженерної точки зору реалізація методології гейміфікації вимагає чіткої роботи серверної частини програми. Бекенд сайту повинен не просто фіксувати правильні відповіді, а й динамічно оновлювати стан профілю користувача. Кожна з 12 мініігор має свій рівень складності, тому кількість балів XP, що нараховуються, розраховується автоматично залежно від типу гри та кількості допущених помилок. Отримані результати миттєво записуються у базу даних, що дозволяє виводить актуальні графіки успішності на фронтенді без затримок.

Механіка оцінювання розмовних навичок у чаті з ШІ-компаньйоном побудована так, щоб не викликати у користувача психологічного дискомфорту. Програма аналізує текстові та голосові повідомлення за допомогою гнучких запитів до мовної моделі. Якщо студент робить помилку, система м'яко вказує на неї в окремому блоці аналітики, не перериваючи ігрову сесію. Коли ж справа доходить до підсумкового іспиту, включається суворіший контроль знань: обмеження у три спроби змушує користувача краще готуватися у тренувальних іграх перед тим, як переходити до фінального тестування.

1.5 Формулювання вимог до функціоналу та архітектури платформи

На основі проведеного аналізу потреб користувачів та вивчення ігрових і ШІ-механік, необхідно чітко сформулювати технічні вимоги до розроблюваного вебдодатка. Це дозволить побудувати правильну структуру сайту та забезпечити стабільну взаємодію між користувачем, серверною логікою та штучним інтелектом. Усі вимоги до системи традиційно поділяються на функціональні, які описують можливості програми, та нефункціональні, що визначають її технічні параметри [14].

Функціональні вимоги до платформи адаптивного навчання мають повністю покривати роботу трьох основних типів користувачів, а саме студентів, вчителів та адміністраторів сайту. Модуль авторизації повинен забезпечувати реєстрацію користувачів, безпечний вхід за паролем та розмежування прав доступу. Для ШІ-компаньйона головною вимогою є ведення текстового та голосового чату із

збереженням контексту розмови та аналізом помилок. Ігровий блок має підтримувати стабільну роботу 12 навчальних мініігор для вивчення та закріплення нових слів.

У модулі іспитів необхідно реалізувати проведення двоетапних зрізів знань та автоматичний контроль обмеження у три спроби. Кабінет вчителя повинен надавати інструменти для перегляду логів діалогів з ШІ та детального аналізу помилок учня за допомогою інтегрованого RAG-агента. У профілі студента має відбуватися нарахування балів досвіду XP та виведення інтерактивних графіків особистої статистики. Наостанок, панель адміністратора повинна містити функції керування обліковими записами, обробки звернень до техпідтримки та загального налаштування параметрів сайту.

Окрім функціональних можливостей, до системи висуваються суворі нефункціональні вимоги. З боку продуктивності та швидкодії затримка при відповіді ШІ-компаньйона не повинна перевищувати кількох секунд, щоб розмова залишалася природною. З боку надійності та стабільності база даних має коректно обробляти паралельні запити, зберігаючи логи чатів та ігрові бали користувачів без втрати інформації. З боку інтерфейсу система повинна бути адаптивною, щоб однаково зручно відображатися як на комп'ютерах, так і на мобільних телефонах користувачів.

Архітектура вебплатформи повинна будуватися за класичним клієнт-серверним принципом [15] із чітким відокремленням рівнів [16]. Клієнтська частина відповідає за відображення інтерфейсу чату, запуск 12 ігор та показ графіків статистики. Серверна частина реалізує бізнес-логіку, прораховує алгоритм інтервальних повторень Лейтнера, нараховує бали XP та виступає безпечним шлюзом для передачі даних до хмарних моделей штучного інтелекту.

Таким чином, чітке формулювання вимог до функціоналу та архітектури створює детальний орієнтир для подальшої практичної реалізації проєкту. Це гарантує, що створений вебдодаток буде не просто набором ігор, а цілісною, високошвидкісною та безпечною інтелектуальною екосистемою для швидкого й ефективного подолання мовного бар'єра.

2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

Переходячи до практичної частини випускної кваліфікаційної роботи, зосереджено увагу на детальному інженерному проєктуванні та безпосередній програмній реалізації всіх модулів вебплатформи FLUEN AI. У цьому розділі наведено опис вибору архітектурних рішень, обґрунтовано сформований технологічний стек, а також продемонстровано алгоритми аналізу та тестування розробленої системи. Це дозволяє наочно показати практичну цінність та повну працездатність створеного інтелектуального продукту.

2.1 Комплексне проектування архітектури, визначення функціоналу та стеку технологій платформи

Практичний етап реалізації освітньої платформи FLUEN AI було розпочато із системного проєктування та точного визначення її технологічного базису. Головна інженерна мета полягала у створенні швидкого, масштабованого вебдодатка, здатного в реальному часі обробляти голосові й текстові дані, оперувати гейміфікованими алгоритмами навчання та координувати асинхронну взаємодію з двома різними ШІ-агентами. Для цього було спроєктовано модульну архітектуру із чітким розділенням на клієнтський фронтенд, серверний бекенд, аналітичний шар та контур штучного інтелекту.

Для клієнтської частини додатка обрано архітектуру односторінкового застосунку (Single Page Application, SPA) [17] на базі сучасного JavaScript-фреймворку, завдяки чому браузер завантажує інтерфейс лише один раз, а всі внутрішні переходи між кабінетом, словником та іграми відбуваються миттєво. Візуальний стиль сторінок виконано у темній колірній гамі, щоб суттєво знизити зорове навантаження на користувача під час тривалого навчання.

Серверну логіку побудовано на мові програмування Python із застосуванням асинхронного фреймворку FastAPI [18] та серверного рушія Uvicorn. Цей вибір зумовлений високою швидкодією інструменту та його нативною підтримкою

асинхронності, яка є критично важливою для системи при паралельній обробці великих аудіофайлів та тривалому очікуванні відповідей від нейромереж. Додатково задіяно можливості бібліотеки Pydantic для автоматичної валідації вхідних JSON-пакетів, що дозволяє відсікати некоректні запити користувачів ще на етапі їхнього надходження до бізнес-логіки.

Основним сховищем даних у проєкті визначено реляційну базу PostgreSQL [19], яка гарантує повну транзакційну цілісність під час збереження ігрових балів та профілів. Ключовою технологічною перевагою стало впровадження розширення pgvector [20], за допомогою якого базу даних перетворено на векторне сховище для обробки числових ембеддінгів та виконання семантичного косинусного пошуку інформації. Детальний аналіз розроблених таблиць та зв'язків між ними наведено у наступному підрозділі роботи.

Особливу увагу приділено підсистемі збору аналітики, де для фіксації мікродій користувачів без перевантаження бази даних інтегровано брокер черг повідомлень Apache Kafka [21]. Фронтенд додатка фіксує кожен крок студента і надсилає повідомлення у топік черги, звідки фоновий Python-скрипт безперервно забирає дані та пакетно записує їх у проміжну буферну таблицю. Для остаточного розрахунку часу активності реалізовано другий незалежний скрипт, який запускається за розкладом кожні тридцять хвилин, агрегує сесії, вираховує чисті хвилини, оновлює статистику профілю та очищує сирі логи для економії дискового простору.

Інтелектуальний контур платформи розділено на двох самостійних агентів:

- першого агента спроєктовано як розмовного компаньйона, робота якого базується на моделях Whisper та GPT через офіційний API для транскрипції, генерації реплік та прихованого аналізу мовлення;
- другого агента реалізовано для підтримки кабінету вчителя за архітектурою Retrieval-Augmented Generation (RAG) із підключенням векторної бази знань та інструментів динамічного пошуку в мережі Інтернет [32], що дозволяє йому автоматично знаходити актуальні граматичні правила в мережі та формувати індивідуальні тести.

Уся розроблена програмна система повністю контейнеризується за допомогою технології Docker [22], завдяки чому створено ізольовані контейнери для бекенду, бази даних, брокера Kafka та фонових служб. Фінальне хмарне розгортання платформи виконано на сервісі Render [23], обравши його через оптимальні умови для безкоштовного тестування, моніторингу робочих процесів та автоматичної інтеграції коду безпосередньо з репозиторію.

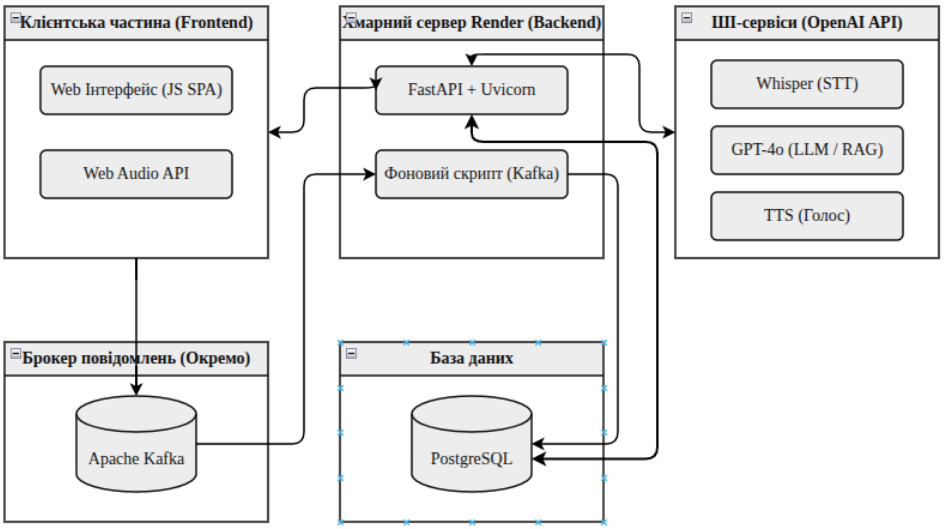

Рисунок 2.1 – Схема загальної архітектури та технологічних блоків екосистеми FLUEN AI

Функціональний контур системи розбито на кілька незалежних робочих просторів, доступ до яких суворо контролюється модулем автентифікації. Для формалізації взаємодії користувачів із функціональними блоками системи розроблено діаграму варіантів використання, яку зображено на рисунку 2.2. Вона відображає ролі трьох основних акторів платформи, якими у проєкті виступають Студент, Вчитель у вигляді інтелектуального агента та Адміністратор.

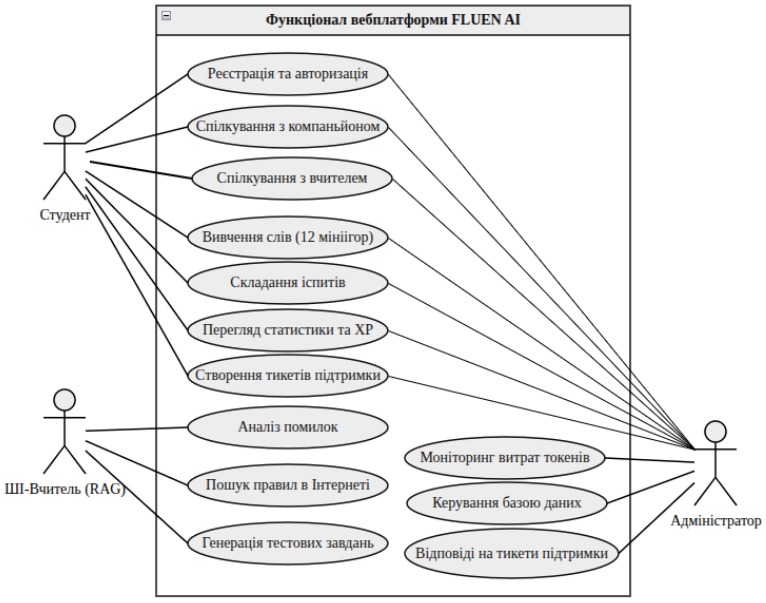

Рисунок 2.2 – Діаграма варіантів використання системи FLUEN AI

У системі Студент є ключовим актором, який має доступ до реєстрації, ведення текстового й голосового діалогу з компаньйоном, вивчення слів у дванадцяти мінііграх, проходження двоетапних іспитів та перегляду графіків аналітики. Вчитель виступає як інтелектуальний агент, який автоматично аналізує повідомлення. Адміністратор має повний доступ до панелі управління. Створене чітке архітектурне та функціональне розділення сформувало надійний фундамент для подальшої практичної розробки та проєктування реляційної схеми даних.

Для деталізації внутрішньої структури програмного забезпечення, опису об’єктів системи та їхньої поведінки на рівні вихідного коду було розроблено UML-діаграму класів (рис. 2.3). Вона відображає реалізацію патерну Layered Architecture, що дозволяє відокремити мережевий API-інтерфейс (Controller Layer) від безпосередньої бізнес-логіки застосунку (Service Layer).

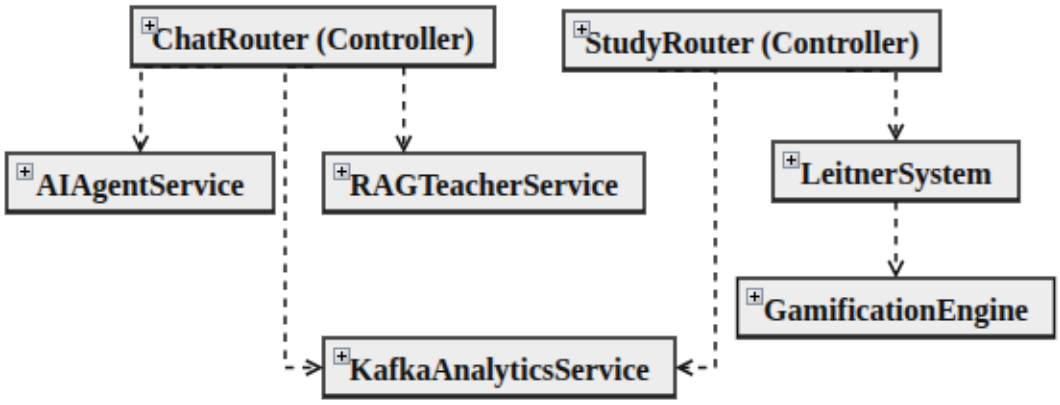

Рисунок 2.3 – UML-діаграма класів програмної архітектури бекенду

На розробленій діаграмі представлені ключові класи програмної архітектури бекенду, розподілені за функціональними шарами:

1.Клас ChatRouter (Controller)

Виконує роль точки входу для діалогових запитів користувача.

- Функції: Маршрутизація запитів, аутентифікація повідомлень, керування історією чатів.
- Параметри та методи:
  - post_audio_message(audio: File, user: AuthenticatedUser) — приймає голосовий потік, передає його в AIService та повертає текстову відповідь.
  - get_chat_history(chat_id: int) — повертає лог повідомлень для конкретної сесії.
  - delete_chat(chat_id: int) — видаляє сесію з бази даних.

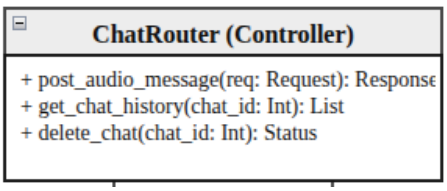

Рисунок 2.4 — Структура класу ChatRouter

2. Клас AIAgentService

Основний клас для взаємодії зі штучним інтелектом.

- Функції: Обробка мови (STT/TTS), генерація відповідей, аналіз граматики.
- Параметри та методи:
  - analyze_and_respond(user_text: str, chat_history: list) — головний метод бізнес-логіки; формує промпт, викликає OpenAI API, обробляє tool-calls та повертає результат у форматі JSON.
  - speech_to_text(audio: bytes) — викликає модель Whisper для транскрипції.
  - text_to_speech(text: str) — викликає модель TTS для генерації голосового файлу.
  - add_more_information_for_words(words: str) — метод збагачення словника (генерує контекст та транскрипцію для нових слів).

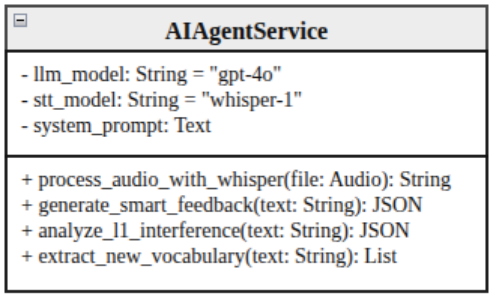

Рисунок 2.5 — Структура класу AIAgentService

3. Клас RAGTeacherService (Service Layer)

Клас для роботи з контекстними даними.

- Функції: Пошук граматичних правил та підготовка навчального контексту.
- Параметри та методи:
  - cosine_similarity_search(query: str) — виконує пошук у векторному просторі бази даних (pgvector), повертаючи найрелевантніші чанки тексту.

- chunk_and_embed_document(doc: File) — метод, що розбиває PDF-документи на логічні частини та створює для них вектори.

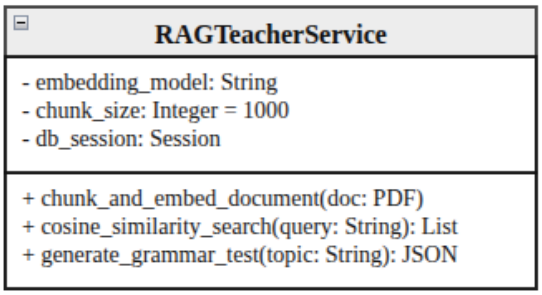

Рисунок 2.6 — Структура класу RAGTeacherService

4. Клас KafkaAnalyticsService (Infrastructure)

– Функції: Асинхронне логування подій для аналітики.
– Параметри та методи:
  - produce_event(event_type: str, data: dict) — відправляє подію (клік, початок гри) у чергу повідомлень.
  - aggregate_hourly_sessions() — метод для фонового скрипта, який підсумовує загальний час навчання за годину.

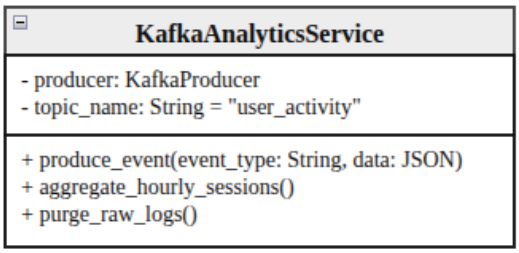

Рисунок 2.7 — Структура класу KafkaAnalyticsService

5. Клас StudyRouter (Controller Layer)

Точка входу для навчального контуру.

– Функції: Управління процесом навчання та оцінювання.
– Параметри та методи:
  - get_due_words() — повертає слова, час повторення яких настав (звертається до LeitnerSystem).

- submit_exam_result(payload: JSON) — приймає результати іспиту та ініціює оновлення XP та рівня знань слова.

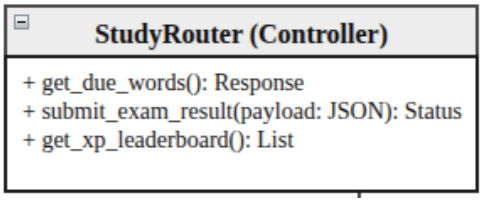

Рисунок 2.8 — Структура класу StudyRouter

6. Клас LeitnerSystem (Service Layer)

Алгоритмічний модуль адаптивного навчання.

– Функції: Керування інтервалами повторення слів за методом Лейтнера.
– Параметри та методи:
  - calculate_next_review(word_id: int, success: bool) — приймає результат відповіді користувача та розраховує нову дату появи слова в іспиті.
  - verify_pronunciation(audio: bytes) — порівнює вимову користувача з еталоном (транскрипцією).
  - downgrade_level_on_fail(word_id: int) — скидає статус засвоєння слова, якщо користувач припустився помилки.

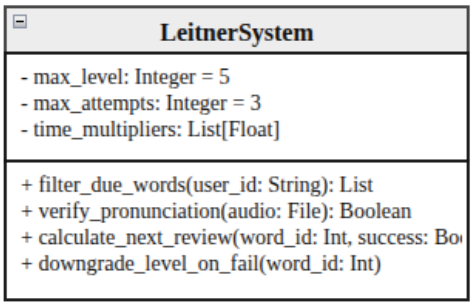

Рисунок 2.9 — Структура класу LeitnerSystem

7. Клас GamificationEngine (Service Layer)

Клас для підтримки ігрових механік.

