Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Інтерактивний помічник викладача іноземної мови з використанням інтелектуальних агентів»

Виконав студент гр. КН-21 _____ Місюра А. В.

Керівник _____ Рубан С. А.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *29* » *січня* 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-21 *Місюрі Анастасії Володимирівні*

1. Тема кваліфікаційної роботи: *«Інтерактивний помічник викладача іноземної мови з використанням інтелектуальних агентів»*

затверджено наказом по університету № *254с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *06.06.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 92с., додатки, презентація у Microsoft PowerPoint (28 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Рубан С. А.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.03.25*
2	*Розділ 1*	*05.04.25*
3	*Розділ 2*	*01.05.25*
4	*Висновки*	*25.05.25*
5	*Оформлення кваліфікаційної роботи*	*28.05.25*
6	*Підготовка презентації та графічного матеріалу*	*20.05.25*
7	*Підготовка доповіді до захисту*	*05.06.25*

6. Дата видачі завдання: *28.01.2025р.*

Керівник _____ / Рубан С. А./

7. Запевнення: *Я, Місюра Анастасія Володимирівна, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Місюра А В./

АНОТАЦІЯ

Місюра А.В. Інтерактивний помічник викладача іноземної мови з використанням інтелектуальних агентів: кваліфікаційна робота бакалавра : 122 – Комп'ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 64 с.

Проєктування сучасного освітнього програмного забезпечення потребує впровадження інтелектуальних технологій. Розробка інтерактивного помічника викладача іноземної мови із використанням штучного інтелекту є актуальною задачею.

Метою роботи є створення вебзастосунку, що надає викладачам інструменти для автоматизованого генерування навчальних завдань з граматики, лексики, читання та спілкування за допомогою мовних моделей.

У вступі обґрунтовується актуальність теми роботи, сформульована мета та завдання для її досягнення.

У першому розділі проаналізовано проаналізовано сучасні підходи до створення освітніх інтелектуальних систем, роль мовних моделей у навчальному процесі та інструменти для побудови інтерактивних помічників викладача.

Другий розділ присвячений безпосередньому проєктуванню вебзастосунку: створенню клієнтської частини з використанням React, серверної частини на Flask, підключення до локальних моделей штучного інтелекту через Ollama, а також реалізації бази даних MSSQL для збереження облікових записів, вправ та історій чатів.

Практична цінність полягає в можливості використання застосунку викладачами для швидкого створення персоналізованих завдань.

Ключові слова:

ШТУЧНИЙ ІНТЕЛЕКТ, ІНТЕРАКТИВНИЙ ПОМІЧНИК, ОСВІТНІЙ ВЕБЗАСТОСУНОК, МОВНА МОДЕЛЬ, FLASK, REACT, OLLAMA, MSSQL, ГЕНЕРАЦІЯ ЗАВДАНЬ

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій і штучного інтелекту значно зростає потреба у впровадженні інноваційних засобів підтримки процесу навчання іноземних мов. Традиційне створення навчальних матеріалів, складання вправ та адаптація їх під індивідуальні потреби учнів вимагають великих затрат часу й ресурсів зі сторони викладача. Найкращі світові практики демонструють, що автоматизація та інтелектуальне наповнення навчальних платформ дозволяють підвищити продуктивність праці педагога, знизити ступінь монотонності рутинних операцій та поліпшити якість освітнього контенту.

Об'єктом дослідження у роботі є веб-сайт, що надає комплекс інструментів для викладача іноземної мови, а предметом — моделі й алгоритми інтеграції інтелектуальних агентів для автоматичного генерування вправ із різних лінгвістичних розділів (граматика, лексика, читання) та організації інтерактивного спілкування у форматі «говоріння».

Метою роботи є розробка й реалізація програмно-апаратного комплексу, який забезпечить:

Автоматизацію створення навчальних завдань різних типів (граматичних, лексичних, на розуміння тексту) із можливістю обраного налаштування моделі ШІ, рівня володіння мовою та тематики.

Інтерактивний чат-модуль («говоріння») з вибором ролі ШІ, моделі, рівня та теми дискусії для розвитку комунікативних навичок.

Систему збереження та управління згенерованими вправами й діалогами із опціями редагування, повторної генерації та архівування результатів.

Для досягнення поставленої мети вирішено такі завдання:

- провести аналіз існуючих платформ і середовищ генерації лінгвістичних завдань із використанням ШІ;
- спроектувати архітектуру клієнт-серверного додатка на базі React (фронтенд), Flask (бекенд) та MSSQL (БД);

- інтегрувати локальні моделі ШІ (DeepSeek v1 та v3, MISTRA, LAMA v3.1) через Ollama для обробки лінгвістичних запитів;
- реалізувати модулі вибору типу вправ, рівня англійської мови та тематики в розділах «Граматика», «Лексика», «Читання»;
- розробити інтерфейс інтерактивного чату з можливістю налаштування ролі й рівня співрозмовника-ШІ;
- забезпечити функціонал реєстрації, аутентифікації, збереження й керування сесіями користувачів;
- оформити дизайн сайту з приємними анімаціями та інтуїтивною навігацією для викладачів іноземних мов.

Реалізація вказаних завдань дозволить створити ефективний інструмент для підвищення якості та оперативності підготовки навчальних матеріалів і забезпечить новий рівень взаємодії між викладачем та цифровим помічником.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СУЧАСНОГО СТАНУ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ У ГАЛУЗІ АВТОМАТИЗАЦІЇ НАВЧАННЯ ІНОЗЕМНИХ МОВ

1.1 Аналіз предметної області

1.1.1 Загальна характеристика досліджуваного процесу

Вивчення іноземної мови - це багатогранний і динамічний процес, який передбачає поступовий розвиток комунікативної компетенції в кількох взаємопов'язаних сферах. Ці сфери зазвичай включають поповнення словникового запасу, граматичну точність, розуміння на слух, читання, вільне володіння мовою та письмове мовлення. Хоча шлях кожного учня унікальний, загальна структура мовної освіти має тенденцію слідувати чітко визначеній послідовності, заснованій на стандартизованих рівнях володіння мовою, таких як ті, що описані в Загальноєвропейських рекомендаціях з мовної освіти (CEFR).

З системно-орієнтованої та обчислювальної точки зору, процес вивчення мови можна розуміти як послідовність поетапних взаємодій між вхідними даними та результатами навчання, опосередкованих керованою практикою та зворотним зв'язком. Кожен мовний рівень вимагає систематичного введення нових лінгвістичних структур і лексики, які закріплюються за допомогою вправ, що сприяють розпізнаванню, контрольованому продукуванню і, нарешті, спонтанному використанню в контексті. Це робить процес навчання високомодульним і заснованим на моделях - характеристиках, які особливо добре піддаються алгоритмічному представленню і створенню контенту за допомогою ШІ. [2]

В інституційних умовах викладання іноземної мови, як правило, організовано в тематичні блоки, кожен з яких зосереджений навколо комунікативної мети, що підтримується конкретними граматичними пунктами

та наборами лексики. Уроки часто включають кілька типів завдань - такі як встановлення відповідності, заповнення пропусків, переклади, керовані діалоги та відкриті запитання - і всі вони відповідають повторюваним структурним шаблонам. Ця закономірність дозволяє класифікувати завдання та матеріали у багаторазові формати, якими можна ефективно керувати та створювати за допомогою інтелектуальних систем.

1.1.2 Основні виклики, з якими стикаються викладачі іноземних мов при створенні персоналізованих навчальних матеріалів

У сучасному освітньому середовищі викладачам доводиться балансувати між якісним викладанням, що відповідає рівню студентів, та вимогами зростаючого робочого навантаження і швидких технологічних змін. Незважаючи на те, що ресурсів зараз більше, ніж будь-коли, викладачі іноземних мов все ще стикаються з постійними проблемами, які перешкоджають як ефективності, так і здатності адаптувати навчання до індивідуальних особливостей студентів.

Однією з найбільш нагальних проблем є постійна потреба у створенні індивідуальних навчальних матеріалів для студентів з різним рівнем володіння мовою. Уявіть собі вчителя, який працює з класом зі змішаними здібностями: початківцям можуть знадобитися прості структури речень, тоді як просунуті учні потребують нюансованих граматичних пояснень і складної лексики. Задоволення цих різноманітних потреб означає підготовку кількох варіантів граматичних вправ, словникових списків та розмовних завдань - кожен з яких має бути адаптований до рівня складності. Цей процес не лише забирає багато часу, але й виснажує психіку, оскільки викладачам часто доводиться розробляти абсолютно окремі набори матеріалів для різних груп. [1]

Сучасна освіта робить акцент на персоналізованому навчанні, яке адаптується до унікальних потреб, інтересів та прогресу учнів. Проте в реальності вчителі стикаються зі щільним графіком, великим адміністративним

навантаженням і переповненими класами, що робить справжню індивідуалізацію майже неможливою. Ідеальним варіантом була б розробка вправ, пристосованих до слабких сторін кожного учня, або надання детального, персоналізованого зворотного зв'язку, але більшість викладачів змушені покладатися на узагальнені матеріали. І в результаті, багато студентів не отримують цілеспрямованої практики, необхідної для подолання їхніх конкретних проблем. [8]

1.1.3 Можливості оптимізації навчального процесу шляхом автоматизації

Викладання іноземної мови передбачає незліченну кількість повторюваних завдань, які, хоч і є важливими, але можуть забирати час та енергію викладача. Щодня викладачі розробляють вправи, складають словникові списки, створюють приклади діалогів та моделюють розмови - все це дуже важливо для навчання, але часто шаблонно за структурою. Ці завдання, хоч і є педагогічно обґрунтованими, забирають багато часу, який можна було б витратити на те, що справді важливо: розуміння індивідуальних потреб учнів, вдосконалення стратегій уроку та сприяння глибшому залученню учнів. Питання полягає в тому, як оптимізувати ці процеси, не жертвуючи якістю, і тут автоматизація стає потужним союзником.

Одне з найпростіших застосувань автоматизації - створення вправ. Граматичні вправи, підбір лексики, трансформація речень і запитання з декількома варіантами відповідей - все це відбувається за передбачуваними шаблонами. Замість того, щоб створювати кожну вправу вручну, викладачі можуть використовувати шаблони, засновані на лінгвістичних правилах або моделях, керованих штучним інтелектом, які динамічно генерують контент на основі теми, складності та навчальних цілей. Це не лише заощадить час, але й забезпечить послідовність і дозволить швидко вносити корективи відповідно до прогресу учнів.

Підбір словника - ще одна сфера, яка потребує оптимізації. Викладачі часто перебирають списки слів, підлаштовуючи їх під різні рівні володіння мовою або тематичні блоки. Інтелектуальна система, навчена за стандартами Загальноєвропейських рекомендацій з мовної освіти або базами даних, узгодженими з навчальною програмою, може миттєво запропонувати відповідні лексичні одиниці з контекстуальними прикладами. Таким чином, викладачі витрачають менше часу на складання списків і більше часу на викладання цих слів у змістовний, цікавий спосіб.

Створення діалогів також виграє від цього. Удосконалені мовні моделі можуть створювати реалістичні, відповідні рівню діалоги, пристосовані до конкретних граматичних моментів або тем. Ці діалоги слугують готовими прикладами для вправ на розуміння, рольових ігор або письмових завдань. Більше того, їх можна змінювати в режимі реального часу - спрощувати для початківців або збагачувати нюансами для просунутих учнів - без необхідності переписувати їх з нуля.

Можливо, найбільший трансформаційний потенціал закладений у діалогових інструментах, керованих штучним інтелектом. Уявіть собі асистента, який допомагає вчителю спонтанно генерувати вправи під час уроку, імітувати рольові ігри або надавати приклади миттєвого зворотного зв'язку. Такі інструменти не замінять вчителя, а скоріше виступатимуть у ролі динамічного другого пілота, забезпечуючи більш адаптивне та оперативне навчання.

Зрештою, автоматизація в мовній освіті - це не усунення людського фактору, а його посилення. Переклавши повторювані завдання на інтелектуальні системи, викладачі можуть спрямувати свої зусилля на те, що у них виходить найкраще: спрямовувати, надихати і спілкуватися зі своїми студентами. У такій динамічній галузі, як викладання мов, де потреби кожного учня є унікальними, такий зсув може мати вирішальне значення. [5]

1.2 Огляд існуючих програмних засобів для автоматизації створення навчальних завдань

1.2.1 Використання штучного інтелекту в освітньому середовищі: потенціал для автоматизації завдань

Останніми роками спостерігається поступова інтеграція штучного інтелекту в освітнє середовище, але не як заміна педагогам, а як допоміжний інструмент, що сприяє зменшенню їхнього навантаження. У сучасному освітньому середовищі вчителі стикаються зі зростаючим набором вимог. Це і розробка цікавих уроків, і адаптація контенту до різноманітних потреб учнів, і виконання адміністративних завдань. Водночас вони прагнуть дотримуватися освітніх стандартів. Як наслідок, спостерігається помітне зростання використання штучного інтелекту (ШІ) для виконання різних завдань, включаючи планування уроків і створення вправ. Серед найперспективніших розробок - великі мовні моделі (ВММ), які продемонстрували вражаючу здатність справлятися з рутинними завданнями, тим самим звільняючи викладачів, щоб зосередитися на тому, що справді має значення: змістовній взаємодії зі студентами та більш глибокому педагогічному залученню. [13]

Цінність LLM, таких як GPT, LLaMA, DeepSeek і Mistral, в освіті пояснюється їхньою адаптивністю. Ці моделі дозволяють створювати вправи, які відповідають конкретним мовним чи граматичним цілям, з рівнями складності, адаптованими для різних студентів. Чи потрібно пропонувати набір вправ на заповнення пропусків для студентів на рівні A2? В якості альтернативи, для просунутого класу граматики можна використати серію розмовних запитань. Магістр може створити їх за лічені секунди, що значно скоротить час, який викладач міг би витратити на їх створення вручну. Для викладачів іноземних мов, які часто використовують подібні формати вправ на різних уроках, така автоматизація є значним нововведенням. Замість того, щоб витрачати час і зусилля на створення нових блоків, викладачі можуть

використовувати штучний інтелект для автоматизації повторюваних завдань і зосередитися на вдосконаленні контенту, щоб він краще відповідав навчальним потребам своїх студентів.

Однак справжня перевага цих інструментів полягає в їхній здатності до взаємодії. Сучасні освітні платформи сприяють співпраці між викладачами та штучним інтелектом у режимі реального часу, дозволяючи уточнювати результати, налаштовувати параметри та генерувати завдання на місці. Така динамічна взаємодія гарантує, що викладачі зберігають повний контроль над навчальними матеріалами, користуючись при цьому швидкістю та ефективністю ШІ. Щоб проілюструвати це, можна розглянути сценарій, в якому викладач наказує зосередити вправу виключно на теперішньому доконаному часі або накладає обмеження на словниковий запас, обмежуючи його словами, які були введені раніше. Такі обмеження змушують штучний інтелект створювати вправи, які ідеально відповідають цілям навчальної програми, що робить його справжнім помічником, а не автономним творцем контенту. [9]

1.2.2 Порівняльний аналіз наявних рішень

Одним з ключових елементів системи є інтелектуальний агент, який визначається як програма, що може функціонувати самостійно, взаємодіяти з іншими програмами та взаємодіяти з користувачем у режимі реального часу. У контексті іншомовної освіти особливий інтерес викликає роль віртуальних співрозмовників, рецензентів та фасилітаторів. Функції цих агентів еволюціонують, а їхня роль розширюється.

Табл.2.1. Порівняння відомих ШІ сервісів

Критерій	ChatGPT (OpenAI)	Grammarly	Duolingo Max	Otter.ai	Perplexity AI
Генерація навчальних матеріалів	Можливість створення вправ за запитом, проте відсутня спеціалізація для педагогічних потреб	Відсутня	Адаптивні вправи з поясненнями (на базі GPT-4)	Відсутня	Відсутня
Інструменти для педагогів	Відсутні	Відсутні	Орієнтовано на самостійне навчання	Відсутні	Відсутні
Адаптація до рівня знань	Можливий лише при явному зазначенні рівня	Відсутня	Автоматична адаптація (алгоритми Duolingo)	Відсутня	Відсутня
Інтерактивний режим для мовної практики	Високоякісна текстова взаємодія, проте без педагогічної структури	Відсутній	Рольові діалоги (Duolingo Max)	Відсутній	Можливість відповідей на запитання, але без навчального контексту
Вибір моделі ШІ	Використовується єдина модель (GPT)	Фіксована модель	Використовується GPT-4	Фіксована модель	Фіксована модель
Спеціалізація на мовному навчанні	Універсальний інструмент без спеціалізації	Корекція граматики та стилю	Мовні курси з елементами ШІ	Транскрипція аудіо	Пошукова система з елементами ШІ
Цільова аудиторія	Універсальний інструмент	Користувачі, що працюють з текстом	Самостійні учні	Користувачі, що потребують транскрибації	Дослідники та аналітики

ChatGPT (OpenAI) - це надзвичайно гнучка мовна модель, яка може створювати широкий спектр завдань, включаючи граматичні вправи, словникові головоломки, реалістичні розмови та нюансовані, контекстно-залежні відповіді. Найвизначнішою особливістю програми є її лінгвістична гнучкість, яка дозволяє їй розуміти і генерувати текст у спосіб, що максимально нагадує людське спілкування. Отже, викладачі можуть використовувати цю технологію для розробки цікавих вправ і практичних навчальних завдань.

Однак слід зазначити, що впровадження ChatGPT в освітніх установах не є простим рішенням. На відміну від спеціалізованих освітніх платформ, він не пропонує готових схем уроків, систем автоматичного оцінювання або адаптивного масштабування складності, якщо тільки викладач не налаштує ці параметри вручну. Таким чином, викладачі мають самостійно організовувати навчальний процес, включаючи розробку навчальних матеріалів, узгодження педагогічних підходів із встановленими цілями навчальної програми та адаптацію контенту відповідно до рівня підготовки студентів. Хоча ChatGPT є ефективним інструментом для створення ідей, він більше схожий на

високотехнологічний колаборатор, ніж на самостійний навчальний інструмент.[10]

Grammarly, навпаки, працює як прецизійний інструмент редагування, сфокусований на вдосконаленні тексту. Його алгоритми чудово виявляють граматичні помилки, відшліфовують пунктуацію та оптимізують стилістичну ясність - по суті, виступаючи в ролі тренера з написання текстів у режимі реального часу. На відміну від генеративних моделей ШІ, Grammarly не створює планів уроків, не імітує розмовну практику і не коригує рекомендації на основі рівня користувача за шкалою. Хоча цей інструмент є безцінним для шліфування чернеток, йому бракує інструментів, необхідних для розробки адаптивних навчальних програм або інтерактивних методів навчання.[3]

Duolingo Max є прикладом системного підходу до викладання мови шляхом інтеграції адаптивних технологій навчання. Платформа включає в себе складні можливості обробки мови в рамках структурованої навчальної програми, надаючи контент, який автоматично масштабується за складністю відповідно до продемонстрованого рівня володіння мовою учнями. Цей механізм динамічного коригування сприяє індивідуальній траєкторії розвитку навичок. Інтерактивні компоненти, включно з імітацією розмовних ситуацій, надають контекстуальні можливості для практики, які зміцнюють навички сприйняття та продукування мовлення. Хоча ці функції ефективно підтримують автономне навчання, платформа наразі пропонує обмежені можливості для кастомізації під керівництвом викладача, включаючи створення індивідуальних навчальних матеріалів або всебічну аналітику успішності учнів.[7]

Otter.ai має дещо іншу мету: це насамперед програма для транскрибування, яка перетворює усне введення на письмовий текст. Хоча програма не призначена спеціально для викладання мов, її можна ефективно використовувати в освітніх цілях - наприклад, для транскрибування лекцій, презентацій або дискусій між студентами. Ця можливість може покращити сприйняття на слух і навички конспектування, особливо для більш просунутих

користувачів. Однак платформа не створює навчальних матеріалів, не адаптується до різних рівнів володіння мовою та не відповідає педагогічним методикам, що обмежує її пряме застосування в іншомовній освіті.[12]

Perplexity AI функціонує як пошукова система, що використовує розмовний штучний інтелект. Він надає інформативні відповіді та пояснення у відповідь на запити користувачів, що може допомогти в пошуку та роз'ясненні. Хоча її результати можуть ненавмисно відкрити користувачам нову лексику і покращити розуміння, системі бракує навчальної підтримки. Вона не пропонує граматичних чи словникових вправ, інтерактивних діалогів для вивчення мови чи контенту, організованого відповідно до принципів вивчення мови. Хоча відповіді є інформативними, вони не структуровані таким чином, щоб сприяти систематичному розвитку мови.[4]

Хоча сучасні платформи демонструють значний технологічний прогрес у комунікації та створенні контенту на основі штучного інтелекту, їхня безпосередня освітня цінність у викладанні іноземних мов залишається предметом дискусій. Загальним недоліком більшості існуючих сервісів є відсутність спеціалізованого інтерфейсу, пристосованого до потреб викладачів - такого, який би підтримував створення індивідуальних навчальних завдань, дозволяв ефективно відстежувати прогрес студентів та забезпечував відповідність стандартам навчальної програми.

Враховуючи зростаючий обсяг адміністративних, методичних та комунікативних обов'язків, з якими стикаються вчителі, потреба в інтелектуальному асистенті стає не просто зручністю, а необхідністю. Педагогам часто не вистачає часу і ресурсів для персоналізації навчання, розробки додаткових матеріалів, оцінки результатів навчання або адаптації завдань на основі рівня володіння мовою чи когнітивного розвитку учнів. У цьому контексті інструмент на основі штучного інтелекту, спеціально розроблений для задоволення реальних повсякденних потреб викладачів іноземних мов, міг би стати суттєвою підтримкою.

Важливо визнати, що вчителі мають унікальний погляд на якість і доречність контенту, створеного штучним інтелектом, який не завжди збігається з тим, як його сприймають учні. У той час як учні можуть сприймати автоматизований зворотний зв'язок як критику, викладачі можуть оцінювати його через педагогічну призму, зосереджуючись на його навчальній цінності. Таким чином, підхід, орієнтований на викладача, сприяє створенню більш спільного навчального середовища, де студенти є не просто пасивними одержувачами завдань, згенерованих штучним інтелектом, а активними учасниками власного інтелектуального зростання.

