МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проект

здобувача освіти денної форми навчання

РП.07.19.000.ДП

Стремецького-(Скорубського) Максима Віталійовича

м. Одеса

2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: ***121 «Інженерія програмного забеспечення»***

Освітньо-професійна програма: ***«Розробка програмного забезпечення»***

Група: ***4РП-07***

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка сервера для цифрових помічників на основі штучного інтелекту

Проєктний матеріал складається з пояснювальної записки на **63** сторінках та графічного (перзентаційного) матеріалу на **14** аркушах (слайдах).

Дипломник _____Стеремецький-(Скорубський М.В.)

Керівник _____ (Кунуп Т. В)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист « 17 » 06 2024 р. Протокол ДКК № 1

Оцінка ДКК 41 добре/180 б.

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Ком'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 – «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ

Заст. дир. з НВР *Беркань І. В.*

« 15 » 07 *2024* року

З А В Д А Н Н Я

на дипломний проєкт (роботу)

Стремецькому - (Скорубському) Максиму Віталійовичу

1. Тема проєкту (роботи) *Розробка сервера для цифрових помічників на основі штучного інтелекту*

Затверджена наказом по коледжу від « *02* » *11* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту (роботи) *10.06.2024р*

3. Вихідні дані до проєкту (роботи)

Розробити сервер, для цифрових помічників на основі штучного інтелекту, мова програмування JavaScript. з використанням платформи з відкритим кодом Nodejs, фреймворк для створення серверних додатків NestJS, фреймворк LangChain, OpenAi, технологія Swagger.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз існуючих програмних рішень; Формування вимог до сервера; Порядок розробки серверу; Проектування структури сервера; Розробка бази даних; Тестування створеного програмного продукту; Економічний розрахунок; Охорона праці та техніка безпеки.

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

Презентація Power Point – 12 слайдів

Структура проекта; Контролери маршрутів; Схема роботи методу create; Структура сервера; Скриншот схеми бази даних. MongoDB; Результати тестування.

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Кунуп Т. В.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання 15.01.24

Керівник *Кунуп Т. В.* _____
(підпис)

Завдання прийняв до виконання *Стремецький -Скорубський М.В..* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*29.04.24*	Виконано
2	*Огляд основних компонентів серверної архітектури*	*01.05.24*	виконано
3	*Обгрунтування вибору розробки серверних рішень*	*03.05.24*	виконано
4	*Проєктування сервісу*	*05.05.24*	виконано
5	*Налаштування структури проекту*	*07.05.24*	виконано
6	*Інтеграція бази даних*	*09.05.24*	виконано
7	*Тестування створеного програмного продукту*	*15.05.24*	виконано
8	*Економічний розрахунок*	*17.05.24*	виконано
9	*Опис охорони праці та техніки безпеки*	*19.05.24*	виконано
10	*Аналіз результатів проєктування.*	*21.05.24*	виконано
11	*Оформлення пояснювальної записки*	*23.05.24*	виконано
12	*Оформлення графічної (презентаційної)*	*30.05.24*	виконано
13	*частини*	*05.06.24*	виконано
14	*Підготовка доповіді для захисту*	*07.06.24*	виконано
15	*Малий захист дипломного проєкту*	*10.06.24*	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 19 000. 00 ДП	Дипломний проєкт		
А4			РП 07. 19 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 19 000. 00 ДП ПЗ		
Розробив		Стремецький		10.06	Розробка сервера для цифрових помічників на основі штучного інтелекту	Літ.	Аркуш 4 / Аркушів 60
		Скорубський М.		10.06		Н Д П	
Перевірив		Кунуп Т.В.		10.06		ВСП «ОТФК ОНТУ» гр.4РП-07	
Н. Контр.		Петрашова В.І.		10.06			
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП 7

1. ОСНОВНИЙ РОЗДІЛ 8

1.1 Огляд штучного інтелекту та його застосування у цифрових помічниках 8

1.2 Архітектура серверів для систем на основі штучного інтелекту 10

1.2.1 Огляд основних компонентів серверної архітектури 10

1.2.2 Використання Node.js та NestJS для розробки серверних рішень 11

1.2.3 Роль MongoDB і Pinecone у зберіганні та обробці даних 11

1.2.4 Swagger для документації API 12

1.2.5 OpenAI API 12

1.3 Схема серверної архітектури для систем на базі ШІ 12

1.4 Впровадження штучного інтелекту через API 14

1.4.1 Використання OpenAI для інтеграції LLM моделі 14

1.5 Практична частина розробки сервера 16

1.5.1 Планування проекту 16

1.5.2 Вибір технологічного стеку та обґрунтування вибору 17

1.5.3 Розробка сервера та інтеграція 19

1.5.4 Реалізація сервера 20

1.5.5 Ініціалізація NestJs 20

1.5.6 Налаштування структури проекту 21

1.5.7 Створення ендпоїнтів 23

1.5.8 Інтеграція бази даних MongoDB та Pinecone 24

1.6 Інтеграція векторної бази даних Pinecone 25

1.7 Методи сервіса EmbeddingService 32

1.8 Підключення та використання OpenAI API 37

1.8.1 Налаштування доступу до OpenAI API 37

1.8.2 Опис API за допомогою Swagger 39

1.9. Робота програми 42

2. ЕКОНОМІЧНА РОЗДІЛ 49

2.1 Резюме 50
2.2 Вивчення трудомісткості розробки програмного забезпечення 51
2.3 Розрахунок ціни програмного продукту 52
3. РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 55
3.1 Аналіз та безпека умов праці працівника на робочому місці 55
3.2 Розробка заходів з охорони праці 56
3.3 Організація робочого місця користувача ПК 57
3.4 Пожежна безпека 57
ВИСНОВКИ 59
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 60
Додаток А Слайди мультимедійної презентації 61

ВСТУП

Штучний інтелект (ШІ) визначається як галузь науки та технологій, зосереджена на створенні машин, які демонструють інтелектуальні здібності, та комп'ютерних програм, які імітують людський інтелект. Ці системи намагаються не лише розуміти, але й відтворювати людський розум, проте не обмежуються лише методами, що імітують природній біологічний процес.

Станом на 2021 рік у сфері штучного інтелекту існує кілька основних напрямків:

1. Артифіційний суперінтелект (ASI) — це теоретична форма ШІ, яка може не тільки імітувати, але й перевершувати людські здібності, включаючи можливість впливати на людські думки та емоції.

2. Штучний загальний інтелект (AGI) є ще одним гіпотетичним видом ШІ, рівень якого є нижчим від ASI. Прихильники цієї ідеї вважають, що такий інтелект зможе виконувати різноманітні дії на рівні людини.

3.Слабкий, або обмежений інтелект (ANI) характеризується здатністю виконувати лише певні завдання, для яких він був спеціально розроблений, не маючи здатності до самостійного розвитку чи незалежної поведінки.

Інтелект, як психологічний термін, означає здатність особи пристосовуватися до нових умов, навчатися на основі досвіду, розуміти та застосовувати абстрактні поняття, а також використовувати набуті знання для керування своїм оточенням. Ця загальна когнітивна здатність включає сприйняття, пам'ять, мислення та уяву.

Реалізація даного проекту є створення сервера для цифрових помічників на основі штучного інтелекту є перспективним напрямком у розвитку інформаційних технологій.

Створена платформа може бути використана у різних галузях, від бізнесу до освіти, і значно підвищує якість і доступність цифрових сервісів.

1. ОСНОВНА ЧАСТИНА

1.1 Огляд штучного інтелекту та його застосування у цифрових помічниках

Визначення та типи цифрових помічників:

Що таке цифрові помічники: Цифрові помічники - це програмні агенти, які можуть виконувати завдання або надавати послуги для індивідуума на основі команд або питань. Ці завдання часто виконуються за допомогою взаємодії, подібної до природної мови.

Типи цифрових помічників.

1. Голосові помічники: Працюють переважно за допомогою голосових команд. Приклади включають Alexa від Amazon, Siri від Apple і Google Assistant.

2. Текстові помічники: Взаємодіють переважно через текст (введений або вибраний). Приклади включають чат-ботів на вебсайтах або віртуальних агентів підтримки клієнтів.

3. Гібридні помічники: Поєднують як голосові, так і текстові функціональності, пропонуючи універсальні режими взаємодії.

Приклади включають Cortana від Microsoft і Bixby від Samsung.

4. Можливості та використання:

Голосові помічники зазвичай використовуються для взаємодії без використання рук, тоді як текстові помічники відмінно підходять у середовищах, де необхідно зберігати тишу або це переважно. Гібридні помічники забезпечують гнучкість, пристосовуючись до поточних потреб і контексту користувача.

Переваги інтеграції штучного інтелекту в цифрових помічників: Покращена взаємодія з користувачами: Завдяки обробці природної мови (NLP), цифрові помічники можуть розуміти та генерувати відповіді, подібні до людських, що робить взаємодію більш інтуїтивною та природною.

Покращена персоналізація: Алгоритми машинного навчання аналізують минулі взаємодії та вподобання користувача, щоб адаптувати відповіді та пропозиції, покращуючи загальний досвід користувача.

Збільшена ефективність і точність: Штучний інтелект дозволяє цифровим помічникам швидко та точно виконувати широкий спектр завдань, від встановлення нагадувань до отримання інформації та керування пристроями розумного дому.

Постійне навчання та адаптація: Цифрові помічники на базі ШІ навчаються з кожної взаємодії, адаптуючись до вподобань і поведінкових моделей користувача з часом, що постійно покращує їх функціональність і актуальність.

Розширення на поточні тенденції та майбутні напрямки:

Інтеграція з Інтернетом речей (IoT): З розширенням Інтернету речей (IoT) цифрові помічники все більше інтегруються з розумними домашніми пристроями, носимими технологіями та навіть автомобілями. Ця інтеграція дозволяє створити безперервний користувацький досвід, де цифрові помічники можуть керувати різноманітними пристроями та надавати всебічну підтримку в повсякденній діяльності.

Розширене контекстуальне розуміння: Майбутні досягнення в галузі ШІ дозволять цифровим помічникам глибше розуміти контекст, розпізнаючи нюанси в командах користувача та надаючи більш точні та релевантні відповіді. Це включає краще розуміння контексту з попередніх взаємодій, послуги на основі місця розташування та навіть емоційний тон.

Багатомодальні взаємодії: Майбутнє цифрових помічників, ймовірно, включатиме багатомодальні взаємодії, де користувачі можуть легко переключатися між голосовими, текстовими та візуальними введеннями.

Наприклад, користувач може розпочати команду за допомогою голосу на розумній колонці, продовжити її на смартфоні за допомогою тексту та завершити її на розумному годиннику за допомогою сенсорного інтерфейсу.

Підвищена безпека та конфіденційність: Оскільки цифрові помічники виконують дедалі більше конфіденційних завдань та обробляють особисту

інформацію, забезпечення надійних заходів безпеки та конфіденційності стане критично важливим. Майбутні розробки зосереджуватимуться на передових методах шифрування, децентралізованій обробці даних та більшій контролі користувача над особистими даними.

Галузеві специфічні застосування: Цифрові помічники також стають більш спеціалізованими, з'являються рішення, адаптовані для конкретних галузей, таких як охорона здоров'я, фінанси та освіта. Ці галузеві помічники можуть надавати експертні поради, управляти складними завданнями та підвищувати ефективність у професійних умовах.

Етичні та соціальні аспекти: Широке впровадження цифрових помічників також піднімає важливі етичні та соціальні питання. Це включає забезпечення рівного доступу до технологій, запобігання упередженості в алгоритмах ШІ та вирішення потенційного впливу на зайнятість і людські взаємодії.

Підтвердження підсумку: Я надав розширений огляд ШІ, описав типи цифрових помічників з прикладами та обговорив конкретні переваги інтеграції ШІ в цифрових помічників. Це включає покращення взаємодії на основі природної мови, персоналізацію користувацького досвіду, підвищення операційної ефективності, забезпечення постійного навчання та адаптації, а також дослідження майбутніх тенденцій, таких як інтеграція IoT, розширене контекстуальне розуміння, багатомодальні взаємодії, підвищена безпека, галузеві специфічні застосування та етичні міркування.

1.2 Архітектура серверів для систем на основі штучного інтелекту

1.2.1 Огляд основних компонентів серверної архітектури

Серверна архітектура для систем на базі штучного інтелекту вимагає високої продуктивності, масштабованості та гнучкості. Основні компоненти такої архітектури включають серверне середовище, бази даних для зберігання та

обробки даних, інтерфейси для взаємодії з іншими системами та компонентами, а також інструменти для моніторингу та управління ресурсами.

1.2.2 Використання Node.js та NestJS для розробки серверних рішень

Node.js – це потужна платформа для розробки серверного програмного забезпечення на JavaScript, яка дозволяє виконувати JavaScript на сервері. Вона підтримує асинхронне програмування, що дозволяє обробляти велику кількість з'єднань без блокування основного потоку виконання. NestJS, модерний фреймворк для Node.js, надає розробникам інструменти для створення добре структурованих, масштабованих додатків. Використання NestJS допомагає в реалізації таких патернів, як Dependency Injection та Decorators, що сприяє кращій модулярності та зменшенню залежностей між компонентами.

1.2.3 Роль MongoDB і Pinecone у зберіганні та обробці даних

MongoDB – це NoSQL база даних, яка використовує гнучку модель документів для зберігання даних, що є ідеальним для динамічних запитів та агрегації великих обсягів даних. MongoDB особливо ефективна в обробці великих наборів даних, що є типовим для AI-додатків, де потрібно швидко обробляти неструктуровані дані. Pinecone, з іншого боку, є векторною базою даних, спеціалізованою на швидкому пошуку схожості, що є ключовим для функцій, які використовують AI для аналізу великих даних. Це дозволяє системам швидко знаходити найбільш релевантні дані серед великих обсягів інформації, що підвищує продуктивність та точність AI-додатків.

Pinecone – це векторна база даних, оптимізована для швидкого і точного пошуку великих обсягів векторів, які зазвичай використовуються в AI для представлення вбудованих даних (embeddings).

Інтеграція Pinecone дозволяє ефективно виконувати складні запити подібності та ранжування, що є ключовим для функціонування рекомендаційних систем та інших AI-застосунків.

1.2.4 Swagger для документації API

Swagger використовується для проектування, будівництва та документування REST API. Це забезпечує чітке визначення API для розробників, які можуть використовувати це для розуміння та взаємодії з API без потреби заглиблюватися в код. Swagger спрощує процес тестування та інтеграції API, що є критичним для забезпечення надійності серверних додатків.

1.2.5 OpenAI API

OpenAI API дозволяє інтегрувати передові моделі машинного навчання, особливо моделі мовлення, які можуть здійснювати завдання перекладу, відповідей на запитання, аналізу текстів та багато іншого. Використання цього API відкриває доступ до потужних можливостей обробки мови, що є фундаментальним для створення інтерактивних та інтелектуальних цифрових помічників.

Ці компоненти формують основу для розробки масштабованої, високопродуктивної серверної архітектури, спроможної підтримувати складні AI-застосунки в реальному часі.

1.3 Схема серверної архітектури для систем на базі ШІ

Щоб створити схему для архітектури сервера, описаної вище, давайте детально опишемо компоненти та їх взаємодію в системі. Ця схема допоможе візуалізувати структуру та зв'язки між Node.js, NestJS, MongoDB, Pinecone та їх роль у серверних системах, заснованих на AI. Клієнт-серверна архітектура набула своєї популярності завдяки динамічному розвитку мережі Інтернет та зосередження значної частини інформації в базах даних на серверах.

Клієнтський шар:

Інтерфейс користувача: Фронтенд, через який користувачі взаємодіють, можливо, через веб-або мобільні додатки.

