

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**  
**Факультет комп'ютерних наук, фізики та математики**  
**Кафедра комп'ютерних наук та програмної інженерії**

**Створення інтерактивного помічника студента зі застосуванням**  
**Штучного Інтелекту**

**Кваліфікаційна робота (проект)**  
на здобуття ступеня вищої освіти «магістр»

Виконав: студент 1 магістерського курсу  
241м групи

Спеціальності: 121 Інженерія  
програмного забезпечення

Освітньо-професійної програми:

Інженерія програмного забезпечення

Карпов І.В.

Керівник: Ректор Херсонського  
державного університету, професор

Співаковський О.В.

Доцент, кандидат фізико-  
математичних наук

Єрмолаєв В.А.

Рецензент: фріланс, full stack developer

Кльонон Дмитро Михайлович

## ЗМІСТ

<b>ВСТУП.....</b>	<b>3</b>
<b>РОЗДІЛ 1. Overview, Related work та SWOT аналіз чат-ботів зі штучним інтелектом.....</b>	<b>6</b>
1.1. Огляд різновидів чат-ботів зі штучним інтелектом.	6
1.2. Related works чат-ботів зі штучним інтелектом .....	12
1.3. SWOT-аналіз чат-ботів конкурентів .....	20
<b>РОЗДІЛ 2. Технічна специфікація чат-бота «Єва».....</b>	<b>28</b>
2.1. Виявлення та формування вимог .....	28
2.2. Підготовка до проєктування діаграм.....	32
2.3. Проєктування діаграм та їх опис.....	51
<b>РОЗДІЛ 3. Реалізація технічної специфікації інтерактивного помічника «Єва».....</b>	<b>59</b>
3.1. Опис розробки інтерактивного помічника зі штучним інтелектом на основі технічної специфікації .....	59
3.2. Проєктування інтерактивного помічника «Єва» та його тестування .....	62
<b>ВИСНОВКИ.....</b>	<b>73</b>
<b>References .....</b>	<b>74</b>
<b>ДОДАТКИ</b>	
Додаток А Кодекс академічної доброчесності.....	78
Додаток Б .....	80

## ВСТУП

Штучний інтелект з кожним роком став набувати усе більше популярності серед різних сфер діяльності. І з кожним роком розвиватиметься до великих масштабів з точки зору бізнес аналітики. Сьогодні ми можемо в інтернеті взаємодіяти зі штучним інтелектом з функцією чат-бота на будь яку тему, від звичайного повсякденного спілкування до обговорення якихось складних тем з фізики чи попросити, щоб чат-бот провів огляд вашого програмного коду(останній пункт дуже популярний серед програмістів на сьогодні).

Станом на сьогодні, ми можемо спостерігати конкуренцію чат-ботів зі штучним інтелектом. Якщо два роки тому ми могли гратись з ChatGPT, то станом на зараз спроектовано та розроблено багато нових штучних інтелектів, які спрямовані на різні сфери діяльності. Є медійні, текстові, графічні та багато різних видів штучного інтелекту і всі вони доступні користувачам інтернету. Наприклад медійний штучний інтелект українці почали використовувати для створення діпфейків з відомими людьми під час війни, наприклад відеозвернення президента України з унікальним текстом, який написав користувач а штучний інтелект читає текст голосом президента і робить анімацію мимики губ так, щоб було схоже на те, що ці слова дійсно вимовляє президент. Або, наприклад художній штучний інтелект. Користувач пише штучному інтелекту запит, наприклад намалювати краєвид з видом на гори а під горами місто і штучний інтелект виконуватиме цей запит у будь-якому вигляді і форматі.

Оскільки два роки тому ми зустрічали ChatGPT, то сьогодні ми можемо скористатись:

- Google Bard – це чат-бот на основі моделі Google, який забезпечує інтерактивне спілкування та підтримку для користувачів у різних сферах(так само, як і ChatGPT).
- Microsoft Bing Chat – чат-бот створений на основі самої моделі ChatGPT версії 4.0 та інтегрований з пошуковою системою Bing. Доступний як на персональному комп'ютері, так і у смартфонах. Для взаємодії потрібен браузер від компанії Microsoft – Edge.
- Claude – це чат-бот, який був розроблений компанією Anthropic, він також відомий своєю орієнтацією на безпеку а також на етичні аспекти використання штучного інтелекту.

Більшість різновидів ми розглянемо у розділі 1.

Якщо пірнути у історію[1][23], то ідея штучного інтелекту бере свій початок ще з античних часів, але повноцінні перші реальні спроби створити щось, що ми сьогодні спокійно називаємо штучним інтелектом, з'явилося в середині двадцятого століття. Першим суттєвим кроком до сучасного штучного інтелекту можна вважати створення так званої «логічної машини» у 1950-х роках британським математиком Аланом Тюрінгом[39]. Рівно у 1950 році Тюрінг опублікував наукову статтю під назвою «Обчислювальні машини та розум» («Computing Machinery and Intelligence»), у якій він запропонував свій знаменитий в математиці Тест Тюрінга у якості критерію для визначення, чи є машина інтелектуальною. Наукова стаття Тюрінга заклала теоретичний фундамент для досліджень в галузі штучного інтелекту.[38]

Сам термін «штучний інтелект» вперше був використаний через шість років після публікації наукової статті Алана Тюрінга (у 1956 році) під час Дартмутської конференції, яка була організована Джоном Маккарті, Натанаєлем Рочестером, Марвіном Мінським та Клодом Шенноном. Ця конференція стала вважатись відправною точкою для офіційного початку досліджень у сфері штучного інтелекту. І таким

чином, розвиток штучного інтелекту розпочався вже в середині 20-го століття, і з тих пір пройшов величезний шлях від простих логічних машин до сучасних нейронних мереж і мовних моделей, як той самий легендарний ChatGPT.

**Актуальність теми даної кваліфікаційної роботи** полягає у тому, щоб зробити огляд чат-ботів зі штучним інтелектом конкурентів чат-боту «Єва», щоб описати їх функціонал та зробити це завдяки «SWOT analysis» та «Related works». На основі цих аналізів розробити технічну специфікацію, до якої входять виявлені та сформовані вимоги до чат-бота «Єва», «Use case» діаграми, на яких будуть зображені й описані ролі користувачів та функціонал чат-бота «Єва».

**Наша мета й завдання кваліфікаційної роботи** розробити на основі сформованої технічної специфікації чат-бот «Єва» з функцією надання допомоги та консультації студентам та абітурієнтам.

**Об'єктом дослідження** є аналіз чат-ботів конкурентів, на основі яких буде сформована технічна специфікація. На основі технічної специфікації буде розроблений інтерактивний помічник студента зі штучним інтелектом «Єва».

**Предметом дослідження** в нашій роботі буде технічна специфікація, яка буде вміщати в себе сформовані вимоги до розробки інтерактивного помічника студента із штучним інтелектом, «UML» діаграми, в яких будуть зображені ролі користувачів та принцип роботи майбутнього чат-бота «Єва».

## Розділ 1.

### **Overview, Related work та SWOT аналіз чат-ботів зі штучним інтелектом**

Цей розділ присвячено розгляду аналогів чат-бота зі штучним інтелектом «Єва». Ми розглянемо їх особливості та порівняємо між собою, виявимо позитивні та негативні риси та зробимо це за допомогою SWOT аналізу.

#### **1.1 Огляд різновидів чат-ботів зі штучним інтелектом**

Перш ніж почнемо наш огляд на існуючі на сьогодні, давайте трохи дізнаємось про історичний розвиток штучного інтелекту[25], бо як відомо, штучний інтелект[33] [34] пройшов довгий шлях від теоритичних ідей до сучасних систем, які здатні обробляти великі обсяги даних, навчатися на основі досвіду та вирішувати складні завдання від користувачів. Задовго до появи комп'ютерів, мислителі розмірковували про механічні пристрої, які мали б можливість імітувати людське мислення. У сімнадцятому столітті філософи, такі, як Рене Декарт та Томас Гоббс, розмірковували про обчислення, як процес, схожий на людське мислення. А у 1800-х роках британський математик Джордж Буль розробив алгебру, яка стала основою логічних операцій, що використовуються в комп'ютерах і посяк день. Дана «булева логіка» являється основою для будь-якого цифрового обчислення.

Як ми вже обговорювали у вступі, у 1950 році британський математик та криптограф Аллан Тюрінг написав відому наукову роботу «Computing Machinery and Intelligence». В цій науковій роботі Тюрінг запропонував ідею «Тесту Тюрінга», який міг тестувати на здатність машини імітувати людський інтелект. Але, у той ж час американський математик Клод Шеннон почав дослідження, як створити програму для

гри в шахи, що, власне стало однією з ранніх спроб розробити інтелектуальні системи. 1956 рік можна сміливо вважати офіційним роком народження штучного інтелекту, як наукової дисципліни. В тому році відбулася конференція вчених, серед яких був Джон Маккарті, Марвін Мінскі, Нейтан Рочестер та Клод Шеннон. Ця група вчених запропонувала дослідження в галузі штучного інтелекту. Після цієї події до 1960-х років почався процес створення перших програм для гри в шахи, включаючи «Turochamp» Аллана Тюрінга.[24] Також, з'явилися програми, ціль яких було вирішення проблем на основі простих правил логіки. Перші подібні системи могли розв'язувати математичні завдання або виконувати логічні операції. Період 1974-1980 роки прозвали «Зимовим періодом» через те, що на той час через технічні обмеження, бо в 1960-х та 70-х роках обчислювальна потужність була недостатньою для реалізації складних алгоритмів та великих обсягів даних, програми, такі як експертні системи, зіткнулися з обмеженнями у масштабуванні, коли додавання нових правил до системи призводило до суттєвих проблем в обробці і виведенні результатів. Більшість програм і проектів, таких як ранні системи машинного перекладу, зазнали невдач, що спричинило розчарування у спонсорів та науковців, наприклад, відомий проект ALPAC (Automated Language Processing Advisory Committee)[2] у 1966 році визнав, що машини не здатні ефективно перекладати тексти, що і призвело до скорочення фінансування подібних досліджень. Саме розчарування у прогресі штучного інтелекту стало наслідком різкого скорочення державного і приватного фінансування. На той момент уряди та корпорації почали інвестування коштів в інші технології, зокрема в апаратне забезпечення та обчислювальні системи, інтерес до штучного інтелекту почав падати. [10] У 1980-х роках з'явилися системи, які могли допомагати фахівцям у прийнятті рішень. Системи використовували бази знань, які було створено відповідними експертами. Найвідомішими стали «MYCIN» для діагностики хвороб і

«XCON» для конфігурації самих комп'ютерів. У той ж час було розроблено нові методи, на яких базувалися принципи роботи мозку, - штучні нейронні мережі. Одним з піонерів штучних нейронних мереж став Джеффри Гінтон, який розробив метод зворотного поширення помилок для навчання нейронних мереж.

Інтерактивні помічники для студентів, які використовують штучний інтелект (ШІ)[42], стають все більш популярними в освітніх закладах по всьому світу. Ці технології активно інтегруються в навчальний процес для поліпшення досвіду студентів. Наприклад, Atlas — це один із таких помічників, який допомагає студентам у навчанні. Цей помічник створено спеціально для того, щоб надавати точні й контекстуально правильні відповіді на основі матеріалів курсу студента, зокрема підручників, лекцій і завдань. Atlas також інтегрує персоналізовану базу знань, яка зберігає прогрес студента, що дозволяє краще адаптуватися до його потреб під час навчання. Крім того, Atlas дозволяє студентам співпрацювати й обмінюватися знаннями між собою, що створює підтримуючу спільноту навчання. Також, багато університетів використовують розмовні ШІ-помічники для надання студентам оперативної підтримки, відповідей на адміністративні питання та навіть допомоги в процесі вступу до університету. Такі помічники допомагають розвантажити адміністративний персонал і надають студентам доступ до інформації в будь-який час доби. Таким чином, впровадження ШІ-помічників у навчальні заклади стає важливою складовою сучасної освіти, що сприяє більш ефективному та персоналізованому навчальному процесу.

Крім Atlas, є ще кілька подібних інтерактивних помічників, які використовують штучний інтелект для підтримки студентів:



- ChatGPT-стилізовані помічники: В деяких університетах вже впроваджуються помічники на базі таких моделей, як ChatGPT. Вони використовуються для відповідей на питання студентів, допомоги з навчальними матеріалами та навіть для автоматизації адміністративних завдань. Ці помічники можуть допомагати студентам у реальному часі та надавати персоналізовану підтримку, адаптуючись до індивідуальних потреб кожного студента.
- DuoLingo: Хоча цей інструмент спочатку був розроблений для вивчення мов, його алгоритми ШІ значно покращують ефективність навчання, адаптуючи завдання до рівня знань користувача. Це дозволяє студентам отримувати відповідну інформацію й підтримку, яка допомагає їм рухатися вперед у навчанні.
- Squirrel AI: Це платформа, яка використовує адаптивні технології для створення персоналізованих навчальних шляхів. Squirrel AI активно застосовується в китайських школах і коледжах, де вона аналізує прогрес студентів та коригує навчальні плани відповідно до їхніх потреб, що робить навчання більш ефективним.

Таким чином, освітні платформи, які використовують ШІ, стають важливим інструментом для створення більш персоналізованого та інтерактивного навчального процесу, забезпечуючи підтримку для студентів на всіх етапах їхньої освіти.

Також є кілька відомих chatGPT-стилізованих помічників, які активно використовуються у сфері освіти, такі як:

- **Purdue University's Chatbot[16]:** Університет Пердью у США впровадив chatGPT-помічника для надання допомоги студентам. Цей чатбот відповідає на питання, пов'язані з адміністративними процесами, наприклад, реєстрацією на курси, отриманням фінансової допомоги, і навіть дає поради щодо підготовки до іспитів. Університет Пердью використовує платформу Ivu.ai для створення інтерактивного чатбота, який надає студентам адміністративну та технічну підтримку. Ivu.ai забезпечує відповіді на питання про реєстрацію на курси, фінансову допомогу, розклад занять та інші запити. Завдяки штучному інтелекту, Ivu.ai може працювати цілодобово, надаючи студентам швидкий доступ до необхідної інформації. Крім того, цей чатбот допомагає університетам оптимізувати комунікацію зі студентами, зменшуючи навантаження на адміністративний персонал і покращуючи загальну якість обслуговування студентів. Ivu.ai використовується у багатьох інших навчальних закладах і пропонує широкий спектр інтеграцій, що робить його популярним рішенням у сфері вищої освіти.

