

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка та Web-дизайн»

Група: 4КГ-08

Дипломний проект

здобувача освіти денної форми навчання

КГ.08.19.000.ДП

***РОМАНЕНКА
МИКИТИ
ОЛЕКСАНДРОВИЧА***

м. Одеса

2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка та Web-дизайн»

Група: 4КГ-08

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Створення web-застосунку цифрового помічника з використанням Open AI

Проектний матеріал складається з пояснювальної записки на 76 сторінках та графічного (перзентаційного) матеріалу на 13 аркушах (слайдах).

Дипломник _____ (Романенко М.О.)

Керівник _____ (Кунуп Т. В)

Консультанти:

з економічного розділу _____ (Канський М. Ю.)

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 _____ (Краснокутська К. Г.)

Захист «23» червня 2025 р.

Протокол ЕК № 3

Оцінка ЕК 4(добре) / 802

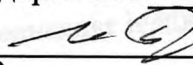
Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення	<u>Ком'ютерних систем</u>	Комісія	<u>КТ та ПІ</u>
Спеціальність	<u>123 «Комп'ютерна інженерія»</u>		
Освітня програма	<u>«Комп'ютерна графіка та Web-дизайн»</u>		

ЗАТВЕРДЖУЮ

Заст. дир. з НВР Беркань І. В.


« 19 » 05 2025 року

ЗАВДАННЯ

на дипломний проєкт (роботу)

Романенко Микити Олександровича

1. Тема проєкту (роботи) Створення web-застосунку цифрового помічника з використанням Open AI

Затверджена наказом по коледжу від « 14 » 11 2025 р., наказ № 246

2 Термін здачі закінченого проєкту (роботи) _____

3. Вихідні дані до проєкту (роботи)

Створення web-застосунку 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 – 13 слайдів Титульний слайд ДП; Схема серверної архітектури

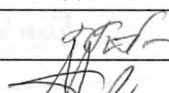
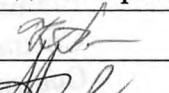
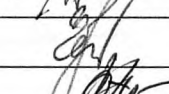
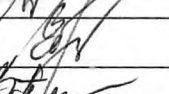
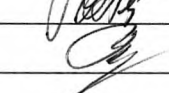
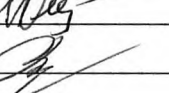
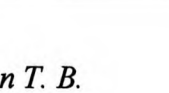
для систем на базі штучного інтелекту; Практична частина розробки; Структура

проекта; Скриншот структури папки модуля OpenAI; Контролери маршрутів;

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

Результати тестування; Висновки.

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	Канський М. Ю.		
Розділ охорони праці	Чорновол Н. І.		
Нормоконтроль	Петрашова В. І.		
Старший консультант	Кривченко Ю. В.		

7. Дата видачі завдання

29.04.25

Керівник

Кунуп Т. В.

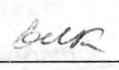
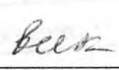
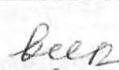
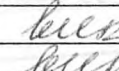
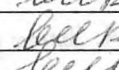
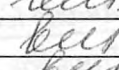
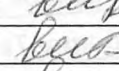
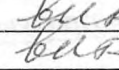
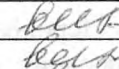
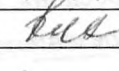
(підпис)

Завдання прийняв до виконання

Романенко М.О.

(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 проєктування	29.04.25	
2	Огляд основних компонентів серверної архітектури	01.05.25	
3	Обґрунтування вибору розробки серверних рішень	03.05.25	
4	Проектування сервісу	05.05.25	
5	Налаштування структури проєкту	07.05.25	
6	Інтеграція бази даних	09.05.25	
7	Тестування створеного програмного продукту	15.05.25	
8	Економічний розрахунок	17.05.25	
9	Опис охорони праці та техніки безпеки	19.05.25	
10	Аналіз результатів проєктування.	21.05.25	
11	Оформлення пояснювальної записки	23.05.25	
12	Оформлення графічної (презентаційної)	30.05.25	
13	частини	05.06.25	
14	Підготовка доповіді для захисту	15.06.25	
15	Малий захист дипломного проєкту	16.06.25	

Дипломник

(підпис)

Керівник

(підпис)

[illegible]

ЗМІСТ

Вступ.....	7
1 Основний розділ.....	8
1.1 Аналіз можливостей штучного інтелекту та його застосування у цифрових помічниках.....	8
1.1.1 Переваги використання ШІ в цифрових помічниках.....	9
1.1.2 Сучасні тренди та перспективи розвитку.....	9
1.1.3 Розширені можливості цифрових помічників та роль ШІ.....	10
1.2 Архітектура серверів для систем на основі штучного інтелекту.....	11
1.2.1 Огляд основних компонентів серверної архітектури.....	11
1.2.2 Використання Node.js та NestJS для розробки серверних рішень.....	11
1.2.3 MongoDB і Pinecone у зберіганні та обробці даних.....	11
1.2.4 Swagger для документації API.....	12
1.2.5 Можливості OpenAI API.....	13
1.3 Схема серверної архітектури для систем на базі ШІ.....	13
1.4 Впровадження штучного інтелекту через API.....	15
1.4.1 Використання OpenAI для інтеграції LLM моделі.....	16
1.5 Реалізація серверної частини розробки сервера.....	17
1.5.1 Планування проекту.....	17
1.5.2 Апаратні вимоги.....	18
1.5.3 Програмні вимоги.....	18
1.5.4 Вимоги до безпеки.....	18
1.5.5 Вибір технологічного стеку та обґрунтування вибору.....	19
1.5.6 Розробка сервера та інтеграція.....	21
1.5.7 Реалізація сервера.....	21
1.5.8 Ініціалізація NestJs.....	22
1.5.9 Налаштування структури проекту.....	22
1.5.10 Створення ендпоїнтів.....	26
1.5.11 Інтеграція бази даних MongoDB та Pinecone.....	26

1.6 Векторна база даних Pinecone та її роль у сегменту пошуку.....	27
1.7 Методи сервіса EmbeddingService.....	31
1.8 Підключення та використання OpenAI API.....	40
1.8.1 Налаштування доступу до OpenAI API.....	40
1.8.2 Опис API за допомогою Swagger.....	44
1.9. Робота програми.....	46
2 Економічна розділ.....	54
2.1.1 Резюме.....	54
2.2 Вивчення трудомісткості розробки програмного забезпечення.....	54
2.3 Розрахунок ціни програмного продукту.....	57
3 Розділ охорони праці та техніки безпеки.....	58
3.1 Аналіз та безпека умов праці працівника на робочому місці.....	59
3.2 Розробка заходів з охорони праці.....	60
3.3 Організація робочого місця користувача ПК.....	61
3.4 Пожежна безпека.....	62
Висновки.....	64
Перелік використаних інформаційних джерел.....	65
Додаток А. Програмний код «Опис ендпоїнтів».....	66
Додаток Б. Слайди мультимедійної презентації.....	67

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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		7

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз можливостей штучного інтелекту у сфері цифрових помічників

Визначення та типи цифрових помічників:

Цифрові помічники - це програмні інструменти, здатні виконувати завдання або надавати послуги користувачам у відповідь на їхні команди або запити. Зазвичай взаємодія з ними здійснюється у формі, наближеній до природного мовлення, що робить їх зручними та інтуїтивно зрозумілими. Типи цифрових помічників:

1. Голосові помічники - функціонують через голосові команди. Приклади: Alexa (Amazon), Siri (Apple), Google Assistant.
2. Текстові помічники - працюють із текстовим введенням. Їх можна знайти у вигляді чат-ботів на сайтах або у службах підтримки.
3. Гібридні помічники - поєднують голосову та текстову взаємодію. Приклади: Cortana (Microsoft), Bixby (Samsung). Приклади включають Cortana від Microsoft і Bixby від Samsung.

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.

Цифрові помічники підтримують широкий спектр функцій: від нагадувань і пошуку інформації до управління пристроями та автоматизації рутинних завдань.

З розвитком штучного інтелекту цифрові помічники стають дедалі розумнішими. Вони здатні розуміти контекст, аналізувати історію запитів, персоналізувати відповіді, а також інтегруватися з іншими сервісами, як-от календарі, пошта, CRM-системи тощо. Застосування цифрових помічників активно розширюється в бізнесі, освіті, охороні здоров'я та повсякденному житті.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		8

1.1.1 Переваги використання ІІІ в цифрових помічниках

Покращене розуміння користувача: завдяки технологіям обробки природної мови цифрові помічники здатні краще розуміти запити та надавати відповіді, подібні до людських.

Індивідуалізований досвід: моделі машинного навчання аналізують поведінку користувача, щоб надавати персоналізовані відповіді та рекомендації.

Висока точність та швидкість: ІІІ дозволяє автоматизувати рутинні процеси, як-от встановлення нагадувань, пошук інформації чи керування пристроями.

Адаптивність: з кожною новою взаємодією цифровий помічник вдосконалює свою здатність передбачати потреби користувача.

1.1.2 Сучасні тренди та перспективи розвитку

Інтеграція з IoT: цифрові помічники дедалі частіше взаємодіють із «розумними» пристроями - від побутової техніки до автомобілів.

Глибше контекстуальне розуміння: у майбутньому ІІІ зможе краще розпізнавати контекст, включно з історією взаємодій, геолокацією та емоційним станом користувача.

Мультимодальна взаємодія: користувачі зможуть комбінувати голос, текст і візуальний інтерфейс, легко перемикаючись між ними на різних пристроях. Підвищена конфіденційність: актуальними стають безпечне зберігання даних, шифрування та контроль над приватністю.

Галузева спеціалізація: з'являються цифрові помічники для конкретних сфер - медицини, освіти, фінансів.

Етичні виклики: важливо забезпечити рівний доступ до технологій, уникати упередженості в алгоритмах та враховувати соціальні наслідки.

Постійне навчання та адаптація: Цифрові помічники на базі ІІІ 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:

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		9

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 поміщикам 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.

1.1.3 Розширені можливості цифрових помічників та роль ШІ

Багатоканальна взаємодія. У майбутньому цифрові помічники, ймовірно, підтримуватимуть багатоканальну комунікацію, що дозволить користувачам безперешкодно переходити між голосовими командами, текстовими повідомленнями та візуальними елементами інтерфейсу. Наприклад, команда може починатися голосом через смарт-колонку, продовжуватися введенням тексту зі смартфона, а завершуватись жестовим керуванням через сенсорний екран розумного годинника.

Покращення безпеки та захисту даних.

З огляду на зростання обсягу конфіденційної інформації, яку обробляють цифрові помічники, ключовим завданням стане забезпечення високого рівня безпеки та конфіденційності. Подальші розробки зосередяться на впровадженні сучасних технологій шифрування, використанні децентралізованої обробки даних і розширенні можливостей користувача щодо контролю своїх персональних даних.

Галузева адаптація. Цифрові помічники еволюціонують у бік спеціалізації для окремих галузей - охорони здоров'я, фінансів, освіти тощо. Такі вузькопрофільні

					<i>КГ 08. 19 001. 00 ДП ПЗ</i>	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		10

рішення дозволяють ефективніше вирішувати складні професійні завдання, надавати висококваліфіковані поради та підвищувати продуктивність.

Етичні та соціальні виклики. Масове впровадження цифрових помічників порушує низку важливих питань. Серед них - забезпечення справедливості доступу до новітніх технологій, подолання можливих упереджень у роботі алгоритмів, а також оцінка впливу на ринок праці та міжособистісну взаємодію.