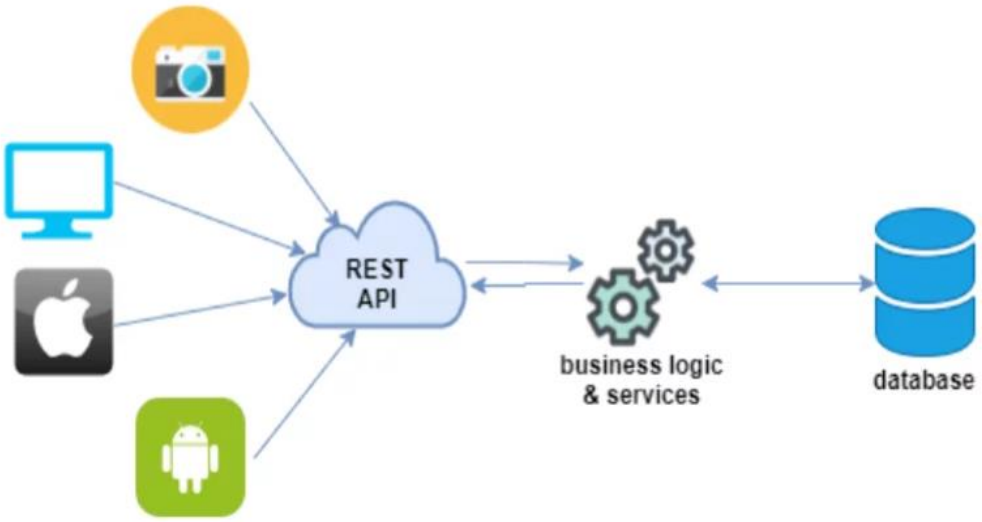


Рисунок 1.1. Схема серверної архітектури для систем на базі ШІ

Додатковий шар (Node.js/NestJS):

- Контролери**:** Обробляють вхідні HTTP запити та делегують завдання сервісам.
- Сервіси**:** Містять бізнес-логіку додатка, взаємодіючи з шаром бази даних та зовнішніми API.
- Модулі**:** Організують код у логічні групи, кожен модуль представляє основну функціональність додатка.
- Шар даних MongoDB**:**

Сховище документів**:** Зберігає структуровані та неструктуровані дані у документах схожих на JSON. Ідеально підходить для динамічних запитів та гнучкої розробки.

Колекції**:** Групи документів, кожна колекція використовується для різних наборів даних (наприклад, профілі користувачів, дані сесій).

- Векторна база даних**:** Зберігає та індексує вектори даних для швидкого пошуку схожості, що є важливим для функцій, таких як системи рекомендацій або виявлення контенту.

Інтеграційний шар:

Шлюз API (Swagger)**:** Керує, документує та маршрутизує запити API до відповідних сервісів, забезпечуючи єдину точку входу для всіх клієнтських запитів.

API OpenAI**:** Підключається до моделей OpenAI для виконання завдань обробки природної мови, підвищуючи можливості ШІ цифрових помічників.

1.4 Впровадження штучного інтелекту через API

API (інтерфейси програмування додатків) є ключовими елементами сучасного програмного забезпечення, що дозволяють різним компонентам системи ефективно взаємодіяти. API надають стандартизований спосіб зв'язку між програмними компонентами, що значно спрощує інтеграцію та взаємодію в складних системах. Ось як API сприяють взаємодії компонентів системи:

1. Абстрагування складності

API дозволяють абстрагувати складні функціональні можливості програмного забезпечення, надаючи прості інтерфейси для розробників. Це означає, що розробники можуть використовувати складні функції без необхідності розуміти внутрішню реалізацію.

2. Підвищення модульності

API допомагають розробляти модульні системи, де кожен компонент може бути розроблений, протестований і розгорнутий незалежно від інших. Це зменшує складність розробки та дозволяє легше масштабувати систему.

3. Зміцнення інтеграції

API полегшують інтеграцію різних систем і сервісів. Наприклад, веб-сервіси можуть взаємодіяти з базами даних, зовнішніми сервісами та іншими програмними компонентами через стандартні API виклики.

4. Підвищення гнучкості

Завдяки API, зміни в одній частині системи не потребують змін в інших частинах. Це забезпечує гнучкість при оновленнях і розширеннях системи, оскільки розробники можуть змінювати або додавати нові функціональні можливості, не впливаючи на існуючі компоненти.

5. Забезпечення безпеки

API можуть включати механізми аутентифікації та авторизації, що забезпечують безпеку при доступі до ресурсів системи. Це важливо для захисту чутливих даних та забезпечення контрольованого доступу.

1.4.1 Використання OpenAI для інтеграції LLM моделі

OpenAI надає потужні інструменти для інтеграції моделей великих мовних моделей (LLM) через API. Це дозволяє розробникам легко використовувати можливості штучного інтелекту в своїх додатках. Ось основні аспекти використання OpenAI для інтеграції LLM моделі:

1. Простота інтеграції

API OpenAI надає простий і зручний спосіб інтеграції LLM в додатки. Розробники можуть відправляти текстові запити до моделі і отримувати відповідні результати, що спрощує процес розробки.

2. Висока якість відповіді

Моделі OpenAI, такі як GPT-3, відомі своєю здатністю генерувати високоякісні та контекстно релевантні відповіді. Це дозволяє створювати

додатки, які можуть забезпечувати інтелектуальну підтримку, створювати тексти, відповідати на питання і багато іншого.

3. Гнучкість налаштувань

API OpenAI дозволяє налаштовувати параметри запитів, що дозволяє отримувати різні типи відповідей залежно від потреб додатку. Розробники можуть контролювати довжину відповіді, тон та стиль тексту, що генерується.

4. Масштабованість

OpenAI API побудований з урахуванням масштабованості, що дозволяє обробляти великий обсяг запитів і підтримувати високий рівень продуктивності. Це важливо для додатків, які потребують обробки великої кількості даних або обслуговування багатьох користувачів одночасно.

5. Безперервне навчання

Моделі OpenAI постійно оновлюються та вдосконалюються, що забезпечує актуальність та точність відповідей. Це дозволяє розробникам забезпечувати користувачів найсучаснішими можливостями штучного інтелекту.

6. Приклади використання

OpenAI API може використовуватися в різних галузях, включаючи:

- Підтримка клієнтів: автоматизовані відповіді на запити клієнтів.
- Контент-генерація: створення текстового контенту для вебсайтів та соціальних медіа.
- Аналіз даних: витяг і аналіз текстової інформації з великих обсягів даних.
- Освітні платформи: допомога студентам у навчанні та наданні освітніх матеріалів.

Розглянувши технології, зробимо такий висновок, що впровадження штучного інтелекту через API значно спрощує процес інтеграції AI технологій у додатки. Використання OpenAI для інтеграції моделей великих мовних моделей надає

розробникам потужні інструменти для створення інтелектуальних додатків з високою якістю взаємодії та широкими можливостями налаштувань.

Завдяки простоті використання, гнучкості та масштабованості, API OpenAI відкриває нові горизонти для впровадження штучного інтелекту в різних галузях, забезпечуючи інноваційні рішення для сучасних технологічних викликів.

1.5 Практична частина розробки сервера

1.5.1 Планування проекту

Вимоги до сервера:

Початковий етап розробки сервера включає визначення вимог до сервера на основі передбачуваної функціональності та потреб користувачів. Це включає:

1. Апаратні характеристики: Необхідна потужність процесора, обсяг пам'яті та обсяг зберігання для обробки очікуваного навантаження.
2. Програмні вимоги: Операційна система, системи баз даних та інші програмні залежності, необхідні для роботи сервера.
3. Заходи безпеки: Впровадження відповідних протоколів безпеки для захисту даних та забезпечення конфіденційності користувачів.

1.5.2 Вибір технологічного стеку та обґрунтування вибору

Вибір правильного технологічного стеку є критичним для успішної розробки та розгортання сервера. Процес вибору включає оцінку різних технологій на основі таких факторів, як продуктивність, масштабованість та легкість інтеграції. Для цього проекту обраний стек включає:

Node.js - це потужна платформа для виконання JavaScript на сервері, яка використовується для створення швидких та масштабованих мережевих додатків.

NestJS є сучасним фреймворком для побудови ефективних та масштабованих серверних додатків на базі Node.js. NestJS використовує прогресивний JavaScript, є побудованим з використанням TypeScript (але дозволяє використовувати чистий JavaScript) і повністю підтримує ООП (об'єктно-орієнтоване програмування), ФП (функціональне програмування) і ФРП (функціонально-реактивне програмування).

Основні особливості NestJS:

Модульність: Фреймворк організований у вигляді модулів, які можуть бути легко повторно використані у різних частинах додатку.

Ін'єкція залежностей: NestJS має вбудовану систему ін'єкції залежностей, що робить додаток легшим для тестування та розширення.

Розширюваність: Завдяки інтеграції з іншими бібліотеками та інструментами, такими як TypeORM, Mongoose та інші, NestJS робить легшою інтеграцію з різними базами даних.

Керування станом: NestJS підтримує RxJS, бібліотеку для асинхронного програмування з використанням засобів реактивного програмування, що допомагає керувати станом у великих додатках.

Тестування: NestJS надає потужні інструменти для юніт-тестування і інтеграційного тестування, роблячи розробку більш надійною.

NestJS є сучасним фреймворком для побудови ефективних та масштабованих серверних додатків на базі Node.js MongoDB розроблена для забезпечення високої продуктивності, високої доступності та легкої масштабованості.

Pinecone: це база даних, розроблена спеціально для ефективної роботи з векторними даними. Це рішення дозволяє зберігати, управляти та швидко отримувати доступ до векторів, що особливо корисно для завдань машинного навчання та аналізу великих даних.

Pinecone забезпечує швидкий пошук близьких за значеннями векторів, що робить його ідеальним для систем рекомендацій, пошуку зображень або текстів за схожістю та інших аналітичних завдань, де швидкість і точність важливі.

API OpenAI: це інтерфейс програмування застосунків, який дозволяє розробникам інтегрувати штучний інтелект OpenAI в свої програми та сервіси.

Цей API включає доступ до різних моделей машинного навчання, таких як GPT (Generative Pre-trained Transformer) для обробки мови, DALL-E для генерації зображень та інших алгоритмів. Використання API OpenAI може значно покращити можливості програмного забезпечення, додаючи функції, які реалізують розуміння мови, автоматичну відповідь на запитання, резюмування текстів, створення контенту та багато іншого.

Swagger: це набір інструментів для проектування, побудови, документування та використання RESTful веб-сервісів.

Основна мета Swagger полягає в тому, щоб допомогти розробникам легко управляти API їхніх застосунків протягом усього життєвого циклу розробки. Інструменти Swagger дозволяють автоматично генерувати інтерактивну документацію, SDK та інтерфейси користувачів на основі вашого API, що полегшує тестування і інтеграцію. Swagger часто використовується для спрощення розробки API та забезпечення зручності використання та сумісності між різними системами.

1.5.3 Розробка сервера та інтеграція

Сервер буде розроблений за наступними етапами:

1. Налаштування та конфігурація: Встановлення Node.js, NestJS, MongoDB та Pinecone. Конфігурація середовища розробки та ініціалізація структури проекту.
2. Розробка API: Створення кінцевих точок для різних функцій, включаючи підключення до API OpenAI для інтеграції LLM. Використання Swagger для документування API.
3. Інтеграція баз даних: Впровадження моделей даних та схем для MongoDB та конфігурація Pinecone для зберігання та пошуку векторних даних.

4. Тестування: Проведення всебічного тестування для забезпечення функціональності, продуктивності та безпеки. Розгортання сервера в продуктивне середовище.

1.5.4 Реалізація сервера

1.Налаштування середовища розробки NodeJs і Docker

Для початку розробки серверу необхідно налаштувати локальне середовище, яке буде включати VSCode (редактор коду), Node.js та Docker. Це забезпечить стабільність і відтворюваність середовища розробки та розгортання.

Встановлення Node.js: Завантажте та встановіть останню стабільну версію Node.js з офіційного сайту або використав програму NVM.Щоб підтвердити, чи Nodejs встановлений, і визначити версію, ми вводимо команду `node -v`.

Встановлення Docker: Встановіть Docker для вашої операційної системи. Це дозволить створювати та керувати контейнерами для різних сервісів. Для перевірки встановленої версії: `docker version`.

Налаштування Docker Compose: Створіть файл docker-compose.yml для налаштування сервісів, які будуть використовуватись у проекті. Для перевірки встановленої версії: `docker compose version`.

1.5.5 Ініціалізація NestJs

NestJs є прогресивним фреймворком для створення ефективних, масштабованих Node.js серверних додатків.

Ініціалізація проекту: Створіть нову папку для вашого проекту та перейдіть у неї у терміналі. Виконайте команду npm init -y - ініціалізація нового проект з npm .

Ініціалізація NestJs: Щоб розгорнути проект за допомогою Nest CLI, виконайте команди `npm i -g @nestjs/cli` та `nest new project-name`. Це створить новий каталог проекту та заповнить його початковими базовими файлами Nest

і допоміжними модулями, формуючи стандартну базову структуру для вашого проекту. Створення нового проекту з використанням Nest CLI рекомендується новим користувачам. Ми продовжимо використовувати цей підхід у перших кроках.

1.5.6 Налаштування структури проекту

Для якісної розробки додамо необов'язкові, але важливі елементи, які сприятимуть процесу розробки. Встановимо та налаштуємо такі програми, як ESLint та Prettier — ці інструменти допоможуть валідувати та форматувати код. Інструкції щодо їхнього встановлення можна знайти на відповідних вебсайтах.

Для локального запуску бази даних MongoDB необхідно створити dockerfile-compose.yml file:

```
services:
  mongodb:
    image: mongo:latest
    environment:
      - MONGODB_DATABASE="mongodb"
    ports:
      - 27017:27017
```

Структура проекта представлена на рисунку 1.2.

На рисунку 1.2 представлена структура проекту NestJs з різними файлами та папками. Ось детальний опис кожного елементу:

Папки:

1. dist**:** Ця папка містить згенеровані файли JavaScript після компіляції TypeScript файлів. Вона створюється під час виконання `npm run build` або аналогічної команди.
2. node_modules: Тут знаходяться всі встановлені пакети та залежності для проекту, які завантажуються через npm або yarn.
3. src**:** Це основна папка з вихідними кодами проекту. Тут зберігаються всі модулі, контролери, сервіси та інші компоненти додатку.

4. test**:** Папка для тестів, де зберігаються тестові файли для різних модулів проекту.

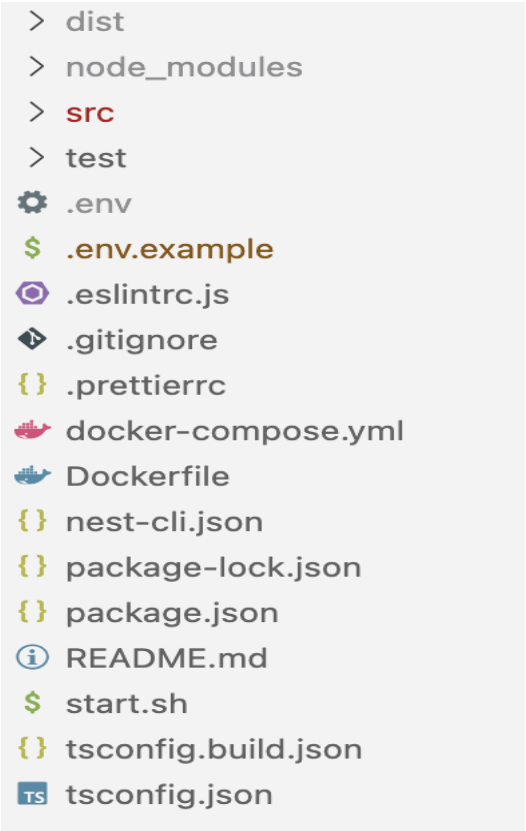


Рисунок 1. 2. Структура проекта

Файли:

1. .env**:** Файл конфігурації середовища, де зберігаються змінні середовища, які використовуються в проекті.
2. .env.example**:** Приклад файлу `.env`, що містить шаблон змінних середовища, які повинні бути вказані в реальному `.env` файлі.
3. .eslintrc.js**:** Конфігураційний файл ESLint, який визначає правила та налаштування для лінтингу коду.
4. .gitignore: Файл, який визначає, які файли і папки слід ігнорувати в системі контролю версій Git.
5. .prettierrc: Конфігураційний файл Prettier, що використовується для форматування коду.

6. docker-compose.yml**:** Файл конфігурації Docker Compose, який описує сервіси, мережі та томи для Docker контейнерів.
7. Dockerfile**:** Файл, який містить інструкції для створення Docker образу для цього проекту.
8. nest-cli.json**:** Конфігураційний файл для Nest CLI, що використовується для налаштування поведінки інструментів командного рядка Nest.
9. package-lock.json**:** Автоматично згенерований файл, який фіксує версії встановлених залежностей для забезпечення детермінованих установок у майбутньому.
10. package.json**:** Основний файл конфігурації npm проекту, що містить метадані проекту, скрипти, залежності та іншу інформацію.
11. README.md**:** Файл документації проекту, де зазвичай міститься інформація про проект, інструкції з встановлення та використання.
12. start.sh: Скрипт для запуску або налаштування проекту.
13. tsconfig.build.json**:** Конфігураційний файл TypeScript для налаштувань компіляції при створенні збірки.
14. tsconfig.json**:** Основний конфігураційний файл TypeScript, що визначає налаштування компілятора для проекту.