### **Основні функції:**

- 1 Автоматичні відповіді на часті запити
- 2 Цілодобова підтримка студентів
- 3 Інтеграція з різними університетськими платформами для доступу до ресурсів.

Цей чат-бот допомагає не лише студентам, а й розвантажує адміністративний персонал, оптимізуючи роботу університету.

- **Socratic[19]:** це освітня платформа, створена для студентів середніх шкіл та університетів, яка використовує технології штучного інтелекту (в тому числі GPT) для допомоги у вирішенні складних навчальних питань. Студенти можуть сфотографувати завдання або задати текстове питання, а Socratic надасть відповідь, пояснення чи навчальні ресурси, які допоможуть краще зрозуміти тему. Вона особливо популярна для вивчення предметів, таких як математика, хімія та фізика.

Основні функції:

- 1 Можливість сфотографувати завдання для отримання допомоги
  - 2 Пояснення складних концепцій через приклади
  - 3 Персоналізовані навчальні ресурси
- **Packback[13]:** Це освітня платформа, яка використовує AI, схожий на chatGPT, для створення інтерактивних дискусій у класах. Packback дозволяє студентам задавати питання й отримувати на них відповіді в реальному часі, сприяючи активному обговоренню та покращенню розуміння матеріалу.
  - **Ivy.ai[9]:** це потужна платформа штучного інтелекту, створена в першу чергу для вищих навчальних закладів, що забезпечує вдосконалені чат-боти для автоматизації спілкування між студентами та адміністрацією. Платформа вже навчена на мільйонах запитань і відповідей з університетських сайтів, що дозволяє швидко надавати персоналізовані відповіді.

Основні можливості Ivu.ai включають:

- 1 **Мультиканальна комунікація:** Ivu.ai інтегрується з різними платформами, такими як SMS, електронна пошта, Facebook Messenger та навіть голосові помічники на кшталт Amazon Alexa, щоб забезпечити студентам різноманітні способи зв'язку.
- 2 **Автоматизація та ефективність:** Чат-бот автоматично відповідає на часті запити студентів, наприклад, щодо балансу рахунку або фінансової допомоги, що знижує навантаження на адміністрацію та покращує швидкість і якість обслуговування. Платформа також інтегрується з понад 30 системами університету, що дозволяє легко отримувати дані та виконувати завдання між відділами.
- 3 **Аналітика даних:** Ivu.ai збирає корисні дані в реальному часі про взаємодію студентів, що допомагає навчальним закладам краще розуміти їхні потреби та коригувати стратегії взаємодії.
- 4 **Доступність та інклюзивність:** Платформа сприяє рівному доступу до інформації для всіх студентів, незалежно від їхніх обставин або походження.

Ці chatGPT-стилізовані помічники значно покращують доступність навчання та надають студентам можливість отримувати своєчасну допомогу, що робить їх навчальний досвід більш персоналізованим та ефективним.

## **1.2 Related works чат-ботів зі штучним інтелектом**

Related works – це дуже важливий елемент будь-якого дослідницького проекту або наукової роботи, бо він полягає у перегляді та аналізі попередніх досліджень та робіт, які стосуються теми нашого дослідження. Оскільки у нас Related works стосується

інтерактивних помічників студента зі штучним інтелектом, то такий огляд виконує кілька важливих функцій та надає численні переваги. Наведемо основні причини, чому нам необхідно робити Related works:

### 1. Контекст і розуміння поточного стану дослідження

- Огляд попередніх робіт допомагає зрозуміти, які дослідження вже проводилися, які методи використовувалися та які результати було отримано. Це допомагає нам, як дослідникам чітко побачити, де знаходиться їхня тема в межах широкого контексту досліджень та які є прогалини.

### 2. Запобігання дублюванню

- Related works дозволяє уникнути повторення вже проведених досліджень, так як він надає огляд на те, що вже зроблено в даній галузі. Це допомагає спрямувати дослідницькі зусилля на нові, ще не вивчені аспекти.

### 3. Ідентифікація існуючих рішень та підходів

- Огляд інтерактивних помічників студента зі штучним інтелектом дозволяє розробникам або дослідникам побачити, які рішення вже були створені для студентів. Це може включати аналіз таких систем, як Ivi.ai, Packback або інших відомих чат-ботів для освіти.
- Важливо зрозуміти, які технології та алгоритми (NLP, машинне навчання) вже були використані в цих рішеннях,

щоб не повторювати ті ж самі підходи або вибрати оптимальний для свого проекту.

#### 4. Виявлення переваг та недоліків існуючих систем

- Related works допомагає виявити сильні та слабкі сторони вже існуючих інтерактивних помічників. Наприклад, деякі системи можуть бути дуже ефективними в обробці природної мови, але мати проблеми з інтеграцією у певні освітні платформи.
- Оцінка успішних та неуспішних аспектів інтерактивних помічників дозволяє уникнути повторення помилок та запозичити вдалі рішення.

#### 5. Аналіз досвіду користувачів (User Experience)

- Огляд попередніх робіт допомагає краще зрозуміти потреби студентів і оцінити, як інтерактивні помічники вже використовуються у реальних умовах. Це включає в себе вивчення зворотного зв'язку від користувачів та User Experience (UX).
- Важливо врахувати досвід використання таких помічників, як-от проблеми доступності або студентські відгуки, щоб зробити нову систему більш доступною та зручною.

#### 6. Визначення майбутніх напрямків розвитку

- Related works допомагає визначити прогалини у дослідженнях і можливі напрями для покращення або розвитку. Наприклад, якщо виявлено, що більшість помічників не забезпечують достатньої адаптивності до навчальних потреб, це може стати точкою розвитку нової системи. Також це може відкрити нові можливості для

інтеграції, наприклад, з когнітивними обчисленнями або віртуальними реальностями.

#### 7. Оцінка ефективності технічних рішень

- Огляд попередніх робіт дозволяє побачити, які алгоритми, інфраструктура та технології найкраще підходять для реалізації інтерактивних помічників у контексті освітнього процесу. Наприклад, можна дізнатися про ефективність різних моделей обробки природної мови (GPT, BERT) або використання конкретних інструментів, як Azure AI. Це допомагає зробити вибір найбільш оптимальних технологій для розробки нових рішень.

#### 8. Обґрунтування необхідності нового рішення

- Аналіз наявних рішень показує, чому потрібно створити нового або вдосконаленого помічника. Якщо існуючі рішення мають обмеження, новий продукт може бути спрямований на подолання цих проблем, які зроблять його більш затребуваним та актуальним.

#### 9. Конкурентний аналіз

- Related works дає можливість зробити конкурентний аналіз, визначивши слабкі та сильні сторони конкурентів і знайти свою унікальну нішу на ринку інтерактивних помічників. Це допомагає побудувати стратегію розвитку нового продукту на основі конкурентних переваг.

Таким чином, Related works допомагає забезпечити більш глибоке розуміння поточного стану розробок, мінімізувати ризики дублювання,

підвищити якість та конкурентоспроможність нового рішення, а також спрямувати його на подолання поточних викликів та обмежень у сфері освітніх інтерактивних помічників зі штучним інтелектом. А тепер розглянемо наш Related works до розглянутих нами аналогів інтерактивних помічників студента зі штучним інтелектом.

## 1. Цільова аудиторія та застосування

- Ivy.ai[11] [12]: Призначений для університетів і коледжів з основною метою надання адміністративної допомоги студентам і технічної підтримки. Чат-бот допомагає у вирішенні питань, пов'язаних із курсами, реєстрацією, фінансовою допомогою, і може бути інтегрований з університетськими системами. Основна ціль — оптимізація комунікації між студентами й адміністрацією університету.
- Socratic (Google)[20]: Створений для студентів середньої та вищої школи, з акцентом на допомогу в навчанні. Використовується для пояснення складних концепцій і надання студентам відповідей на питання з різних предметів, таких як математика, хімія і фізика. Він інтегрує технології штучного інтелекту для полегшення навчального процесу та розуміння студентами матеріалу.
- Paskback[14] [15]: Основна аудиторія — університети та студенти, які беруть участь в інтерактивних обговореннях у класі або онлайн. Paskback фокусується на розвитку критичного мислення через постановку відкритих питань, що



сприяють дискусіям між студентами. Викладачі можуть контролювати ці обговорення та надавати фідбек, тоді як штучний інтелект стимулює студентів задавати запитання і брати активну участь.

## 2. Функціональність

- Ivu.ai: Найсильніша сторона — універсальна підтримка адміністративних процесів у навчальних закладах. Її функції спрямовані на автоматизацію рутинних запитів студентів (наприклад, як зареєструватися на курс чи подати заявку на стипендію), що допомагає зменшити навантаження на університетський персонал. Чат-бот підтримує інтеграцію з платформами управління навчанням (LMS) та іншими внутрішніми системами університетів.
- Socratic: Її основна функція полягає у допомозі з конкретними питаннями та завданнями. Використовуючи камеру телефону, студенти можуть фотографувати завдання, і Socratic[21] надасть пояснення або рішення. Це корисно для самостійного навчання, але не передбачає живих дискусій або участі викладачів, як у Packback.
- Packback: Платформа фокусується на стимулюванні дискусій та запитань у форматі класної кімнати або онлайн-навчання. Її функції включають AI-модерацію та оцінку якості запитань студентів, забезпечення зворотного зв'язку та підвищення рівня залученості студентів до навчання. Packback також дозволяє

викладачам спостерігати за дискусіями та направляти їх, але не надає індивідуальних відповідей на конкретні навчальні питання, як це робить Socratic.

### 3. Метод взаємодії

- Ivu.ai: Взаємодія відбувається через текстовий інтерфейс, де студент вводить питання і отримує автоматизовані відповіді, часто пов'язані з адміністративними або технічними процесами. Його основна функція — надання швидкої та точної інформації про університетські процедури.
- Socratic: Студенти взаємодіють з платформою через фото і текст. Вони можуть фотографувати або друкувати питання, і штучний інтелект надає пояснення та рішення. Ця платформа більше зосереджена на індивідуальній підтримці студента під час навчального процесу.
- Paskback: Взаємодія відбувається через обговорення в класі або онлайн, де студенти задають питання, відповідають на них і ведуть дискусії. Викладачі й AI модерують ці дискусії, забезпечуючи зворотній зв'язок для покращення навчального процесу.

### 4. Роль AI

- Ivu.ai: ШІ відповідає за автоматизацію адміністративних завдань і швидко надання інформації студентам, що значно

розвантажує адміністративний персонал. Він може інтегруватися з існуючими системами університету для доступу до внутрішньої інформації.

- Socratic: ШІ забезпечує аналіз зображень і тексту для створення навчальних матеріалів, які допомагають студентам вирішувати завдання і краще розуміти навчальні теми. Алгоритми GPT полегшують цей процес, автоматично пояснюючи складні концепції.[37]
- Packback: Штучний Інтелект використовується для модерації дискусій, визначення якості питань студентів і заохочення критичного мислення. Він автоматично генерує фідбек і стимулює студентів ставити більш складні та вдумливі питання.

Підсумок порівняння:

- Ivu.ai найкраще підходить для адміністративних завдань і підтримки студентів з організаційними питаннями.
- Socratic забезпечує індивідуальну навчальну підтримку, допомагаючи студентам вирішувати завдання та розуміти складні теми через алгоритми ШІ.
- Packback фокусується на групових дискусіях та розвитку критичного мислення, підтримуючи активну взаємодію між студентами та викладачами через AI-аналіз.

### 1.3 SWOT-аналіз чат-ботів конкурентів

SWOT-аналіз являється важливим інструментом для стратегічного планування, котре дозволяє оцінити внутрішні та зовнішні чинники, які впливають на проект, ініціативу або компанію. Даний метод використовується для прийняття рішень, планування розвитку, аналізу конкурентного середовища та виявлення сильних і слабких сторін, загроз і можливостей.

#### Компоненти SWOT-аналізу:

##### 1. Сильні сторони (або ж Strengths):

- Внутрішні переваги або унікальні ресурси, які забезпечують конкурентні переваги. Це можуть бути високоякісні продукти, ефективні бізнес-процеси, значний досвід або ж добре розвинена команда.
- Приклад: компанія може мати унікальну технологію або сильний бренд.

##### 2. Слабкі сторони (або ж Weaknesses):

- Внутрішні недоліки або обмеження, котрі можуть стримувати розвиток ефективність організації. Це можуть бути недосконалі продукти, відсутність досвіду або обмежений доступ до ресурсів.
- Приклад: низька впізнаваність бренду, нестача фінансування.

##### 3. Можливості (або ж Opportunities):

- Зовнішні чинники, котрі можуть сприяти покращенню або ж зростанню результатів. Це можуть бути технічні

інновації, нові ринки, зміни у споживних вподобаннях або нові регуляторні політики.

- Приклад: зростаючий попит на екологічно чисті продукти.

#### 4. Загрози (або ж Threats):

- Зовнішні чинники, які можуть створювати ризики для організації, такі як нові економічні кризи, зміни в законодавстві, нові конкуренти або технологічні зрушення.
- Приклад: зміни в ринкових умовах або посилення конкуренції.