В цьому підрозділі було висвітлено ключові аспекти розвитку цифрових помічників, класифікацію їхніх типів, приклади використання та конкретні переваги застосування штучного інтелекту. Зокрема, розглянуто покращення природномовної взаємодії, персоналізацію, зростання ефективності, здатність до самонавчання, а також перспективні напрями, такі як IoT-інтеграція, глибше контекстуальне розуміння, мультимодальні інтерфейси, безпека даних, галузеві рішення та етичні питання.

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

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		11

патернів, як Dependency Injection та Decorators, що сприяє кращій модулярності та зменшенню залежностей між компонентами.

1.2.3 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 – це фреймворк з відкритим кодом, який використовується для створення, опису, документування та візуального тестування RESTful API. У 2016 році Swagger став частиною проекту OpenAPI Specification (OAS), і зараз терміни Swagger та OpenAPI часто вживаються як синоніми, хоча технічно Swagger – це набір інструментів, що реалізують OpenAPI. Swagger дозволяє:

1. Створювати машиночитабельну документацію API у форматі JSON або YAML.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		12

2. Автоматично генерувати інтерактивну документацію, яка дозволяє тестувати API прямо в браузері. Генерувати код клієнтів та серверів на різних мовах програмування на основі опису API. Основні компоненти Swagger:

1. Swagger/OpenAPI Specification (OAS) – формат опису REST API. Він визначає ендпоінти, методи, параметри, типи запитів і відповідей.
2. Swagger UI – веб-інтерфейс для перегляду і тестування API. Працює прямо в браузері.
3. Swagger Editor – онлайн/локальний редактор YAML/JSON для написання документації з автопідказками.
4. Swagger Codegen – інструмент для генерації серверного та клієнтського коду на основі специфікації.
5. SwaggerHub – хмарна платформа для колаборації над API-документацією.

1.2.5 Можливості 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:

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		13

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

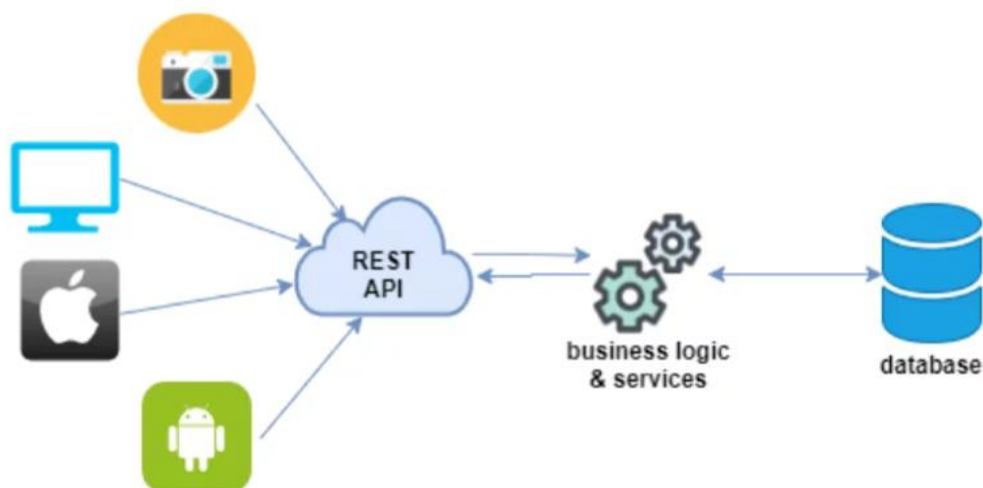


Рисунок 1.1. Модель серверної архітектури для рішень на основі ШІ

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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		15

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

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		16

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 Реалізація серверної частини на практиці

1.5.1 Планування проекту

На початковому етапі проектування серверної інфраструктури визначаються базові технічні вимоги, які формуються на основі аналізу функціональних характеристик системи, бізнес-логіки, а також прогнозованого навантаження з боку кінцевих користувачів. Це включає оцінку обсягу оброблюваних даних, кількості одночасних з'єднань, частоти запитів до серверу, потреб у зберіганні інформації та забезпеченні швидкодії системи.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		17

Особливу увагу приділяють вимогам до масштабованості, відмовостійкості та безпеки, щоб гарантувати стабільну роботу сервера навіть у разі зростання навантаження чи нештатних ситуацій. Також враховуються потреби в інтеграції з зовнішніми сервісами, API або базами даних, що впливає на вибір архітектурного підходу (наприклад, монолітна чи мікросервісна архітектура) та технологічного стека.

1.5.2 Апаратні вимоги

Процесор (CPU): Обчислювальна потужність повинна забезпечувати стабільну обробку запитів у пікові періоди навантаження.

Оперативна пам'ять (RAM): Обсяг пам'яті має бути достатнім для паралельної обробки кількох потоків запитів без затримок.

Сховище (Storage): Повинен бути передбачений резерв дискового простору для зберігання баз даних, логів, кешованих файлів та інших критичних даних.

1.5.3 Програмні вимоги

Операційна система: Linux (Ubuntu 20.04+ або аналогічна), залежно від обраного стеку технологій.

Система керування базами даних (СУБД): PostgreSQL, MySQL або інша відповідно до вимог проєкту.

Додаткові залежності: Node.js, Docker, системи логування (наприклад, ELK Stack), системи контролю доступу, інструменти CI/CD.

1.5.4 Вимоги до безпеки

Шифрування: Всі передані дані мають захищатися протоколом HTTPS (TLS 1.2 або вище).

Автентифікація та авторизація: Використання сучасних стандартів (OAuth 2.0, JWT).

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		18

Контроль доступу: Обмеження прав доступу на основі ролей користувачів (RBAC).

Резервне копіювання: Регулярне автоматизоване створення резервних копій критичних даних.

Моніторинг безпеки: Інтеграція з системами виявлення загроз і сповіщення про підозрілу активність.

1.5.5 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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		19

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.

Pinescone: 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.

Pinescone 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

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		20

розробки API та забезпечення зручності використання та сумісності між різними системами.

1.5.6 Розробка сервера та інтеграція

Сервер буде розроблений за наступними етапами:

1. Налаштування та конфігурація: Встановлення Node.js, NestJS, MongoDB та Pinescone. 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 Pinescone для зберігання та пошуку векторних даних.
4. Тестування: Проведення всебічного тестування для забезпечення функціональності, продуктивності та безпеки. Розгортання сервера в продуктивне середовище.

1.5.7 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`.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		21

1.5.8 Ініціалізація NestJs

NestJS - це прогресивний фреймворк для створення ефективних та масштабованих серверних додатків на платформі Node.js. Він базується на TypeScript і підтримує модульну архітектуру, що робить його зручним для розробки корпоративних рішень. Підготовка середовища:

1. Створіть окрему директорію для нового проєкту та перейдіть до неї за допомогою терміналу:

2. `mkdir my-project && cd my-project`

3. Ініціалізуйте проєкт із використанням Node Package Manager:

4. `npm init -y`

Встановлення Nest CLI та створення проєкту.

Для розгортання стартового шаблону NestJS необхідно встановити глобально офіційний CLI:

`npm install -g @nestjs/cli`

Після встановлення CLI виконайте команду для створення нового проєкту:

`nest new project-name`

У результаті буде згенеровано проєкт із типовою структурою, що включає модулі, контролери, сервіси та конфігураційні файли.

1.5.9 Налаштування структури проєкту

Для забезпечення високої якості розробки доцільно інтегрувати додаткові інструменти, які, хоча й не є обов'язковими, значно покращують процес програмування. Схема налаштування середовища розробки надано на рисунку 1.2.

ESLint - інструмент для статичного аналізу коду, що дозволяє виявляти помилки та недотримання стилістичних правил.

Prettier - автоматичний форматувач коду, який забезпечує послідовне оформлення вихідних файлів. Інтеграція цих засобів дозволяє:

- Автоматично перевіряти стиль та структуру коду;
- Уникати типових помилок на ранніх етапах;
- Забезпечити єдині стандарти форматування в команді.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		22



Рисунок 1.2. Схема налаштування середовища розробки

Інструкції щодо встановлення та конфігурації доступні в офіційній документації: ESLint: <https://eslint.org/docs/latest/user-guide/getting-started>
 - Prettier: <https://prettier.io/docs/en/install.html>

Для зручності розробки рекомендується налаштувати запуск перевірки ESLint та форматування Prettier як частину pre-commit хуків або вбудувати в CI/CD процес.

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 проекту представлена на рисунку 1.3.

На рисунку 1.3 представлений скриншот структури проекту NestJs з різними файлами та папками. Ось детальний опис кожного елементу. Папки:

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		23

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.

```

> dist
> node_modules
> src
> test
⚙ .env
$ .env.example
🔗 .eslintrc.js
📁 .gitignore
📄 .prettierrc
🚢 docker-compose.yml
🐳 Dockerfile
📄 nest-cli.json
📄 package-lock.json
📄 package.json
📖 README.md
$ start.sh
📄 tsconfig.build.json
📄 tsconfig.json

```

Рисунок 1. 3. Скриншот структури проекту

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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		24

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. Завдяки чіткому поділу на окремі компоненти (модулі, сервіси, маршрути, утиліти тощо) стає можливим ефективно управління кодовою базою, спрощується внесення змін і додавання нового функціоналу без порушення вже існуючої логіки.

Крім того, така організація коду сприяє командній роботі: кожен розробник легко орієнтується в структурі проекту, розуміє відповідальність кожного

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		25

модуля та швидко знаходить необхідну частину коду. Це особливо важливо для великих проєктів, де працює кілька учасників команди або відбувається регулярна ротація персоналу. Дотримання структурованого підходу також полегшує автоматизацію тестування, інтеграцію з CI/CD-процесами та підвищує загальну якість розробки.

1.5.10 Створення ендпоїнтів

Ендпоїнти (Endpoints) - це точки доступу до функціональності вебсервісу або API, через які клієнт (наприклад, браузер, мобільний застосунок чи інша система) може надсилати запити до сервера та отримувати відповіді.

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

Було створено три контролери, а саме:

`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.

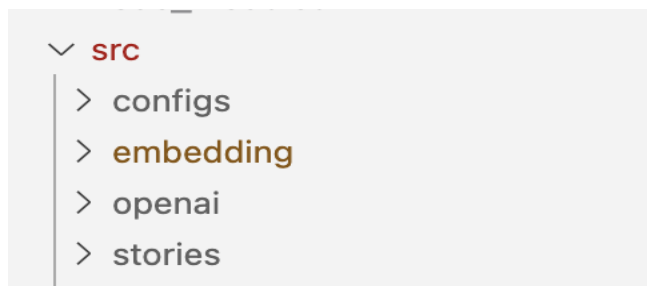


Рисунок 1.4. Скриншот контролерів маршрутів

1.5.11 Інтеграція MongoDB, Pinecone та OpenAI API у проєкт

1. Робота з MongoDB. Для взаємодії з базою даних MongoDB у проєкті було використано бібліотеку Mongoose, яка забезпечує зручний механізм моделювання даних.

2. Встановлення бібліотеки Mongoose. Бібліотеку встановлено через npm: ``npm install mongoose``

3. Підключення до бази даних о MongoDB було реалізовано у конфігураційному файлі, використовуючи стандартні методи Mongoose.

4. Створення схем. Структура даних моделювалась за допомогою схем Mongoose, що дозволяє визначити типи, валідацію та поведінку документів.

