МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
(підпис)
грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-практичної програми «Інформатика»
на тему: «Інформаційна технологія створення інтерактивних інтелектуальних асистентів навчального призначення»
здобувача групи ІН.м-34 Алексенка Артемія Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Артемій АЛЕКСЕНКО
(підпис)

Керівник,
кандидат технічних наук, доцент
Борис КУЗІКОВ
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-практичної програми «Інформатика»
здобувача групи ІН.м-34 Алексенка Артемія Сергійовича

1. Тема роботи: «Інформаційна технологія створення інтерактивних інтелектуальних асистентів навчального призначення»
затверджую наказом по СумДУ від 03.12.2024 року №1257-VI
2. Термін здачі здобувачем кваліфікаційної роботи *до 6 грудня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки інтерактивних інтелектуальних асистентів. 3) Розробка інтерактивного інтелектуального асистента навчального призначення. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка завдань*		
2	*Огляд технологій, що використовуються для розробки інтерактивних інтелектуальних асистентів*		
3	*Розробка інтерактивного інтелектуального асистента навчального призначення*		
4	*Аналіз отриманих результатів*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 55 стр., 42 рис., 2 додатки, 24 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої задачі забезпечення ефективного доступу до навчальних матеріалів шляхом розробки інтерактивного інтелектуального асистента для освітніх платформ. Зростання попиту на IT-освіту та потреба у персоналізованих навчальних ресурсах, здатних адаптуватися до інтересів і рівня знань користувачів, підкреслюють значущість даного дослідження.

Об’єкт дослідження — процес створення інтерактивних інтелектуальних асистентів навчального призначення.

Мета роботи — розробка інтерактивного інтелектуального асистента навчального призначення, який забезпечує ефективний доступ до персоналізованих рекомендацій і підтримку користувача у навчальному процесі.

Методи дослідження — аналітичний огляд існуючих аналогів, сучасні підходи до реалізації чат-ботів на основі штучного інтелекту, алгоритми обробки запитів користувача, а також нотації і діаграми для моделювання бізнес-процесів.

Результати — розроблено інтерактивного інтелектуального асистента, що має можливість аналізувати запити користувачів, пропонувати навчальні матеріали відповідно до інтересів і рівня знань, забезпечувати швидкий доступ до бази знань та підтримувати користувача без залучення людини.

ІНТЕРАКТИВНИЙ АСИСТЕНТ, НАВЧАЛЬНА ПЛАТФОРМА, ШТУЧНИЙ ІНТЕЛЕКТ, ЧАТ-БОТ, АНАЛІЗ ДАНИХ, ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ, PYTHON, FLASK, LANG CHAIN, PINECONE.

ЗМІСТ

ВСТУП

Актуальність роботи. В епоху стрімкого розвитку інформаційних технологій спостерігається значне підвищення попиту на нових IT-спеціалістів, внаслідок чого зростає кількість бажаючих опанувати дану професію або поглибити вже існуючі навички. Дані тенденції зумовлені світовою діджиталізацією, стрімким розвитком глобального ринку IT праці та можливістю кар'єрного розвитку не залежно від місцезнаходження фахівця. Саме тому, важливим завданням для сучасної освіти стало забезпечення якісних та зручних у використанні навчальних ресурсів.

Останніми роками зросла потреба в оптимізації пошуку навчальних матеріалів з використанням інтелектуальних інструментів, здатних персоналізувати освітній процес. Сучасні асистенти на базі штучного інтелекту можуть аналізувати запити користувачів і пропонувати курси відповідно до інтересів та рівня знань, що робить процес їх підбору більш продуктивним та індивідуальним.

Інтеграція інтерактивного інтелектуального асистента на освітніх платформах надає ряд переваг, таких як швидкий та зручний доступ до широкої бази знань, миттєвий зворотній зв'язок та підтримку без залучення людини.

Об'єкт дослідження – процес створення інтерактивних інтелектуальних асистентів навчального призначення.

Предмет дослідження – методи та інструменти для створення інтерактивного інтелектуального асистента навчального призначення, що інтегрований у веборієнтовану систему навчальних курсів.

Новизна. У порівняні з існуючими аналогами інтерактивних асистентів, інформаційна технологія, що буде розроблена у рамках даної навчальної роботи дозволить не лише персоналізувати пошук і вибір найоптимальніших навчальних матеріалів, але й зберігати контекст та історію комунікації з користувачем, що надає можливість формувати більш точні рекомендації,

опираючись на попередній досвід спілкування зі студентом.

Структура роботи. У першому розділі виконаний аналітичний огляд існуючих аналогів за обраною тематикою. У другому розділі розглянуті сучасні підходи до реалізації чат-ботів, а також виконаний огляд необхідних фреймворків та хмарних інструментів. У третьому розділі реалізоване проектування системи. У четвертому розділі описується процес імплементації серверної та клієнтської частин чат-боту. У фінальному розділі виконаний огляд можливостей реалізованого інтелектуального асистента із подальшим формуванням висновку та списку використаних джерел.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Поширення інтелектуальних асистентів в онлайн-освіті

За останні десять років кількість людей, що навчаються онлайн різко зросла і цей процес продовжує набирати обертів. Так у 2012 році 75% студентів відвідували всі заняття особисто, 13,1% відвідували деякі онлайн-заняття, а 12,4% студентів навчалися повністю в онлайн форматі. Ці цифри різко змінилися після пандемії Covid-19: лише 26,6% студентів у 2020 році повідомили, що вони не відвідували онлайн-занять [1].

Попри завершення піку пандемії та повернення більшості навчальних закладів до традиційного формату навчання, очікуваного значного скорочення обсягів використання онлайн-платформ не відбулося. Наразі понад 60% студентів за можливості обирають онлайн-заняття та/або користуються сторонніми освітніми платформами [2].

Варто зазначити, що онлайн-учні становлять гетерогенну групу, що включає студентів бакалаврату та аспірантів, осіб різної статі, а також тих, хто здобуває першу професію, і тих, хто прагне перекваліфікуватися. Саме тому кожному студенту необхідна індивідуальна консультація у виборі відповідних навчальних матеріалів. При цьому критеріїв, що впливають на цей вибір – велика кількість, як наслідок виникає необхідність у використанні інтерактивного асистента на базі штучного інтелекту.

Зважаючи на актуальність теми, інтерес до використання ШІ-асистентів невпинно зростає перш за все серед науковців. Такі висновки ми зробили, провівши дослідження на основі найбільшої бібліографічної та реферативної бази даних Scopus в розрізі предметної області – «Інформатика» (рис.1.1). Дані проведеного аналізу показали тенденцію до збільшення наукових робіт з тематики «AI assistant». Так ще у 2018 році кількість робіт на дану тематику сягала лише 8 шт., проте на 2024 рік кількість робіт зросла приблизно у 15 разів і сягнула свого максимуму – 118 [3].

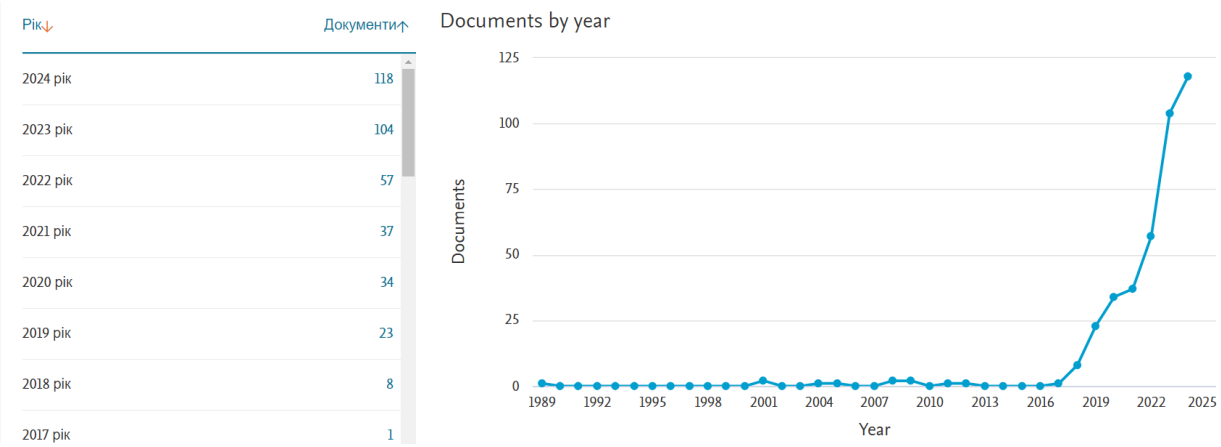

Рисунок 1.1 – Щорічна кількість публікацій на тему «AI assistant» в бібліографічній базі даних Scopus.

Крім того, американське опитування від Aiprm в якому прийняло участь більше 6 тисяч осіб показало, що найпопулярнішим форматом використання штучного інтелекту в особистих цілях стали саме віртуальні асистенти (рис.1.2). Приблизно три з п’яти людей (61,4%) стверджували, що використовують цю форму штучного інтелекту [4].

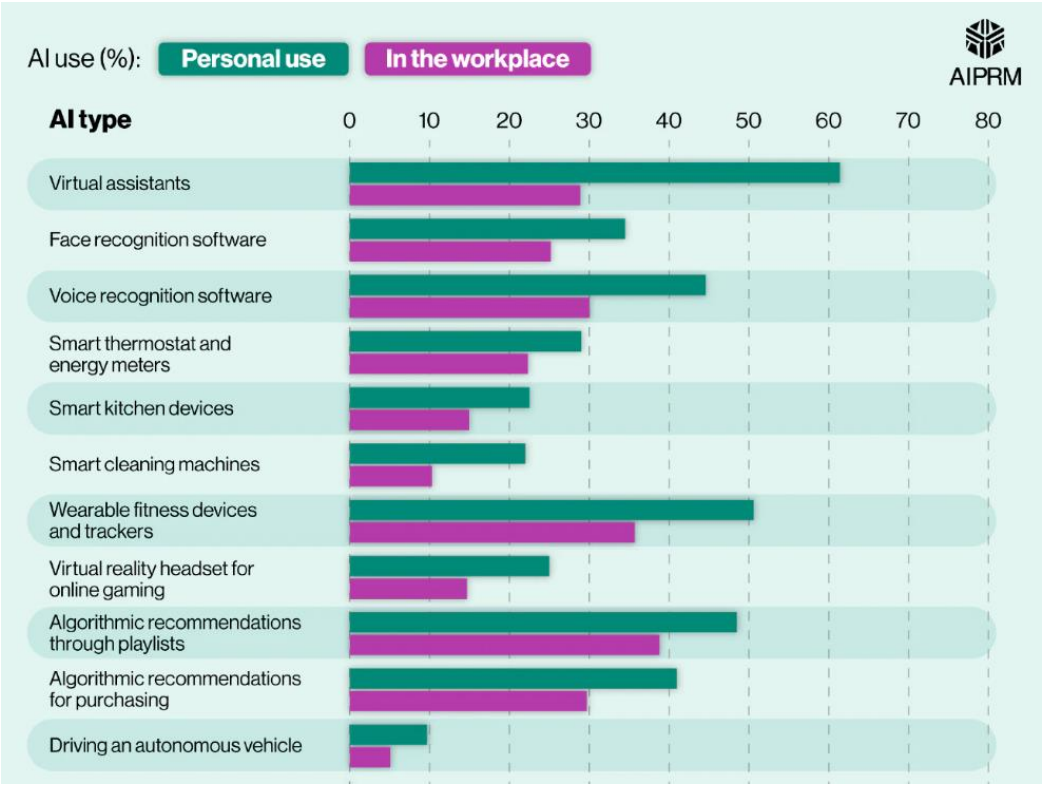

Рисунок 1.2 – Використання різних типів штучного інтелекту в робочих та особистих цілях.

Отже, розвиток інтерактивних інтелектуальних асистентів навчального призначення є комплексним рішенням на сучасні виклики освітньої сфери. Різке зростання кількості навчальних онлайн-платформ і індивідуальність студентської авдиторії вимагають нових підходів для забезпечення персоналізованих рекомендацій та адаптації навчального процесу. Оскільки сучасні технології штучного інтелекту надають великий вибір інструментів та підходів – наступним важливим кроком у нашому дослідженні є аналіз методів розробки інтелектуальних асистентів навчального призначення на основі існуючих аналогів.

1.2 Аналіз методів розробки асистентів на базі ШІ на прикладі існуючих аналогів

Реалізація інтерактивних інтелектуальних асистентів для навчальних платформ на основі штучного інтелекту полягає у використанні сучасних методів та технологій, завдяки яким ми маємо можливість забезпечити індивідуальний підхід до користувача та покращити якість навчання. Розглянемо сутність, види та ключові компоненти інтелектуального асистента.

ШІ-асистент – це тип програмного забезпечення із застосуванням технології штучного інтелекту, що використовується для надання допомоги в форматі взаємодії, властивій людині [5].

Із розвитком можливостей самого штучного інтелекту зростає і кількість типів віртуальних помічників, побудованих на його основі. Окремі з них взаємодіють з людиною виключно через голосовий інтерфейс, інші ж інтегруються в середовище розробки (IDE) для покращення програмного коду, тоді як деякі можуть бути приховані від користувача задля внутрішньої оптимізації пристроїв чи застосунків. У рамках нашого дослідження ми зосередимось на реалізації інтелектуального асистента, інтегрованого у навчальну вебплатформу, у форматі текстового чат-боту.

ШІ чат-боти – це комп’ютерні програми, які служать віртуальними помічниками та спілкуються з користувачами через текстові інтерфейси на вебплатформах, у соціальних мережах і програмах обміну повідомленнями. Дані чат-боти можуть допомагати клієнтам, відповідати на запити або починати з ними дискусію.

Чат-боти зі штучним інтелектом використовують алгоритми обробки природної мови (NLP) і машинного навчання (ML), щоб розуміти введені користувачем дані, виробляти доречні відповіді та з часом покращувати свою продуктивність, навчаючись на цих взаємодіях [6]. Зазвичай вони допомагають кінцевим користувачам, виконуючи наступні задачі:

- формування рекомендацій на основі заданих вподобань користувача;
- впровадження індивідуального плану розвитку за обраною тематикою;
- надання інформації, яку зазвичай доводилось шукати у веббраузері;
- автоматизація процесів для певних сервісів (пошук та бронювання готелів, створення нагадувань, прокладання маршрутів тощо).

Для побудови інтелектуальних асистентів необхідно розглянути ключові технології, що використовуються при створенні програмного забезпечення на основі штучного інтелекту.