Потреба в індивідуальних інструментах, розроблених спеціально для освітян, підкреслюється не лише відсутністю спеціалізованих рішень на ринку, але й мінливим характером викладання в цифровому контексті. Щоб підвищити ефективність педагогічної роботи, інструменти повинні відповідати реаліям сучасної освіти. Добре розроблена платформа, яка забезпечує доступність, масштабованість і конфігурованість, може слугувати стратегічним ресурсом, підтримуючи як нагальні навчальні потреби, так і довгострокові освітні цілі в середовищі, насиченому технологіями.

1.4 Постановка задачі

1.4.1 Мета, основні завдання проєкту та вимоги користувача

Цей проект має на меті розробити інтерактивного помічника для викладачів іноземних мов, який використовує інтелектуальні агенти для автоматизації створення навчальних завдань, таких як граматичні вправи, завдання на вивчення лексики та розмови на основі штучного інтелекту. Система спрямована на зменшення рутинного навантаження, підвищення різноманітності завдань та забезпечення гнучкості в адаптації вправ до конкретного рівня студентів та мовних тем.

Для досягнення цієї мети під час розробки системи необхідно виконати наступні загальні завдання:

- розробка зручного, інтуїтивно зрозумілого інтерфейсу, що не вимагатиме технічних знань від викладачів мов;
- розробка серверної інфраструктури, здатної обробляти запити, взаємодіяти з моделями ШІ через OLAMA та керувати постійним зберіганням даних;
- забезпечення безпечної реєстрації користувачів, автентифікації та управління сеансами;
- створення модульної структури системи для підтримки різних типів навчальних завдань (граматика, лексика, читання, говоріння);
- реалізація інтеграції з локальним хабом моделей штучного інтелекту для забезпечення динамічного вибору та використання різних моделей ШІ;
- створення схеми бази даних для зберігання згенерованих вправ та історії чату для кожного облікового запису користувача;
- розробка адаптивних елементів інтерфейсу користувача з сучасними анімаціями для підвищення зручності та доступності.

Конкретні завдання, пов'язані з цією веб-програмою, включають:

- уможливлення створення завдань у розділі граматики на основі вибраного типу завдання, моделі штучного інтелекту, рівня мови та теми граматики;
- уможливлення створення завдань зі словника за допомогою вибраного типу завдання, моделі штучного інтелекту, рівня мови та власного списку слів;
- уможливлення створення завдань з читання, дозволяючи користувачам завантажувати власні тексти для аналізу штучним інтелектом та створення завдань;
- уможливлення інтерактивної функції чату для практики мовлення, з вибором ролі та теми, а також опціями редагування, повторного створення та перезапуску розмови;

- забезпечення функціоналу для збереження, видалення та копіювання створених завдань у кожній категорії (граматика, лексика, читання);
- забезпечення оперативної взаємодії з моделями штучного інтелекту, враховуючи вибрані користувачем конфігурації;
- впровадження безпечного зберігання та пошуку всього створеного навчального контенту, пов'язаного з зареєстрованим користувачем;
- забезпечення послідовного та формального дизайну UX/UI, відповідного для навчальних цілей.

Викладачі мов, як основні користувачі системи, мають специфічні освітні та практичні потреби, які визначають функціональність інтерактивного помічника. Дизайн платформи повинен базуватися на цих потребах, щоб забезпечити її актуальність та зручність у використанні. Наступні вимоги користувачів визначають основні очікування від системи:

- різноманітність завдань та адаптивність: доступ до широкого спектру типів вправ (граматика, лексика, розуміння прочитаного та завдання на говоріння) з можливістю адаптації до різних навчальних цілей та профілів учнів. Важливо мати можливість вибору з попередньо визначених типів завдань;
- контент, адаптований до рівня: відповідність вправ обраним рівням володіння англійською мовою (наприклад, A1–C2) для підтримки диференційованого навчання. Система повинна автоматично фільтрувати граматичні теми та складність лексики на основі обраного рівня;
- вибір моделі ШІ: можливість вибору моделі ШІ, яка найкраще відповідає педагогічним цілям, оскільки різні моделі можуть створювати контент різної якості або стилю;
- опції налаштування: можливість вводити конкретні дані, такі як список слів або текст для розуміння прочитаного, для подальшої генерації вправ на основі введених даних;

- ефективність генерації контенту: мінімізація рутини ручного створення вправ шляхом автоматизації процесу, зберігаючи при цьому педагогічну правильність;
- інтерактивна комунікація: у завданнях з говоріння можливість визначати ролі (наприклад, вчитель-учень, інтерв'юер-кандидат) та ініціювати інтерактивні діалоги ШІ, адаптовані до конкретних тем розмови;
- управління завданнями: можливість зберігати створені завдання в особистих облікових записах, переглядати їх пізніше, копіювати для повторного використання або видаляти, якщо вони більше не потрібні;
- редагований вихідний текст ШІ: можливість редагувати або повторно генерувати відповіді, створені ШІ, в інтерфейсі чату для більшого контролю;
- зручний інтерфейс: інтуїтивно зрозумілий дизайн з чіткою навігацією, візуально привабливими елементами та мінімальною кривою навчання, з огляду на те, що багато вчителів можуть не мати технічної підготовки.

1.4.2. Функціональні та нефункціональні вимоги до програмного забезпечення

Функціональні вимоги визначають основні можливості та функції, які система повинна надавати для задоволення потреб користувачів та забезпечення ефективного виконання завдань. Ці вимоги визначають очікувані взаємодії між користувачами та компонентами системи.

Рис. 1.1 – Use Case діаграма функціональних можливостей

1. Аутентифікація користувачів та управління обліковими записами. Система повинна підтримувати процеси реєстрації та аутентифікації користувачів для створення та управління особистими обліковими записами. Це дозволяє персоналізувати доступ до функцій системи та зберігати дані користувачів, такі як збережені вправи та історія чатів.

2. Модуль граматики:

- система повинна дозволяти користувачам вибирати з різних типів граматичних вправ (наприклад, заповнення пропусків, вибір з декількох варіантів, перетворення речень);

- вона повинна надавати інтерфейс для вибору моделі штучного інтелекту, відповідальної за генерацію вправ, що забезпечує гнучкість у стратегіях генерації завдань;
- користувачі повинні мати можливість вибирати рівень володіння англійською мовою (наприклад, початковий, середній, просунутий), що фільтрує відповідні граматичні теми;
- на основі обраного рівня система повинна представити список граматичних тем для вибору (наприклад, часи, умовні речення, модальні дієслова);
- система повинна динамічно генерувати граматичні вправи за допомогою обраної моделі штучного інтелекту, адаптованої до параметрів, обраних користувачем.

3. Модуль лексики та словника:

- система повинна дозволяти вибір різних типів вправ на лексику (наприклад, підбір слів, заповнення пропусків, визначення синонімів/антонімів);
- користувачі повинні вибрати модель штучного інтелекту та рівень володіння англійською мовою, щоб визначити складність і стиль вправ;
- система повинна дозволяти користувачам вводити або вибирати власний список слів, які будуть включені в генеровані завдання;
- вправи на лексику, згенеровані штучним інтелектом, повинні відповідати вибраним словам, рівню та типу вправи, що сприяє цілеспрямованій практиці лексики.

4. Модуль розуміння прочитаного:

- система повинна надавати інтерфейс, за допомогою якого користувачі можуть завантажувати або вводити тексти для аналізу;
- користувачі повинні мати можливість вказати бажаний тип завдання на читання (наприклад, питання на розуміння, написання резюме, лексика в контексті);

- система повинна аналізувати наданий текст за допомогою штучного інтелекту та відповідно генерувати відповідні вправи;
- результат повинен бути налаштований відповідно до уподобань та рівня користувача.

5. Модуль говоріння (чат з ШІ):

- система повинна дозволяти користувачам вибирати ролі для себе та співрозмовника ШІ (наприклад, вчитель-учень, інтерв'юер-інтерв'юйований);
- користувачі повинні вибирати модель ШІ, рівень володіння англійською мовою та тему розмови, щоб налаштувати досвід чату;
- система повинна сприяти взаємодії з ШІ в режимі реального часу, імітуючи усний діалог;
- вона повинна дозволяти користувачам редагувати відповіді ШІ, відтворювати їх або перезапускати розмови для поліпшення навчального досвіду.

6. Управління вправами та чатом:

- система повинна зберігати згенеровані вправи з граматики, лексики та читання в обліковому записі користувача, що дозволить переглядати та повторно використовувати їх у майбутньому;
- користувачі повинні мати можливість зручно переглядати, видаляти або копіювати збережені вправи;
- що стосується чатів, користувачі повинні мати можливість редагувати, відтворювати або перезапускати розмови, покращуючи персоналізацію та контроль.

7. Інтеграція з AI Backend:

- система повинна підключатися до служби AI Backend для динамічного надсилання запитів та отримання згенерованого контенту;
- ця інтеграція повинна підтримувати кілька моделей AI та типів завдань, абстрагуючи складність від користувача.

8. Інтерфейс користувача та навігація:

- система повинна забезпечувати чіткі шляхи навігації для безперешкодного доступу до всіх модулів (граматика, лексика, читання, говоріння);
- вибір користувача в одному кроці повинен динамічно впливати на доступні опції в наступних кроках (наприклад, вибір фільтрів рівня граматичних тем);
- інтерфейс повинен супроводжувати користувачів під час створення завдань за допомогою покрокових робочих процесів, щоб мінімізувати плутанину та помилки.

Нефункціональні вимоги визначають, як система повинна працювати, та визначають характеристики якості:

- зручність використання: система повинна мати інтуїтивно зрозумілий та зручний інтерфейс, придатний для викладачів мов, з чіткою навігацією та послідовним макетом;
- продуктивність: система повинна забезпечувати швидкий час відгуку на дії користувача та процеси генерації ШІ;
- надійність: Система повинна працювати стабільно і бути доступною під час використання;
- безпека: Система повинна забезпечувати базові заходи безпеки для облікових записів користувачів, включаючи безпечну обробку паролів та безпечну комунікацію з базою даних;
- портативність: Система повинна бути доступною з сучасних настільних браузерів та адаптуватися до різних роздільних здатностей екранів.

1.4.3 Оцінка ризиків і визначення пріоритетності завдань

При розробці програмного забезпечення, особливо складних систем, що включають моделі штучного інтелекту та дані користувачів, дуже важливо

визначити потенційні ризики, які можуть вплинути на успіх проекту. Основні ризики, пов'язані з цим проектом, включають:

- високі вимоги до апаратного забезпечення сервера для локального зберігання моделей штучного інтелекту: моделі штучного інтелекту, особливо великі мовні моделі або мультиагентні системи, вимагають значних обчислювальних ресурсів і пам'яті. Розгортання цих моделей на локальному рівні вимагає потужного серверного обладнання, що може збільшити витрати і ускладнити управління інфраструктурою;
- складність інтеграції декількох моделей штучного інтелекту в єдину систему: об'єднання різних моделей штучного інтелекту з різною архітектурою та інтерфейсами в єдину платформу створює проблеми з інтеграцією. Забезпечення послідовної комунікації, потоку даних та синхронізації між моделями вимагає ретельного проектування і може призвести до появи помилок або проблем з продуктивністю;
- ризик недостатньої якості завдань, генерованих штучним інтелектом: якість та релевантність завдань, генерованих ШІ, залежать від точності моделі та даних для навчання. Існує ризик, що генеровані вправи можуть не відповідати педагогічним стандартам або не відповідати потребам користувачів, що знижує корисність системи для освітян;
- ризик порушення конфіденційності даних користувачів: обробка даних користувачів, включаючи інформацію для входу та персоналізовані дані завдань, створює потенційні ризики для безпеки та конфіденційності.
- складність підтримки багатокористувацького середовища: навіть без детального розмежування ролей управління одночасними користувачами, збереження сеансів та послідовна обробка даних між обліковими записами додають складності системі. Це може вплинути на надійність та користувацький досвід.

Встановлення пріоритетності функціональних вимог є надзвичайно важливим для того, щоб зосередити зусилля на найважливіших функціях,

забезпечуючи реалістичність проекту та задоволення користувачів. (Див. Табл. 1.2).

Таблиця 1.2 — Ризики розробки вебсайту

Рівень пріоритету	Функціональна вимога	Опис
Високий	Реєстрація та вхід користувача	Безпечна та проста автентифікація для надання доступу користувачам та збереження персональних даних
Високий	Генерація граматичних вправ	Вибір типу завдання, моделі ШІ, рівня англійської мови та граматичної теми з генерацією вправ
Високий	Генерація завдань на лексику та словниковий запас	Можливість вибору типу завдання, моделі ШІ, рівня англійської мови та введення списку слів для генерації завдань
Високий	Розділ читання із завантаженням тексту та завданнями, згенерованими ШІ	Завантаження текстів, аналіз ШІ та генерація вправ на основі вибору вчителя
Високий	Розмовний чат на основі ШІ	Інтерфейс чату з можливістю вибору ролі ШІ, моделі, рівня мови та теми розмови
Середній	Збереження, редагування та видалення згенерованих завдань	Дозволяє користувачам керувати згенерованими вправами в межах свого облікового запису
Середній	Редагування та регенерація відповідей чату ШІ	Функції для зміни, регенерації або перезапуску розмов із ШІ
Низький	Дизайн інтерфейсу користувача з сучасними анімаціями	Покращення взаємодії з користувачем за допомогою інтуїтивно зрозумілого та візуально привабливого інтерфейсу

Висновки до розділу:

– Проаналізовано проблеми, з якими стикаються викладачі іноземних мов при створенні персоналізованих завдань, а також можливості їх вирішення за допомогою автоматизації та штучного інтелекту.

– Проведено огляд і порівняльний аналіз існуючих програмних засобів, що застосовують AI у навчанні, визначено їхні переваги та обмеження.

– Сформульовано мету й завдання дипломного проєкту, визначено функціональні вимоги, а також оцінено ризики та пріоритети для ефективної реалізації інтерактивного помічника.

РОЗДІЛ 2
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО ПОМІЧНИКА ВИКЛАДАЧА ІНОЗЕМНОЇ МОВИ З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ

2.1 Вибір технологій та обґрунтування їх застосування при розробці інформаційної системи

2.1.1 Обґрунтування вибору React для реалізації клієнтської частини

При розробці інтерактивного помічника для викладачів іноземних мов, фронтенд-частина додатку відіграє вирішальну роль у забезпеченні зручності використання, швидкості реагування та ефективної взаємодії між кінцевим користувачем та інструментами штучного інтелекту. З цією метою бібліотека React була обрана як основна технологія для побудови клієнтської частини веб-додатку.

React - це сучасна бібліотека JavaScript з відкритим вихідним кодом, розроблена і підтримувана компанією Facebook, призначена для створення динамічних і високопродуктивних користувацьких інтерфейсів. React використовує компонентну архітектуру, що дозволяє створювати модульний, багаторазовий код, який легко тестується. У контексті цього проекту React відповідає за рендеринг інтерактивного інтерфейсу для вибору завдань, моделей, рівнів англійської, граматики чи словникових тем, завантаження текстів для читання, керування чатами з агентом ШІ та автентифікацію через спеціальні сторінки для входу та реєстрації.[6]

React буде взаємодвяти безпосередньо з бекендом Flask за допомогою HTTP-запитів (використовуючи axios), щоб отримувати граматичні теми на основі вибраних рівнів англійської мови, генерувати вправи з використанням різних моделей ШІ.

Рішення використовувати React обґрунтовано наступними факторами:

Компонентний дизайн: React дозволяє інкапсулювати елементи інтерфейсу як компоненти для повторного використання. Наприклад, випадаючі списки для моделей ШІ, граматичних тем та рівнів англійської мови реалізовані як окремі компоненти, які можна повторно використовувати на різних сторінках завдань (Граматика, Лексика, Читання).

Управління станом та життєвим циклом: Використання хуків React, таких як useState та useEffect, забезпечує ефективний контроль над станами компонентів та побічними ефектами (наприклад, отримання тем при виборі рівня).

Маршрутизація: Інтеграція react-router-dom уможливлює поведінку односторінкового додатку (SPA), забезпечуючи плавні та швидкі переходи між такими розділами, як Граматика, Словник, Читання, Про програму та Чат, без повного перезавантаження сторінки.

Анімація та динаміка інтерфейсу: Для покращення користувацького досвіду містить інтегровану бібліотеку framer-motion. Вона дозволяє реалізувати сучасну, плавну анімацію, яка сприяє інтуїтивно зрозумілій та візуально привабливій взаємодії, що відповідає меті проектування - зробити систему приємною для вчителів.

Комунікація з бекендом: Бібліотека axios використовується для створення HTTP-запитів до сервера Flask. Це спрощує виклики API для надсилання та отримання даних, таких як конфігурації вправ, облікові дані користувачів та контент, згенерований штучним інтелектом.

Веб-сайт буде складатися із високорівневих компонентів (сторінок), кожен з яких представляє функціональну частину навчального асистента. (Див. Рис. 2.1).

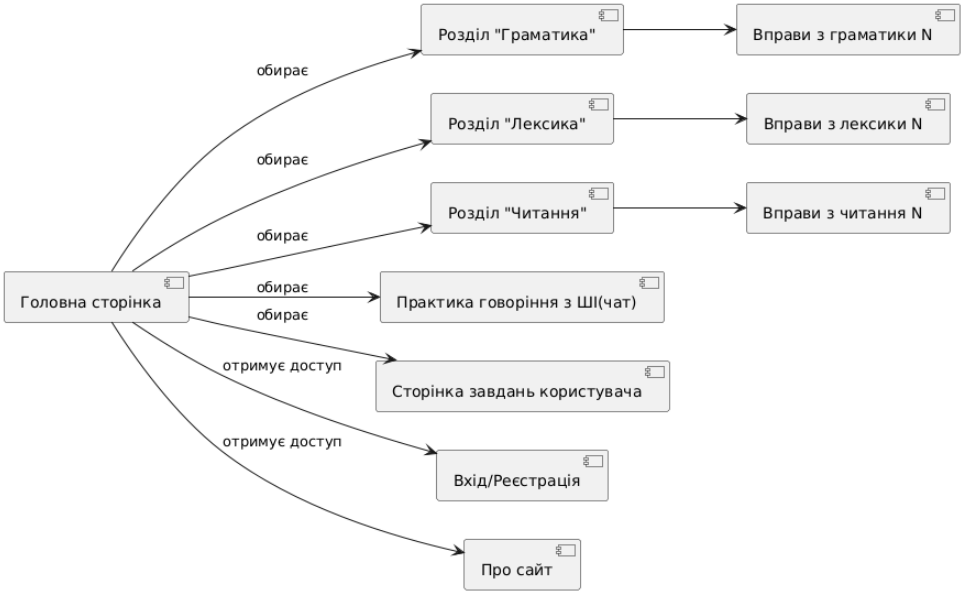


Рисунок 2.1 — Діаграма структури вебсайту

1. Головна сторінка: початкова сторінка, яка представляється користувачеві. Вона надає доступ до всіх основних розділів веб-сайту, включаючи меню завдань, дії в обліковому записі користувача та загальну інформацію про проект.

2. Розділ граматики. Цей розділ доступний через пункт меню «Граматика». Він містить:

- селектор завдань з граматики: інтерфейс, який дозволяє вчителям вибрати потрібний тип граматичної вправи;
- сторінка граматичної вправи: після вибору система генерує та відображає відповідне граматичне завдання, використовуючи моделі ШІ.

3. Розділ лексики. Цей розділ функціонує аналогічно до розділу граматики:

- вибір словникового завдання: викладач обирає тип словникової вправи та вводить список слів;
- сторінка словникової вправи: відображає згенероване штучним інтелектом лексичне завдання.

4. Розділ читання. Цей розділ включає в себе:

- селектор завдань для читання: вчителі можуть завантажувати власні тексти та обирати тип завдання на розуміння прочитаного;
- сторінка вправи для читання: відображає створену вправу для читання.

5. Сторінка чату: присвячена розмові зі штучним інтелектом. Користувач обирає модель ШІ, ролі (вчитель / учень тощо), рівень англійської мови та тему. Відкривається інтерактивний чат у реальному часі.

6. Сторінка завдань користувача: персоналізована сторінка для аутентифікованих користувачів, де зберігаються всі збережені вправи (з граматики, лексики, читання та говоріння). Користувачі можуть переглядати, видаляти або копіювати попередні завдання, створені ШІ.

7. Сторінки входу та реєстрації: сторінки, які забезпечують функціональність доступу до облікового запису. Сюди входить автентифікація користувача, керування сеансами та створення облікового запису.

8. Сторінка «Про сайт»: ця сторінка представляє мету сайту та пояснює, як технології штучного інтелекту інтегровані для допомоги вчителям.

2.1.2 Обгрунтування вибору Python для реалізаціїї бекенду

Внутрішня частина системи розроблена на мові Python з використанням веб-фреймворку Flask. Python було обрано з кількох обґрунтованих причин:

Потужна підтримка інтеграції ШІ: Python є стандартом де-факто в розробці штучного інтелекту та машинного навчання завдяки широкій екосистемі бібліотек та фреймворків (наприклад, PyTorch, TensorFlow, HuggingFace Transformers). Оскільки проект інтегрується з локальними LLM через Ollama, Python забезпечує пряму сумісність і спрощений контроль над виконанням моделі та обробкою запитів до API.[14]

Швидка розробка з Flask: Flask — це легкий, але потужний мікрофреймворк для Python, який дозволяє швидко розробляти веб-API. Він

підтримує модульну архітектуру, управління маршрутами та розширення проміжного програмного забезпечення, що має вирішальне значення для розділення таких проблем, як автентифікація, доступ до даних та взаємодія з LLM.

Простота інтеграції з MSSQL та зовнішніми сервісами: Python підтримує ORM-інструменти, такі як SQLAlchemy, та бібліотеки, такі як pyodbc, що забезпечує безперешкодну інтеграцію з Microsoft SQL Server, який слугує основною базою даних системи. Ця сумісність забезпечує надійні CRUD-операції над даними користувача, згенерованими вправами та журналами чату.

Зручність у супроводі та читабельність: Чистий синтаксис Python та потужна підтримка спільноти роблять бекенд дуже зручним для підтримки. Оскільки додаток призначений для використання та розширення в освітньому середовищі, використання Python гарантує майбутню адаптивність.