5. Інтеграція з Pinecone. Сервіс Pinecone був використаний для роботи з векторними даними - наприклад, збереження та пошуку векторних представлень текстів або об'єктів.

6. Використання офіційної бібліотеки або API було інтеграцію реалізовано через Pinecone SDK або прямі REST-запити.

7. Налаштування з'єднання. Конфігураційні параметри (API-ключ, середовище, індекс тощо) були додані до. env файлу та імпортовані у відповідні модулі для ініціалізації клієнта.

8. Підключення OpenAI API. Отримання доступу. Для роботи з OpenAI необхідно зареєструватися на <https://platform.openai.com> та згенерувати API-ключ. Встановлення бібліотеки: Інсталяція офіційного клієнта: ``npm install openai``

1.6 Векторна база даних Pinecone та її роль у семантичному пошуку

Pinecone - це спеціалізована векторна база даних, призначена для зберігання та обробки векторів, що генеруються моделями глибинного навчання. Ці вектори використовуються для семантичного пошуку та векторних вбудовувань (embeddings). У поєднанні з моделями OpenAI, Pinecone дає змогу здійснювати високоточне порівняння об'єктів на основі їхніх векторних представлень, що значно покращує якість пошуку схожих елементів.

1. Retrieval-Augmented Generation (RAG) у великих мовних моделях (LLM)

Retrieval-Augmented Generation (RAG) - це технологія, яка поєднує механізми пошуку у зовнішніх базах знань (retrieval) з можливостями генерації тексту на основі моделей штучного інтелекту. Такий підхід дає змогу LLM:

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		27

1. Генерувати текст на основі актуальних даних, а не лише знань, отриманих під час навчання.
2. Динамічно включати зовнішню інформацію для формування більш точних і релевантних відповідей.

Переваги RAG:

1. Підвищена точність - доступ до зовнішніх джерел дозволяє формувати відповіді на основі актуальних та перевірених даних.
2. Покращене контекстуальне розуміння - можливість витягувати специфічну інформацію з бази знань збагачує генерацію і забезпечує кращу відповідність змісту запиту.
3. Адаптивність - здатність відповідати на запити, що потребують оновлених або вузькоспеціалізованих знань, яких модель не мала під час навчання.

Причини вибору Pinecone як векторної бази даних.

На ринку доступні різні векторні сховища, однак Pinecone було обрано з огляду на такі ключові переваги:

1. Висока продуктивність і масштабованість - Pinecone оптимізована для обробки великої кількості запитів у реальному часі.
2. Точність відповідей - забезпечує високий рівень відповідності між запитами та знайденими векторами.

Ці характеристики роблять Pinecone ідеальним вибором для підтримки розширених можливостей RAG (Retrieval-Augmented Generation) у великих мовних моделях (LLM), забезпечуючи швидке та ефективне витягування релевантних даних, необхідних для генерації точних і контекстуально значущих відповідей. Pinecone відзначається високою продуктивністю, низькою затримкою при пошуку векторів, масштабованістю та надійністю, що особливо важливо при роботі з великими обсягами векторизованої інформації. Його здатність швидко знаходити найбільш релевантні вектори дозволяє LLM оперативно використовувати зовнішні знання, доповнюючи контекст запитів користувача актуальними фактами, документами чи інструкціями.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		28

Це, в свою чергу, підвищує точність відповідей, знижує ризик галюцинацій (вигаданих відповідей), а також дозволяє моделі краще адаптуватися до специфіки конкретного домену чи задачі.

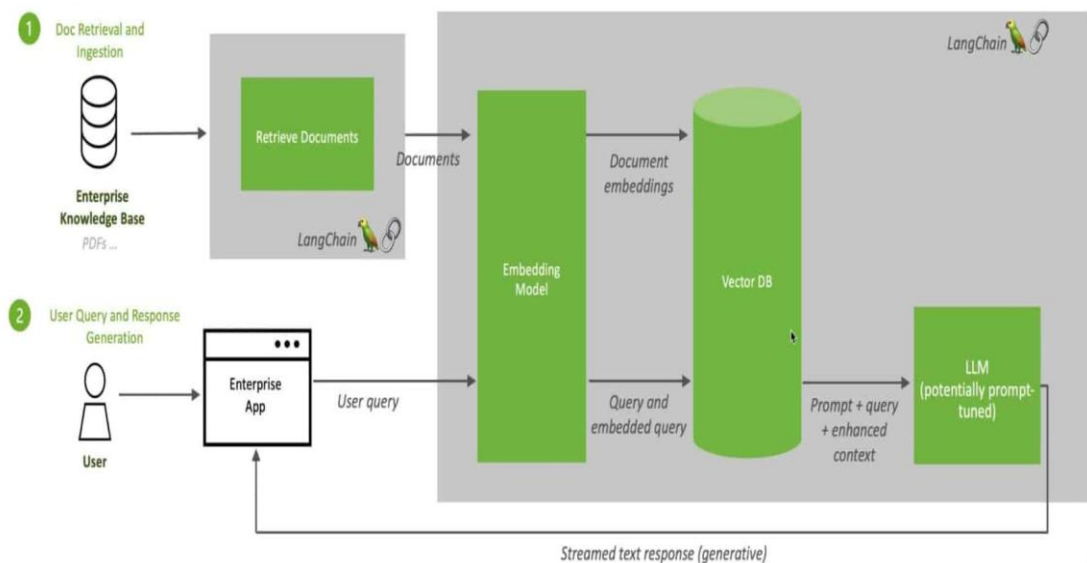


Рисунок 1.5. Схема Retrieval-Augmented Generation

Крок 1: Встановлення необхідних залежностей

Встановили необхідні пакети для роботи з Pinecone та OpenAI.

`npm install pinecone-client openai`

Крок 2: Налаштування Pinecone

Створили модуль embedding для управління підключенням до Pinecone та процесом.

`nest generate module embedding`

`nest generate service embedding`

`nest generate controller embedding`

Цей код визначає модуль EmbeddingModule у проєкті NestJS.

```

embedding > 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 1.6. Скриншот кода модуля EmbeddingModule

Опишемо тепер EmbeddingModule у NestJS.

1. Основні імпорти.

Module - імпортується з пакету @nestjs/common. Використовується для оголошення модуля в NestJS через відповідний декоратор.

ConfigService - надається пакетом @nestjs/config. Дозволяє отримувати доступ до змінних середовища (env) у межах усього застосунку. EmbeddingController та EmbeddingService - імпортуються з локальних файлів. Відповідають за обробку HTTP-запитів і реалізацію бізнес-логіки, пов'язаної з вбудовуваннями (embeddings).

1. Декоратор @Module()

Декоратор @Module() перетворює клас на NestJS-модуль і приймає об'єкт конфігурації з такими ключовими властивостями:

providers: список сервісів, які будуть створені в межах модуля і доступні для ін'єкції через механізм Dependency Injection. У цьому випадку:

1. ConfigService - сервіс для доступу до конфігурацій.

2. EmbeddingService - сервіс, який реалізує основну логіку роботи з embedding-даними. controllers: масив контролерів, які обробляють вхідні HTTP-запити. У даному випадку - EmbeddingController.

Клас EmbeddingModule.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		30

EmbeddingModule - це оголошення модуля, який включає EmbeddingController і EmbeddingService у свою конфігурацію. Цей модуль може бути експортований та імпортований в інші частини NestJS-застосунку, що дозволяє використовувати його функціональність повторно.

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.

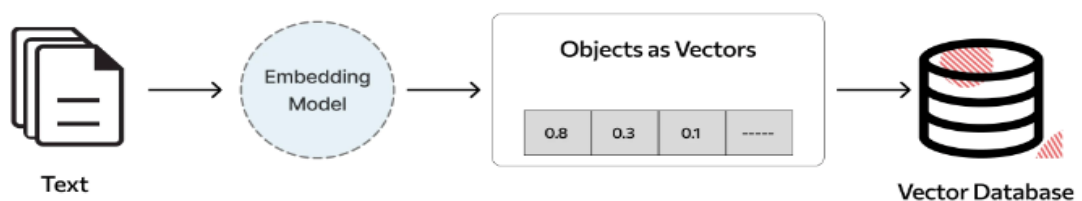


Рисунок 1.7. Схема роботи методу create

```
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 1.8 Скриншот коду методу Create

1. Створення нового вкладення документа

Цей метод відповідає за створення нового документа-вбудовування (embedding) у системі. Його логіка реалізована поетапно наступним чином:

2 Аргументи методу

Метод приймає параметр createEmbeddingDto, який є екземпляром класу CreateEmbeddingDto. Цей клас виконує роль Data Transfer Object (DTO) і містить усі необхідні дані для створення нового документа, зокрема:

pageContent -текстовий вміст сторінки;

metadata - додаткові метадані, пов'язані з документом.

3 Витяг даних із DTO

DTO розпаковується для вилучення вищезгаданих властивостей (pageContent, metadata). Ці дані використовуються для формування нового екземпляра документа.

4 Створення документа

На основі отриманих даних створюється об'єкт Document, який потім додається до масиву docs. Цей масив є колекцією документів, які готуються до збереження.

5 Клас Document

Клас Document є універсальним шаблонним класом, що приймає узагальнений тип Metadata. Він реалізує два інтерфейси - DocumentInput та DocumentInterface.

6.Збереження у векторну базу

Документи з масиву docs передаються у метод PineconeStore.fromDocuments(...), який приймає три параметри:

1. Масив документів docs;
2. Інстанс вбудовувань this.openAIEmbeddings;
3. Об'єкт опцій з параметрами збереження.

Цей метод відповідає за завантаження та збереження документів у векторну базу Pinecone.

Метод query виконує функцію обробки запиту користувача:

1. Приймає вхідні дані (текст запиту);
2. Здійснює семантичний пошук у векторній базі Pinecone;
3. Отримує найбільш релевантні документи;
4. Повертає їх у вигляді відповіді користувачу.

Метод ілюстровано на рисунку 1.8 скріншоті коду -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 1.9. Скріншот 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 1.10. 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.

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.

```

src > stories > TS stories.module.ts > StoriesModule
1  import { Module } from '@nestjs/common';
2  import { MongooseModule } from '@nestjs/mongoose';
3  import { StoriesController } from './stories.controller';
4  import { StoriesService } from './stories.service';
5  import { Story, StorySchema } from './schemas/story.schema';
6
7  @Module({
8    imports: [
9      MongooseModule.forFeature([
10       { name: Story.name, schema: StorySchema }
11     ]),
12    ],
13    controllers: [StoriesController],
14    providers: [StoriesService],
15  })
16  export class StoriesModule {}

```

Рисунок 1.11. Скриншот кода налаштування підключення у файлі.

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.

```

ories > schemas > TS story.schema.ts > Story > data
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type StoryDocument = HydratedDocument<Story>;

@Schema()
export class Story {
  @Prop()
  data: string;

  @Prop()
  name: string;
}

export const StorySchema = SchemaFactory.createForClass(Story);

```

Рисунок 1.12. Скриншот схеми бази даних MongoDB

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		36

Реалізація CRUD-операцій у модулі Stories (NestJS + MongoDB)

1. Create - Створення нового запису. Операція створення нового документа в базі даних MongoDB за допомогою сервісу StoriesService.

Метод сервісу:

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

Пояснення:

1. Метод приймає об'єкт createStoryDto, екземпляр класу CreateStoryDto (DTO);
2. DTO містить структуру даних для створення нового документа;
3. Метод create створює документ, await - очікує завершення операції;
4. Повертається створений об'єкт Story.