– Функції: Розрахунок прогресу та нарахування винагород.

– Параметри та методи:
  - award_xp_for_game(user_id: str, score: int) — нараховує бали досвіду з урахуванням складності гри.
  - update_daily_streak(user_id: str) — оновлює лічильник "серії днів" (streak), перевіряючи дату останньої активності.

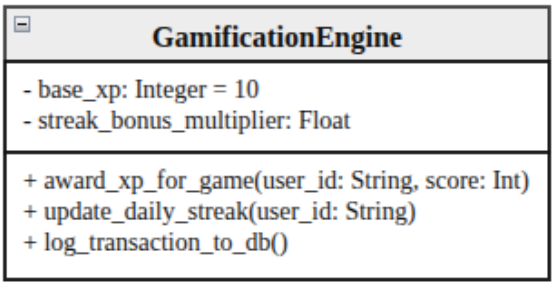

Рисунок 2.10 — Структура класу GamificationEngine

Для моделювання динамічної поведінки системи та візуалізації взаємодії об’єктів у часі під час виконання ключових функціональних сценаріїв було розроблено UML-діаграми послідовностей (Sequence Diagrams).

Сценарій 1. Обробка голосового повідомлення користувача та генерація Smart Feedback Цей сценарій відображає процес асинхронного аналізу мовлення студента розмовним ШІ-компаньйоном (рис. 2.11):

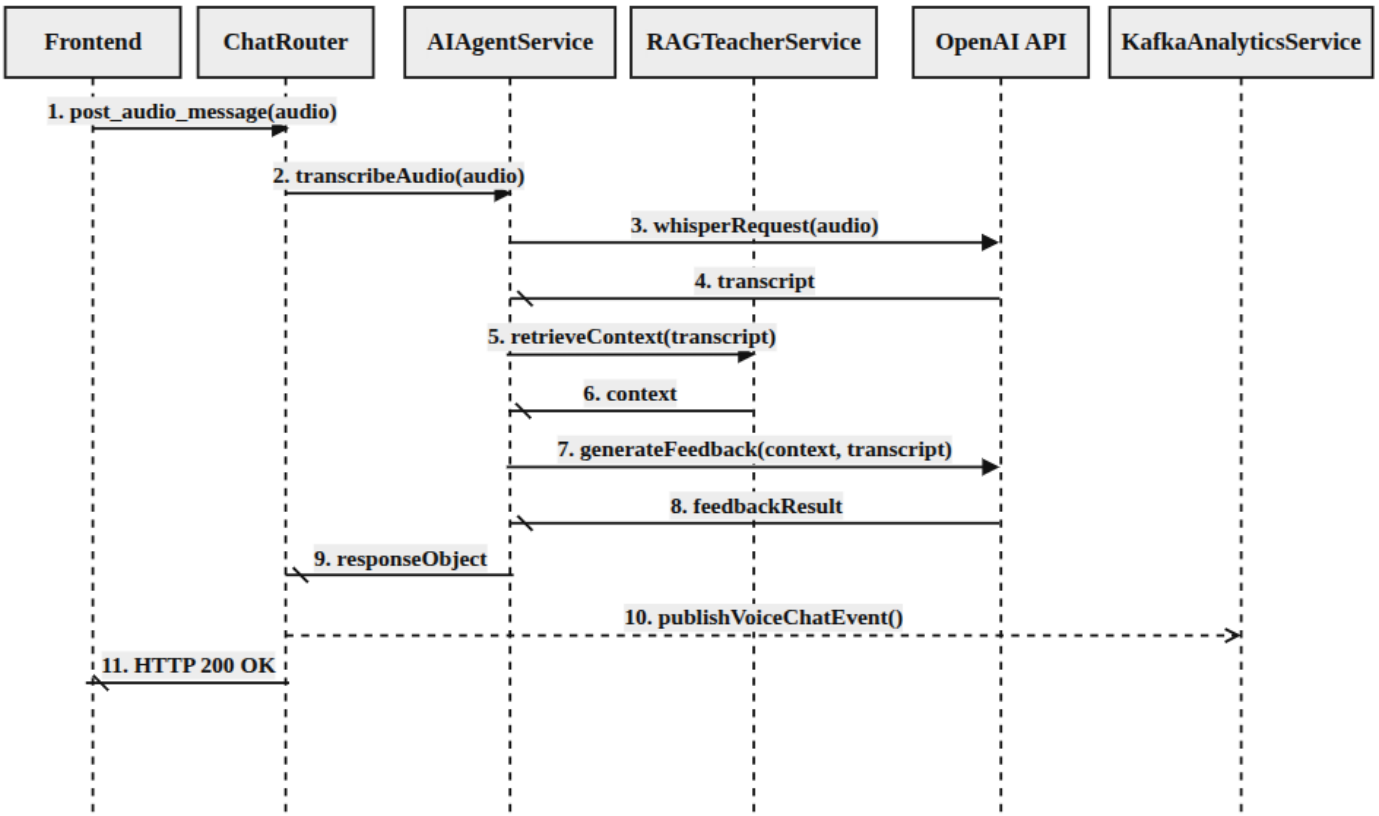

Рисунок 2.11 – Діаграма послідовностей для сценарію обробки голосового повідомлення

Сценарій 2. Проходження іспиту та адаптивний перерахунок інтервалів повторення Цей сценарій описує логіку роботи навчального контуру за методом Лейтнера при фіксації прогресу (рис. 2.12):

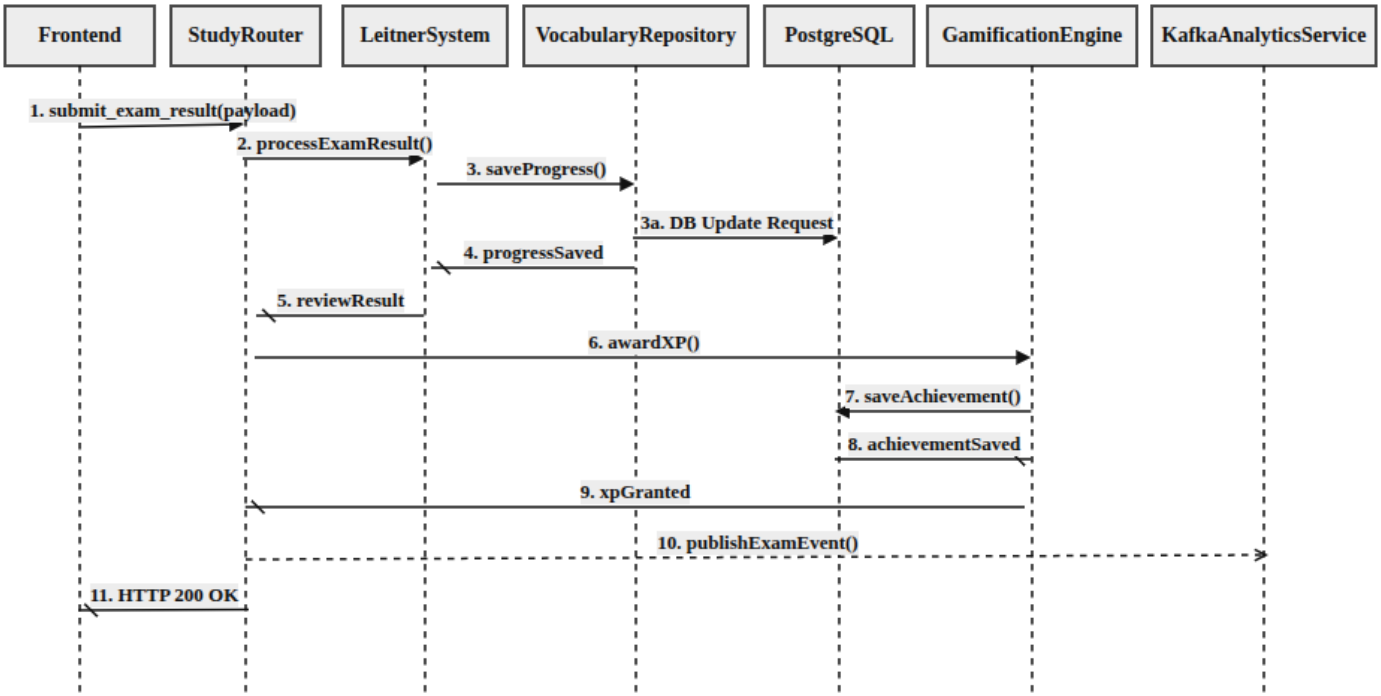

Рисунок 2.12 – Діаграма послідовностей для сценарію проходження іспиту за методом Лейтнера»

2.2 Проектування схеми бази даних та обґрунтування структури таблиць

Для забезпечення цілісності та швидкого доступу до інформації спроєктовано логічну структуру реляційної бази даних платформи. Оскільки серверний бекенд розроблено на мові Python із використанням фреймворку FastAPI, для ефективної взаємодії зі сховищем задіяно технологію об'єктно-реляційного відображення SQLAlchemy [24]. Це інженерне рішення дозволило описати всі таблиці у вигляді програмних класів, де кожна колонка відповідає певному атрибуту класу. Як основну систему керування базами даних обрано PostgreSQL [19], спираючись на її високу продуктивність під навантаженням та нативну підтримку розширень для роботи з векторами.

Центральною ланкою архітектури даних визначено таблицю користувачів, представлену в коді класом User. У ній передбачено збереження первинного ключа, хешу пароля, електронної пошти, а також індивідуальних лінгвістичних налаштувань, таких як рідна мова та мова вивчення. Окрім цього, у структурі цієї

таблиці виділено поля для глобальних ігрових метрик студента, включаючи бали досвіду та лічильник днів безперервної активності. Щоб захистити систему від появи ізольованих або некоректних записів, таблицю користувачів пов'язано з іншими модулями бази даних відношеннями один до багатьох із налаштуванням каскадного видалення. Завдяки цьому при видаленні профілю база даних самостійно очистить усі зв'язані словники, логи та історію повідомлень користувача.

Наступним компонентом системи реалізовано ігровий контур та механізм інтервального повторення в межах таблиці UserVocabulary. Кожен рядок тут прив'язано до ідентифікатора студента через зовнішній ключ. Окрім текстових полів для збереження самого слова, його транскрипції та перекладу, задіяно тип даних JSON для динамічного збереження масивів синонімів, антонімів та прикладів використання у контексті, що позбавило від створення громіздких дрібних таблиць та прискорило завантаження даних під час мініігор. Для керування алгоритмом Лейтнера передбачено цілочисельне поле рівня та часову мітку наступної перевірки, що дозволяє серверу одним SQL-запитом відфільтрувати лексику для повторення. Усі нарахування балів досвіду налаштовано на фіксацію в окремій таблиці XpLog для прозорого аудиту досягнень.

Комунікаційний контур для взаємодії зі штучним інтелектом розділено на дві взаємопов'язані таблиці. У таблиці Chat фіксуються окремі діалогові сесії та поточний режим роботи ШІ-агента, а в таблиці Message зберігається безпосередній зміст реплік, шляхи до аудіофайлів на сервері та результати аналізу граматичних помилок від нейромережі. Аналітичний блок системи, який працює у зв'язці з Apache Kafka, побудовано на основі таблиці UserActivityLog для швидкого логування мікродій клієнта та таблиці DailyActivity, куди фонові скрипти записують уже агрегований чистий навчальний час користувача за кожну добу.

Для забезпечення роботи кабінету ШІ-вчителя спроєктовано таблицю KnowledgeBase. Її головна особливість полягає в інтеграції спеціального типу даних vector за допомогою розширення pgvector [20]. Це дало змогу зберігати згенеровані числові ембеддінги лінгвістичних правил розмірністю у 1536 ознак та

виконувати швидкий семантичний косинусний пошук безпосередньо на рівні системи керування базами даних, без надлишкового навантаження на оперативну пам'ять сервера. Інженерну частину сховища доповнено таблицею SupportTicket для фіксації текстовых звернень користувачів, їхніх статусів та відповідей з панелі адміністратора.

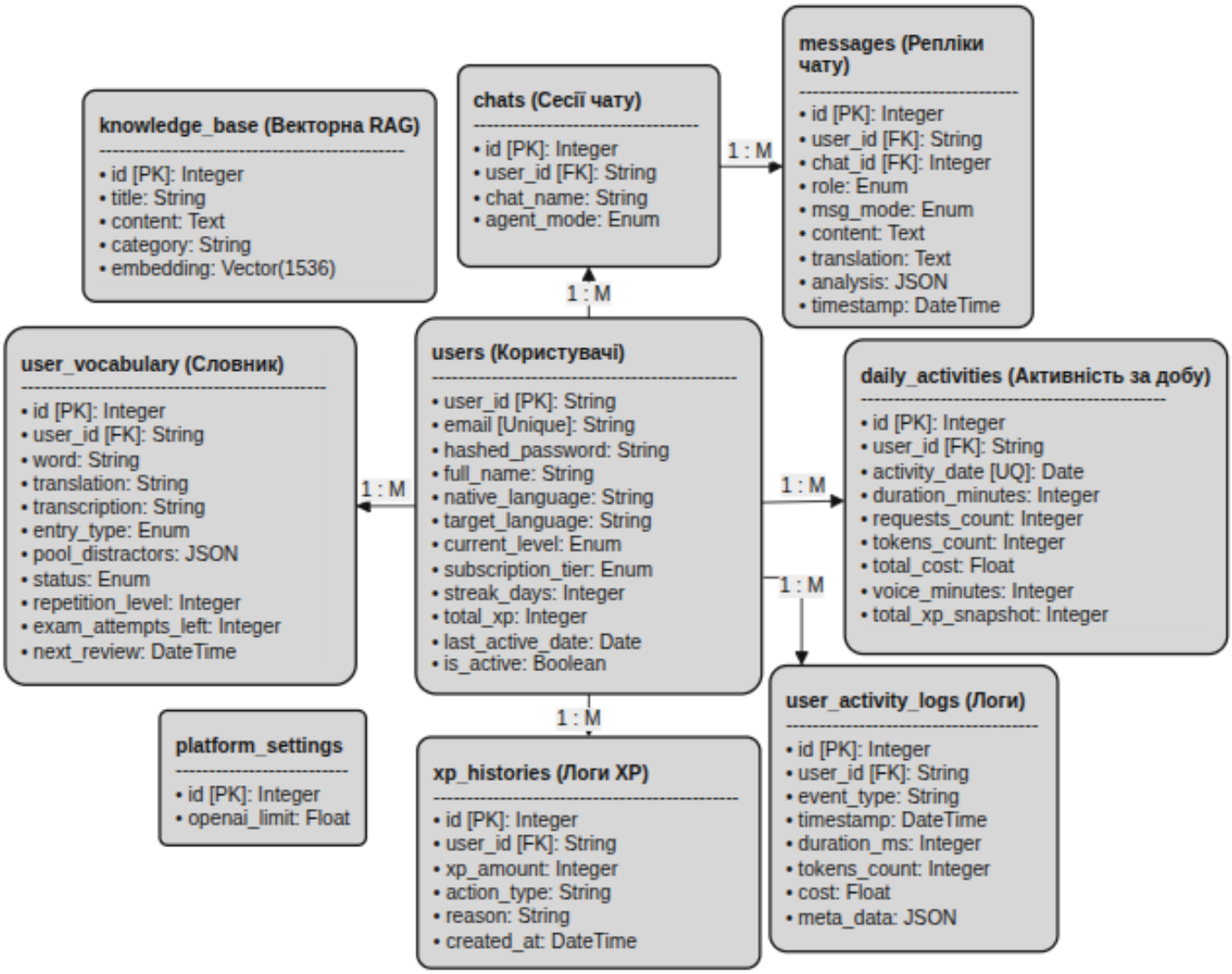

Рисунок 2.13 – Логічна схема бази даних та структура таблиць платформи

У результаті проведеної інженерної роботи розроблена схема повністю відповідає вимогам третьої нормальної форми [25], що виключає дублювання даних та аномалії оновлення. Використання оптимізованих типів даних, індексів для векторного пошуку та чітких каскадних зв'язків між первинними і зовнішніми ключами гарантує високу швидкість виконання запитів. Така структура дозволить

легко масштабувати додаток при збільшенні навантаження на систему та забезпечує стабільну роботу алгоритмів гейміфікації.

2.3 Реалізація модулів реєстрації, автентифікації та профілю користувача з аналітикою активності й XP

Наступним важливим етапом практичного створення інтелектуальної навчальної платформи FLUEN AI визначено розробку підсистеми автентифікації користувачів та модуля особистого кабінету студента. Безпечний і персоналізований доступ до функцій сайту організовано через класичні екранні форми реєстрації та входу, зовнішній вигляд яких наведено на рисунках 2.4 та 2.5 відповідно. При проєктуванні логіки авторизації особливу увагу приділено захисту конфіденційності та валідації користувацького введення на серверному рівні. Для безпечного збереження даних впроваджено відмову від запису паролів у відкритому вигляді та реалізовано їх криптографічне хешування за алгоритмом bcrypt [26], а сам процес авторизації побудовано на базі стандарту JSON Web Token [27], що дозволяє серверу надійно ідентифікувати клієнта при кожному наступному запиті.

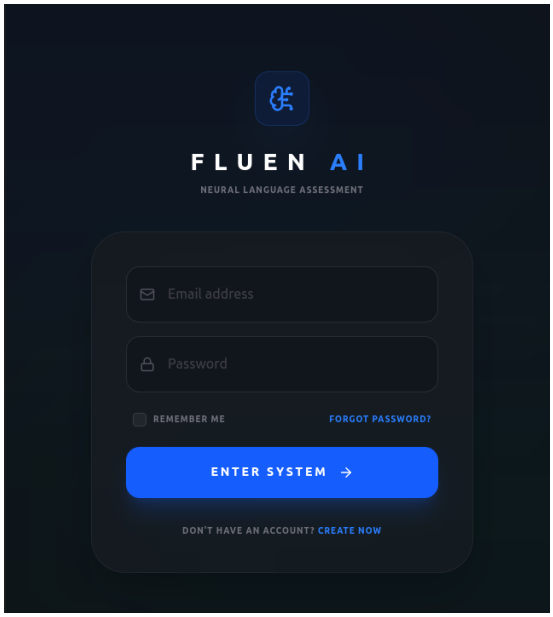

Рисунок 2.14 – Екранна форма входу до системи FLUEN AI

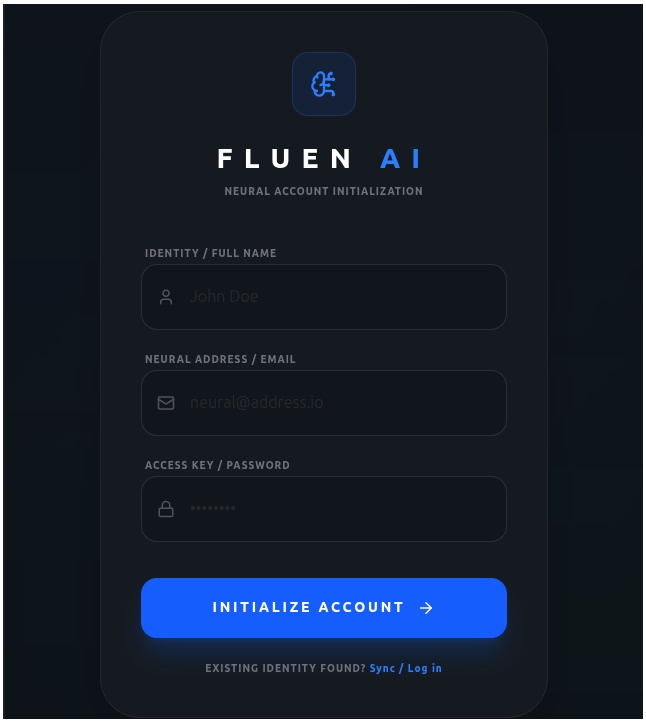

Рисунок 2.15 – Екранна форма ініціалізації нового облікового запису

Для контролю введених даних запрограмовано алгоритми перевірки, які унеможливлюють повторне використання вже зареєстрованої електронної пошти, миттєво відправляючи відповідне інтерфейсне сповіщення про конфлікт даних. Окрім цього, на рівні серверних обробників FastAPI реалізовано автоматичний контроль стійкості паролів. Програмний код аналізує довжину та складність комбінації символів, повністю блокуючи спроби встановлення занадто спрощених паролів та виводячи на екран попередження з вимогою додати цифри або спеціальні знаки.