Чому SWOT-аналіз важливий? Ну, почнемо з того, що SWOT-аналіз дозволяє виявити, які сильні сторони можна використовувати для захоплення можливостей, а які слабкі сторони потрібно усунути для мінімізації загроз. Також, аналіз допомагає керівникам та менеджерам компаній ухвалювати рішення на основі комплексного розуміння як внутрішніх чинників (сильні та слабкі сторони), так і зовнішніх (можливості та загрози). Також, SWOT-аналіз дозволяє швидко реагувати на зміни в ринковому середовищі, відстежуючи потенційні загрози та можливості, такі як нові конкуренти або зміни в економіці. Аналіз також допомагає зрозуміти, на які аспекти бізнесу варто звернути увагу, щоб максимально ефективно використовувати свої сильні сторони і мінімізувати вплив слабких сторін. І на додачу, SWOT-аналіз дозволяє оцінити своє місце на ринку порівняно з конкурентами та виявити ключові відмінності або переваги. Як ви можете помітити, SWOT-аналіз є неймовірно корисним інструментом для будь-якої організації, котра хоче розвиватися, ефективно використовувати свої можливості та мінімізувати ризики для досягнення успіху. Отже розглянемо SWOT-аналіз до наших розглянутих інтерактивних помічників.

## Strengths (Сильні сторони)

### 1. Ivy.ai:

- Сильна адміністративна підтримка для університетів.
- Інтеграція з внутрішніми системами, такими як платформи управління навчанням (LMS).
- Швидке надання студентам відповідей на запити щодо реєстрації, фінансової допомоги та іншої організаційної інформації.
- Зменшення навантаження на адміністративний персонал.

### 2. Socratic:

- Підходить для швидкого та персоналізованого вирішення завдань і пояснення складних концепцій через AI.
- Використовує функцію сканування зображень, що робить взаємодію з платформою дуже простою.
- Широкий спектр предметів, що охоплюються (математика, хімія, фізика тощо).

### 3. Packback:

- Стимулює активні дискусії в університетському середовищі.
- AI аналізує та оцінює якість запитань студентів, заохочуючи критичне мислення.

- Платформа забезпечує інтерактивність між студентами та викладачами.

#### 4. ChatGPT (Azure AI Studio):

- Найбільш універсальний варіант серед платформ: може бути налаштований на різноманітні завдання.
- Потужна кастомізація через Azure AI Studio для створення унікальних моделей для будь-яких потреб.
- Можливість інтеграції з іншими інструментами та службами Azure для покращення роботи з великими наборами даних.

### **Weaknesses (Слабкі сторони)**

#### 1. Ivy.ai:

- Основний фокус на адміністративні питання, обмежена функціональність у навчальних процесах.
- Відсутність інтерактивних освітніх функцій або підтримки для поглибленого навчання.

#### 2. Socratic:

- Обмежений у взаємодії з навчальним процесом — це більше індивідуальний помічник, а не дискусійна платформа.
- Відсутність підтримки складних дискусій або інтерактивних функцій, як у Packback.

### 3. Packback:

- Наголос на дискусіях обмежує використання платформи в індивідуальному навчанні.
- Потрібен постійний нагляд викладачів, щоб забезпечити якість обговорень.

### 4. ChatGPT (Azure AI Studio):

- Можливість створення дуже складних моделей вимагає значних технічних знань.
- Без налаштувань ChatGPT може бути занадто загальним і менш ефективним у вузьких навчальних застосуваннях.

## **Opportunities (Можливості)**

### 1. Ivy.ai:

- Може розширити свою функціональність для включення освітньої підтримки, наприклад, допомога з матеріалами для курсів або оцінка успішності.
- Розвиток інтеграцій з популярними освітніми платформами.



## 2. Socratic:

- Інтеграція з університетськими системами для покращення взаємодії зі студентами та викладачами.
- Розширення можливостей для більш складних завдань та створення інструментів для вивчення нових предметів.

## 3. Packback:

- Можливе впровадження індивідуальних навчальних програм або персоналізованих планів навчання.
- Інтеграція з LMS для автоматизованого оцінювання та обліку студентської активності.

## 4. ChatGPT (Azure AI Studio):

- Розширення в університетах для підтримки як індивідуального навчання, так і групових дискусій.
- Потенційні можливості для інтеграції з іншими сервісами Microsoft (наприклад, Teams, OneNote) для підтримки навчання та співпраці в класі.

## Threats (Загрози)

### 1. Ivy.ai:

- Конкуренція з іншими адміністративними чат-ботами або системами управління навчальними процесами, які мають більше освітніх функцій.
- Залежність від внутрішніх інтеграцій університету, що може бути технічно складним або обмеженим.

### 2. Socratic:

- Відставання в розвитку у порівнянні з більш гнучкими платформами, такими як ChatGPT.
- Висока залежність від конкретних предметних областей і шкільної аудиторії.

### 3. Packback:

- Можливий ризик, що студенти будуть уникати глибоких дискусій або користуватися платформою лише формально.
- Інші платформи з кращою інтерактивністю можуть перевершити Packback.

#### 4. ChatGPT (Azure AI Studio):

- Залежність від налаштувань і технічних знань може відлякати менш технічно підкованих користувачів.
- Конкуренція зі спеціалізованими платформами, такими як Socratic і Packback, може ускладнити позиціонування на ринку без належної кастомізації.

Висновок щодо даних чатботів зі штучним інтелектом:

ChatGPT на Azure AI Studio є найбільш гнучким і багатофункціональним рішенням, яке може адаптуватися під різні навчальні та адміністративні потреби. Проте спеціалізовані рішення, такі як Ivy.ai, Socratic, і Packback, мають свої сильні сторони, коли йдеться про певні завдання (адміністративна підтримка, допомога в навчанні або стимулювання дискусій).

## РОЗДІЛ 2.

### Технічна специфікація чат-бота «Єва»

В цьому розділі ми виявимо список технічних та не технічних вимог до майбутнього інтерактивного помічника студента зі штучним інтелектом «Єва», розглянемо етапи розробки та створимо прототип.

#### 2.1. Виявлення та формування вимог

Для створення інтерактивного помічника студента на основі штучного інтелекту важливо врахувати як технічні, так і не технічні вимоги. Ці вимоги допоможуть нам забезпечити ефективність роботи системи, а також її прийняття користувачами, в нашому випадку це студенти, абітурієнти та працівники університету. Отже, почнемо.

Технічні вимоги:

1. Обробка природної мови (або ж Natural language processing)
  - Помічник повинен підтримувати алгоритми обробки природної мови для розуміння та формування відповідей природньою мовою.
  - Підтримка багатомовності для забезпечення взаємодії з користувачами різних мовних груп.
2. Глибоке та машинне навчання:
  - Використання глибокого навчання для навчання моделей великими наборами даних, щоб покращити точність відповідей на питання користувачів.
  - Постійне навчання системи на основі нових даних.

### 3. Інтеграція з базами даних та іншими системами:

- Помічник має бути здатний отримувати доступ до баз даних і зовнішніх систем для отримання актуальної інформації, як-от інформація про навчальні програми, контактні дані деканатів факультетів університету, посилання на актуальні розклади пар, дзвінків тощо.

### 4. Персоналізація:

- Система повинна зберігати профіль студента (наприклад, поточні курси, оцінки) для надання персоналізованих відповідей та рекомендацій.
- Рекомендації можуть також стосуватися навчальних матеріалів або графіку підготовки студентів до іспитів.

### 5. Текстовий інтерфейс:

- Інтерактивний помічник має підтримувати текстові команди для забезпечення зручності взаємодії з різними групами студентів університету.

### 6. Реальний час обробки даних:

- Помічник повинен працювати в режимі реального часу з мінімальною затримкою для забезпечення швидкої відповіді на запити користувачів.

### 7. Взаємодія через API:

- Забезпечення можливості інтеграції через API для взаємодії з іншими програмними рішеннями, мобільними застосунками а також веб-сервісами.

## 8. Захист даних:

- Забезпечити високий рівень безпеки даних та конфіденційності, зокрема через шифрування та відповідність нормативним вимогам, наприклад, як GDPR або FERPA, щоб надавати захист приватної інформації студентів.

## 9. Можливість розширення (тобто, Scalability)

- Система має бути розроблена з можливістю легкого масштабування для підтримки великої кількості одночасних користувачів і нових функцій.

## 10. Аналіз контексту:

- Розпізнавання контексту запитів для надання більш релевантних і персоналізованих відповідей (розуміння минулих взаємодій з користувачем).

## 11. Адаптивність:

- Система повинна адаптуватися до зміни контексту студентських запитів, розуміючи повторювані або змінені запити без необхідності вводити заново всю інформацію.

## 12. Інтеграція з календарями:

- Система повинна бути інтегрована з календарями для синхронізації важливих дат (наприклад, екзамени, дедлайни завдань або зустрічі з викладачами).

Не технічні вимоги:

1. Зручність використання (тобто, User Experience):
  - Інтерфейс інтерактивного помічника повинен бути інтуїтивно зрозумілим, з функцією очищення листування, щоб користувачі могли за необхідності чистити історію листування з інтерактивним помічником.
  
2. Ефективність комунікації:
  - Важливо, щоб відповіді були чіткими, лаконічними та зрозумілими користувачу. Уникнення надмірної кількості інформації, котра може перевантажити користувача.
  
3. Можливості навчання користувача:
  - Інтерактивний помічник має вітатись з користувачем, назвати своє ім'я, за необхідністю, пояснювати користувачеві як працює система та як нею користуватись. Це важливо для нових користувачів.
  
4. Мотиваційні функції:
  - Система повинна включати елементи мотивації, такі як нагадування про дедлайни, заохочення до досягнення цілей, поради з тайм-менеджменту.
  
5. Зворотній зв'язок:
  - Система повинна надавати студентам залишати зворотній зв'язок про ефективність її роботи, щоб розробники мали можливість покращувати функціонал.

#### 6. Підтримка емоційного інтелекту:

- Надати інтерактивному помічнику можливість розпізнавання емоцій користувача на основі тексту і надавати відповідну реакцію для створення більш людяного та приємного досвіду.

#### 7. Етичність:

- Дотримання етичних стандартів при взаємодії з користувачем, включаючи уникання упереджень і забезпечення прозорості щодо того, як і чому система приймає певні рішення.

#### 8. Відповідність культурним особистостям:

- Інтерактивний помічник має враховувати культурні особливості, адаптуючи спілкування відповідно до релігійних норм та традицій.

На основі сформульованих та виявлених вимогах ми можемо приступити до проектування нашого майбутнього інтерактивного помічника зі штучним інтелектом. Розпишемо етапи розробки даного помічника та створимо його прототип і протестуємо, щоб зрозуміти чого нам не вистачає, щоб створити повноцінного інтерактивного помічника.

## **2.2. Підготовка до проектування діаграм**

Діаграми відіграють дуже ключову роль у процесі розробки програмного забезпечення, оскільки вони допомагають організувати, візуалізувати та пояснити структуру і функціонування системи. Діаграми дозволяють розробникам краще зрозуміти архітектуру системи та її компоненти. Вони являються спрощеним способом візуалізації складних процесів, які роблять їх доступними для всіх учасників проекту, незалежно від рівня технічних знань. Наприклад, діаграма послідовності показує, як компоненти системи взаємодіють між собою, що полегшує розуміння загальної картини. Також діаграми надають



структуру для планування і проектування системи. Вони можуть слугувати планом для розробників, допомагаючи точно визначити, як різні модулі чи сервіси повинні бути інтегровані. Наприклад, діаграма варіантів використання (Use Case Diagram) допомагає визначити основні сценарії взаємодії користувачів із системою і фокусуватися на вимогах до функціональності. У процесі розробки програмного забезпечення діаграми є ефективним засобом комунікації між членами команди. Вони допомагають забезпечити однакове розуміння вимог, логіки та архітектури системи. Візуальні презентації полегшують обговорення і усунення непорозумінь у командах. Діаграми слугують важливою частиною технічної документації, яку розробники можуть використовувати для подальшої підтримки, модернізації або розширення системи. Вони допомагають не тільки в поточному розвитку, а й під час майбутньої роботи з кодом, коли нові розробники можуть легко зрозуміти, як система працює. Діаграми також дозволяють виявити можливі архітектурні проблеми на ранніх етапах розробки. Наприклад, під час проектування архітектурних діаграм можна побачити потенційні вузькі місця, які можуть негативно вплинути на продуктивність або безпеку системи. До того ж, діаграми допомагають переконатися, що всі функції та сценарії взаємодії користувача враховані, а система розробляється у відповідності до специфікації та бізнес-вимог. Це важливо для запобігання невідповідностей між тим, що було заплановано, і тим, що реалізовано.

В нашому проекті фігуруватимуть такі діаграми:

- Діаграми варіантів використання (Use Case Diagrams)[30] [31] – це інструмент моделювання, який використовується в розробці програмного забезпечення для опису функціональних можливостей системи та взаємодій між користувачами (вони ж актори) та системою. Дана діаграма належить до мови моделювання (Unified Modeling Language), яка є стандартом для побудови моделей програмного забезпечення. Основними елементами діграми варіантів використання є:
  1. Актори – це користувачі або інші системи, які взаємодіють із системою. Акторами можуть бути як люди, так і інші системи або ж сервіси. В діаграмі актори зображуються у вигляді людини або простого кружечка із назвою над ним.
  2. Варіанти використання (Use cases) – це функціональні можливості, які система надає акторам. Кожен варіант використання описує певну задачу або функцію, яку система може виконати у відповідь дії користувача. В діаграмі варіанти використання зображуються у вигляді овалів, всередині яких знаходяться назви сценаріїв.
  3. Зв'язки:
    - Асоціація(Association) – це зв'язок між актором та варіантом використання. Вона показує, що певний актор взаємодіє з певним варіантом використання.

- Інклюзія(Include) використовується для позначення того, що один варіант використання завжди включає інший варіант як частину своєї поведінки.
- Розширення (Extend) показує, що варіант використання може бути розширений додатковими сценаріями за певних умов.

#### 4. Система

- В діаграмі зображується у вигляді прямокутника, всередині якого знаходяться варіанти використання, які представляють функції системи. Актори знаходяться за межами системного прямокутника, взаємодіючи із системою через варіанти використання.

Отже, для чого нам потрібна діаграма варіантів використання?

1. Візуалізація вимог: Діаграми варіантів використання допомагають зрозуміти, які функції має виконувати система з точки зору користувача. Це дозволяє зацікавленим сторонам та розробникам краще уявити, як система буде працювати та які задачі вона вирішує.
2. Зрозумілий опис для всіх учасників проекту: Дана діаграма легко зрозуміла як технічним спеціалістам (архітекторам, розробникам), так і нетехнічним учасникам проекту (замовникам, менеджерам). Вона є універсальним способом опису функцій системи.