Ця структура допомагає організувати проект таким чином, щоб забезпечити його легкість в підтримці, масштабованість та зрозумілість для розробників.

1.5.7 Створення ендпоїнтів

Ендпоїнти - це точки входу для клієнтських запитів до вашого серверу.

Створення контролерів: Для створення нових контролерів була використана команда з фреймворка NestJs *nest g controller controller-name.*

- Було створено три контролери.

embedding: відповідає за роботу з базою даних Pinecone для ембедингу даних.

openai: для роботи з API OpenAI.

stories: для роботи з базою даних MongoDB, з метою зберігання певних даних.

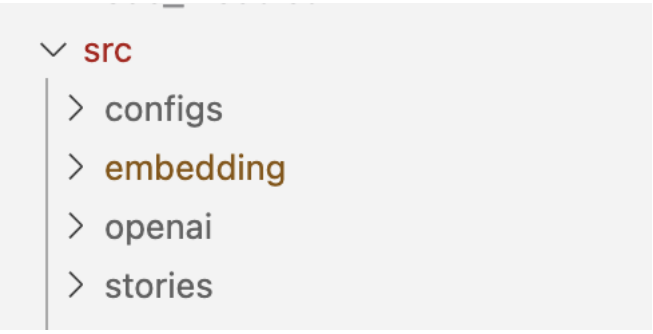


Рисунок 1.3 Контролери маршрутів

1.5.8 Інтеграція бази даних MongoDB та Pinecone

MongoDB: Використовуйте бібліотеку Mongoose для взаємодії з MongoDB.

Встановлення Mongoose: Виконайте команду npm install mongoose для встановлення бібліотеки.

Підключення до MongoDB: Налаштуйте підключення до бази даних у файлі конфігурації.

Створення схем: Визначте схеми даних за допомогою Mongoose для структурування даних у базі.

Pinecone: Pinecone використовується для роботи з векторними даними.

Інтеграція Pinecone**:** Використовуйте офіційну бібліотеку або API для підключення до Pinecone.

Конфігурація Pinecone: Налаштуйте з'єднання з Pinecone у вашому проекті.

Підключення та використання OpenAI API:

Реєстрація та отримання API ключа**:** Зареєструйтесь на платформі OpenAI та отримайте API ключ для доступу до сервісів OpenAI.

Встановлення клієнтської бібліотеки: Виконайте команду npm install openai для встановлення бібліотеки.

Конфігурація API ключа: Додайте API ключ до файлу конфігурації вашого проекту.

Використання OpenAI API **с**творіть сервіси або контролери для взаємодії з OpenAI API, виконуючи запити для отримання відповідей від моделей ШІ.

1.6 Інтеграція векторної бази даних Pinecone

Векторна база даних Pinecone необхідна для зберігання і обробки векторів, що генеруються моделями глибинного навчання для семантичного пошуку і векторних вбудовувань (*embedding)*. Використання Pinecone разом з моделями OpenAI дозволяє здійснювати високоточні семантичні пошуки, знаходячи схожі об'єкти на основі їхніх векторних представлень.

Retrieval-Augmented Generation (RAG) в Large Language Models (LLM)

RAG є технологією, яка поєднує можливості витягування інформації з великих баз даних (retrieval) із генерацією тексту на основі моделей машинного навчання. Це дозволяє моделям LLM не лише генерувати текст на основі внутрішніх знань, отриманих під час тренування, але й динамічно інтегрувати зовнішні дані, що робить відповіді більш точними та інформативними. Ось основні переваги використання RAG в LLM:

1. Покращена точність інформації: Завдяки доступу до великої бази даних, RAG може забезпечити актуальні та точні дані, що значно покращує якість відповідей.
2. Більш глибоке розуміння контексту**:** RAG дозволяє моделям адаптувати відповіді з урахуванням специфічної інформації, яку можна отримати з зовнішніх джерел, тим самим збагачуючи контекстуальне розуміння запитань.
3. Гнучкість у відповідях: Моделі, що використовують RAG, можуть генерувати відповіді на запитання, які вимагають актуальних даних або специфічних знань, яких немає у тренувальних даних.

Вибір Pinecone як векторної бази даних. Хоча на ринку існує багато векторних баз даних, ми вибрали Pinecone з кількох причин:

1. Швидкість і масштабованість**:** Pinecone оптимізована для швидких операцій з векторами, що є критично важливим для ефективного витягування даних у реальному часі.

2. Точність витягування даних: Pinecone має високу точність при роботі з векторними запитами, що забезпечує більш точне співставлення запитів і даних.

3. Легкість інтеграції: Pinecone легко інтегрувати з різними програмними середовищами, що дозволяє гладко вбудовувати її у різні системи, не вдаючись до складних налаштувань.

Ці характеристики роблять Pinecone ідеальним вибором для підтримки розширених можливостей RAG у LLM, забезпечуючи швидке та ефективне витягування даних, необхідних для генерації відпов

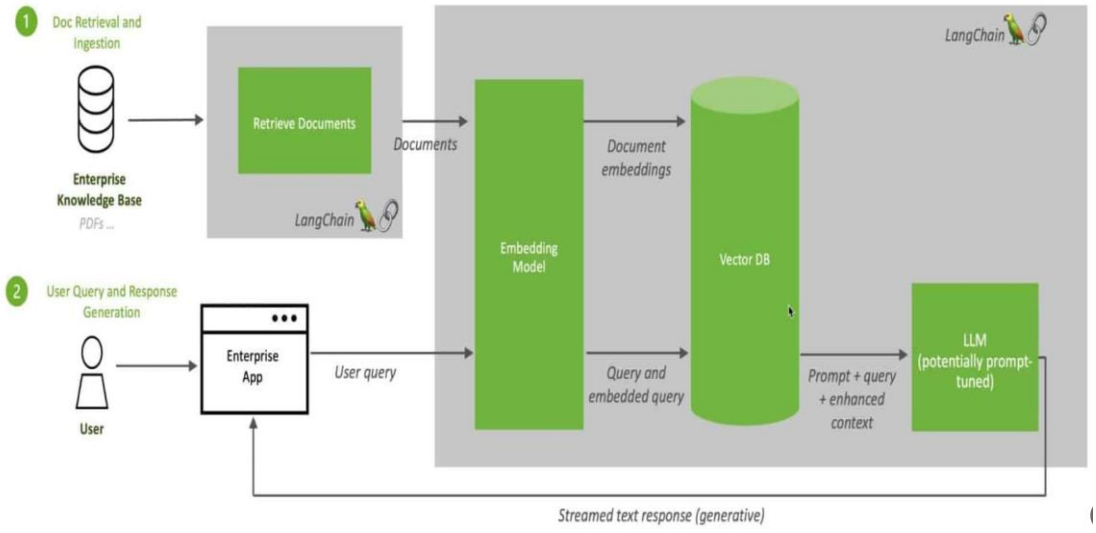


Рисунок 1.4. Retrieval-Augmented Generation

Крок 1: Встановлення необхідних залежностей

Встановили необхідні пакети для роботи з Pinecone та OpenAI.

npm install pinecone-client opena

Крок 2: Налаштування Pinecone

Створили модуль embedding для управління підключенням до Pinecone та процессом.

nest generate module embedding
nest generate service embedding
nest generate controller embedding

Цей код визначає модуль `EmbeddingModule` у проекті NestJS.

```
edding > TS embedding.module.ts > ...
import { Module } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
import { EmbeddingController } from './embedding.controller';
import { EmbeddingService } from './embedding.service';

@Module({
  providers: [ConfigService, EmbeddingService],
  controllers: [EmbeddingController],
})
export class EmbeddingModule { }
```

Рисунок 1.5. Скриншот кода модуля EmbeddingModule

Module: Імпортується з @nestjs/common і використовується для декларування модулів у NestJS.

ConfigService: Імпортується з @nestjs/config і використовується для доступу до конфігураційних змінних.

EmbeddingController та EmbeddingService: Імпортуються з локальних файлів і представляють контролер та сервіс для вбудовувань відповідно.

@Module(): Це декоратор, який перетворює клас у модуль NestJS. Він приймає об'єкт з конфігурацією модуля.

providers: Це масив сервісів, які будуть створені в цьому модулі і доступні для ін'єкції через Dependency Injection (DI). В даному випадку, це ConfigService і EmbeddingService.

ConfigService: Дозволяє отримувати доступ до конфігураційних змінних у всьому додатку.

EmbeddingService: Сервіс, який, ймовірно, містить бізнес-логіку для роботи з вбудовуваннями.

controllers: Це масив контролерів, які будуть зареєстровані в цьому модулі. В даному випадку, це EmbeddingController.

EmbeddingController: Контролер, який обробляє HTTP-запити і повертає відповіді клієнтам.

EmbeddingModule**:** Оголошення класу модуля, який експортований і може бути імпортований іншими модулями в додатку.

1.7 Методи сервіса EmbeddingService

Для взаємодії з сервісами бази даних та ембедингу мовних моделей був використаний фреймворк Langchain. У цьому файлі зосереджена вся бізнес-логіка модуля. Він містить два методи:

create - отримує дані, вбудовує (Embedding) за допомогою LLM та записує дані разом із їх векторами у векторну базу даних Pinecone. Для спрощення роботи було використано фреймворк Langchain.

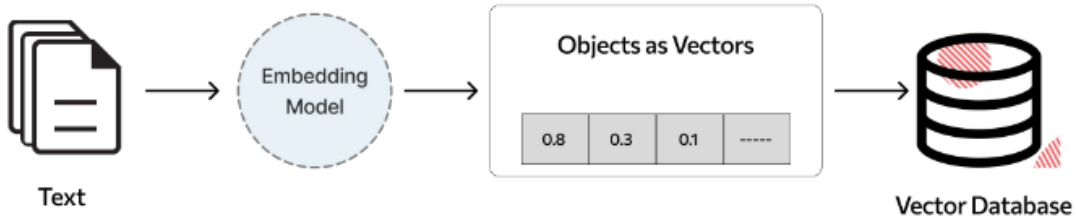


Рисунок 1.6. Схема роботи методу create

```
create(createEmbeddingDto: CreateEmbeddingDto) {
  const { pageContent, metadata } = createEmbeddingDto;
  const docs = [
    new Document({
      metadata,
      pageContent,
    }),
  ];
  return PineconeStore.fromDocuments(docs, this.
  openAIEmbeddings, {
    pineconeIndex: this.pinecone.Index(this.pineconeIndex),
    maxConcurrency: 5,
  });
}
```

Рисунок 1.7 Скриншот кода метода Create

Цей метод відповідає за створення нового вкладення документу. Роз'яснення його функціональності:

Аргумент: Метод приймає аргумент під назвою `createEmbeddingDto`, який є екземпляром класу `CreateEmbeddingDto`. Цей клас функціонує як Об'єкт Передачі Даних (DTO), що інкапсулює дані, необхідні для створення нового вкладення документу.

Вилучення даних: DTO розкладається для вилучення властивостей pageContent і metadata, які потім використовуються для створення нового об'єкту Document.

Створення документа: Створюється об'єкт Document з вилученими pageContent та metadata. Цей об'єкт зберігається в масиві під назвою docs.

Клас документа: Клас Document є загальним класом, який приймає параметр типу Metadata і реалізує два інтерфейси: DocumentInput та DocumentInterface.

Зберігання документів: Викликається метод PineconeStore.fromDocuments з трьома аргументами: масивом docs, this.openAIEmbeddings, та об'єктом опцій. Цей метод, ймовірно, відповідає за зберігання документів у базі даних Pinecone.

Query - отримує запит від клієнта, виконує семантичний пошук у векторній базі даних, отримує релевантну інформацію и повертає користувачеві. Скриншот код метода Query

```
async query(queryEmbeddingDto: QueryEmbeddingDto) {
  const { query, metadata } = queryEmbeddingDto;
  const vectorStore = await PineconeStore.fromExistingIndex(
    this.openAIEmbeddings,
    { pineconeIndex: this.pinecone.Index(this.pineconeIndex) },
  );
  return vectorStore.similaritySearch(query, 1, metadata);
}
```

Рисунок 1.8 Скриншот код метода Query

Зображення показує метод `query` виконує пошук схожості серед документів з використанням Pinecone бази даних. Метод використовує векторний пошук для знаходження документів, схожих на заданий запит. Логіка методу:

1. Метод `query` приймає один параметр — `queryEmbeddingDto`, який є екземпляром класу `QueryEmbeddingDto`. Цей об'єкт містить дані запиту, які будуть використані для пошуку схожості.

2. З `queryEmbeddingDto` витягуються `query` та `metadata`. Це дані, які будуть використовуватися для визначення схожості.

3. Метод використовує `PineconeStore.fromExistingIndex` для ініціалізації `vectorStore` з існуючого індексу, який пов'язаний з `this.openAIEmbeddings` та конкретним `pineconeIndex`.

4. `vectorStore.similaritySearch`, використовується для пошуку схожих документів на основі запиту. Він передає вектор запиту, кількість результатів (у цьому випадку 1) та метадані для пошуку.

Цей метод є частиною більшої системи рекомендацій або пошукової системи, яка використовує машинне навчання та векторні бази даних для забезпечення релевантних та ефективних результатів пошуку на основі семантичної схожості.

Для цих методів у контролері EmbeddingController створено два ендпойнти.

```
@Post('create')
async create(@Body() createEmbeddingDto: CreateEmbeddingDto) {
  try {
    await this.embeddingService.create(createEmbeddingDto);
    return 'Embedding created successfully';
  } catch (error) {
    console.log(error);
    return error;
  }
}

@Post('query')
async query(@Body() queryEmbeddingDto: QueryEmbeddingDto) {
  try {
    const result = await this.embeddingService.query
    (queryEmbeddingDto);
    console.log(result);
    return result;
  } catch (error) {
    console.log(error);
    return error;
  }
}
```

Рисунок 1.9 Скриншот ендпойнти контролера EmbeddingController

На зображенні відображені два методи контролера в NestJS-додатку: create та query. Ці методи відповідають за обробку POST-запитів для створення нових вкладень та запитів до вже існуючих вкладень відповідно.

Ось детальний опис кожного методу:

Метод create:

- Декоратор: @Post('create') вказує, що цей метод обробляє POST-запити на шляху /create.
- Асинхронна функція: Метод приймає параметр createEmbeddingDto типу CreateEmbeddingDto, який містить дані для створення нового вкладення.

Обробка запиту: Викликається метод create з сервісу embeddingService, передаючи createEmbeddingDto як аргумент.

Обробка помилок: У випадку виникнення помилки, вона логується та повертається як результат методу.

Повернення результату: Якщо операція успішна, метод повертає рядок "Embedding created successfully".

Метод query

Декоратор: @Post('query') вказує, що цей метод обробляє POST-запити на шляху /query.

Асинхронна функція: Метод приймає параметр queryEmbeddingDto типу QueryEmbeddingDto, який містить дані для запиту схожих вкладень. Обробка запиту: Викликається метод query з сервісу embeddingService, передаючи queryEmbeddingDto як аргумент. Обробка помилок: У випадку виникнення помилки, вона логується та повертається як результат методу. Повернення результату: Метод повертає результат запиту.

Інтеграція бази даних MongoDB

Крок 1: Встановлення залежностей MongoDB

Встановили необхідні пакети MongoDB.

npm install @nestjs/mongoose mongoose

Крок 2**.** Налаштування підключення до MongoDB

Згенерували модуль

nest generate module stories

Налаштували підключення у файлі stories.module.ts. Скриншот кода налаштування підключення у файлі.

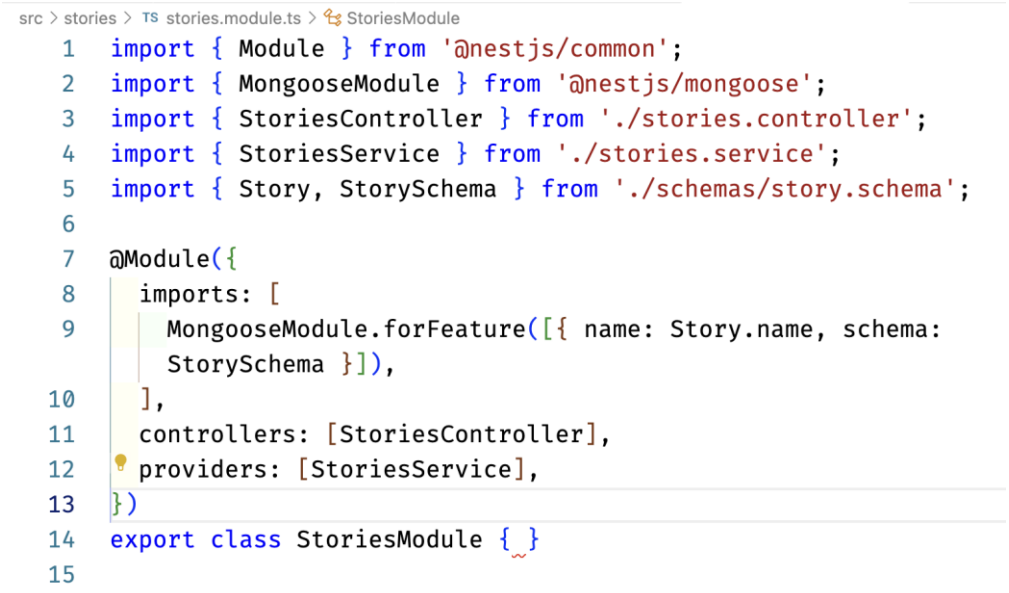


Рисунок 1.9 Скриншот кода налаштування підключення у файлі.