Під час проєктування візуального оформлення платформи виконано значний обсяг роботи для вибору найкращого колірного та структурного рішення інтерфейсу. Весь дизайн реалізовано у сучасній темній темі, яка мінімізує навантаження на зір користувача при тривалій роботі та робить взаємодію комфортною. Після успішного проходження авторизації користувач потрапляє у свій персональний кабінет (рисунок 2.16), де передбачено повну підтримку мультимовності, що дозволяє студенту в будь-який момент змінити мову

відображення всього додатка між українською та англійською за допомогою зручного перемикача у верхній частині екрана.

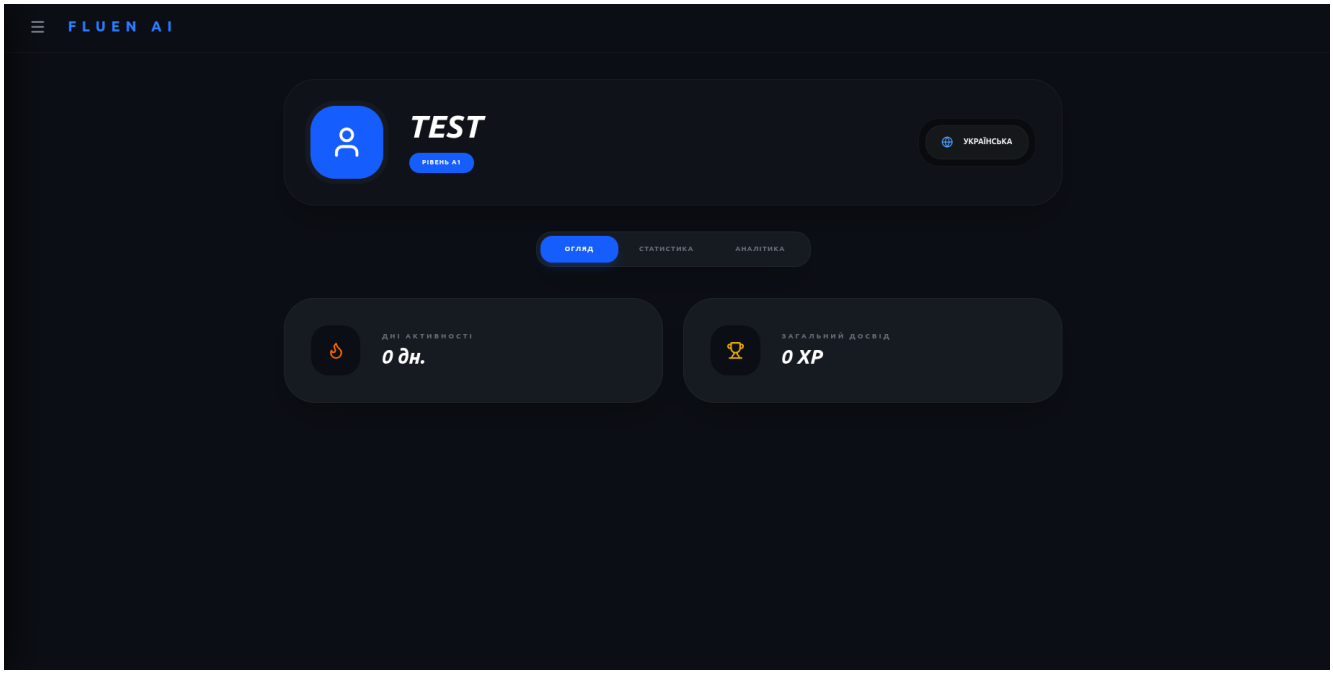

Рисунок 2.16 – Графічний інтерфейс профілю користувача платформи FLUEN AI з українською локалізацією

В особистому кабінеті студента створено блоки візуалізації індивідуального прогресу, які структурно розділено на три основні вкладки: огляд, статистика та аналітика. На першій вкладці огляду виведено загальні картки з показниками поточної серії днів відвідування та балансу балів. На вкладці статистики, яку зображено на рисунку 2.7, відображається детальний розподіл слів у персональному словнику. Інтерфейс демонструє загальну кількість лексем, а також динамічні індикатори прогресу для трьох категорій лінгвістичного стану, що відповідають словам у процесі активного вивчення, словам у режимі інтервального повторення перед іспитом та повністю засвоєним словам, які успішно перейшли у довготривалу пам'ять.

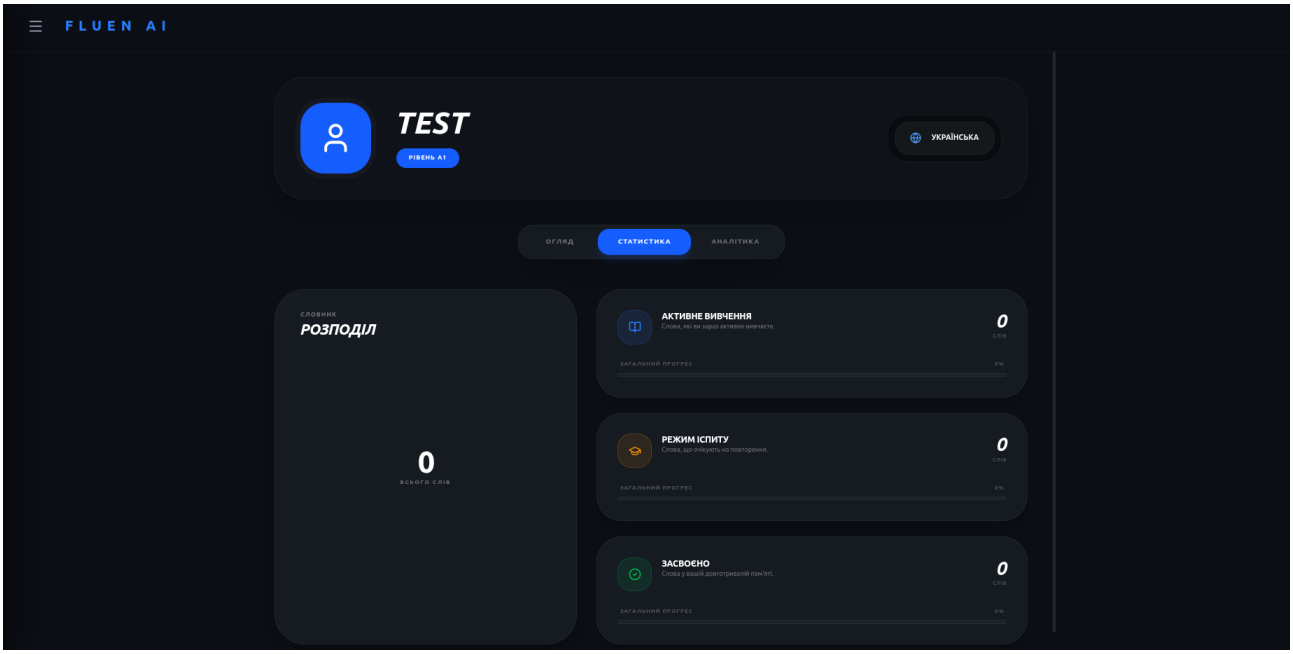

Рисунок 2.17 – Екранна форма розподілу та прогресу вивчення слів у словнику

Особливе місце в архітектурі платформи займає вкладка аналітики (рисунок 2.8), для якої розроблено алгоритми відстеження зростання балів досвіду XP та історії активності користувача. Бали досвіду нараховуються автоматично на сервері за кожну корисну дію користувача, включаючи спілкування з розмовним ШІ-компаньйоном та успішне завершення навчальних мініігор. На вбудованих інтерактивних діаграмах користувач може самостійно обирати часовий період для відображення даних, перемикаючись між переглядом статистики за тиждень, місяць, рік або за весь час ведення профілю.

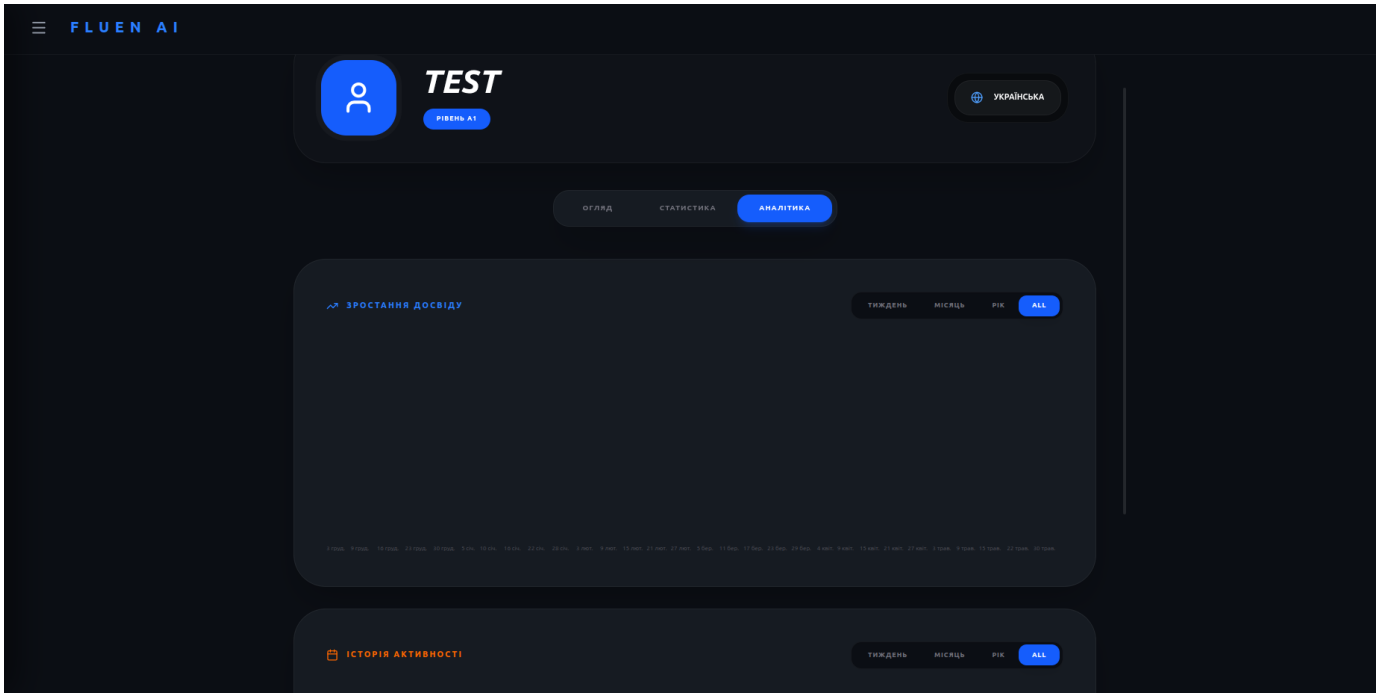

Рисунок 2.18 – Екранна форма загальної аналітики досвіду та періодів активності

З технічного боку збір аналітики часу та дій організовано через асинхронний буферний шар на базі брокера черг Apache Kafka, щоб повністю уникнути перевантаження пулу з'єднань бази даних PostgreSQL під час інтенсивної роботи. Клієнтська частина платформи асинхронно надсилає JSON-пакети подій в чергу, звідки фоновий Python-скрипт безперервно зчитує сирі записи та пакетними транзакціями переносить їх у проміжну таблицю бази даних. Повний лістинг цієї конфігурації винесено у додаток А до роботи.

Для мінімізації обсягу збережених даних реалізовано дворівневу модель агрегації метрик за допомогою другого фонового скрипта-планувальника, який запускається кожні тридять хвилин. Алгоритм виконує лінійну кластеризацію сесій, вважаючи її завершеною, якщо дельта бездіяльності користувача перевищує п'ять хвилин. Після розрахунку чистих хвилин онлайн-роботи та обсягу використаних токенів штучного інтелекту, скрипт оновлює фінальну таблицю щоденної активності та автоматично очищує сирі логи з буфера за допомогою каскадної очистки на рівні бази даних, що суттєво економить дисковий простір.

2.4 Інтеграція інтелектуального ШІ-компаньйона для мовленнєвої практики та контекстного аналізу помилок

Головним інструментом для подолання мовного бар'єра на створеній платформі визначено модуль інтелектуального ШІ-компаньйона. При проєктуванні цього контуру ставилося завдання забезпечити максимально природну та безперервну мовленнєву практику для користувача. Для цього розроблено комплексну архітектуру взаємодії між клієнтським додатком на JavaScript, сервером FastAPI та хмарними сервісами штучного інтелекту. Як основну мовну модель обрано GPT-4o через її здатність глибоко розуміти лінгвістичні нюанси та швидко генерувати відповіді. Додатково реалізовано повноцінну панель управління діалогами, де користувач має можливість створювати нові бесіди, перейменовувати їх для зручної навігації або видаляти неактуальні чати, що робить структурування інформації на платформі максимально чітким.

Інтерфейс цього модуля розроблявся з урахуванням гнучкості у виборі формату спілкування. Оскільки на початкових етапах навчання студенту може бути важко одразу розмовляти голосом, передбачено класичний текстовий режим чату. У цьому режимі користувач набирає свої повідомлення за допомогою клавіатури, а ШІ-агент швидко генерує відповідь, стабільно підтримуючи контекст поточної розмови.

Значну інженерну увагу приділено реалізації голосового режиму спілкування, для якого написано програмний код інтеграції технологій розпізнавання та синтезу мовлення. Коли користувач активує мікрофон, клієнтська частина за допомогою Web Audio API записує голос, стискає його та надсилає на сервер. Далі бекенд асинхронно передає аудіофайл до нейромережі Whisper, яка перетворює його на текстовий рядок. Унікальною особливістю створеного ШІ-компаньйона є можливість вибору режиму озвучення, завдяки чому агент може генерувати повноцінні голосові відповіді через модулі Text-to-Speech. Для наочної візуалізації цього процесу в інтерфейс додатка інтегровано анімований плеєр.

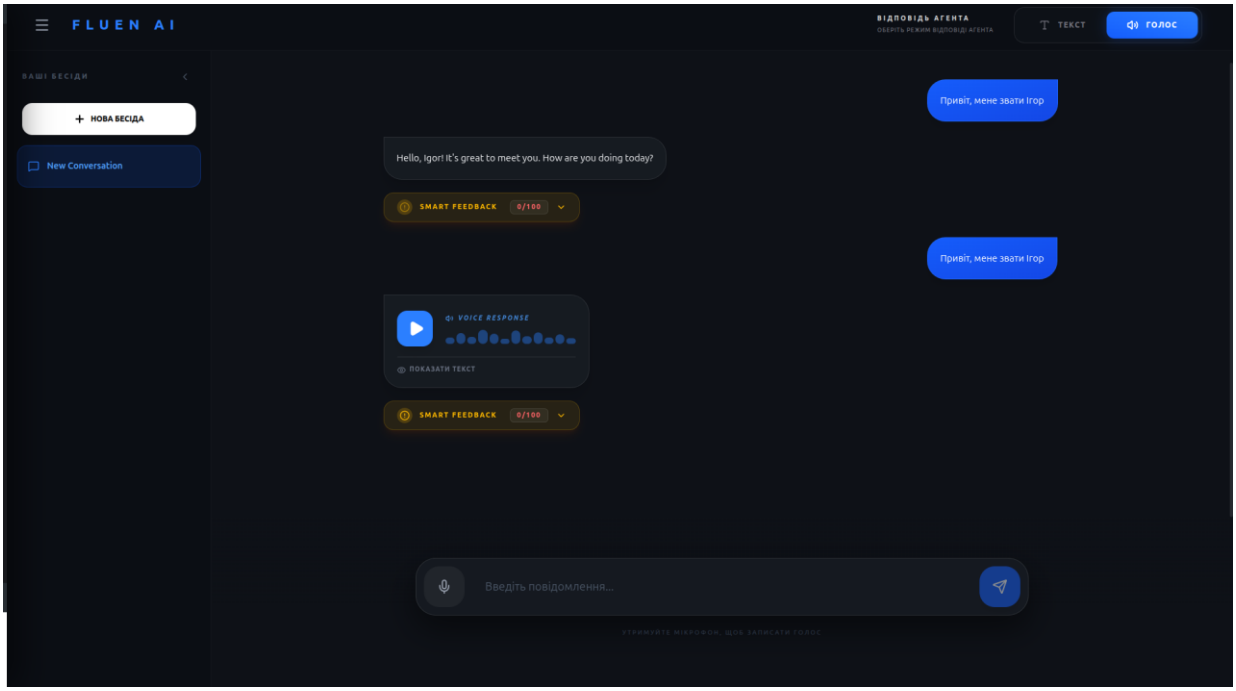

Рисунок 2.19 – Інтерфейс голосового чату з прихованою транскрипцією

Під час тестування функціоналу стало зрозуміло, що студентам часто буває важко сприймати швидку англійську вимову агента на слух. Щоб вирішити цю проблему, було розроблено спеціальний блок транскрипції. Користувач може розгорнути його, і аудіовідповідь буде продубльована англійським текстом. Якщо ж зміст повідомлення все одно залишається незрозумілим, система дозволяє в один клік перекласти цю транскрипцію на українську мову, що робить процес навчання комфортним і позбавленим стресу.

Окремим етапом роботи стало налаштування правильної поведінки віртуального співрозмовника за допомогою методів інженерії підказок (Prompt Engineering). Його внутрішній алгоритм запрограмовано так, щоб він завжди стимулював користувача говорити більше, а моделі суворо заборонено ставити закриті запитання, на які можна відповісти лише «так» чи «ні». Окрім цього, агента навчено розуміти змішаний мовний контекст: якщо студент забув певне слово, він може вільно сказати частину речення англійською, а іншу частину - українською мовою. ШІ-компаньйон автоматично розпізнає такий перехід, безперешкодно зрозуміє загальний зміст і у своєму аналізі м'яко виправить репліку, надавши правильний англійський еквівалент.

Окрім підтримання бесіди, створений ШІ-компаньйон виконує функцію невидимого експерта, який проводить глибокий контекстний аналіз кожної репліки у фоновому режимі. За допомогою спеціальних налаштувань API модель генерує відповідь у строго структурованому форматі JSON. Розроблена логіка бекенду отримує цей пакет, витягує текст для відправки в чат, а згенеровану аналітику зберігає у базу даних PostgreSQL. Після розмови або безпосередньо під час неї студент може відкрити розширену панель розумного зворотного зв'язку (Smart Feedback), яку структурно розділено на чотири аналітичні блоки. Перший блок відповідає за загальну оцінку вільності мовлення, де алгоритм вираховує бал від нуля до ста, а також демонструє ідеально сформульований варіант репліки та розбирає структуру побудови речення, показуючи, як цю саму думку висловив би носій мови у природному комунікативному середовищі.