3. Моделювання взаємодій: Діаграма варіантів використання показує, як актори взаємодіють із системою та дозволяє продумати сценарії використання системи на практиці. Це може допомогти виявити упущені або зайві функції на ранніх стадіях проектування.
4. Визначення основних функцій: Дана діаграма допомагає розробникам та замовникам узгодити перелік основних функцій, які повинна використовувати система та впорядковувати процес розробки. Також це може бути основою для побудови тестових сценаріїв.
5. Планування системи: Діаграма варіантів використання допомагає на етапі планування визначити, які компоненти або модулі необхідно розробити для підтримки кожного сценарію використання та сприяє розподілу завдань між членами команди.

Наведемо невеличкий приклад використання діаграми варіантів використання:

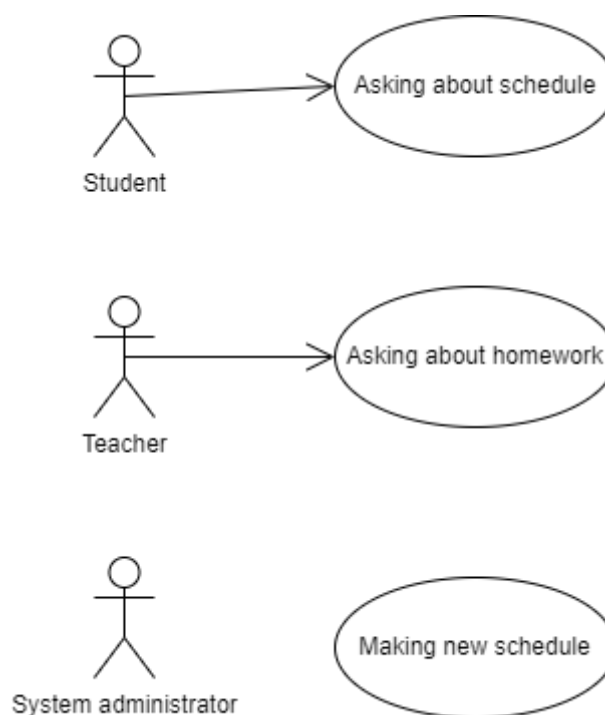


Рисунок 2.1. Приклад використання діаграми варіантів використання

У випадку рисунку 2.1 ми можемо спостерігати таких акторів як: студент, викладач та системний адміністратор. Студент асоціюється з запитом про розклад, викладач асоціюється з запитом про домашнє завдання та системний адміністратор має асоціацію з випадком «робити новий розклад». Таким чином, діаграма варіантів використання дає можливість побачити, як різні актори взаємодіють із системою і які функції необхідні для забезпечення цих взаємодій. Такі діаграми являються важливим інструментом на етапі аналізу та проектування системи, оскільки вони дозволяють всім учасникам проекту зосередитися на функціональних вимогах, зрозуміти, що саме система повинна виконувати та підготувати основу для подальших етапів розробки.

- Діаграми послідовностей (Sequence Diagrams)[28] [29] – це один із ключових типів діаграм в UML (Unified Modeling Language), який використовується для опису динамічної взаємодії між різними об'єктами системи з акцентом на порядок обміну повідомленнями між ними. Основна мета даної діаграми – це візуалізувати процес взаємодії системних компонентів або акторів у часі для вирішення певної задачі. Із основних елементів діаграми послідовностей можна відмітити такі:

1. Актори та об'єкти:

- Актор – елемент, котрий взаємодіє із системою(наприклад, інша система або користувач).
- Об'єкт – елемент, котрий представляє частину системи, яка бере участь у взаємодії, наприклад компонент або модуль системи.
- Актори та об'єкти в діаграмі зображуються у вигляді прямокутників зверху та розташовуються по горизонталі.

2. Життєва лінія (Lifeline) – це вертикальна пунктирна лінія в діаграмі послідовностей, яка виходить із актора або об'єкта. Вона показує існування об'єкта протягом часу на діаграмі. Чим довша жива лінія, тим довше актор або об'єкт існує в рамках певного сценарію.

3. Повідомлення (Messages) – комунікації, які передаються між об'єктами та акторами для виконання певних дій. Вони позначаються в діаграмі стрілками, які йдуть від одного об'єкта до іншого та позначають послідовність виклику методів або передачі даних. Повідомлення можуть бути Синхронними (де об'єкт, який надіслав запит, чекає на відповідь) та Асинхронними (де відповідь не потрібна негайно).
4. Активація (Activation) – це в діаграмі послідовностей прямокутники на життєвій лінії, які вказують на період часу, коли об'єкт виконує певну операцію або дію. Активізація починається тоді, коли об'єкт отримує повідомлення та триває, поки об'єкт обробляє запит.
5. Повернення (Return) – це стрілка в діаграмі послідовностей, яка вказує на те, що система або об'єкт завершив дію та передає відповідь або результат.

Тепер давайте розглянемо для чого нам потрібні діаграми послідовностей:

#### 1. Моделювання динамічної поведінки системи:

- Діаграми послідовностей використовуються для детального моделювання того, як компоненти системи взаємодіють один з одним, що дає уявлення про хід процесу в часі. Це особливо

важливо для складних систем, де взаємодія між компонентами може бути складною.

## 2. Візуалізація взаємодії компонентів:

- Діаграма дозволяє побачити, як окремі елементи (актори, модулі, системи) співпрацюють у рамках певної задачі. Це може бути важливо для виявлення проблем у дизайні системи або в логіці обміну даними.

## 3. Опис алгоритмів та бізнес-процесів:

- Діаграма послідовностей може використовуватися для моделювання алгоритмів (як на рівні програмного коду, так і на рівні бізнес-логіки). Це зручно при описі роботи складних процесів, які залежать від послідовності виконання кроків.

## 4. Аналіз часових взаємодій:

- Діаграма послідовностей демонструє часові аспекти взаємодії між компонентами, які можуть допомогти виявити проблеми з продуктивністю або визначити, де можливі вузькі місця.



## 5. Комунікація між бізнес-аналітиками та розробниками:

- Діаграма є зрозумілим інструментом як для технічних спеціалістів (тобто, розробників), так і для нетехнічних учасників проекту (менеджерів, бізнес-аналітиків), допомагаючи пояснити функціональність та взаємодію системи.

## 6. Підготовка до написання коду:

- Дана діаграма може служити основою для програмістів при реалізації взаємодій у кодї, оскільки вона показує, які методи викликаються і в якій послідовності.

Нижче наведемо невеличкий приклад використання діаграми послідовності:

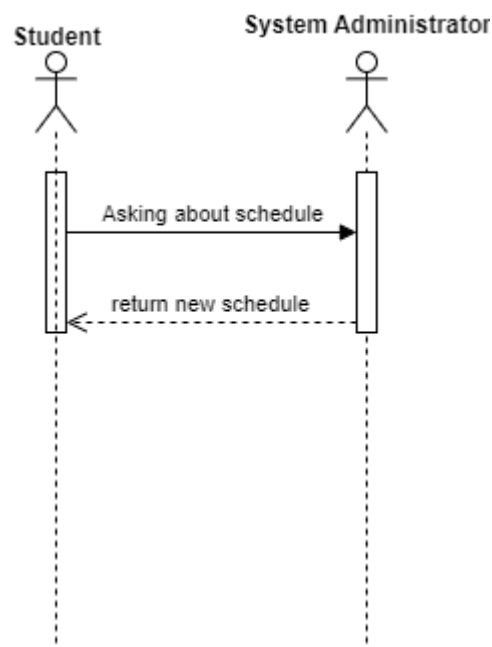


Рисунок 2.2 Приклад використання діаграми послідовності

Як ми можемо побачити в простому прикладі, ми бачимо акторів Студент та Системний адміністратор. Також, можемо звернути

увагу, що студент відправляє повідомлення актору Системний адміністратор про запит отримати розклад занять та як системний адміністратор повертає студенту новий розклад. Це максимально простий, але наглядний приклад. Діаграми послідовностей являється потужним інструментом для розробки складних систем, оскільки вони дозволяють моделювати та візуалізувати динамічну взаємодію об'єктів та процеси обміну даними. Вони корисні як на етапі проектування, так і для документування існуючих систем. Даний тип діаграм являється невід'ємною частиною аналізу систем та допомагає уникнути багатьох можливих помилок у реалізації через неправильно спроектовані або неправильно взаємодіючі компоненти.

- Діаграма розгортання (Deployment Diagram)[8] [27] – це тип діаграми в UML (Unified Modeling Language), яка моделює фізичне розгортання системи. Дана діаграма відображає, як програмні компоненти (вони ж модулі) взаємодіють з апаратним забезпеченням (пристрої, сервери) і між собою після розгортання системи. Це важливо для розуміння того, як працює система на рівні інфраструктури. Наведемо основні елементи діаграми розгортання:

1. Вузли (Nodes) – це віртуальні або фізичні пристрої (клієнтські комп'ютери, мобільні пристрої, сервери), на яких розгортаються компоненти системи. Умовно, це ті елементи, які мають потужність для обробки інформації. В UML діаграмі вузли зображаються у вигляді тривимірних блоків (кубів).

2. Артефакти (Artifacts) представляють програмні компоненти, які фізично розгортаються на вузлах. Це може бути виконуваний код, файли конфігурацій, бази даних, веб-застосунки тощо. Вони позначаються в діаграмі прямокутниками та розміщуються в межах вузлів.
3. Зв'язки (Communication Paths) – це лінії, які з'єднують вузли, вказуючи на фізичну або мережеву взаємодію між компонентами системи. Ці зв'язки демонструють, як дані передаються між вузлами (наприклад, мережеві з'єднання між клієнтами або серверами).
4. Структура вузла: Вузли можуть містити інші артефакти або вузли. Наприклад, фізичний сервер може містити контейнери або віртуальні машини, на яких працюють різні сервіси.

Отже, ми розглянули що таке діаграми розгортання. Давайте ж розберемось, для чого нам потрібні діаграми розгортання:

1. Моделювання фізичної архітектури системи:
  - Діаграма розгортання дозволяє адміністраторам систем та розробникам бачити, як компоненти програмного забезпечення будуть розгорнуті на реальних віртуальних або фізичних машинах.

## 2. Оптимізація ресурсів:

- Діаграма розгортання допомагає визначити, які фізичні ресурси потрібні для роботи системи (мережеве обладнання, сервери) та як оптимізувати використання цих ресурсів.

## 3. Аналіз продуктивності:

- Важливо розуміти, як оброблятимуться дані між різними вузлами та серверами для уникнення можливих проблем із пропускнуою здатністю або продуктивністю мережі.

## 4. Планування та розгортання в хмарних середовищах:

- Сучасні системи часто використовують контейнеризацію (наприклад, Docker) або хмарні обчислення, тому діаграми розгортання необхідні для розуміння, де та як будуть розміщені компоненти та мікросервіси у таких середовищах.

## 5. Комунікація з інфраструктурними командами:

- Діаграма розгортання є важливим інструментом для координації між командами та розробниками, які відповідають за розгортання системи на інфраструктурі, особливо в великих та складних проектах.

Розглянемо невеличкий приклад використання діаграми розгортання:

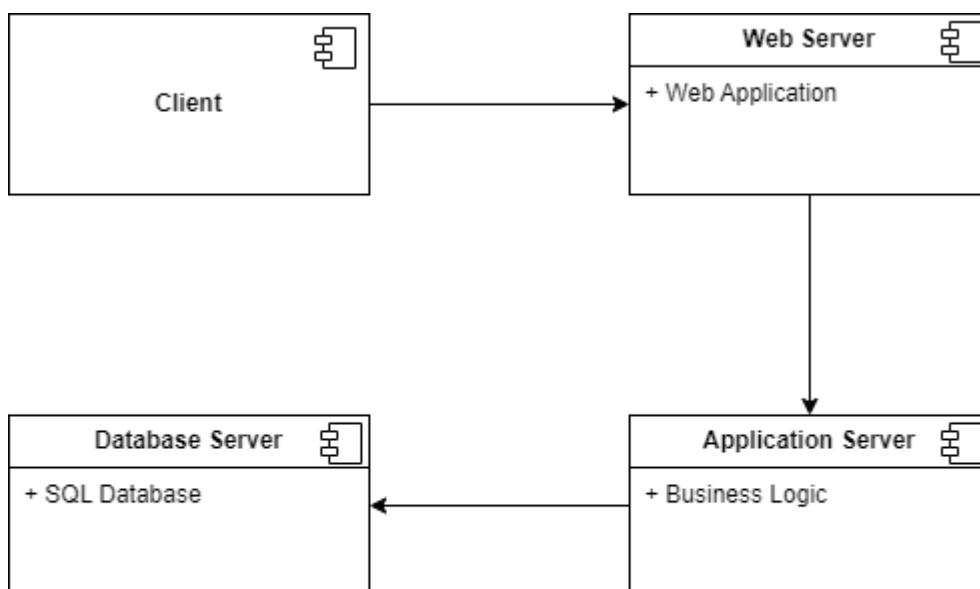


Рисунок 2.3 Приклад використання діаграми розгортання

Як ви можете бачити на рисунку 3, це спрощена версія діаграми розгортання, яку ми використовуємо як приклад до нашої теорії. Дана діаграма показує, як різні компоненти системи (бізнес-логіка, база даних та веб-застосунок) розміщуються на різних вузлах (серверах), що демонструє фізичну архітектуру системи. Отже, ми маємо чотири вузли:

1. Web Server – це сервер, на якому буде розгорнуто наш веб-застосунок, який взаємодіє з клієнтом через HTTP-з'єднання.
2. Application Server – це сервер, на якому працює бізнес-логіка системи, яка обробляє запити та виконує основні операції.
3. Database Server – це сервер, який містить базу даних (SQL Database), куди записується та зберігається інформація.
4. Client – це вузол, з якого надходять запити до сервера.