2. Read - Читання даних

Операції читання забезпечують доступ до існуючих записів у базі даних.

Метод для отримання всіх записів:

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

Повертає всі документи з колекції stories.

Метод для отримання одного запису за ID: async findOne(id: string):

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

Повертає один запис із бази даних за його унікальним ідентифікатором.

3. Update - Оновлення запису

Оновлення існуючого документа у базі даних за ID.

Метод сервісу:

```
async update(id: string, createStoryDto: CreateStoryDto): Promise<Story> {
```

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
						37
Ізм.	Лист	№ докум.	Підпис	Дата		

```

return await this.storyModel
  .findOneAndUpdate({ _id: id }, createStoryDto, { new: true })
  .exec();
}

```

Пояснення: Оновлює запис за ідентифікатором id;

Використовує дані з createStoryDto;

Повертає оновлений об'єкт Story.

4. Delete - Видалення запису.

Видалення документа з бази даних за ID.

Метод сервісу:

```

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

```

Видаляє запис з колекції stories за його унікальним ідентифікатором.

5. Контролер: реалізація ендпоінтів для CRUD. Програмний код тестування:

```

import { Test, TestingModule } from '@nestjs/testing';
import { StoriesController } from './stories.controller';
import { StoriesService } from './stories.service';
import { CreateStoryDto } from './dto/create-story.dto';
import { Story } from './entities/story.entity';

```

```

describe('StoriesController', () => {
  let controller: StoriesController;
  let service: StoriesService;

```

```

const mockStoriesService = {
  create: jest.fn((dto) => Promise.resolve({ id: '1', ...dto })),
  update: jest.fn((id, dto) => Promise.resolve({ id, ...dto })),
  findAll: jest.fn(() => Promise.resolve([{ id: '1', title: 'Test Story' }])),
  findOne: jest.fn((id) => Promise.resolve({ id, title: 'Single Story' })),

```

```

    delete: jest.fn((id) => Promise.resolve({ deleted: true })),
  };

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      controllers: [StoriesController],
      providers: [
        {
          provide: StoriesService,
          useValue: mockStoriesService,
        },
      ],
    }).compile();

    controller = module.get<StoriesController>(StoriesController);
    service = module.get<StoriesService>(StoriesService);
  });

  it('should be defined', () => {
    expect(controller).toBeDefined();
  });

  it('should create a story', async () => {
    const dto: CreateStoryDto = { title: 'New Story', content: 'Some content' };
    expect(await controller.create(dto)).toEqual({ id: '1', ...dto });
    expect(service.create).toHaveBeenCalledWith(dto);
  });

  it('should return all stories', async () => {
    expect(await controller.findAll()).toEqual([
      { id: '1', title: 'Test Story' }
    ]);
  });

```

```

it('should return one story', async () => {
  expect(await controller.findOne('1')).toEqual({ id: '1', title: 'Single Story' });
  expect(service.findOne).toHaveBeenCalledWith('1');
});

it('should update a story', async () => {
  const dto: CreateStoryDto = { title: 'Updated Title', content: 'Updated content' };
  expect(await controller.update('1', dto)).toEqual({ id: '1', ...dto });
  expect(service.update).toHaveBeenCalledWith('1', dto);
});

it('should delete a story', async () => {
  expect(await controller.delete('1')).toEqual({ deleted: true });
  expect(service.delete).toHaveBeenCalledWith('1');
});

@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 1.13. Скриншот коду Контролери StoriesController

Контролер StoriesController має відповідні HTTP-ендпоїнти, які викликають методи сервісу StoriesService. Ці ендпоїнти реалізують повний набір CRUD-

1.8 Підключення та використання OpenAI API

1.8.1 Налаштування доступу до OpenAI API

Для інтеграції з сервісами OpenAI було виконано наступні кроки:

1. Зареєстровано обліковий запис на офіційному веб-сайті OpenAI (<https://platform.openai.com>).
2. Після реєстрації створено новий проєкт, у межах якого згенеровано API-ключ, що надає доступ до функціоналу OpenAI API.
3. API-ключ було отримано у вкладці "API Keys" в особистому кабінеті користувача. Цей ключ використовується для автентифікації запитів до OpenAI API. Для забезпечення безпеки:
 1. API-ключ та інші чутливі дані не зберігаються у відкритому коді;
 2. Використовую стандартний механізм NestJS для керування конфігураційними змінними;
 3. Ключі зберігаються у файлі .env, який додано до .gitignore для запобігання потраплянню до публічного репозиторію.

```
example
PORT=3000

OPEN_AI_KEY=sk- ...

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

MONGODB_URI=mongodb://localhost:27017/mongodb

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

Рисунок 1.14. Скриншот файлу env.example

Файли .env.example - шаблону змінних середовища.

.env.example є шаблонним файлом, який демонструє структуру конфігураційного файлу .env. Цей файл використовується у програмних проєктах для зберігання змінних середовища - параметрів, що можуть змінюватися залежно від середовища запуску: розробка, тестування, продакшн.

Опис змінних:

PORT=3000, вказує порт, на якому працює застосунок. У цьому випадку - 3000.

OPEN_AI_KEY=sk-...

API-ключ для авторизованого доступу до сервісів OpenAI. Значення "sk-..." потрібно замінити на ваш реальний ключ.

#embedding model

Це коментар, який пояснює, що наступна змінна стосується налаштування моделі вбудовування. modelName=text-embedding-ada-002

Назва моделі OpenAI, яка використовується для генерації embedding-векторів. MONGODB_URI=mongodb://localhost:27017/mongodb

Рядок підключення до локальної бази даних MongoDB на порту 27017 з назвою бази "mongodb". PINECONE_INDEX=ai-course

Назва індексу у базі Pinecone, що використовується для зберігання або пошуку векторних даних. PINECONE_API_KEY=

Поле для введення вашого API-ключа Pinecone.

PINECONE_ENVIRONMENT=gcp-starter

Середовище, у якому запущено Pinecone. Значення "gcp-starter" вказує на початковий кластер у Google Cloud.

Примітка: Файл .env.example слугує шаблоном для створення конфігураційного файлу .env. Після копіювання шаблону необхідно ввести реальні значення для змінних. Файл .env не слід додавати до репозиторію, оскільки він може містити конфіденційну інформацію.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		42

```

  ✓ openai
    ✓ dto
      | TS completion.dto.ts
    TS openai.controller.spec.ts
    TS openai.controller.ts
    TS openai.module.ts
    TS openai.service.spec.ts
    TS openai.service.ts

```

Рисунок 1.15. 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 зручну та зрозумілу документацію API, що допомагає розробникам швидко орієнтуватися у доступних кінцевих точках та тестувати їх без потреби заглиблюватися в код.

Ключові переваги Swagger:

1. Спрощення співпраці - Swagger уніфікує структуру документації, забезпечуючи точне описання методів, параметрів і моделей даних. Це дозволяє ефективніше взаємодіяти між командами.
2. Інтерактивність - за допомогою Swagger UI можна безпосередньо з інтерфейсу браузера виконувати запити до API для тестування та діагностики.
3. Універсальність - підтримка широкого спектра мов програмування та фреймворків робить Swagger гнучким для різних типів проєктів.
4. Автоматизація - Swagger дозволяє генерувати клієнтський код для API, зменшуючи час розробки та помилки при ручній реалізації.
5. Відповідність стандартам - Swagger базується на специфікації OpenAPI, яка є стандартом у галузі для RESTful API.

Для інтеграції Swagger у NestJS-проєкт були встановлені необхідні пакети за допомогою команди:

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

Ця команда додає потрібні залежності, які забезпечують можливість документування та взаємодії з API через Swagger.

Налаштування Swagger виконується у файлі main.ts - головній точці входу застосунку NestJS. У цьому файлі визначаються базові параметри документації: назва API, версія, опис, а також маршрут для доступу до Swagger UI.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		44

Текст із зображення ілюструє TypeScript-код з прикладом такої конфігурації, що включає імпорти, ініціалізацію документації та її підключення до веб-сервера.

```
main.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 1.16. 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).

1. Функція bootstrap

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		45

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

На скриншоті рисунок 1.15. зображено інтерфейс Swagger UI для проєкту під назвою Diplom test API. Документація реалізована відповідно до специфікації OpenAPI 3.0. Swagger забезпечує інтерактивний інтерфейс для тестування та вивчення API.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		46

Розділи API та їх опис

1. GET / - Кореневий маршрут, використовується для перевірки доступності API.
2. POST /stories - Створення нового запису у колекції stories.
3. GET /stories - Отримання списку всіх наявних stories.
4. PUT /stories/{id} - Оновлення наявного запису story за його ID.
5. GET /stories/{id} - Отримання конкретного запису за ID.
6. DELETE /stories/{id} - Видалення запису з бази даних за ID.
7. POST /openai - Запити до OpenAI API, наприклад, для генерації тексту.
8. POST /embedding/create - Створення embedding-даних та збереження в Pinecone.
9. POST /embedding/query - Виконання семантичного пошуку embedding-даних.



Рисунок 1.17. Скріншот сторінки

Приклади використання розробленого API в освітньому середовищі

1. Пошук інформації в наукових статтях

Розроблене API дає змогу студентам швидко знаходити релевантну інформацію у великих сховищах наукових статей, які зберігаються у базах Pinecone та MongoDB. Наприклад: студенти можуть використовувати API для пошуку ключових досліджень на певну тему при підготовці рефератів або курсових робіт.

2. Автоматичне створення конспектів лекцій

Завдяки інтеграції з OpenAI, API може формувати стислий конспект лекційного матеріалу на основі даних, отриманих через пошук у Pinecone.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		47

Це дає змогу студентам отримувати узагальнені та структуровані резюме складного навчального контенту.

3. Підготовка до іспитів

API дозволяє студентам ефективно готуватись до іспитів, забезпечуючи доступ до відповідей на конкретні запитання на основі лекційного матеріалу та наукових джерел.

Крім того, за допомогою CRUD-операцій можна зберігати результати підготовки, аналізувати прогрес і повторно використовувати дані в майбутньому.

Приклад: Використання API для підготовки до іспитів

Студенти можуть використовувати API для оптимізації підготовки до іспитів шляхом завантаження власних конспектів і додаткових матеріалів. Після цього вони можуть виконувати запити до системи для отримання стислих зведень, пошуку ключових понять та генерації відповідей.

Основна функціональність API:

1. Завантаження та збереження матеріалів лекцій і допоміжних документів у векторну базу Pinecone для подальшого семантичного пошуку.

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.

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

POST /embedding/create

Parameters

No parameters

Request body required

application/json

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/embedding/create' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "pageContent": "Тема: Поняття складності алгоритму. Поняття складності алгоритму У процесі вирішення прикладних задач вибір потрібного алгоритму викликає певні труднощі. І справді, на чому базувати свій вибір, якщо алгоритм повинен задовольняти наступні протиріччя. 1. Бити простим для розуміння, перекладу в програмний код і наладки. 2. Ефективно використовувати комп'ютерні ресурси і виконуватися швидко. Якщо написана програма повинна виконуватися лише декілька разів, то перша вимога найбільш важлива. Вартість робочого часу програміста, звичайно, значно перевищує вартість машинного часу виконання програми, тому вартість програми оптимізується за вартістю написання (а не виконання) програми. Якщо мати справу з задачею, вирішення якої потребує значних обчислювальних затрат, то вартість виконання програми може перевищити вартість написання програми, особливо якщо програма повинна виконуватися багаторазово. Тому, з економічної точки зору, перевагу буде мати складний комплексний алгоритм (в надалі, що результуюча програма буде виконуватися суттєво швидше, ніж більш проста програма). Але і в цій ситуації розуміємо спочатку реалізувати простий алгоритм, щоб визначити, як повинна себе нести більш складна програма. При побудові складної програмної системи бажано реалізувати її простий прототип, на якому можна провести необхідні виміри й змодельовати її поведінку в цілому, перш ніж приступити до розробки кінцевого варіанту. Таким чином, програмісти повинні бути обізнані не тільки з методами побудови швидкодії алгоритмів, але й знати, коли їх потрібно застосовувати. Існує декілька способів оцінки складності алгоритмів. Програмісти, звичайно, зосереджують увагу на швидкості алгоритму, але накинні й інші винюги, наприклад, до розмірів пам'яті, відносно місця на диску або інших ресурсів. Від швидкого алгоритму може бути мало толку, якщо від нього буде потрібно більше пам'яті, ніж встановлено на комп'ютері. Виявлює розрізняти практичну складність, яка є точною мірою часу обчислення і об'єму пам'яті для конкретної моделі обчислювальної машини, і теоретичну складність, яка більш незалежна від практичних умов виконання алгоритму і дає порядок величини вартості. Більшість алгоритмів надає вибір між швидкістю виконання і ресурсами. Задача може виконуватися швидше, використовувач більше пам'яті, або навпаки – повільніше з меншим обсягом пам'яті. Прикладом в даному випадку може слугувати алгоритм знаходження найкоротшого шляху. Задавши карту вулиць міста у вигляді мережі, можна написати алгоритм, що обчислить найкоротшу відстань між будь-якими двома точками цієї мережі. Замість того, щоб кожного разу заново перераховувати найкоротшу відстань між двома заданими точками, можна наперед пророзрахувати її для всіх пар точки і зберегти результати в таблиці. Тоді, щоб знати найкоротшу відстань для двох заданих точка, достатньо буде просто взяти готові значення з таблиці. При цьому отримувати результат, практично, миттєво, але це заважає великий обсяг пам'яті. Карта вулиць для великого міста може містити сотні тисяч точок. Для такої мережі таблиця найкоротших відстаней міститиме 6 більше 10 мільярдів записів. В цьому випадку вибір між часом виконання і обсягом необхідної пам'яті очевидний. Із цього зв'язку випливає ідея просторово-часової складності алгоритмів. При цьому відносна складність алгоритму дорівнює в темпів часу і простору. І зникає поняття складності між ними. При написанні різних алгоритмів завжди повинні як розглядати алгоритм з точки зору його складності.
```