Локальна взаємодія зі штучним інтелектом через Ollama: Ollama надає зручний інтерфейс для локального запуску LLM. Використовуючи Python, сервер може ефективно надсилати підказки моделям, розміщеним на Ollama (DeepSeek v1, DeepSeek v3, MISTRA та LAMA v3.1), отримувати відповіді та реєструвати результати для подальшого використання.

Сервер Flask буде складатися з трьох основних компонентів: Кінцеві точки API, взаємодія з базою даних та зв'язок з моделлю ШІ. (Див. Рис. 2.2).

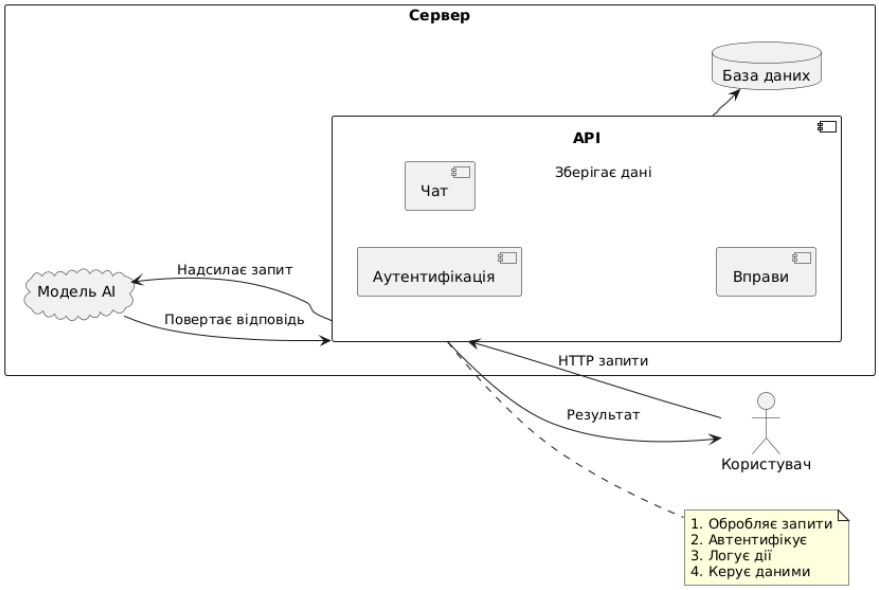


Рисунок 2.2 — Діаграма структури роботи серверу

Автентифікація користувачів: система включає в себе механізм автентифікації, який керує контролем доступу. Нові користувачі можуть створювати облікові записи, а існуючі користувачі отримують доступ до своїх профілів після перевірки облікових даних. Після успішної автентифікації створюються сеанси користувача, які зберігаються протягом всієї взаємодії з системою, що забезпечує безпечну та персоналізовану функціональність.

Робочий процес генерації завдань: користувачі можуть запитувати створення навчального контенту, адаптованого до їхніх потреб. Це можуть бути завдання з граматики, лексики чи читання. Система обробляє ці запити, збираючи необхідні параметри, такі як бажаний тип завдання, рівень володіння мовою та, за необхідності, відповідні тематичні або лексичні дані.

Контекстний пошук даних: для підтримки персоналізації контенту система отримує структуровані лінгвістичні дані з внутрішніх репозиторіїв. Рівні володіння мовою надаються як орієнтири, а теми динамічно фільтруються на основі обраного рівня.

Розмовний інтерфейс: платформа включає інтерактивний розмовний інструмент, призначений для полегшення практики усного мовлення. Користувачі можуть вести діалог з агентом на основі штучного інтелекту, гнучко визначаючи ролі та лінгвістичні параметри. Вхідні дані користувача обробляються, інтерпретуються і перенаправляються до обраної мовної моделі, яка повертає зв'язну і контекстуально відповідну відповідь.

Інтеграція та посередництво ШІ: основною особливістю системи є її інтеграція з різними локальними великими мовними моделями (LLM). Серверний компонент виконує роль посередника, форматуючи запити та інтерпретуючи відповіді в однаковий спосіб незалежно від обраного рушія ШІ. Цей рівень абстракції забезпечує модульність, спрощує комунікацію та полегшує інтеграцію декількох моделей ШІ з різними можливостями та стратегіями оптимізації.

2.1.3 Аргументація використання SQL і Ollama для розширення функціональності системи

У розробці інтерактивного асистента для вчителів іноземних мов база даних відіграє вирішальну роль у забезпеченні безпечного, структурованого та ефективного зберігання даних, пов'язаних з користувачами, вправами, історією чату, конфігураціями штучного інтелекту та лінгвістичними метаданими (наприклад, граматичними рівнями та темами). Після ретельної оцінки доступних реляційних систем управління базами даних (СКБД) було обрано Microsoft SQL Server (MSSQL) як основну технологію баз даних для цього проекту.

Обґрунтування вибору Microsoft SQL Server:

Реляційне моделювання даних: додаток оперує добре структурованими сутностями даних, такими як користувачі, сеанси, шаблони вправ, граматичні рівні, теми та взаємодії зі штучним інтелектом. MSSQL забезпечує надійну підтримку реляційного моделювання даних за допомогою таблиць, первинних/зовнішніх ключів та індексування. Ці функції полегшують чітке визначення зв'язків між рівнями англійської мови та відповідними граматичними темами, забезпечуючи цілісність та узгодженість даних.

Розширені можливості запитів: MSSQL підтримує складні SQL-запити, збережені процедури та функції, які необхідні для динамічного пошуку навчального контенту, пов'язаного з мовою. Наприклад, система знаходить граматичні теми на основі обраного рівня англійської мови за допомогою оптимізованих операцій JOIN, що забезпечує високу продуктивність навіть при роботі з великими наборами даних.[15]

Інтеграція з Python та Flask: бекенд системи буде реалізовано за допомогою Python з Flask, який має розвинену підтримку MSSQL через такі бібліотеки, як pyodbc та SQLAlchemy. Ця безшовна інтеграція забезпечує безпечний зв'язок між сервером додатків і рівнем бази даних, що дозволяє

здійснювати автентифікацію користувачів, управління сеансами і зберігати дані.

Безпека і рольовий доступ: MSSQL забезпечує автентифікацію, шифрування та управління дозволами корпоративного рівня, що має вирішальне значення для захисту конфіденційних даних, таких як облікові дані для входу та історія вправ. Це особливо важливо в освітньому середовищі, де конфіденційність користувачів і контроль даних мають першорядне значення.

Масштабованість і надійність: у міру того, як додаток зростає і потенційно масштабується до декількох користувачів (наприклад, між установами), MSSQL пропонує можливість ефективно справлятися зі зростаючими навантаженнями. Вбудовані інструменти для резервного копіювання, реплікації та моніторингу продуктивності забезпечують високу доступність і відмовостійкість.

Сумісність із середовищем розробки: проект розробляється з використанням Visual Studio Code. MSSQL інтегрується з цим інструментом, полегшуючи проектування схем, виконання запитів та міграцію баз даних в єдиному середовищі розробки.

Підтримка збережених вправ і журналів: однією з найважливіших функціональних можливостей програми є можливість для користувачів зберігати згенеровані ШІ граматику, словниковий запас і вправи на читання. MSSQL підтримує транзакційні операції, що дозволяє надійно зберігати, знаходити та видаляти ці завдання, а також реєструвати взаємодію користувача з моделями ШІ.

Для забезпечення функціональності на основі штучного інтелекту, як-от створення граматичних вправ, завдань на поповнення словникового запасу, вправ на розуміння прочитаного та розмовних чат-ботів, система покладається на локальний фреймворк для роботи з моделями штучного інтелекту. Для цього у проєкту буде використано Ollama - сучасну та легку платформу для локального запуску великих мовних моделей (LLM). (Див. Рис. 2.3).

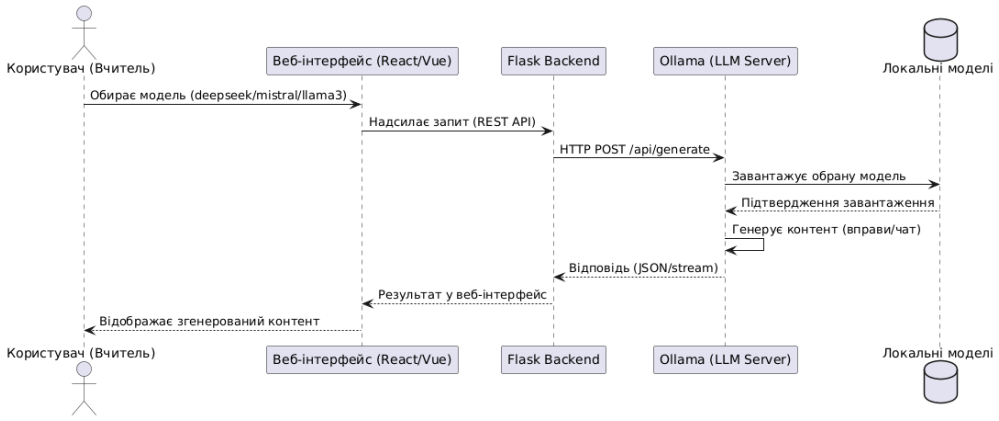


Рисунок 2.3 — Діаграма функціонування програми Ollama та її взаємодія із серверною інфраструктурою

Ollama працює як локальний сервер виконання, який дозволяє розробникам завантажувати та запускати різні ЛММ з відкритим вихідним кодом або створені на замовлення за допомогою простого API-інтерфейсу. Додаток взаємодіє з сервером Ollama за допомогою HTTP-запитів, зазвичай через REST API. Після завантаження моделі (наприклад, DeepSeek-v1, DeepSeek-v3, MISTRA або LAMA-3.1) бекенд надсилає запити до Ollama і отримує структуровані або текстові відповіді у довільній формі.[11]

Базова архітектура цієї взаємодії включає:

- вибір і завантаження моделі;
- оперативне надсилання промпту;
- потокове або стандартне отримання відповіді;
- перемикання моделі без перезапуску сервісу.

Такий підхід дозволяє інтегрувати в систему можливості генерації контенту на основі ШІ, не покладаючись на сторонні хмарні сервіси, зберігаючи повний контроль над даними та поведінкою відповідей.

Обґрунтування використання Ollama:

- локальне розгортання та конфіденційність даних: на відміну від хмарних сервісів штучного інтелекту, ollama працює повністю на локальному комп'ютері або сервері, гарантуючи, що конфіденційні дані користувача - такі як дані учнів, прочитані тексти або створений контент - ніколи не залишають

локальне середовище. Це важливо для дотримання стандартів конфіденційності даних в освітньому контексті;

- гнучкість і розширюваність моделі: ollama підтримує ряд моделей (таких як deepseek, mistra, lama), які можна динамічно перемикати. Це особливо корисно для цієї системи, де користувач (викладач) обирає потрібну модель на основі типу завдання та педагогічної мети;
- низька затримка та можливість роботи в автономному режимі: оскільки моделі розміщуються локально, час відгуку значно скорочується, і система може функціонувати навіть без стабільного інтернет-з'єднання. Це покращує доступність функціональності ШІ під час уроків або тренінгів;
- спрощена інтеграція з бекендом Flask: сервер Flask безпосередньо взаємодіє з Ollama за допомогою HTTP-запитів. Це усуває потребу в складних SDK або пропрієтарних API, скорочуючи час розробки і роблячи архітектуру системи більш зручною для підтримки;
- відкритий вихідний код та економічна ефективність: ollama є безкоштовною у використанні та має відкритий вихідний код, що робить її ідеальним рішенням для академічних проектів, пілотних впроваджень та установ з обмеженим бюджетом. Це усуває постійні витрати, пов'язані з комерційними API;
- підтримка потокового мовлення та тонко налаштованих підказок: платформа підтримує швидке налаштування та потокові відповіді, які корисні для реалізації функцій чату в реальному часі в розділі «Говоріння» додатку.

2.2. Проєктування структури бази даних

2.2.1. Розробка логічної моделі бази даних

База даних системи Engflow розроблена для підтримки основних функцій платформи, включаючи автентифікацію користувачів, зберігання та пошук згенерованих штучним інтелектом вправ, історію взаємодії в чаті, управління

тикетами підтримки та категоризацію рівнів володіння англійською мовою. База даних реалізована за допомогою Microsoft SQL Server і включає кілька взаємопов'язаних таблиць для забезпечення узгодженості даних, масштабованості та оптимізованого доступу. (Див. Рис. 2.4).

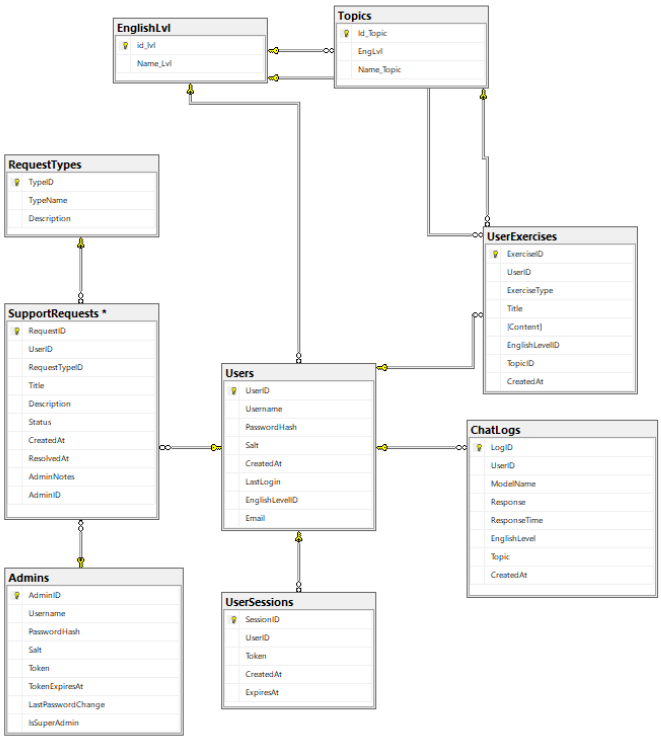


Рисунок 2.4 — Структура бази даних

Таблиця Admins зберігає дані про адміністративних користувачів, які керують платформою. Кожен адміністратор ідентифікується унікальним ідентифікатором AdminID. Таблиця містить поля для безпечного зберігання облікових даних (Username, PasswordHash та Salt), інформації, пов'язаної з сесією (Token, TokenExpiresAt), а також метаданих, таких як LastPasswordChange та булевий прапор IsSuperAdmin, що позначає підвищені привілеї. Ця таблиця є критично важливою для безпечного та ієрархічного адміністрування підтримки користувачів та модерації платформи. (Див. Табл. 2.1)

Таблиця 2.1 — Таблиця «Admins»

Назва стовпця	Тип даних	Призначення
AdminID	int	Унікальний ідентифікатор адміністратора
Username	nvarchar(50)	Ім'я користувача адміністратора
PasswordHash	nvarchar(255)	Хеш паролю адміністратора
Salt	nvarchar(255)	Сіль для хешування паролю
Token	nvarchar(255)	Токен автентифікації (JWT або подібний)
TokenExpiresAt	datetime	Час закінчення дії токена
LastPasswordChange	datetime	Дата останньої зміни паролю
IsSuperAdmin	bit	Ознака, чи є адміністратор суперкористувачем

Таблиця Users зберігає інформацію про зареєстрованих користувачів платформи, переважно викладачів. Кожен користувач ідентифікується унікальним ідентифікатором UserID. Таблиця містить облікові дані (Username, Email, PasswordHash та Salt), позначку часу створення облікового запису (CreatedAt), дату останнього входу (LastLogin) та зовнішній ключ EnglishLevelID, що посилається на таблицю EnglishLvl. Така структура підтримує аутентифікацію користувачів та персоналізацію завдань, згенерованих штучним інтелектом, відповідно до обраного користувачем рівня володіння англійською мовою. (Див. Табл. 2.2)

Таблиця 2.2 — Таблиця «Users»

Назва стовпця	Тип даних	Призначення
UserID	int	Унікальний ідентифікатор користувача
Username	nvarchar(50)	Ім'я користувача
Email	nvarchar(100)	Електронна пошта
PasswordHash	nvarchar(255)	Хеш паролю
Salt	nvarchar(255)	Сіль для хешування
CreatedAt	datetime	Дата створення облікового запису
LastLogin	datetime	Дата останнього входу
EnglishLevelID	int	Рівень англійської (посилання на EnglishLvl)

Таблиця UserSessions керує активними сесіями користувачів для аутентифікації та безпеки на основі токенів. Кожна сесія ідентифікується ідентифікатором SessionID і пов'язана з конкретним користувачем за допомогою зовнішнього ключа UserID. Вона включає Token сесії, позначку часу створення CreatedAt та поле ExpiresAt для забезпечення безпечного доступу з обмеженим часом. Ця таблиця відіграє ключову роль у забезпеченні механізмів аутентифікації без збереження стану між фронтендом та бекендом. (Див. Табл. 2.3)

Таблиця 2.3 — Таблиця «UserSessions»

Назва стовпця	Тип даних	Призначення
SessionID	int	Унікальний ідентифікатор сесії
UserID	int	Посилання на користувача
Token	nvarchar(255)	Токен для авторизації сесії
CreatedAt	datetime	Час створення сесії
ExpiresAt	datetime	Час завершення дії сесії

Таблиця EnglishLvl визначає стандартизовані рівні володіння англійською мовою (наприклад, A1, A2, B1 тощо). Кожен рівень має унікальний ідентифікатор id_lvl та назву (Name_Lvl). Ця система класифікації використовується в кількох таблицях (таких як Users, Topics та UserExercises) для забезпечення того, щоб згенеровані завдання та відповіді чату були адаптовані до відповідної лінгвістичної складності. (Див. Табл. 2.4)

Таблиця 2.4 — Таблиця «EnglishLvl»

Назва стовпця	Тип даних	Призначення
id_lvl	int	Унікальний ідентифікатор рівня
Name_Lvl	nvarchar(50)	Назва рівня

Таблиця Topics зберігає теми граматики або лексики, що відповідають різним рівням англійської мови. Кожна тема має унікальний ідентифікатор Id_Topic, зовнішній ключ EngLvl, що посилається на таблицю EnglishLvl, та описову назву Name_Topic. Ця структура забезпечує фільтрацію пропозицій тем, що відображаються користувачеві, на основі обраного рівня володіння англійською мовою. (Див. Табл. 2.5)

Таблиця 2.5 — Таблиця «Topics»

Назва стовпця	Тип даних	Призначення
Id_Topic	int	Унікальний ідентифікатор теми
EngLvl	int	Посилання на рівень англійської (EnglishLvl)
Name_Topic	varchar(-1)	Назва теми граматики чи лексики

Таблиця UserExercises зберігає вправи, згенеровані штучним інтелектом для конкретних користувачів. Кожна вправа має унікальний ідентифікатор ExerciseID, пов'язана з користувачем (UserID), та містить метадані, такі як ExerciseType, Title, Content, пов'язаний рівень англійської мови (EnglishLevelID) та тему (TopicID). Поле CreatedAt зберігає позначку часу створення. Ця таблиця дозволяє користувачам зберігати, переглядати та керувати завданнями, згенерованими штучним інтелектом. (Див. Табл. 2.6)

Таблиця 2.6 — Таблиця «UserExercises»

Назва стовпця	Тип даних	Призначення
ExerciseID	int	Унікальний ідентифікатор вправи
UserID	int	Посилання на користувача
ExerciseType	nvarchar(100)	Тип вправи (граматика, лексика тощо)
Title	nvarchar(255)	Назва вправи
Content	nvarchar(-1)	Зміст вправи, згенерований ШІ
EnglishLevelID	int	Рівень англійської мови
TopicID	int	Тема, до якої належить вправа
CreatedAt	datetime	Дата створення вправи

Таблиця ChatLogs записує взаємодії між користувачами та агентом штучного інтелекту в розмовному (чатовому) модулі платформи. Кожен запис має унікальний ідентифікатор LogID, посилання на користувача (UserID), та записує назву моделі штучного інтелекту ModelName, згенеровану відповідь Response, час відповіді ResponseTime, рівень англійської мови EnglishLevel, тему Topic та позначку часу CreatedAt. Ця таблиця підтримує аналіз продуктивності, налагодження штучного інтелекту та відстеження зворотного зв'язку. (Див. Табл. 2.7).

Таблиця 2.7 — Таблиця «ChatLogs»

Назва стовпця	Тип даних	Призначення
LogID	int	Унікальний ідентифікатор чату
UserID	int	Посилання на користувача
ModelName	nvarchar(50)	Назва моделі ШІ (наприклад, DeepSeek, LAMA)
Response	nvarchar(-1)	Відповідь, згенерована ШІ
ResponseTime	float	Час відповіді в секундах
EnglishLevel	nvarchar(50)	Рівень англійської
Topic	nvarchar(100)	Тема розмови
CreatedAt	datetime	Дата створення запису

Таблиця RequestTypes зберігає попередньо визначені типи запитів до служби підтримки. Кожен тип запиту має унікальний ідентифікатор TypeID, назву TypeName та необов'язковий опис Description. Вона забезпечує класифікацію технічних або адміністративних проблем, порушених користувачами, сприяючи ефективному сортуванню запитів до служби підтримки. (Див. Табл.. 2.8)

Таблиця 2.8 — Таблиця «RequestTypes»

Назва стовпця	Тип даних	Призначення
TypeID	int	Унікальний ідентифікатор типу
TypeName	nvarchar(50)	Назва типу запиту
Description	nvarchar(255)	Опис типу запиту

Таблиця SupportRequests реєструє запити до служби підтримки, подані користувачами. Кожен запит має унікальний ідентифікатор RequestID, пов'язаний з UserID та класифікований за допомогою RequestTypeID. Запит включає Title, Description, поточний Status (за замовчуванням 'Pending'), позначки часу створення та вирішення (CreatedAt, ResolvedAt), адміністративні примітки (AdminNotes) та посилання на призначеного адміністратора (AdminID). Ця структура підтримує всебічну та відстежувану комунікацію між користувачами та адміністраторами. (Див. Табл. 2.9)

Таблиця 2.9 — Таблиця «SupportRequests»

Назва стовпця	Тип даних	Призначення
RequestID	int	Унікальний ідентифікатор звернення
UserID	int	Посилання на користувача
RequestTypeID	int	Тип запиту (посилання на RequestTypes)
Title	nvarchar(100)	Заголовок запиту
Description	nvarchar(-1)	Повний опис проблеми
Status	nvarchar(20)	Статус запиту (наприклад, "Pending")
CreatedAt	datetime	Дата створення запиту
ResolvedAt	datetime	Дата вирішення (якщо є)
AdminNotes	nvarchar(-1)	Примітки адміністратора
AdminID	int	Адміністратор, який обробив запит

2.2.2 Первинне заповнення таблиць даними

Для забезпечення ефективного функціонування системи та можливості надання персоналізованої підтримки користувачам, має бути розроблено

кілька ключових таблиць для зберігання відповідних даних. Ці таблиці слугують основою для управління взаємодією з користувачами, відстеження прогресу в навчанні та надання адміністративної підтримки.