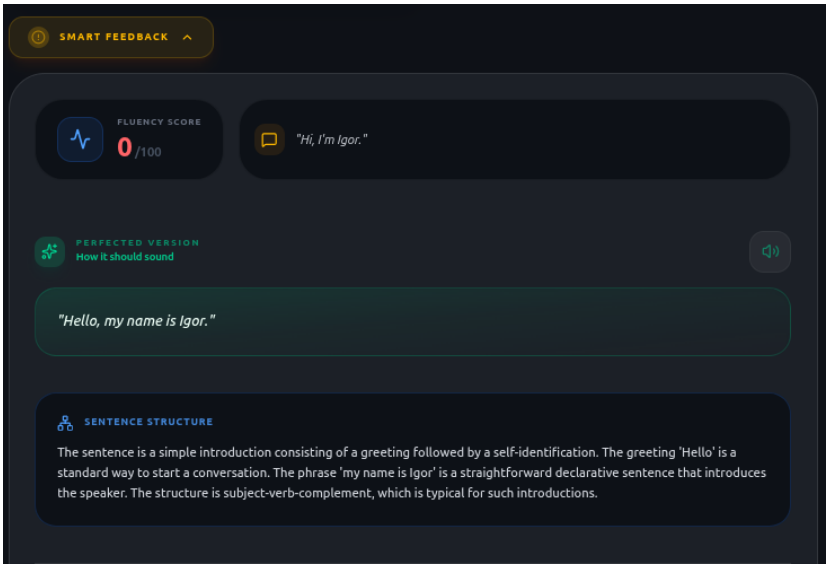

Рисунок 2.20 – Інтерфейс оцінки вільності мовлення та аналізу структури речення

Другий етап аналізу присвячений детальній корекції критичних помилок та виявленню явища інтерференції рідної мови (L1 Interference). Часто студенти намагаються будувати англійські речення за правилами української граматики, використовуючи прямий переклад, що призводить до неприродного звучання комунікації. Розгорнута система автоматично виявляє такі кальки, підсвічує їх

червоним маркером критичної помилки та надає розгорнуте текстове пояснення, чому такий підхід є некоректним з погляду англійської лінгвістики.

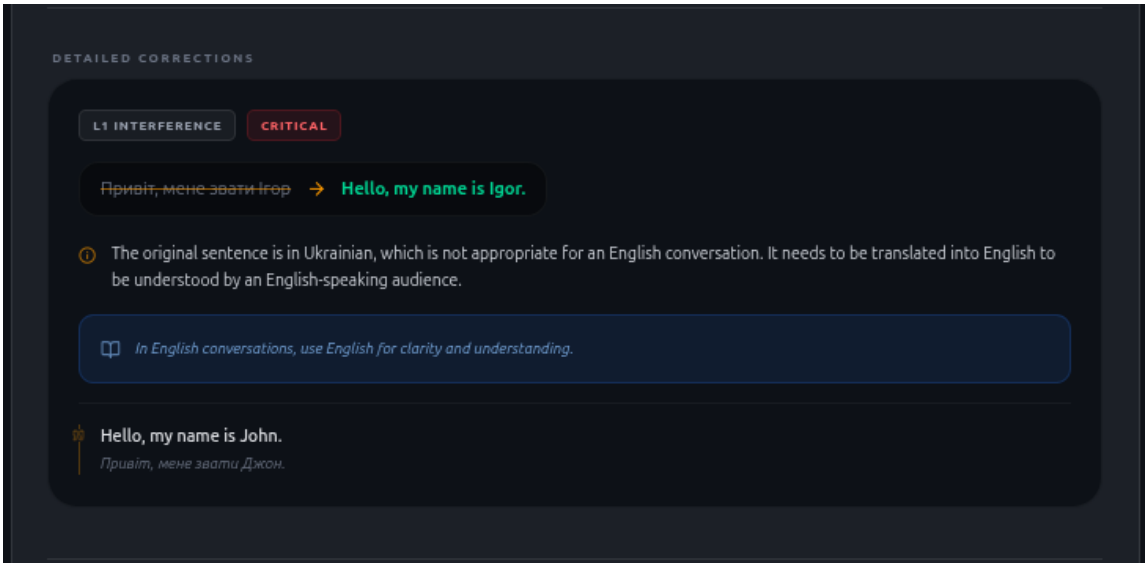

Рисунок 2.21 – Інтерфейс аналізу впливу рідної мови та критичних лексичних виправлень

Для глибокого розуміння структури мови було спроєктовано третій блок граматичного інсайту. Тут нейромережа детально розбирає часи, які студент використав або повинен був використати у своєму повідомленні. Наприклад, система пояснює логіку застосування форми Present Simple у конкретному контексті, розписує формулу побудови із вказівкою підмета та дієслова-зв'язки, а також порівнює цей час із Present Continuous, наводячи додаткові приклади використання. Це дозволяє користувачу не лише побачити свою поточну помилку, але й системно засвоїти фундаментальне граматичне правило, що її викликало.

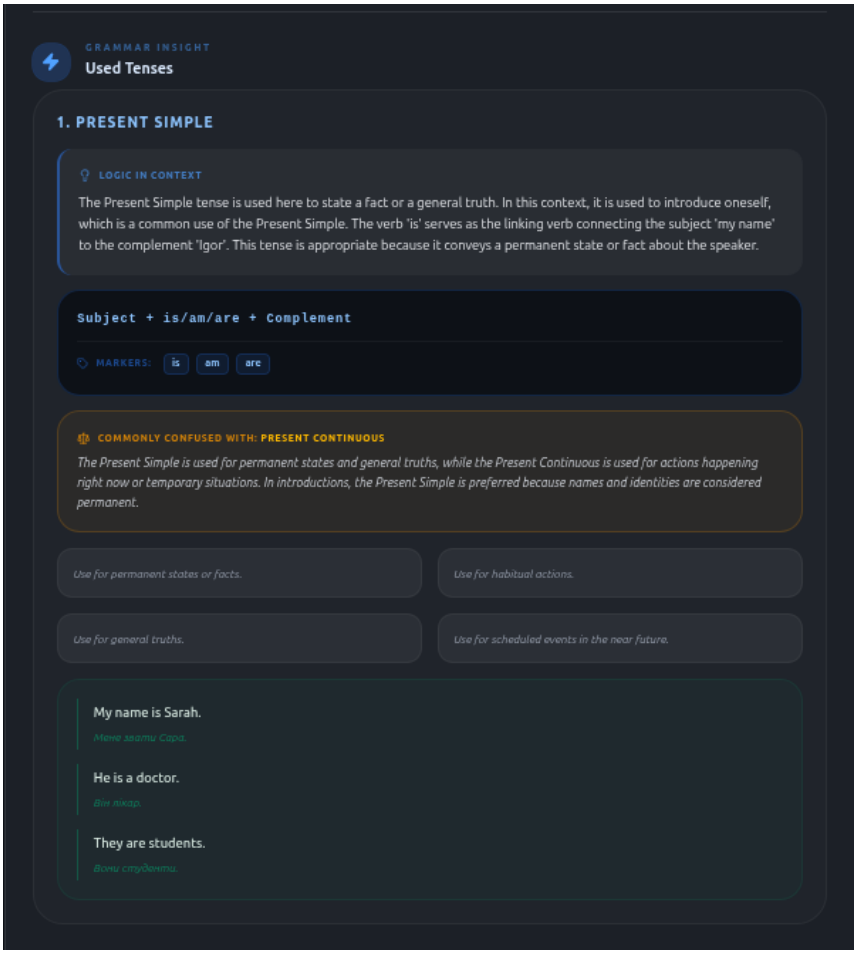

Рисунок 2.22 – Інтерфейс глибокого граматичного аналізу та розбору використаних часів

Завершальним елементом панелі аналітики є інструмент для розширення словникового запасу (Vocabulary Builder). Наприкінці кожного лінгвістичного розбору алгоритм автоматично виокремлює нові корисні слова або цілі фрази з початкової репліки користувача чи із запропонованих ідеальних виправлень. Система самостійно генерує транскрипцію, переклад, пояснює контекст використання для кожного такого виразу та наводить додаткові приклади речень. Завдяки прямому зв'язку з базою даних, студент може натиснути відповідну кнопку та миттєво додати ці терміни до свого персонального словника для подальшого вивчення.

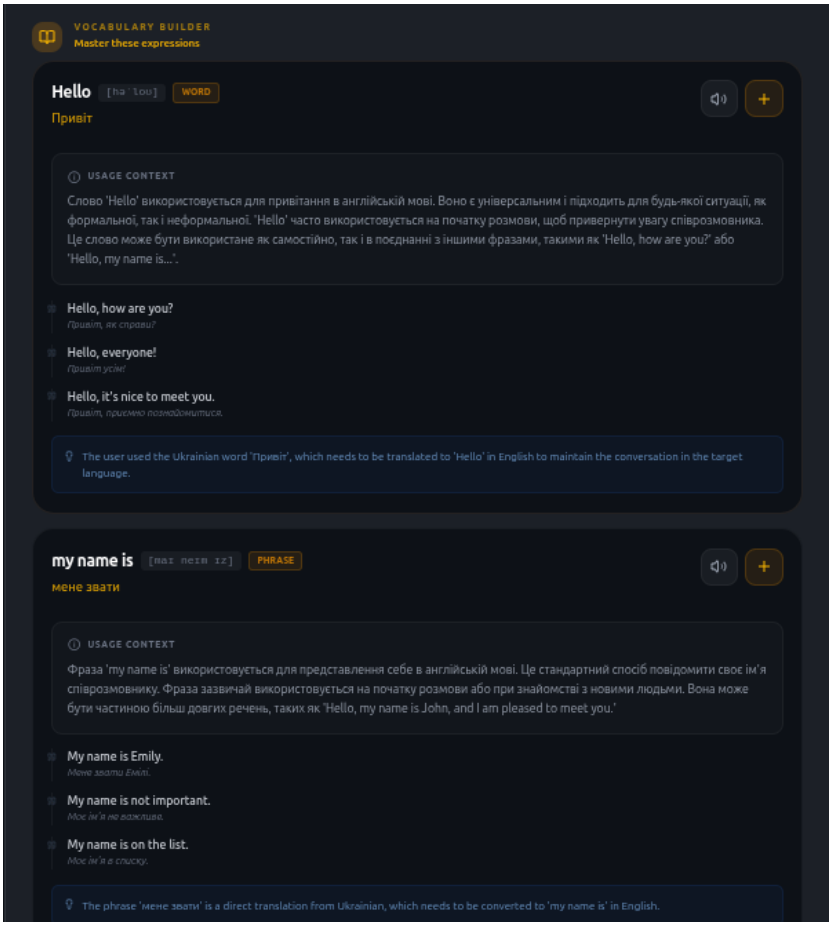

Рисунок 2.23 – Інтерфейс екстракції нової лексики та додавання слів до персонального словника

Таким чином, розроблений та інтегрований модуль ШІ-компаньйона не лише імітує природне живе спілкування, але й виконує роль персонального репетитора, який непомітно коригує студента. Поєднання текстового та голосового режимів із технологіями транскрипції створює безстресове середовище для ефективного подолання мовного бар'єра. Усі виявлені під час бесіди лексичні прогалини користувач може конвертувати у власний словник, а для більш глибокого опрацювання складних граматичних правил в архітектурі платформи передбачено окремий кабінет ШІ-вчителя на базі технології RAG, розробка якого розглядається у наступному підрозділі.

2.5 Розробка кабінету ШІ-вчителя на базі RAG-технології з динамічним доступом до мережі Інтернет

Для забезпечення глибокого розуміння граматичних правил та цілеспрямованої роботи над помилками користувача спроєктовано окремий інтелектуальний простір у вигляді кабінету ШІ-вчителя. На відміну від розмовного компаньйона, який підтримує безперервну мовленнєву практику й адаптується під тон розмови, віртуальний вчитель у додатку виконує роль точного наставника. Системний інструктаж цього агента налаштовано так, щоб він відповідав виключно українською мовою та працював лише у текстовому форматі, що необхідно для максимально ясного й однозначного пояснення складних лінгвістичних концепцій. У графічному інтерфейсі кабінету реалізовано повноцінне керування сесіями, надавши користувачеві можливість самостійно створювати нові діалоги, зберігати їх або перейменовувати для зручної організації свого навчання.

Технічною основою цього модуля визначено архітектуру Retrieval-Augmented Generation (RAG) з інтеграцією динамічного доступу до мережі Інтернет. Завдяки цьому ШІ-вчитель не просто генерує відповіді на основі статичних базових ваг моделі, а здатний у реальному часі шукати актуальну довідкову інформацію у вебпросторі. Окрім зовнішнього пошуку, забезпечено безпечний доступ агента до внутрішньої історії взаємодії студента з платформою. Якщо користувач просить проаналізувати конкретну минулу бесіду з розмовним компаньйоном, серверний алгоритм зчитує збережені репліки з бази даних PostgreSQL і формує комплексний аналітичний відгук, вказуючи на стилістичні недоліки, такі як тавтологія чи відсутність розгорнутих відповідей, та надає чіткі рекомендації щодо покращення навичок ведення бесіди.

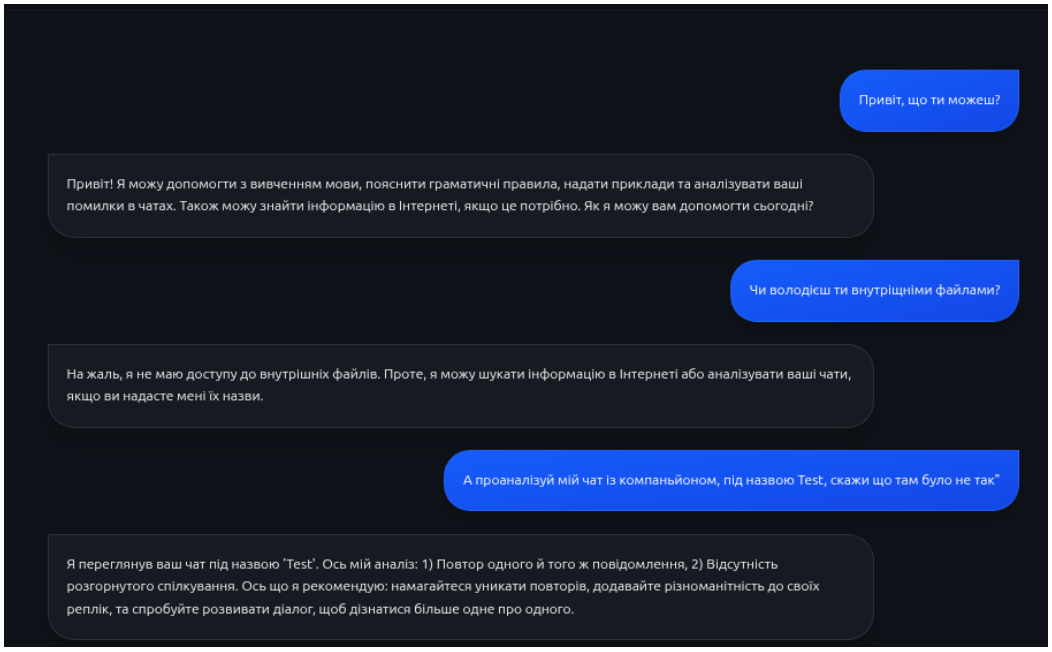

Рисунок 2.24 – Інтерфейс кабінету ШІ-вчителя з прикладом аналізу історії діалогу

Ще однією важливою функцією ШІ-вчителя є здатність генерувати тематичні добірки лексики за індивідуальним запитом. Студент може попросити надати найуживаніші слова та фрази для будь-якої специфічної сфери, наприклад, для галузі інформаційних технологій. Замість звичайного суцільного тексту, розроблений бекенд формує відповідь у вигляді спеціального структурованого пакета, який клієнтський додаток миттєво перетворює на зручну інтерактивну таблицю. У цій таблиці виводяться самі терміни, їхня транскрипція та точний переклад, а навпроти кожного рядка розташована інтерактивна кнопка для швидкого додавання обраного слова до персонального словника.

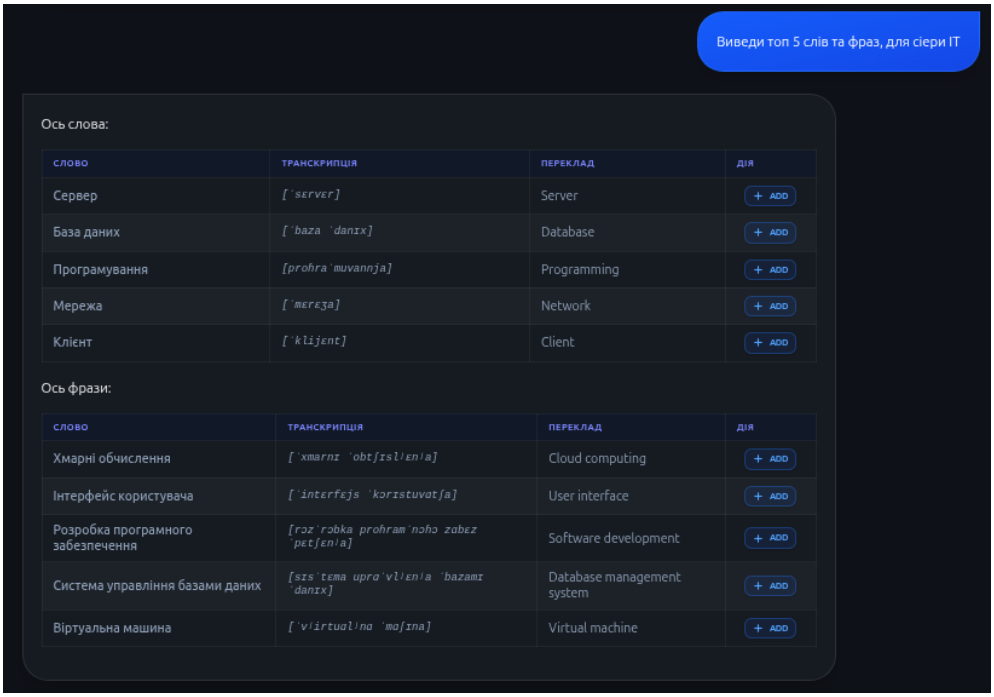

Рисунок 2.25 – Інтерфейс генерації тематичної лексики з можливістю додавання до словника

Для практичного закріплення пройденого матеріалу в алгоритм агента закладено можливість генерації персоналізованих тестових завдань. Студент може самостійно ініціювати перевірку знань з будь-якої теми, наприклад, попросити створити короткий тест на використання теперішнього простого часу. ШІ-вчитель формує нумерований список запитань із варіантами відповідей, після чого користувач надсилає свої варіанти одним зведеним текстовим повідомленням. Система аналізує отриманий рядок, звіряє його з правильними відповідями та видає детальний розбір кожної граматичної конструкції, пояснюючи логіку побудови речення та підтверджуючи рівень засвоєння теми.