Response URL

http://localhost:3000/embedding/create

Server response

Code

Details

201

Response body

Embedding created successfully

Bomarcas header

Download

Рисунок 1.18. Скріншот завантаження

Default Project

Database

Indexes (1)

ai-course

Collections

Manage

API Keys

Members

ai-course

Starter

Convert to serverless

Metric

cosine

Dimensions

1536

Pod Type

starter

Host

https://ai-course-83ce35b.svc.gcp-starter.pinecone.io

Cloud

GCP

Region

Iowa (us-central1)

Environment

gcp-starter

Type

Pods

VECTOR COUNT

3

New!

Convert this into a serverless index and scale past 100k records, all in the Starter tier.

BROWSER

METRICS

NAMESPACES (1)

Namespace

(Default)

Query by Vector

0.45,0.98,0.58,0.55,0.44,0.96,0.64,0.03,0.44,0.91,0.26,0.99,0.70,79.0,8.45,0.66,0.27,0.48,0.55,0.41,0.07,0.12,0.24,0.03,0.2,0.88,0.93

Top K *

10

Query

Metadata Filter

Matches: 1-3 of 3

+ Upsert Records

STARTER USAGE

RUs

0 / 1M

WUs

0 / 2M

Storage

0 / 20B

ID	VALUES
1	56fc56ce-a2... 0.00882261805, 0.00475665508, -0.000126654093, -0.0389064324, -0.00270249695, 0.01467399557, -0.0177364592, -0.0326319556, -0.0402426049...

SCORE

-0.0073

METADATA

subject: "top"

text: "Тема: Поняття складності алгоритму. Поняття складності алгоритму У процесі вирішення прикладних задач вибір потрібного алгоритму викликає певні труднощі. І справді, на чому ба...

Рисунок 1.19. Скріншот 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.

The screenshot displays an API client window for a POST request to the endpoint `/embedding/query`. The request body is a JSON object: `{ "query": "Як програмісти вирішують, який алгоритм вибрати для вирішення прикладних задач?", "metadata": { "subject": "oop" } }`. The response is a 201 status code with a detailed message in Ukrainian about algorithm selection. The interface includes tabs for Parameters, Request body, Responses, Curl, Request URL, Server response, and Response headers.

```
POST /embedding/query
```

Parameters

No parameters

Request body required

application/json

```
{  "query": "Як програмісти вирішують, який алгоритм вибрати для вирішення прикладних задач?",  "metadata": {    "subject": "oop"  }}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:3000/embedding/query' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "query": "Як програмісти вирішують, який алгоритм вибрати для вирішення прикладних задач?",  "metadata": {    "subject": "oop"  }  }'
```

Request URL

`http://localhost:3000/embedding/query`

Server response

Code	Details
201	Response body Програмісти вибирають алгоритми для вирішення прикладних задач, враховуючи кілька факторів. Вони зважують простоту реалізації алгоритму, його ефективність у використанні комп'ютерних ресурсів та швидкість виконання. Якщо алгоритм використовується лише кілька разів, зазвичай вибирають простіший алгоритм, щоб скоротити час на його написання. Для задач, що потребують значних обчислювальних затрат і багаторазового виконання, можуть віддати перевагу більш складним алгоритмам, які забезпечують швидше виконання, навіть якщо вони вимагають більше пам'яті. Програмісти також розглядають компроміс між часом і простором, а також практичну та теоретичну складність алгоритмів.

Response headers

Рисунок 1.20. Скріншот API використовує OpenAI для аналізу та створення відповідей

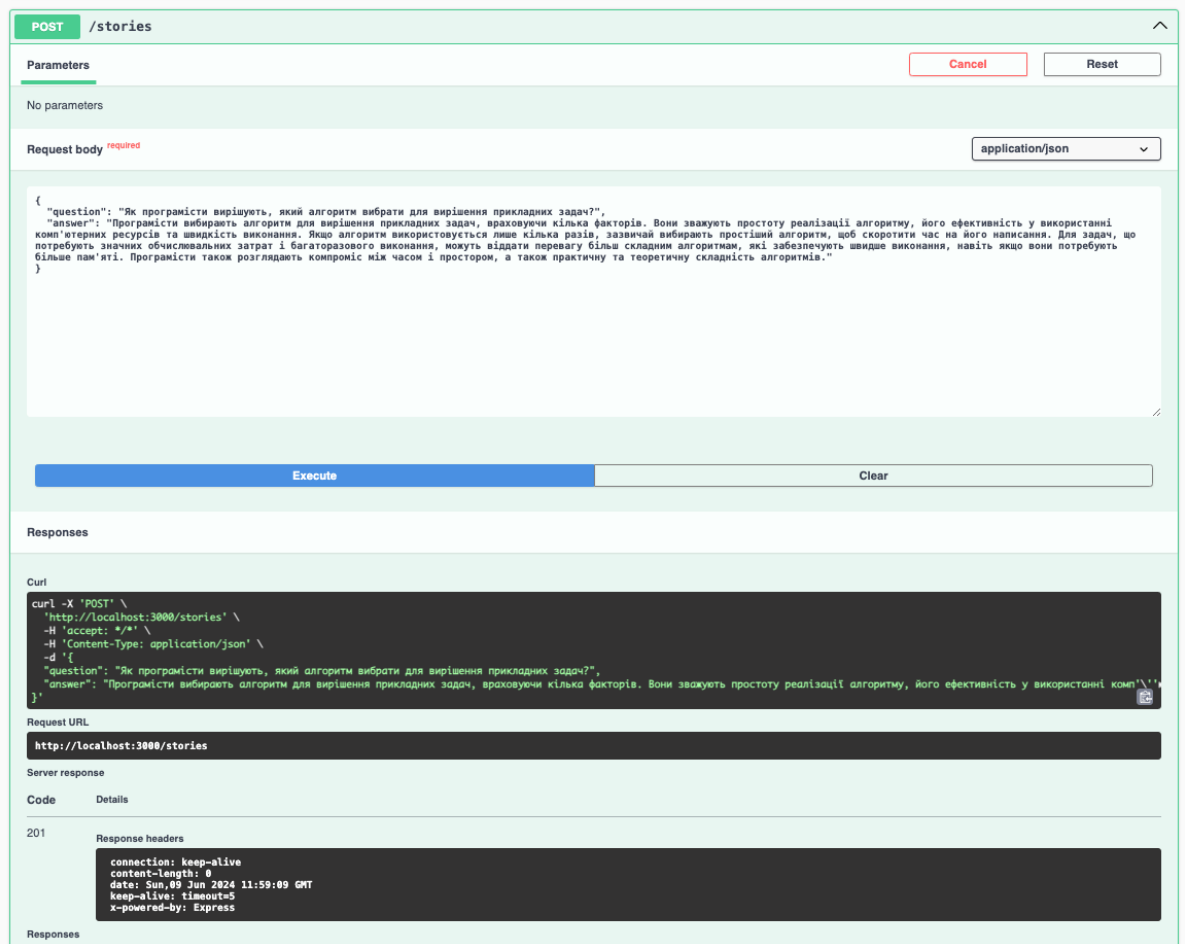


Рисунок 1.21 Скріншот зберігання питання та відповіді в MongoDB

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

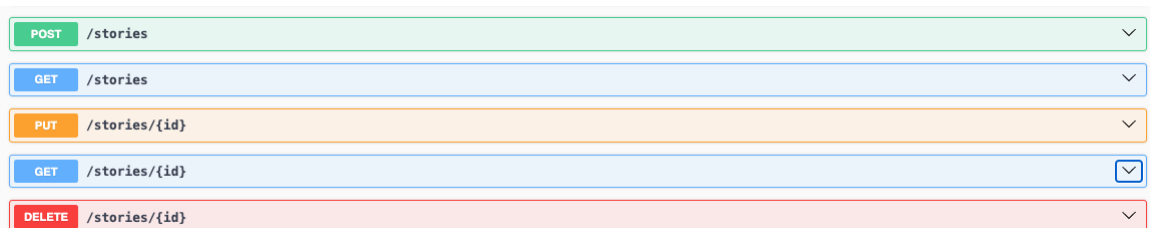


Рисунок 1.22. Скріншот CRUD операції над питаннями та відповідями в базі даних

Операція читання всіх даних із бази.