Таблиця EnglishLvl містить попередньо визначені рівні володіння англійською мовою (див. Рис. 2.5). Для викладача ця категоризація дозволяє узгодити вправи та теми з реальними можливостями студента. Це забезпечує відповідну складність змісту та підтримує адаптивне навчання на основі стандартів CEFR.

	id_lvl	Name_Lvl
1	1112	Elementary
2	1113	Pre-intermediate
3	1114	Intermediate
4	1115	Upper-intermediate
5	1116	Pre-advanced

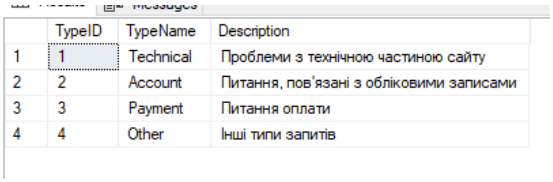
Рисунок 2.5 — Заповнені дані таблиці EnglishLvl

У таблиці Topics зберігаються граматичні та лексичні теми, пов'язані з кожним рівнем англійської мови (див. Рис. 2.6). Це допомагає викладачам спрямовувати студентів через структуроване просування та обирати теми, які відповідають рівню та контекстуально релевантні.

	Id_Topic	EngLvl	Name_Topic
162	111611	1116	Wish, rather, if only, it's time – unreal uses of pa...
163	111612	1116	Unless, even if, provided, as long as, etc. – oth...
164	111613	1116	Distancing – expressions and passive of reportin...
165	111614	1116	Passive verbs with two objects
166	111615	1116	Verb + object + infinitive/gerund – verb patterns
167	111616	1116	Gerunds and infinitives – complex forms
168	111617	1116	Reflexive and reciprocal pronouns
169	111618	1116	Generic pronouns – common-gender pronouns
170	111619	1116	Compound nouns and possessive forms
171	111620	1116	Possessive 's with time expressions – Two hour...
172	111621	1116	Relative clauses – defining and non-defining
173	111622	1116	There and it – preparatory subjects
174	111623	1116	Have – auxiliary or main verb
175	111624	1116	Ellipsis and substitution
176	111625	1116	Inversion with negative adverbials – adding emp...
177	111626	1116	Clauses of contrast, purpose, reason and result
178	111627	1116	Discourse markers – linking words
179	111628	1116	Participle clauses
180	111629	1116	Cleft sentences – adding emphasis

Рисунок 2.6 — Заповнені дані таблиці Topics

Таблиця RequestTypes визначає типи запитів на підтримку, які можуть надсилати користувачі (див. Рис. 2.7). Це дозволяє адміністратору швидко класифікувати та відповідати на запити користувачів, спрощуючи комунікацію та покращуючи загальний користувацький досвід.



	TypeID	TypeName	Description
1	1	Technical	Проблеми з технічною частиною сайту
2	2	Account	Питання, пов'язані з обліковими записами
3	3	Payment	Питання оплати
4	4	Other	Інші типи запитів

Рисунок 2.7 — Заповнені дані таблиці RequestTypes

2.2.3 Визначення умов цілісності даних

Підтримання цілісності даних є критично важливим аспектом будь-якої реляційної системи баз даних, особливо в освітніх платформах, які включають користувацький контент та динамічні взаємодії. У цьому проєкті цілісність даних зберігається завдяки використанню первинних ключів, обмежень зовнішнього ключа, типів даних та значень за замовчуванням у всіх таблицях бази даних.

Первинні ключі визначені для кожної таблиці, щоб забезпечити унікальність ідентифікатора кожного запису. Наприклад, UserID у таблиці Users, ExerciseID у таблиці UserExercises та Id_Topic у таблиці Topics виступають як унікальні ідентифікатори для послідовного посилання.

Обмеження зовнішнього ключа реалізовані для забезпечення цілісності посилань між пов'язаними таблицями. Наприклад:

- UserExercises.UserID посилається на Users.UserID, щоб гарантувати, що вправи завжди пов'язані з дійсними користувачами;
- Topics.EngLvl посилається на EnglishLvl.id_lvl, забезпечуючи зв'язок кожної теми з існуючим рівнем англійської мови;
- SupportRequests.AdminID та SupportRequests.UserID посилаються на таблиці Admins та Users відповідно, для перевірки належності запитів та адміністрування;
- ChatLogs.UserID пов'язаний з таблицею Users, що гарантує походження всіх записів чату від зареєстрованих користувачів.

Ці зв'язки за зовнішнім ключем запобігають появі "загублених" записів і забезпечують логічну послідовність у збережених даних.

Крім того, значення за замовчуванням (наприклад, getdate() для часових міток та 'Pending' для статусів запитів на підтримку) використовуються для забезпечення того, що нові записи створюються з дійсними початковими станами, зменшуючи ризик нульових або суперечливих записів.

Разом ці механізми утворюють надійну основу для забезпечення узгодженості даних, підтримки надійного функціонування для викладачів та адміністраторів, а також сприяння безпечній та передбачуваній взаємодії із системою.

2.3 Реалізація серверної частини системи

2.3.1 Розробка діаграми класів

Архітектура веб-застосунку побудована за принципами модульності, багаторівневості та інкапсуляції логіки. Вона передбачає чітке розділення обов’язків між різними частинами системи: конфігурацією, логікою взаємодії з базою даних, обробкою HTTP-запитів, забезпеченням безпеки та інтеграцією зі сторонніми інтелектуальними сервісами (див. Рис. 2.8). Центральним елементом архітектури є клас FlaskApp, який відповідає за запуск і ініціалізацію застосунку. Окремі модулі реалізують логіку обробки запитів до користувачів, адміністраторів, підтримки та навчального штучного інтелекту. Вся система взаємодіє з базою даних MSSQL, яка містить інформацію про користувачів, адміністрацію, рівні англійської мови, теми, збережені вправи та запити в технічну підтримку.

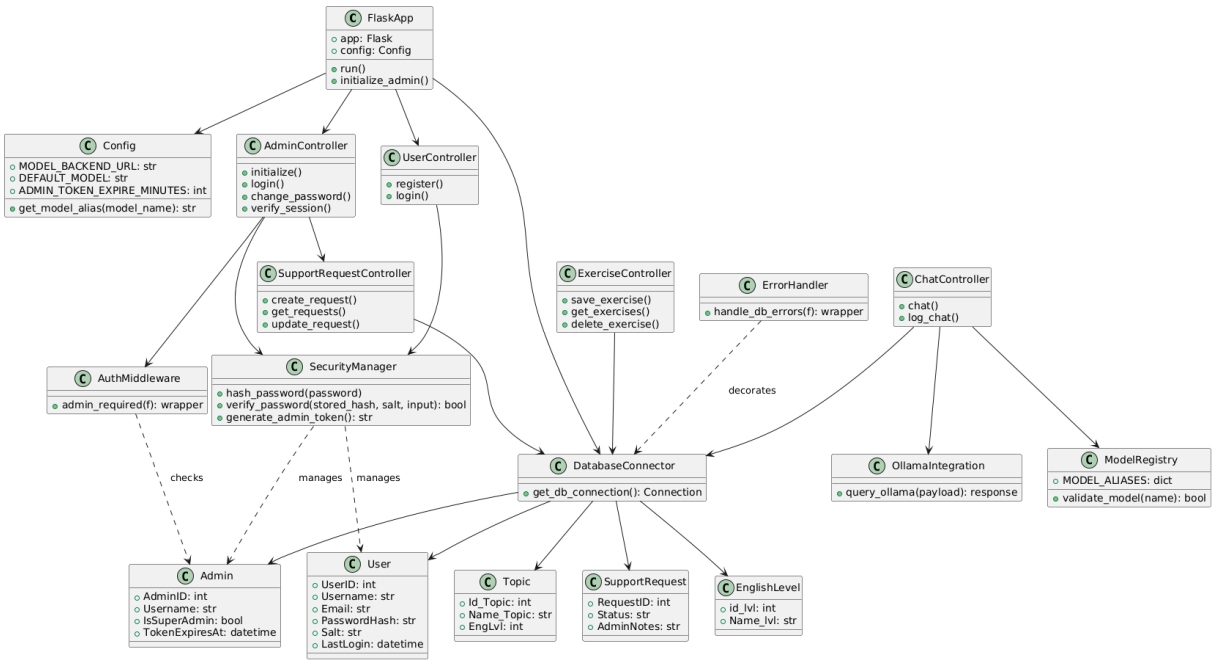


Рисунок 2.8 — Діаграма класів серверної частини вебсайту

Основний клас застосунку: FlaskApp.

Клас FlaskApp виконує роль центрального механізму запуску та ініціалізації застосунку на Flask. У ньому створюється екземпляр застосунку Flask, підключаються конфігураційні параметри, ініціалізуються контролери та викликається метод run(), який запускає веб-сервер. Крім того, клас реалізує метод initialize_admin(), що забезпечує створення облікового запису адміністратора, якщо такий відсутній у базі даних. Через залежності FlaskApp також підключає такі компоненти, як DatabaseConnector, Config, та контролери, які відповідають за обробку HTTP-запитів.

Конфігураційний модуль: Config.

Клас Config відповідає за централізоване зберігання параметрів конфігурації, необхідних для роботи застосунку. До таких параметрів належать URL сервера моделей штучного інтелекту (MODEL_BACKEND_URL), назва моделі за замовчуванням (DEFAULT_MODEL) та тривалість життя токена адміністратора (ADMIN_TOKEN_EXPIRE_MINUTES). Метод get_model_alias() дозволяє отримати зручний псевдонім моделі за її технічною назвою, що спрощує взаємодію з декількома інтегрованими LLM. Таким

чином, клас Config інкапсулює всі глобальні налаштування, підвищуючи гнучкість і масштабованість системи.

Компоненти бази даних: DatabaseConnector і ErrorHandler.

Клас DatabaseConnector реалізує логіку встановлення з'єднання з базою даних MSSQL. Він використовується у всіх контролерах, які потребують читання або запису даних. Метод get_db_connection() повертає активне з'єднання, через яке виконується SQL-запит. Цей підхід дозволяє централізовано керувати доступом до БД, а також забезпечує уніфікацію логіки підключення.

Допоміжним компонентом є ErrorHandler — декоратор, який обгортає функції, що працюють з базою даних, і перехоплює потенційні помилки. Це забезпечує стабільність системи та дозволяє уникнути непередбачуваних збоїв під час виконання операцій із БД. Його застосування зменшує дублювання коду обробки винятків у контролерах.

Механізми безпеки: SecurityManager і AuthMiddleware.

Безпека системи реалізована за допомогою двох основних класів. SecurityManager виконує хешування паролів, створення сольових значень та перевірку паролів при вході. Він також генерує токени адміністратора, які використовуються для автентифікації та авторизації дій, пов'язаних з адміністративним доступом. Клас працює з моделями User та Admin, забезпечуючи зберігання надійно захищених облікових даних.

Другий клас — AuthMiddleware — реалізує механізм перевірки адміністративних прав. Метод admin_required() виступає у ролі декоратора, який перевіряє токен адміністратора перед виконанням захищених маршрутів. Таким чином, цей компонент реалізує контроль доступу до критичних функцій системи.

Контролери: логіка обробки HTTP-запитів.

Контролери відповідають за логіку обробки HTTP-запитів з боку користувачів і адміністраторів, а також за інтеграцію з моделями штучного інтелекту.

AdminController обробляє вхід адміністратора, зміну пароля та перевірку автентифікації. Він використовує SecurityManager для перевірки паролів і генерації токенів, а також AuthMiddleware для захисту маршрутів. Крім того, цей контролер має доступ до SupportRequestController, що дозволяє адміністраторам розглядати звернення користувачів.

UserController обробляє реєстрацію та автентифікацію звичайних користувачів. Для забезпечення безпеки реєстраційних даних він теж використовує SecurityManager.

ChatController відповідає за основну функціональність: надсилання запиту до LLM-моделі та збереження історії чату. Він використовує класи OllamaIntegration та ModelRegistry для надсилання запитів до моделей і перевірки їх назв. Збереження історії чату здійснюється через DatabaseConnector.

ExerciseController дозволяє користувачам зберігати створені вправи, переглядати їх або видаляти. Всі дії здійснюються шляхом взаємодії з базою даних через DatabaseConnector.

SupportRequestController реалізує створення, оновлення й перегляд технічних звернень користувачів. Цей модуль дозволяє адміністраторам відповідати на запити та залишати нотатки.

Інтеграція з LLM: OllamaIntegration і ModelRegistry.

OllamaIntegration — клас, що забезпечує взаємодію із зовнішнім сервером моделей штучного інтелекту через HTTP-запити. Метод query_ollama(payload) надсилає вхідні дані до моделі та повертає відповідь. Цей клас ізольовує логіку запиту, зберігаючи інші компоненти незалежними від особливостей API.

ModelRegistry містить словник псевдонімів моделей (MODEL_ALIASES) та перевіряє, чи дозволено використовувати певну модель. Це забезпечує гнучке налаштування переліку доступних моделей без потреби змінювати код інших компонентів.

Моделі бази даних.

Базові моделі представляють основні сутності в системі. Клас User зберігає облікову інформацію звичайних користувачів, включаючи хешовані паролі, логіни та дату останнього входу. Admin зберігає подібну інформацію про адміністраторів, включаючи рівень прав доступу та термін дії авторизаційного токена.

Класи EnglishLevel і Topic реалізують навчальні компоненти — мовні рівні та теми відповідно. Topic містить зовнішній ключ до EnglishLevel, що дозволяє обмежити теми до певного рівня володіння мовою.

SupportRequest містить звернення користувачів до технічної підтримки, з можливістю оновлення статусу звернення та залишення коментарів адміністратора.

Взаємозв’язки між компонентами: зв’язки між компонентами системи чітко структуровані. FlaskApp є центральною точкою, що ініціалізує інші модулі. Контролери взаємодіють із SecurityManager та DatabaseConnector, а також за потреби викликають методи інтеграції з LLM. ErrorHandler використовується для захисту базових операцій з БД. Таке структурування забезпечує високий рівень підтримуваності, повторного використання коду та масштабованості.

2.3.2 Інтеграція функціональних компонентів у серверну архітектуру

Сервер функціонує як проміжне програмне забезпечення, що поєднує фронтенд-додаток та основну логіку бекенда, включаючи операції з базами даних та зв'язок з великими мовними моделями (LLM). Побудований з використанням Flask, мікро-веб-фреймворку на Python, цей сервер обробляє HTTP-запити, ініційовані з фронтенду React, обробляє логіку, таку як автентифікація, зв'язок з LLM або доступ до бази даних, і повертає структуровані відповіді у форматі JSON. Ця конструкція підтримує модульність, масштабованість та ремонтопридатність.

Програма використовує декілька розширень Flask та стандартні бібліотеки Python:

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import requests
import logging
import time
import pyodbc
import bcrypt
import secrets
import datetime
from functools import wraps
import os
```

Flask: мікрофреймворк для визначення HTTP-маршрутів та обробки запитів/відповідей.

Flask-CORS: дозволяє Cross-Origin Resource Sharing (CORS), дозволяючи клієнтам зовнішнього інтерфейсу (наприклад, React) виконувати AJAX-запити.

requests: використовується для пересилання взаємодій у чаті до зовнішнього backend-сервера моделі ШІ (Ollama).

pyodbc: інтерфейс ODBC для підключення до Microsoft SQL Server (база даних "Engflow").

bcrypt та secrets: забезпечують безпечне хешування паролів та генерацію токенів відповідно.

logging: записує операції та помилки сервера.

datetime, time: управління часовими мітками.

functools.wraps: використовується для створення користувацьких декораторів для перевірки автентифікації.

Словник CONFIG є централізованим сховищем параметрів конфігурації для веб-додатка, побудованого на Flask. Він містить ключові налаштування, що визначають:

- роботу з штучним інтелектом (Ollama API),
- безпеку та автентифікацію,
- поведінку системи за замовчуванням.

```
CONFIG = {
    "MODEL_BACKEND_URL": "http://localhost:11434/api/chat",
    "DEFAULT_MODEL": "deepseek-r1",
    "DEFAULT_MAX_TOKENS": 1000,
    "RESPONSE_TIMEOUT": 60,
    "ENFORCE_ENGLISH_RESPONSES": True,
    "BASE_PROMPT": "You are a helpful AI assistant. Always respond in clear, proper English. Provide accurate and well-structured answers.",
    "MODEL_ALIASES": {
        "deepseek-r1": "deepseek-r1",
        "mistral": "mistral",
        "llama3.1:8b": "llama3.1:8b",
        "nezahatkorkmaz/deepseek-v3": "nezahatkorkmaz/deepseek-v3"
    },
    "ADMIN_TOKEN_EXPIRE_MINUTES": 30,
    "ADMIN_PASSWORD_CHANGE_MINUTES": 30,
    "SUPER_ADMIN_USERNAME": "admin"
}
```

Пояснення елементів CONFIG (див. Табл. 2.10 та табл. 2.11).

Таблиця 2.10 — Налаштування AI-моделей (Ollama API)

Параметр	Пояснення
MODEL_BACKEND_URL	URL-адреса API Ollama для взаємодії з AI-моделями.
DEFAULT_MODEL	Модель за замовчуванням для генерації відповідей.
DEFAULT_MAX_TOKENS	Максимальна кількість токенів у відповіді AI.
RESPONSE_TIMEOUT	Час очікування відповіді (у секундах) перед таймаутом.
ENFORCE_ENGLISH_RESPONSES	Чи обмежувати відповіді англійською мовою.
BASE_PROMPT	Базовий промт для AI, що задає стиль відповідей.
MODEL_ALIASES	Словник для маппінгу зручних назв моделей на їхні технічні ідентифікатор

Таблиця 2.11 — Налаштування безпеки та адміністрування

Параметр	Пояснення
ADMIN_TOKEN_EXPIRE_MINUTES	Час життя JWT-токена адміністратора (у хвилинах).
ADMIN_PASSWORD_CHANGE_MINUTES	Інтервал примусової зміни пароля адміністратора.
SUPER_ADMIN_USERNAME	Логін супер-адміністратора для ініціалізації системи.

Усі взаємодії з базою даних SQL Server використовують єдину допоміжну функцію:

```
def get_db_connection():
    conn_str = (
        "DRIVER={ODBC Driver 17 for SQL Server};"
        "SERVER=DESKTOP-C6L2A2J;"
        "DATABASE=Engflow;"
        "Trusted_Connection=yes;"
    )
    return pyodbc.connect(conn_str)
```

Далі оглянемо ключові процеси та базову логіку цих процесів (див. Рис. 2.9).

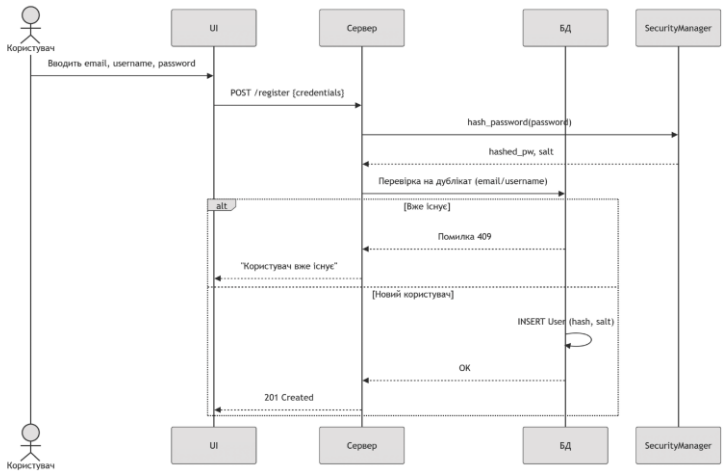


Рисунок 2.9 — Принцип роботи системи реєстрації

Процес реєстрації користувачів забезпечує створення безпечних та унікальних облікових записів у системі. Послідовність починається, коли користувач надсилає свої облікові дані через інтерфейс користувача. Серверна частина отримує ці дані через запит POST /register.

Отримавши пароль, сервер делегує операцію хешування SecurityManager, який застосовує безпечну криптографічну хеш-функцію (наприклад, PBKDF2 або bcrypt) разом із згенерованою сіллю. Цей крок є обов'язковим для захисту конфіденційних даних користувачів від потенційних витоків.

Далі сервер перевіряє, чи надані електронна пошта та ім'я користувача не існують у базі даних. Якщо виявлено дублікат, користувач інформується відповіддю 409 Conflict. В іншому випадку дані нового користувача, включаючи хешований пароль та сіль, зберігаються в таблиці Users, і повертається відповідь 201 Created. Цей потік забезпечує цілісність, безпеку та унікальність записів користувачів. (див. Рис. 2.10).

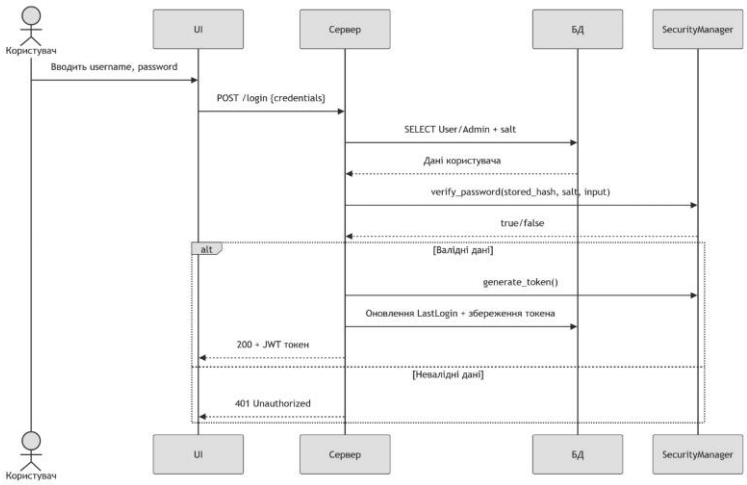


Рисунок 2.10 — Принцип роботи авторизації користувача та адміністратора

Механізм входу перевіряє облікові дані користувача та генерує токен сесії. Коли користувач (або адміністратор) надсилає свої дані для входу через POST /login, сервер запитує базу даних для отримання відповідного запису та солі.

Пароль потім перевіряється за допомогою SecurityManager, який порівнює введений пароль (хешований з отриманою сіллю) зі збереженим хешем. Якщо облікові дані правильні, система генерує безпечний токен за допомогою криптографічної випадковості (наприклад, secrets.token_hex(32)), зберігає його в таблиці UserSessions із часом закінчення терміну дії та відповідає повідомленням 200 OK, що містить токен. Невірні облікові дані призводять до відповіді 401 Unauthorized. Цей підхід забезпечує безпечну автентифікацію, зберігаючи стан сесії. (див. Рис. 2.11).

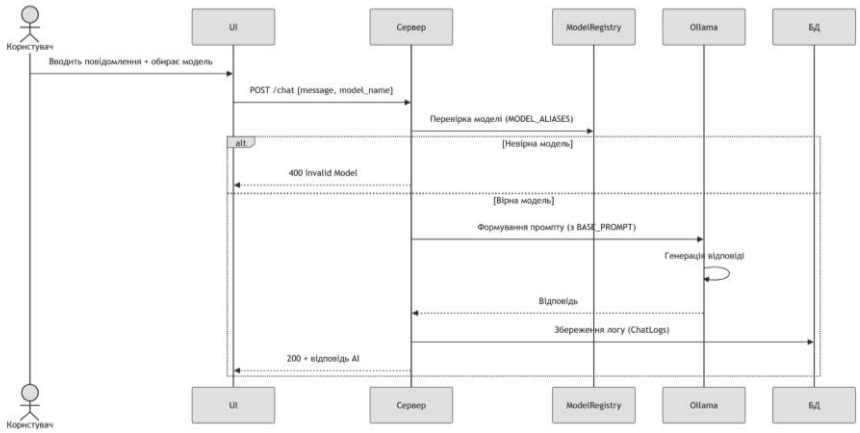


Рисунок 2.11 — Обробка запитів чату через Ollama API

Функція чату дозволяє користувачам взаємодіяти з великими мовними моделями (LLM) через Ollama API. Коли користувач надсилає повідомлення та вибирає модель, запит надсилається як POST /chat.

Сервер спочатку перевіряє, чи підтримується вибрана модель, звертаючись до ModelRegistry, який відображає зрозумілі для людини псевдоніми на ідентифікатори моделей. Якщо модель недійсна, повертається відповідь 400 Bad Request.

Для дійсних моделей сервер формує запит, додаючи повідомлення користувача до попередньо визначеного BASE_PROMPT і надсилає його до сервісу Ollama. Згенерована відповідь потім повертається клієнту. Одночасно журнал розмови зберігається в таблиці ChatLogs. Ця архітектура відокремлює обробку штучного інтелекту від логіки інтерфейсу користувача, забезпечуючи постійне зберігання взаємодій у чаті для аудиту або педагогічного аналізу. (див. Рис. 2.12).

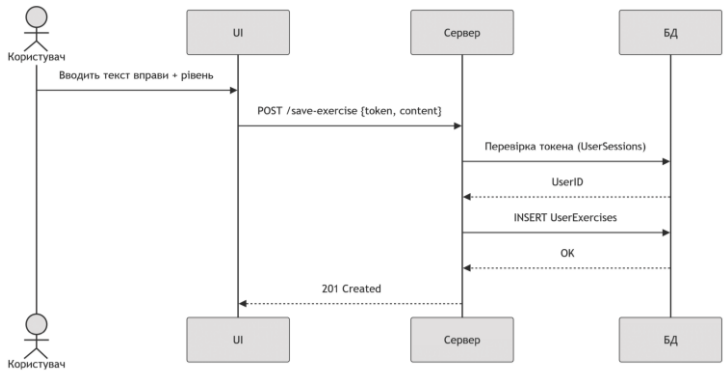


Рисунок 2.12 — Принцип збереження створених вправ

Для полегшення персоналізованого навчання система дозволяє користувачам зберігати власні вправи. Процес починається, коли користувач надсилає вміст вправи та метадані (такі як рівень англійської мови) через POST /save-exercise.

Сервер спочатку перевіряє токен користувача, запитуючи таблицю UserSessions. Після успішної автентифікації він отримує UserID і вставляє нову вправу в таблицю UserExercises. Відповідь 201 Created підтверджує операцію. Цей робочий процес гарантує, що вправи пов'язані з автентифікованими користувачами і можуть бути пізніше отримані або змінені (див. Рис. 2.13).

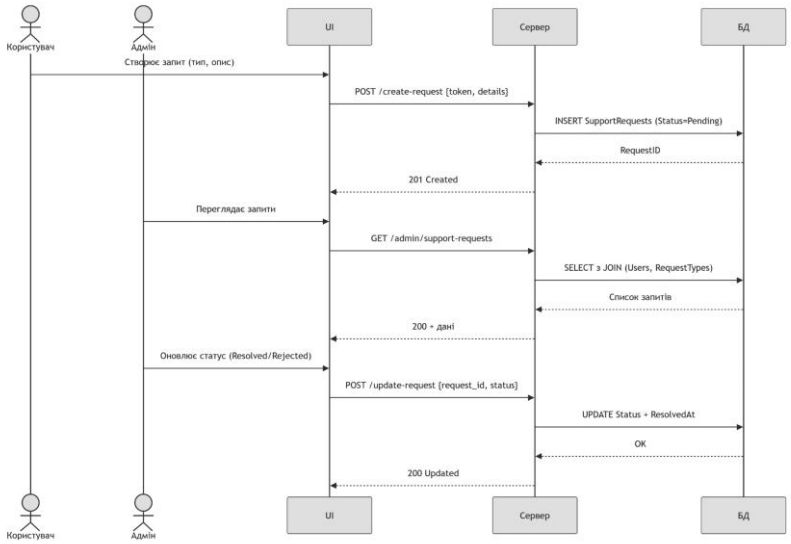


Рисунок 2.13 — Життєвий цикл запитів на підтримку

Система включає функцію підтримки для полегшення зв'язку між користувачами та адміністраторами. Коли користувач надсилає запит на підтримку через POST /create-request, сервер перевіряє токен і вставляє запит у таблицю SupportRequests, ініціалізуючи його статус як "Очікує".

Адміністратори можуть пізніше отримувати ці запити через GET /admin/support-requests, який об'єднує дані з Users та RequestTypes для надання контекстної інформації. При вирішенні або відхиленні запиту адміністратори використовують POST /update-request, який оновлює поля Status та ResolvedAt. Ця повноцінна система підтримки забезпечує підзвітність та відстежуваність для проблем, про які повідомляють користувачі (див. Рис. 2.14).

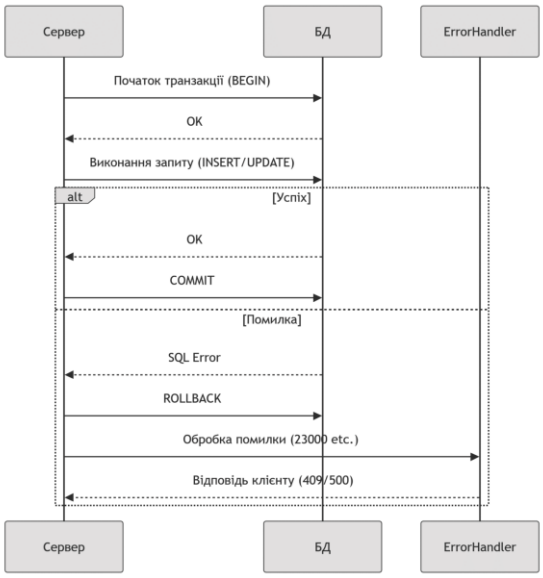


Рисунок 2.14 — Робота системи транзакції у сервері

Система включає функцію підтримки для полегшення зв'язку між користувачами та адміністраторами. Коли користувач надсилає запит на підтримку через POST /create-request, сервер перевіряє токен і вставляє запит у таблицю SupportRequests, ініціалізуючи його статус як "Очікує".

Адміністратори можуть пізніше отримувати ці запити через GET /admin/support-requests, який об'єднує дані з Users та RequestTypes для надання контекстної інформації. При вирішенні або відхиленні запиту адміністратори використовують POST /update-request, який оновлює поля Status та ResolvedAt.

Ця повноцінна система підтримки забезпечує підзвітність та відстежуваність для проблем, про які повідомляють користувачі.

2.4 Реалізація клієнтської частини вебсайту

2.4.1 Проєктування та реалізація головної сторінки вебсайту

Головна сторінка вебдодатку «Інтерактивний помічник для викладачів іноземних мов з використанням інтелектуальних агентів» є центральним вузлом доступу до різноманітних педагогічних інструментів на основі ШІ (див. Рис. 2.15).

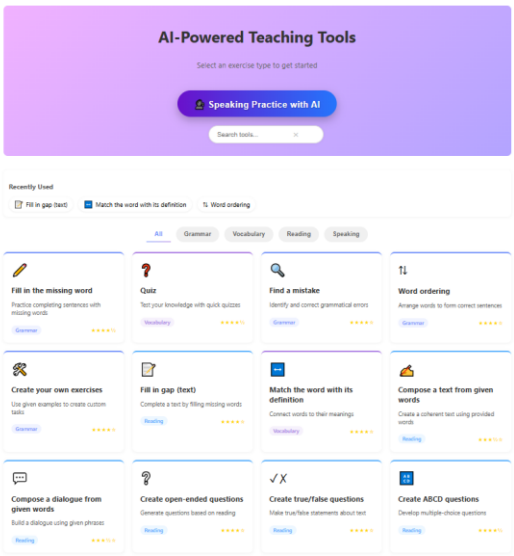


Рисунок 2.15 — Вигляд головної сторінки сайту

Після завантаження головної сторінки користувачі бачать динамічно анімований заголовок та вступне повідомлення, що пропонує обрати тип вправи. Кнопка «Практика говоріння з ШІ» надає швидкий доступ до інтерактивного чат-інтерфейсу на базі штучного інтелекту.

Сторінка містить кілька ключових інтерактивних компонентів:

- вкладки категорій: розташовані поблизу верхньої частини сторінки, вони дозволяють фільтрувати інструменти за тематичними категоріями: "Усі", "Граматика", "Лексика", "Читання" та "Говоріння". Кожна вкладка змінює

відображуваний список доступних інструментів відповідно до обраної категорії;

- панель пошуку: інтегроване безпосередньо під заголовком, це поле введення дозволяє користувачам здійснювати пошук за ключовими словами серед усіх інструментів;
- картки інструментів: кожна доступна вправа візуально представлена у вигляді картки, що містить іконку, короткий опис, позначку категорії та графічний рейтинг популярності;
- нещодавно використані інструменти: додаток також зберігає три останні інструменти, до яких звертався користувач, використовуючи localStorage, та відображає їх у спеціальному розділі "Нещодавно використані" для швидкого доступу;
- підказки та анімація: додаткові функції для покращення взаємодії з користувачем включають анімовані переходи, підказки при наведенні курсору та адаптивну поведінку при взаємодії, реалізовані за допомогою бібліотеки framer-motion.

Головна сторінка реалізована як функціональний React-компонент із застосуванням хуків, таких як useState та useEffect. Маршрутизація керується за допомогою хука useNavigate з бібліотеки react-router-dom, що дозволяє програмну навігацію до підсторінок. Кожен інструмент у системі представлений об'єктом JavaScript, що містить метадані: ідентифікатор, назву, опис, шлях, іконку, категорію та числовий показник популярності.

Логіка фільтрації реалізована за допомогою ланцюгових функцій filter() та includes(), застосованих до повного списку інструментів (allFunctions). Це дозволяє оновлювати сітку інструментів у реальному часі, коли користувачі вводять пошукові запити або перемикаються між вкладками категорій.

2.4.2 Створення навігаційної панелі та загальної структури макету (layout)

Ключовим аспектом розробки адаптивного та орієнтованого на користувача вебдодатку є створення цілісного макета та інтуїтивно зрозумілої навігаційної панелі (заголовка). У контексті проєкту макет слугує структурною основою для відтворення всіх сторінок та інтерактивних компонентів. Заголовок, своєю чергою, забезпечує доступні точки входу до основних функціональних можливостей системи, підтримуючи безперешкодну взаємодію між користувачем та платформою. (Див. Рис. 2.16).

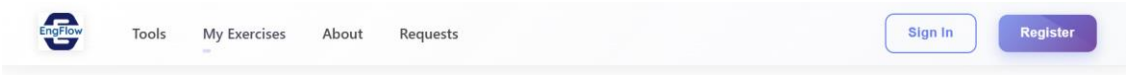


Рисунок 2.16 — Вигляд хедеру сторінки

Компонент Header реалізує основну навігаційну панель, надаючи користувачам прямий доступ до ключових модулів додатка, а саме:

- логотип: візуальний ідентифікатор, що є клікабельним посиланням на домашню сторінку;
- інструменти: доступ до основних функцій на базі штучного інтелекту;
- мої вправи: перегляд та управління збереженими вправами для автентифікованих користувачів;
- про нас: інформація про систему та її призначення;
- запити: можливість надсилати або переглядати запити до служби підтримки та ШІ.

Для запобігання несанкціонованому доступу посилання на «Мої вправи» та «Запити» захищені. При спробі неавтентифікованого користувача отримати доступ до цих маршрутів відбувається перенаправлення на сторінку входу.

Компонент Layout є кореневим контейнером для Header та динамічного вмісту. Header використовує AuthContext для перевірки авторизації та

умовного рендерингу кнопок входу/виходу, а також взаємодіє з React Router для навігації. AuthContext централізує логіку аутентифікації, а React Router забезпечує клієнтську маршрутизацію. (див. Рис. 2.17).

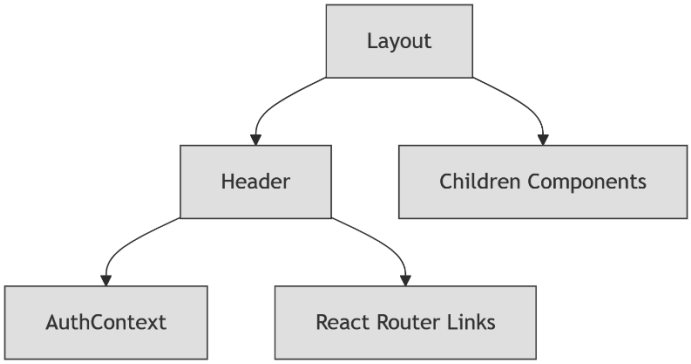


Рисунок 2.17 — Діаграма компонентів