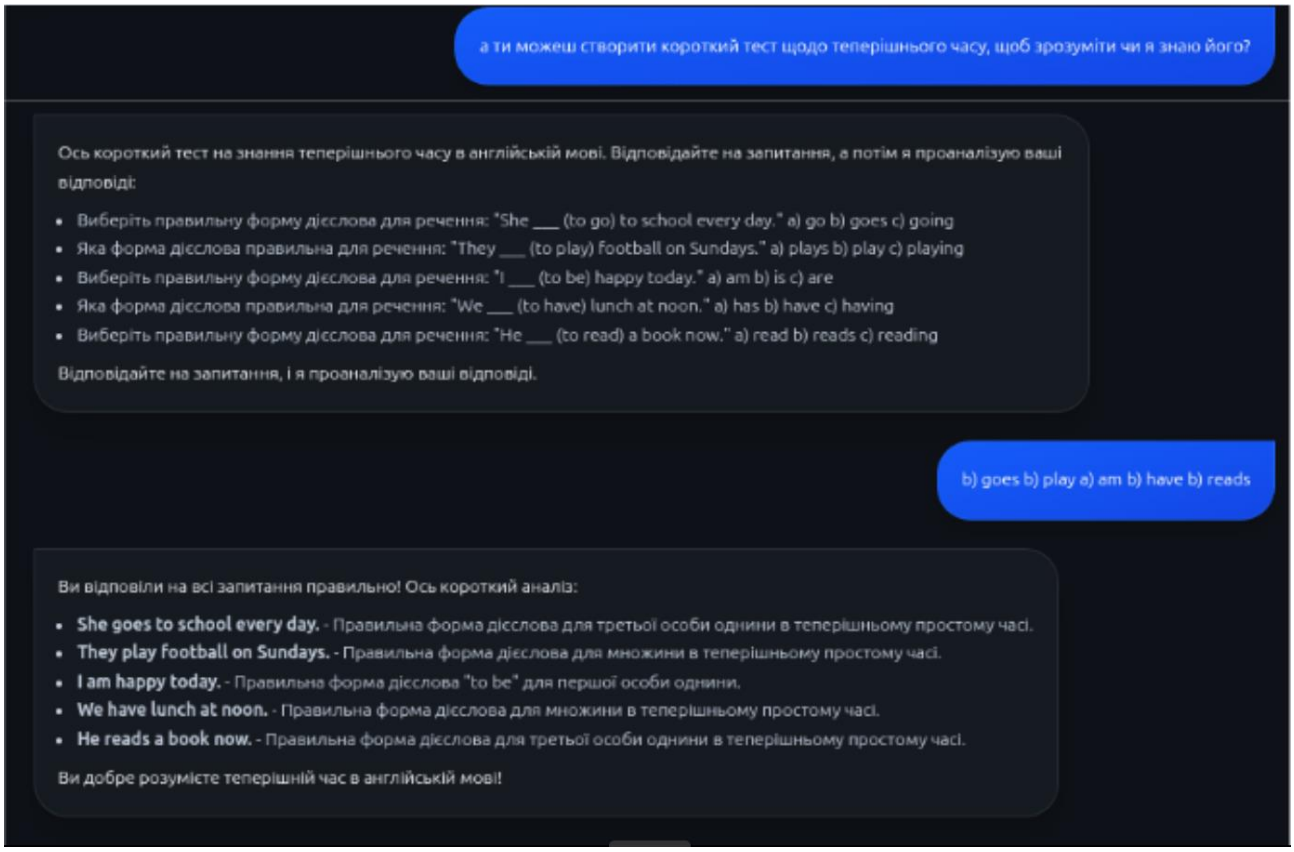

Рисунок 2.26 – Інтерфейс генерації граматичного тесту та аналізу відповідей користувача

Таким чином, розроблений кабінет ШІ-вчителя виступає потужним аналітичним та довідковим центром платформи. Завдяки використанню технології RAG, доступу до інтернету та вмінню глибоко аналізувати минулі діалоги, він забезпечує повністю індивідуальний підхід до пояснення теоретичного матеріалу.

2.6 Реалізація ігрового контуру вивчення лексики на базі мініігор та логіки первинного засвоєння

Усі нові слова та лексичні фрази, які система генерує під час взаємодії з ШІ-компаньйоном або які розбираються за допомогою ШІ-вчителя, налаштовано на автоматичне збереження у персональному словнику користувача. За кожне успішне додавання лексеми реалізовано механізм автоматичного нарахування

балів досвіду (XP). Ці показники виведено безпосередньо у профіль студента для забезпечення гейміфікації та підвищення мотивації до щоденного навчання.

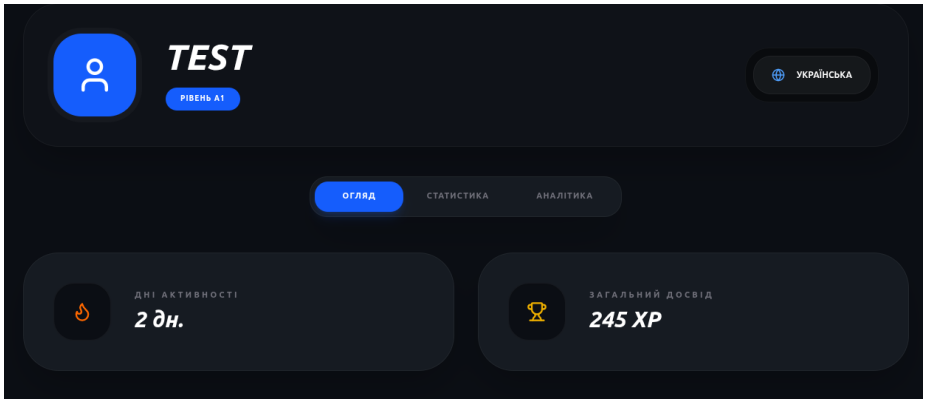

Рисунок 2.27 – Відображення накопичених балів досвіду в профілі користувача

Окрім балансу балів, профіль також слугує головним дашбордом для відстеження загального прогресу. У ньому наочно відображається кількісна статистика збереженої лексики, що дозволяє користувачу візуально оцінити обсяг роботи зі словником.

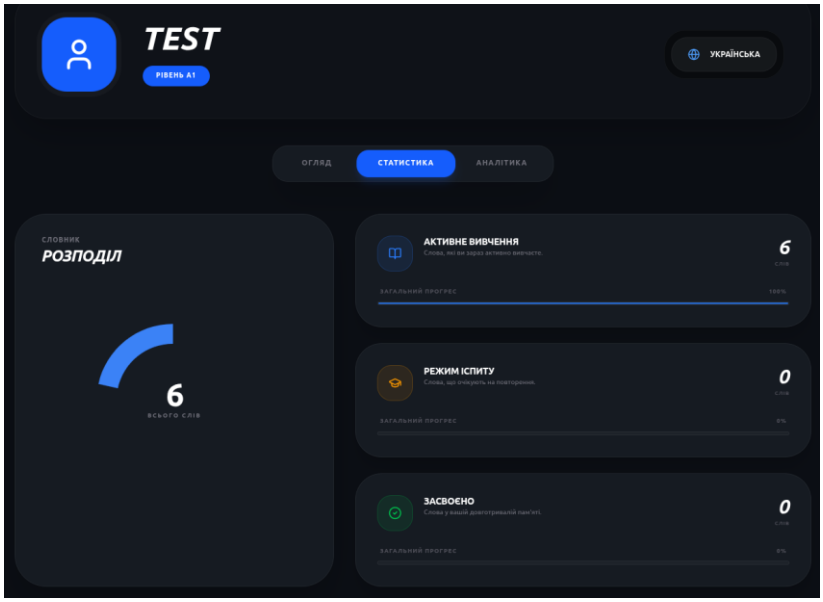

Рисунок 2.28 – Загальная статистика та кількість слів у профілі

Безпосередньо у розділі словника інтерфейс логічно розділено на два функціональні блоки: вивчення та іспити. Відповідно до закладеної бізнес-логики,

для запуску будь-якого ігрового сеансу користувачеві необхідно мати у базі даних щонайменше п'ять слів. Якщо на момент запиту в словнику немає доступних для перевірки лексем або вони перебувають у режимі інтервального очікування, запрограмовано виведення інформаційного повідомлення із точним зазначенням часу до наступного повторення.

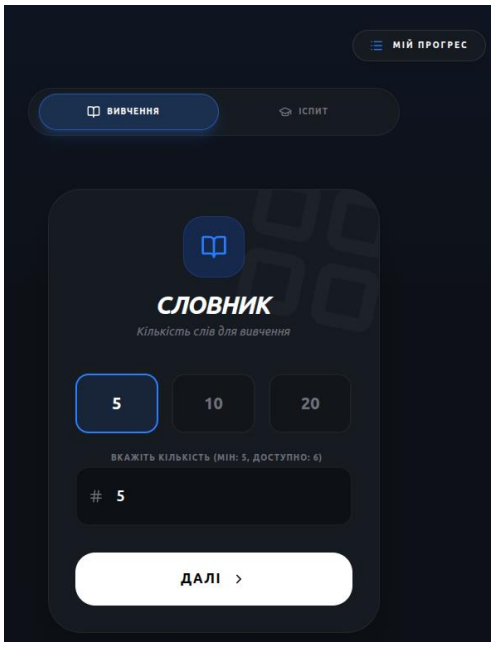

Рисунок 2.29 – Головне меню керування словником та іспитами

У цьому модулі реалізовано систему навігації по базі даних, яка дозволяє користувачеві переглядати список доданих слів та шукати потрібні лексеми через текстове поле. Для старту ігрової сесії передбачено вибір кількості слів: студент може відібрати їх вручну або згенерувати випадковий список із доступного пулу. Щоб уникнути механічного зазубрювання, алгоритм налаштовано так, що він вимагає обрати щонайменше чотири різні мініігри з дванадцяти доступних (Word Builder, Voice Check, Match Up, Synonym Match, Sentence Gap, Rapid Fire, Spell It, Missing Link, Word Master, Text Detective, Word Tower, Proofreader та Brain Teaser). З інженерного погляду в базі даних впроваджено суворе розділення поодиноких слів та довгих фраз, що дозволяє алгоритму автоматично блокувати появу довгих ідіом у режимах, де вони могли б порушити цілісність інтерфейсу додатка.

Під час сесії налаштовано фіксацію прогресу: кожна правильна відповідь додає один бал до рейтингу слова. Метою для етапу первинного засвоєння є досягнення десяти успішних проходжень, після чого слово автоматично вилучається з активного пулу і переміщується до екзаменаційного розділу. Сам іспит розроблено як суворий двокроковий контроль довгострокової пам'яті: спочатку студент пише переклад англійською мовою, а потім активується нейронна перевірка вимови через мікрофон. На складання надано лише три спроби — при їх вичерпанні слово безповоротно повертається до початкових ігор. У разі успіху слово «заморожується» за математичним алгоритмом інтервального повторення і стає доступним через збільшений проміжок часу, а статистика у профілі синхронно оновлюється.

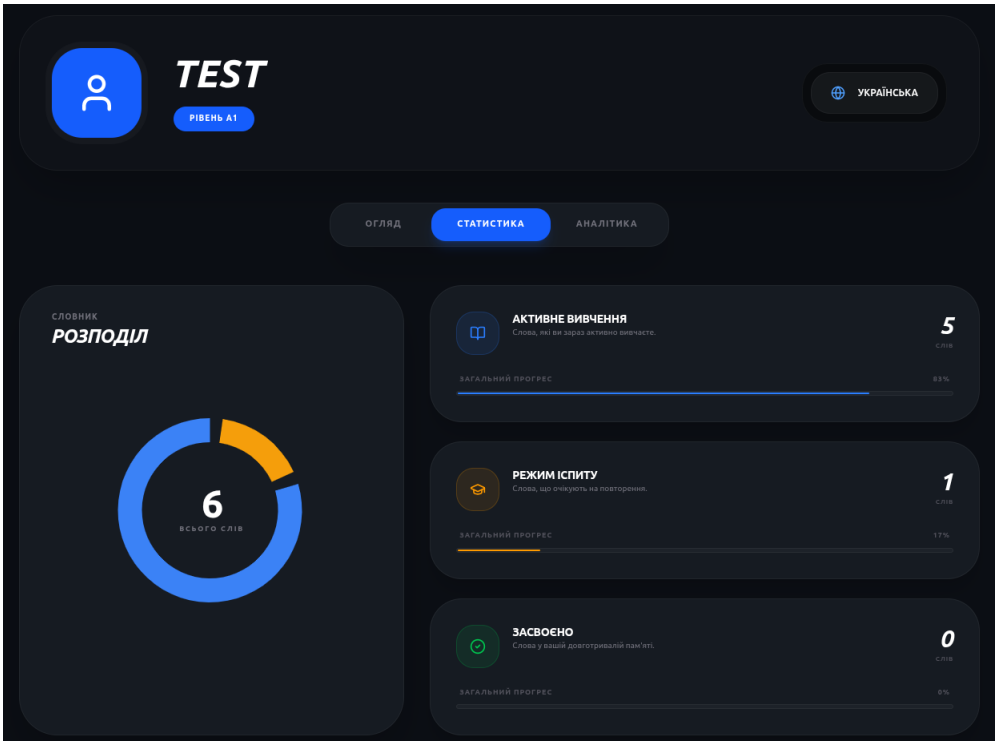

Рисунок 2.30 – Оновлена статистика засвоєних слів після складання іспитів

Таким чином, створений ігровий контур забезпечує інтерактивне первинне засвоєння лексики через поєднання візуальних, аудіальних та текстових механік. Гнучке налаштування сесій виключає монотонність навчання, а двоетапний

екзаменаційний фільтр гарантує перехід до інтервальних повторень лише надійно закріплених у пам'яті лексем.

2.7 Програмна реалізація серверної логіки двоетапних інтервальних іспитів та алгоритму Лейтнера

Для забезпечення ефективного переведення лексики з короткострокової пам'яті користувача у довгострокову реалізовано на серверному рівні платформи FLUEN AI логіку двоетапних інтервальних іспитів на базі класичного алгоритму Лейтнера. У спроєктованій базі даних під кожен запис словника виділено спеціальні стовпці для збереження поточног рівня повторення та точної дати наступного іспиту. Під час ініціалізації перевірки знань серверний скрипт за допомогою оптимізованого SQL-запиту відфільтровує лише ті слова, для яких поточний системний час перевищує або дорівнює збереженому значенню майбутнього повторення, що робить процес навчання математично вивіреним.

Весь екзаменаційний процес розділено на два етапи, обробку бізнес-логіки яких повністю покладено на бекенд FastAPI. На першому текстовому етапі сервер отримує від клієнтської частини рядок із введеним перекладом, після чого алгоритм нормалізує дані (видаляє зайві пробіли, зводить текст до нижнього регістру) та порівнює їх з еталоном у базі. При успішному проходженні текстової перевірки налаштовано автоматичний запуск другого етапу - нейронного голосового контролю. Фронтенд надсилає стиснений аудіофайл із записом вимови студента, який бекенд асинхронно передає нейромережі Whisper для транскрипції, а потім застосовує алгоритми нечіткого порівняння рядків для оцінки правильності вимови, перевіряючи і орфографічну, і фонетичну точність.

Для балансу складності в архітектуру іспиту закладено механіку обмежених спроб, створено на сервері для кожної сесії змінну-лічильник на три помилки. Якщо користувач вводить неправильний текст або демонструє неточну вимову, бекенд зменшує кількість доступних життів. При повному вичерпанні ліміту серверна логіка розцінює це як забування матеріалу, після чого транзакція бази даних

автоматично скидає рівень повторення цього слова до нуля, а його статус змінюється на етап первинного активного вивчення, повертаючи лексему в ігровий контур.

При успішному складанні обох екзаменаційних етапів без перевищення ліміту помилок запрограмовано запуск математичного алгоритму Лейтнера для динамічного збільшення часового проміжку до наступної перевірки. Програмний код зчитує поточний рівень слова та обчислює нову часову дельту для запису в базу даних: після першого успішного іспиту слово «заморожується» рівно на одну годину, після другого інтервал збільшується до чотирьох годин, далі - до дванадцяти годин, одного дня, трьох днів і так далі за експоненційною шкалою.

Після успішного проходження найвищого закладеного рівня алгоритм остаточно маркує лексему як засвоєну та переносить її в категорію довготривалої пам'яті. На фінальній стадії обробки сервер оновлює статистику у профілі користувача та автоматично нараховує підвищену кількість балів досвіду XP, виконуючи всю цю транзакційну логіку асинхронно у фоновому режимі Python для забезпечення миттєвого відгуку інтерфейсу.

2.8 Розробка служби техпідтримки, панелі адміністратора та застосування методів контейнеризації для хмарного розгортання платформи

Для забезпечення стабільного функціонування, оперативного управління та швидкого вирішення проблем користувачів на платформі FLUEN AI розроблено комплекс інженерних інструментів, що включає службу технічної підтримки та спеціалізовану панель адміністратора. На початковому етапі було реалізовано клієнтський інтерфейс служби підтримки, який дозволяє студентам у будь-який момент надіслати текстове звернення безпосередньо через модальне вікно додатка. Бекенд-сервер обробляє ці повідомлення в асинхронному режимі, автоматично фіксує точний час створення, присвоює початковий статус обробки та зберігає запит у таблиці бази даних, відображаючи історію звернень користувача у його інтерфейсі.

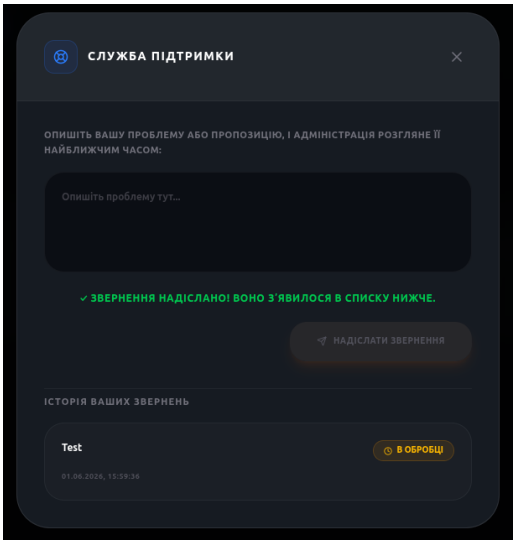

Рисунок 2.31 – Екранна форма інтерфейсу служби технічної підтримки користувачів

Для безпечного розділення прав доступу та ефективного керування застосунком виникла потреба у створенні повноцінного адміністративного контуру. З метою безпеки призначення прав адміністратора реалізовано виключно на рівні прямого доступу до бази даних. Для цього здійснено підключення до розгорнутої на сервісі Render бази даних PostgreSQL ззовні, використавши графічний клієнт DBeaver та скопіювавши унікальний зовнішній рядок підключення External Database URL із розділу налаштувань. Після ручної зміни значення логічного поля адміністратора на True у потрібному рядку таблиці користувачів, клієнтська частина платформи миттєво ідентифікує зміну прав доступу та динамічно виводить у профілі студента кнопку переходу до панелі управління.

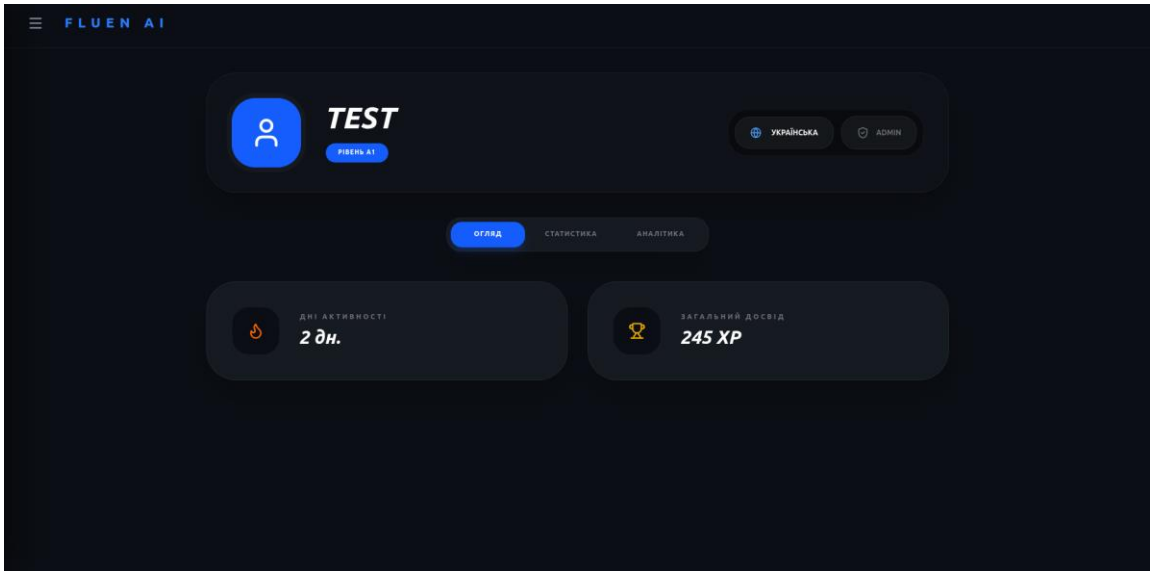

Рисунок 2.32 – Інтерфейс профілю користувача із відображенням кнопки переходу до панелі адміністратора

При натисканні на зазначену кнопку адміністратор перенаправляється до спеціалізованого кабінету управління системою. Головний екран кабінету відображає розгорнутий аналітичний консольний дашборд, який у реальному часі виводить критичні метрики життєдіяльності платформи. Тут інтегровано модулі відстеження поточного фінансового балансу OpenAI API, лічильники активного онлайн-трафіку, показники кількості унікальних відвідувачів за добу, а також складну систему візуальної телеметрії глибокого аудиту інфраструктури, яка будує графіки навантаження на основі пакетів даних з таблиці щоденної активності.