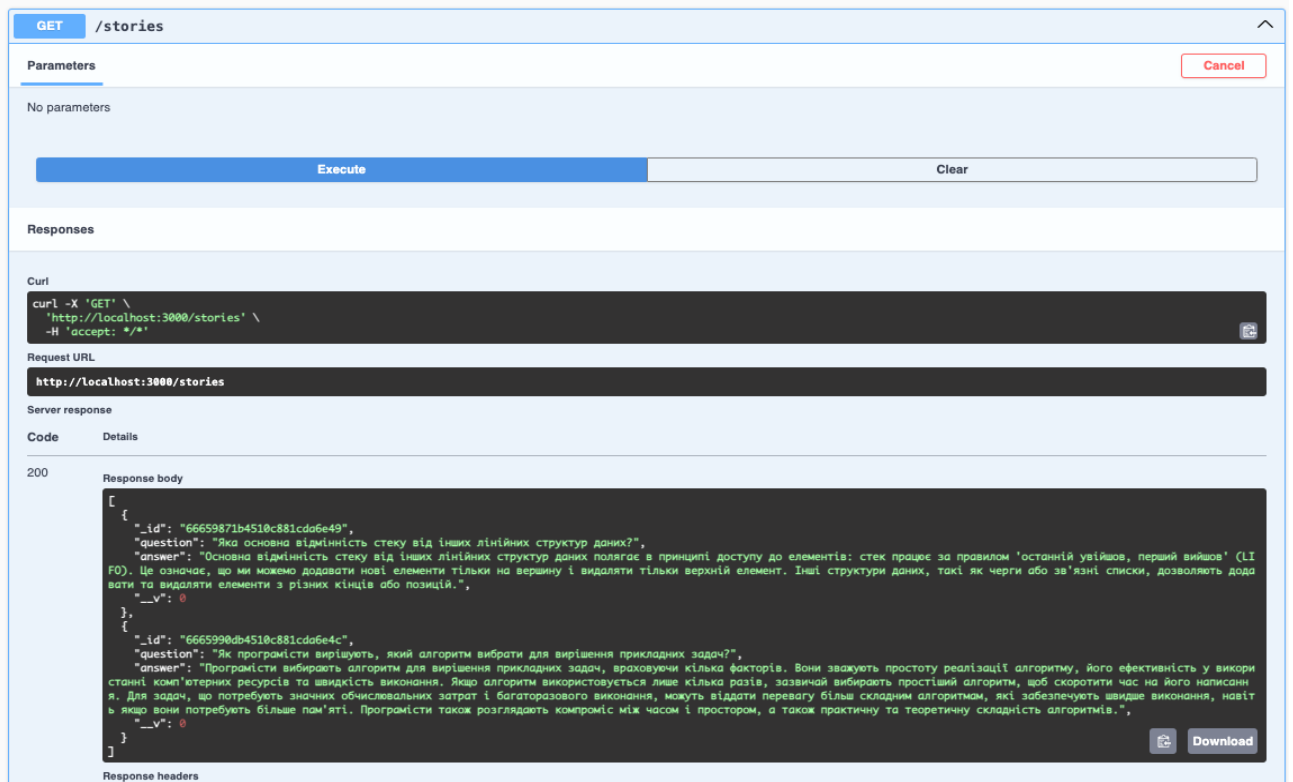


Рисунок 1.23. Скріншот усіх збережених питань і відповідей в бд.

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.

POST /embedding/query

Parameters

No parameters

Request body required

application/json

```
{
  "query": "Створіть 3 запитання по збереженим темам, які можуть задати на іспиті",
  "metadata": {
    "subject": "доп"
  }
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
'http://localhost:3000/embedding/query' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "query": "Створіть 3 запитання по збереженим темам, які можуть задати на іспиті",
  "metadata": {
    "subject": "доп"
  }
}'
```

Request URL

http://localhost:3000/embedding/query

Server response

Code

Details

201

Response body

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.

Download

Рисунок 1.24 Скріншот 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

У результаті виконання дипломного проєкту на тему «Створення web-застосунку цифрового помічника з використанням Open AI» розроблено REST API з CRUD-функціоналом, можливістю створення та пошуку embedding-даних, інтеграцією з OpenAI та підтримкою навчального середовища. API протестовано через Swagger UI, що підтверджує його стабільність і доступність для користувачів.

Розроблена система є повноцінним прикладом сучасного підходу до побудови інтелектуальних цифрових сервісів. Інтеграція штучного інтелекту через OpenAI API, векторну базу Pinecone та гнучку архітектуру на NestJS забезпечила створення ефективної, масштабованої та адаптивної платформи, яка може бути застосована в освіті, бізнесі або будь-якій галузі, де важлива автоматизація, аналіз інформації та інтелектуальна взаємодія з користувачем.

Ефективність програмного забезпечення визначається не лише його якістю, а й успішністю процесу його розробки. Якість ПЗ оцінюють за низкою показників: відповідністю вимогам, ефективністю використання ресурсів і задоволеністю користувача. Також з користувацької перспективи важливими є витрати на створення програмного продукту, зокрема трудові та фінансові. У цьому розділі представлено обчислення вартості розробленого програмного забезпечення.

2.2. Визначення трудомісткості розробки програмного забезпечення.

Тривалість розробки програмного продукту залежить від низки факторів, зокрема обсягу виконуваних робіт, рівня трудомісткості, професійної підготовки розробників, а також строків, обумовлених ринковими вимогами. Для оцінки масштабів програмного забезпечення застосовується метод структурної аналогії, за яким на основі спеціалізованих каталогів аналогічних рішень визначається кількість умовних машинних команд для подібного продукту (у тисячах команд).

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		54

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_0 , усл. машинних командах.
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить V_0 в умовних машинних командах, трудомісткість визначаємо на основі табл.2.

Таблиця.2.2

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	254
3.00	262

На підставі отриманого значення, по довіднику, визначаємо укрупнену норму часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПЗ, тобто в умовах комп'ютера, $K_k=0,7\div 0,8$), для нашого варіанта виділено сірим кольором:

$$T_{ар} = 229 \times 0,8 = 183,2 \text{ (люд/годин)}.$$

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:

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

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

K_T – 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

Код ступеня новизни	Ступінь новизни	Значення K_n
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 K_T
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

$$T_{ТЗ} = T^a * L_1 * K_n = 183,2 * 0,12 * 0,7 = 15,38 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$T_{ТП} = T^a * L_2 * K_n = 183,2 * 0,11 * 0,7 = 14,11 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$T_{РП} = T^a * L_3 * K_n * K_T = 183,2 * 0,61 * 0,7 * 0,6 = 46,94 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{ТЗ} = 2$ (стор), розробка ТП $N_{ТП} = 22$ (стор), розробка

робочого проекту $N_{pp}=10$ (стор), пояснювальна записка відповідно $N_{пз}=26$ (стор)
Розрахунок зведений у таблицю 2.5

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{p_{ТЗ}}=15,38$	$T_{kk}=0,7*N_{ТЗ}=0,7*2=1,4$	$T_{нк}=0,15*N_{ТЗ}=0,15*2=0,3$
2.Розробка ТП	$T_{p_{ТП}}=14,11$	$T_{kk}=0,7*N_{ТП}=0,7*22=15,4$	$T_{нк}=0,15*N_{ТП}=0,15*22=3,3$
3.Розробка РП	$T_{p_{рп}}=46,94$	$T_{kk}=0,7*N_{рп}=0,7*10=7,0$	$T_{нк}=0,15*N_{рп}=0,15*10=1,5$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}=1,5*26=39$	$T_{kk}=0,7*N_{ТЗ}=0,7*26=18,2$	$T_{нк}=0,15*N_{пз}=0,15*26=3,9$
Усього, в т.ч.:	166,43		
- на розробку	$\Sigma T_p=115,43$		
- контроль керівника		$\Sigma T_{kk}=42,0$	
- нормоконтроль			$\Sigma T_{нк}=9,0$

2.3 Розрахунок ціни програмного продукту.

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, загальні витрати на розробку ПЗ. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2025» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2025 року - 8000 гривень; мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	115,43	48,00	5540,64
2.Контроль керівника	42,0	115,00	4830,00
3.Нормоконтроль	9,0	120,00	1080,00
Усього	-	-	$\Sigma \text{Зо} = 11450,64$

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	60	5.0	300
Разом	-	-	-	$B_{mi}=300$
Транспортно–заготівельні витрати (10%)				$B_{mp_3} = 0,1 \times B_{m1} = 30$
Усього				$B_m = B_{mi} + B_{tp_3} = 330$

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	330,0	B_m (див. табл. 2.8.)
2. Основна заробітна плата	11450,64	$З_o$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1145,06	$З_d = 0,1 \times З_o = 11450,64 \times 0,1$
4.Відрахування до єдиного фонду соціального внеску	2771,05	$В_{\epsilon.c.v.} = 0,22 \times (З_o + З_d) = 0,22 \times (11450,64 + 1145,06)$
5. Накладні витрати	4580,26	$В_{нак.} = 0,4 \times З_o = 0,4 \times 11450,64$
6. Повна собівартість	20277,01	$C_{пов} = B_m + З_o + З_d + В_{\epsilon.c.v.} + В_{нак.} = 330,0 + 11450,64 + 1145,06 + 2771,05 + 4580,26$

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

$$П = (C_{п} \cdot P) / 100 = (20277,01 \cdot 10) / 100 = 2027,7 \text{ грн} \quad (2.6)$$

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

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_o = C_{пов} + П = 20277,01 + 2027,7 = 22304,71 = 25920,05 \text{ грн} \quad (2.7)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 22304,71 + 22304,71 \cdot 0.2 = 26765,65 = 5184,01 \text{ грн} \quad (2.8)$$

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

$$Ц_p = Ц_o + ПДВ = 25920,05 + 5184,01 = 31104,06 \text{ грн} \quad (2.9)$$

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		58

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Охорона праці на підприємстві завжди має ключове значення, оскільки саме завдяки дотриманню відповідних рекомендацій працівники формують безпечні алгоритми виконання службових обов'язків. Головною метою охорони праці є забезпечення безпеки життя, збереження працездатності та здоров'я працівників у процесі виконання ними трудових функцій.

У сфері роботи з комп'ютерною технікою, як і в інших галузях, необхідно враховувати такі параметри, як:

- рівень освітлення;
- температура повітря;
- відносна вологість;
- рівень вібрації.

Особливу увагу при цьому слід приділяти дотриманню вимог пожежної безпеки, зокрема:

- вогнестійкості приміщення;
- допустимому рівню шумового навантаження;
- характеристикам електромагнітного, ультрафіолетового та інфрачервоного випромінювання.

У межах дипломного проєкту розглянуто питання охорони праці, зосереджуючи увагу на аналізі безпеки професійної діяльності розробника веб-сторінок в умовах офісного середовища.

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 скласти технічне завдання разом з клієнтом.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		59

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

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		60

м/с. Якщо ці норми перевищені, робочий 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 $f_{\text{мер}}=100$ Гц, що відповідає умові $f_{\text{мер}}>70$ Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехтливність 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.

Забезпечення здорових і безпечних умов праці покладається на адміністрацію підприємств, установ, організацій. Вона зобов'язана запроваджувати сучасні засоби безпеки праці, які мають попереджувати виробничий травматизм і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		61

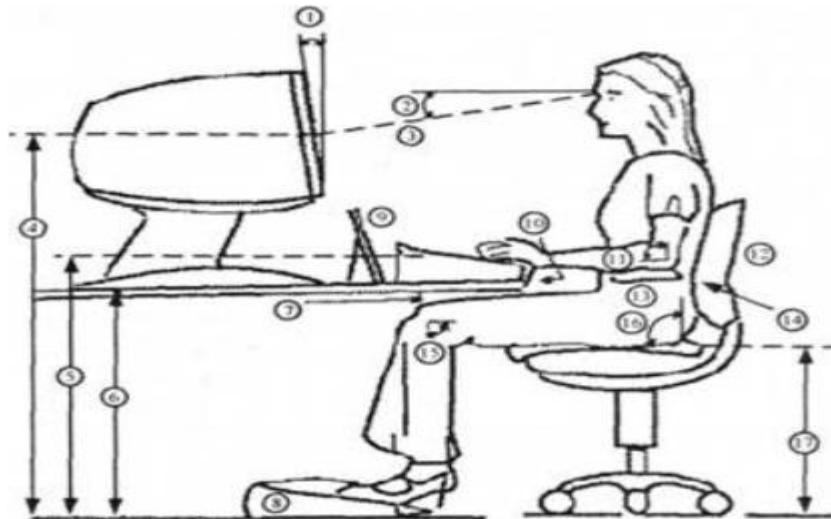


Рисунок 3.1. Робоче місце користувача ПК

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, ломы 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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		62

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.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		63

ВИСНОВКИ

У результаті виконання дипломного проєкту на було досягнуто таких основних результатів.