2.4.3 Інтеграція механізмів маршрутизації (на базі React Router)

Конфігурація маршрутизації здійснюється в головному компоненті App.js. Застосовано компоненти <Routes> та <Route>, які визначають відповідність між URL-шляхами та компонентами React, що мають бути відрендерені.(Див. Рис. 2.18).

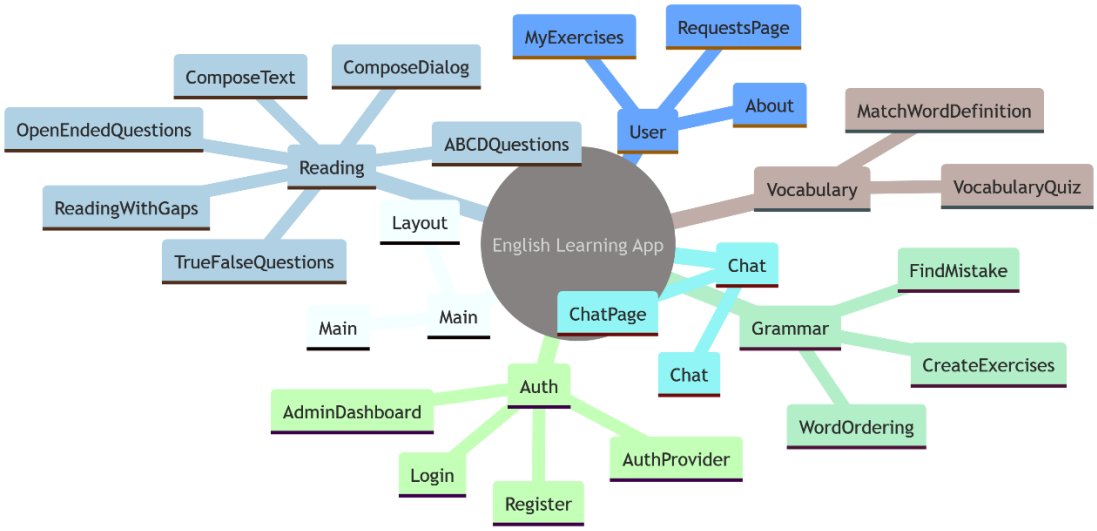


Рисунок 2.18 — Діаграма компонентів App.js

Інтеграція React Router дозволила створити модульну, масштабовану та зручну навігаційну систему. Це забезпечило ефективний механізм для

управління доступом до різноманітних функціональних можливостей помічника, включаючи створення вправ, засоби практики на основі ШІ, адміністративний контроль та управління даними користувачів. Ця архітектура є ключовою для забезпечення цілісного та інтуїтивно зрозумілого інтерфейсу для викладачів та адміністраторів системи.

2.4.4 Розробка інформаційної сторінки «Про проект»

Інформаційна сторінка «Про сайт» є невід'ємним компонентом сучасних веб-додатків, що забезпечує ефективну адаптацію користувачів, роз'яснення функціональних можливостей системи та підвищення рівня довіри

Сторінка «Про сайт» реалізована як незалежний React-компонент. Розробка здійснювалась з використанням сучасних стандартів JavaScript та синтаксису JSX. Для стилізації застосовані каскадні таблиці стилів (CSS). Анімаційні ефекти інтегровані за допомогою бібліотеки Framer Motion. Структурно сторінка поділена на кілька візуально відокремлених секцій, кожна з яких виконує специфічну функціональну роль у поданні інформації. (Див. Рис. 2.19)

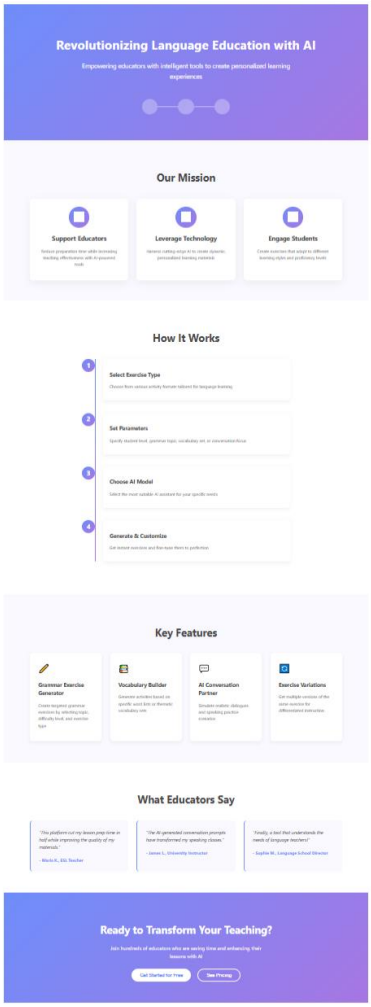


Рисунок 2.19 — Зовнішній вигляд сторінки «Про сайт»

2.5 Впровадження підсистеми автентифікації користувачів

2.5.1 Розробка інтерфейсу та логіки сторінки реєстрації нового користувача

Сторінка реєстрації є точкою входу для нових користувачів у систему. Вона розроблена як інтуїтивно зрозуміла та безпечна, забезпечуючи як зручність використання, так і захист персональних даних. Сторінка реалізована за допомогою React, сучасної бібліотеки JavaScript для створення користувацьких інтерфейсів. Вона взаємодіє з серверною частиною через HTTP-запити для завершення процесу реєстрації.(див. Рис. 2.20).

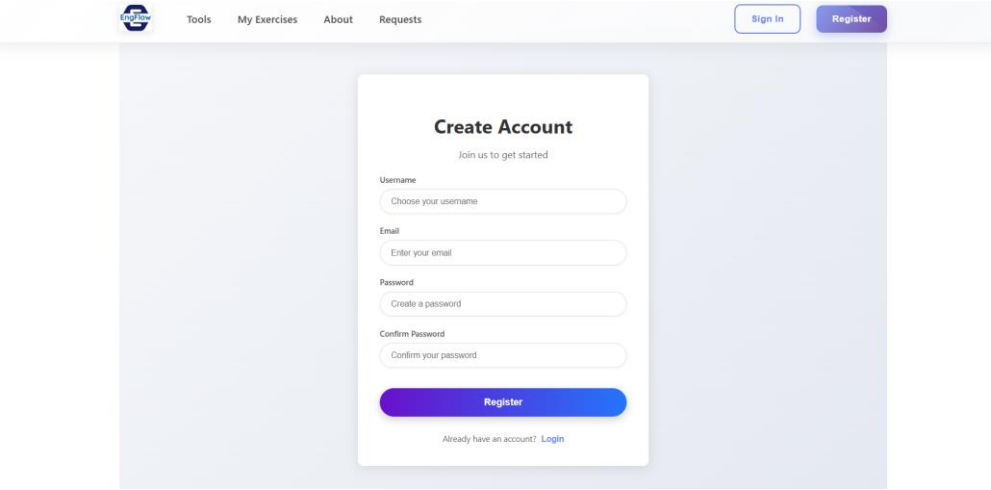


Рисунок 2.20 — Зовнішній вигляд сторінки реєстрації

Інтерфейс реєстрації надає структуровану форму, яка пропонує користувачам ввести таку інформацію: ім'я користувача, електронна пошта, пароль.

Після надсилання форми введені дані проходять валідацію на стороні клієнта перед відправленням на сервер. Це включає перевірку відповідності введених паролів. Якщо валідація проходить успішно, POST-запит надсилається до кінцевої точки backend /register, передаючи дані користувача у форматі JSON. (Див. Рис. 2.21).

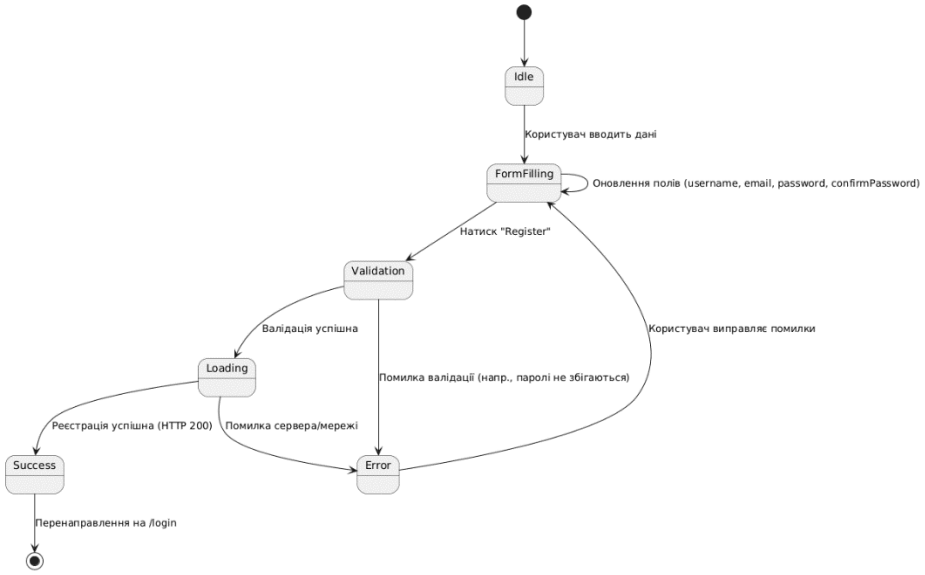


Рисунок 2.21 — Діаграма стану форми реєстрації

Початковий стан Idle представляє сторінку до будь-якої взаємодії з користувачем.

У стані FormFilling користувачі вводять свої облікові дані. Кожне натискання клавіші оновлює відповідний стан у компоненті React.

Після натискання кнопки «Зареєструватися» система переходить у стан Validation, де вхідні дані програмно перевіряються.

Якщо валідація успішна, система переходить до Loading, під час якого відбувається комунікація з бекенд-сервером.

Успішна відповідь від сервера переводить додаток у стан Success, що викликає перенаправлення на сторінку входу.

Якщо виникає помилка (через введені дані або відповідь сервера), активується стан Error. Потім система повертається до FormFilling, дозволяючи користувачеві переглянути свої дані.

2.5.2 Розробка інтерфейсу та логіки сторінки входу

Інтерфейс входу дозволяє користувачу вводити свої облікові дані та надсилати їх для автентифікації. Він підтримує два типи користувачів: стандартних користувачів та адміністраторів. Користувач може перемикатися між цими режимами за допомогою спеціальної кнопки, що динамічно змінює текст-заповнювач та цільові кінцеві точки. (див. Рис. 2.22)

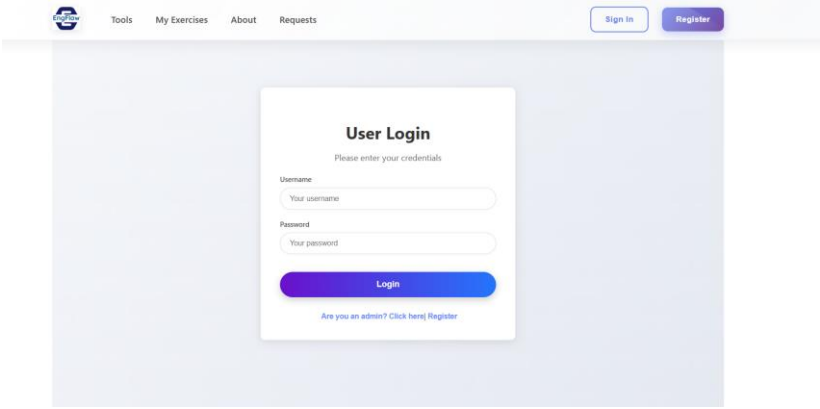


Рисунок 2.22 — Зовнішній вигляд сторінки авторизації

Змінні стану керуються за допомогою хука useState з React, який обробляє дані форми (username, password та тип входу), стани завантаження, повідомлення про помилки та умовне відображення компонентів, таких як форма зміни пароля для адміністраторів. Хук useNavigate з react-router-dom використовується для програмного перенаправлення користувачів після успішної автентифікації.

Після надсилання форми спрацьовує функція handleSubmit. Залежно від типу входу, вона надсилає POST-запит до кінцевої точки /login або /admin/login за допомогою Axios. У разі успішного входу токен зберігається в локальному сховищі, а інформація про користувача зберігається в глобальному контексті автентифікації за допомогою методу login або adminLogin. (Див. Рис. 2.23).

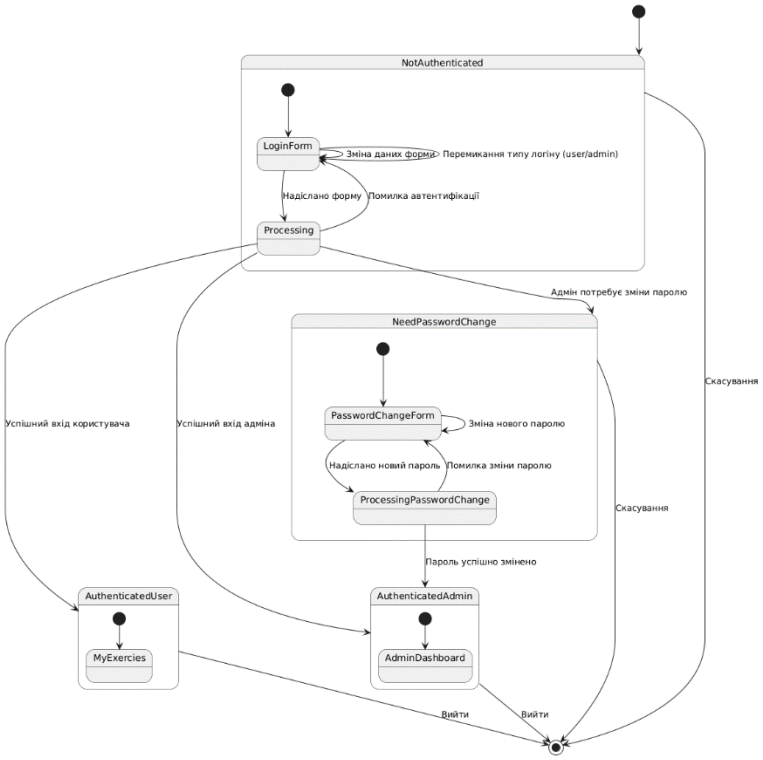


Рисунок 2.23 — Діаграма стану форми авторизації

Початковий стан – NotAuthenticated, де користувачі взаємодіють з LoginForm. Тут вони можуть змінювати вхідні дані та перемикатися між режимами користувача та адміністратора.

Після надсилання форми процес переходить у стан Processing. Залежно від відповіді бекенду, він або:

- Повертається до LoginForm з помилкою,
- Переходить до AuthenticatedUser або AuthenticatedAdmin після успішного входу,
- Або переходить до NeedPasswordChange, якщо адміністратор повинен змінити свій пароль.

У стані NeedPasswordChange користувач взаємодіє з PasswordChangeForm. Цей стан включає введення нового пароля та його надсилання для обробки. У разі успіху адміністратор перенаправляється до стану AuthenticatedAdmin.

Після автентифікації користувачі перенаправляються на MyExercises, а адміністратори — на AdminDashboard.

2.5.3 Імплементація механізму контексту автентифікації

Для забезпечення безпечного та постійного контролю доступу в застосунку було реалізовано механізм контексту автентифікації з використанням React Context API. Цей механізм надає можливість глобального керування станом користувацьких сесій у всьому застосунку, підтримуючи як стандартних користувачів, так і адміністративні ролі. Він централізує логіку автентифікації та дозволяє дочірнім компонентам отримувати доступ до даних користувача та функцій автентифікації без "prop drilling".(див. Рис. 2.24).

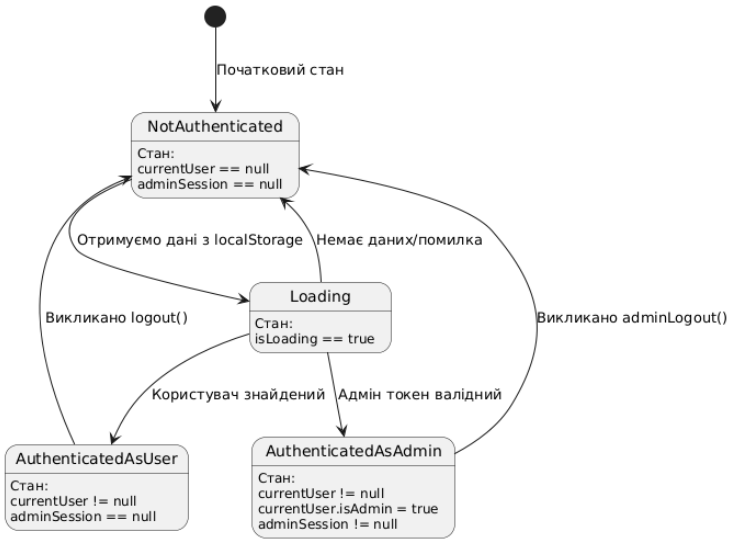


Рисунок 2.24 — Діаграма стану форми авторизації

Застосунок починається у стані NotAuthenticated, де жоден користувач не авторизований.

Після монтування застосунок переходить у стан Loading, де він намагається прочитати дані сесії з localStorage та перевірити їх.

Залежно від результату, система переходить або в AuthenticatedAsUser, або в AuthenticatedAsAdmin. Ці стани визначаються наявністю дійсного об'єкта currentUser та, за бажанням, дійсного adminSession.

Система повертається до NotAuthenticated при виконанні logout() або adminLogout(), очищаючи всі дані сесії.

2.6 Розробка основного функціоналу системи

2.6.1 Розробка інтерфейсу для завдань з граматики

Інтерфейс для граматичних завдань розроблено для допомоги викладачам мови у генеруванні індивідуальних граматичних вправ з використанням моделей штучного інтелекту. Основною метою є автоматизація створення завдань на основі обраної граматичної теми та рівня володіння

мовою. Таким чином, система зменшує ручне навантаження на викладачів та підвищує різноманітність і креативність вправ.(Див. Рис. 2.25)

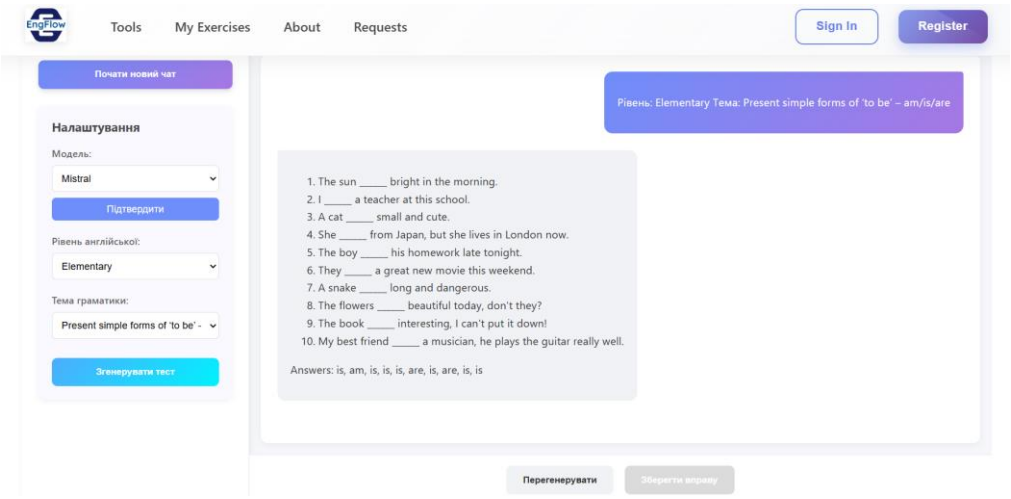


Рисунок 2.25 — Зовнішній інтерфейс сторінки для завдань з граматики

Цей компонент системи дозволяє користувачам (див. Рис. 2.26):

- вибрати цільовий рівень володіння англійською мовою
- вибрати відповідну граматичну тему
- взаємодіяти з моделлю штучного інтелекту для генерації контекстно-релевантних граматичних вправ.
- редагувати та уточнювати згенерований штучним інтелектом контент.
- зберігати остаточну версію завдання для подальшого використання.

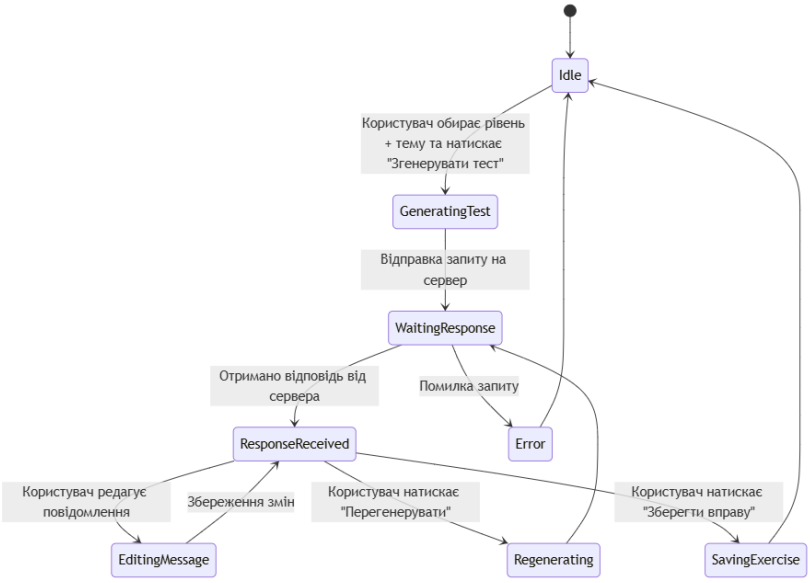


Рисунок 2.26 — Діаграма станів для сторінок граматичних вправ

У початковому неактивному стані (idle) інтерфейс залишається пасивним до моменту, коли користувач обирає рівень володіння мовою та тему, а потім натискає кнопку «Згенерувати тест». У цей момент клієнт ініціює HTTP-запит до бекенду, переходячи в стан очікування відповіді (waiting-for-response). Сервер генерує зміст тесту (наприклад, шляхом звернення до моделі штучного інтелекту або вилучення шаблону) та надсилає його у відповідь. Після успішного отримання даних користувацький інтерфейс переходить у стан «Відповідь отримана» (ResponseReceived), відображаючи новостворений тест. У разі виникнення помилки (збій мережі або виняток на стороні сервера) інтерфейс негайно повертається до неактивного стану, надаючи користувачеві можливість повторної спроби.

Після відображення тесту користувач може змінити його зміст — це переводить інтерфейс у стан «Редагування повідомлення» (EditingMessage), під час якого всі модифікації застосовуються локально. Збереження цих змін повертає стан до «Відповідь отримана». З цього стану натискання кнопки «Перегенерувати» надсилає новий запит до сервера (переходячи через стан «Перегенерація» (Regenerating) назад до «Очікування відповіді»), що призводить до створення нового змісту тесту. Альтернативно, коли користувач задоволений результатом і натискає «Зберегти вправу», надсилається запит на збереження даних до бази даних. Після підтвердження операції стан повертається до неактивного, готового до наступної взаємодії.

На сторінках із граматичними завданнями використовуються різні промпти для штучного інтелекту, що залежать від типу вправи. Ці промпти скеровують мовну модель на генерацію відповідних граматичних завдань, адаптованих до рівня та мети користувача. Використання специфічних промптів забезпечує педагогічну релевантність та послідовність завдань у різних граматичних категоріях.