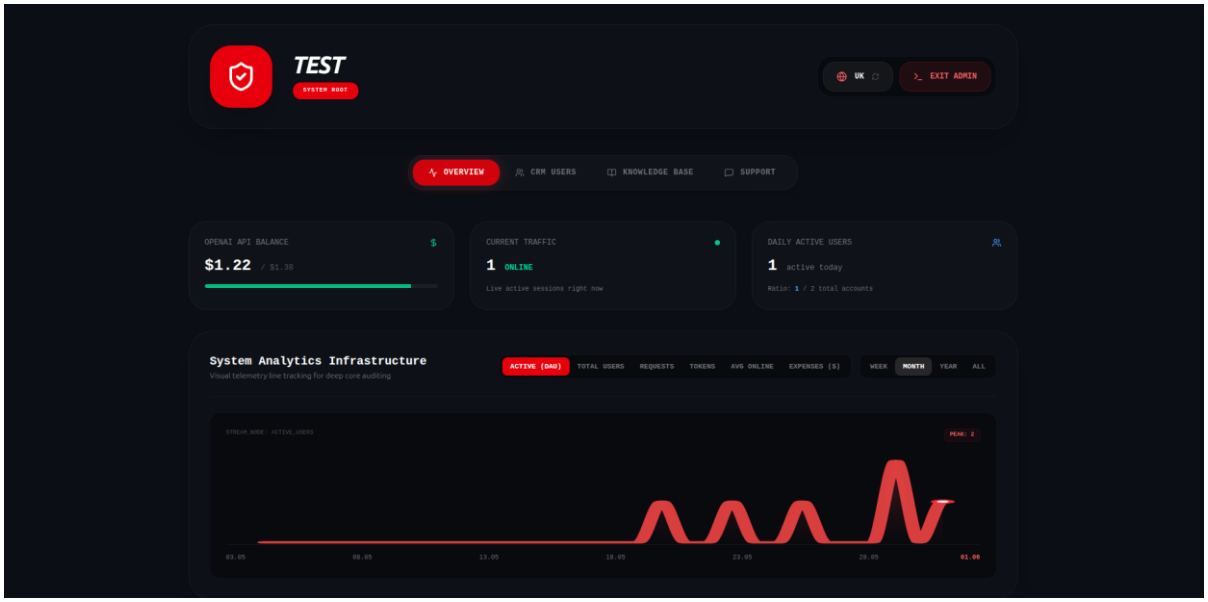

Рисунок 2.33 – Головна екранна форма панелі адміністратора із системною телеметрією

Важливою частиною розробленої панелі є вкладка Customer Relationship Management (CRM), призначена для повного моніторингу облікових записів користувачів системи. Вона містить таблицю з усім переліком зареєстрованих осіб.

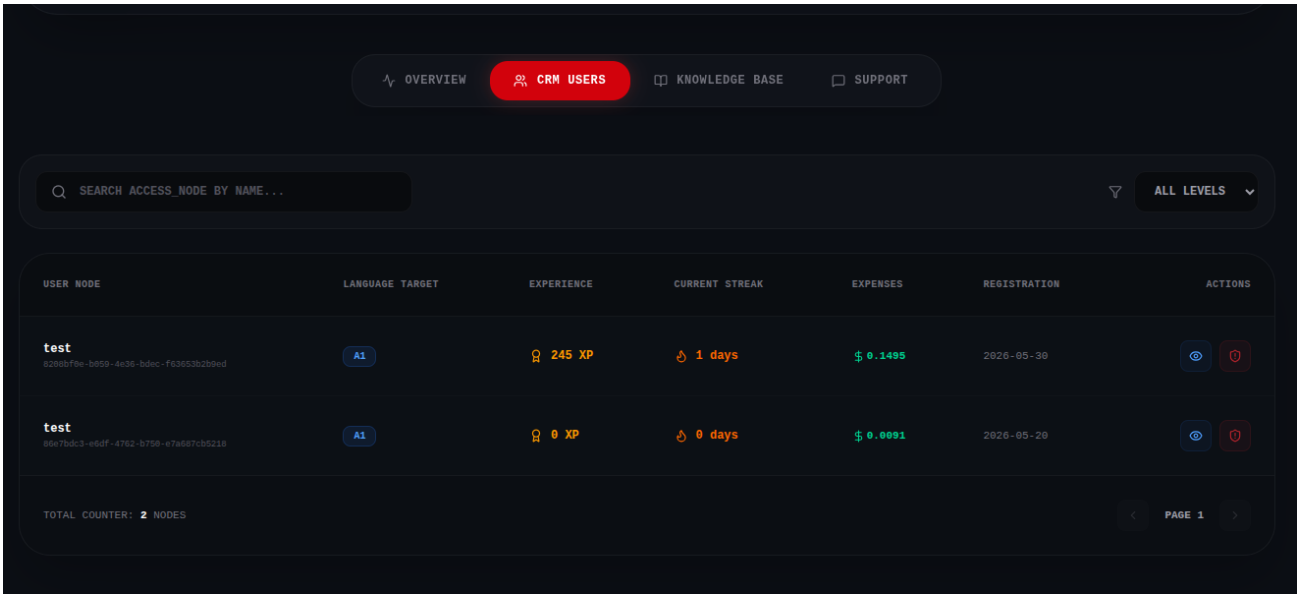

Рисунок 2.34 – Інтерфейс вкладки управління користувачами (CRM USERS) у панелі адміністратора

Для детального аналізу конкретного студента запрограмовано функцію інспектування окремого вузла мережі. При натисканні на кнопку перегляду відкривається персональне модальне вікно аудиту користувача, куди бекенд підтягує його індивідуальні графіки щоденної активності, сумарну кількість набраних балів досвіду XP, тривалість серії безперервних занять, обсяг вивчених слів, а також детальну калькуляцію фінансових витрат на токени штучного інтелекту, використані під час його навчання за обраний проміжок часу.

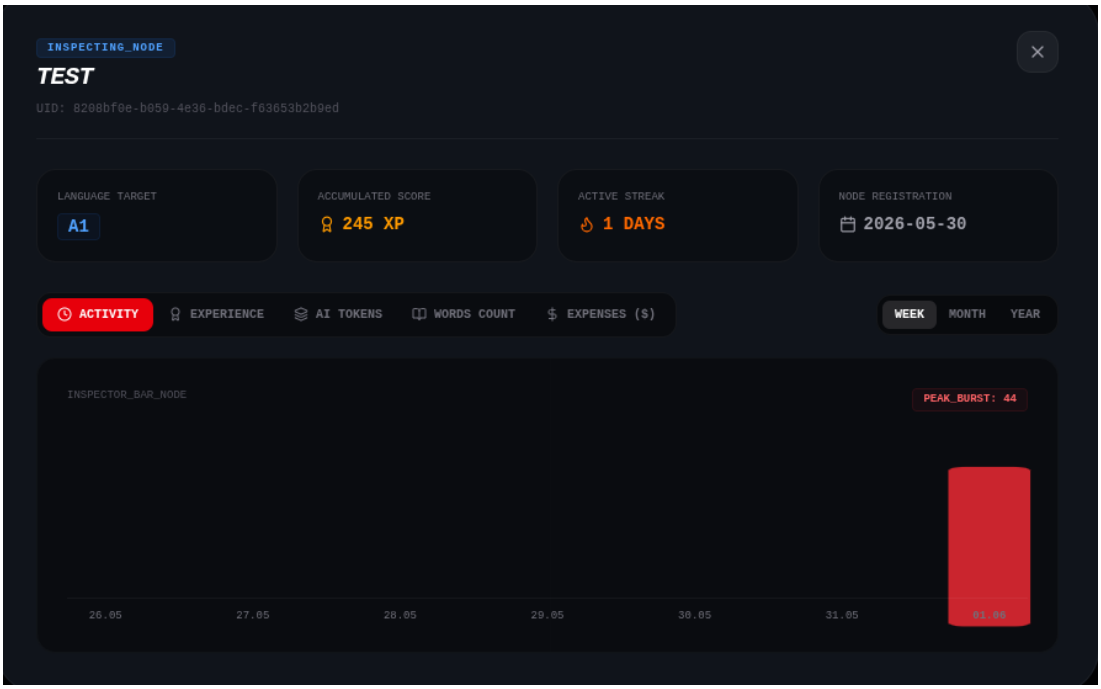

Рисунок 2.35 – Вікно детального аудиту метрик та активності обраного користувача

Для ефективного управління інтелектуальним ядром ШІ-вчителя у загальну панель було інтегровано окремий простір керування векторною базою знань (Knowledge Base). Початковий стан цього інтерфейсу відображає порожній кластер пам'яті, і якщо користувач на цьому етапі запитає агента про наявність локальних файлів, ШІ-вчитель коректно відповість, що не має доступу до додаткових завантажених джерел та використовує лише базові знання.

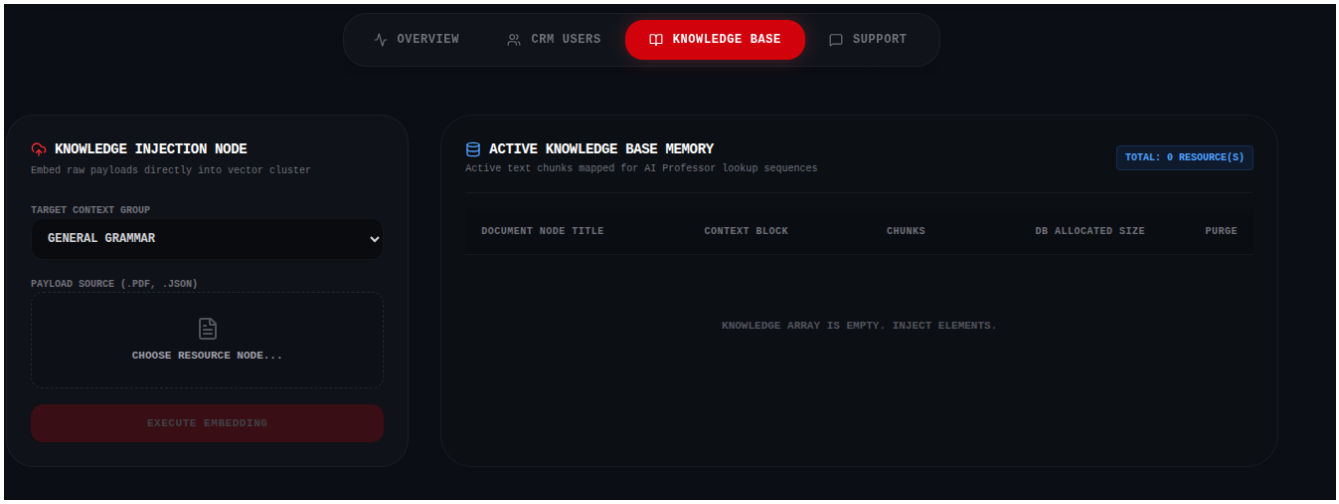

Рисунок 2.36 – Початковий стан інтерфейсу керування базою знань

З метою наповнення бази знань актуальними лінгвістичними матеріалами без зупинки та заморожування інтерфейсу користувача, спроєктовано та реалізовано асинхронний конвеєр завантаження файлів. Адміністратор обирає локальний документ у форматі PDF або JSON та запускає процес ін'єкції даних. Спеціальний фоновий скрипт від FastAPI перехоплює обробку файлу, переносячи його у внутрішній буферний потік сервера, що дозволяє адміністратору миттєво залишити поточну сторінку та вільно перейти до будь-яких інших закладок панелі управління, поки триває процес обробки.

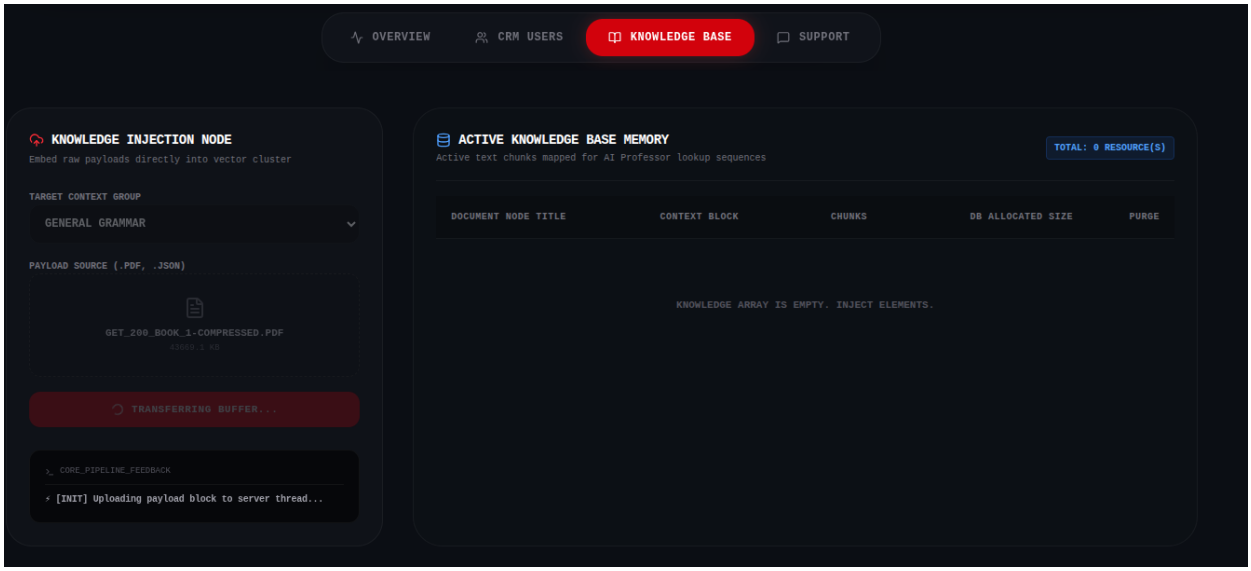

Рисунок 2.37 – Процес асинхронного завантаження та буферизації файлу на сервер

У фоновому режимі серверний скрипт повністю зчитує завантажений документ, очищує його від системного сміття, розбиває текстовий масив на логічні семантичні чанки, генерує для кожного з них числовий вектор розмірністю у 1536 ознак та записує інформацію у таблицю KnowledgeBase. Після успішного завершення операції оновлений файл з'являється у списку активної векторної пам'яті із зазначенням виділеного дискового розміру та кількості створених чанків, а також отримує інструмент для каскадного видалення. Після цього ШІ-вчитель під час діалогу підтверджує успішну синхронізацію з новими внутрішніми файлами та починає використовувати їх як пріоритетний контекст для RAG-генерації відповідей і тестів.

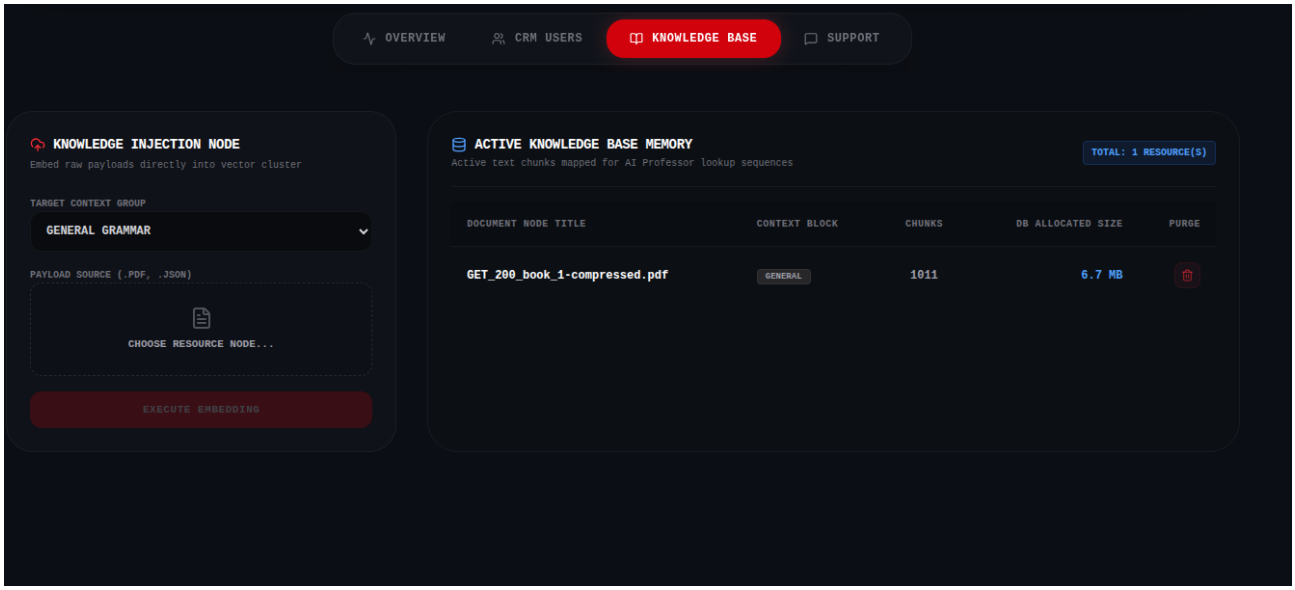

Рисунок 2.38 – Відображення успішно обробленого документа у векторній базі знань

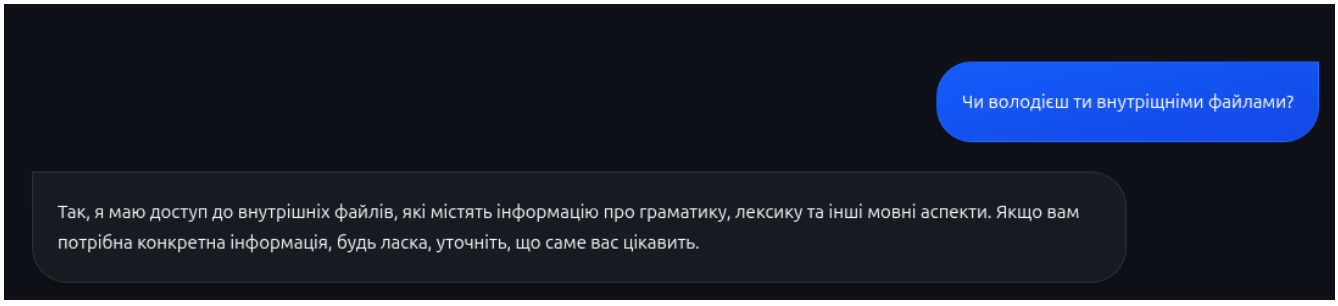

Рисунок 2.39 – Інтерфейс підтвердження доступу ШІ-вчителя до оновленого контексту знань

Для повноцінного закриття адміністративного контуру було реалізовано фінальну вкладку служби підтримки (Support), куди спрямовуються всі активні звернення від користувачів платформи. Через цей інтерфейс адміністратор системи може детально вивчити суть проблеми, написати офіційну текстову відповідь, яка надійде у кабінет студента, та змінити статус тикета на завершений. Після повної програмної реалізації всіх модулів, бекенд, окремі скрипти споживачів Kafka та аналітичні планувальники були упаковані в ізольовані Docker-контейнери та розгорнуті на хмарній платформі Render, яка завдяки наявності безкоштовного тарифного плану забезпечила стабільний CI/CD деплой без фінансових витрат.