Крок 3. Створення схеми.

Згенерували новий модуль і сервіс для сутності, яку хочемо зберігати в MongoDB.

nest generate service stories

nest generate controller stories

У директорії модуля stories створили директорію schemas і додали файл схеми, story.schema.ts.

Визначили схему:

Крок 4 Реалізація методів сервісу

Оновили файл stories.service.ts, щоб включити CRUD операції з використанням Mongoose.

CRUD операції (Create, Read, Update, Delete) – це основні операції, які використовуються для управління даними в базі даних. Давайте детальніше розглянемо реалізацію CRUD операцій у модулі stories для додатку NestJS з MongoDB.

Create (Створення)

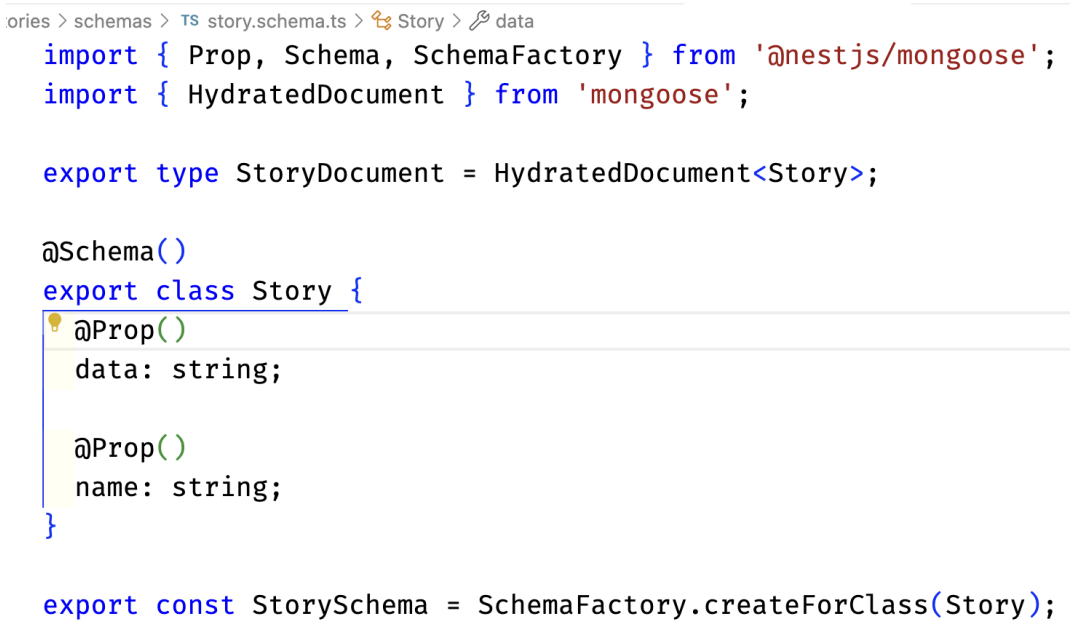


Рисунок 1.10. Скриншот схеми бази даних MongoDB

Операція створення нового запису в базі даних.

Метод у сервісі StoriesService:

```
async create(createCatDto: CreateStoryDto): Promise<Story> {
  const createdStory = await this.storyModel.create(createCatDto);
  return createdStory;
}
```

Цей метод приймає об'єкт CreateStoryDto, який є типом CreateStoryDto. DTO (Data Transfer Object) використовується для перенесення даних між процесами. В даному випадку, CreateStoryDto просто відповідає структурі даних, необхідних для створення нового об'єкта Story. Виконується метод create моделі storyModel, передаючи createCatDto як аргумент. Використання ключового слова await зупиняє виконання функції, поки не виконується проміс, що повертає storyModel.create(createCatDto). Функція завершується поверненням createdStory, що є новоствореним об'єктом Story, створеним за допомогою даних з createCatDto та зберігає його в базі даних.

Read (Читання)

Операції читання дозволяють отримувати дані з бази даних.

Метод для отримання всіх записів у сервісі StoriesService:

```
async findAll(): Promise<Story[]> {
  return this.storyModel.find().exec();
 }
```

Цей метод повертає всі записи з колекції *stories* в базі даних.

Метод для отримання одного запису за ідентифікатором у сервісі *StoriesService:*

```
async findOne(id: string): Promise<Story> {
  return this.storyModel.findOne({ _id: id }).exec();
 }
```

Update (Оновлення)-операція оновлення існуючого запису в базі даних.

Метод у сервісі StoriesService для оновлення запису:

```
async update(id: string, createCatDto: CreateStoryDto): Promise<Story> {
  return await this.storyModel
    .findOneAndUpdate({ _id: id }, createCatDto, { new: true })
    .exec();
}
```

Цей метод оновлює запис за ідентифікатором та повертає оновлений запис.

Delete (Видалення)

Операція видалення запису з бази даних.

1. Метод у сервісі StoriesService для видалення запису:

```
async delete(id: string) {
  return await this.storyModel.findOneAndDelete({ _id: id }).exec();
 }
```

Цей метод видаляє запис з колекції stories за його ідентифікатором.

Ці CRUD операції дозволяють створювати, читати, оновлювати та видаляти записи в базі даних MongoDB через модуль stories у додатку NestJS.

Крок 5 Реалізація методів контролера

Ендпойнти контролера для CRUD-операцій сервісу

```
@Controller('stories')
export class StoriesController {
  constructor(private readonly storiesService: StoriesService) { }
  @Post()
  async create(@Body() createCatDto: CreateStoryDto) {
    await this.storiesService.create(createCatDto);
  }
  @Put(':id')
  async update(@Param('id') id: string, @Body() createCatDto:
  CreateStoryDto) {
    return this.storiesService.update(id, createCatDto);
  }
  @Get()
  async findAll(): Promise<Story[]> {
    return this.storiesService.findAll();
  }
  @Get(':id')
  async findOne(@Param('id') id: string): Promise<Story> {
    return this.storiesService.findOne(id);
  }
  @Delete(':id')
  async delete(@Param('id') id: string) {
    return this.storiesService.delete(id);
  }
}
```

Рисунок 1.11. Скриншот коду Контролери StoriesController

1.8 Підключення та використання OpenAI API

1.8.1 Налаштування доступу до OpenAI API:

1. Реєстрація на платформі OpenAI та створення акаунта

Було здійснено реєстрацію на веб-сайті OpenAI. Після реєстрації створено новий проект та отримано API-ключ, який забезпечив доступ до сервісів OpenAI.

API-ключ було отримано для автентифікації запитів до API OpenAI. У особистому кабінеті OpenAI було вибрано вкладку "API Keys", де створено новий ключ.

Щоб уникнути потрапляння ключів та секретних даних у git-репозиторій, було використано функціонал NestJS для зберігання цих даних у файлі .env.

example

```
PORT=3000

OPEN_AI_KEY=sk- ...

#emmbedding model
modelName=text-embedding-ada-002

MONGODB_URI=mongodb://localhost:27017/mongodb

PINECONE_INDEX=
PINECONE_API_KEY=
PINECONE_ENVIRONMEN=gcp-starter
```

Рисунок 1.12. Скриншот файлу env.example

Текст із зображення описує вміст файлу .env.example, який є шаблоном для створення файлу оточення в проектах програмного забезпечення. Ці змінні оточення дозволяють зберігати конфіденційні дані та налаштування, необхідні для функціонування додатку, і варіювати їх залежно від середовища (розробка, стейджинг, продакшн). Ось детальний опис кожної змінної, представленої у файлі:

PORT=3000: Задає порт, на якому запускається ваш застосунок. У даному випадку встановлено порт 3000.

OPEN_AI_KEY=sk-...: Це ключ API для доступу до сервісів OpenAI. Вам потрібно замінити sk-... на ваш фактичний API-ключ.

#embedding model: Це коментар, який не обробляється додатком. Він дає контекст розробникам і вказує, що наступний параметр пов'язаний з моделлю вкладення.

modelName=text-embedding-ada-002: Вказує назву моделі, яка використовується в застосунку для виконання певних операцій. Тут використовується модель text-embedding-ada-002.

MONGODB_URI=mongodb://localhost:27017/mongodb: Це рядок підключення до бази даних MongoDB. Він вказує, що база даних розташована локально на порту 27017 і має назву mongodb.

PINECONE_INDEX=ai-course: Вказує індекс у базі даних Pinecone, який використовується для зберігання або виклику даних. Тут задано індекс ai-course.

PINECONE_API_KEY=: Тут потрібно вставити ваш ключ API для доступу до сервісів Pinecone.

PINECONE_ENVIRONMENT=gcp-starter: Вказує середовище, в якому запущено Pinecone. У цьому прикладі використовується gcp-starter, що може відноситися до стартового пакету на Google Cloud Platform.

Файл .env.example служить основою для створення файлу .env, в який ви вводите реальні значення змінних. Файл .env зазвичай не додається до системи контролю версій, щоб уникнути витоку конфіденційної інформації.

Рисунок 1.13. Скриншот структури папки модуля OpenAI

Зображення показує структуру папки модуля OpenAI у проекті, розробленому з використанням TypeScript та NestJS. Ось детальний опис файлів, представлених у папці:

1. Папка dto (Data Transfer Object)

completion.dto.ts**:** Цей файл містить клас або інтерфейс, який визначає структуру даних для запитів генерації тексту в OpenAI API. DTO використовується для передачі даних між клієнтом і сервером, забезпечуючи валідацію та типізацію вхідних даних для API.

2. Файли специфікації та імплементації

openai.controller.spec.ts: Цей файл містить юніт-тести для контролера OpenAI. Тести перевіряють, що контролер коректно обробляє вхідні запити та відповіді.

openai.controller.ts**:** Файл контролера, який обробляє HTTP-запити до OpenAI API. Контролер приймає запити, передає дані до сервісного шару для обробки та повертає відповіді клієнту.

openai.module.ts: Модуль, що інкапсулює всі компоненти OpenAI (сервіси, контролери) і можливі імпорти з інших модулів. Модуль забезпечує ізоляцію та організацію функціоналу, пов'язаного з OpenAI.

openai.service.spec.ts**:** Файл з тестами для сервісу OpenAI. Ці тести забезпечують перевірку логіки, що використовується сервісом для взаємодії з API OpenAI.

openai.service.ts**:** Сервіс, що містить бізнес-логіку для взаємодії з OpenAI API. Сервіс виконує запити до API, обробляє відповіді та передає їх до контролера.

Ця структура папок і файлів демонструє чітке відділення відповідальності між різними компонентами додатку, дозволяючи легше управління кодом і підтримку.

1.8.2 Опис API за допомогою Swagger

Swagger забезпечує документацію API, яка допомагає розробникам легко розуміти та тестувати кінцеві точки API. Swagger є потужним інструментом для документації API, який має кілька значущих переваг:

Спрощення співпраці: Swagger уніфікує документацію API, забезпечуючи точне і зрозуміле описання методів, параметрів та моделей даних. Це дозволяє розробникам швидко розуміти та використовувати API, не заглиблюючись у код.

Інтерактивність: Завдяки Swagger UI розробники мають змогу виконувати запити до API прямо з веб-інтерфейсу, що значно спрощує тестування та відладку API.

Універсальність: Swagger підтримує багато мов програмування та фреймворків, що робить його відмінним вибором для проектів різної складності.

Автоматизація: З документацією Swagger можливе автоматичне генерування коду клієнтської частини для різних платформ, що значно знижує час на розробку.

Підтримка стандартів: Swagger дотримується специфікації OpenAPI, яка є широко прийнятим стандартом для опису RESTful API.

Підготовка до виконання:

Першочергово були встановлені необхідні пакети для роботи з Swagger у NestJS. Виконано команду:

npm install --save @nestjs/swagger swagger-ui-express

Це дозволило інтегрувати необхідні залежності до проекту, що є основою для подальшої конфігурації Swagger.

Конфігурація Swagger модуля

Конфігурація Swagger модуля була виконана у файлі ініціалізації додатка main.ts.

Визначено базові параметри документації API:

Текст з зображення описує код TypeScript, який є вхідною точкою додатку NestJS з налаштуванням документації Swagger. Ось детальний аналіз кожної частини коду:

main.ts > ...

```ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { ConfigService } from '@nestjs/config';
import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  const configService = app.get(ConfigService);
  const port = configService.get<number>('port');

  const config = new DocumentBuilder()
    .setTitle('Diplom test API')
    .setDescription('Diplom test API')
    .setVersion('1.0')
    .addTag('test')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api', app, document);

  await app.listen(port);
}
bootstrap();
```

Рисунок 1.14. Скриншот коду конфігурація Swagger модуля

1. Імпорт модулів

Код починається з імпорту необхідних модулів та компонентів:

NestFactory: Використовується для створення інстанції додатку NestJS.

AppModule**:** Це основний модуль додатку, який інтегрує всі інші модулі.

ConfigService**:** Модуль для завантаження та використання конфігураційних змінних.

SwaggerModule і DocumentBuilder: Модулі, використовувані для створення

документації API за допомогою OpenAPI (раніше відомого як Swagger).

2. Функція bootstrap

Це асинхронна функція, яка ініціалізує та запускає додаток:

Створення екземпляра додатку з використанням AppModule.

Отримання екземпляра ConfigService через метод app.get().

Використання ConfigService для отримання номеру порту, на якому має слухати додаток.

3. Налаштування Swagger

- DocumentBuilder**:** Конфігуратор для Swagger, в якому встановлюються заголовок, опис, версія документації та теги.
- Створення документа Swagger з використанням SwaggerModule.createDocument(), передаючи туди додаток та конфігурацію Swagger.
- Налаштування Swagger UI через SwaggerModule.setup(), вказуючи шлях до документації та асоціюючи його з додатком.

4. Запуск додатку.

Додаток запускається на вказаному порту з використанням app.listen(), що дозволяє обслуговувати вхідні HTTP-запити.

5. Виклик функції bootstrap

Виклик bootstrap() розпочинає виконання функції та ініціацію додатку.

Цей код є типовим прикладом налаштування NestJS-додатку з інтеграцією Swagger для документації API, що робить його доступним для розробників і сторонніх сервісів для легкого взаємодії з додатком. Swagger забезпечує чітке визначення ендпойнтів, параметрів запитів і форматів відповідей, що сприяє швидшій розробці та інтеграції.

1.9. Робота програми

Рисунок 1.15. Скриншот сторінки

Приклад роботи розробленої програми API:

1. Пошук інформації в наукових статтях:
   - Студенти можуть використовувати API для швидкого пошуку релевантної інформації у великих базах даних наукових статей, збережених у Pinecone та MongoDB.
   - Наприклад, пошук ключових досліджень про певну тему для написання рефератів або курсових робіт.
2. Автоматичне створення конспектів лекцій:
   - Запити до OpenAI можуть використовуватись для створення конспектів лекцій на основі матеріалів, знайдених через Pinecone.
   - Це дозволить студентам отримувати короткі, стислі резюме лекційного матеріалу.
3. Підготовка до іспитів;

   API може допомогти студентам готуватись до іспитів, відповідаючи на специфічні запитання на основі записаних лекцій та наукових матеріалів.

Використання CRUD-операцій для зберігання результатів підготовки та подальшого аналізу.

Приклад "Підготовка до іспитів"

Використання API для підготовки до іспитів

Студенти можуть використовувати API для оптимізації підготовки до іспитів, завантажуючи свої конспекти та матеріали, а потім виконуючи запити для отримання зведень, пошуку важливої інформації та генерації відповідей.

Функціональність API:

1.Завантажуємо та збереження матеріалів лекцій та додаткових матеріалів в Pinecone:

Студент завантажує свої конспекти, лекційні записи та інші матеріали через /embedding/create endpoint.