Нейронна мережа – основа, на якій базується ШІ, особливо у завданнях розпізнавання шаблонів та пода льшого вдосконалення моделей. Ця технологія дозволяє приймати рішення подібно до людського мозку, використовуючи процеси, які імітують те, як біологічні нейрони працюють разом, щоб ідентифікувати явища, зважувати варіанти та робити висновки [7].

Особливу увагу слід звернути на методи та дані на основі яких ШІ буде давати відповіді. Оскільки на даному етапі нейронна мережа немає жодного уявлення про контекст подальшої комунікації. Для вирішення цього питання розглянемо наступну технологію.

Мовна модель – це спеціалізований вид нейронної мережі, що була

навчена на великій кількості даних, аби розуміти людську мову, генерувати текст та давати відповіді на запити користувачів. Основним завданням мовної моделі є аналіз контексту діалогу, генерації персоналізованих відповідей на основі отриманих даних та подальше вибудовування комунікації.

Наступним кроком є використання сформованої мовної моделі для реалізації чат боту.

Чат-бот представляє собою фінальний інтерфейс задля комунікації кінцевого користувача з навченою мовною моделлю. Сучасні чат-боти на базі штучного інтелекту використовують розуміння природної мови та підготовлений набір даних, щоб розпізнавати значення введеного запита від користувача, долаючи будь-які помилки передачі контексту. Розширені інструменти штучного інтелекту потім відображають це значення для конкретного «наміру», на який користувач очікує відповідну реакцію від чат-бота, з подальшим формуванням необхідної відповіді [8].

Варто зазначити, що повномасштабне проектування кожної вищезазначеної технології – є надскладним та ресурсоємним процесом, що вимагає значних обсягів даних, часу для навчання моделей та використання великих обчислювальних можливостей [9]. Яскравим прикладом може слугувати поточна ситуація в компанії-лідера з розробки штучного інтелекту – Microsoft. Голова відділу штучного інтелекту Microsoft Мустафи Сулеймана зазначає: «Асистенти зі штучним інтелектом із справді хорошою довгостроковою пам’яттю з’являться приблизно через рік» [10].

Тому у нашій роботі реалізація буде побудована на інтеграції з вже навченою моделлю.

Для більш глибокого розуміння функціональних можливостей чат-ботів та способів їх інтеграції у вебплатформи пропонуємо розглянути декілька існуючих аналогів.

Для порівняння було обрано три найпопулярніших ресурси, котрі впровадили на своїй платформі інтелектуальних асистентів, зокрема Intercom, Livechat, Custom GPT.

Дані аналізу, що був проведений представлено в таблиці 1.1

Критерій	Intercom	Livechat	CustomGPT
Підтримка української локалізації	+	-	+
Можливість ескалації до оператора	-	+	+
Захист конфіденційності та безпеки даних	+	+	-
Вдало інтегрований у дизайн платформи	+	+	-
Інтуїтивно зручний у використанні	-	+	+
Відповіді чітко відповідають темі поставленого запитання	+	-	+
Збереження контексту діалогу	+	-	+
Відсутність лімітів на комунікацію	+	+	+
Швидкість генерації відповідей	+	-	+
Гнучкість сценаріїв відповіді	+	-	+
Використання власних мовних моделей.	+	+	-

**Таблиця складена автором на основі сайтів з інтегрованими асистентами [11;12;13].*

Таблиця 1.1 – Порівняльний аналіз чат-ботів на базі штучного інтелекту, інтегрованих у вебплатформи

На основі проведеного аналізу можна зробити кілька важливих висновків.

По-перше, кожен з чат-ботів має певні переваги в інтеграції дизайну, що дасть користувачам можливість легко розпочати взаємодію з асистентами, проте візуальне відображення чат-бота у CustomGPT є занадто великим, що може завадити користувачу паралельно використовувати вебплатформу.

По-друге, чат-боти Livechat та CustomGPT забезпечують кращий досвід взаємодії завдяки зрозумілому інтерфейсу, у той час як Intercom не завжди інтуїтивно зрозумілий для користувача через надмірне використання ілюстрацій, блоків контекстного меню тощо.

По-третє, варто звернути увагу на час формування відповідей від інтелектуальних асистентів. Для Intercom та CustomGPT результати показують середній час генерації від 1 до 2 секунд, у той час як Livechat генерує окремі відповіді до 5 секунд. Дані показники мають суттєвий вплив на враження користувачів від комунікації з чат-ботом, адже дослідження Kissmetrics виявило, що більше половини користувачів переживають роздратованість,

якщо час відповіді онлайн ресурсу перевищує 3 секунди [14].

Крім того, іноді для користувачів може гостро стояти питання конфіденційності, адже індивідуальні параметри, що можуть бути надані асистенту, у випадку CustomGPT будуть в подальшому використані для навчання мовної моделі на базі якої побудований чат-бот. У випадку CustomGPT уся інформація буде передана ChatGPT на базі якого побудований даний асистент. Проте, Intercom і Livechat використовують власні мовні моделі, що дає їм змогу уникнути передачі конфіденційних даних на сторонні ресурси. Критичність даного аспекту слід визначати в залежності від основних задач асистентів, якщо вони не передбачають надання конфіденційної інформації при комунікації користувача з чат-ботом, даний ризик можна вважати некритичним.

Окремо розглянемо інформативність та точність формування відповідей. В асистентах Livechat та CustomGPT відповіді повноцінно розкривають питання користувача та чітко співпадають з тематикою самого питання, у той час як чат-бот Intercom може надати надто розлогі відповіді, які виходять за рамки обраної користувачем тематики, або частково їй не відповідають.

Особливу увагу хочеться звернути на збереження та повторне використання контексту діалогу. Даний критерій має ключове значення в комунікації між користувачем та асистентом, адже він надає можливість ставити асистенту уточнюючі запитання, дозволяє користувачу корегувати деталі отриманих відповідей. У ході дослідження виявилось, що Livechat не посилається при запитанні користувача на попередню історію комунікації, як наслідок користувач не зможе поставити питання без повторного задання попередньої теми обговорення (рис. 1.3).

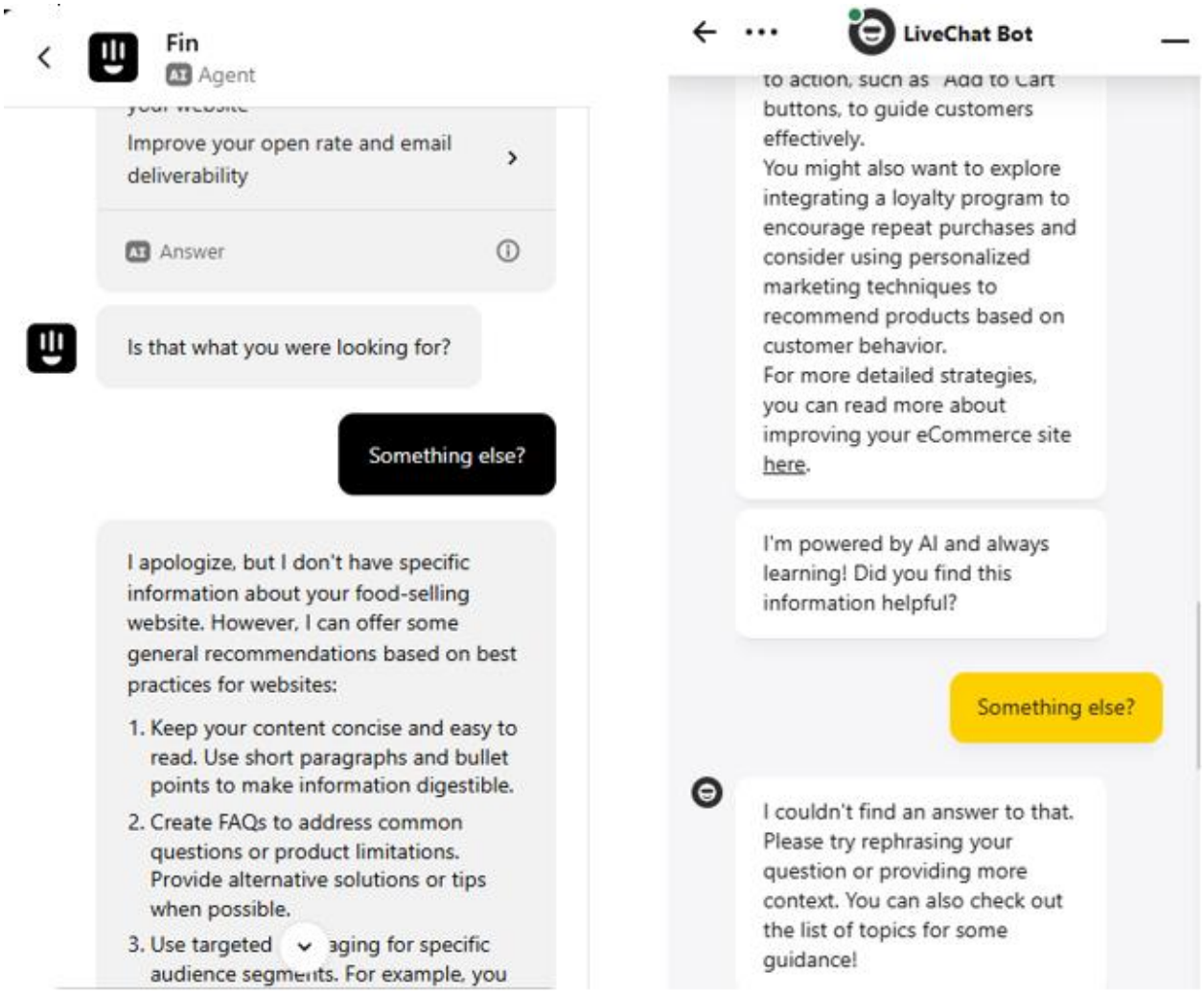

Рисунок 1.3 – Відповіді асистентів на основі звернення до контексту попередніх повідомлень.

У той же час Intercom зберігає контекст діалогу та формує розширену відповідь на основі свого попереднього повідомлення.

Таким чином, провівши аналіз існуючих аналогів, ми визначили ключовий функціонал, зокрема збереження контексту комунікації та надання швидких персоналізованих відповідей на запити користувачів.

1.3 Постановка задачі

Метою роботи є реалізація інтерактивного асистента на базі штучного інтелекту, який допомагає студентам вебплатформи з вибором необхідних IT

курсів та відповідної літератури.

Ключовими функціональними вимогами є:

- асистент доступний для взаємодії на всіх вебсторінках платформи;
- історія комунікації зберігається при переході між сторінками;
- рекомендація чітко відповідає темі поставленого запитання;
- асистент зберігає та використовує попередній контекст комунікації при наступних рекомендаціях.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) Проаналізувати можливості сучасних бібліотек та фреймворків для інтеграції з існуючими мовними моделями;
2) Розглянути допоміжні інструменти та плагіни, які будуть використані в парі з бібліотеками задля досягнення всіх необхідних умов для реалізації та інтеграції асистента у вебплатформу;
3) Підготувати архітектурні рішення та дизайн інтерфейсу задля забезпечення наступної функціональності:
    - можливість взаємодії з асистентом в будь-якій частині вебплатформи;
    - надання персоналізованих рекомендацій у виборі ІТ курсів та літератури;
    - можливість зберігати чат-ботом контексту діалогу для його подальшого використання;
    - збереження чат-ботом історії повідомлень для подальшого продовження комунікації.
4) Реалізувати програмну частину інтерактивного інтелектуального асистента навчального призначення та інтегрувати його в веборієнтовану системи керування навчальними курсами.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Інтеграція мовної моделі із зовнішнім джерелом даних

На сьогоднішній день існує велика кількість навчених мовних моделей, що можуть забезпечити необхідну якість відповідей та вибудовувати персоналізовану комунікацію [15]. Для реалізації асистента нами було обрано один із провідних інструментів на ринку штучного інтелекту – ChatGPT. Динамічна конфігурація, широкий вибір бібліотек та фреймворків для інтеграцій, а також постійно зростаюча спільнота розробників роблять ChatGPT оптимальним вибором для реалізації асистента навчального призначення. Відповідно до рейтингу Toolify, ChatGPT посідає перше місце серед інструментів штучного інтелекту, з щомісячним відвідуванням у 3,1 мільярди користувачів [16].

Окрему увагу хочеться звернути на наповнення обраної моделі необхідним нам джерелом даних. Цей крок є ключовим у процесі інтеграції, оскільки початково ChatGPT не володіє інформацією про навчальні курси та відповідну літературу. Дані відомості зберігається в реляційній базі даних навчальної вебплатформи. Тому ми маємо використати фреймворк, що має необхідний набір методів для зчитування даних зі сховища та їх подальшої конвертації у зрозумілий для мовної моделі формат. Одним із таких інструментів є LangChain.

LangChain — це платформа з відкритим вихідним кодом для створення програм на основі великих мовних моделей. LangChain надає інструменти та абстракції для покращення налаштування, точності та релевантності інформації, яку генерують моделі. За допомогою LangChain розробники можуть гнучко адаптувати мовну модель до конкретних бізнес-контекстів, визначаючи кроки, необхідні для отримання бажаного результату [17].

Основним принципом роботи LangChain є вибудовування так званих «ланцюгів» на основі підготовлених методів фреймворку. Відповідно до поставлених цілей процес обробки джерела даних буде складатися з наступних

кроків (рис. 2.1):

1. Зчитуємо за допомогою відповідного Loader-у LangChain інформацію про курси та літературу зі сховища навчальної вебплатформи.
2. Після зчитування LangChain розбиває отриману інформацію на порції, так звані «чанки», а далі на основі мовної моделі генеруємо текстові embeddings. Простими словами це числове представлення тексту для подальшого отримання його векторного відображення.
3. Завантаження даних у векторну базу даних для подальшого використання.