Отже, як ви можете помітити, діаграми розгортання є необхідним інструментом для Dev Ops-команд та системних архітекторів, оскільки вони дозволяють моделювати фізичну інфраструктуру системи та демонструвати, як компоненти програмного забезпечення розгорнуті на ній. Також, вони допомагають зрозуміти архітектуру системи та забезпечують кращу комунікацію між командами, відповідальними за розробку, тестування та підтримку системи.

- Діаграма компонентів (Component Diagram)[7] [26] – це один із типів діаграм в UML (Unified Modeling Language), який описує фізичну архітектуру системи на рівні компонентів. Дана діаграма фокусується на тому, як окремі частини програмного забезпечення співвідносяться одна з одною, які інтерфейси використовуються для їх взаємодії та як компоненти інтегруються у загальну архітектуру. Наведемо основні компоненти діаграми компонентів:

#### 1. Компоненти (Components):

- Це основні будівельні блоки системи. Компоненти – це автономна одиниця програмного забезпечення, яка виконує певну функцію та взаємодіє з іншими компонентами через інтерфейси. В UML діаграмі компоненти зазвичай позначаються прямокутниками з іконкою у вигляді двох невеликих прямокутників з лівого боку.

## 2. Інтерфейси:

- Інтерфейси визначають спосіб, у який компоненти взаємодіють між собою. Вони показують, як один компонент може використовувати функції іншого компонента. В діаграмі інтерфейси зображуються як кола (надання сервісу) або напівкруги (виклик сервісу).

## 3. Зв'язки (Dependencies):

- Зв'язки – це лінії, які з'єднують компоненти та відображають залежності між ними. Вони показують, як один компонент залежить від іншого (Наприклад, використовує його інтерфейси для виконання певних завдань).

## 4. Порти:

- Порти представляють точки доступу до компонента, через які інші компоненти можуть надсилати або отримувати дані.

Отже, ми розглянули діаграму компонентів з теоритичної сторони, давайте розберемося, для чого нам потрібні діаграми компонентів:

### 1. Декомпозиція системи:

- Діаграма компонентів дозволяє розділити складні програмні системи на менші, зрозумілі модулі або

компоненти. Це дозволяє краще організувати архітектуру системи та чітко визначити відповідальності кожного компонента.

## 2. Моделювання інтерфейсів та залежностей:

- Дана діаграма надає розробникам візуальне уявлення про те, які компоненти відповідають за певні функції та як вони взаємодіють між собою через інтерфейси. Це особливо важливо при роботі з великими системами, в яких залежності можуть бути складними.

## 3. Полегшення модульності та повторного використання:

- Використовуючи діаграму компонентів, можна звернути увагу, що компоненти є автономними, що спрощує їх повторне використання в інших системах або проектах. Це також допомагає підтримувати систему в майбутньому, оскільки компоненти можуть бути легко замінені або оновлені.

## 4. Комунікація між командами:

- Діаграми компонентів важливі для комунікації між різними командами розробників, які працюють над різними частинами великого проекту. Вони також допомагають чітко визначити, як окремі частини системи повинні взаємодіяти.



## 5. Забезпечення масштабованості та інтеграції:

- З розвитком системи може знадобитися інтеграція нових компонентів або модулів. Діаграми компонентів полегшують це, оскільки вони показують, які компоненти можна замінити або розширити, не порушуючи основної архітектури.

Отже, ми розібрали, для чого нам потрібні діаграми компонентів та що воно таке. Тепер, розглянемо невеличкий приклад використання діаграми компонентів:

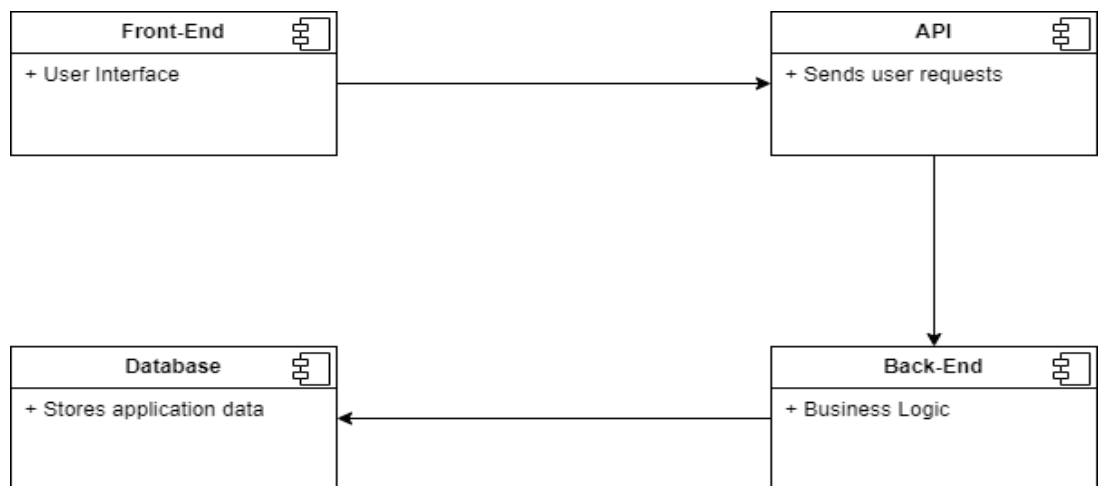


Рисунок 2.4 Приклад використання діаграми компонентів

Отже, як ви можете помітити, це приклад діаграми компонентів для розробки веб застосунку. Дана діаграма чітко розбиває компоненти веб-застосунку на ключові частини, такі як:

1. **Front-End:** Даний компонент відповідає за інтерфейс користувача, тобто взаємодію між застосунком та користувачем

2. API: Цей компонент забезпечує зв'язок між Front-End та Back-End частинами, тобто відправляє запити користувачів до Back-End сервера.
3. Back-End: Даний компонент обробляє бізнес-логіку та обробку запитів користувачів, а також виконує основну функцію програми як генерацію відповіді.
4. Database: А цей компонент зберігає усі дані програми, тобто це сховище, у якому зберігається інформація користувачів та дані веб-застосунку.

Зв'язки між даними компонентами дозволяють нам чітко візуалізувати взаємодії в системі, показуючи, як Front-End надсилає через API запити, а Back-End зберігає та отримує інформацію з бази даних. Дана діаграма демонструє, як кожен компонент має свою конкретну роль та як вони взаємодіють для досягнення загальної мети функціонування веб-застосунку.

Поговоримо трохи проте, чому ці діаграми нам важливі перед розробкою нашого проекту:

1. Візуалізація складних систем: Дані діаграми допомагають візуалізувати складні системи та зрозуміти їхню структуру та взаємодію на різних рівнях.
2. Планування архітектури: Діаграми використовуються для планування системної архітектури, яка дозволяє створити міцну й стійку структуру програмного забезпечення.
3. Спрощення комунікації: Ці діаграми забезпечують кращу комунікацію між розробниками, менеджерами, архітекторами та клієнтами, оскільки наочно показують, як працює система.

4. Аналіз проблем: Дані діаграми допомагають швидко визначити слабкі місця в системі, зокрема в архітектурі, зв'язках між модулями або в послідовності обміну даними.
5. Полегшення підтримки та розвитку: Діаграми полегшують підтримку та розвиток системи, оскільки вони документують, як компоненти працюють і взаємодіють, що дозволяє розробникам швидко знаходити та виправляти помилки.

### 2.3 Проектування діаграм та їх опис

Отже, ми розглянули необхідні нам діаграми для реалізації проекту, розглянули їх, як з теоритичної, так і з практичної сторони. А тепер перейдемо до процесу проектування діаграм:

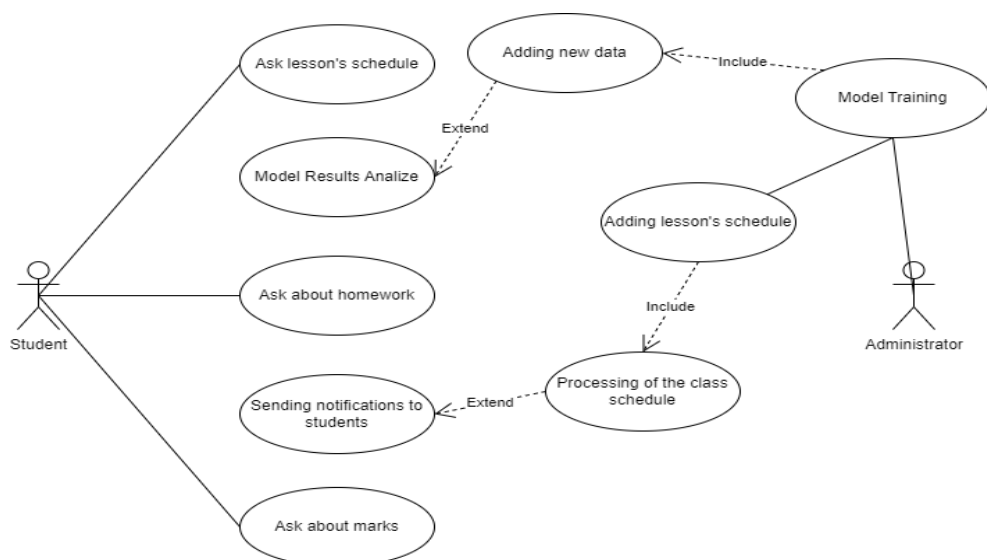


Рисунок 2.5 Діаграма варіантів використання інтерактивного помічника

## 1. Діаграма варіантів використання (Use Case)

Дана діаграма демонструє різні функціональні можливості інтерактивного помічника студента зі штучним інтелектом, зосереджуючи увагу на взаємодії користувача з системою.

Розглянемо діаграму на рисунку 5 детальніше:

- **Ask lesson's schedule** – користувач робить запит про новий розклад занять у інтерактивного помічника.
- **Model Results Analyze** – проведення аналізу результатів після додавання нових тренувальних даних.
- **Adding new data** – додавання адміністратором нових тренувальних даних до моделі.
- **Model Training** – є частиною додавання нових даних та служить для тренування моделі.
- **Ask about homework** – користувач робить запит інтерактивному помічнику про домашнє завдання.
- **Sending notifications to students** – розсилка студентам повідомлень про новий розклад занять.
- **Processing of the class schedule** – після додавання адміністратором нового розкладу в модель, вона робить після цього етапу розсилку.

- **Ask about marks** – Користувач робить запит інтерактивному помічнику про наявність списку отриманих оцінок.

Дана діаграма допомагає визначити ключові функції системи з точки зору користувача та їх взаємозв'язки.

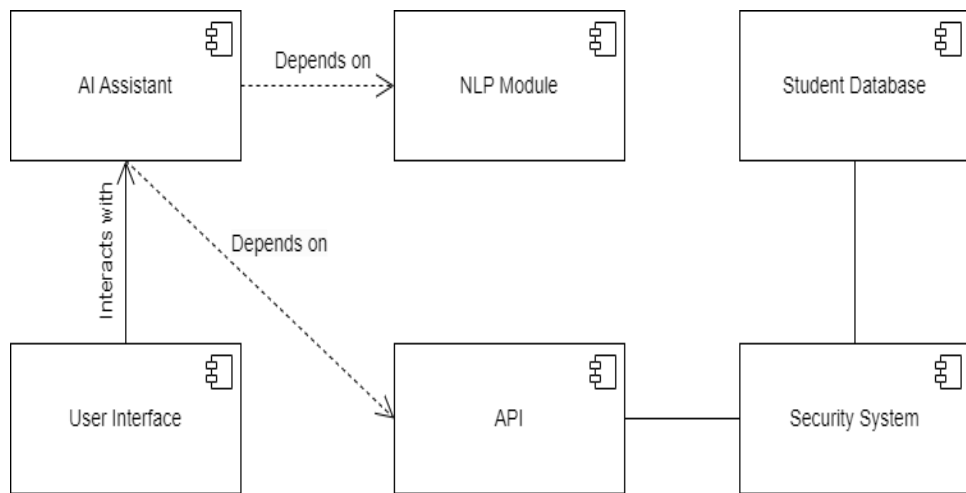


Рисунок 2.6 Діаграма компонентів до інтерактивного помічника

## 2. Діаграма компонентів (Components Diagram)

Дана діаграма на рисунку 6 описує архітектурні компоненти інтерактивного помічника та їх взаємодію між собою. Ми бачимо з вами на даній діаграмі шість компонентів, такі як:

- **AI Assistant** – це основний компонент системи, який взаємодіє з користувачем та обробляє його запити. Залежить цей компонент від компоненту, який відповідає за обробку природної мови та інтерфейсу для зв'язку між компонентами системи та зовнішніми сервісами цієї системи.

- NLP Module – це компонент, від якого залежить компонент інтерактивного помічника. Цей компонент відповідає за обробку природної мови, який якраз забезпечує розуміння запитів користувача, переводячи людську мову в машинну.
- Student Database – це база даних, яка зберігає інформацію про студентів Херсонського державного університету.
- API – цей компонент впроваджує інтерфейс для зв'язку між компонентами системи а також з зовнішніми сервісами, такі як студентська база даних, та її кібербезпека. Від даного компонента залежить також компонент інтерактивного помічника студента.
- User Interface – цей компонент відповідає за інтерфейс користувача, через який здійснюється взаємодія самим користувачем. Через цей компонент відбувається взаємодія з компонентом інтерактивного помічника студента, якому користувач дає запит та отримує на нього відповідь.
- Security System – цей компонент відповідає за систему безпеки, яка забезпечує захист даних та доступ до них. Даний компонент взаємодіє через компонент інтерфейсу для зв'язку з компонентом бази даних.

Кожен з даних компонентів виконує свою певну функцію та взаємодіє один з одним для забезпечення роботи всієї системи.