Проведено глибоке дослідження типів і класифікацій цифрових помічників (голосові, текстові, гібридні) та проаналізовано їх можливості, переваги й перспективи розвитку з використанням технологій штучного інтелекту.

Показано, як ШІ підвищує ефективність цифрових сервісів через індивідуалізацію, адаптивність, обробку природної мови, а також інтеграцію з IoT-пристроями. Визначено важливість безпеки, етики та галузевої спеціалізації при впровадженні таких систем.

Створено архітектурну модель серверного застосунку на основі Node.js та NestJS, яка включає взаємодію з базами даних MongoDB та Pinecone, використання OpenAI API, документацію Swagger та розширену підтримку інтеграції з клієнтськими додатками.

Успішно реалізовано взаємодію з OpenAI API для роботи з великими мовними моделями (LLM) з підтримкою семантичного пошуку через Pinecone. Показано переваги підходу Retrieval-Augmented Generation (RAG) для генерації точних і релевантних відповідей.

Успішно виконано всі етапи розробки від планування до впровадження. Проведено тестування, продемонстровано збереження та обробку даних, створення embedding-даних і генерацію відповідей на освітні запити.

Розроблена система є повноцінним прикладом сучасного підходу до побудови інтелектуальних цифрових сервісів.

Інтеграція штучного інтелекту через OpenAI API, векторну базу Pinecone та гнучку архітектуру на NestJS забезпечила створення ефективної, масштабованої та адаптивної платформи, яка може бути застосована в освіті, бізнесі або будь-якій галузі, де важлива автоматизація, аналіз інформації та інтелектуальна взаємодія з користувачем.

.

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		64

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Андрущенко І. М. Docker у практиці DevOps: Навчальний посібник- Київ: Університет «Україна», 2021- 180 с.
2. Бондаренко О. О. Основи веб-програмування. Навчальний посібник- Київ: КНЕУ, 2020- 248 с.
3. Гуменюк І. І., Степаненко О. П. Основи роботи з Node.js у веб-розробці- Івано-Франківськ: НАІР, 2022- 98 с.
4. Коваленко Ю. В., Пархоменко О. І. Технології програмування у хмарних середовищах- Чернігів: ЧНТУ, 2020- 160 с.
5. Кузьмінський А. В., Романюк І. В. Інтернет-технології: навчальний посібник- Київ: Центр учбової літератури, 2019- 296 с.
6. Литвиненко В. М. Штучний інтелект та машинне навчання: теорія і практика- Дніпро: ДНУ ім. Олеся Гончара, 2023- 205 с.
7. Цимбалюк С. О., Івахненко Н. В. Технології та засоби створення веб-орієнтованих інформаційних систем- Львів: ЛНУ імені Івана Франка, 2022- 184 с.
8. Черевко О. В. Інформаційні системи і технології в управлінні- Харків: ХНЕУ, 2021- 312 с.
9. Документація NestJS (переклад українською мовою) [Електронний ресурс]- Режим доступу: <https://docs.nestjs.com>
10. Офіційна документація Docker українською [Електронний ресурс]- Режим доступу: <https://docs.docker.com>
11. <https://blog.webnrsolutions.com/visual-studio-code> – [Overview of VS Code].
12. <https://generalassemb.ly/blog/what-is-a-javascript-framework> – [JavaScript Frameworks Explained].
13. <https://itvdn.com> – [Онлайн-платформа з курсами з програмування].
14. <https://www.monocubed.com/blog/advantages-of-vue-js> – [Переваги Vue.js у веб-розробці].

					КГ 08. 19 001. 00 ДП ПЗ	Арк.
Ізм.	Лист	№ докум.	Підпис	Дата		65

Додаток А. Програмний код «Опис ендпоїнтів»

openapi: 3.0.0

info:

title: User Management API

version: 1.0.0

description: API для керування користувачами

paths:

/users:

get:

summary: Отримати список користувачів

responses:

'200':

description: Успішна відповідь

content:

application/json:

schema:

type: array

items:

\$ref: '#/components/schemas/User'

post:

summary: Створити нового користувача

requestBody:

required: true

content:

application/json:

schema:

\$ref: '#/components/schemas/NewUser'

responses:

'201':

description: Користувача створено

/users/{id}:

get:

summary: Отримати користувача за ID

parameters:

- name: id

in: path

required: true

schema:

type: integer

responses:

'200':

description: Користувача знайдено

content:

```
    application/json:
      schema:
        $ref: '#/components/schemas/User'
      '404':
        description: Користувача не знайдено
  put:
    summary: Оновити користувача за ID
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/NewUser'
    responses:
      '200':
        description: Успішне оновлення
  delete:
    summary: Видалити користувача
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    responses:
      '204':
        description: Користувача видалено
components:
  schemas:
    User:
      type: object
      properties:
        id:
          type: integer
        name:
          type: string
        email:
          type: string
```

NewUser:
type: object
 required:
 - *name*
 - *email*
 properties:
 name:
 type: string
 email:
 type: string

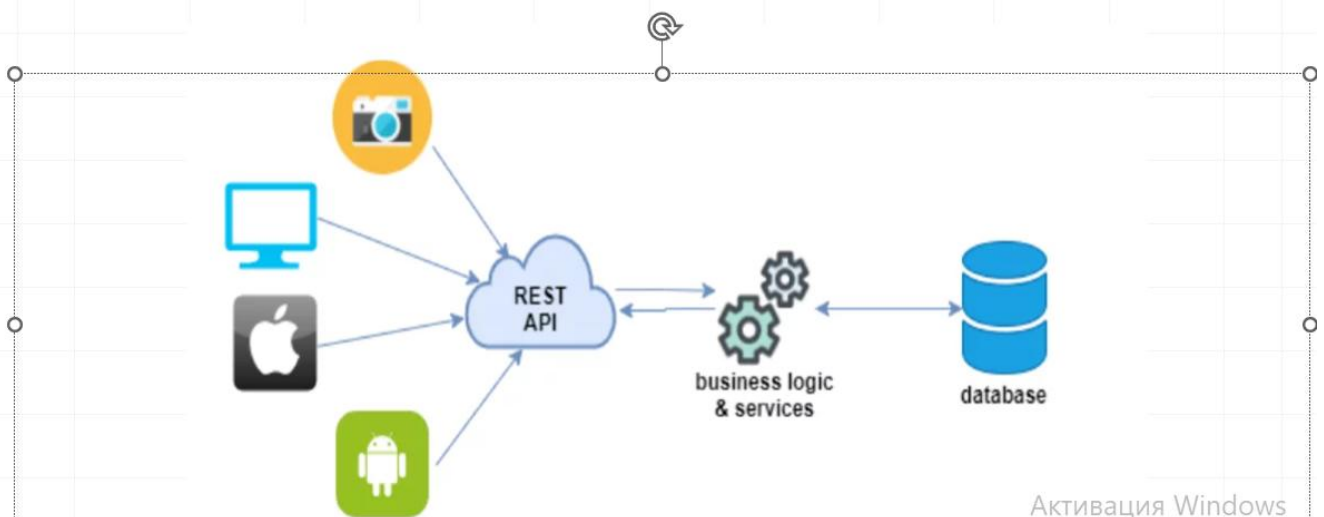
ДИПЛОМНИЙ ПРОЕКТ

- «Створення web-застосунку цифрового помічника на основі штучного інтелекту»

Виконав: студент гр.4 КГ-08 Романенко М.О.

- Керівник: к.т.н. Кунуп Т.В

Схема серверної архітектури для систем на базі штучного інтелекту



Практична частина розробки

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

СТРУКТУРА ПРОЕКТУ

```
> dist
> node_modules
> src
> test
⚙ .env
$ .env.example
🔗 .eslinttrc.js
🔗 .gitignore
📄 .prettierrc
🐳 docker-compose.yml
🐳 Dockerfile
📄 nest-cli.json
📄 package-lock.json
📄 package.json
📖 README.md
$ start.sh
📄 tsconfig.build.json
📄 tsconfig.json
```

Активация Windows
Установите лицензию Windows

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.

```
▼ src
  > configs
  > embedding
  > openai
  > stories
```

Активация Windows

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-

▼ `openai`

▼ `dto`

| `TS completion.dto.ts`

`TS openai.controller.spec.ts`

`TS openai.controller.ts`

`TS openai.module.ts`

`TS openai.service.spec.ts`

`TS openai.service.ts`

Щоб активувати Windows, перейдіть до "Параметри".

Робота програми(скриншот сторінки)

Diplom test API ^{1.0} OAS 3.0

Diplom test API

test

default

GET

/

POST

/stories

GET

/stories

PUT

/stories/{id}

GET

/stories/{id}

DELETE

/stories/{id}

POST

/openai

POST

/embedding/create

POST

/embedding/query

Активация Windows

Чтобы активировать Windows, перейдите к "Параметры".