Таким чином, розробка служби технічної підтримки, інтеграція багатофункціональної панелі адміністратора з асинхронним RAG-конвеєром ін'єкції знань та автоматизоване хмарне розгортання за допомогою Docker на сервісі Render дозволили створити повністю автономну, захищену та готову до масштабування екосистему. Створені інструменти візуальної телеметрії, управління лімітами ШІ-токенів та контролю бази даних забезпечують простоту довгострокового супроводу платформи. Після завершення всіх практичних етапів проєктування та реалізації програмного забезпечення постало завдання комплексно оцінити якість створеного продукту, тому детальне тестування роботи вебдодатка, валідація відповідей штучного інтелекту та аналіз навантажувальної стійкості архітектури розглядаються у наступному підрозділі роботи.

2.9 Комплексне тестування роботи вебдодатка, валідація відповідей ШІ та наскрізна верифікація

Фінальним та найбільш відповідальним етапом практичної реалізації інтелектуальної навчальної платформи FLUEN AI стало проведення комплексного тестування всіх її архітектурних шарів. Оскільки створена система має розгалужену структуру, що об'єднує клієнтську частину на базі React, асинхронний сервер на FastAPI та інтеграції із зовнішніми сервісами штучного інтелекту, виникла інженерна потреба впровадити надійну стратегію верифікації. Цей процес

було побудовано на основі поєднання модульних перевірок ізольованого коду бекенду та фронтенду із розгорнутим наскрізним (E2E) моделюванням повного життєвого циклу користувацьких сесій у реальному браузері. Такий підхід дозволив повністю виключити ризики виникнення помилок невідповідності форматів даних, перевірити стійкість логіки валідації та гарантувати високу стабільність інтерфейсу застосунку.

Модульне тестування серверної логіки платформи було реалізовано на базі інструментарію pytest. Автоматизованими перевірками було покрито критичні маршрути REST API, що відповідають за реєстрацію нових облікових записів та авторизацію користувачів. Під час проєктування тестів приділено особливу увагу негативним сценаріям та крайовим випадкам для перевірки надійності політики безпеки паролів. Написаний тестовий набір окремо перевіряє реакцію системи на введення занадто коротких комбінацій (менше 8 символів) та спроби встановлення паролів, що складаються виключно з цифр. У обох випадках сервер успішно блокує запити, повертаючи статус HTTP 400 та очікувану детальну помилку про недостатню стійкість. Крім того, протестовано логіку успішного входу та сценарії з некоректними обліковими даними, де система повертає статус HTTP 401 Unauthorized. Загальні результати запуску серверного тестового набору продемонстрували успішне виконання 44 тестів за 13.42 секунди із одним інформаційним попередженням щодо застарілих конфігураційних конструкцій, які згодом було оптимізовано.

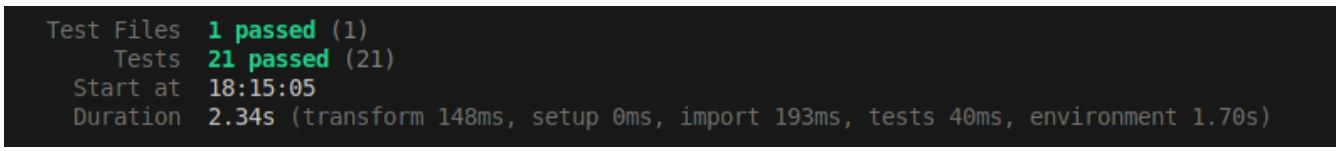

Рисунок 2.40 – Звіт про проходження модульних тестів серверної частини у фреймворку pytest

Для валідації клієнтської логіки вебдодатка та перевірки коректності взаємодії з REST API бекенду було реалізовано пакет модульних та інтеграційних

тестів на базі фреймворку Vitest. Тестуванням було охоплено 21 критичний сценарій, розподілений за основними функціональними блоками: автентифікація, керування сесіями чатів і повідомлень (Chats & Messaging), адміністрування та управління модулем бази знань (Admin & Knowledge Base), ігрова контурна гейміфікація (XP & Vocabulary), а також пряма взаємодія з захищеними серверними ендпоінтами (Native Fetch Endpoints). Для забезпечення повної ізоляції клієнтських тестів замаковано бібліотеку axios та налаштовано очищення сховища localStorage перед кожним тест-кейсом. Скрипти успішно перевіряють логіку створення чатів, їхнього перейменування через PATCH-запити, видалення, передачу FormData для аудіофайлів у Whisper, а також автоматичне зчитування токенів з локального сховища та їх ін'єкцію в HTTP-заголовки. Результати успішного запуску цього тестового набору (100% пройдених тестів за 2.34 секунди) доводять повну стабільність інтерфейсної логіки.

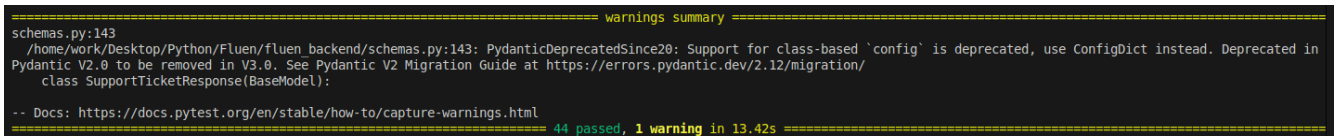

Рисунок 2.41 – Звіт про проходження модульних тестів клієнтської частини у середовищі Vitest

Окремим важливим інженерним завданням у межах тестування стала валідація відповідей від розмовного ШІ-компаньйона та ШІ-вчителя. Оскільки великі мовні моделі є стохастичними за своєю природою, у процесі розробки та тестування було зафіксовано моменти, коли GPT-4o повертав неповні структури JSON, що могло порушити стабільність графічного інтерфейсу фронтенду. Для унеможливлення таких збоїв на рівні серверних обробників FastAPI інтегровано суворий шар валідації на базі Pydantic-моделей. Якщо надісланий моделлю JSON-об'єкт не відповідав закладеній схемі або не містив обов'язкових полів аналізу помилок L1-інтерференції чи блоку Vocabulary Builder, серверна логіка автоматично перехоплювала цю помилку і відправляла повторний прихований

запит до ШІ для самокорекції, гарантуючи надійність структури даних та відсутність помилок серіалізації типу HTTP 422.

Найбільш комплексним етапом інженерної перевірки стало наскрізне тестування системи за допомогою розробленого автоматизованого сценарію test_e2e_fluen.py, повний програмний код якого наведено у додатку Б до випускної кваліфікаційної роботи. Метою цього тесту було повне моделювання поведінки реального користувача в безголовому браузері Chromium для перевірки інтеграції всіх модулів платформи у єдиний функціональний ланцюжок. Написаний тестовий скрипт послідовно виконує ініціалізацію браузера, проходить крок реєстрації та авторизації, перевіряє успішний вхід та перенаправлення в профіль, після чого відкриває модуль ШІ-компаньйона. Далі Playwright автоматизує кліки по меню, створює нову бесіду, надсилає репліку, очікує на відповідь нейромережі з генерацією Smart Feedback і видаляє чат. Сценарій безшовно перемикається на режим ШІ-вчителя, ініціює запит на вивід топ-5 слів зі сфери IT, очікує рендерингу картки лінгвістичного аналізу Mastery Insights і послідовно натискає кнопку «+ ADD» для всіх п'яти згенерованих слів, автоматично записуючи їх у словник бази даних через REST API. На фінальних кроках скрипт переходить до розділу словника, запускає ігрову сесію, перевіряє форму підтримки всередині компонента SupportModal і здійснює безпечний вихід із системи.

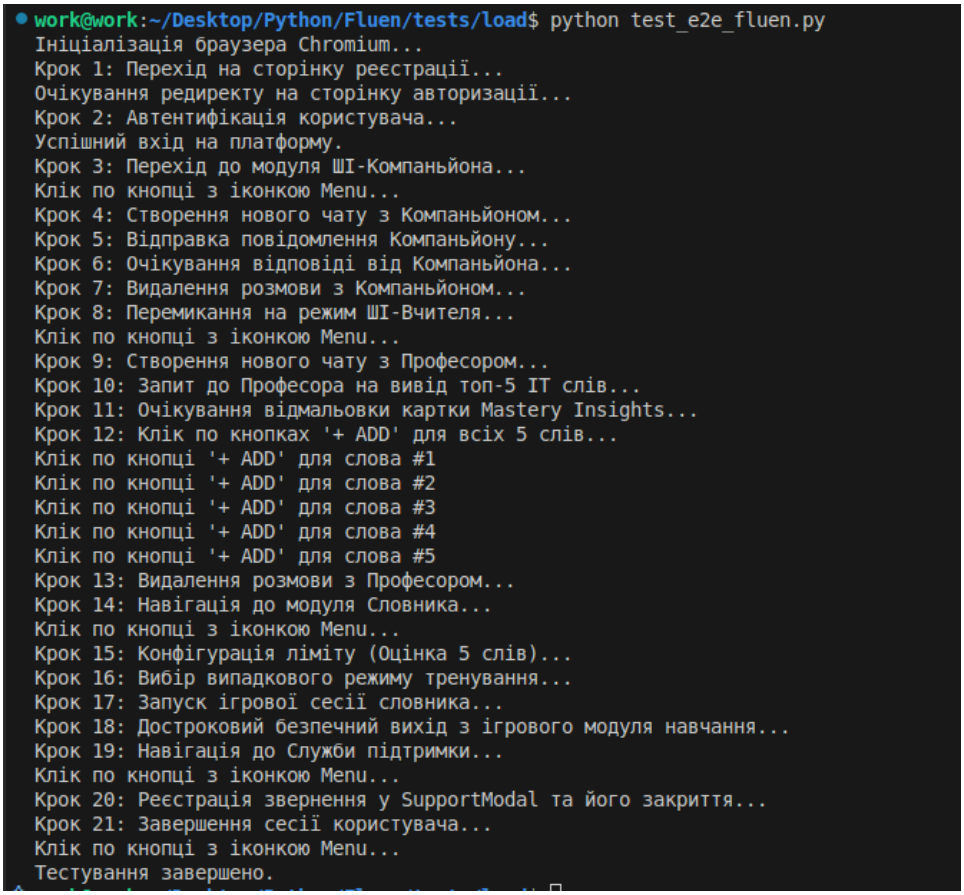

Рисунок 2.42 – Лог консольного виводу успішного виконання наскрізного сценарію автоматизації Playwright

Таким чином, проведене комплексне тестування, поєднання детальних модульних тестів для клієнта і сервера з розгорнутим наскрізним сценарієм Playwright підтвердили високу інженерну надійність розробленої вебплатформи. Автоматизація перевірок дозволила повністю виключити ризики появи критичних багів у логіці автентифікації чи серіалізації даних штучного інтелекту, а успішна інтеграція всіх компонентів доводить архітектурну стійкість застосунку. У результаті інтелектуальна навчальна платформа FLUEN AI продемонструвала стовідсоткову готовність до релізу, безпечного масштабування та ефективного практичного використання у сфері сучасної цифрової мовної освіти, повністю закриваючи всі завдання етапу практичної реалізації дослідження.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У цьому розділі наведено комплексну характеристику життєдіяльності людини в процесі взаємодії з сучасними засобами обчислювальної техніки та оцінено вплив чинників навколишнього середовища. Особливу увагу приділено деталізації вимог ергономіки до організації інженерного робочого місця оператора персонального комп'ютера, розробці раціонального режиму праці й відпочинку.

3.1 Характеристика життєдіяльності людини у системі людина - машина - середовище існування

Під час виконання будь-якої практичної роботи, пов'язаної з розробкою програмного проєкту, написанням вихідного коду, тестуванням програмного забезпечення або опрацюванням лінгвістичних даних, діяльність розробника ніколи не відбувається відірвано від зовнішніх факторів. Уся трудова активність завжди здійснюється всередині взаємопов'язаної структури, яку прийнято визначати як систему «людина - машина - середовище існування». Це базове поняття в безпеці життєдіяльності, оскільки воно демонструє, що не можна розглядати засоби обчислювальної техніки або робоче приміщення ізольовано від самого працівника. Всі ці три елементи перебувають у постійній динамічній взаємодії. Якщо хоча б один із компонентів функціонує некоректно або перебуває в незадовільному стані, страждає вся система загалом: падає продуктивність праці, зростає ймовірність виникнення помилок через втому, а здоров'я оператора опиняється під загрозою. Надійна та безпечна взаємодія в межах цієї структури є фундаментальною основою організації робочого місця [24].

Для детального розуміння структури зазначеної системи необхідно виділити кожен її елемент, щоб чітко усвідомлювати їхню інженерну та біологічну роль. Перший компонент - це людина, під якою у даному випадку розуміється безпосередньо інженер-програміст, розробник та оператор, який виконує кваліфікаційні завдання за комп'ютером. Людина в цій системі є керуючою ланкою,

яка приймає рішення, контролює робочі процеси та аналізує потоки інформації. Другий елемент - це машина, під якою мається на увазі весь комплекс технічних і програмних засобів (системний блок ЕОМ, монітор, клавіатура, маніпулятор «миша», а також додаткова периферія, мережеві пристрої та блоки безперебійного живлення). Третій елемент - це середовище існування, тобто безпосередній простір (робочий кабінет, кімната), де розташована техніка та працює оператор. Середовище характеризується комплексом параметрів, таких як якість і склад повітря, температура, відносна вологість, рівень шуму та коефіцієнт освітленості.

Для того щоб робота була безпечною і приносила якісні інженерні результати, необхідно досягти повної гармонії та відповідності між усіма трьома елементами. Проте головна проблема функціонування системи полягає в тому, що обчислювальна техніка може працювати безперервно протягом багатьох годин, а середовище залишається статичним, тоді як людський організм має чіткі фізичні та фізіологічні межі. Органи зору втомлюються від тривалого сприйняття випромінювання екрана, опорно-руховий апарат зазнає статичних навантажень від довгого перебування в одній позі, а центральна нервова система втрачає концентрацію після кількох годин інтенсивної розумової праці. Саме тому людина є найбільш вразливим елементом у цій трійці, що детально досліджується у профільних рекомендаціях з безпеки життєдіяльності розробників [25]. Більшість помилок у коді або збоїв у програмах через неуважність стаються саме тоді, коли настає перевтома, а умови мікроклімату є незадовільними.

Коли оператор тривалий час працює над проєктом за комп'ютером, на його організм діє величезна кількість різноманітних факторів, які поділяються на дві великі групи:

Фізичні чинники - зумовлені впливом технічних засобів та параметрів приміщення. До них належать нерівномірне або недостатнє освітлення, що викликає зорову втому, безперервний монотонний шум вентиляторів системного блоку чи кондиціонера, який виснажує нервову систему, а також знижена вологість повітря в опалювальний період та електромагнітні поля від дротів живлення.

Психофізіологічні чинники - пов'язані з особливостями функціонування організму під час тривалої розумової праці. Програмування та відлагодження алгоритмів штучного інтелекту вимагають високої концентрації уваги, що спричиняє тривале зорове навантаження, нервово-емоційне перенапруження та стрес через обмежені терміни. Статична сидяча поза призводить до перенапруження м'язів шиї і спини, погіршення циркуляції крові та появи болю в хребті.

Щоб мінімізувати ці шкідливі впливи, взаємодію всередині системи будують на трьох важливих принципах сумісності техніки та людини. Перший принцип - інформаційна відповідність. Це означає, що графічний інтерфейс платформи та кодова база налаштовані так, щоб інформація на екрані легко сприймалася: розміри шрифтів є оптимальними для читання, а кольорова гама не викликає втоми очей. Другий принцип - розмірна або біомеханічна відповідність. Вона полягає в тому, що всі меблі робочої зони підбираються під антропометричні параметри тіла згідно з діючими методичними рекомендаціями щодо ергономічної організації інженерних робочих місць [26]. Стіл повинен мати таку висоту, щоб руки лежали вільно і не німіли, а крісло підтримувало хребет у природному положенні. Третій принцип стосується енергетичної відповідності. Під час розробки ПЗ не виконується важка фізична праця, але витрачається багато нервової та розумової енергії. Тому органи керування (клавіатура та миша) мають бути максимально ергономічними, натискання на клавіші - м'яким, а форма миші - зручною для долоні.

Оскільки правильне налаштування всіх трьох елементів цієї системи (забезпечення нормативного середовища в кімнаті, вибір якісної техніки та турбота про здоров'я працівника) дає максимальний інженерний ефект, розробник отримує можливість працювати набагато швидше, робить менше помилок через неуважність і зберігає високу працездатність протягом усього робочого дня. Безпека життєдіяльності доводить, що раціональна організація праці є головним рушієм, який забезпечує якісне створення проєкту та ефективне функціонування сучасних інформаційних технологій.

3.2 Вимоги ергономіки до організації робочого місця оператора ПК

Для того щоб теоретичні правила безпеки життєдіяльності втілити в реальну інженерну практику, необхідно правильно облаштувати робоче місце, де безпосередньо розробляється програмна система за комп'ютером. Цим процесом займається спеціальна наука - ергономіка. Головне завдання ергономічної організації простору полягає в тому, щоб пристосувати робоче місце, меблі та техніку під фізичні й фізіологічні можливості людського організму. Тобто працівник не повинен підлаштовуватися під незручний стіл чи монітор, а навпаки, весь простір навколо нього організується так, щоб тілу було максимально природно і комфортно виконувати трудові операції.

Параметри меблів робочої зони повинні суворо відповідати вимогам охорони праці. Робочий стіл має бути достатньо просторим - на його поверхні повинні вільно розміщуватися монітор, клавіатура, миша, а також блокнот для інженерних записів та друковані схеми архітектури платформи. Висота робочого столу повинна становити 750 міліметрів від підлоги, що є оптимальним стандартом. Завдяки цьому, у сидячому положенні лікті оператора утворюють кут приблизно в дев'яносто градусів і вільно лежать на поверхні. Рекомендована глибина столу становить 850 міліметрів, що дозволяє відсунути монітор на безпечну відстань від очей.

Наступний важливий компонент - робоче крісло. Звичайні тверді стільці абсолютно непридатні для тривалої роботи, тому обов'язковим є використання підйомно-поворотного крісла. Воно повинно легко обертатися навколо своєї осі, що позбавляє від необхідності здійснювати нахили та повороти хребта під час дотягування до обладнання поруч. У кріслі необхідно чітко відрегулювати висоту сидіння під довжину ніг: у сидячому положенні стопи повинні повністю стояти на підлозі, а стегна розташовуватися паралельно їй, що захищає судини під колінами від перетискання. Спинка крісла повинна мати анатомічну форму, яка повторює природні вигини хребта і надійно підтримує поперековий відділ.

При розміщенні технічних засобів на столі слід керуватися правилами захисту зору та профілактики тунельних синдромів рук. Монітор комп'ютера є головним джерелом інформації, тому його встановлюють безпосередньо перед оператором, а не збоку, щоб уникнути тривалого статичного повороту голови, від якого перенапружуються м'язи шиї. Безпечна відстань від очей до екрана становить близько 650 міліметрів, що захищає органи зору від передчасної втоми. Висоту монітора підбирають так, щоб верхня лінія тексту на екрані знаходилася приблизно на рівні очей. Тоді центр екрану розташовується трохи нижче, погляд спрямований природно зверху вниз під невеликим кутом, що захищає очні яблука від пересихання (синдрому сухого ока).

Клавіатуру та маніпулятор «миша» розміщують на відстані близько 150 міліметрів від краю столу. Цей вільний простір необхідний для того, щоб оператор міг вільно покласти кисті та передпліччя на стіл під час тривалого написання коду. Якби клавіатура лежала на самому краю, руки постійно перебували б у висячому положенні, м'язи передпліччя перенапружувалися б, що з часом викликає хронічний біль у суглобах. Маніпулятор повинен підходити за розміром під долоню розробника для розслабленого положення пальців.

Окрему увагу приділяють освітленню робочої зони, оскільки світловий потік суттєво впливає на стомлюваність очей. Для інженерної розробки найкращим є природне денне світло, тому робочий стіл рекомендується встановлювати боком до вікна так, щоб світло падало з лівого боку. Це повністю захищає від появи сліпучих сонячних відблисків на склі монітора. Забезпечення нормативних параметрів освітленості здійснюється згідно з вимогами діючих інструкцій виробничого освітлення операторів ПК [27] та державними стандартами ДБН В.2.5-28-2018 «Природне і штучне освітлення» [28]. Для вечірнього часу використовується штучне освітлення: окрім загальної системи освітлення на стелі (люстри, світильників), на столі встановлюється настільна лампа з м'яким розсіяним світлом, що виключає появу різких тіней на клавіатурі.