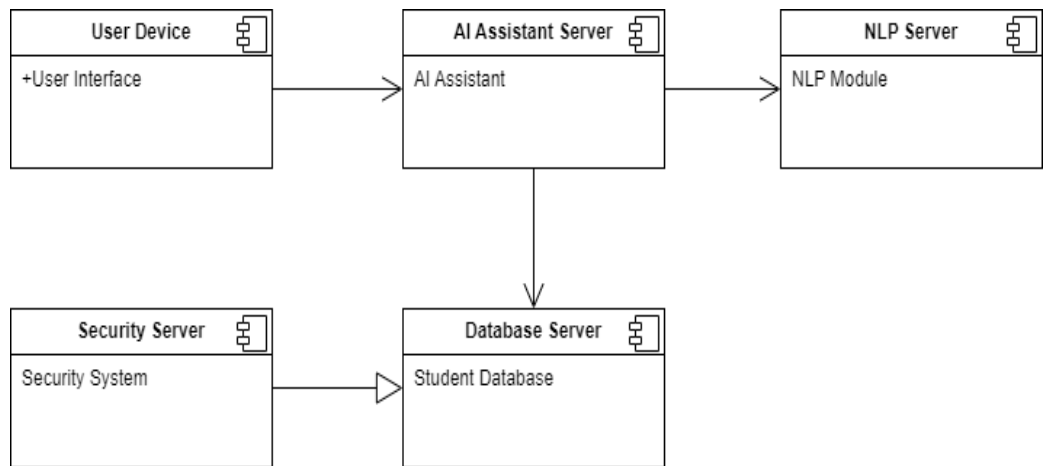


Рисунок 2.7 Діаграма розгортання інтерактивного помічника студента зі штучним інтелектом

### 3. Діаграма розгортання

Дана діаграма демонструє фізичне розташування компонентів системи на серверах. Як діаграма компонентів, діаграма розгортання дозволяє нам побачити та зрозуміти взаємодію компонентів системи між собою на різних серверах. Дана діаграма включає в себе такі компоненти, як:

- User Device – це пристрій користувача, завдяки якому здійснюється доступ до системи, тобто, до інтерфейсу користувача.
- AI Assistant Server – це сервер, у якому буде розгорнуто головний компонент інтерактивного помічника студента зі штучним інтелектом. До даного компонента здійснюється доступ з пристрою користувача та отримується запит від самого користувача для подальшої обробки та надання відповіді.

- NLP Server – це сервер, який відповідає за обробку природної мови. Після отримання запиту сервером інтерактивного помічника студента, він проходить обробку у сервері, який відповідає за обробку природної мови та повертає відповідь на запит назад до серверу інтерактивного помічника.
- Security Server – цей сервер відповідає за безпеку системи та доступу до бази даних студентів.
- Database Server – це сервер, який зберігає базу даних студентів.

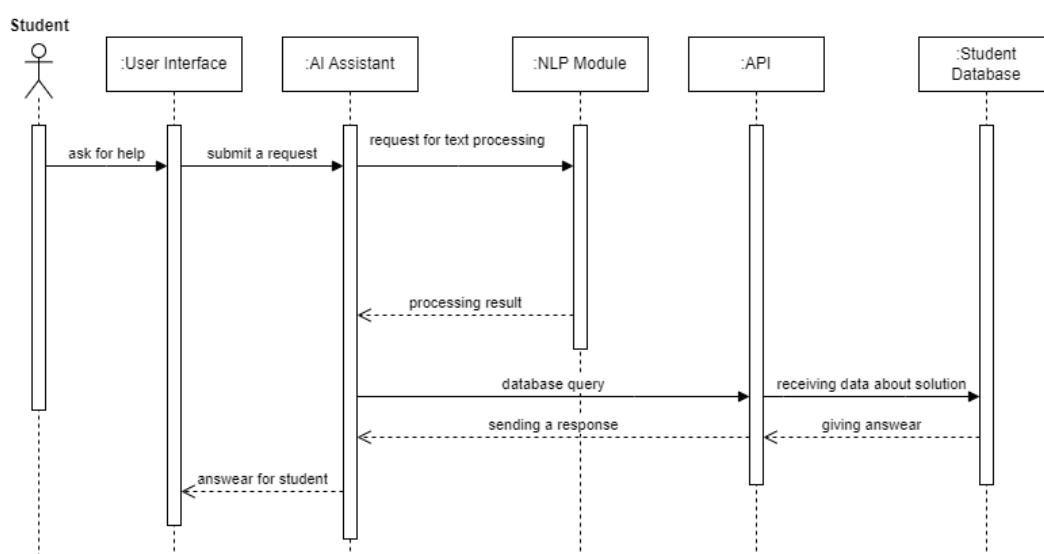


Рисунок 2.8 Діаграма послідовності інтерактивного помічника студента зі штучним інтелектом

#### 4. Діаграма послідовності

Дана діаграма детальніше ілюструє нам кожен крок, починаючи з моменту запиту користувача та закінчуючи отриманням відповіді.



У цій діаграмі чітко показано покрокову та послідовну взаємодію між компонентами. Тут ми можемо побачити знайомі нам компоненти, такі як:

- **User interface:** Актор (студент), робить запит до системи про надання допомоги, наприклад запит на отримання нового розкладу занять чи актуального розкладу дзвінків. Після надання користувачем запиту, цей запит передається до інтерактивного помічника студента.
- **AI Assistant:** Даний компонент отримує наданий користувачем запит та здійснює його передачу до NLP-модулю на обробку та аналіз мови, бо, як ми знаємо, будь-яка існуюча на сьогодні програма розуміє машинну і тільки машинну мову. Після обробки, запит повертається до інтерактивного помічника та проходить через API.
- **NLP Module:** Даний модуль обробляє запит користувача, проводить аналіз запиту та визначає, який тип відповіді очікує користувач від інтерактивного помічника. Після аналізу та типу відповіді, модуль обробки природної мови надсилає запит до бази даних для отримання необхідної інформації через API.
- **API:** Даний компонент отримує оброблений та проаналізований результат від NLP-модуля та передає його до бази даних для отримання інформації. Після передачі обробленого запиту компонент отримує його з необхідною інформацією з бази даних та надсилає його до інтерактивного помічника студента.

- Student Database: Дана база даних зберігає інформацію про кожного студента Херсонського державного університету а також про їх розклад, оцінки тощо. До неї потрапляє запит користувача через API для отримання певної інформації, оскільки база даних також слугує як модель ChatGPT.
- Відповідь користувачу: Уся зібрана інформація проходить зворотний шлях: база даних передає запит з інформацією через API, після чого запит йде до інтерактивного помічника, який у свою чергу надає відповідь на запит користувачеві, наприклад новий розклад занять або як отримати диплом.

## РОЗДІЛ 3.

### Реалізація технічної специфікації інтерактивного помічника «Єва»

#### 3.1. Опис розробки інтерактивного помічника зі штучним інтелектом на основі технічної специфікації

Отже, ми з вами дізнались невеличку історію[22] появи та розвитку штучного інтелекта, розглянули аналоги нашому інтерактивному помічнику студента зі штучним інтелектом, зробили SWOT-аналіз цих аналогів та порівняли з нашим майбутнім чат-ботом. Також, ми розглянули необхідні нам UML діаграми, розглянули їх теоретично та практично. Також, розглянули наш варіант діаграм. Тепер час приступити до створення прототипу нашого чат-бота зі штучним інтелектом. Але, перш ніж почнемо, давайте розглянемо платформу, у якій ми будемо займатись створенням прототипу інтерактивного помічника студента зі штучним інтелектом, це Azure AI Studio[3].

Azure AI Studio[4] [35] – це хмарна платформа від компанії Microsoft, яка призначена для розгортання, розробки та тестування штучного інтелекту. Вона об'єднує широкий набір моделей та інструментів, зокрема від OpenAI та інших партнерів (наприклад Google Gemini), щоб спростити процес створення власних рішень штучного інтелекту. Azure AI Studio дозволяє створювати та інтегрувати моделі у свої проекти як досвідченим, так і розробникам-початківцям. Дана платформа є частиною екосистеми Microsoft Azure AI, яка розвивається як частина стратегії компанії з демократизації технологій штучного інтелекту. Спочатку компанія Microsoft запускала базові сервіси для машинного навчання, але з часом, коли потреба в більш складних та персоналізованих рішеннях штучного інтелекту зросла, компанія

розширювала можливості платформи, додавши підтримку генеративних моделей, як GPT від компанії OpenAI. Azure AI Studio[5] надає доступ до потужних інструментів, таких як Azure Machine Learning, які дозволяють легко налаштовувати моделі, працювати з великими наборами даних, та розгортати їх у реальному часі. Наразі студія активно використовується для багатьох галузей, включаючи охорону здоров'я, обслуговування клієнтів, освіту та багато іншого. Azure AI Studio наразі забезпечує такі можливості як:

1. **Простота у використанні:** Завдяки інтуїтивно зрозумілому інтерфейсу та вбудованим моделям, навіть користувачі без глибоких технічних знань можуть створювати застосунки зі штучним інтелектом.
2. **Широкий функціонал:** Студія дозволяє працювати з обробкою природної мови (NLP), генерацією мовлення, комп'ютерним зором, прогнозуваннями тощо.
3. **Гнучкість для розробників:** Досвідчені розробники можуть інтегрувати власний код, використовувати потужні API, налаштування моделі для більш точного налаштування рішень.
4. **Масштабованість:** Azure AI Studio дозволяє створювати моделі, які можуть працювати з великими обсягами даних і бути інтегрованими у бізнес-процеси будь-якої складності.

Як ви могли помітити, Azure AI Studio надає багатий набір моделей та інструментів для їхнього налаштування, також, завдяки інтуїтивному інтерфейсу дає можливість новачкам розібратись з платформою набагато швидше. А тепер давайте розглянемо навіщо нам потрібна Azure AI Studio:

- **Спрощення розробки:** Дана студія знижує поріг входу для використання штучного інтелекта, дозволяючи розробникам та компаніям створювати інноваційні рішення без глибоких знань у машинному навчанні.
- **Масштабованість:** Azure AI Studio забезпечує легке масштабування від невеликих прототипів до промислових рішень.
- **Швидке розгортання:** Інтеграція з іншими сервісами компанії Microsoft дозволяє швидко запускати моделі в реальному часі та середовищі.
- **Ефективність для бізнесу:** Azure AI Studio використовується для автоматизації рутинних процесів, покращення взаємодії з клієнтами, а також прийняття рішень на основі даних.

Це тільки основні причини навіщо нам потрібна Azure AI Studio, тепер можемо розглянути процес тренування моделі:

1. **Збір даних:** Першим етапом в нас є підготовка даних для тренування. Це можуть бути графічні, текстові або інші дані, які необхідні для конкретної моделі. Azure AI Studio дозволяє інтегрувати набори даних з різних джерел.
2. **Попередня обробка даних:** Зібрані дані нормалізуються, очищуються та готуються для моделі. У таку обробку входить видалення шумів, дубльованих записів або пропущених значень. У випадку з текстовими даними, то необхідно створювати особливий для Azure AI Studio формат файлу, про який ми поговоримо у наступному розділі.
3. **Вибір та налаштування моделі:** Розробник може вибрати готову модель із бібліотеки Azure або створити кастомізовану власну версію. Azure AI Studio підтримує використання різних типів

моделей для різних завдань, наприклад, для прогнозування або класифікації.

4. **Тренування моделі:** Після налаштування модель проходить процес тренування на обраному наборі даних. Під час цього етапу відбувається коригування ваг та параметрів моделі з метою максимізації точності. Процес тренування може займати від кількох хвилин до кількох годин або днів, залежно від складності завдання та розміру даних.
5. **Оцінка та оптимізація:** Після завершення тренування моделі, вона оцінюється за допомогою текстових наборів даних. Важливо перевірити, чи добре працює модель на нових даних та чи відповідає вона поставленим цілям. Якщо потрібно, розробники можуть оптимізувати параметри моделі.
6. **Розгортання:** Коли модель повністю готова, вона розгортається у реальному середовищі для використання кінцевими користувачами. Azure AI Studio дозволяє розгортати моделі у хмарі, забезпечуючи масштабованість та інтеграцію з іншими службами.
7. **Моніторинг та підтримка:** Після розгортання важливо стежити за продуктивністю моделі та її точністю, та при необхідності виконувати оновлення чи повторне тренування з новими даними.

### **3.2 Проектування інтерактивного помічника «Єва» та його тестування**

В минулому підрозділі ми розглянули з вами етапи розробки та тренування моделі, отже підемо за цими пунктами. В цьому підрозділі ми підемо по пунктах та розглянемо детально кожен етап. В нашому випадку збір даних відбувався за допомогою гугл опитування студентів та працівників Херсонського державного університету. В дане

опитування входили найчастіші питання, які виникали у студентів до викладачів, деканів та секретарів деканів. В процесі проходження опитувань, більшу активність, нажаль, проявили саме викладачі та декани різних факультетів університету. Також збір даних відбувався завдяки платформ з пошуку роботи, наприклад [робота.юа](#) чи [djinni](#). Основуючись на цю інформацію, ми зробили так звану RDF Model. RDF (Resource Description Framework)[17] [18] [36]– це технологія, яка використовується для опису даних у семантичній павутині, проте, це стандарт для опису інформації у веб-просторі та дозволяє представляти дані у вигляді так званих «триплетів»: предикат, об'єкт та суб'єкт. Подібна структура дозволяє створювати прості взаємозв'язки між різними елементами інформації. RDF фокусується на описі даних та використовує URI для ідентифікації ресурсів, а також описує їхні зв'язки та властивості. З основних характеристик можна включити триплетну модель, у якій зберігається інформація (предикат-об'єкт-суб'єкт), відкриті формати: RDF використовує різні формати для зберігання даних (наприклад, RDF/XML, JSON-LD, Turtle) та семантику, бо RDF забезпечує базовий рівень семантичного опису даних, тобто описує тільки факти (або ж, знання) без складних логічних залежностей.