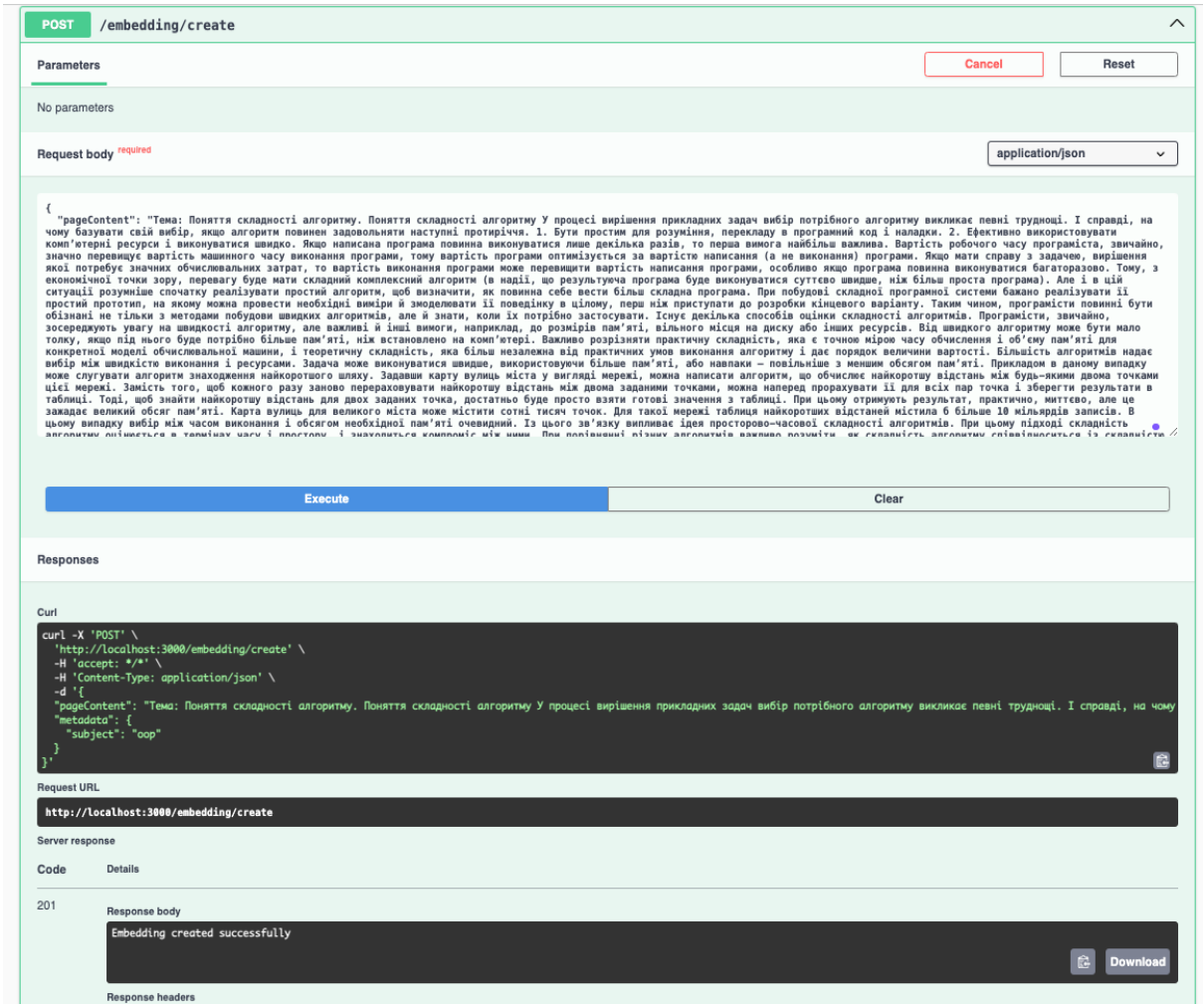


Рисунок 1.16. Скриншот завантаження

Завантажені дані зберігаються в Pinecone для подальшого використання і пошуку по них. Було використано базу даних на сайті: https://app.pinecone.io/

Дані конспектів обробляються для створення векторних представлень (embedding).

Векторні представлення зберігаються у Pinecone для швидкого та ефективного пошуку.

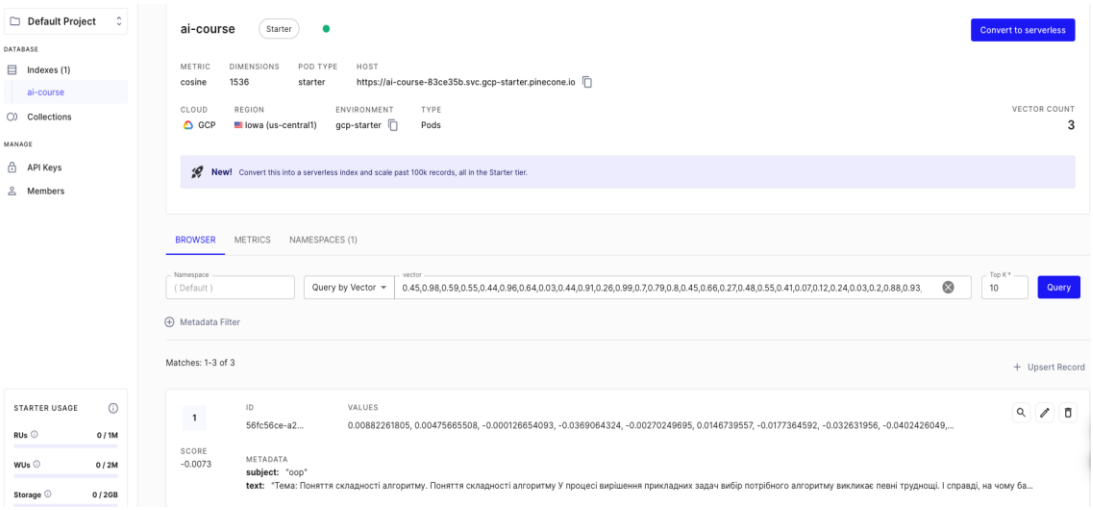


Рисунок1.17. Скриншот збереженої теми «Поняття складності алгоритму» в векторній БД.

2. Генерація відповідей:

Студент може виконати запит до API для отримання відповідей із завантажених матеріалів.API використовує OpenAI для аналізу та створення відповідей.

3. Зберігання та CRUD операції над відповідями та питаннями для подальшого використання:

ендпоінт POST /stories дозволяє зберігати створені питання та відповіді в MongoDB.

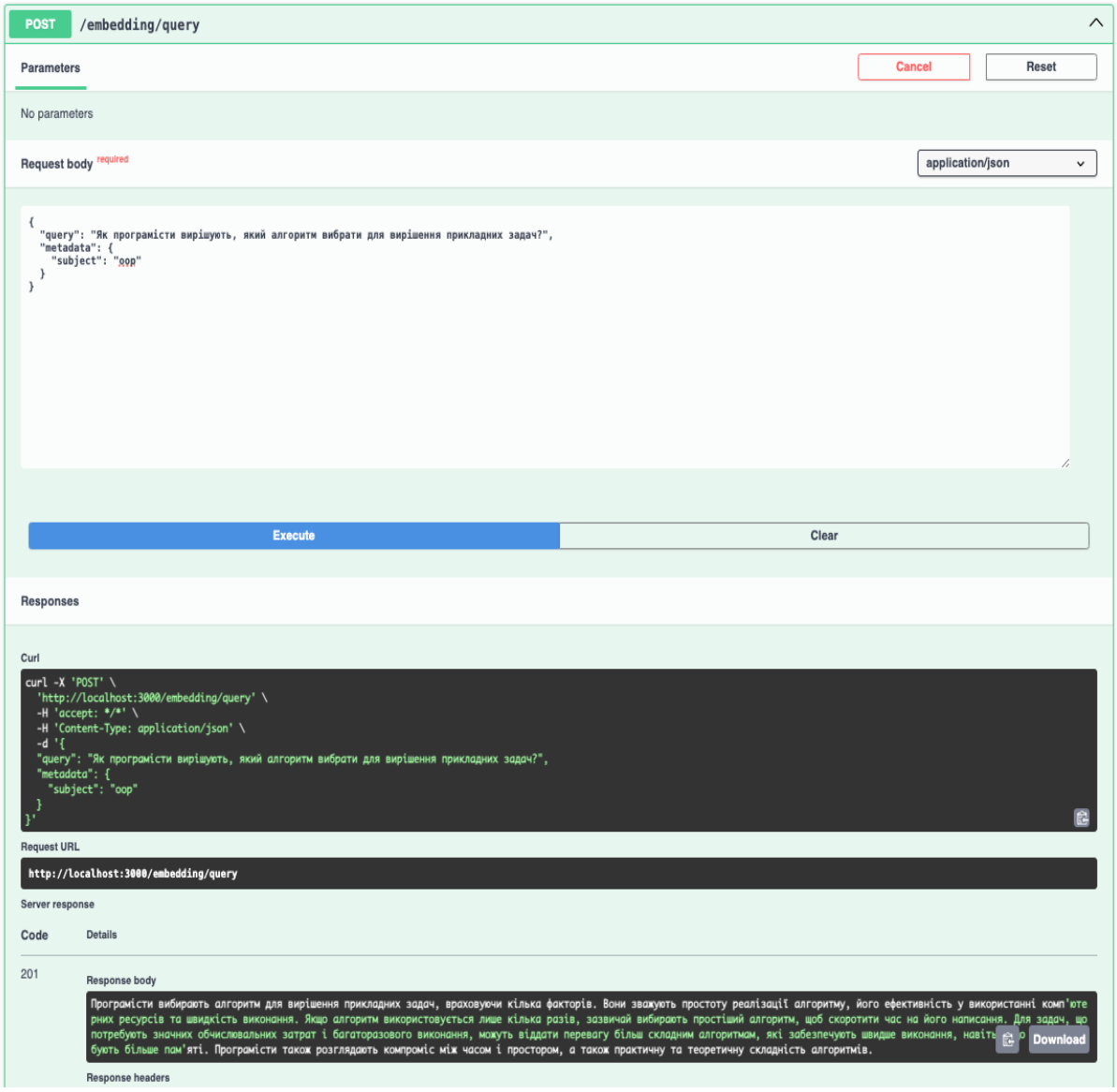


Рисунок 1.18 Скриншот API використовує OpenAI для аналізу та створення відповідей

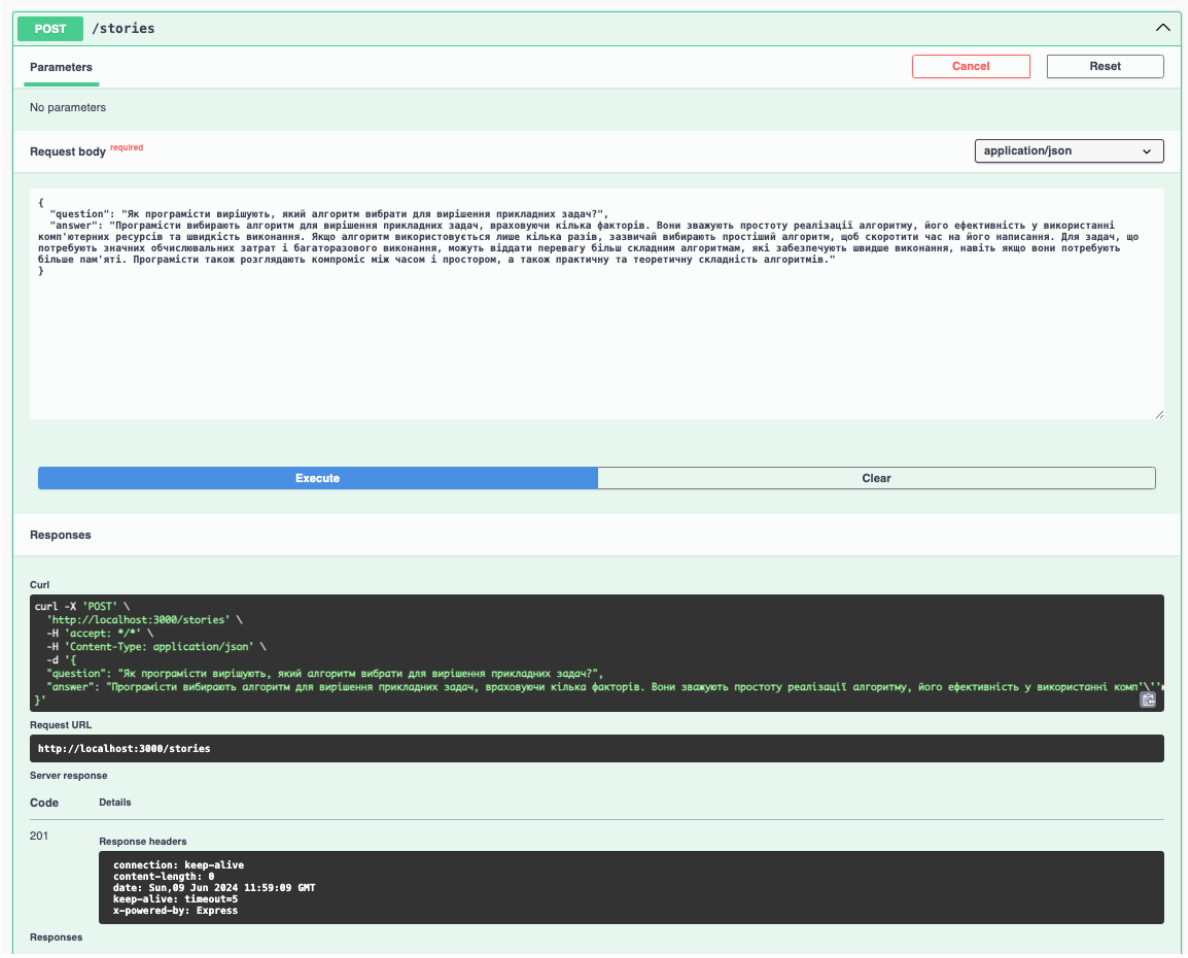


Рисунок 1.19 Скриншот зберігання питання та відповіді в MongoDB

API підтримує CRUD (створення, читання, оновлення, видалення) операції для ефективного управління даними.

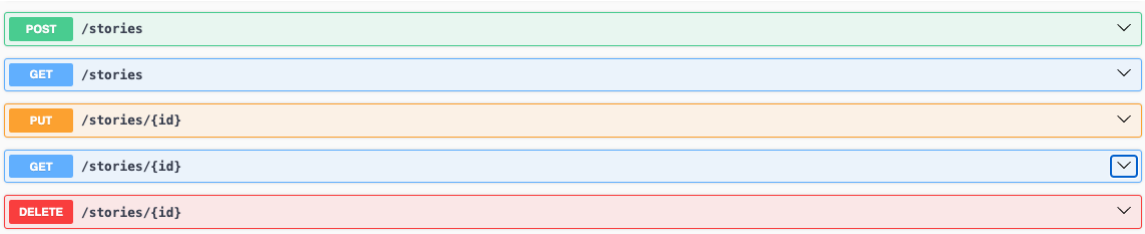


Рисунок 1.20. Скриншот CRUD операції над питаннями та відповідями в базі даних

Операція читання всіх даних із бази.