Наостанок, критично важливим аспектом є правильний режим праці та відпочинку. Жодне ергономічне робоче місце не здатне захистити здоров'я, якщо

безперервно сидіти за комп'ютером по вісім годин. Оскільки загальний час роботи за день під час проєктування та реалізації системи часто перевищує чотири години, обов'язковим є впровадження чіткого регламенту пауз. Кожні півтори години безперервного написання коду необхідно робити перерву на 10-15 хвилин. Під час цієї паузи не дозволяється перемикатися на гортання соціальних мереж у мобільному телефоні; необхідно встати з крісла, пройтися по приміщенню, виконати кілька простих фізичних вправ для розминання шийно-комірцевої зони та відновлення нормального кровообігу. Також виконується гімнастика для очей - заплющення їх на кілька секунд, а потім фокусування погляду через вікно на віддалених об'єктах на горизонті. Дотримання цих ергономічних вимог дозволяє зберегти високу працездатність та відмінне самопочуття під час створення складних програмних продуктів.

Окрім профілактичних заходів мікроклімату та ергономіки, під час розробки програмного забезпечення обов'язковим є володіння базовими навичками безпеки, включаючи вивчення офіційного порядку надання домедичної допомоги при невідкладних станах згідно з чинними наказами МОЗ України [29] та внутрішніми інструкціями університету щодо першої допомоги при побутових травмах, термічних ураженнях чи переломах [30].

ВИСНОВКИ

На основі аналізу сучасних психолінгвістичних підходів до подолання мовного бар'єра, методів побудови інтерактивних компаньйонів у лінгвістичній сфері та математичного обґрунтування алгоритмів інтервального повторення було розроблено інтелектуальну навчальну платформу FLUEN AI для автоматизації процесів лінгвістичного аналізу, гейміфікованого засвоєння та довгострокового закріплення іншомовної лексики. В межах роботи проведено глибокий огляд наукових джерел, патентних розробок, методологій гейміфікації та прикладних рішень, що дозволило обґрунтувати вимоги до функціоналу та спроєктувати розподілену архітектуру системи. Зокрема, було реалізовано та інтегровано клієнт-серверний технологічний стек на базі асинхронного фреймворку FastAPI, компонентного фронтенду на React та реляційної бази даних PostgreSQL із розширенням pgvector, оптимізовані для забезпечення індивідуальної освітньої траєкторії користувача.

Розроблена система дозволяє автоматично здійснювати контекстний аналіз граматичних помилок у фоновому режимі (Smart Feedback), виявляти явища інтерференції рідної мови (L1 Interference), детально розбирати структуру використаних часів та екстрагувати нові корисні слова з реплік безпосередньо у персональний словник. Застосування технології Retrieval-Augmented Generation (RAG) з динамічним доступом до мережі Інтернет дало змогу забезпечити кабінет ШІ-вчителя функціями генерації персоналізованих граматичних тестів та тематичних добірок лексики за індивідуальними запитами користувачів. Проведений аналіз метрик автоматизованого тестування підтверсив високу надійність системи: модульні тести на базі pytest успішно верифікували 44 серверні маршрути, а модульні тести у Vitest підтвердили стабільність 21 клієнтського сценарію за рекордні 2.34 секунди з повним усуненням помилок серіалізації даних (HTTP 422). Наскрізне (E2E) тестування повного користувацького циклу через Playwright (програмний сценарій test_e2e_fluen.py винесено в додатки) довело

безпомилкову інтеграцію всіх шарів інфраструктури в реальному браузері Chromium.

Інженерний та візуальний аналіз результатів роботи ігрового контуру з 12 мініігор підтвердив високу адаптивність розробленого розподілу поодиноких слів та довгих фраз на рівні бази даних, що повністю ліквідувало ризики візуального руйнування інтерфейсу. Реалізація суворих двоетапних екзаменаційних перевірок знань із використанням нейронного розпізнавання мовлення через Whisper та обмеженням у три спроби забезпечила надійний контроль довгострокової пам'яті за математичним алгоритмом Лейтнера. Оцінка функціонування кабінету адміністратора продемонструвала стабільність асинхронного конвеєра ін'єкції знань (Knowledge Base Memory), який успішно трансформує важкі PDF-документи у 1536-вимірні вектори без блокування інтерфейсу управління, а також ефективність інструментів CRM для глибокого аудиту фінансових витрат на ШІ-токени. Додатково, у межах дослідження було детально організовано безпечні та ергономічні умови праці розробника програмного забезпечення та визначено заходи електробезпеки відповідно до чинних державних стандартів.

Результати роботи дозволяють повністю автоматизувати процеси індивідуального вивчення іноземних мов, забезпечуючи безстресове комунікативне середовище з інтелектуальним персональним репетитором. Створена платформа може бути безпосередньо впроваджена в освітній процес вищих навчальних закладів, лінгвістичних центрів або використана як гнучке корпоративне EdTech-рішення для підвищення кваліфікації спеціалістів у сферах, де критично важливим є швидке засвоєння специфічного термінологічного пулу. Застосування методів контейнеризації Docker та оркестрації Docker-compose у зв'язці з конвеєрпом CI/CD на хмарній платформі Render гарантує високу відмовостійкість, простоту масштабування та низьку вартість довгострокового супроводу інфраструктури додатка.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ефективність застосування систем інтервальних повторень (Spaced Repetition Systems) у мовній освіті: комплексний огляд. MDPI [Електронний ресурс]. — Режим доступу: URL: https://www.mdpi.com/2227-7102/14/3/214 (дата звернення: 17.05.2026).

2. Computer-assisted language learning. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Computer-assisted_language_learning (дата звернення: 02.06.2026).

3. Radford Alec, Kim Jong Wook, Xu Tao [et al.]. Robust Speech Recognition via Large-Scale Weak Supervision [Електронний ресурс]. — OpenAI, 2022. — 9 с. — Режим доступу: URL: https://arxiv.org/abs/2212.04356 (дата звернення: 02.06.2026).

4. Large language model. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Large_language_model (дата звернення: 02.06.2026).

5. Prompt engineering. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Prompt_engineering (дата звернення: 02.06.2026).

6. Server-sent events. MDN Web Docs. Mozilla Developer Network [Електронний ресурс]. — Режим доступу: URL: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events (дата звернення: 02.06.2026).

7. Forgetting curve. Wikipedia, the free encyclopedia [Електронний ресурс]. — Remim доступу: URL: https://en.wikipedia.org/wiki/Forgetting_curve (дата звернення: 02.06.2026).

8. Spaced repetition. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Spaced_repetition (дата звернення: 02.06.2026).

9. Leitner system. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Leitner_system (дата звернення: 02.06.2026).

10. Tabibian B., Upadhyay U., De A. [et al.]. Enhancing human learning via spaced repetition optimization [Електронний ресурс]. — Proceedings of the National Academy of Sciences (PNAS), 2019. — Vol. 116, No. 10. — P. 3988-3993. — Режим доступу: URL: https://www.pnas.org/doi/10.1073/pnas.1815156116 (дата звернення: 02.06.2026).

11. Gamification. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Gamification (дата звернення: 02.06.2026).

12. Gamification of learning. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Gamification_of_learning (дата звернення: 02.06.2026).

13. Subhash S., Cudney E. A. Gamified learning in higher education: A systematic review of literature [Електронний ресурс]. — Computers in Human Behavior, 2018. — Vol. 87. — P. 192–206. — Режим доступу: URL: https://doi.org/10.1016/j.chb.2018.05.028 (дата звернення: 02.06.2026).

14. Non-functional requirement. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Non-functional_requirement (дата звернення: 02.06.2026).

15. Client–server model. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Client%E2%80%93server_model (дата звернення: 02.06.2026).

16. Multitier architecture. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Multitier_architecture (дата звернення: 02.06.2026).

17. Single-page application. Wikipedia, the free encyclopedia [Електронний ресурс]. — Режим доступу: URL: https://en.wikipedia.org/wiki/Single-page_application (дата звернення: 02.06.2026).

18. Програмна архітектура та документація фреймворку FastAPI. FastAPI Tiangolo [Електронний ресурс]. — Режим доступу: URL: https://fastapi.tiangolo.com (дата звернення: 02.06.2026).

19. PostgreSQL Documentation. PostgreSQL Global Development Group [Електронний ресурс]. — Режим доступу: URL: https://www.postgresql.org/docs/ (дата звернення: 02.06.2026).

20. Документація розширення pgvector для PostgreSQL. GitHub Repository [Електронний ресурс]. — Режим доступу: URL: https://github.com/pgvector/pgvector (дата звернення: 02.06.2026).

21. Apache Kafka Documentation. Apache Software Foundation [Електронний ресурс]. — Режим доступу: URL: https://kafka.apache.org/documentation/ (дата звернення: 02.06.2026).

22. Docker architecture and containerization guide. Docker Docs [Електронний ресурс]. — Режим доступу: URL: https://docs.docker.com/get-started/overview/ (дата звернення: 02.06.2026).

23. Render cloud platform documentation. Render hosting [Електронний ресурс]. — Режим доступу: URL: https://render.com/docs (дата звернення: 02.06.2026).

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 - Програмний код конфігурації та інтеграції брокера черг Apache Kafka

```
version: '3.8'

services:
  kafka:
    image: apache/kafka:latest
    container_name: fluen_kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_PROCESS_ROLES: 'broker,controller'
      KAFKA_CONTROLLER_QUORUM_VOTERS: '1@127.0.0.1:9093'

      KAFKA_LISTENERS:
'EXTERNAL://0.0.0.0:9092,INTERNAL://0.0.0.0:29092,CONTROLLER://0.0.0
.0:9093'
      KAFKA_ADVERTISED_LISTENERS: INTERNAL://metal-rocks-
fold.loca.lt:443

      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
'EXTERNAL:PLAINTEXT,INTERNAL:PLAINTEXT,CONTROLLER:PLAINTEXT'
      KAFKA_CONTROLLER_LISTENER_NAMES: 'CONTROLLER'
      KAFKA_INTER_BROKER_LISTENER_NAME: 'INTERNAL'

      KAFKA_LOG_DIRS: '/var/lib/kafka/data'

      KAFKA_NUM_PARTITIONS: 4
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: 'true'
      LOG4J_LOGGER_ORG_APACHE_KAFKA: 'WARN'

      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1

    volumes:
      - local_kafka_data:/var/lib/kafka/data

    entrypoint: [ "/bin/sh", "-c" ]
    command: >
      "export KAFKA_CLUSTER_ID=$$(bin/kafka-storage.sh random-
cluster-id) &&
      bin/kafka-storage.sh format -t $$KAFKA_CLUSTER_ID -c
config/kraft/server.properties &&
      exec bin/kafka-server-start.sh config/kraft/server.properties"
```

ДОДАТОК Б - Програмний код сценарію наскрізного тестування платформи test_e2e_fluen.py

```
import random
import time
from playwright.sync_api import sync_playwright

def test_fluen_mega_user_flow():
    with sync_playwright() as p:
        print("Ініціалізація браузера Chromium...")
        browser = p.chromium.launch(headless=False, slow_mo=900)
        page = browser.new_page()

        page.on("dialog", lambda dialog: dialog.accept())

        random_id = random.randint(1000, 9999)
        test_email = f"student_{random_id}@fluen.ai"
        test_password = "StrongPass2026!"
        test_name = "QA Automation Expert"

        def open_sidebar():
            print("Клік по кнопці з іконкою Menu...")
            sidebar_button = page.locator("header button",
has=page.locator("svg.lucide-menu, .lucide-menu")).first
            sidebar_button.click()
            time.sleep(1.5)

        print("Крок 1: Перехід на сторінку реєстрації...")
        page.goto("https://fluen-frontend.onrender.com/register")
        page.wait_for_load_state("networkidle")

        page.get_by_placeholder("John Doe").fill(test_name)

page.get_by_placeholder("neural@address.io").fill(test_email)
        page.get_by_placeholder("••••••••").fill(test_password)
        page.locator("button[type='submit']").click()

        print("Очікування редиректу на сторінку авторизації...")
        page.wait_for_url("**/login", timeout=12000)

        print("Крок 2: Автентифікація користувача...")
        page.get_by_placeholder("Email address").fill(test_email)
        page.get_by_placeholder("Password").fill(test_password)
        page.locator("button[type='submit']").click()

        page.wait_for_url("**/main", timeout=12000)
        print("Успішний вхід на платформу.")
        time.sleep(2)

        print("Крок 3: Перехід до модуля ШІ-Компаньйона...")
        open_sidebar()
```

```
        page.locator("aside button:has-text('Компаньйон'), aside
button:has-text('Companion')").first.click()
        time.sleep(2)

        expand_chat_list_btn = page.locator("button:has(svg.lucide-
chevron-right), .lucide-chevron-right").first
        if expand_chat_list_btn.is_visible():
            expand_chat_list_btn.click()
            time.sleep(1)

        print("Крок 4: Створення нового чату з Компаньйоном...")
        create_btn = page.locator("aside button",
has=page.locator("svg.lucide-plus, .lucide-plus")).first
        if not create_btn.is_visible():
            create_btn = page.locator("button:has-text('New
Conversation'), button:has-text('Нова розмова')").first
        create_btn.click()
        time.sleep(4)

        print("Крок 5: Відправка повідомлення Компаньйону...")
        chat_input = page.locator("textarea,
input[placeholder*='message'], input[placeholder*='повідомлення'],
input[placeholder*='Введіть']").first
        chat_input.wait_for(state="visible", timeout=5000)
        chat_input.fill("Hello! як ти?")

        send_btn = page.locator("button",
has=page.locator("svg.lucide-send, .lucide-send")).first
        send_btn.click()
        time.sleep(10)

        print("Крок 6: Очікування відповіді від Компаньйона...")
        ai_response_text = page.locator(".prose").last
        ai_response_text.wait_for(state="visible", timeout=20000)
        time.sleep(5)

        print("Крок 7: Видалення розмови з Компаньйоном...")
        delete_btn = page.locator("nav button, header button, div
button", has=page.locator("svg.lucide-trash-2, .lucide-trash-2,
svg.lucide-trash")).first
        delete_btn.click(force=True)
        time.sleep(2)

        print("Крок 8: Перемикання на режим ШІ-Вчителя...")
        open_sidebar()
        page.locator("aside button:has-text('AI Професор'), aside
button:has-text('Professor'), aside button:has-
text('Вчитель')").first.click()
        time.sleep(2)

        if expand_chat_list_btn.is_visible():
            expand_chat_list_btn.click()
            time.sleep(1)
```

```
        print("Крок 9: Створення нового чату з Професором...")
        create_btn = page.locator("aside button",
has=page.locator("svg.lucide-plus, .lucide-plus")).first
        if not create_btn.is_visible():
            create_btn = page.locator("button:has-text('New
Conversation'), button:has-text('Нова розмова')").first
        create_btn.click()
        time.sleep(4)

        print("Крок 10: Запит до Професора на вивід топ-5 IT
слів...")
        chat_input = page.locator("textarea,
input[placeholder*='message'], input[placeholder*='повідомлення'],
input[placeholder*='Введіть']").first
        chat_input.wait_for(state="visible", timeout=5000)
        chat_input.fill("Виведи мені топ 5 слів для сфери IT у
вигляді таблиці.")

        send_btn = page.locator("button",
has=page.locator("svg.lucide-send, .lucide-send")).first
        send_btn.click()
        time.sleep(10)

        print("Крок 11: Очікування відмальовки картки Mastery
Insights...")
        insights_box = page.locator("div:has-text('MASTERY
INSIGHTS')").last
        insights_box.wait_for(state="visible", timeout=20000)

        print("Крок 12: Клік по кнопках '+ ADD' для всіх 5 слів...")
        time.sleep(2)
        for i in range(5):
            print(f"Клік по кнопці '+ ADD' для слова #{i+1}")
            current_btn = page.locator(".prose button:has-
text('ADD'):not([disabled]), div[class*='bg-white'] button:has-
text('ADD'):not([disabled])").first
            current_btn.wait_for(state="visible", timeout=5000)
            current_btn.click(force=True)
            time.sleep(2.5)

        time.sleep(10)

        print("Крок 13: Видалення розмови з Професором...")
        delete_btn = page.locator("nav button, header button, main
button", has=page.locator("svg.lucide-trash-2, .lucide-trash-2,
svg.lucide-trash")).first
        delete_btn.wait_for(state="visible", timeout=5000)
        delete_btn.click(force=True)
        time.sleep(3)

        print("Крок 14: Навігація до модуля Словника...")
        open_sidebar()
```

```
        page.locator("aside button:has-text('Словник'), aside
button:has-text('Vocabulary')").first.click()
        time.sleep(4)

        print("Крок 15: Конфігурація ліміту (Оцінка 5 слів)...")
        page.locator(".grid button:has-text('5'), bg-blue-500
button:has-text('5')").first.click(force=True)
        time.sleep(2)

        next_step_btn = page.locator("button:has-text('ДАЛІ'),
button:has-text('Далі'), button:has-text('Next')").first
        next_step_btn.wait_for(state="visible", timeout=5000)
        next_step_btn.click(force=True)
        time.sleep(2)

        print("Крок 16: Вибір випадкового режиму тренування...")
        mix_games_btn = page.locator("button",
has=page.locator("svg.lucide-shuffle, .lucide-shuffle")).first
        if not mix_games_btn.is_visible():
            mix_games_btn = page.locator("button:has-text('Mix'),
button:has-text('Мікс')").first
        mix_games_btn.click()
        time.sleep(2)

        print("Крок 17: Запуск ігрової сесії словника...")
        start_learn_btn = page.locator("button",
has=page.locator("svg.lucide-play, .lucide-play")).first
        start_learn_btn.click()
        time.sleep(5)

        print("Крок 18: Достроковий безпечний вихід з ігрового
модуля навчання...")
        exit_game_btn = page.locator("main button:has(svg.lucide-
log-out), div[class*='bg-red'] button, button[class*='text-red-
500']:has(svg)").first
        if not exit_game_btn.is_visible():
            exit_game_btn = page.locator("button[class*='bg-red-
100'], button[class*='bg-red-50']").first
        exit_game_btn.click(force=True)
        time.sleep(2)

        print("Крок 19: Навігація до Служби підтримки...")
        open_sidebar()
        page.locator("aside button:has-text('Служба підтримки'),
aside button:has-text('Support')").first.click()
        time.sleep(3)

        print("Крок 20: Реєстрація звернення у SupportModal та його
закриття...")
        support_textarea = page.locator("textarea").first
        support_textarea.fill("Automated system pipeline stress
check.")
        time.sleep(1)
```

```
        submit_ticket_btn = page.locator("button:has-
text('Надіслати'), button:has-text('Send')").first
        submit_ticket_btn.click()
        time.sleep(3)

         close_modal_btn = page.locator("h2:has-text('Служба
підтримки') + button, h2:has-text('Support Service') + button,
button:has(svg.lucide-x)").last

        try:
            close_modal_btn.click(timeout=3000)
        except Exception:
            print("Хрестик не клікнувся стандартним методом,
пробуємо закрити через клік по задньому фону (Оверлею)...")
            backdrop = page.locator("div.fixed.inset-0.z-\\[100\\] >
div.absolute.inset-0").first
            if backdrop.is_visible():
                backdrop.click(force=True)
        time.sleep(2)
        print("Крок 21: Завершення сесії користувача...")
        open_sidebar()
        page.locator("aside button:has-text('Вийти'), aside
button:has-text('Logout')").first.click()

        page.wait_for_url("**/login", timeout=10000)
        print("Тестування завершено.")
        browser.close()

if __name__ == "__main__":
    test_fluen_mega_user_flow()
```