Рисунок 3.1 RDF модель із зібраними даними

Окрім RDF [6]моделей(дивитись рисунок 3.1), також розглядалась мова онтології, тобто OWL, або ж Web Ontology Language[32]. OWL – це мова, яка побудована на основі RDF, але дозволяє створювати складніші ієрархії та логічні правила для опису понять. OWL розширює нам можливості RDF, дозволяючи моделювати більш складні семантичні відношення та зв'язки між даними. З основних характеристик ми можемо включити логічні залежності, тобто OWL додає нам можливість визначити класи, логічні висновки та їх обмеження, зв'язки між класами та підкласи, наприклад, ми можемо вказати, що певний клас є підкласом іншого, або ж, що два класи не є сумісними; типи OWL, тобто, існує кілька версій OWL, як OWL DL, OWL Lite, OWL Full, які відрізняються складністю обробки та рівнем виразності; виведення знань, тобто OWL дозволяє нам робити логічні висновки на основі визначених відносин, що є важливим для розширеної обробки знань у семантичній павутині. Потрібен нам OWL для того, щоб створювати складні онтології структурованих моделей, які описують концепції та відносини у певній галузі, а також для складнішої обробки даних, включаючи виведення нових знань на основі вже існуючих. Відмінністю між RDF та OWL є рівень складності, бо RDF – це базовий рівень опису даних, тоді як OWL дозволяє створювати складніші логічні ієрархії та структури, також відмінність є у можливостях, тобто, RDF фокусується на описі зв'язків та фактів між ресурсами, тоді як OWL додає логічні та семантичні залежності, які дозволяють моделювати онтології, ще відмінністю можна назвати виведення знань, бо RDF сам по собі не підтримує логічні висновки, тоді як OWL має вбудовані механізми для реалізації цього процесу. Ми ж будемо використовувати саме RDF модель, так як її краще використовувати для опису простих зв'язків між даними та їх інтеграції з різних джерел. OWL, зі свого боку, підходить для створення складних моделей знань з можливістю логічних висновків та більш глибокого аналізу даних.



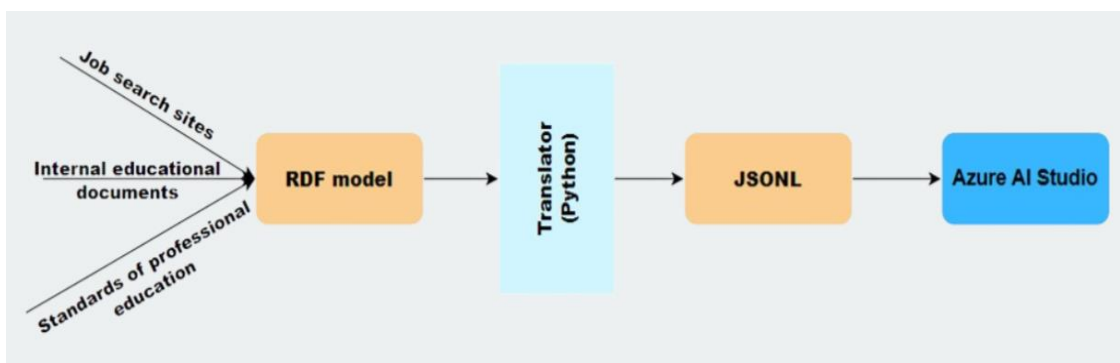


Рисунок 3.2 План-схема процесу створення тренувального файлу для Azure AI Studio

У план-схемі на рисунку 3.2, ви можете побачити поетапний процес створення якої інформації використовується у RDF моделі. У дану модель, як згадувалось раніше, входять дані, зібрані з платформ з пошуку роботи, такі як роботаюа, Work.ua, Djinni, програми навчання для абітурієнтів, які мають бажання вступити до Херсонського державного університету та стандарти під обрану вакансію на одній з платформ з пошуку роботи, а також, до цього входять часто задавані питання від студентів до працівників Херсонського державного університету.

І ось після етапу збору інформації, ми можемо перейти тепер до обробки зібраних нами даних. Наступним пунктом у нашій план-схемі йде перекладач файлів RDF, написаний мовою програмування Python. При написанні даного парсеру, було використано бібліотеку для мови програмування Python – RDFLib[40] для роботи з RDF файлами. Оскільки RDF використовується для представлення інформації про ресурси семантичній павутині, RDFLib надає потужний та зручний інструмент для, створення, маніпулювання, а також запитів RDF-triples, які вважаються основними одиницями даних в RDF. З основних можливостей можемо включити якраз роботу з RDF-triples, тобто, RDF-дані представлені у вигляді трійки, як об’єкт, предикат та суб’єкт. RDFLib дозволяє з легкістю маніпулювати з трійками та створювати їх;

Серіалізація та парсинг, бо RDFLib підтримує кілька форматів для читання та запису RDF-даних, зокрема Turtle, XML, JSON-LD та N-Triples. Ці маніпуляції дозволяють перетворювати дані в різні RDF-формати серіалізації; Управління графами, тобто RDFLib обробляє наші RDF-дані як граф, у якому вузли представляють об'єкти та суб'єкти, а їх ребра – предикати. Цей процес полегшує нам керування та запити до складних відносин; SPARQL-запити, мається на увазі, що RDFLib підтримує SPARQL – стандартну мову запитів для роботи з RDF-даними. Це дозволяє виконувати складні запити для отримання конкретної інформації; Інтеграція з інструментами Семантичного павутиння, тобто ми можемо RDFLib інтегрувати з іншими технологіями семантичного павутиння, наприклад SHACL (Shapes Constraint Language) та вже відомий нам OWL, що робить цю бібліотеку корисною для веб та застосунків зі штучним інтелектом.

```

1 from rdflib import Graph
2 from rdflib.namespace import RDF, RDFS
3
4 # Load RDF file
5 g = Graph()
6 g.parse("AIKSU.rdf", format="xml")
7
8 # Bind namespace prefixes
9 g.bind("rdf", RDF)
10 g.bind("rdfs", RDFS)
11
12 search_criteria = "Property"
13
14 # SPARQL query
15 query = f"""
16 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
17 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
18
19 SELECT ?subclass_label ?comment
20 WHERE {{
21   ?subclass rdfs:subClassOf <file:///E:/Student/Shadow_Killer046/Magister_Practise/AI_Student/Igor/AIKSU.rdf#{search_criteria}> ;
22   rdfs:label ?subclass_label ;
23   rdfs:comment ?comment .
24 }}
25 """
26
27 # Execute SPARQL query and get results
28 result = g.query(query)
29
30 # Print results
31 print("SPARQL query results:\n*****")
32 for row in result:
33     result_dict = {
34         "prompt": str(row.subclass_label),
35         "completion": str(row.comment)
36     }
37     print(result_dict)
38
39 # Save results to a text file

```

Рисунок 3.3 Програмний код Python парсеру, з використання RDFLib

Програмний Python код, який ви можете побачити у рисунку 11 це наш парсер, який використовую бібліотеку RDFLib[41], який обробляє нашу створену RDF модель з використанням SPARQL-запитів.

Розглянемо наш програмний код більш детально, починаємо ми звично, з імпортування необхідних нам бібліотек, а саме, вже відомої нам, RDFLib. Імпортуємо з даної бібліотеки три необхідні нам класи, як Graph, основний клас нашої бібліотеки, який дозволяє нам працювати з RDF моделлю (тим самим тривимірним набором: об'єкт-предикат-суб'єкт) та RDF з RDFS, які ми використали для зв'язування префіксів з відповідними просторами імен RDF та RDFS (так званих стандартів RDF моделі). Наступним нашим кроком буде завантаження нашого RDF файлу з моделлю. В програмному коді ми створили об'єкт графа (g) та пишемо команду для парсингу нашого файлу. У нашому випадку, ми використовуємо файл «AIKSU.rdf», який містить в собі RDF модель у форматі XML. Після завантаження та парсингу, ми робимо прив'язку префіксів простору імен, таких як RDF та RDFS, щоб їх можна було нам використовувати у SPARQL- запитах замість довгих URI. Наступним кроком у нас ідуть SPARQL-запит. Ми створюємо SPARQL-запит, який шукає усі класи, які є підкласами зазначеного ресурсу у нашому RDF файлі. Такий запит використовує префікси rdf та rdfs для пошуку у файлі. Також, у нашому запиті запитуються підкласи ресурсу Property, їхні мітки (тобто, label) та їхні коментарі (comments). У результаті будуть знайдені відповідні нам підкласи з їхніми описами. Виконання такого SPARQL-запиту над RDF моделлю ми зберігаємо у змінній result. Цикл виведення проходить по кожному результату, дістає мітку підкласу (subclass\_label) та коментар (comment), після чого виводить обрані дані на екран. Для виведених даних на екран, створюється файл із назвою, який відповідає пошуковому критерію та записуються в нього виведені результати у JSON форматі з ключами «prompt», тобто можливий запит користувача та «completion», тобто відповідь на запит користувача. Після успішного збереження результатів у файлі наш парсер виводить нам готовий JSON файл з обробленими даними з таким форматом:

```

{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}

```

```

1 {"prompt": "You are an AI student assistant of Kherson State University named as Eve(In ukrainian writes like \"Єва\") that helps students find information. You must use
2 {"prompt": "Hi", "completion": "Hi! I'm Eve. Your personal student assistant of Kherson State University. How can I help you?"}
3 {"prompt": "I want become a Software Engineer", "completion": "Sure! Software Engineer is Vacancy. A professional involved in the design, development, testing, and mainte
4 {"prompt": "what do I need to learn to become a Software Engineer?", "completion": "You will need to join the Kherson State University on Computer Science or Operation S
5 {"prompt": "which disciplines do this faculty prefer?", "completion": "Faculty of Computer Science, Physics and Mathematics providing lots of disciplines like Modeling L
6 {"prompt": "If I want to learn myself what will I need to learn to become a Software Engineer?", "completion": " Start with a programming language. Common choices include
7 {"prompt": "Do I need learn something new if I already can programming?", "completion": " Software engineering is a constantly evolving field. Stay updated with new techr
8 {"prompt": "which soft skills do I need have?", "completion": " You will must have soft skills like: Develop communication, teamwork, and problem-solving skills. Softwar
9 {"prompt": "what is a Modeling languages:UML?", "completion": "It is Competencies. A proficiency in Unified Modeling Language (UML), a standardized general-purpose modeli
10 {"prompt": "what is a Modeling languages:sysML?", "completion": "It is Competencies. A proficiency in Systems Modeling Language (SysML), a general-purpose modeling langua
11 {"prompt": "Thanks", "completion": "You're welcome! If you need something, I will be here!"}
12 |

```

### Рисунок 3.4 Приклад структури JSONL файлу

На рисунку 12 ви можете побачити англomовний приклад структури JSONL файлу, відформатований файл, що використовується у проєкті буде набагато більше, так як кількість запитів студентів в кількості набагато більше та відноситься до різних факультетів Херсонського державного університету, така сама ситуація стосується і платформ з пошуку роботи та навчальних програм для абітурієнтів та курсів, які необхідні для опанування певної навички.

Отже, ми маємо на руках зроблений файл для тренування моделі, тепер можемо перейти до платформи Azure AI Studio, де завантажимо наш файл для подальшого створення власної моделі, яка буде навчатись завдяки нашому створеному файлу. Якщо пригадати минулий підпункт, то ви можете звернути увагу, що дана платформа має дуже зрозумілий та інтуїтивний інтерфейс, що дає можливість використовувати платформу новачкам. Завантаження файлу відбувається у вкладці «Data files» та після завантаження файлу, ми можемо переходити до етапу створення власної моделі прямо у платформі Azure AI Studio.

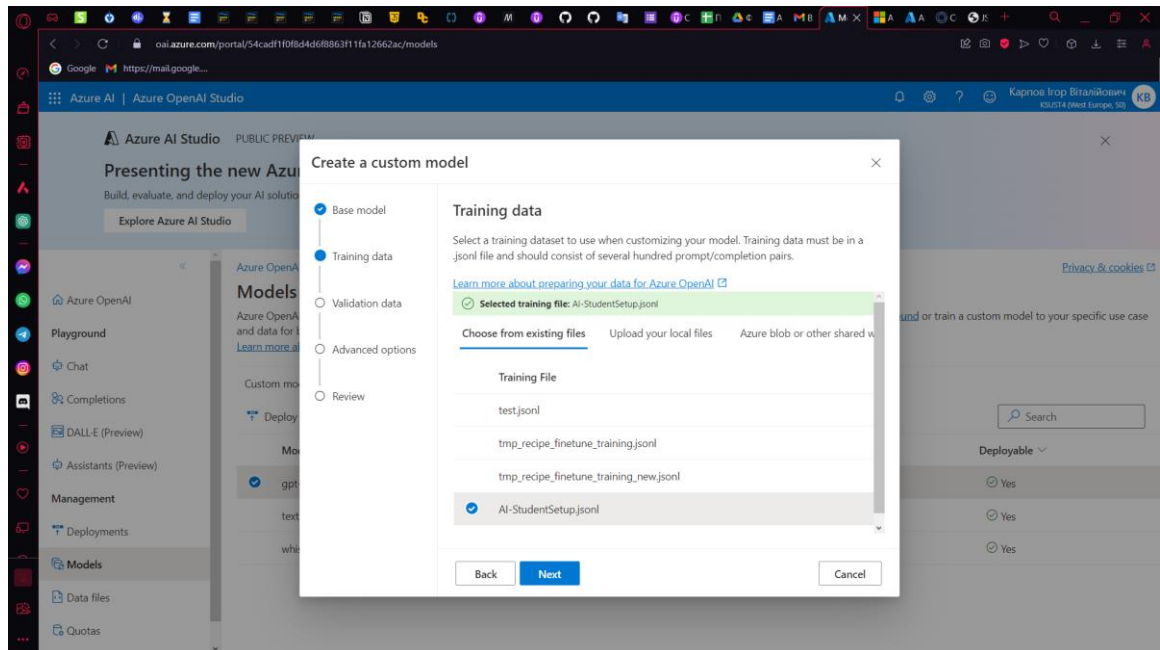


Рисунок 3.5 Вікно процесу створення власної моделі

На рисунку 13 ви можете звернути увагу на процес створення власної моделі на основі наших завантажених даних. У вікні створення власної моделі ми маємо можливість налаштувати певну кількість токенів, дати моделі певне ім'я та після налаштувань потрібно зачекати поки ця модель пройде процес тренування. Цей процес, в залежності від об'єму файлу займає певну кількість часу. Іноді такий процес може зайняти декілька днів, якщо ви працюєте з файлом, який має великий об'єм даних. Після довгоочікуваного завершення створення власної моделі, навчаної нашими тренувальними даними, ми маємо можливість її розгорнути. Для того, щоб розгорнути нашу новостворену модель, нам потрібно перейти до вкладки зі списком моделей у платформі Azure AI Studio та обрати зі списку створених нами моделей нашу новостворену, після цього, натискаємо кнопку розгортання та після натискання у нас відкриться вікно налаштувань перед розгортанням моделі. У вікні налаштувань ми можемо обирати модель, яку ми хочемо розгорнути, обрати її версію, так як різні версії моделей підтримують та не

підтримують деякі функції, які необхідні для тренування та тестування моделі. Також ми маємо можливість дати ім'я розгорнутої моделі. З додаткових налаштувань, ми можемо налаштувати фільтрування контенту, тип розгортання та налаштувати кількість tokenів. Кількість tokenів нам необхідна для подальшого використання нашої моделі. Деякий відсоток tokenів використовується для наших даних, запитів та відповідей самої моделі. Чим більше tokenів, тим більше та довше ми можемо взаємодіяти з нашою моделлю.