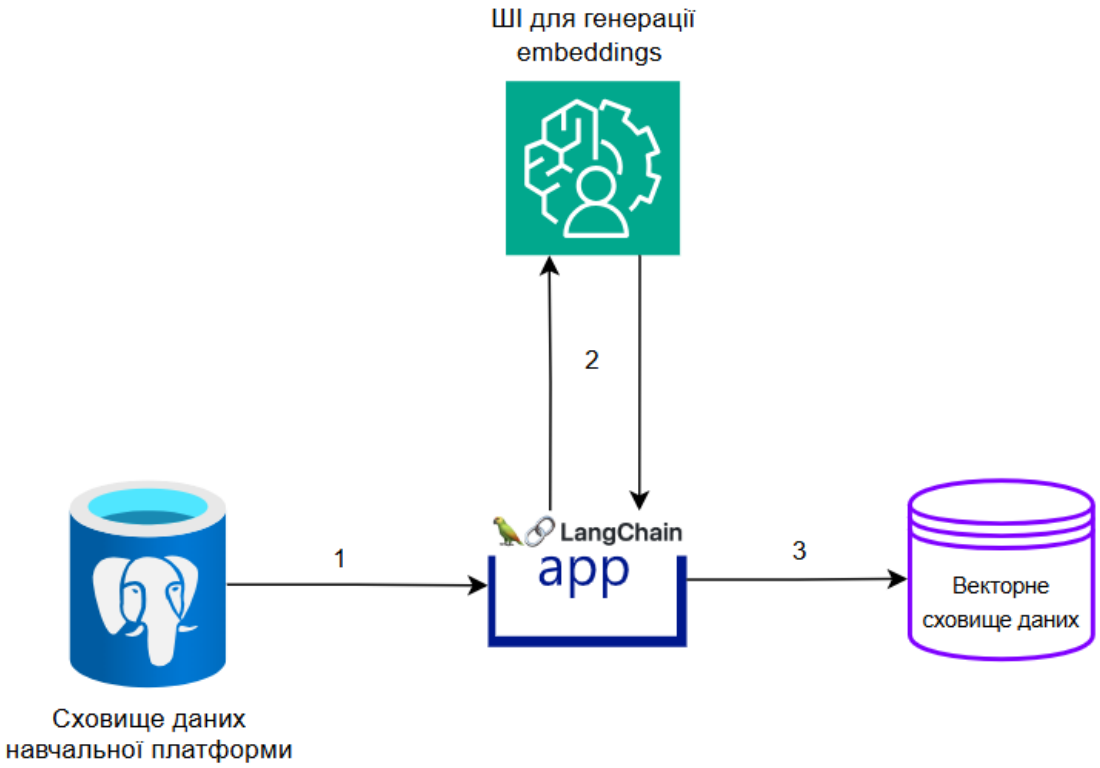

Рисунок 2.1 – Процес зчитування та обробки даних навчальної вебплатформи за допомогою фреймворку LangChain

Перш ніж перейти до алгоритму обробки запиту користувача та опрацювання мовною моделлю даних навчальної платформи, варто визначитись із деякими компонентами попереднього етапу.

На другому етапі рисунку 2.1 зазначено, що LangChain використовує

штучний інтелект для генерування embeddings. Обрана нами мовна модель ChatGPT надає такі можливості, а LangChain - відповідну інтеграцію «з коробки».

На третьому ж етапі ми маємо визначитися з імплементацією векторного сховища даних. Нами була обрана хмарна платформа векторних баз даних Pinecone, яка розроблена спеціальна для підтримки додатків на базі штучного інтелекту, має мінімальну затримку та детальну документацію з усіма необхідними покроковими інструкціями. Більш того, у LangChain вже реалізована інтеграція з Pinecone, що дозволяє використовувати готові класи та методи для взаємодії з ним.

Отже, ми визначилися з усіма необхідними інструментами та підготували дані для їх пошуку та подальшої обробки штучним інтелектом. Фінальним етапом буде створення «ланцюжка» за допомогою LangChain для поєднання усіх інструментів у єдиний процес для генерації відповідей на питання користувача (рис. 2.2):

1. Отримуємо питання від користувача, зберігаємо для подальшої обробки.
2. Створюємо та конфігуруємо відповідний ланцюжок та конвертуємо питання користувача у векторний вигляд.
3. Виконуємо пошук у векторній базі даних Pinecone та отримуємо релевантні вектори.
4. Передаємо питання та вже сформовані релевантні дані, що будуть використані у формуванні відповідей нашою мовною моделлю.
5. Штучний інтелект генерує відповідь на основі наданої інформації та повертає готовий результат, який в подальшому буде перенаправлений користувачу.

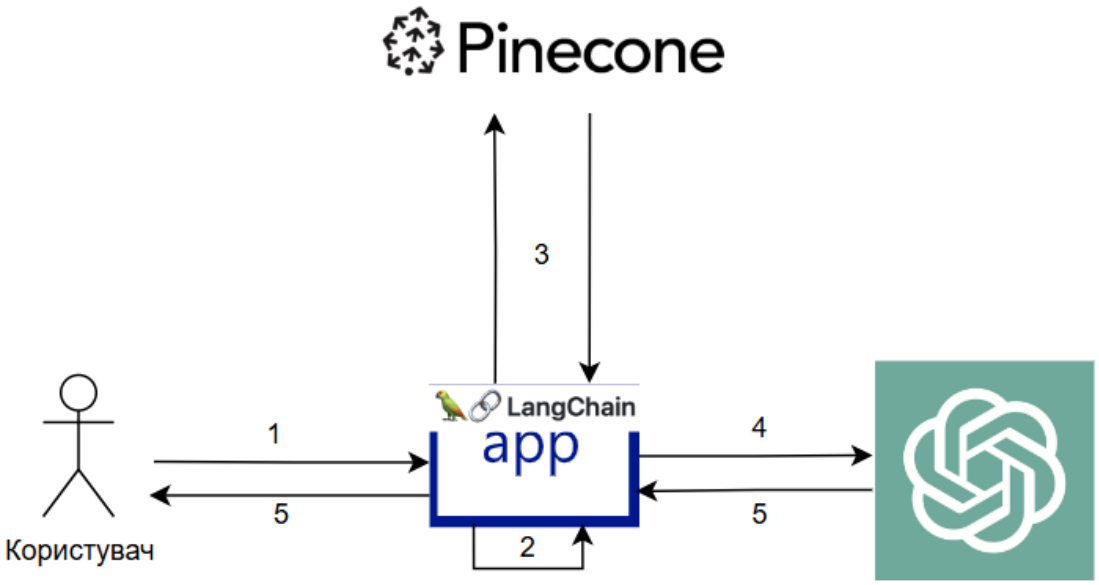

Рисунок 2.2 – Процес обробки питання користувача та подальша генерація відповіді

Таким чином, проаналізувавши вимоги, ми сформували набір інструментів та фреймворків, що допомогли нам обрати метод вирішення поставленої задачі. Використовуючи вищеописаний підхід, наш додаток зможе забезпечити якісну інтеграцію мовної моделі із зовнішнім джерелом даних.

2.2 Серверна частина інтелектуального асистента

Усі сучасні інструменти штучного інтелекту потребують програмної платформи для функціонування, тому наступним кроком буде вибір мови програмування, яка б надала можливість інтегрувати фреймворки та інструменти описані в розділі 2.1.

Згідно з даними порталу Index, найбільш адаптованою мовою програмування для штучного інтелекту є Python. Наявність великої кількісті спеціалізованих бібліотек і фреймворків, а також легкість їх впровадження роблять Python кращим вибором для реалізації додатків на базі штучного інтелекту [18].

Виконавши дослідження можливостей мови програмування Python, було

виявлено, що всі інструменти та фреймворки описані в розділі 2.1 можуть бути використані на його основі.

Окремо варто зазначити, що крім ШІ-фреймворків та базового процесу генерування відповідей на питання користувача, мова програмування також має підтримувати взаємодію з клієнтською частиною – вебінтерфейсом асистента. Це передбачає наявність хоча б однієї бібліотеки чи фреймворка для реалізації клієнт-серверної взаємодії. Phyton володіє великою кількістю подібних інструментів, один із таких - Flask.

Flask – це вебплатформа Python, яка широко використовується для розробки вебдодатків, API та мікросервісів. Як наслідок, простота, гнучкість, масштабованість, надійність і здатність обробляти збільшений трафік є ключовими факторами її популярності серед розробників [19].

Отже, використання мови програмування Python у поєднанні з фреймворком Flask дозволить реалізувати надійну клієнт-серверну комунікацію, а також інтегрувати усі необхідні інструменти на базі штучного інтелекту для обробки запитів користувачів.

2.3 Клієнтський вебінтерфейс інтерактивного чат-бота

Для комфортної комунікації користувача з інтелектуальним асистентом ми маємо підготувати відповідний дизайн та налаштувати комунікацію з серверною частиною.

В традиційній розробці клієнтської частини, першим кроком мав би стати аналіз та вибір технологій на базі яких буде побудований інтелектуальний асистент. Проте у даному випадку чат-бот буде інтегрований у існуючу навчальну веборієнтовану платформу, що була реалізована в рамках дипломної роботи бакалавра [20]. Таким чином правильним рішенням для розробки клієнтської частини інтелектуального асистента буде використання бібліотек та фреймворків на яких базується навчальна система, а саме мова

програмування TypeScript і фреймворк Angular.

Наступним кроком буде побудова макета інтелектуального асистента.

Варто зазначити, що при дизайні необхідно врахувати попередньо описаний функціонал, аби відобразити на макеті усі можливі способи взаємодії користувача з чат-ботом. З результатами виконаного проектування можна ознайомитися на рисунку 2.3.

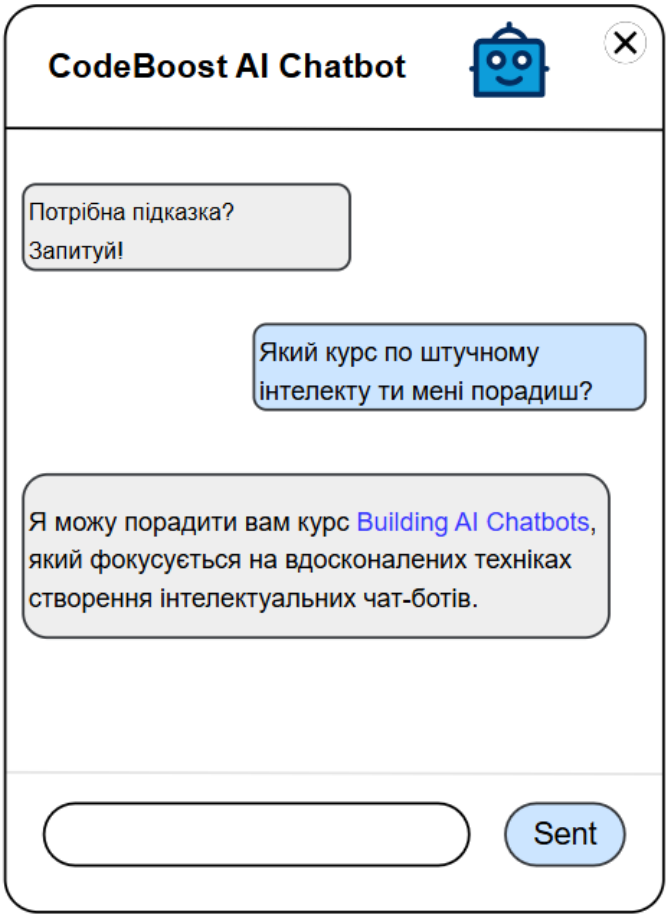

Рисунок 2.3 – Процес обробки питання користувача та подальша генерація відповіді

Отже, сформований дизайн інтелектуального асистента та обрані фреймворки задля його реалізації надають основу у подальшій реалізації всіх функціональних можливостей. Даний підхід забезпечує модульність та легку інтеграцію інтелектуального асистента у будь-яку частину навчальної вебплатформи.

3 АРХІТЕКТУРНЕ ПРОЕКТУВАННЯ АСИСТЕНТА

3.1 Діаграма прецедентів

Одним із важливих етапів проектування додатка є визначення та документування основного функціоналу системи та опис ключових ролей, що будуть з нею взаємодіяти. Для таких цілей може бути використана діаграма прецедентів (Use Case Diagram).

В уніфікованій мові моделювання (UML) діаграма прецедентів може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодію з системою. Ефективна діаграма варіантів використання може допомогти вашій команді обговорити та представити [21]:

- сценарії, у яких система взаємодіє з людьми, організаціями або зовнішніми системами;
- цілі, яких допомагає досягти система цим об’єктам (відомим як актори);
- область вашої системи.

У рамках нашої системи будуть присутні двоє споживачів: «Користувач» та «Адміністратор» навчальної системи, а з іншого боку двоє постачальників: інструмент штучного інтелекту «ChatGPT» та хмарна векторна база даних «Pinecone».

Основними функціями взаємодії акторів виступають:

- створення питання до асистента;
- уточнення поставленого питання до асистента;
- додавання курсів та/або літератури;
- редагування курсів та/або літератури.

Реалізована діаграма прецедентів для інтерактивного інтелектуального асистента, що буде інтегрований у вебсистему з навчальними курсами представлена на рисунку 3.1.

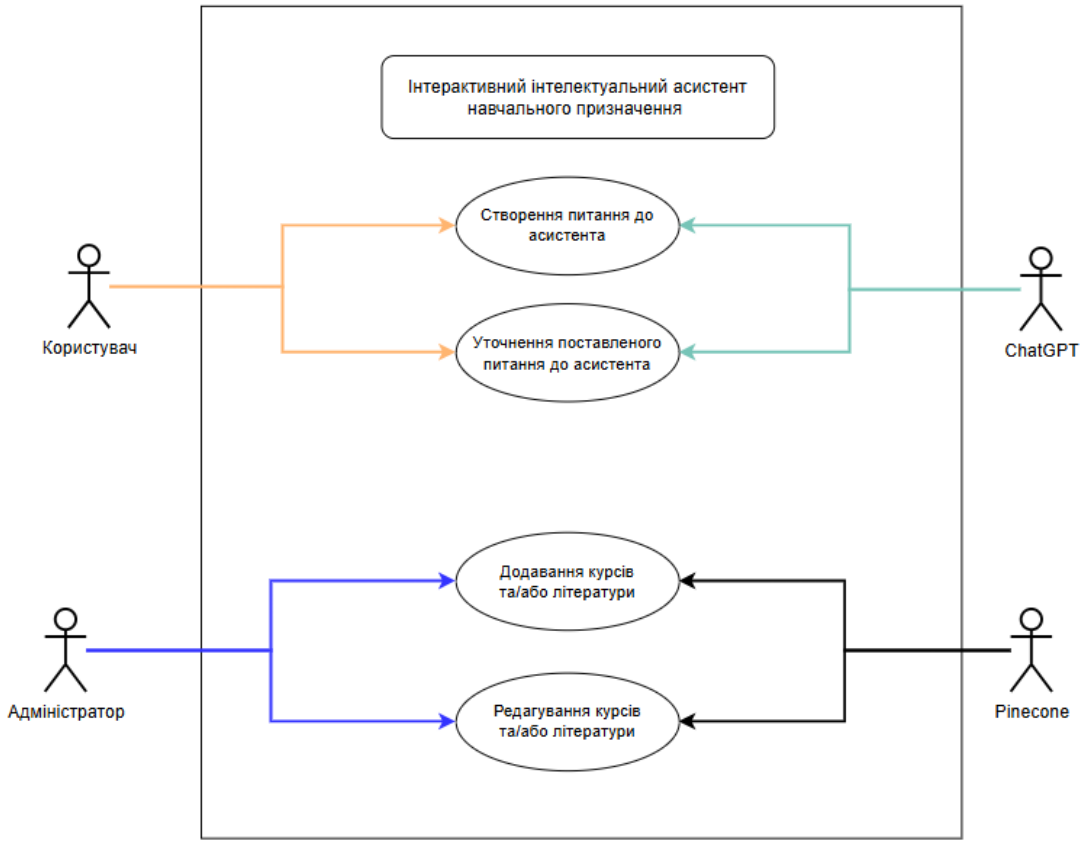

Рисунок 3.1 – Діаграма прецедентів чат-боту

Таким чином, створивши діаграми прецедентів, ми змогли визначити основні сценарії використання, структурувати вимоги та побудувати основну логіку роботи чат-боту.