```
const generatedQuery = `
You are an expert English language test creator.
Your task is to design a missing-word exercise focusing on ${topicText} for a ${englishLevelText} level learner.
Requirements:
Exercise Type: Fill in the blank  Each sentence must contain one missing word ("_____") that aligns with the specified grammar topic.
Number of Sentences: 10 unique and varied sentences.
Grammar Focus: Ensure the missing word strictly follows the rules of ${topicText}.
Creativity & Variation:
Sentences should be natural, engaging, and contextually diverse (e.g., different subjects, tenses, or scenarios).
Avoid repetition in structure or vocabulary.
Difficulty should match the ${englishLevelText} level.
Output Format:
First, list all 10 sentences with blanks.
After the sentences, provide a numbered answer key in the order of the blanks.
`;
```

Рисунок 2.26 — Приклад промпту для завдання

2.6.2 Розробка інтерфейсу для завдань з лексики

Цей функціональний блок призначений для генерації вправ на основі заданого користувачем словника. Відправною точкою у побудові таких завдань є механізм, аналогічний до чату: користувач взаємодіє з інтерфейсом, обирає параметри (рівень англійської мови, модель ШІ) та вводить список лексичних одиниць, після чого модель генерує вправу за визначеним шаблоном. Один із прикладів таких завдань — «Match the word with its definition», де ШІ створює парні відповідності між словами і їхніми визначеннями. (Див. Рис. 2.27)

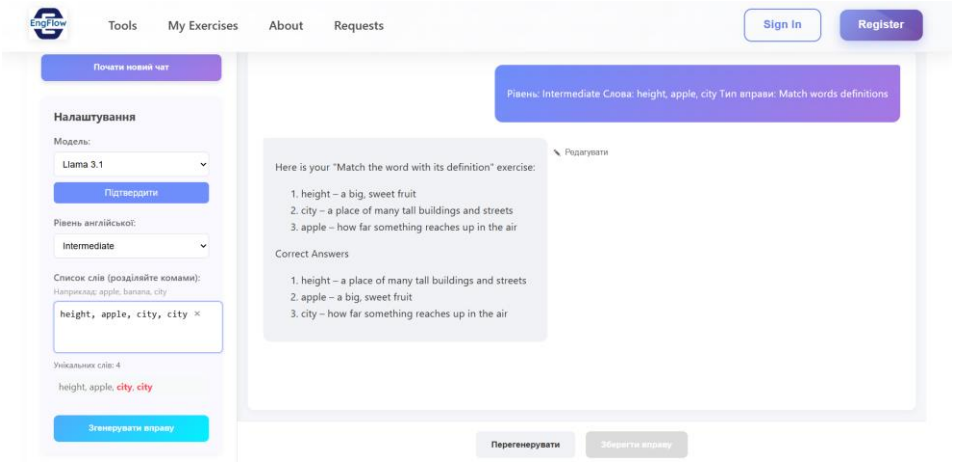


Рисунок 2.27 — Зовнішній вигляд сторінки для завдань із лексики

У технічному плані компонент підтримує збереження стану діалогу (через useState), а також використовує контекст авторизації користувача для верифікації при збереженні вправ (useAuth). Дані про рівні англійської мови отримуються з бекенду через запит до http://localhost:5000/english-levels.

Користувач вводить лексичні одиниці у вигляді списку слів, який обробляється з урахуванням валідації, унікальності та відсутності заборонених символів. Після цього формується запит з інструкцією для ШІ у вигляді англомовного промпту, де чітко вказується рівень складності, структура вправи та очікуваний формат відповіді. (Див. Рис. 2.28).

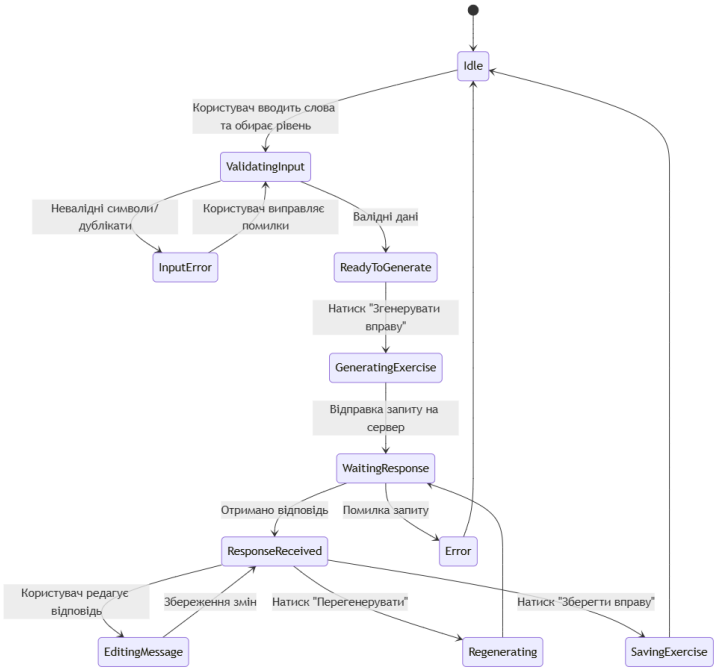


Рисунок 2.28 — Діаграма станів для сторінок лексичних завдань

Якщо результат користувача задовольняє, його можна зберегти через окремий запит до /save-exercise, що включає тип вправи, рівень, вміст, заголовок і токен авторизації користувача. Таким чином, реалізовано повний цикл: від введення лексичного матеріалу до збереження вправи в базі даних.

2.6.3 Розробка інтерфейсу для завдань з читання

Сторінка створення вправ із читання у вебзастосунку викладача англійської мови призначена для автоматизованого генерування завдань на основі введеного навчального тексту. Вона дозволяє користувачеві створити вправу на розуміння прочитаного з типом завдань «True/False», орієнтуючись на заданий рівень володіння мовою. Користувач вводить текст, обирає мовну

модель (наприклад, Deepseek або Mistral) і відповідний рівень англійської, після чого система формує інструкцію для ШІ, який повертає сформовану вправу. Таким чином, сторінка слугує інтерфейсом для швидкого створення інтерактивних завдань, що враховують як зміст тексту, так і складність лексики та граматики, відповідно до рівня підготовки учня.

2.6.4 Реалізація функціоналу збереження вправ користувачів

Однією з ключових особливостей інтерактивного помічника для викладачів іноземних мов є можливість зберігання та управління вправами, згенерованими штучним інтелектом. Цей функціонал дозволяє користувачам отримувати, переглядати, групувати та видаляти вправи в будь-який час, надаючи необхідні інструменти для планування уроків та педагогічного відстеження. Доступ до реалізації цієї функції здійснюється через сторінку «Мої вправи», яка функціонує як персоналізоване сховище збережених завдань. (Див. Рис. 2.29).

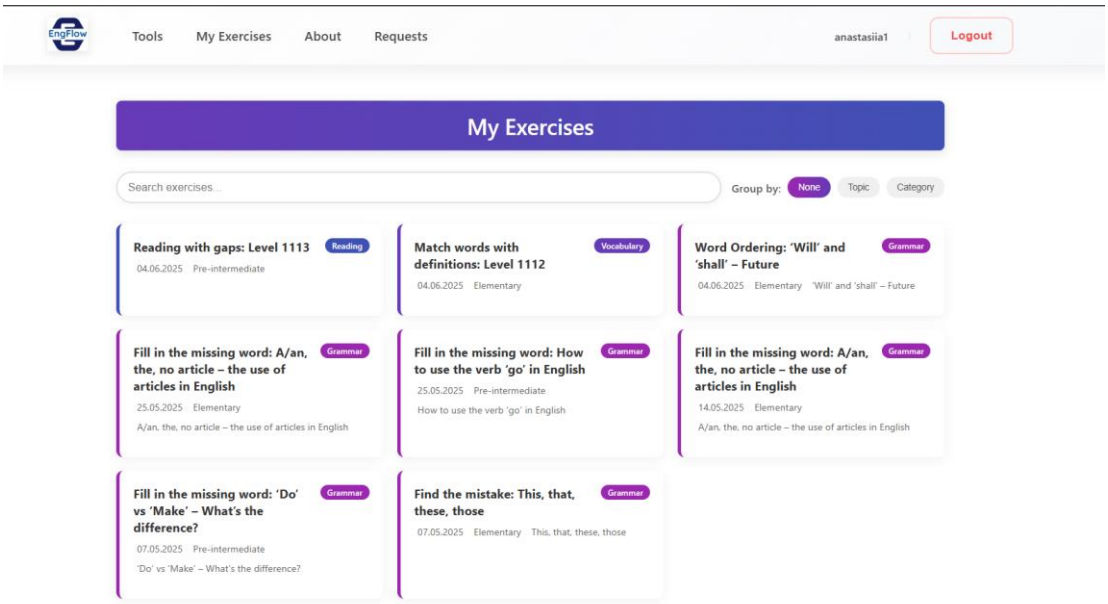


Рисунок 2.29 — Зовнішній вигляд сторінки збережених вправ

Інтерфейс надає кілька інтерактивних функцій (див. Рис. 2.30):

- пошук та фільтрація: користувачі можуть шукати вправи за назвою, змістом або темою;
- опції групування: вправи можуть динамічно групуватися за темою або категорією;
- розширювані картки: кожна вправа відображається як згортається картка, що дозволяє користувачам переглядати повний вміст за запитом;
- функція видалення: користувачі можуть безповоротно видаляти вправи. система підтверджує дію та надсилає запит на видалення до серверної частини через кінцеву точку /delete-exercise.

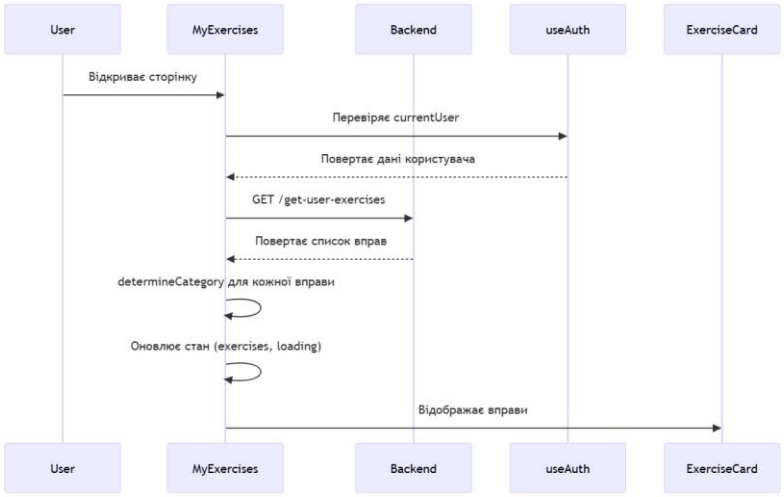


Рисунок 2.30 — Діаграма послідовності взаємодії компонентів при завантаженні вправ

Після доступу до сторінки запускається послідовність операцій для забезпечення безпечного та релевантного отримання даних. Спочатку система перевіряє ідентичність авторизованого користувача через контекст useAuth, який надає доступ до облікових даних поточного користувача. Якщо користувач автентифікований, система ініціює асинхронний запит до серверної кінцевої точки /get-user-exercises, передаючи захищений токен для перевірки. Серверна частина відповідає списком вправ, раніше збережених користувачем. Цей обмін проілюстровано на відповідній діаграмі послідовності.

Кожна отримана вправа проходить категоризацію на стороні клієнта за допомогою функції determineCategory(). Алгоритм аналізує назву кожної вправи та зіставляє її з попередньо визначеними наборами ключових слів, щоб віднести її до однієї з основних категорій: граматика, лексика, читання або інше.

2.7 Реалізація функціоналу чату зі штучним інтелектом

Однією з ключових функцій застосунку є інтерфейс чату на основі штучного інтелекту, що дає змогу користувачам інтерактивно практикувати англійську мову, спілкуючись із віртуальним помічником. Цей функціонал є важливим для імітації реальних діалогів, адаптації відповідей на основі рівня володіння мовою, вподобань щодо теми та обраної розмовної ролі користувача. Він забезпечує персоналізовані та захопливі практичні заняття, що значно сприяє вивченню мови та мотивації учнів.

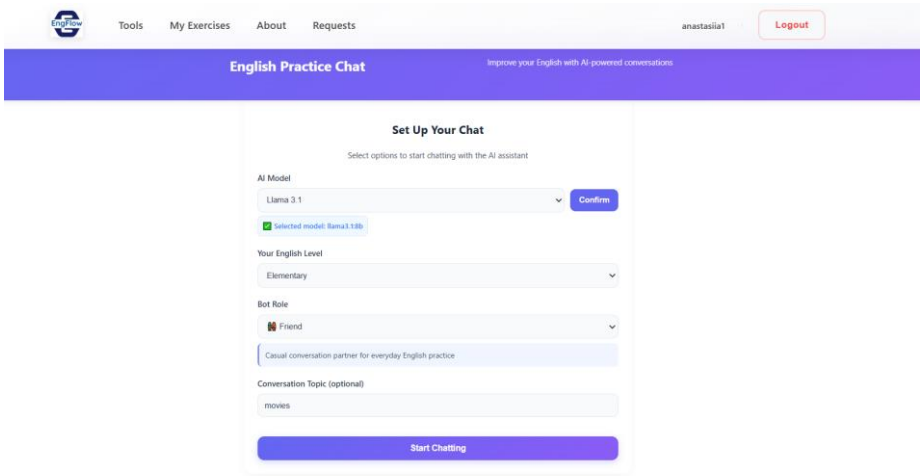


Рисунок 2.31 — Зовнішній вигляд сторінки чату із ШІ

Система чату реалізована за допомогою React на інтерфейсній частині та тісно інтегрована з внутрішньою частиною на Flask, яка обробляє запити природною мовою за допомогою обраних моделей штучного інтелекту. Користувацький інтерфейс має зрозумілий та доступний вигляд, що динамічно адаптується на основі введених користувачем даних та вибраних налаштувань.

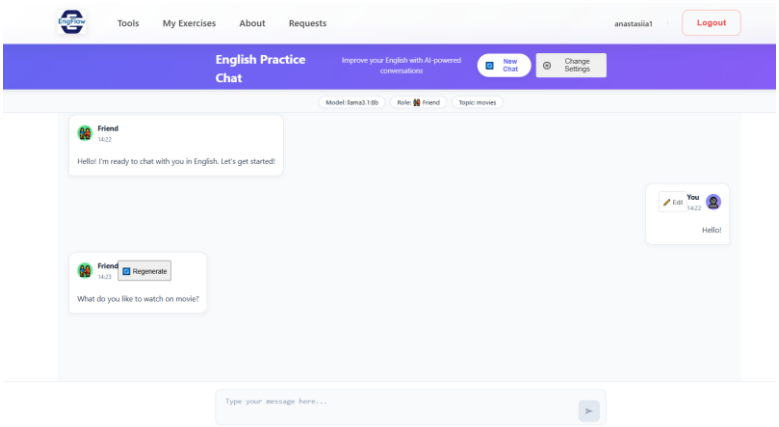


Рисунок 2.32 — Зовнішній вигляд сторінки чату із ШІ

Коли користувач отримує доступ до сторінки чату (компонент ChatPage), хук useEffect ініціює запит на отримання доступних рівнів англійської мови з бекенду (axios.get('http://localhost:5000/english-levels')). Після того, як користувач налаштовує параметри та починає сеанс чату за допомогою startChat(), генерується вступне повідомлення помічника, яке зберігається у стані conversation.

Основний цикл чату полягає у надсиланні повідомлення користувача та отриманні відповіді, згенерованої моделлю. Кожне повідомлення має часову мітку та додається до масиву conversation. Підказки для моделі генеруються динамічно на основі обраної ролі та рівня.

Додаткові функції користувача включають редагування попереднього повідомлення (editUserMessage), збереження відредагованих повідомлень та регенерацію відповіді помічника (saveEditedMessage), скидання сеансу (resetChat) та зупинку запиту під час виконання (handleStop).

Основні функціїї:

1. Вибір і підтвердження AI-моделі: дозволяє користувачу обрати модель штучного інтелекту для генерації відповідей.

2. Вибір рівня англійської мови: персоналізує відповіді ШІ відповідно до мовного рівня користувача.

3. Вибір ролі бота: визначає стиль і поведінку ШІ під час спілкування.

4. Введення теми розмови: дозволяє сконцентрувати діалог на конкретній темі чи сфері лексики.

5. Ініціалізація чату: створює стартове повідомлення бота та запускає діалог.

6. Відправлення повідомлення: дозволяє користувачу надсилати запит до ШІ й отримувати відповіді.

7. Редагування повідомлення користувача: дозволяє користувачу змінити раніше відправлене повідомлення.

8. Перегенерація останньої відповіді бота: оновлює останню відповідь ШІ у разі незадоволення нею.

9. Скасування відповіді (зупинка запиту): дозволяє користувачу зупинити генерацію відповіді ШІ.

10. Почати нову розмову / скинути налаштування: дозволяє очистити історію чату або повністю перезапустити сеанс.

11. Автоматичне прокручування вниз: Забезпечує автоматичну прокрутку до останнього повідомлення у чаті.

2.8 Впровадження функціоналу управління запитами

2.8.1 Розробка сторінки відправки нових запитів

Сторінка подання запитів на підтримку є функціональним компонентом програми, що дає змогу автентифікованим користувачам надсилати запити на підтримку системним адміністраторам. Ця сторінка слугує інтерфейсом зв'язку між користувачами та обслуговуючим персоналом системи, дозволяючи користувачам повідомляти про проблеми, ставити запитання або надавати відгуки. (Див. Рис. 2.33)

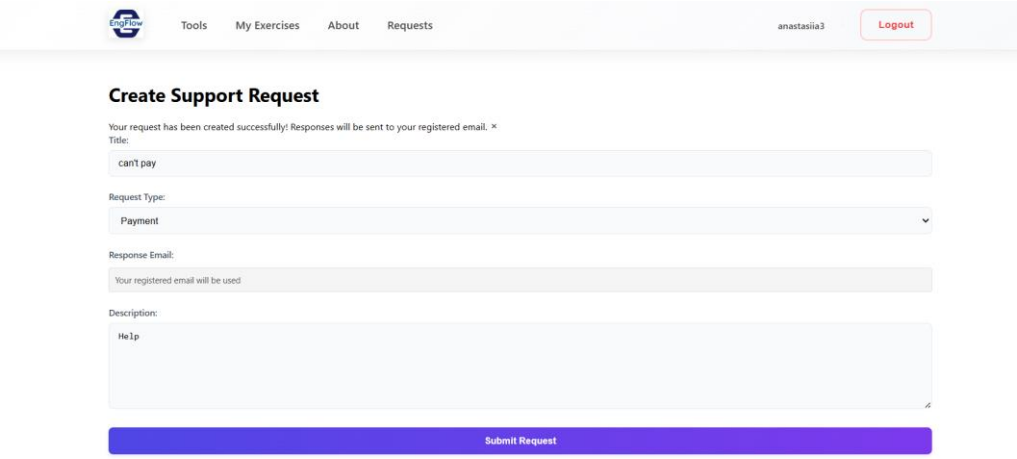


Рисунок 2.33 — Зовнішній вигляд сторінки надсилання запитів

Метою цієї сторінки є надання зареєстрованим користувачам можливості створювати структуровані запити на підтримку шляхом вибору заздалегідь визначеного типу запиту, надання короткої назви та детального опису проблеми. Ця функція необхідна для підтримки зв'язку між користувачами та адміністраторами, особливо при виникненні потреби в допомозі, повідомленні про помилки або проблемах, пов'язаних із контентом. Основна функціональність включає (див. Рис. 2.34):

- перевірку автентифікації користувача;
- отримання списку типів запитів з бекенду;
- відображення структурованої форми для подання запиту на підтримку;
- виконання валідації форми;
- обробку як успішних, так і неуспішних надсилань.

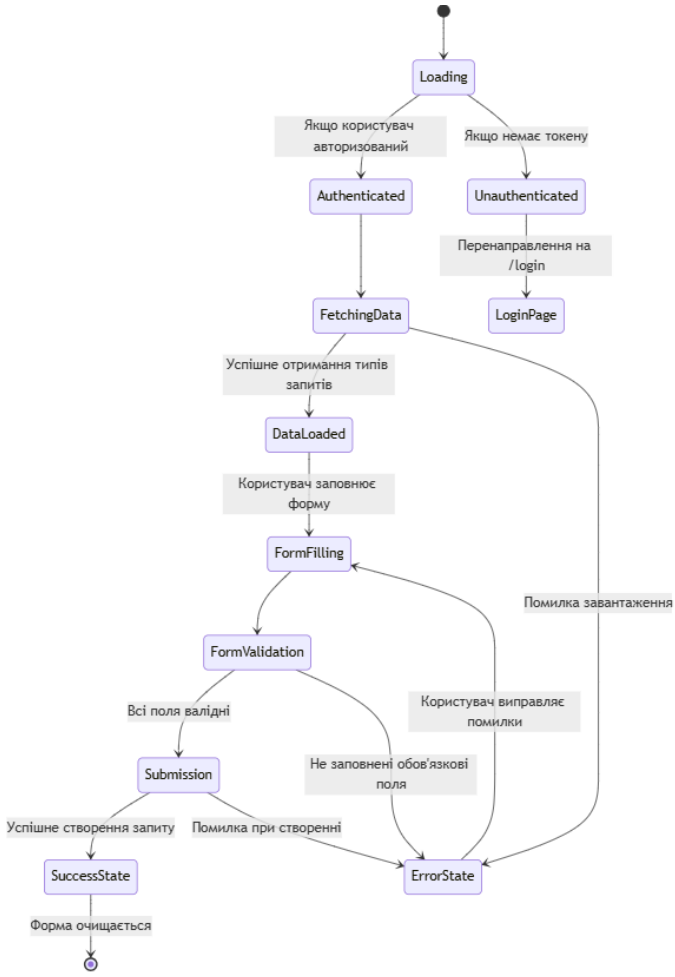


Рисунок 2.34 — Діаграма стану сторінки надсилання запитів

При монтуванні компонента система переходить у стан "Завантаження" (Loading). Спрацьовує хук useEffect для перевірки статусу автентифікації користувача за допомогою об'єкта currentUser з AuthContext та токена, що зберігається в localStorage.

Перевірка автентифікації: якщо користувач не автентифікований (!currentUser або відсутній токен), система переходить у стан "Неавтентифікований" (Unauthenticated), що викликає перенаправлення на сторінку входу (Maps('/login')).

Отримання типів запитів: якщо користувач автентифікований, система переходить у стан "Отримання даних" (FetchingData). Запит axios.get() до /api/request-types отримує список доступних категорій запитів із сервера. У разі успіху компонент переходить у стан "Дані завантажено" (DataLoaded), зберігаючи отримані дані у стані requestTypes.

Взаємодія з формою: у стані "Заповнення форми" (FormFilling) користувач вводить назву запиту, опис та вибирає type_id. Відповідні змінні стану (title, description, selectedType) оновлюються за допомогою обробників onChange, прив'язаних до елементів введення.

Валідація форми : При надсиланні форми (onSubmit на <form>) функція handleSubmit виконує валідацію. Якщо будь-яке обов'язкове поле порожнє, система переходить у стан "Стан помилки" (ErrorState), відображаючи повідомлення за допомогою функції setError().

Надсилання форми: якщо валідація успішна, система переходить у стан "Надсилання" (Submission). POST-запит надсилається на /api/create-request з даними форми та токеном. Корисне навантаження включає title, description та type_id. Токен передається в заголовку Authorization.

Обробка відповіді:

- у разі успіху (response.data.status === "success") сторінка переходить у стан "Стан успіху" (SuccessState). Поля форми очищаються, і відображається повідомлення про успіх за допомогою setSuccess().
- у разі помилки (наприклад, недійсний токен, проблема на сервері) система повертається у стан "Стан помилки" (ErrorState), і відображається повідомлення про помилку.

2.8.2 Реалізація адміністративної панелі для перегляду та керування запитами

Розробка сторінки керування запитами на підтримку була спрямована на надання адміністраторам централізованого інтерфейсу для моніторингу, оновлення та керування всіма запитами на підтримку, надісланими користувачами. (Див. Рис. 2.35).

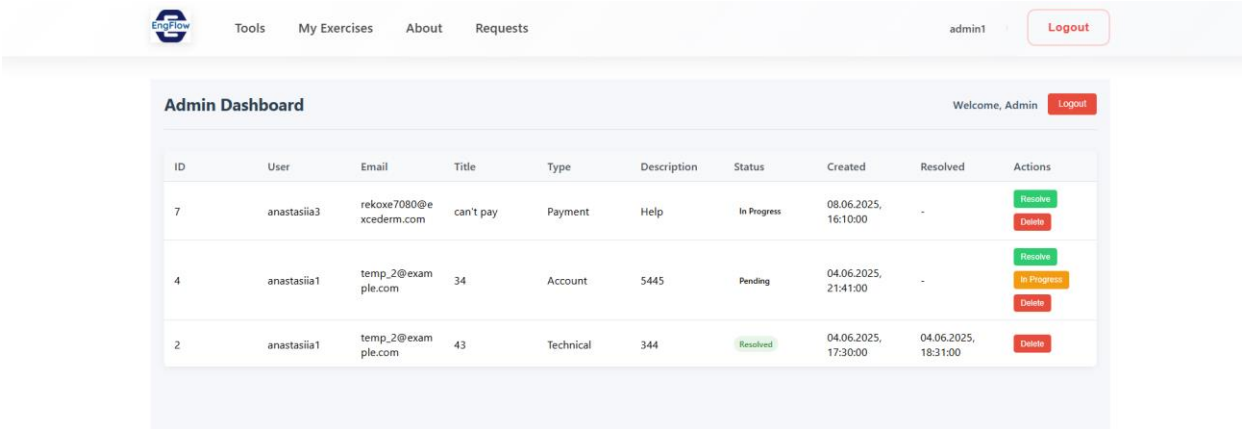


Рисунок 2.35 — Зовнішній вигляд сторінки перегляду запитів користувачів

Сторінка реалізована як React-компонент, який відповідає за отримання, відображення, оновлення та видалення записів запитів на підтримку. Компонент використовує хуки useState та useEffect для керування станом та побічних ефектів відповідно. Він також застосовує користувацький контекст useAuth для обробки автентифікації та перевірки сесії, забезпечуючи доступ до інформаційної панелі лише авторизованим адміністраторам.