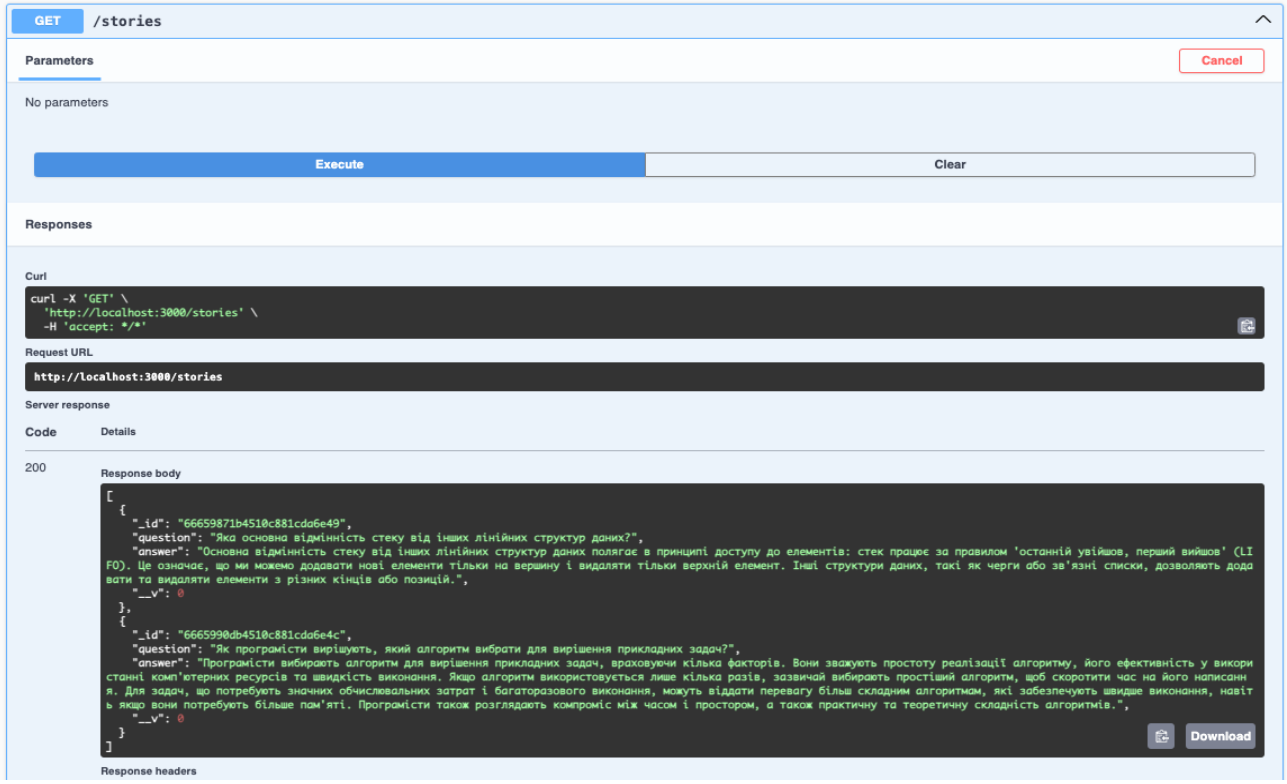


Рисунок 1.20. Скриншот усіх збереженних питаннь і відповідей в бд.

Запит: Створіть 3 запитання по збереженим темам, які можуть задати на іспиті.

Відповідь:

1. Поясніть різницю між теоретичною та практичною складністю алгоритму. Наведіть приклади, коли кожен з підходів до оцінки складності є корисним.
2. Опишіть класифікацію структур даних за наявністю зв'язків між елементами. Які існують приклади зв'язних і незв'язних структур даних?
3. Як обирати між простими та інтегрованими структурами даних при розробці алгоритмів? Наведіть приклади задач, де кожен тип структури є найбільш ефективним.

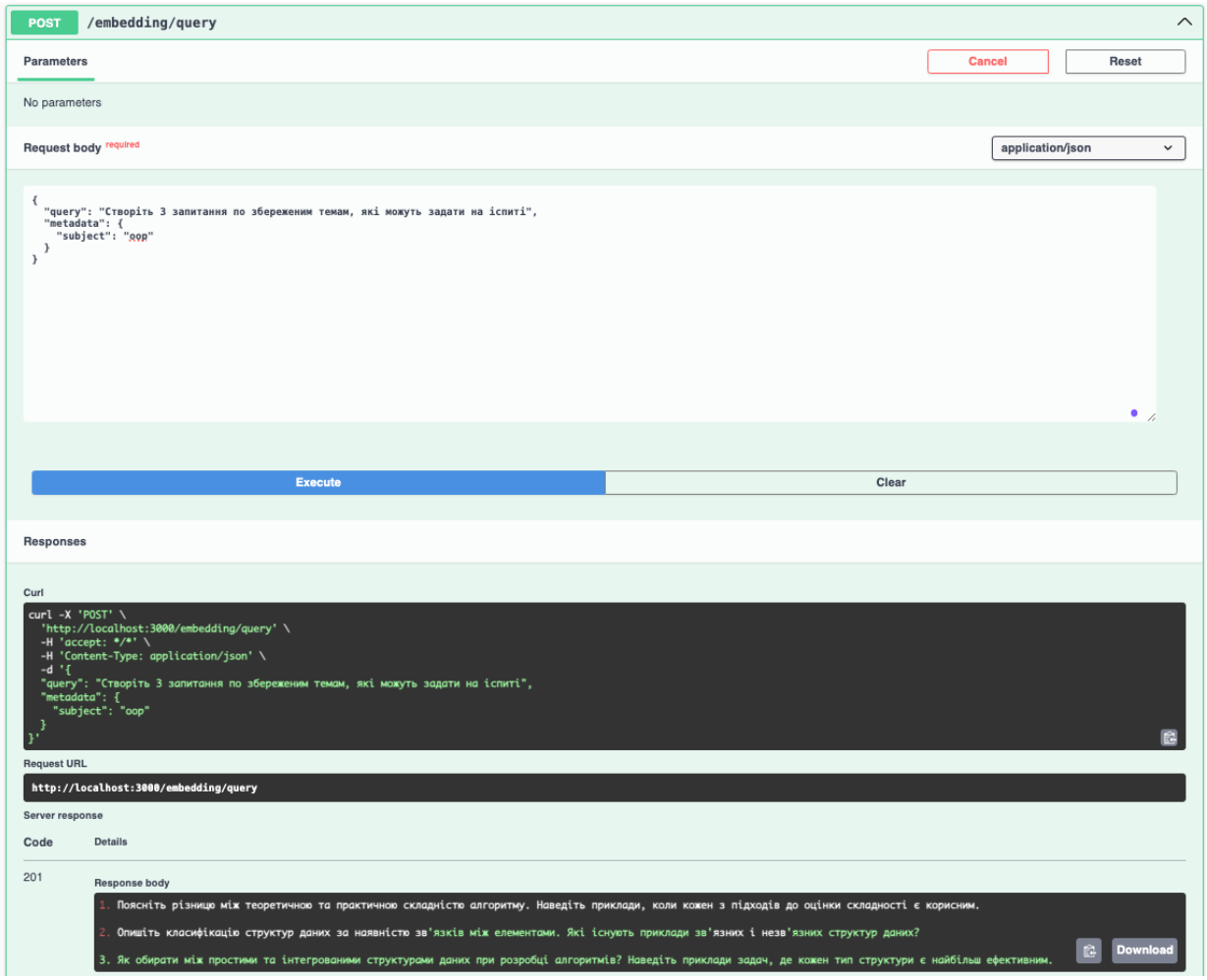


Рисунок 1.21 Скриншот відповіді

Запит: Створіть 3 запитання по збереженим темам, які можуть задати на іспиті

Відповідь:

1. Поясніть різницю між теоретичною та практичною складністю алгоритму. Наведіть приклади, коли кожен з підходів до оцінки складності є корисним.

2. Опишіть класифікацію структур даних за наявністю зв'язків між елементами. Які існують приклади зв'язних і незв'язних структур даних?

3. Як обирати між простими та інтегрованими структурами даних при розробці алгоритмів? Наведіть приклади задач, де кожен тип структури є найбільш ефективним.

2.ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою даного дипломного проекту є «Розробка сервера для цифрових помічників на основі штучного інтелекту». Програма такого сервера дозволяє нам забезпечити високу ефективність обробки даних і здійснення інтерактивного спілкування з користувачами.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначаємо обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт

Таблиця 2.1 Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП організації обчислювального процесу	13000 – 10200

Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): $T^a = 262 \times 0,8 = 209,6$ (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4.).

Таблиця 2.2.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Кодступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 209,6 *0,12*1,0= 25,15 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =209,6 *0,15*1,0 = 31,44 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т =209,6 *0,58*1,0*0,9=109,41 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стор), розробка ТП $N_{тп}$=18 (стор), розробка робочого проекту $N_{рп}$=25(стор), пояснювальна записка відповідно $N_{пз}$= 10 (стор) Розрахунок зведений у таблицю 2.5

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Pтз}$=25,15	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2=0,30
2.Розробка ТП	$T_{Pтп}$=31,44	$T_{кк}=0,7*N_{тп}$=0,7*18=12,6	$T_{нк}=0,15*N_{тп}$=0,15*18=2,7
3.Розробка РП	$T_{Pрп}$=109,41	$T_{кк}=0,7*N_{рп}$=0,7*25=17,5	$T_{нк}=0,15*N_{рп}$=0,15*25=3.8
4.Розробка ПЗ	$T_{пз}=1,5*$ $*N_{пз}$=1,5*10=15	$T_{кк}=0,7*N_{тз}$=0,7*10=7	$T_{нк}=0,15*N_{пз}$=0,15*10 =1,5
Усього, в т.ч.:	ΣT=227,80		
- на розробку	$\Sigma T_{р}$=181,00		
- контроль керівника		$\Sigma T_{кк}$=38,5	
- нормоконтроль			$\Sigma T_{нк}$=8,3

2.3 Розрахунок ціни програмного продукту.

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	181,00	50,00	9050,00
2.Контроль керівника	38,5	100.00	3850,00
3.Нормоконт-роль	8,3	100.00	830,00
Усього	-	-	ΣЗо= 13730,00

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.7

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	55	3.0	165
Разом	-	-	-	$В_{м\,і}$ =
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1 = 16{,}5$
Усього				$В_м = В_{мі} + В_{тр\_з=}$ 181,5

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	181,5	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	13730,00	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	2059,50	$Зд = 0{,}15 \times Зо$ =0,15*13730,00
4.Відрахування до єдиного фонду соціального внеску	3473,69	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (13730,00+2059,50)
5. Накладні витрати	4119,00	$Внак. = 0{,}3 \times Зо$ =0,3*13730,00
6. Повна собівартість	23563,69	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 181,5+13730,00+2059,50+3473,69+ 4119,00

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100=$ (23563,69*10)/100=2356,36 грн (2.6)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П =$ 23563,69+ 2356,36 грн =25920,05 грн (2.7)

Податок на додану вартість визначаємо по наступній формулі:

ПДВ = 0.2 * $Ц_о$ = 0,2*25920,05 = 5184,01 грн (2.8)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р=Ц_о+$ПДВ = 25920,05 + 5184,01 = 31104,06 грн (2.9)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Охорона праці на виробництві завжди була дуже важлива, отже саме завдяки рекомендаціям з охорони праці, персонал, який працює на підприємстві створює алгоритм виконання робочих завдань з чітким дотриманням рекомендацій. Основне завдання охорони праці – це створення та проведення заходів, спрямованих на захист життя, працездатності та здоров'я людини у процесі трудової діяльності.

При роботі з комп'ютером, як і в багатьох інших галузях, повинні враховуватись нормативи освітлення, температура, відносна вологість і сили вібрації. Але при роботі у приміщенні з комп'ютером найважливішим є дотримання правил пожежної безпеки, це вогнестійкість приміщення, також рівень звукового шуму, характеристики електромагнітних, ультрафіолетових та інфрачервоних полів.

Для аналізу охорони праці у дипломному проєкті досліджується безпека праці розробника веб-сторінок у офісному приміщені.

3.1 Аналіз та безпека умов праці працівника на робочому місці

Під час будь-якого виду роботи за комп'ютером, працівник наражає себе на небезпечні фактори виробничого середовища, а саме: фізичні та психофізіологічні небезпечні й шкідливі виробничі фактори. Це підвищена температура повітря робочої зони, підвищений рівень шуму, знижена вологість повітря. У нервово-психічних перевантаженнях програміст зазнає перенапругу аналізаторів та монотонність праці, інколи, ще й розмовну перенапругу, коли розробнику потрібно складати технічне завдання разом з клієнтом.

Це виробничі фактори, які виникають при роботі у приміщеннях з комп'ютерами,

Окрім цього, комп'ютер випромінює електростатичні та електромагнітні поля у діапазоні від 5 Гц до 2 кГц та від 2 до 400 кГц, тож робота за комп'ютером

включає ще підвищений рівень електромагнітний випромінювання та підвищений рівень статичної електрики.

У офісних приміщеннях не завжди є достатня кількість природного освітлення у такому разі присутня велика кількість штучного освітлення, яке у свою чергу не завжди правильно налаштоване, з цього виникає, що світло може бути недостатньо яскравим або дуже яскравим.

3.2 Розробка заходів з охорони праці

Виробниче освітлення

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, у разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення, додатково встановлюються світильники місцевого освітлення).

Зазначення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк , а освітленість екрана має не перевищувати 300лк

Мікроклімат

При роботі у приміщеннях з великою кількістю комп'ютерів, приміщення класифікуються як приміщення з підвищеною небезпекою електротравм, температура повітря влітку може становити більше 35 С, що дуже погано впливає на здоров'я людини, тож у таких приміщеннях повітря повинне охолоджуватись та понижена вологість повітря повинна регулюватись спеціальним обладнанням.

Відповідно до норм ДСН 3.3.6.042-99 температура повітря в офісі повинна становити 22-25 С, вологість повітря 40-60%, швидкість руху повітря не більше 0,1 м/с. Якщо ці норми перевищені, робочій день працівника повинен бути скорочений на 10%.

3.3 Організація робочого місця користувача ПК

Конструкція робочого місця користувача ПК й взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно росту працюючих для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в у припустимі рамки від 60 до 90 см. Частота мерехтіння екрана fмер=100 Гц, що відповідає умові fмер>70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехтливість екрана від вікон і приладів штучного освітлення.

Для офісів та приміщень, обладнаних персональними комп'ютерами або технікою для бізнесу допустимий рівень шуму цілодобово - 50 дБА, а максимальний- 65 дБА

Згідно темі дипломного проекту робоче місце програміста укомплектовано пристроями з електромагнітним випромінюванням.

3.4 Пожежна безпека

Забезпечення пожежної безпеки на об’єкті праці є важливою частиною роботи по створенню безпечних та здорових умов праці.

Прохід до аварійних виходів повинен бути вільний, шириною не менше 1 метру, у разі великої кількості горючих відходів потрібно використовувати відведені сміттєзбірники. Електроприлади повинні використовуватися тільки для їхнього прямого призначення, а у разі пошкодження приладів, слід вимкнути їх живлення та привести до пожежобезпечного стану.

Первинні засоби пожежогасіння застосовуються для боротьби з пожежами на початковій стадії. До них належать: пожежні кран-комплекти, вогнегасники, пожежний інвентар (резервуари з водою, ящики з піском, пожежні відра, лопати), а також різний переносний пожежний інструмент (кирки, сокири, багри, ломи ін.).

Для гасіння пожеж промисловість випускає різні вогнегасники. Найбільшого поширення набули водопінні, водяні, газові (вуглекислотні) і порошкові. За ефективністю пожежогасіння гасіння, економічністю та іншими показниками більш перспективними вважаються порошкові вогнегасники.

Первинні засоби пожежогасіння розміщують на пожежних щитах, які встановлюють на виробничій території з розрахунку один щит на 5000 м2. Вони фарбуються у червоний колір.

Згідно Правил, на кожному поверсі будинку адміністративного призначення повинно знаходитися не менше двох вогнегасників з масою заряду вогнегасної речовини 5 кг і більше. Експлуатація вогнегасників без призначення відповідального за організацію цієї роботи не допускається. Забороняється палити на підприємстві, крім спеціально виведених для цього місцях, забороняється зберігати легкозаймисті матеріали, такі як папір ближче ніж 1 метр від електрощитів, 0,15 м від приладів центрального водяного опалення та 0,6 м від сповіщувачів автоматичної пожежної сигналізації, також документація повинна зберігатися у спеціально відведених для цього шафах.

Для запобігання розповсюдження пожежі встановлюють протипожежні системи, які складаються з датчиків, звукових сповіщувачів, аварійних кнопок, приймально-контрольної панелі, яка виступає як аналізатор інформації, яку отримали датчики і відправляє ці данні на пульт пожежної охорони. Протипожежна сигналізація призначення для виявлення пожежі на початковому етапі.

Підприємство крім установки пожежної сигналізації на своєму об'єкті, має укласти договір на обслуговування даної системи з фірмою, що має на це ліцензію.

В обслуговування входить проведення встановлених нормами регламентних робіт, а так само усунення несправностей в роботі системи. Періодичність перевірки узгоджується з замовником, але повинна бути не рідше ніж один раз на місяць.

У разі, якщо пожежі не вдалось уникнути, необхідно:

1. терміново повідомити пожежну охорону по телефону 101, вказати при цьому адресу, кількість поверхів, місце виникнення пожежі, наявність людей, своє прізвище;
2. організувати евакуацію людей та матеріальних цінностей;
3. повідомити про виникнення пожежі адміністрацію та чергового (за його наявності);
4. вимкнути, у разі необхідності, струмоприймачі та вентиляцію;
5. розпочати гасіння пожежі наявними первинними засобами пожежогасіння;
6. організувати зустріч підрозділів пожежної охорони й надати їм консультаційну та іншу допомогу в процесі гасіння пожежі.