3.2 Опис функціональних процесів додатку у нотації IDEF0

Окремим кроком є визначення функціональних бізнес процесів нашого асистента. На відміну від діаграми прецедентів, де ми поверхнево описували основні функції та їх акторів, на даному етапі необхідно визначити окремі деталі певного процесу, а саме:

- тригер (ініціація процесу);
- управління (умова, або попередній процес);
- механізм (інструмент, додаток для реалізації задачі);

- фінальний результат.

Для визначених цілей є доречним застосувати методологію моделювання IDEF (Integrated DEFinition).

IDEF (Integrated Definition) – це графічна методологія моделювання процесів, яка використовується для впровадження систем та програмного забезпечення. Ці методи використовуються для функціонального моделювання даних, симуляції, об’єктно-орієнтованого аналізу та отримання знань. IDEF відноситься до сімейства мов моделювання, яке включає 16 різних методів. Ці методи моделювання процесу охоплюють широкий спектр використання, і кожен метод охоплює певний тип даних. У таблиці нижче наведено 16 методів IDEF. Методи від IDEF0 до IDEF4 є найбільш часто використовуваними методами [22].

У рамках моделювання інтелектуального асистента доречним буде обрати перший рівень методології IDEF – IDEF0.

У рамках нашого додатку визначимо дві ключові функції: створення нового запиту до асистента, та уточнення попереднього. Дані бізнес-процеси представлені на рисунках 3.2, 3.3.

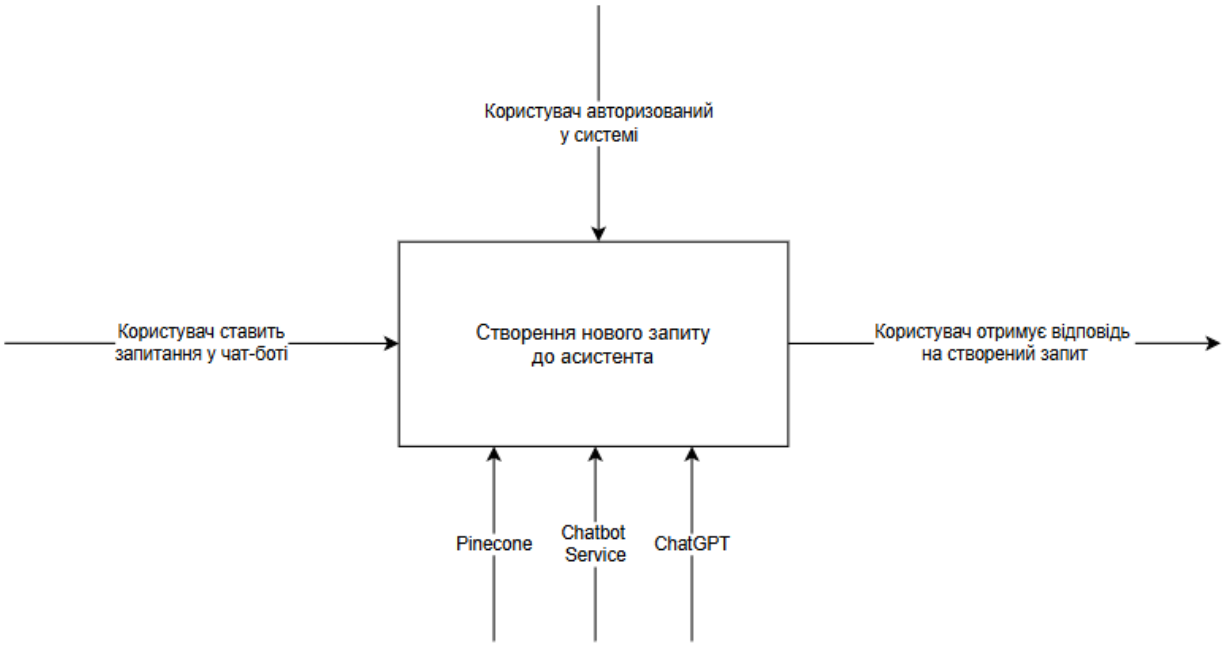

Рисунок 3.2 – IDEF0 діаграма функціоналу «Створення нового запиту до асистента»

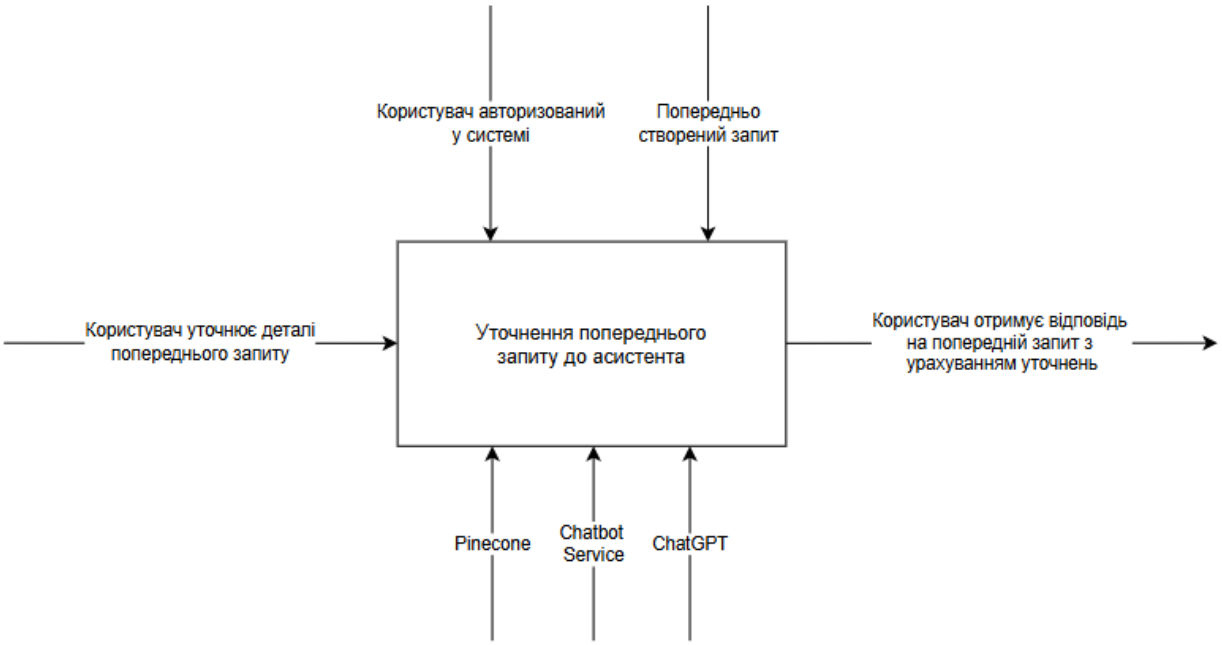

Рисунок 3.3 – IDEF0 діаграма функціоналу «Уточнення попереднього запиту до асистента»

Як результат ми деталізували аспекти бізнес-процесів інтелектуального асистента за допомогою нотації IDEF0, що допоможе нам врахувати всі деталі при подальшій розробці додатку.

3.2 Діаграма розгортання

У даному етапі проектування ми побудуємо діаграму розгортання. Задля чіткого розуміння мети та способу її досягнення при побудові цього типу діаграми пропонуємо детальніше розглянути його цілі та переваги.

Діаграми розгортання — це різновид структурної діаграми, яка використовується для моделювання фізичних аспектів системи. Вони часто використовуються для моделювання статичного вигляду розгортання системи (топології апаратного забезпечення). Основним призначенням діаграми розгортання є [23]:

- опис загальної структури вузлів системи;

- подальше використання для побудови архітектури;
- планування апаратних елементів та демонстрація зв’язку між ними.

Для реалізації проекту було виділено два основних вузла: backend сервіс, що являє собою Web API реалізований на платформі Python, а також frontend сервіс, котрий представлений вже реалізованою вебсистемою навчальних курсів.

Варто зазначити, що в рамках клієнтського вузла реалізований sub-компонент нашого чат-боту, який містить основну логіку комунікації із серверним вузлом.

Окремими вузлами на діаграмі розгортання необхідно виокремити хмарні сервіси. Вони функціонують на віддалених серверах, а отже нам немає необхідності локально конфігурувати та запускати відповідні вузли. Подібний підхід хоч і має певні недоліки у вигляді latency у передачі даних, проте надає можливість значно економити ресурси локального сервера та зменшити витрати часу на конфігурування та підтримку.

Виділивши всі існуючі вузли, а також визначивши алгоритми комунікації між ними, можемо візуалізувати діаграму розгортання. Отриманий результат відображено на рисунку 3.4.

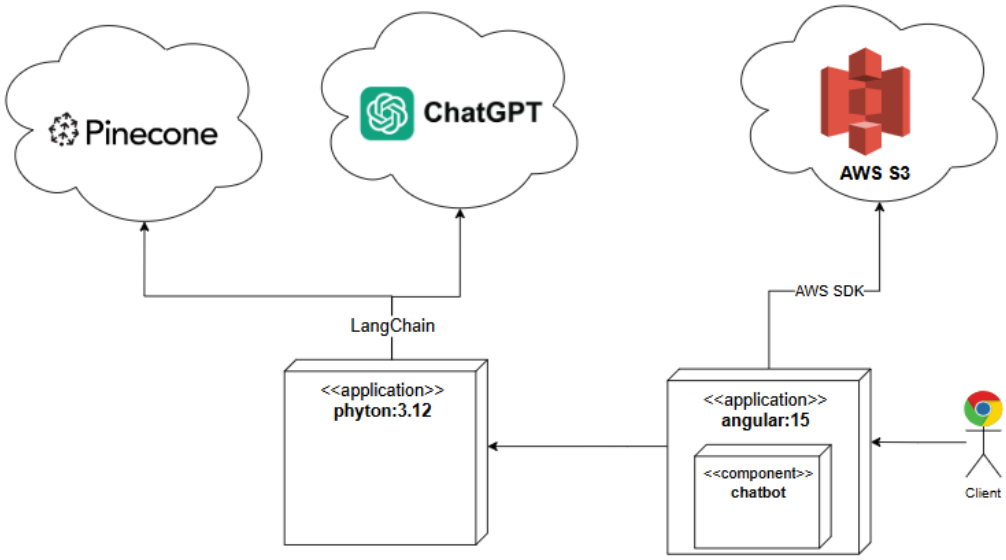

Рисунок 3.4 – Діаграма розгортання інтелектуального асистента

Таким чином, знизивши рівень абстракції для описання програмних вузлів та способів їх комунікації ми маємо чітке уявлення про всі існуючі програмні компоненти для їх подальшої розробки та інтеграції.

3.3 Діаграма послідовності

Фінальним кроком у проектуванні є детальна візуалізація процесів взаємодії між компонентами додатку, а також відображення їх функціональних кроків. Для побудови відповідних процесів була обрана діаграма послідовностей.

Діаграми послідовності – це діаграми взаємодії, які детально описують, як виконуються операції. Вони фіксують взаємодію між об’єктами в контексті співпраці. Діаграми послідовності зосереджені на часі, і вони візуально показують порядок взаємодії за допомогою вертикальної осі діаграми. Діаграми послідовності фіксують взаємодію високого рівня між користувачем та системою, а також між системою та іншими системами [24].

Для опису діаграми послідовності інтелектуального асистента було обрано процесу створення користувачем питання до асистента (рис. 3.5), оскільки саме він відображає ключовий функціонал системи.

Ключовими елементами у контексті співпраці є:

- користувач – ініціатор процесу, задає питання асистенту та отримує відповідь;
- frontend – Angular компоненту в навчальній вебплатформі, що надає користувачу можливість вводити запитання через інтерфейс чат-бота та відповідає за відправлення відповідного запиту на backend сервіс;
- backend – web API на базі Python, що надає відповідний endpoint для обробки запиту, проводить валідацію тексту питання користувача, конвертує його у векторний вид та виконує запит у Pinecone, формує історію комунікації задля збереження контексту та передає підготовлені дані у ChatGPT;
- Pinecone – векторна база даних у якій виконується пошук збігів питання користувача з наявними даними про курси та літературу. Повертає знайдені вектори для подальшого їх використання у ChatGPT;
- ChatGPT – інструмент штучного інтелекту, що отримує питання користувача, знайдені збіги з Pinecone та приймає рішення у формуванні фінальної відповіді.