Після монтування хук useEffect перевіряє наявність adminSession. За його відсутності користувач перенаправляється на сторінку входу адміністратора за допомогою функції Maps з react-router-dom. Це забезпечує безпеку та запобігає несанкціонованому доступу до конфіденційних адміністративних функцій.

Основний набір даних отримується за допомогою асинхронної функції fetchRequests, яка надсилає GET-запит до кінцевої точки /admin/support-requests за допомогою axios. Успішна відповідь заповнює стан requests списком запитів на підтримку, які потім відображаються в таблиці у блоці повернення JSX.

Кожен запит відображається з основними полями метаданих, такими як id, username, email, title, type, status, created_at та resolved_at. Дані можуть бути відсортовані за будь-яким із цих полів за допомогою функції handleSort, яка перемикає напрямок сортування та встановлює активний ключ сортування в sortConfig. Відсортовані дані меморизуються за допомогою React.useMemo для оптимізації продуктивності.

Вигляд таблиці запитів у SQL (див. Рис. 2.36):

RequestID	UserID	RequestTypeID	Title	Description	Status	CreatedAt	ResolvedAt	AdminNotes	Email
2	2	1	43	344	Resolved	2025-06-04 17:3...	2025-06-04 18:3...	Changed to Res...	user@example....
4	2	2	34	5445	Pending	2025-06-04 21:4...	NULL	NULL	anastasa26akim...
7	3	3	can't pay	Help	In Progress	2025-06-08 16:1...	NULL	Changed to In ...	rekoxe7080@ex...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.36 — Таблиця запитів вигляд у SQL

2.9 Практична апробація та опис методики використання інформаційної системи

Інформаційна система, реалізована у межах даного проєкту, була апробована у тестовому середовищі з метою оцінки її функціональних можливостей та зручності використання. Основною цільовою аудиторією системи є викладачі та учні, які займаються вивченням англійської мови на різних рівнях.

Після відкриття сайту користувач має змогу розпочати роботу без обов'язкової реєстрації, проте функція реєстрації доступна через відповідний пункт меню.

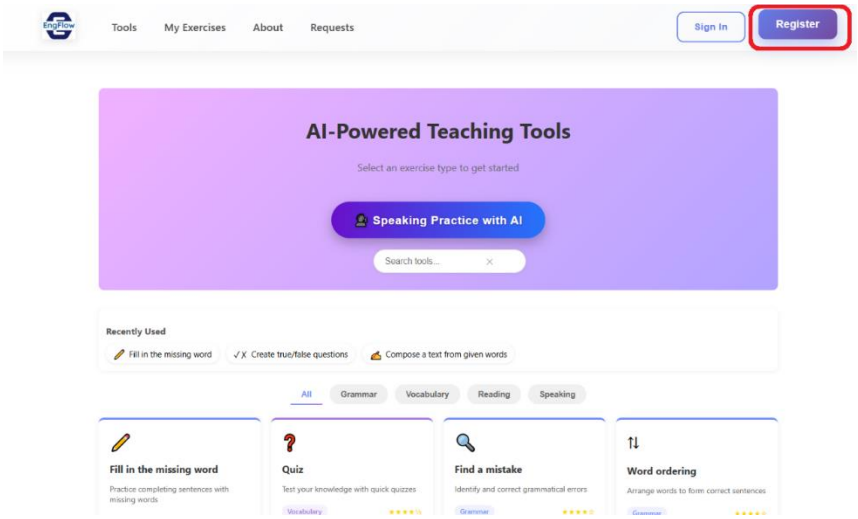


Рисунок 2.37 — Зовнішній вигляд головної сторінки сайту

Для реєстрації необхідно вказати ім'я користувача, електронну адресу та пароль, який необхідно підтвердити.

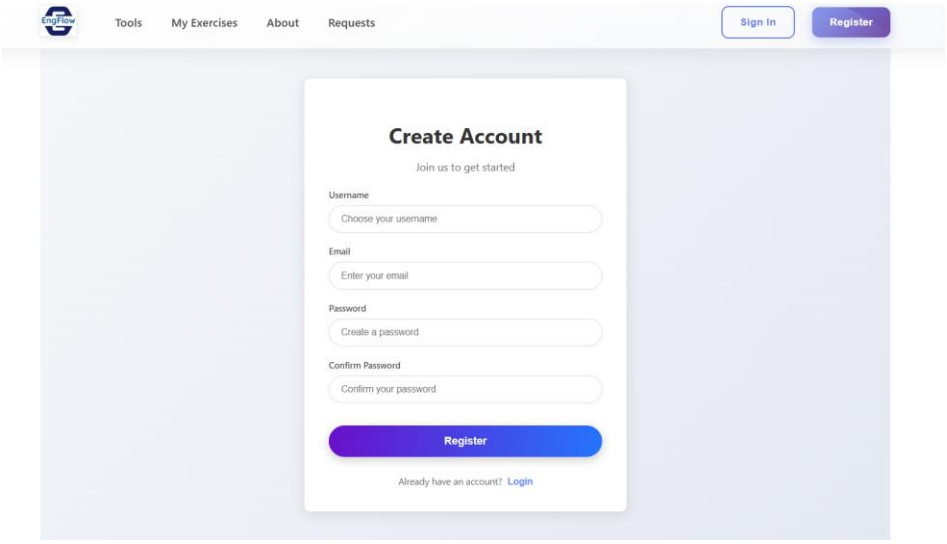


Рисунок 2.38 — Зовнішній вигляд сторінки реєстрації користувача

Зареєстровані користувачі отримують розширений функціонал, а саме:

- збереження створених вправ у персональному кабінеті;
- можливість надсилання запитів до адміністрації;
- перегляд та управління власними вправами (видалення, редагування, сортування);
- редагування відповідей, згенерованих штучним інтелектом, з можливістю їх подальшого збереження.

Незареєстровані користувачі також можуть створювати вправи та взаємодіяти з чат-ботом, проте результати таких сесій не зберігаються.

На головній сторінці користувачу пропонується обрати тип взаємодії відповідно до освітніх потреб. Доступні наступні опції:

граматичні завдання;

лексичні завдання;

завдання з читання;

чат із ботом.

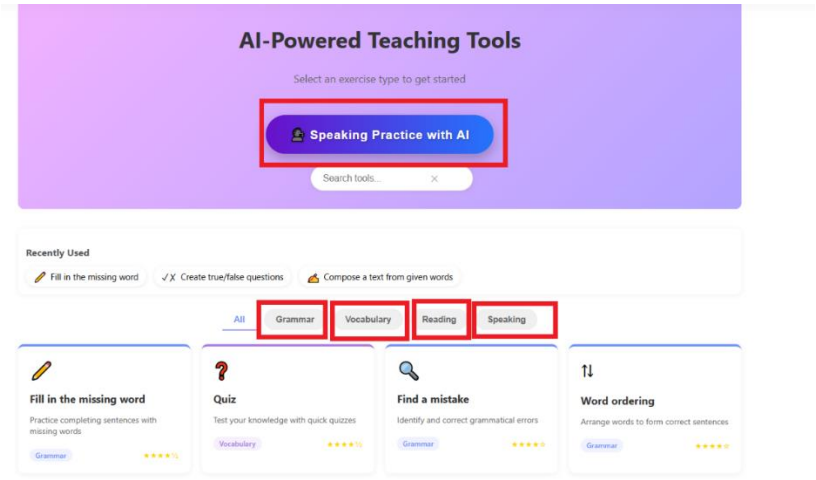


Рисунок 2.39 — Опції головної сторінки сайту

Після вибору конкретного типу вправи користувач може налаштувати наступні параметри:

- модель штучного інтелекту (наприклад, DeepSeek, MISTRA, LAMA);
- рівень володіння англійською мовою;
- тематичні параметри (наприклад, граматичну тему, словник або текст для читання).

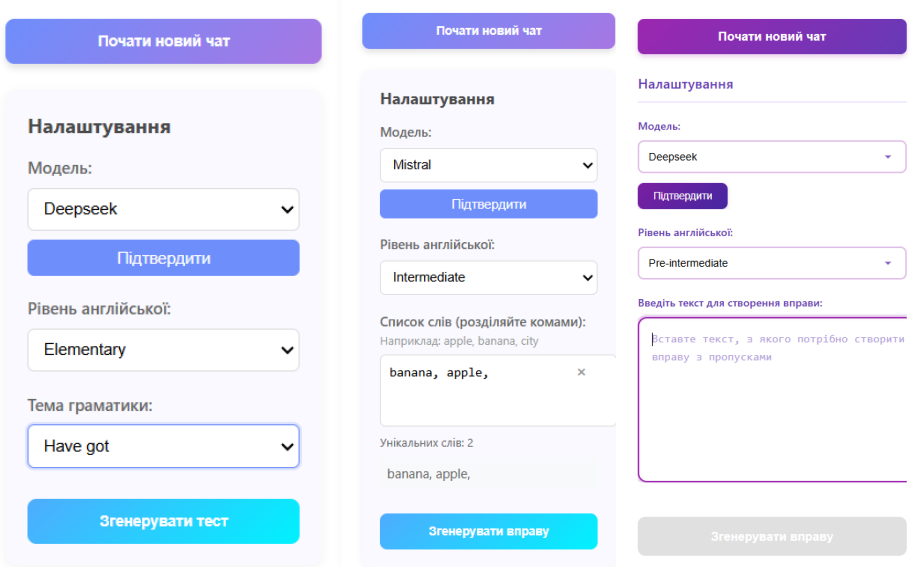


Рисунок 2.40 — Доступні налаштування для різних типів завдань

Функції налаштування перед початком чату:

Вибір моделі ШІ (DeepSeek, Mistral, Llama 3.1 тощо) – користувач обирає, яка модель відповідатиме в чаті.

Вибір рівня володіння англійською мовою – наприклад, A1, B2 тощо (підтягується з БД).

Вибір ролі бота – користувач може обрати тип співрозмовника (вчитель , друг , гід , співрозмовник для інтерв'ю тощо), що визначає стиль відповідей ШІ.

Введення теми розмови – опціонально можна задати тему (наприклад, «подорожі», «робота»), щоб звузити контекст спілкування.

Початок чату – активується лише після вибору всіх обов'язкових параметрів.

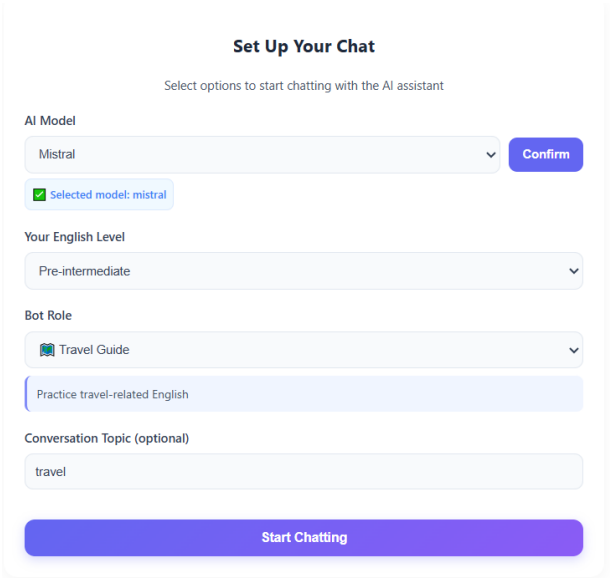


Рисунок 2.41 — Налаштування сторінки чату із ШІ

Відповіді штучного інтелекту, що генеруються для граматичних, лексичних або завдань з читання, можуть бути відредаговані користувачем у разі виявлення помилок.

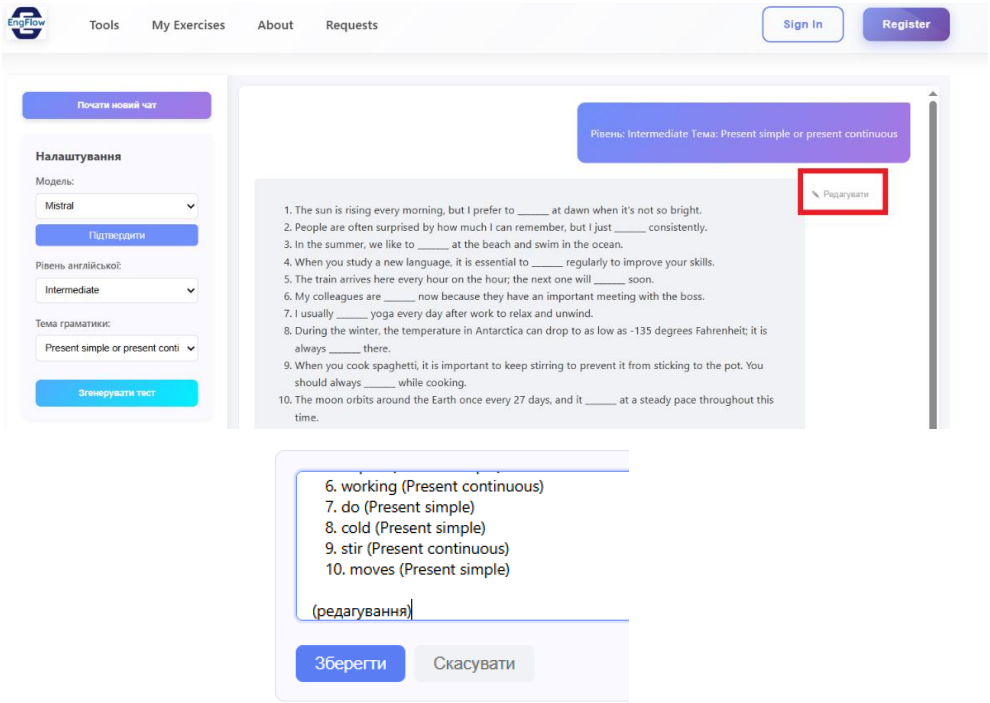


Рисунок 2.42 — Опції чатів лексики, граматики та читання

Редаговані вправи можна зберегти у власному профілі. Зберігання доступне лише для авторизованих користувачів. Відповідь бота можна перегенеровувати задля отримання найкращого результату.

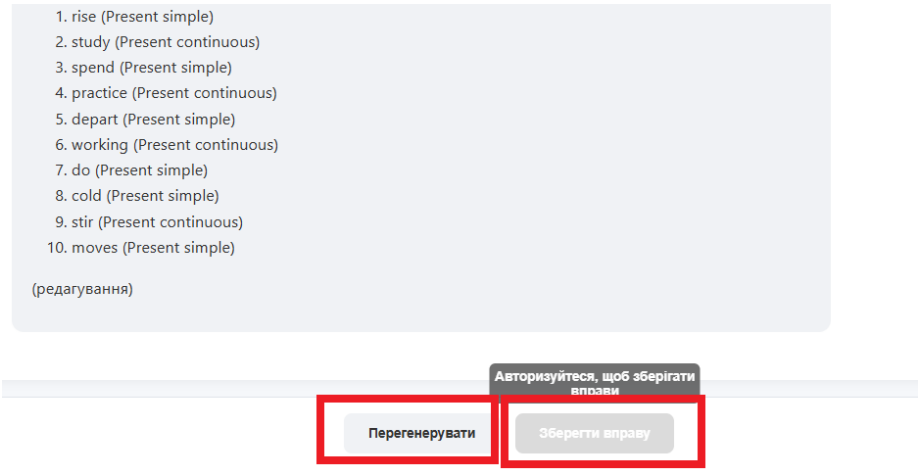


Рисунок 2.43 — Опції чатів лексики, граматики та читання

Розмови в чаті з ботом не зберігаються, проте користувач може розпочинати нові бесіди необмежену кількість разів, обираючи нові теми, ролі та мовні рівні.

Функції взаємодії з ботом:

- надсилання повідомлень – користувач може спілкуватися з ботом, як у звичайному чаті;
- отримання відповідей від ШІ – відповіді генеруються на основі історії чату, ролі бота, теми та рівня мови;
- markdown-відображення – повідомлення можуть містити таблиці або інший форматований текст.

Функції редагування:

- редагування власного повідомлення – користувач може змінити своє попереднє повідомлення;
- автоматична регенерація відповіді бота – після редагування повідомлення ШІ оновлює відповідь відповідно до нового контексту;
- повторне генерування останньої відповіді ШІ – якщо відповідь не задовольняє, користувач може її перегенерувати.

Додаткові функції під час чату:

- перезапуск чату – очищення історії діалогу та повернення до етапу налаштування;
- зміна параметрів (моделі, ролі тощо) – можна змінити налаштування вже після початку чату;
- відміна запиту – користувач може скасувати довгий або завислий запит до сервера.

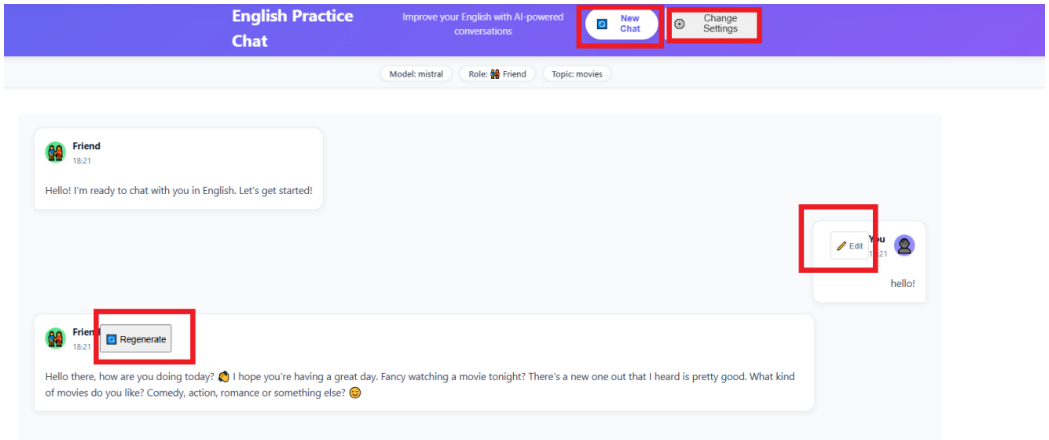


Рисунок 2.44 — Можливі опції та налаштування чату із ШІ

Висновки до розділу:

– Обґрунтовано вибір сучасного технологічного стеку (React, Python, SQL, Ollama), що забезпечив ефективну реалізацію клієнтської та серверної частин системи, гнучкість розробки та інтеграцію зі штучним інтелектом

– Спроєктовано й реалізовано повноцінну архітектуру інформаційної системи: базу даних, серверну логіку, інтерфейс користувача, механізми маршрутизації та автентифікації.

– Упроваджено основні функціональні модулі — генерація навчальних завдань, чат зі ШІ, збереження вправ, адміністративне керування — що забезпечують практичну цінність для викладачів у створенні персоналізованих матеріалів.

ВИСНОВКИ

Виконана робота присвячена розробці інтерактивного веб-додатка — помічника викладача іноземної мови, який використовує інтелектуальні агенти для автоматичного створення навчальних завдань і організації діалогового спілкування. У ході дослідження і реалізації проєкту було вирішено низку важливих завдань, що стосуються як аналізу предметної області, так і розробки технічної частини системи.

Було проведено детальний аналіз проблем, з якими стикаються викладачі при створенні персоналізованих навчальних матеріалів, а також вивчено можливості автоматизації цих процесів із використанням сучасних штучних інтелектів. Проведений огляд існуючих платформ і технологій дав змогу обґрунтувати вибір архітектури додатка та необхідного набору інструментів, зокрема React для клієнтської частини, Python (Flask) для серверної логіки, MSSQL для зберігання даних, а також локальну інтеграцію з моделями штучного інтелекту через Ollama.

Особливу увагу було приділено проєктуванню бази даних, що дозволяє ефективно зберігати інформацію про користувачів, їхні вправи, теми та рівні володіння мовою. Розроблені логічні моделі і налаштовані механізми підтримки цілісності даних забезпечують стабільність і надійність системи. Серверна частина інтегрувала різноманітні функціональні компоненти, включаючи механізми аутентифікації, обробку запитів до ШІ, збереження і керування створеними завданнями.

На рівні клієнтського інтерфейсу було реалізовано зручну навігацію між основними розділами — граматикою, лексикою, читанням та чатом із штучним інтелектом. Інтерфейси відповідають принципам інтуїтивності та сучасного дизайну, містять анімації, які покращують користувацький досвід викладачів. Окремо було реалізовано механізми вибору моделі ШІ, рівня знань, тематики та типу завдань, що дозволяє гнучко адаптувати навчальний матеріал до індивідуальних потреб.

Значним результатом стало впровадження інтерактивного чату, де викладач може вести діалог із штучним інтелектом у різних ролях і тематиках, що відкриває нові можливості для розвитку комунікативних навичок. Функціонал чату також передбачає редагування відповідей, перегенерацію та початок нових сесій, що підвищує якість взаємодії.

Практична апробація системи підтвердила ефективність реалізованих рішень і їхню відповідність поставленим вимогам. Розроблений комплексний підхід до інтеграції локальних моделей ШІ, збереження даних та інтуїтивно зрозумілого інтерфейсу сприяє автоматизації процесу підготовки навчальних матеріалів і зменшує навантаження на викладача.

Отже, виконана робота має вагоме прикладне значення для сучасної освітньої сфери, оскільки створена інформаційна система дозволяє підвищити продуктивність, якість і персоналізацію навчального процесу за допомогою інтелектуальних технологій. Надалі передбачено розширення функціоналу, оптимізацію моделей ШІ та адаптацію системи під різні мови та освітні стандарти, що зробить її ще більш універсальним і корисним інструментом для викладачів іноземних мов.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Akash Takyar. AI agents in research: applications & use cases, key components, capabilities, benefits and implementation. *LeewayHertz*. URL: https://www.leewayhertz.com/ai-agents-in-research/ (дата звернення: 07.04.2025).

2. Amalia Maria Fyka. Artificial intelligence in foreign language education. *EJ-ENG European Journal of Engineering and Technology Research*. URL: https://www.ej-eng.org/index.php/ejeng/article/view/3236 (дата звернення: 06.04.2025).

3. Angelito Calma, Valeria Cotronei-Baird, Austin Chia. Grammarly: an instructional intervention for writing enhancement in management education. *The international journal of management education*. 2022. Т. 20, № 3. 100704. URL: https://doi.org/10.1016/j.ijme.2022.100704. (дата звернення: 10.04.2025).

4. Antonietta Gerarda Gravina Raffaele Pellegrino Giovanna Palladino Giuseppe Imperio Andrea Ventura Alessandro Federico. Charting new AI education in gastroenterology: Cross-sectional evaluation of ChatGPT and perplexity AI in medical residency exam. *Digestive and liver disease*. 2024. Vol. 56, no. 8. P. 1304–1311. URL: https://doi.org/10.1016/j.dld.2024.02.019. (date of access: 10.04.2025).

5. Conversational agents in language learning / Feiwen Xiao та ін. *Journal of china computer-assisted language learning*. 2024. Т. 4, № 2. С. 300–325. URL: https://doi.org/10.1515/jccall-2022-0032 (дата звернення: 07.04.2025).

6. Describing the UI. *React*. URL: https://react.dev/learn/describing-the-ui (дата звернення: 01.05.2025).

7. Duolingo Team. Introducing Duolingo Max, a learning experience powered by GPT-4. *blog*. URL: https://blog.duolingo.com/duolingo-max/ (дата звернення: 10.04.2025).

8. Gerda Urbaite. Adaptive learning with AI: how bots personalize foreign language education. *Luminis applied science and engineering*. 2025. С. 13–18.

9. Jerry Registre. What's the BEST AI For Language Learning? (CLEAR winner), 2023. *YouTube*. URL: https://www.youtube.com/watch?v=kXChJfHZrjU&ab_channel=JerryRegistre (дата звернення: 09.04.2025).

10. Rahman M. M., Yutaka Watanobe. ChatGPT for education and research: opportunities, threats, and strategies. *Applied sciences*. 2023. Т. 13, № 9. URL: https://doi.org/10.3390/app13095783 (дата звернення: 10.04.2025).

11. Tahir. What is ollama: running large language models locally. *Medium*. URL: https://medium.com/@tahirbalarabe2/what-is-ollama-running-large-language-models-locally-e917ca40defe (date of access: 03.05.2025).

12. The #1 AI meeting agent. *Otter.ai*. URL: https://otter.ai/ (дата звернення: 10.04.2025).

13. The impact of ai-powered language learning tools on second language acquisition: a mixed-methods study. *International journal of linguistics literature & translation*. 2025. С. 270–271.

14. What flask is, what flask is not. *Flask*. URL: https://flask.palletsprojects.com/en/stable/design/#what-flask-is-what-flask-is-not (дата звернення: 02.05.2025).

15. What is a SQL database?. *SOLARWINDS*. URL: https://www.solarwinds.com/resources/it-glossary/sql-database (дата звернення: 05.05.2025).

16. Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для студентів спеціальності 122 “Комп’ютерні науки”. Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.

17. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

18. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

19. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

20. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).