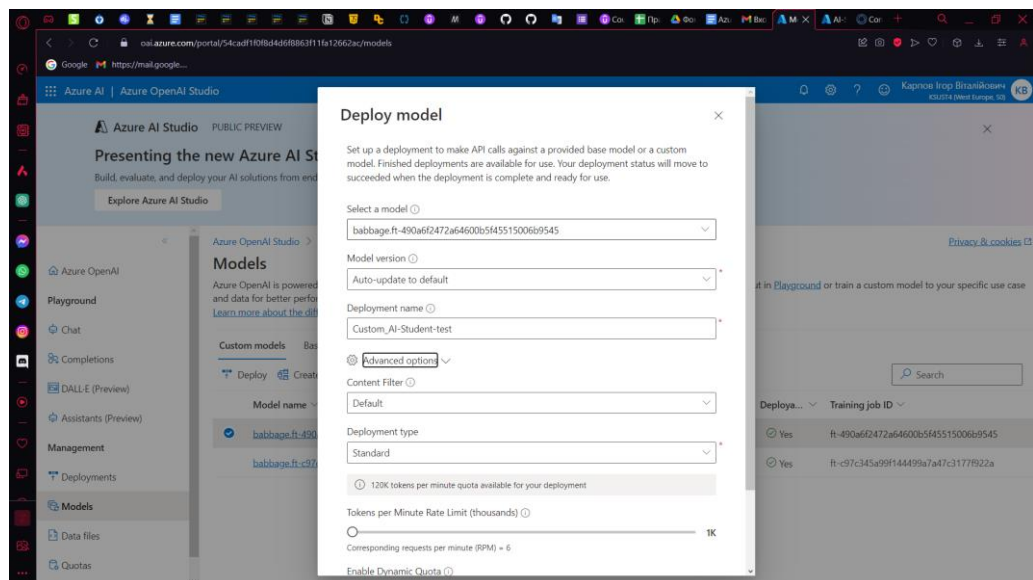


Рисунок 3.6 Вікно налаштування власної моделі перед розгортанням

Отже, після завантаження тренувального файлу, створення на основі цього файлу власної моделі та його розгортання, ми можемо перейти до процесу тестування нашої натренованої моделі. Тестування ми можемо проводити у вкладці «Completions». У цій вкладці ми маємо можливість задавати параметри цій моделі, наприклад, для того, щоб модель відповідала менш розгорнуто або навпаки, більш розгорнуто або, щоб модель надавала чітко за нашими даними або більш креативно, використовуючи наші дані. У полі для вводу ми можемо взаємодіяти з

самою моделлю, наприклад, можемо ввести якусь певну строку з нашого тренувального файлу та подивитись як наша модель сприймає наш запит та як надає на нього відповідь. Azure AI Studio надає нам можливість налаштовувати параметри моделі та тестувати її поведінку та підлаштовувати параметри під необхідну нам поведінку, наприклад, ми хочемо, щоб наша модель надавала відповідь чітко по нашим запитам з тренувального файлу, як по сценарію або, щоб модель надавала відповідь більш креативно, типу у стилі ChatGPT з використанням деяких пунктів з нашого тренувального файлу. Після тестування власної моделі ми маємо можливість експортувати JSON файл з налаштуваннями, які ми виставили у платформі та використати його у власному API застосунку. Це дуже спрощує нам процес створення власного помічника для різних галузей, наприклад, для консультації клієнтів по товарах на сайті інтернет-магазину або для будь-якої іншої галузі. У нашому випадку ми тренували інтерактивного помічника зі штучним інтелектом. Якщо ми тренуємо моделі ChatGPT, платформа Azure AI Studio дає можливість розробникам створювати веб-застосунок прямо у платформі та навіть можливість зберігати історію листування з моделлю в базі даних. Ви можете подивитись створений веб-застосунок у рисунку 15.

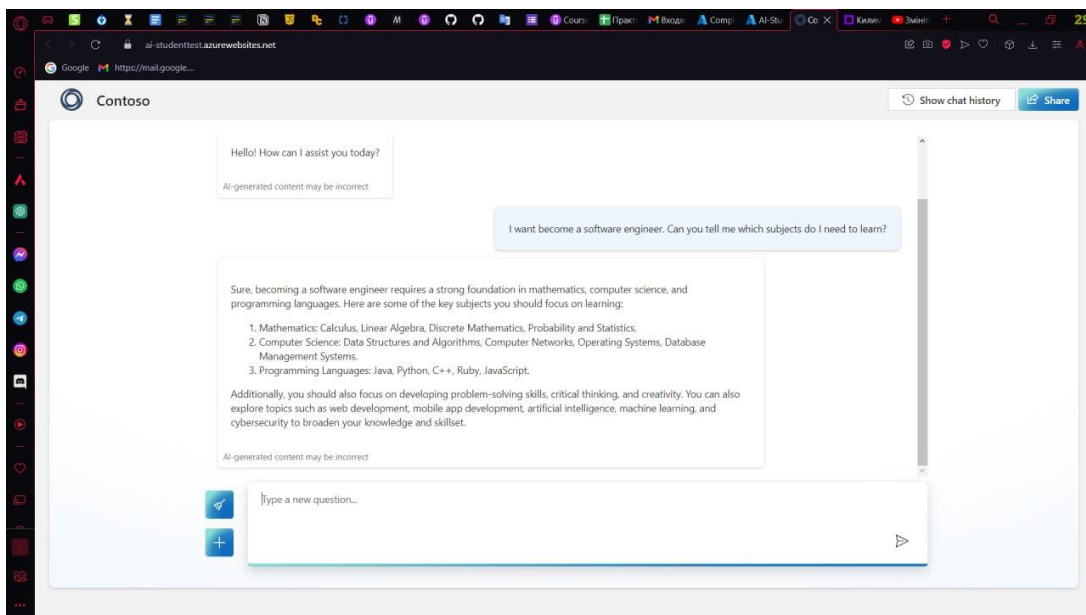


Рисунок 3.7 Створений веб-застосунок з моделлю завдяки Azure AI Studio



## ВИСНОВКИ

Отже, під час написання даної кваліфікаційної роботи, нами було виконано розробку інтерактивного помічника зі застосуванням штучного інтелекта, який має мету спростити взаємодію між студентами та навчальними матеріалами. Також, в ході кваліфікаційної роботи було нами вирішено такі задачі, як аналіз існуючих технологій штучного інтелекта, а саме, було досліджено сучасні технології, як машинне навчання, обробка природної мови, а також інтеграцію з базами даних та API; розробка архітектури системи, тобто, нами було обрано багаторівневу архітектуру, яка включає у себе компоненти для обробки мовних запитів, доступ до бази даних про студентів та управління інформацією, такою як розклади занять, завдання, оцінки та консультація студентів з питань університету; використання сучасних методів навчання моделей штучного інтелекта, тут особлива увага була приділена процесу тренування власної моделі за допомогою власних тренувальних даних, зібраних з різних джерел та оброблених завдяки файловому парсеру, написаному мовою програмування Python; Реалізація інтерфейсу користувача, було розроблено зручний на зрозумілий інтерфейс, який дозволяє студентам взаємодіяти з системою, отримуючи необхідну інформацію від помічника в інтерактивному режимі.

Відповідно, таким чином, розроблений нами інтерактивний помічник значно спрощує взаємодію студентів з загальноуніверситетськими питаннями, такими як дізнаватися про свої оцінки або як і де отримати свій диплом. Таке рішення має потенціал для подальшого розвитку та інтеграції з іншими системами управління навчальним процесом та зменшує потік однакових питань до працівників факультету від студентів.

## References

1. 12 game-changing moments in the history of artificial intelligence (AI): <https://www.livescience.com/technology/artificial-intelligence/12-game-changing-moments-in-the-history-of-ai> ;
2. artificial intelligence: <https://www.britannica.com/technology/artificial-intelligence>;
3. Azure AI Studio documentation: <https://learn.microsoft.com/uk-ua/azure/ai-studio/> ;
4. Azure AI Studio: Building Custom Chatbots with ChatGPT: <https://azure.microsoft.com/en-us/products/machine-learning/studio/> ;
5. Azure AI Studio: <https://azure.microsoft.com/en-us/products/ai-studio>;
6. Chatbot «Eva» PERSONALIZATION OF LEARNING IN AN EDUCATIONAL ONLINE ENVIRONMENT: <https://about.ai.kspu.edu/#/1> ;
7. Component Diagram Tutorial: <https://www.lucidchart.com/pages/uml-component-diagram> ;
8. Deployment Diagram Tutorial: <https://www.lucidchart.com/pages/uml-deployment-diagram> ;
9. Higher-ed and Healthcare are transforming customer service with Ivy.ai: <https://www.ceocfointerviews.com/ivyaiinc20.html> ;
10. History of AI: Timeline and the Future: <https://online.maryville.edu/blog/history-of-ai/> ;
11. Ivy.ai: Advancing AI in Higher Education: <https://www.pymnts.com/artificial-intelligence-2/2024/ivy-ai-takes-chatbots-beyond-higher-educations-hallowed-halls/> ;
12. Ivy.ai: <https://serp.ai/tools/ivy-ai/> ;
13. Packback AI: Engaging Students through Critical Thinking: <https://www.packback.co/about/> ;

14. Packback Questions: <https://www.packback.co/product/questions/> ;
15. Packback: Improving Communication and Critical Thinking: <https://www.packback.co/stories/improving-communication-and-critical-thinking-with-packback/> ;
16. Purdue University's AI-Powered Chatbot (Ivy.ai): <https://it.purdue.edu/newsroom/articles/230828-ivy-chatbot0live.php> ;
17. Resource Description Framework (RDF): <https://www.w3.org/RDF/> ;
18. Resource Description Framework: [https://en.wikipedia.org/wiki/Resource\\_Description\\_Framework](https://en.wikipedia.org/wiki/Resource_Description_Framework) ;
19. Socratic by Google: <https://blog.google/outreach-initiatives/education/socratic-by-google/> ;
20. Socratic by Google: Pricing, Reviews & Top 4 Alternatives [2024]: <https://tome.app/productivity-tips/socratic-by-google-pricing-reviews-and-alternatives> ;
21. Socratic Teaching Techniques for Effective Learning: <https://www.structural-learning.com/post/socratic-teaching-techniques-for-effective-learning> ;
22. The Future of Education: AI Learning Tools for Students and Teachers: <https://www.packback.co/stories/the-future-of-education-ai-learning-tools-for-students-and-teachers/> ;
23. The History of AI: A Timeline of Artificial Intelligence: <https://www.coursera.org/articles/history-of-ai> ;
24. The History of Artificial Intelligence from the 1950s to Today: <https://www.freecodecamp.org/news/the-history-of-ai/> ;
25. The History of Artificial Intelligence: <https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf> ;
26. UML Component Diagrams – Tutorial: [https://www.tutorialspoint.com/uml/uml\\_component\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_component_diagram.htm) ;

**27. UML Deployment Diagrams – Tutorial:**

[https://www.tutorialspoint.com/uml/uml\\_deployment\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_deployment_diagram.htm) ;

**28. UML Sequence Diagram Tutorial:**

<https://www.lucidchart.com/pages/uml-sequence-diagram> ;

**29. UML Sequence Diagrams – Tutorial:**

<https://www.tutorialspoint.com/uml/index.htm> ;

**30. UML Use Case Diagram Tutorial:**

<https://www.lucidchart.com/pages/uml-use-case-diagram> ;

**31. UML Use Case Diagrams – Tutorial:**

[https://www.tutorialspoint.com/uml/uml\\_use\\_case\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm) ;

**32. Web Ontology Language (OWL):**<https://www.w3.org/OWL/> ;

**33. What is AI? Artificial Intelligence explained:**

<https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence> ;

**34. What is artificial intelligence (AI)?:**

<https://www.ibm.com/topics/artificial-intelligence> ;

**35. What is Azure AI Studio?:** [https://learn.microsoft.com/en-](https://learn.microsoft.com/en-us/azure/ai-studio/what-is-ai-studio)

[us/azure/ai-studio/what-is-ai-studio](https://learn.microsoft.com/en-us/azure/ai-studio/what-is-ai-studio) ;

**36. What Is RDF?:**

<https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf/> ;

**37. What is Socratic and How to Use It?:**

<https://neuralnetworkpress.com/what-is-socratic-and-how-to-use-it> ;

**38. What is the history of artificial intelligence (AI)?:**

<https://www.tableau.com/data-insights/ai/history> ;

**39. Історія штучного інтелекту на Britannica:**

<https://www.britannica.com/technology/artificial-intelligence> ;

**40. Офіційна документація RDFLib:** <https://rdflib.readthedocs.io/> ;

**41. Репозиторій RDFLib на GitHub:** <https://github.com/RDFLib/rdflib> ;

**42. Штучний інтелект:**

<https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87>

[%D0%BD%D0%B8%D0%B9 %D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82 ;](#)