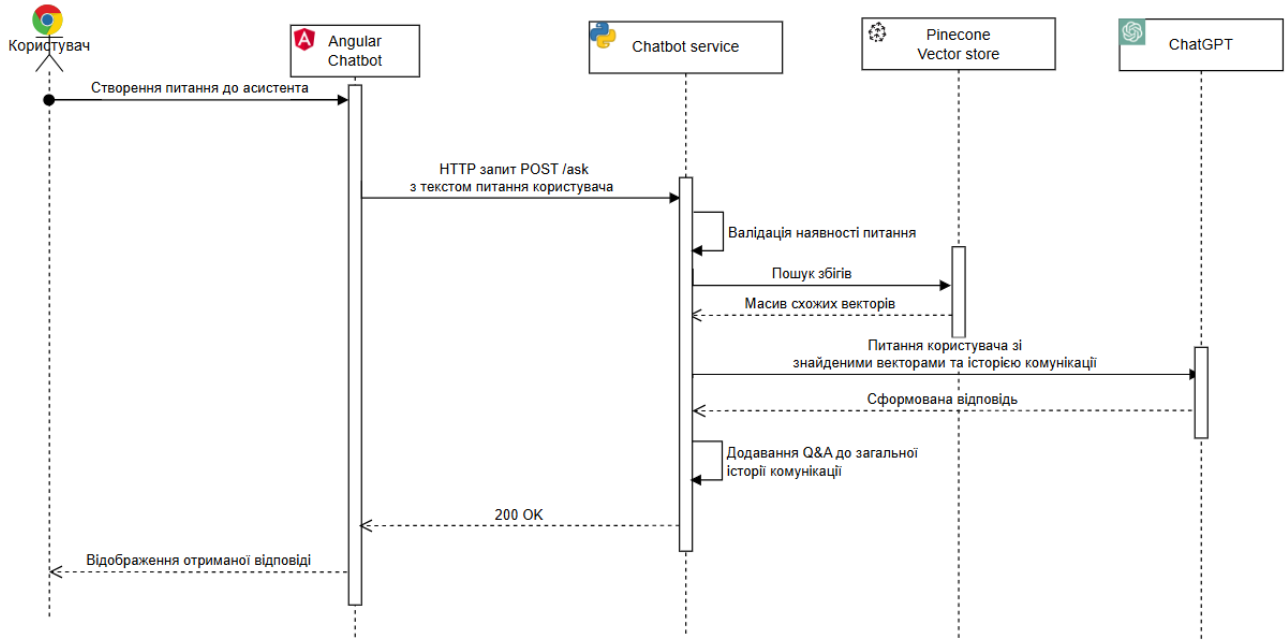

Рисунок 3.5 – Діаграма послідовності для бізнес-процесу створення користувачем питання до асистента

4 ПРОГРАМНА РЕАЛІЗАЦІЯ АСИСТЕНТА

4.1 Налаштування GhatGPT

Перш ніж перейти до програмної реалізації необхідно виконати конфігурацію хмарних застосунків, які будуть використовуватися нашим додатком. В описах конфігурації хмарних інструментів будуть пропущені кроки реєстрації, підтвердження пошти та будь які інші, що не мають прямого впливу на функціонування розроблюваного застосунку.

З метою налаштування інструменту штучного інтелекту ChatGPT був використаний посібник з інтеграції на офіційному сайті продукту [25].

Першим кроком, після проходження базових етапів авторизації, є створення власного проекту, він є основою для подальшої взаємодії з сервісом.

Наступним кроком, у рамках створеного проекту, необхідно створити API Key (рис. 4.1). Він представляє собою унікальний ключ, який буде використаний при інтеграції з нашим backend сервісом задля подальшої авторизації у системі ChatGPT. Назва ключа може бути обрана довільна, у нашому випадку це «CodeBoostChatBot», де префікс означає назву нашої навчальної платформи.

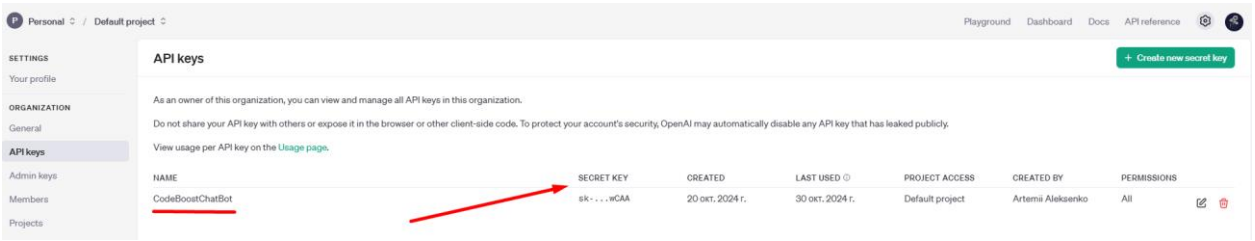

Рисунок 4.1 – Розділ меню зі створеним API Key

Окремо варто зазначити, що при спробі на даному етапі виконати запит до сервісу – виникне помилка. Причиною є відсутність безкоштовної або навіть тимчасово безкоштовної інтеграції. Тому наступним кроком є поповнення рахунку на мінімальну суму (рис. 4.2). В масштабах нашого проекту п’яти доларів буде достатньо для інтеграції, її тестування та

подальшого використання.

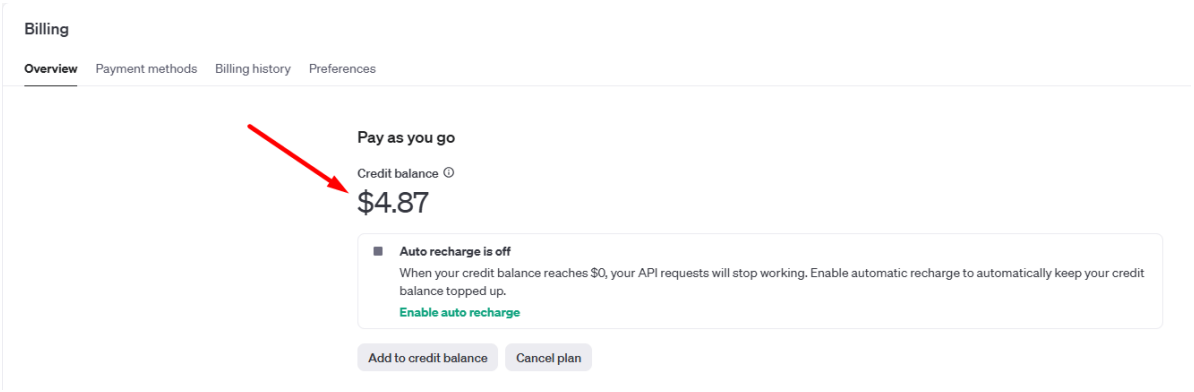

Рисунок 4.2 – Розділ меню для поповнення та відстеження балансу

Гарної практикою також є встановлення нотифікацій про зменшення суми до критичного мінімуму та лімітів на зняття коштів, проте в рамках ресурсів залучених до нашого проекту даний шаг є необов’язковим.

4.2 Налаштування Pinecone

Наступним кроком є конфігурація хмарної векторної бази даних Pinecone. Аналогічно до попереднього сервісу виконуємо реєстрацію, підтвердження пошти та створюємо проект на базі якого будемо робити інтеграцію.

На відміну від ChatGPT, Pinecone надає можливість безкоштовного використання на основі обмеженого функціоналу та об’ємів даних. Проте у рамках нашого інтелектуального асистенту наданих можливостей буде цілком достатньо для реалізації ключових функцій. Після успішної авторизації та створення проекту нам пропонується створити індекс для початку роботи з векторним сховищем (рис. 4.3)

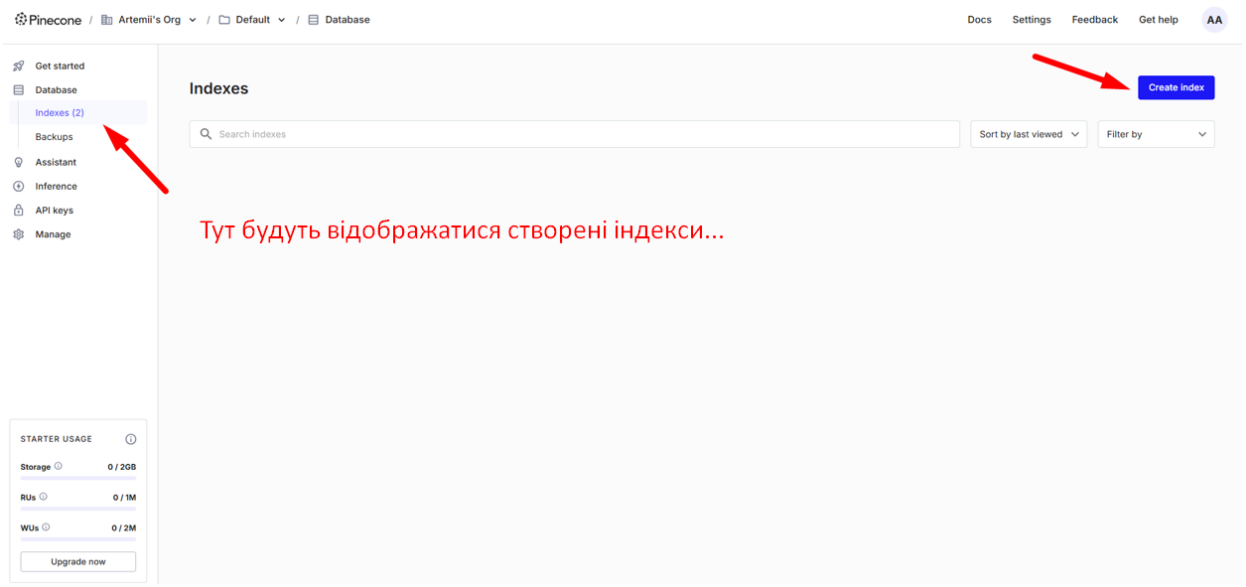

Рисунок 4.3 – Розділ сховища зі списком створених індексів

Індекс — це організаційна одиниця найвищого рівня векторних даних у Pinecone. Він приймає та зберігає вектори, обслуговує запити над векторами, які містить, і виконує інші векторні операції над його вмістом [26].

При ініціалізації створення індексу нас буде направлено на відповідну сторінку (рис. 4.4). Тут нам необхідно провести базові налаштування майбутнього індексу. Однією із ключових можливостей є автоматична конфігурація на основі мовної моделі, котра буде використана при розробці. Проте для забезпечення найкращих умов роботи з моделлю автоматично можуть бити використані функції, що недоступні нам за безкоштовним тарифним планом, але оскільки дані умови не є обов'язковими для функціонування нашого додатку, при створені індексу будуть використані базові налаштування, а саме: Cloud Provider – AWS та Region – us-east-1.

Після успішного створення індексу він буде відображений у загальному списку (рис. 4.5), де можна буде переглянути загальну інформацію про нього:

- host, на якому знаходиться розгорнута база даних;
- status, що описує поточний стан сервісу;
- cloud, region, type – це ті конфігурації, що ми вказували при створенні.

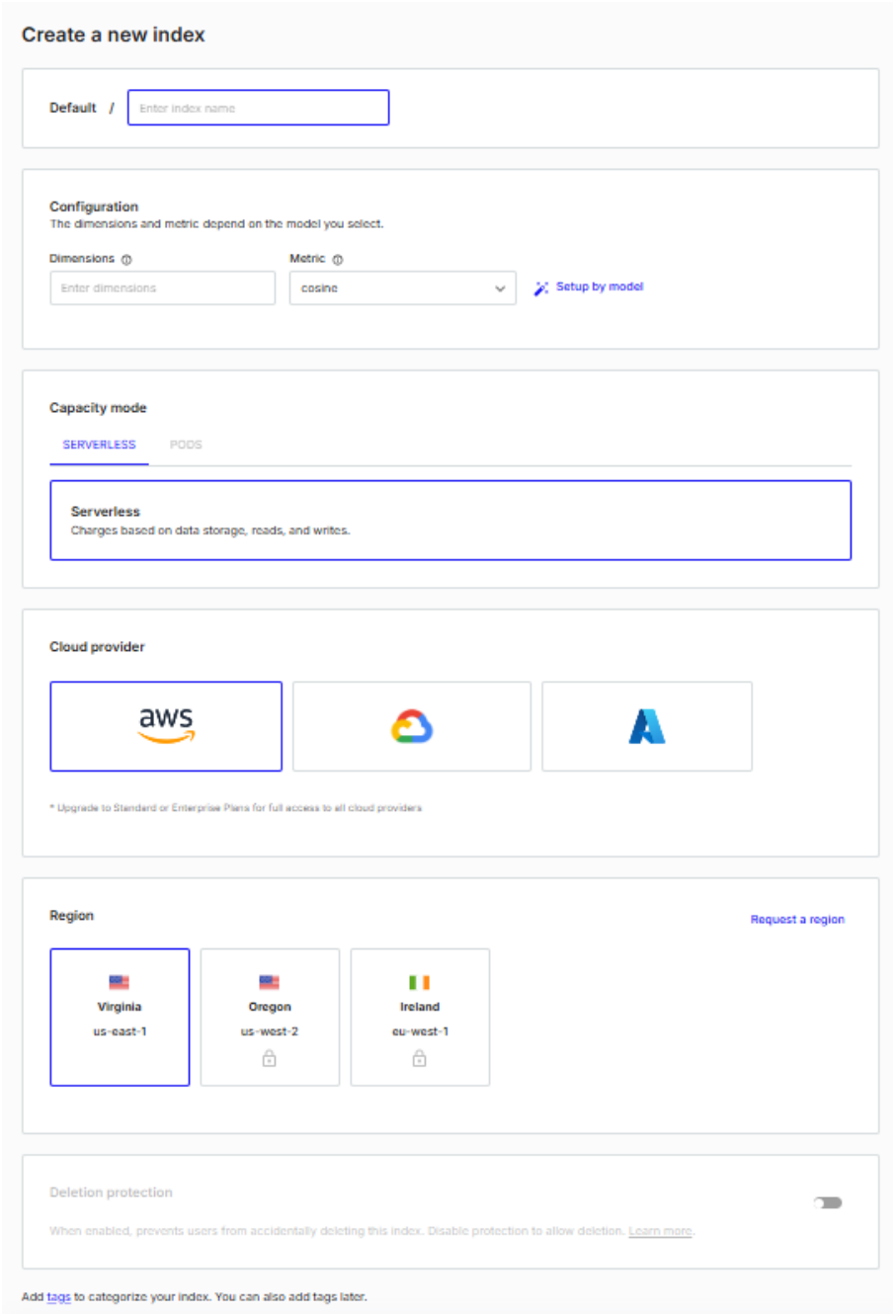

Рисунок 4.4 – Сторінка для створення нового індексу

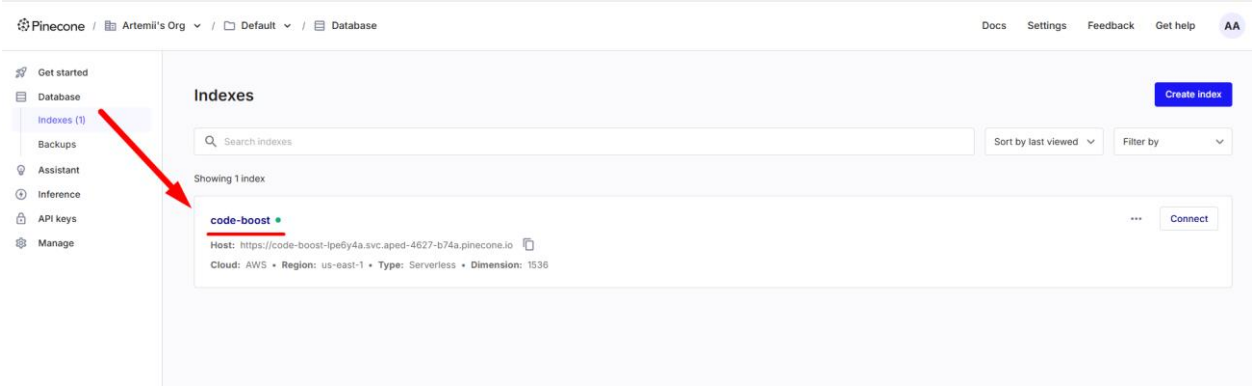

Рисунок 4.5 – Розділ сховища зі створеним індексом

Фінальним кроком у налаштуванні Pinecone є створення API Key (рис. 4.6). Принцип найменування та спосіб подальшого використання при інтеграції з серверною частиною є аналогічним до описаного в розділі 4.1.