ВИСНОВКИ

У рамках даного дипл омного проекту було розроблено сервер для цифрових помічників, який базується на технологіях штучного інтелекту. Розробка такого сервера дозволяє забезпечити високу ефективність обробки даних і здійснення інтерактивного спілкування з користувачами. Важливими аспектами реалізації проекту стали:

Було створено масштабовану і гнучку архітектуру сервера, що дозволяє інтегрувати різні модулі штучного інтелекту, зокрема для обробки природної мови та машинного навчання. Архітектура підтримує високу доступність і надійність системи.

Сервер використовує передові алгоритми штучного інтелекту для аналізу вхідних даних від користувачів та забезпечення відповідей, які є релевантними та контекстуально збагаченими. Це забезпечує високу якість взаємодії цифрових помічників з кінцевими користувачами.

У проекті були впроваджені сучасні засоби захисту даних, що гарантує безпеку персональної інформації користувачів. Система реалізована з урахуванням найкращих практик і стандартів у сфері інформаційної безпеки.

Сервер був ретельно протестований на стабільність і здатність витримувати велике навантаження. Тестування показало, що сервер здатний ефективно обробляти запити в реальному часі з мінімальною затримкою.

Розроблена система має високий потенціал масштабування, що дозволяє збільшувати обсяги обробки даних і кількість одночасних користувачів без значної втрати продуктивності.

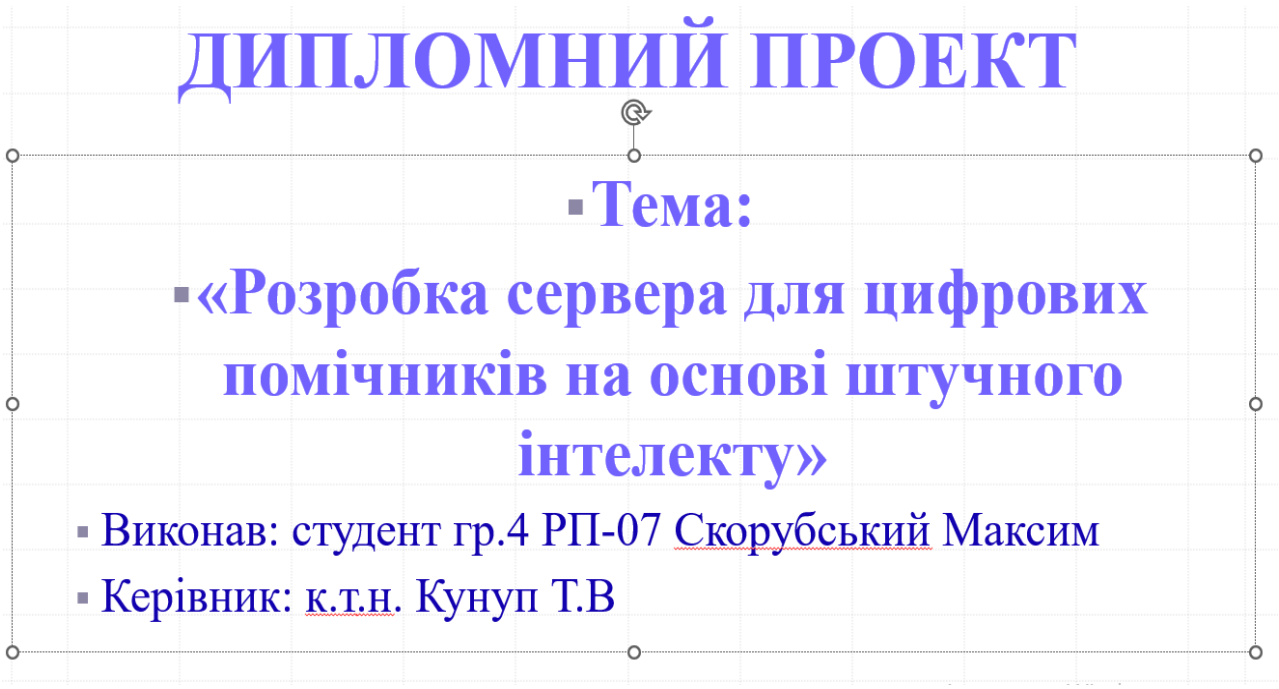
Реалізація даного проекту показала, що створення сервера для цифрових помічників на основі штучного інтелекту є перспективним напрямком у розвитку інформаційних технологій. Створена платформа може бути використана в різних галузях від бізнесу до освіти і значно підвищує доступність і якість цифрових сервісів.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

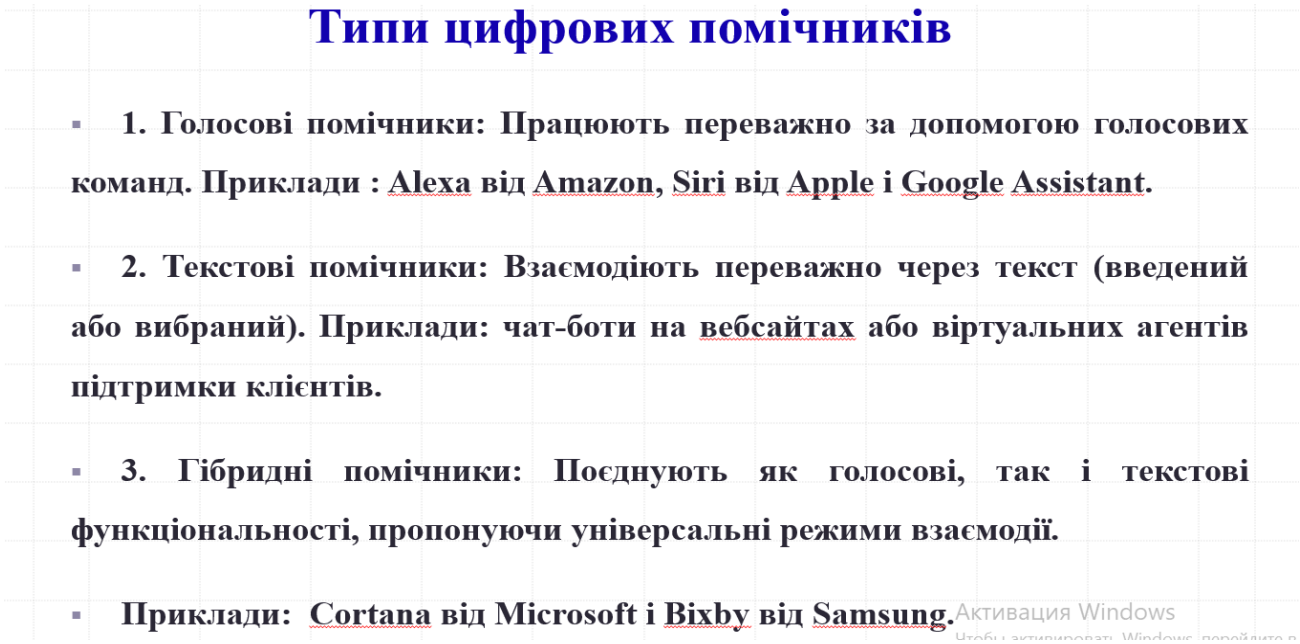
1. URL: https://upsales.com.ua/posts/tipy-i-vidy-sajjtov
2. URL: https://mavr.ua/ua/kakoy-konstruktor-saytov-vybrat/
3. Закон України про охорону праці [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text.
4. Охорона праці і пожежна безпека [Електронний ресурс] URL: https://oppb.com.ua/articles/yaki-vstanovleno-normi-ploshchi-pri-roboti-z-kompyuternoyu-tehnikoyu-yaki-shkidlivi-ta
5. Буділов В. Основи програмуваня для Інтернета.: «БХВ-П», 2023.- 733с.
6. Глушков С. В. Програмування Web-сторінок / С. В. Глушков, І. А. Жакін, Т. С. Хачиров. – Харків: Фоліо, 2005. – 390 с.
7. Проценко О.Б. Web-програмування та web-дизайн. Технологія XML: Навчальний посібник. – Суми: Видавництво СумДУ, 2009. – 166 с.
6. Пасічник О.Г. Основи веб-дизайну. - К.:BHV. - 2009. – 354 с.
7. Холл М., Браун Л. Програмуваня для Web.- М.: Видавничий дім «Вільямс», 2002.- 1259 с.
9. Томсон Л., Веллінг Л. Разрабка Web–приложений на PHP и MySQL.-К.: DiaSoft, 2003.- 650 с.
10. Скляр Д. PHP.C\Збірник рецептів.-М.: Символ-Плюс, 2003.- 643 с.
11. Інформаційні системи і технології: Навчальний посібник/ В.С. Пономаренко Р.К., Бутова І.В. Журавльова, Г.Н. Назарова та ін.; за ред. В.С. Пономаренка. - Київ: Академія, 202. - 542с.
12. Книга про HTML http://htmlbook./ 3.Курс лекцій по HTML для початківців. http://distancelearning.nuft.edu.ua/freecourses/html/index.html

ДОДАТОК А

СЛАЙДИ МУЛЬТИМЕДІЙНОЇ ПРЕЗЕНТАЦІЇ

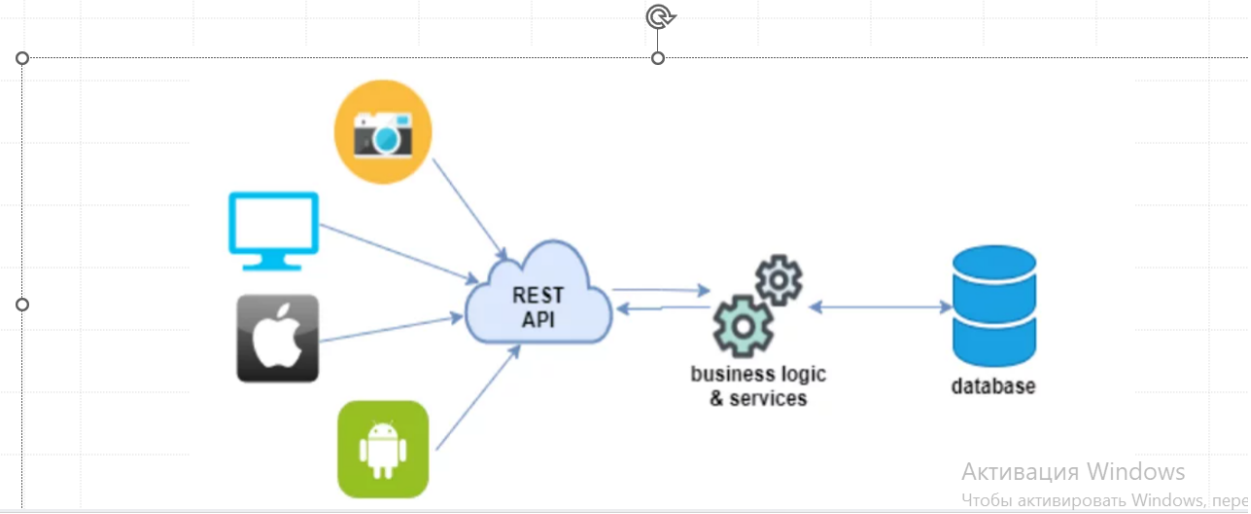


Слайд 1.



Слайд 2

Схема серверної архітектури для систем на ба штучного інтелекту



Слайд 3.

Практична частина розробки

- **dockerfile-compose.yml file:**
- ***services:***
- ***mongodb:***
- ***image: mongo:latest***
- ***environment:***
- ***- MONGODB_DATABASE="mongodb"***
- ***ports:***
- ***- 27017:27017***

СТРУКТУРА ПРОЕКТУ

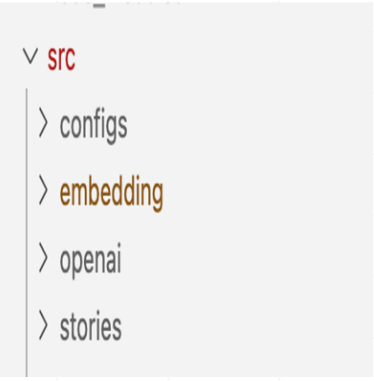
Слайд 4.

Контролери маршрутів

embedding: відповідає за роботу з базою даних Pinecone для ембедингу даних.

openai: для роботи з API OpenAI.

stories: для роботи з базою даних MongoDB, з метою зберігання певних даних.

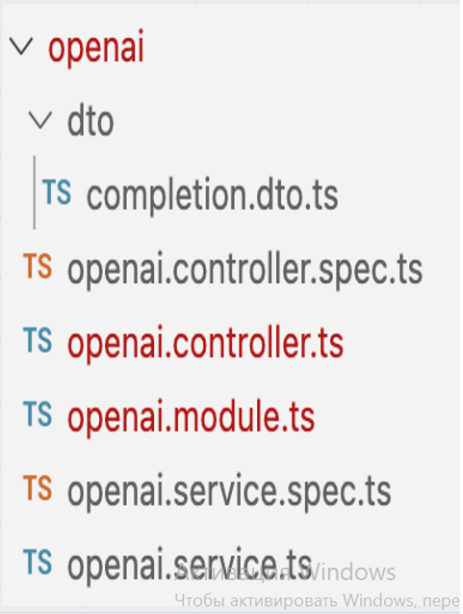


Слайд 5

Скриншот структури папки модуля OpenAI

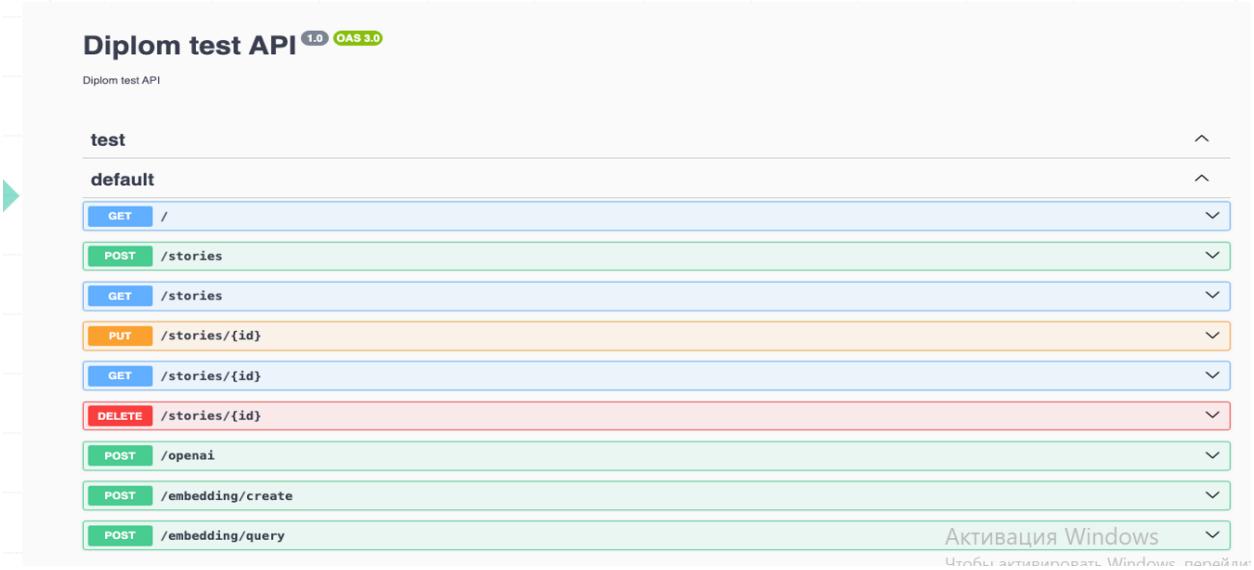
Папка dto (Data Transfer Object)

completion.dto.ts - файл містить клас або інтерфейс, який визначає структуру даних для запитів генерації тексту в OpenAI API.
openai.controller.spec.ts - файл містить юніт-тести для контролера OpenAI.
openai.controller.ts - файл контролера, який обробляє HTTP-запити до OpenAI API.
openai.module.ts - модуль, що інкапсулює всі компоненти OpenAI (сервіси, контролери) і можливі імпорти з інших модулів.
openai.service.spec.ts - файл з тестами для сервісу OpenAI.
openai.service.ts – сервіс, що містить бізнес-