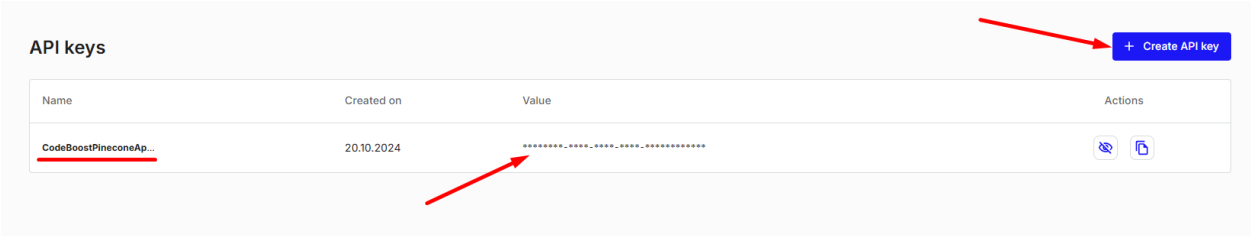

Рисунок 4.6 – Створений API Key для Pinecone

4.3 Реалізація серверної частини додатка

При розробці серверної частини інтерактивного інтелектуального асистента буде використана мова програмування Python, а також фреймворки Flask та LangChain. Причини вибору та аргументація на користь даних інструментів розробки описані в розділі 2.2.

Першим кроком є створення проекту Python, у нашому випадку для ініціалізації проекту та подальшого написання коду використовується інтегроване середовище розробки (IDE) – PyCharm через його зручність та великий набір функцій.

Другим кроком є створення файлу .env (рис. 4.7), що буде містити змінні середовища, а саме: API Key для ChatGPT, API Key для Pinecone та назва створеного нами індексу, а також URL бази даних нашої навчальної платформи задля зчитування інформації про навчальні курси та літературу.

```
OPENAI_API_KEY=sk-proj-tim6UDRadc6Zn
INDEX_NAME=code-boost
PINECONE_API_KEY=57a2ca51-64d8-
DATABASE_URL=postgresql://postgres:
```

Рисунок 4.7 – .env файл зі змінними середовища

Наступним кроком є написання коду, що відповідає за зчитування інформації про курси та літературу з бази даних навчальної вебсистеми. Для цього буде використано фреймворк LangChain.

Ініціюємо зв’язок з базою даних навчального ресурсу та реалізуємо завантаження необхідних даних (рис. 4.8).

```
# Init CodeBoost database connection
db = SQLDatabase.from_uri(database_uri=os.environ.get("DATABASE_URL"))

# Load  document
loader = SQLDatabaseLoader(query=course_query_course, db=db)
document = loader.load()
```

Рисунок 4.8 – Імплементація підключення до бази даних для подальшого зчитування інформації

Далі нам необхідно завантажити отриману інформацію до хмарного сховища Pinecone, але оскільки формат зберігання даних є векторним, нам буде необхідно їх видозмінити. Попередньо імпортуємо необхідні залежності та виконуємо наступні маніпуляції з даними:

1. Розділяємо отримані дані на «чанки» (рис. 4.9);
2. Формуємо векторні embeddings за допомогою запиту до мовної моделі ChatGPT (рис. 4.10);
3. Завантажуємо отримані вектори у Pinecone (рис. 4.10).

```
# Split entire data into chunks
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
texts = text_splitter.split_documents(document)
print(f"created {len(texts)} chunks")
```

Рисунок 4.9 – Імплементація розділення даних на «чанки»

```
# Create vector embeddings and save it in pinecone database
embeddings = OpenAIEmbeddings(
    openai_api_type=os.environ.get("OPENAI_API_KEY"))
PineconeVectorStore.from_documents(texts, embeddings,
                                   index_name=os.environ.get("INDEX_NAME"))
```

Рисунок 4.10 – Імплементація створення векторних embeddings та подальше завантаження у Pinecone

Таким чином ми підготували дані нашої навчальної системи для подальшої обробки штучним інтелектом. Зміст файлу див. Додаток А. Наступним кроком буде створення Web API endpoint, що буде приймати запит від клієнтської частини зі змістом звернення користувача (рис. 4.11). Виконаємо наступні кроки:

1. Імпортуємо всі необхідні залежності;
2. Налаштовуємо CORS для всіх доменів, для подальшої безперебійної взаємодії з клієнтською частиною;
3. Описуємо метод endpoint з вказанням кінцевого шляху та HTTP методу для формування запиту. У нашому випадку це буде POST із кінцевою точкою в «/ask»
4. Додаємо перевірку, що питання від користувача було надіслано в запиті, якщо ні – повертаємо відповідь з 400 помилкою.

```python
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)

CORS(app)

@app.route( rule: "/ask", methods=["POST"])
def ask_question():
    data = request.json
    question = data.get("question")

    if not question:
        return jsonify({"error": "No question provided"}), 400
```

Рисунок 4.11 – Імплементація endpoint для обробки клієнтських запитів

Наступним кроком проведемо конфігурацію об’єктів для реалізації основної логіки (рис. 4.12). У нашому випадку необхідно буде підготувати наступні об’єкти:

1. За аналогією до попереднього прикладу створюємо embeddings на базі ChatGPT та підключення до векторної бази Pinecone, але на цей раз вони будуть використовуватися для пошуку співпадінь;
2. Формуємо об’єкт «chat» для надсилання запитань та отримання згенерованої відповіді від ChatGPT. Важливим пунктом є виставлення параметра «temperature» зі значенням 0, аби мовна модель надавала як можна коротші відповіді без зайвих пояснень чи прикладів;
3. Налаштовуємо ланцюжок дій за допомогою LangChain, щоб використати вищезазначені об’єкти для пошуку співпадінь та їх подальшому надсиланні в ChatGPT.

```
embeddings = OpenAIEmbeddings(openai_api_type=os.environ.get("OPENAI_API_KEY"))
vectorstore = PineconeVectorStore(
    index_name=os.environ["INDEX_NAME"], embedding=embeddings
)

chat = ChatOpenAI(verbose=True, temperature=0, model_name="gpt-3.5-turbo")

qa = ConversationalRetrievalChain.from_llm(
    llm=chat, chain_type="stuff", retriever=vectorstore.as_retriever()
)
```

Рисунок 4.12 – Конфігурація необхідних функціональних об’єктів

На завершальному етапі необхідно отримати відповідь на запит користувача та повернути її клієнтській частині додатку. Для цього реалізуємо наступний функціонал (рис. 4.13):

1. Ініціюємо виклик раніше створеного ланцюга LangChain із параметром «Питання користувача», а також передаємо попередньо збережену історію повідомлень, якщо така існує, аби ChatGPT формував відповідь на основі попереднього досвіду спілкування.
2. Отримуємо відповідь від штучного інтелекту та зберігаємо дану пару питання/відповідь у історію комунікації для використання у наступному запиті.
3. Формуємо відповідь на основі отриманих даних та повертаємо її до клієнтської частини інтелектуального асистента, де у подальшому вона буде оброблена та відображена користувачу.

```
response = qa.invoke({"question": question, "chat_history": chat_history})
history = (response["question"], response["answer"])

chat_history.append(history)

response = jsonify({"response": response})
response.headers.add( _key: 'Access-Control-Allow-Origin', _value: '*')

return response
```

Рисунок 4.13 – Імплементація логіки отримання відповіді від ChatGPT із формуванням історії комунікації

Отже, завдяки аналізу та вибору необхідних фреймворків та інструментів штучного інтелекту, а також їх коректній конфігурації нами була реалізована серверна частина інтерактивного асистента навчального призначення. Зміст файлу див. Додаток Б.

4.4 Розробка клієнтської частини додатку

Для реалізації клієнтської частини були використані аналогічні інструменти, що і в навчальній вебплатформі, а саме мова програмування TypeScript і фреймворк Angular. Переваги вибору описані в розділі 2.3.

Базовим кроком є створення компоненту чат-боту за допомогою команди у терміналі ***ng g c ChatBotComponent***. Результат генерації компоненту відображений на рисунку 4.14.

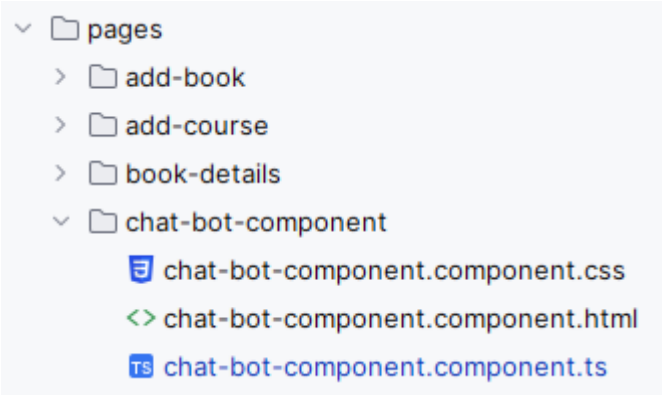

Рисунок 4.14 – Структура згенерованого компонента чат-бот

Розглянемо основні файли згенерованого компонента:

- HTML – файл, що містить шаблон гіпертекстової розмітки для компонента. У даному файлі описується, як виглядає компонент на сторінці та реалізується основна структура інтерфейсу користувача, а також додаються прив'язки даних і Angular директиви.
- CSS – файл, що зберігає стилі для компонента, він визначає візуальні стилі (шрифти, кольори, відступи) і застосовується лише до відповідного компонента, що забезпечує ізольованість стилів.
- TS – файл, що містить головну бізнес-логіку компонента. Тут описується клас, який реалізує поведінку та взаємодію з даними. TypeScript-файл включає властивості і методи, потрібні для обробки подій, керування даними та обробки взаємодій з користувачем.

Варто зазначити, що генерація компонента та уся його подальша розробка відбувається в рамках кодової бази навчальної вебсистеми, що у майбутньому надасть можливість інтегрувати чат-бота на будь яку сторінку ресурсу.

Першим кроком реалізуємо TypeScript-файл, для цього опишемо ключові поля, що будуть зберігати ключові дані та стан нашого чат-бота: видимість для користувача, повідомлення від користувача, історія повідомлень, відповідь, отримана від серверної частини асистента. Описані поля зображені на рисунку 4.15.

```
@Component({
  selector: 'app-chat',
  templateUrl: './chat-bot-component.component.html',
  styleUrls: ['./chat-bot-component.component.css']
})
export class ChatComponent implements OnInit {

  chatVisible: boolean = false;
  userMessage: string = '';
  messages: { text: string, isBot: boolean }[] = [];
  response: string = '';
```

Рисунок 4.15 – Опис полів TypeScript-класу для компонента чат-бота

Другим кроком використаємо метод ***ngOnInit*** (рис. 4.16) аби при завантаженні компонента завантажувалася попередня історія комунікації користувача з асистентом. Подібна ініціалізація допоможе продовжити комунікацію з будь-якого місця навчальної платформи.

```
ngOnInit() : void {
  this.messages = this.chatService.getMessages();
}
```

Рисунок 4.16 – Реалізація ngOnInit методу

Третім кроком реалізуємо основні методи маніпуляції чат-ботом. Реалізація розгортання та скролінг чату при надсиланні повідомлення зображені на рисунках 4.17, 4.18 відповідно.

```
toggleChat() : void {
  this.chatVisible = !this.chatVisible;

  // If the chat is open and there are no messages, send the first message from the bot
  if (this.chatVisible && this.messages.length === 0) {
    this.sendBotMessage( text: 'Потрібна підказка? Запитуй!');
  }
}
```

Рисунок 4.17 – Реалізація розгортання чату

```
scrollToBottom() : void {
  setTimeout( handler: () : void => {
    this.chatContent.nativeElement.scrollTop = this.chatContent.nativeElement.scrollHeight;
  }, timeout: 100);
}
```

Рисунок 4.18 – Реалізація скролінгу чату

Фінальним кроком реалізуємо ключові методи комунікації, що відповідають за надсилання запиту на серверну частину, а також відображення запитань користувачів із подальшою генерацією відповіді інтелектуального асистента. Результат імплементації методів відображений на рисунках 4.19-4.21 відповідно.

```
ask(question: string) : void {
  this.chatService.askQuestion(question).subscribe(
    next: (res) : void => {
      // Receive the bot's response
      let answer = res.response.answer;
      // Using a regular expression to find words in quotes
      const quotedWords = answer.match(/"([^"]+)"/g);
      // Extract course names by removing quotes
      const wordsToSend = quotedWords ? quotedWords.map((word: string) => word.slice(1, -1)) : [];

      this.makeTitlesHyperlink(wordsToSend, answer).subscribe( observerOrNext: (answerWithHyperlinkTitles : string ) : void => {
        this.sendBotMessage(answerWithHyperlinkTitles);
      });
    },
    error: (err) : void => {
      console.error(err);
      this.sendBotMessage( text: 'Сталася помилка. Спробуйте ще раз.');
    }
  );
}
```

Рисунок 4.19 – Реалізація відправки запиту на серверну частину

```
sendMessage() : void {
  if (this.userMessage.trim()) {
    this.chatService.addMessage({text: this.userMessage, isBot: false});
    const question : string = this.userMessage;
    this.userMessage = '';
    this.scrollToBottom();

    this.ask(question);
  }
}
```

Рисунок 4.20 – Реалізація відображення питання користувача

```
sendBotMessage(text: string) : void {
  this.chatService.addMessage({text, isBot: true});
  this.scrollToBottom();
}
```

Рисунок 4.21 – Реалізація відображення питання асистента

Таким чином ми реалізували головну бізнес-логіку для нашого додатку, проте вона невід’ємно пов’язана з ключовим HTML-блоками, аби отримати чітке розуміння взаємодії візуального інтерфейсу та відповідної бізнес-логіки розглянемо реалізацію HTML-файлу на рисунку 4.22.

```
<link href="https://fonts.googleapis.com/css2?family=Exo+2:wght@400;700&display=swap" rel="stylesheet">
<div class="chat-container">
  <!-- Chat Icon -->
  <div class="chat-icon" (click)="toggleChat()">
    <img *ngIf="!chatVisible" [src]=" '../../../assets/images/chat-bot-icon.png'"
      alt="Chatbot Icon"
      class="icon-image"
    />
  </div>

  <!-- Chat Window -->
  <div class="chat-box" [ngClass]="{'active': chatVisible}">
    <div class="chat-header">
      <h3>CodeBoost AI Chatbot</h3>
      <img src="../../../assets/images/mini-chat-bot-icon.png" class="mini-chat-bot-icon" alt="">
      <button class="close-btn" (click)="toggleChat()">×</button>
    </div>
     Artemii Aleksenko
    <div class="chat-content" #chatContent>
      <div *ngFor="let message of messages" [ngClass]="{'from-bot': message.isBot, 'from-user': !message.isBot}">
        <div class="message" [innerHTML]="message.text"></div>
      </div>
    </div>
    <div class="chat-input">
      <input type="text" [(ngModel)]="userMessage" (keyup.enter)="sendMessage()" placeholder="Type a message..." />
      <button (click)="sendMessage()">Send</button>
    </div>
  </div>
</div>
```

Рисунок 4.22 – Реалізація HTML-файлу

Отже, завдяки аналізу та вибору відповідних фронтенд-фреймворків і бібліотек, а також їх належній інтеграції, нами була реалізована клієнтська частина інтерактивного асистента навчального призначення. Вона забезпечує користувацький інтерфейс для взаємодії з асистентом, включаючи відображення повідомлень, кнопок, полів введення та інших елементів інтерфейсу.