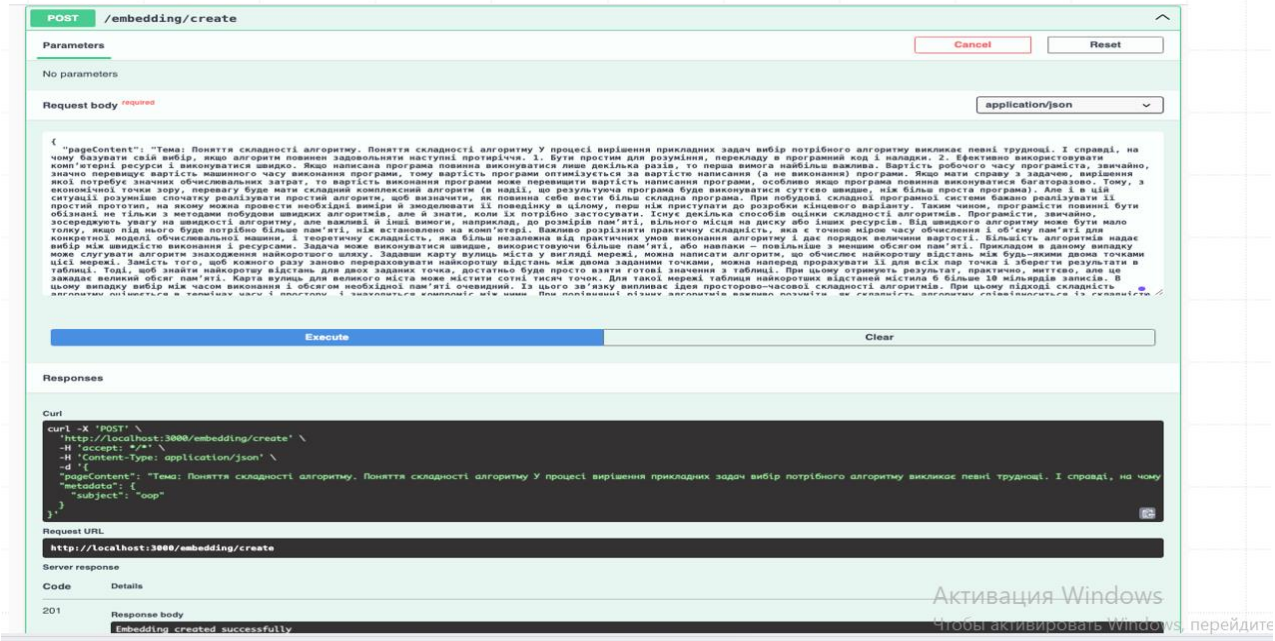
Слайд 6

Робота програми(скриншот сторінки)



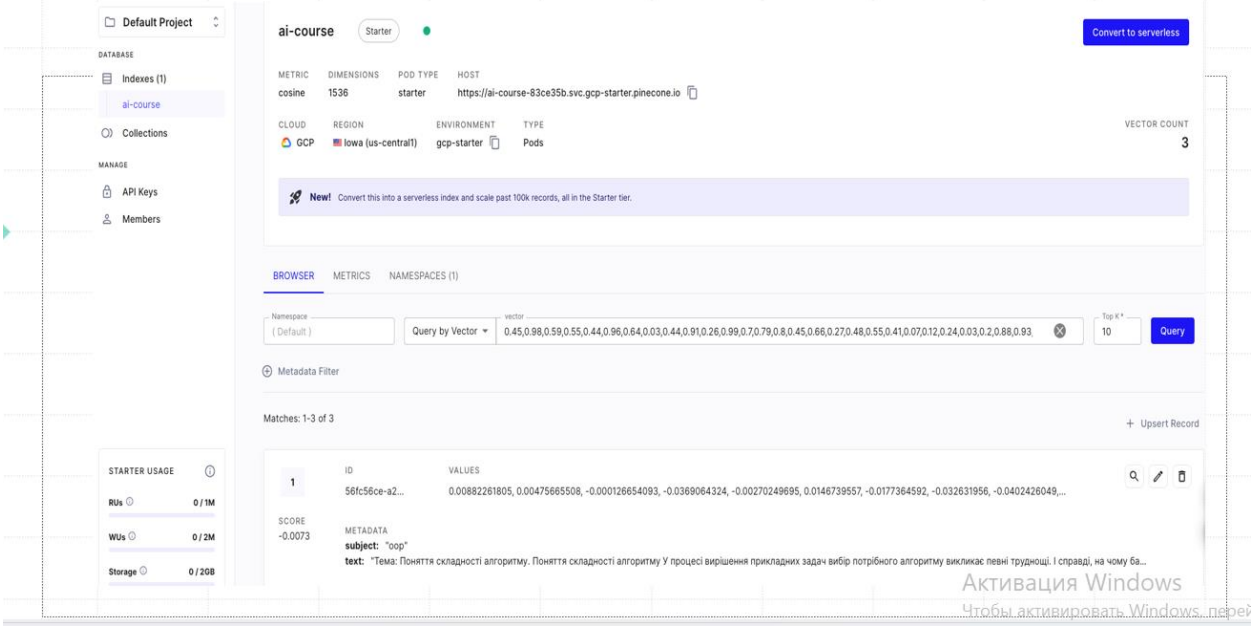
Слайд 7

Скриншот завантаження лекцій через /embedding/create endpoint



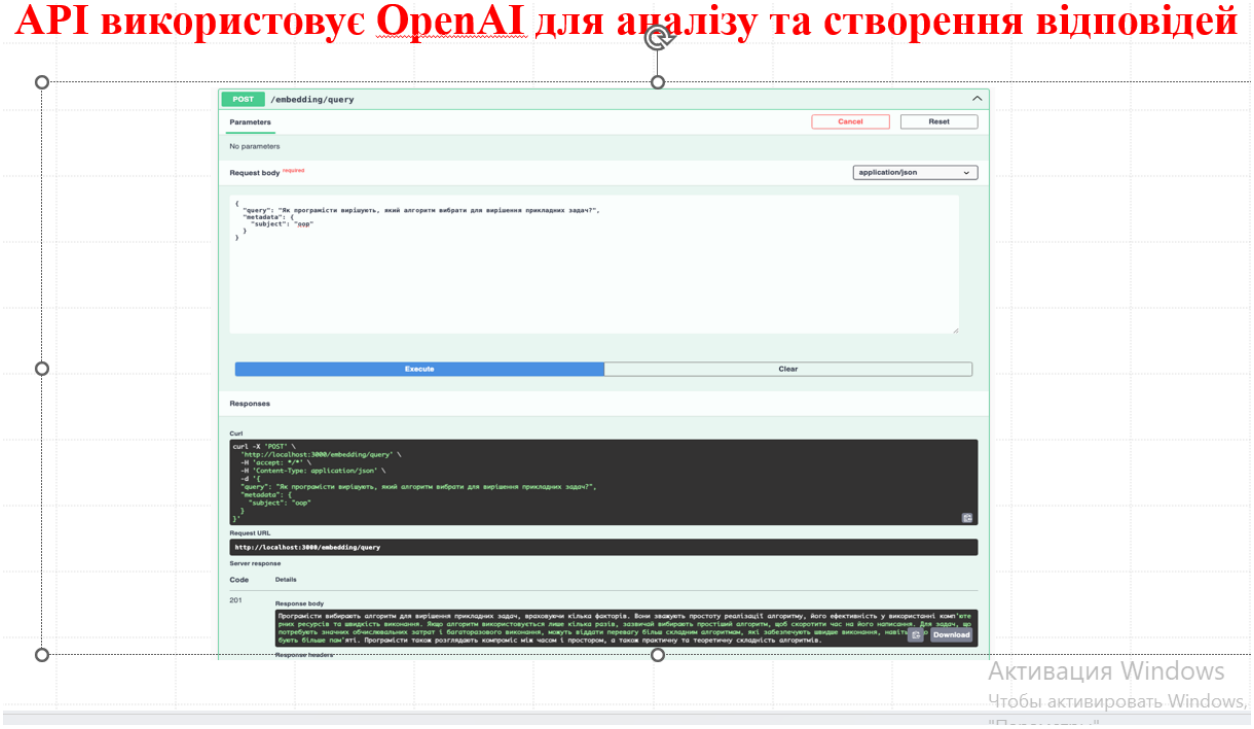
Слайд 8

Скриншот збереженої теми «Поняття складності алгоритму» в векторній БД.



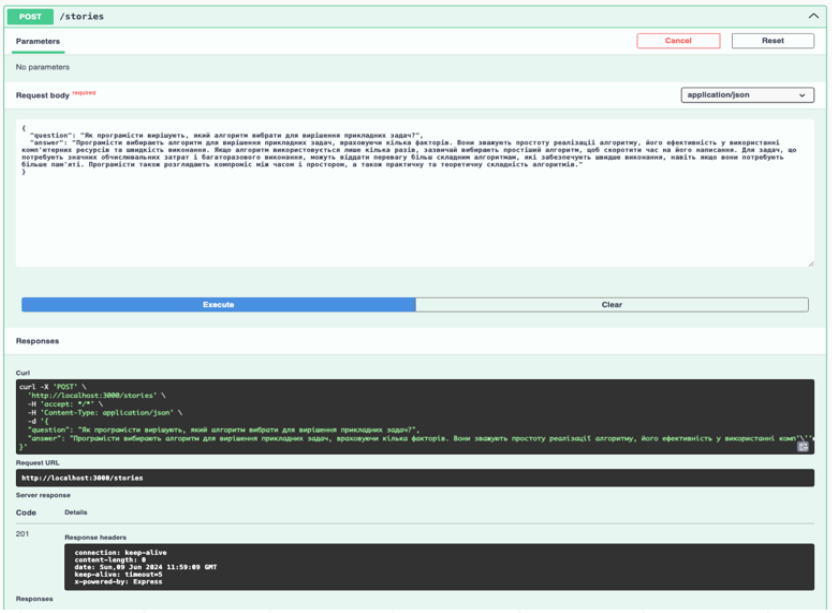
Слайд 9

API використовує OpenAI для аналізу та створення відповідей



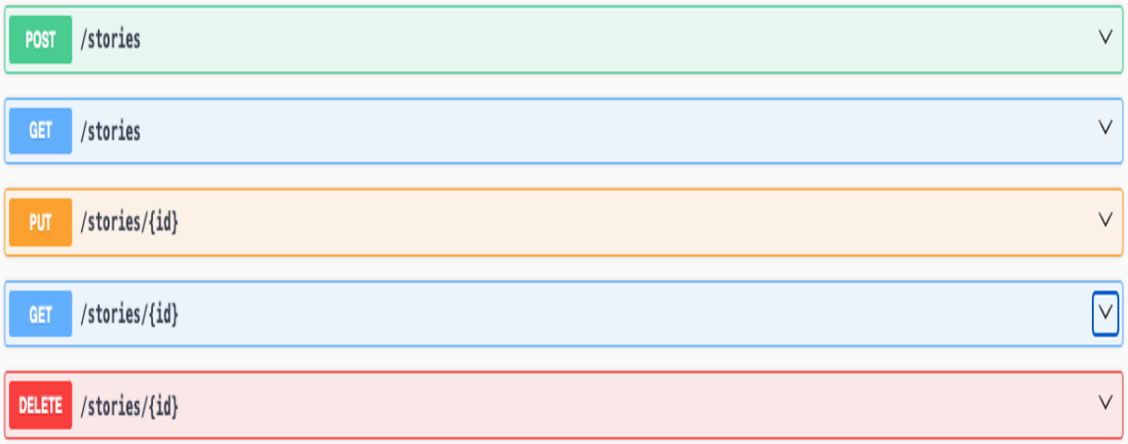
Слайд 10

Скриншот зберігання питання та відповіді в MongoDB



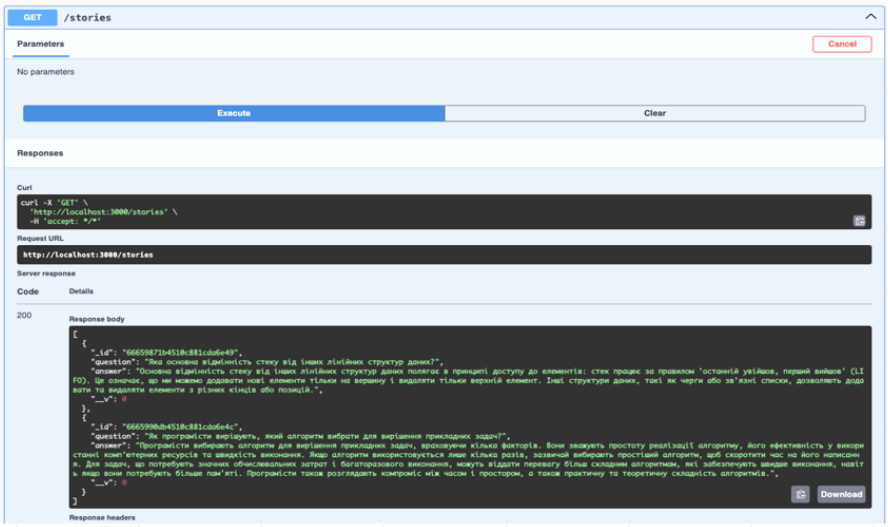
Слайд 11

. Скриншот CRUD операції над питаннями та відповідями в базі даних



Слайд 12

Скриншот усіх збережених питанн і відповідей в бд.



Слайд 13

•Дякую за увагу!

Слайд 14

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стремецькому-(Скорубському) Максиму Віталійовичу

(прізвище, ім'я та по батькові)

Спеціальність: 121 " Інженерія програмного забезпечення"

Освітня програма «Розробка програмного забезпечення»

Тема дипломного проекту: Розробка сервера для цифрових додатків на основі штучного інтелекта

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 63 сторінки. У пояснювальній записці виконано опис штучного інтелекта та виконана розробка сервера для цифрових додатків на основі штучного інтелекта, а також його програмне забезпечення. Графічна частина складається з 14 слайдів мультимедійної презентації, які також містять креслення, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра. розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Стремецький-(Скорубський) М.В. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника

в) теоретична підготовка випускника (випускниці): Здобувач освіти Стремецький-(Скорубський) М.В. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Стремецький-(Скорубський) М.В. мав змогу самостійно приймати окремі рішення з розробки сервера для цифрових помічників на основі штучного інтелекта та показав вміння організовано працювати над поставленим завданням, та розробляти програмні коди за допомогою сучасних комп'ютерних програмних засобів та мов програмування.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Кунуп Тетяна

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії,

Підпис

«10» червня 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стремецького (Скорубського) Максима Віталійовича

(прізвище, ім`я та по батькові)

Спеціальність **121 Інженерія програмного забезпечення**

Освітня програма **« Розробка програмного забезпечення»**

Керівник дипломного проекту (роботи) к.т.н Кунуп Т.В.

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) **Розробка сервера для цифрових помічників на основі штучного інтелекту**

Обсяг розрахунково-пояснювальної записки 60 сторінок

Обсяг графічної (презентаційної) частини 14 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію робота відповідає затверджений темі та виконаний відповідно технічному завданню. Дипломний проект є актуальним з погляду останніх рекомендацій розробки програмного забезпечення для створення цифрових помічників на базі штучного інтелекту.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Пояснювальна записка складається з технологічної частини, розробки структури програми та засобів програмування програми для створення створення цифрових помічників на базі штучного інтелекту, економічної частини, розділу охорони праці та додатку. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічна частина проекту містить розрахунок затрат на виконання програми

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи)

Графічна частина складається з 12 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структуру проекта , скріншоти роботи програми, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки висока, розробку виконано у повному обсязі

г) перелік позитивних якостей дипломного проекту (роботи)_____

У роботі досить обґрунтовано та розроблено програму створення повідомлень на базі штучного інтелекту.

д) основні недоліки дипломного проекту (роботи) _____

1.У роботі відсутні схеми та діаграми, що ілюструють етапи розробки серверу.

2. Застосування штучного інтелекту цифрових помічників недостатньо обґрунтовувано.

3. Робота містить помилки оформлення, помилки у тексті, оформлена не дуже неохайно.

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента к.т.н. Кірєєв Ігор Анатолійович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, доцент каф. інформаційної безпеки та передачі даних

Підпис: _____

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

«14» 06 2024

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016339303

Дата перевірки:
09.06.2024 20:57:02 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2024 22:21:42 EEST

ID користувача:
100011688

Назва документа: 4РП-07 Скорубський М

Кількість сторінок: 54 Кількість слів: 8434 Кількість символів: 64753 Розмір файлу: 3.08 MB ID файлу: 1016140388

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

20.5%
Схожість

Найбільша схожість: 11.2% з Інтернет-джерелом (https://card-file.ontu.edu.ua/server/api/core/bitstreams/bfd0b042-5cf...

20.5% Джерела з Інтернету | 441 ..... Сторінка 56

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 35

Підозріле форматування | 11 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Скорубський-Стремецький Максим Віталійович
здобувач освіти гр.4РП-07, та

Кунуп Тетяна Василівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи фахового молодшого бакалавра на тему:

«Розробка сервера для цифрових помічників на основі штучного інтелекту» (автор роботи – Скорубський-Стремецький М.В., керівник роботи – Кунуп Т.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Скорубський-Стремецький М.В./

Керівник _____ / Кунуп Т.В./

«10» 06 2024 р.