5 ВИКОРИСТАННЯ ЧАТ-БОТУ

5.1 Загальний огляд інтеграції чат-боту у навчальну систему

Першим кроком розглянемо результат інтеграції чат-боту у веборієнтовану навчальну систему. Після авторизації користувач за замовчуванням буде направлений на домашню сторінку, де він одразу матиме можливість взаємодіяти з інтелектуальним асистентом у правій нижній частині сторінки (рис. 5.1-5.2).

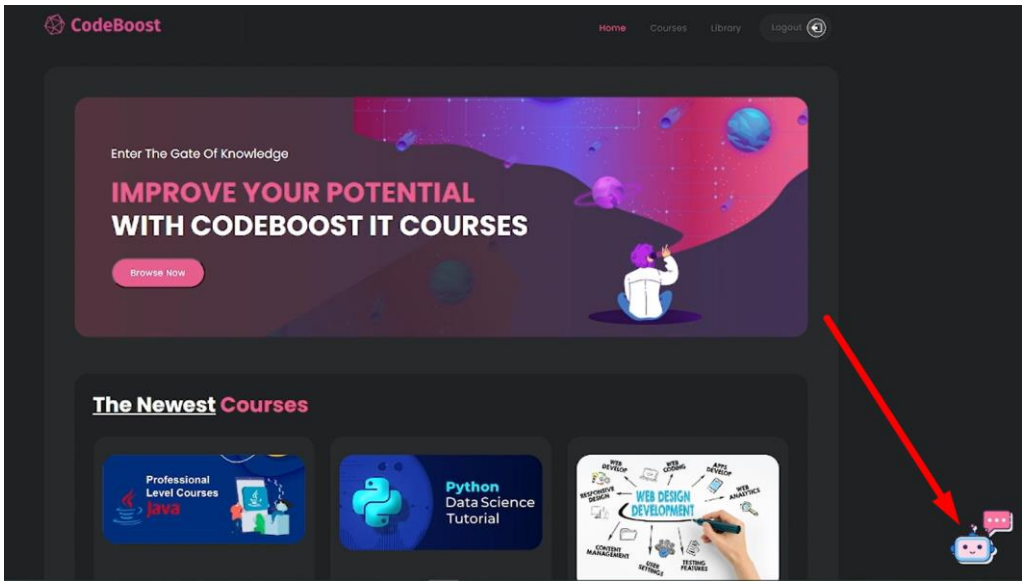

Рисунок 5.1 – Відображення згорнутого чат-бота на домашній сторінці

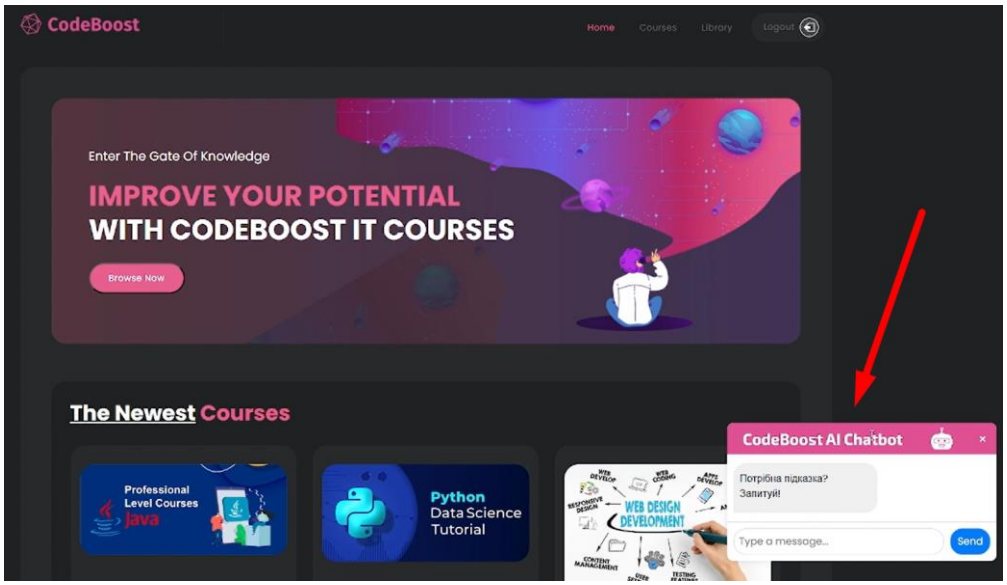

Рисунок 5.2 – Відображення розгорнутого чат-бота на домашній сторінці

Оскільки наші вимоги передбачали можливість взаємодії з асистентом з будь якої частини системи, аналогічна можливість присутня на будь-якій вебсторінці навчальної платформи. Для прикладу розглянемо сторінку з детальною інформацією про курс на рисунку 5.3.

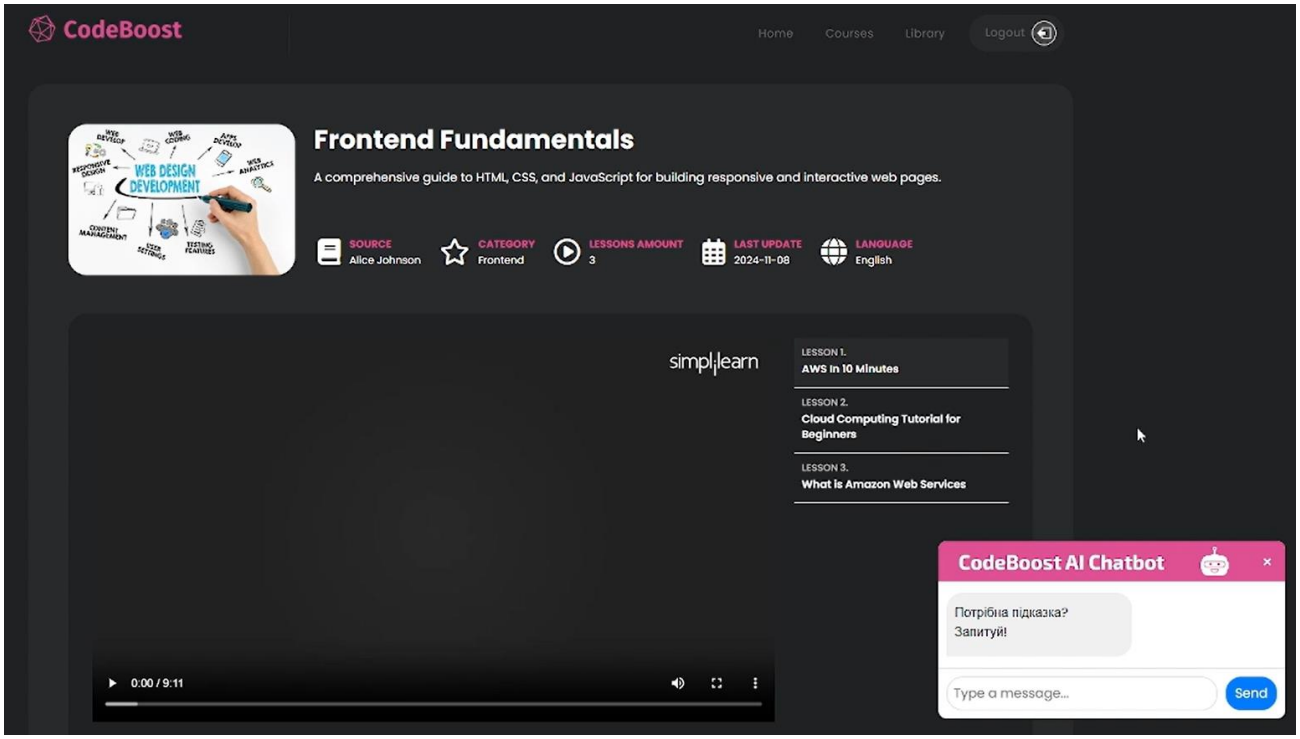

Рисунок 5.3 – Відображення розгорнутого чат-бота на сторінці з деталями курсу

5.2 Комунікація з інтелектуальним асистентом

Наступним кроком розглянемо ключову логіку асистента навчального призначення, для цього сформулюємо питання до бота «Які курси по штучному інтелекту ти мені порадиш?». Як результат асистент генерує відповідну рекомендацію і пропонує нам два курси на вибір, що найкраще підпадають під наш запит (рис. 5.4).

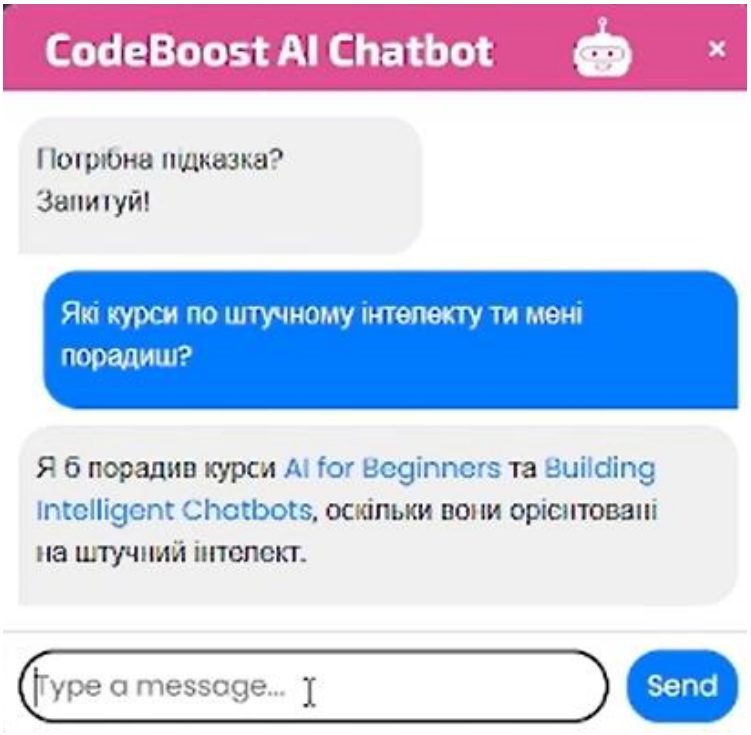

Рисунок 5.4 – Рекомендація асистента по IT-курсам

Варто зазначити, що однією із вимог було розуміння та зберігання асистентом попереднього контексту комунікації. Для тестування даного процесу задаємо уточнення відносно нашого попереднього питання «Але я новачок, що із цих курсів мені підійде?». У даному питанні немає певного контексту щодо навчальних матеріалів, його чат-бот має виокремити із попередньої комунікації і таким чином сформувати коректну відповідь про курси зі штучного інтелекту для новачків (рис. 5.5).

Також корисною функцією, що дуже пришвидшить аналіз користувачем відповідей чат-бота є гіперпосилання на рекомендований курс. При натисканні на назву запропонованого курсу або літератури користувач буду переадресований на сторінку деталей відповідного навчального матеріалу (рис. 5.6).

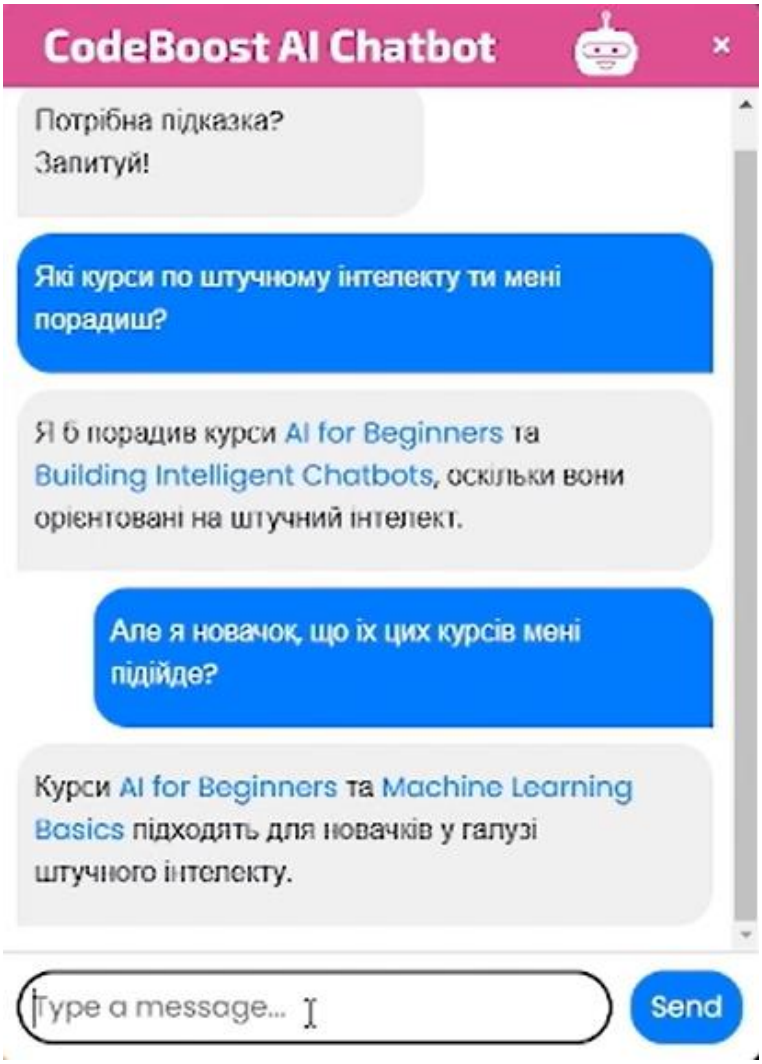

Рисунок 5.5 – Рекомендація асистента на уточнення попереднього питання

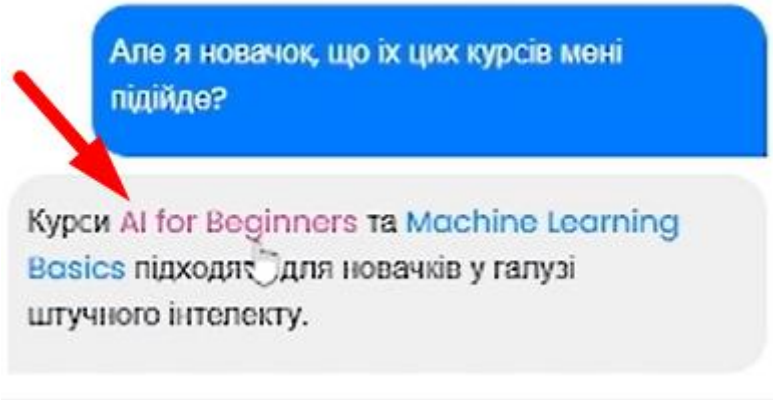

Рисунок 5.6 – Гіперпосилання на деталі навчального курсу

Фінальним етапом тестування є запит на рекомендації іншого типу навчального матеріалу, а саме літератури. Для цього сформуємо відповідний запит «Яку літературу ти мені порадиш про штучний інтелект?». У якості

відповіді отримуємо наступну рекомендацію від нашого чат-боту (рис. 5.7). Варто зазначити, що усі властивості асистента: розуміння контексту, збереження історії та гіперпосилання на ресурси аналогічно функціонують і з літературними навчальними матеріалами.

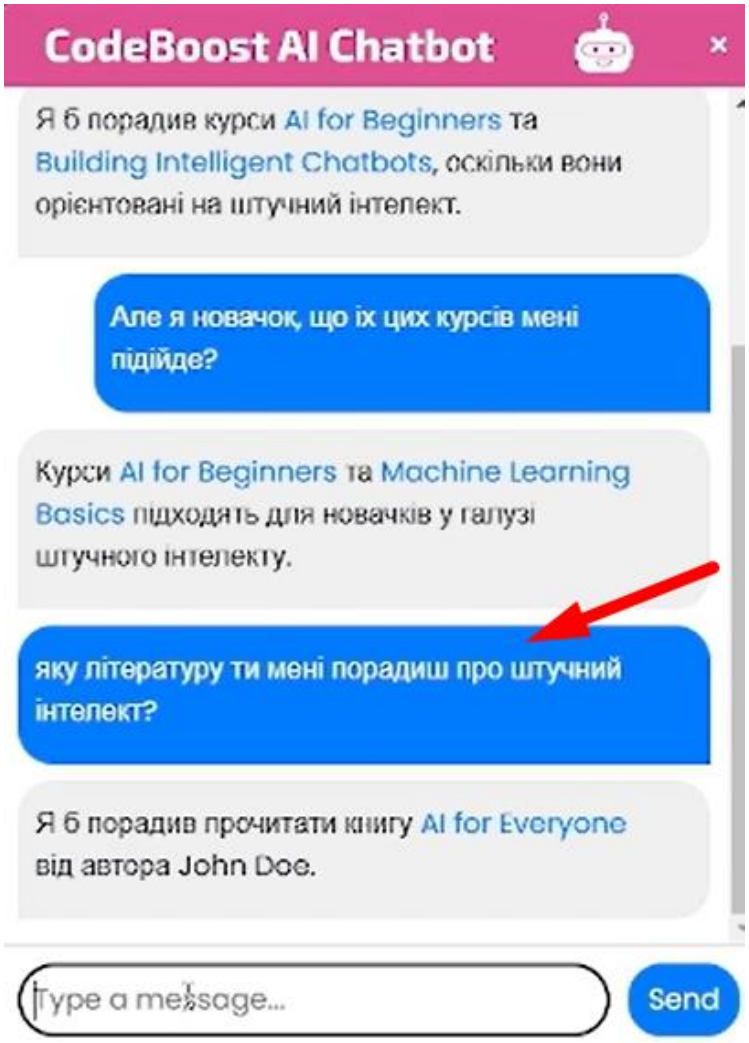

Рисунок 5.7 – Рекомендація асистента з навчальної літератури

Отже, у цьому розділі було продемонстровано ключовий функціонал інтерактивного інтелектуального асистента, включаючи його здатність аналізувати запити користувачів, надавати персоналізовані рекомендації та використовувати попередній контекст для подальшої взаємодії. Показано, як асистент інтегрується в освітню платформу, оптимізуючи вибір курсів та літератури відповідно до індивідуальних потреб користувачів.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було здійснено аналітичний огляд існуючих аналогів за обраною тематикою, зокрема таких продуктів як CustomGPT, LiveChat та Intercom. Це дозволило виявити їх сильні сторони: відсутність лімітів на комунікацію, гнучкість та персоналізація відповідей, а також недоліки, серед яких обмежене використання контексту діалогу та інтуїтивно незрозумілий інтерфейс. Розглянуто сучасні підходи до реалізації чат-ботів на основі штучного інтелекту, зокрема використання фреймворків, бібліотек, навчених мовних моделей та хмарних інструментів, які забезпечують створення програмного забезпечення відповідної якості.

На етапі проектування було змоделювано функціональні бізнес-процеси із застосуванням нотації IDEF0, а також виконано проектування програмної частини асистента з використанням діаграм: діаграми прецедентів, діаграми розгортання та діаграми послідовності, що допомогло детально описати всі аспекти системи.

Також було налаштовано хмарні інструменти для зберігання та обробки запитів користувачів, що забезпечує надійність і масштабованість додатку. Реалізовані серверна та клієнтська частини додатку, які забезпечують стабільне функціонування інтелектуального асистента та ефективну взаємодію з користувачем. Наприкінці було проведено огляд реалізованого асистента для підтвердження відповідності поставленим вимогам та якості роботи.

Результатом роботи є інтерактивний інтелектуальний асистент навчального призначення, що був інтегрований у веборієнтовану систему керування ІТ-курсами. Розроблений чат-бот повністю відповідає поставленим вимогам, забезпечує надійну роботу з великими обсягами даних і має високу здатність до масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Number and percentage of students enrolled in degree-granting postsecondary institutions, by distance education participation, location of student, level of enrollment, and control and level of institution: fall 2021 and fall 2022. *National Center for Education Statistics (NCES) Home Page, a part of the U.S. Department of Education*. URL: https://nces.ed.gov/programs/digest/d23/tables/dt23_311.15.asp (date of access: 07.11.2024).

2. Percent of students enrolled in distance education courses by distance education status of student. National Center for Education Statistics (NCES) Home Page, a part of the U.S. Department of Education. URL: https://nces.ed.gov/ipeds/TrendGenerator/app/trend-table/2/42?trending=row&cid=85 (date of access: 07.11.2024).

3. База даних "Scopus". Scopus. URL: https://www.scopus.com/authredirect.uri?txGid=dbabab26c6ea521dba199ac828d4017d (date of access: 07.11.2024).

4. AI Statistics 2024 · AIPRM. *AIPRM: Your Cheat Code for AI like ChatGPT, Claude & Midjourney · AIPRM*. URL: https://www.aiprm.com/ai-statistics/ (date of access: 07.11.2024).

5. What is an AI assistant?. *Botpress | the Complete AI Agent Platform*. URL: https://botpress.com/blog/what-is-an-ai-assistant (date of access: 07.11.2024).

6. Yasar K., Botelho B. What is a Virtual Assistant (AI Assistant)? | Definition from TechTarget. *Search Customer Experience*. URL: https://www.techtarget.com/searchcustomerexperience/definition/virtual-assistant-AI-assistant (date of access: 07.11.2024).

7. IBM. What is a Neural Network? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/neural-networks (date of access: 07.11.2024).

8. IBM. What Is a Chatbot? | IBM. *IBM - United States*.

URL: https://www.ibm.com/topics/chatbots (date of access: 07.11.2024).

9. Karthik G Vaithianathan How long will it take to train an LLM model like GPT-3? URL: https://karvai.medium.com/how-long-will-it-take-to-train-an-llm-model-like-gpt-3-d48407198077 (date of access: 07.11.2024).

10. Kleinman Z. Microsoft: 'ever present' personal AI assistants are coming. *BBC Home - Breaking News, World News, US News, Sports, Business, Innovation, Climate, Culture, Travel, Video & Audio*. URL: https://www.bbc.com/news/articles/czj9vmnlv9zo (date of access: 07.11.2024).

11. Intercom: The best AI agent built on the best customer service platform. *Intercom: The best AI agent built on the best customer service platform*. URL: https://www.intercom.com/ (date of access: 07.11.2024).

12. Live chat software for e-commerce | livechat. *LiveChat*. URL: https://www.livechat.com/ (date of access: 07.11.2024).

13. CustomGPT.ai | support & research AI agents for business. *CustomGPT*. URL: https://customgpt.ai (date of access: 07.11.2024).

14. Як перевірити швидкість завантаження сайту: онлайн-сервіси – Lemarbet. Lemarbet | Створення та розвиток інтернет-магазинів. URL: https://lemarbet.com/ua/razvitie-internet-magazina/kak-proverit-skorost-zagruzki-sajta-onlajn-servisy/ (date of access: 07.11.2024).

15. Cardillo A. List of the Best 21 Large Language Models (LLMs) (September 2024). *Exploding Topics*. URL: https://explodingtopics.com/blog/list-of-llms (date of access: 07.11.2024).

16. Top AI Website & AI Tools in October 2024 - Toolify. *Best AI Tools Directory & AI Tools List - Toolify*. URL: https://www.toolify.ai/Best-trending-AI-Tools (date of access: 07.11.2024).

17. What is LangChain? - LangChain Explained - AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/what-is/langchain/?nc1=h_ls (date of access: 07.11.2024).

18. 5 best programming languages for artificial intelligence (AI) in

2024. *Index • Hire Vetted Software Developers and other Tech Talent from the Top 5%*. URL: https://www.index.dev/blog/top-ai-programming-languages-2024 (date of access: 07.11.2024).

19. What Is Flask Used for? (9 Use Cases) - PLANEKS. *PLANEKS*. URL: https://www.planeks.net/what-is-flask-used-for/ (date of access: 07.11.2024).

20. Алексенко А. SumDU Repository: інформаційна веборієнтована система керування навчальними курсами. *SumDU Repository: Home*. URL: https://essuir.sumdu.edu.ua/handle/123456789/92806 (дата звернення: 07.11.2024).

21. UML Use Case Diagram Tutorial. *Lucidchart*. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (date of access: 07.11.2024).

22. What is IDEF - Definition, Methods, and Benefits - Edraw. *[OFFICIAL] Edraw Software: Unlock Diagram Possibilities*. URL: https://www.edrawsoft.com/what-is-idef.html (date of access: 07.11.2024).

23. What is Deployment Diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (date of access: 07.11.2024).

24. What is Sequence Diagram?. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ (date of access: 07.11.2024).

25. OpenAI Docs. *OpenAI developer platform*. URL: https://platform.openai.com/docs/overview (date of access: 07.11.2024).

26. Understanding indexes - Pinecone Docs. *Pinecone Documentation - Pinecone Docs*. URL: https://docs.pinecone.io/guides/indexes/understanding-indexes (date of access: 07.11.2024).

ДОДАТОК А

```
import os

from dotenv import load_dotenv
from langchain_community.document_loaders.sql_database import SQLDatabaseLoader
from langchain_community.utilities import SQLDatabase
from langchain_openai import OpenAIEmbeddings
from langchain_pinecone import PineconeVectorStore
from langchain_text_splitters import CharacterTextSplitter

load_dotenv()

course_query_course = "SELECT title, description, language, category FROM course.course"
course_query_book = "SELECT title, author, description, language, category FROM
library.book"

if __name__ == "__main__":
  print("Start uploading data...")

  # Init CodeBoost database connection
  db = SQLDatabase.from_uri(database_uri=os.environ.get("DATABASE_URL"))

  # Load  document
  loader = SQLDatabaseLoader(query=course_query_course, db=db)
  document = loader.load()

  # Split entire data into chunks
  text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
  texts = text_splitter.split_documents(document)
  print(f"created {len(texts)} chunks")

  # Create vector embeddings and save it in pinecone database
  embeddings = OpenAIEmbeddings(
     openai_api_type=os.environ.get("OPENAI_API_KEY"))
  PineconeVectorStore.from_documents(texts, embeddings,
                              index_name=os.environ.get("INDEX_NAME"))

  print("Finish uploading data...")
```

ДОДАТОК Б

```
import os

from dotenv import load_dotenv
from flask import Flask, request, jsonify
from flask_cors import CORS
from langchain.chains.conversational_retrieval.base import \
  ConversationalRetrievalChain
from langchain_community.chat_models import ChatOpenAI
from langchain_openai import OpenAIEmbeddings
from langchain_pinecone import PineconeVectorStore

load_dotenv()
chat_history = []
app = Flask(__name__)

# Allow CORS for all domains
CORS(app)

embeddings = OpenAIEmbeddings(openai_api_type=os.environ.get("OPENAI_API_KEY"))
vectorstore = PineconeVectorStore(
   index_name=os.environ["INDEX_NAME"], embedding=embeddings
)

chat = ChatOpenAI(verbose=True, temperature=0, model_name="gpt-3.5-turbo")

qa = ConversationalRetrievalChain.from_llm(
   llm=chat, chain_type="stuff", retriever=vectorstore.as_retriever()
)

@app.route("/ask", methods=["POST"])
def ask_question():
  data = request.json
  question = data.get("question")

  if not question:
     return jsonify({"error": "No question provided"}), 400

  response = qa.invoke({"question": question, "chat_history": chat_history})
  history = (response["question"], response["answer"])

  chat_history.append(history)

  response = jsonify({"response": response})
  response.headers.add('Access-Control-Allow-Origin', '*')

  return response

if __name__ == "__main__":
  app.run(debug=True)
```