Скриншот завантаження лекцій через /embedding/create endpoint

The screenshot shows a REST client interface with a POST request to `/embedding/create`. The request body is a JSON object containing a `pageContent` field with a long Ukrainian text about algorithm complexity. The response is a 201 status code with a `Embedding created successfully` message.

```
POST /embedding/create

{
  "pageContent": "Тема: Поняття складності алгоритму. Поняття складності алгоритму У процесі вирішення прикладних задач вибір потрібного алгоритму викликає певні труднощі. І справді, на чому базувати свій вибір, якщо алгоритм повинен задовольнити наступні протиріччя. 1. Бути простим для розуміння, перекладу в програмний код і налагодки. 2. Ефективно використовувати комп'ютерні ресурси і виконуватися швидко. Якщо написана програма повинна виконуватися лише декілька разів, то перша вигода найбільша вагомим. Вартість робочого часу програміста, звичайно, значно перевищує вартість машинного часу виконання програми, тому вартість програми оптимізується за вартістю написання (а не виконання) програми. Якщо мати справу з задачею, вирішення якої потребує значних обчислювальних затрат, то вартість виконання програми може перевищити вартість написання програми, особливо якщо програма повинна виконуватися багаторазово. Тому, з економічної точки зору, перевагу буде мати складний комплексний алгоритм (є надія, що результуюча програма буде виконуватися суттєво швидше, ніж більш проста програма). Але і в цій ситуації розуміємо спонукати реалізувати простий алгоритм, щоб визначити, чи повинна себе висті більш складна програма. При побудові складної програмної системи бажано реалізувати її простий прототип, на якому можна провести необхідні виміри й змодельувати її поведінку в цілому, перш ніж приступати до розробки кінцевого варіанту. Таким чином, програмісти повинні бути обізнані не тільки з методами побудови швидких алгоритмів, але й знати, коли їх потрібно застосувати. Існує декілька способів оцінки складності алгоритма. Програмісти, звичайно, зосереджують увагу на швидкості алгоритму, але важливі й інші фактори, наприклад, до розмірів пам'яті, вільного місця на диску або інших ресурсів. Від швидкого алгоритму може бути мало тому, якщо під нього буде потрібно більше пам'яті, ніж встановлено на комп'ютері. Важливо розрізняти практичну складність, яка є точкою міром часу обчислення і об'єму пам'яті для конкретної моделі обчислювальної машини, і теоретичну складність, яка більш незалежна від практичних умов виконання алгоритму і дає порядок величини вартості. Більшість алгоритмів надає вибір між швидкістю виконання і ресурсами. Задача може виконуватися швидше, використовуючи більше пам'яті, або навпаки – повільніше з меншим обсягом пам'яті. Прикладом в даному випадку може слугувати алгоритм знаходження найкоротшого шляху. Задавши карту вулиць міста у вигляді мережі, можна написати алгоритм, що обчислить найкоротшу відстань між будь-якими двома точками цієї мережі. Замість того, щоб кожного разу заново перерахувати найкоротшу відстань між двома заданими точками, можна наперед прорахувати її для всіх пар точок і зберегти результати в таблиці. Тоді, щоб знати найкоротшу відстань для двох заданих точок, достатньо буде просто взяти готові значення з таблиці. При цьому отримувати результат, практично, миттєво, але це задає величезний обсяг пам'яті. Карта вулиць для великого міста може містити сотні тисяч точок. Для такої мережі таблиця найкоротших відстаней міститиме 6 більше 10 мільярдів записів. В цьому випадку вибір між часом виконання і обсягом необхідної пам'яті очевидний. І з цього зв'язку впливає ідея просторово-часової складності алгоритмів. При цьому підході складність алгоритму лінійно зростає в часі його виконання і залишається незмінною, якщо зменшити алгоритм, спростуючи його структуру."
}
```

```
curl -X 'POST' \
  "http://localhost:3000/embedding/create" \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "pageContent": "Тема: Поняття складності алгоритму. Поняття складності алгоритму У процесі вирішення прикладних задач вибір потрібного алгоритму викликає певні труднощі. І справді, на чому базувати свій вибір, якщо алгоритм повинен задовольнити наступні протиріччя. 1. Бути простим для розуміння, перекладу в програмний код і налагодки. 2. Ефективно використовувати комп'ютерні ресурси і виконуватися швидко. Якщо написана програма повинна виконуватися лише декілька разів, то перша вигода найбільша вагомим. Вартість робочого часу програміста, звичайно, значно перевищує вартість машинного часу виконання програми, тому вартість програми оптимізується за вартістю написання (а не виконання) програми. Якщо мати справу з задачею, вирішення якої потребує значних обчислювальних затрат, то вартість виконання програми може перевищити вартість написання програми, особливо якщо програма повинна виконуватися багаторазово. Тому, з економічної точки зору, перевагу буде мати складний комплексний алгоритм (є надія, що результуюча програма буде виконуватися суттєво швидше, ніж більш проста програма). Але і в цій ситуації розуміємо спонукати реалізувати простий алгоритм, щоб визначити, чи повинна себе висті більш складна програма. При побудові складної програмної системи бажано реалізувати її простий прототип, на якому можна провести необхідні виміри й змодельувати її поведінку в цілому, перш ніж приступати до розробки кінцевого варіанту. Таким чином, програмісти повинні бути обізнані не тільки з методами побудови швидких алгоритмів, але й знати, коли їх потрібно застосувати. Існує декілька способів оцінки складності алгоритма. Програмісти, звичайно, зосереджують увагу на швидкості алгоритму, але важливі й інші фактори, наприклад, до розмірів пам'яті, вільного місця на диску або інших ресурсів. Від швидкого алгоритму може бути мало тому, якщо під нього буде потрібно більше пам'яті, ніж встановлено на комп'ютері. Важливо розрізняти практичну складність, яка є точкою міром часу обчислення і об'єму пам'яті для конкретної моделі обчислювальної машини, і теоретичну складність, яка більш незалежна від практичних умов виконання алгоритму і дає порядок величини вартості. Більшість алгоритмів надає вибір між швидкістю виконання і ресурсами. Задача може виконуватися швидше, використовуючи більше пам'яті, або навпаки – повільніше з меншим обсягом пам'яті. Прикладом в даному випадку може слугувати алгоритм знаходження найкоротшого шляху. Задавши карту вулиць міста у вигляді мережі, можна написати алгоритм, що обчислить найкоротшу відстань між будь-якими двома точками цієї мережі. Замість того, щоб кожного разу заново перерахувати найкоротшу відстань між двома заданими точками, можна наперед прорахувати її для всіх пар точок і зберегти результати в таблиці. Тоді, щоб знати найкоротшу відстань для двох заданих точок, достатньо буде просто взяти готові значення з таблиці. При цьому отримувати результат, практично, миттєво, але це задає величезний обсяг пам'яті. Карта вулиць для великого міста може містити сотні тисяч точок. Для такої мережі таблиця найкоротших відстаней міститиме 6 більше 10 мільярдів записів. В цьому випадку вибір між часом виконання і обсягом необхідної пам'яті очевидний. І з цього зв'язку впливає ідея просторово-часової складності алгоритмів. При цьому підході складність алгоритму лінійно зростає в часі його виконання і залишається незмінною, якщо зменшити алгоритм, спростуючи його структуру."
  }'

Request URL
http://localhost:3000/embedding/create

Server response
Code Details
201 Response body
Embedding created successfully
```

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

The screenshot shows a vector database interface with a query result. The query was performed on a vector index named `ai-course`. The result shows a match with a score of `-0.0073` and a metadata field `subject: 'oop'`. The text of the match is the same as the text in the first screenshot.

```
ai-course Starter Convert to serverless

METRIC DIMENSIONS POD TYPE HOST
cosine 1536 starter https://ai-course-83ce35b.svc.gcp-starter.pinecone.io

CLOUD REGION ENVIRONMENT TYPE VECTOR COUNT
GCP Iowa (us-central1) gcp-starter Pods 3

New! Convert this into a serverless index and scale past 100k records, all in the Starter tier.

BROWSER METRICS NAMESPACES (1)
Namespace (Default) Query by Vector 0.45,0.98,0.59,0.55,0.44,0.96,0.64,0.03,0.44,0.91,0.26,0.99,0.70,79,0.8,0.45,0.66,0.27,0.48,0.55,0.41,0.07,0.12,0.24,0.03,0.2,0.88,0.93 Top K: 10 Query

Metadata Filter

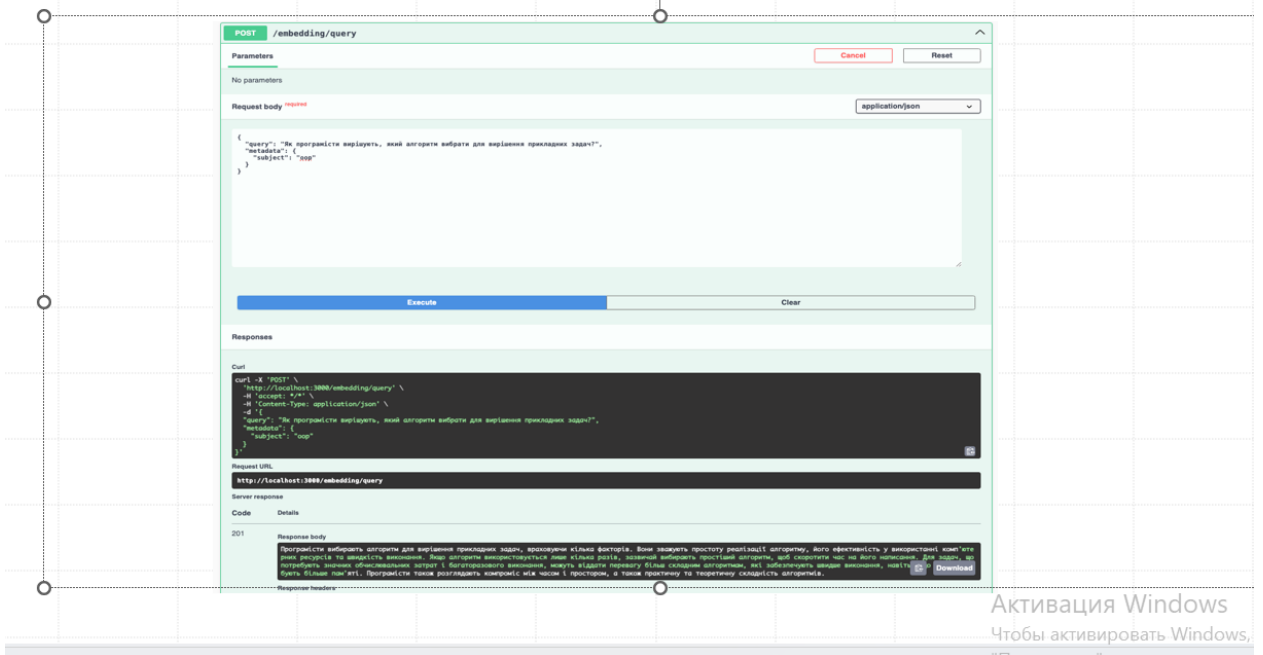
Matches: 1-3 of 3

STARTER USAGE
RUs 0 / 1M
WUs 0 / 2M
Storage 0 / 2GB

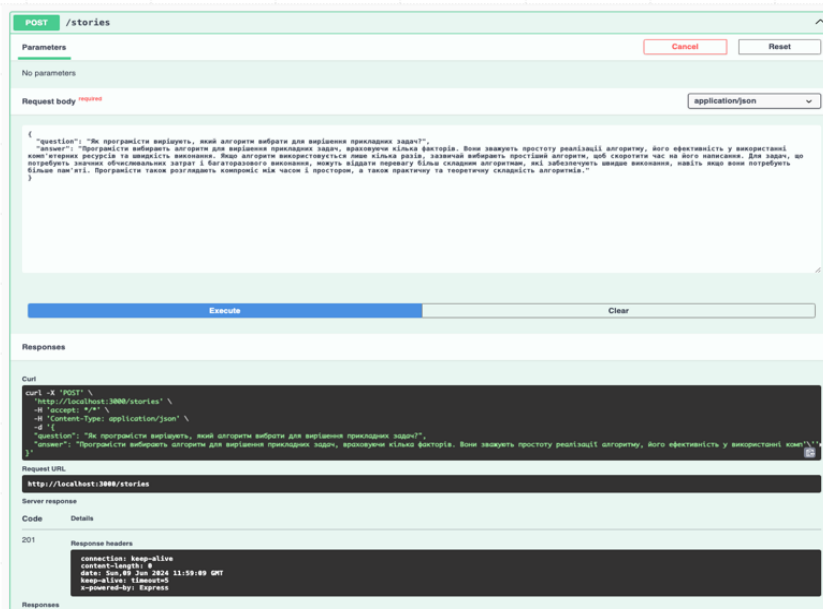
1 ID VALUES
56fc56ce-a2... 0.00882261805, 0.00475665508, -0.000126654093, -0.0369064324, -0.00270249695, 0.01467399557, -0.0177364592, -0.032631956, -0.0402426049...

SCORE -0.0073
METADATA
subject: 'oop'
text: "Тема: Поняття складності алгоритму. Поняття складності алгоритму У процесі вирішення прикладних задач вибір потрібного алгоритму викликає певні труднощі. І справді, на чому ба..."
```

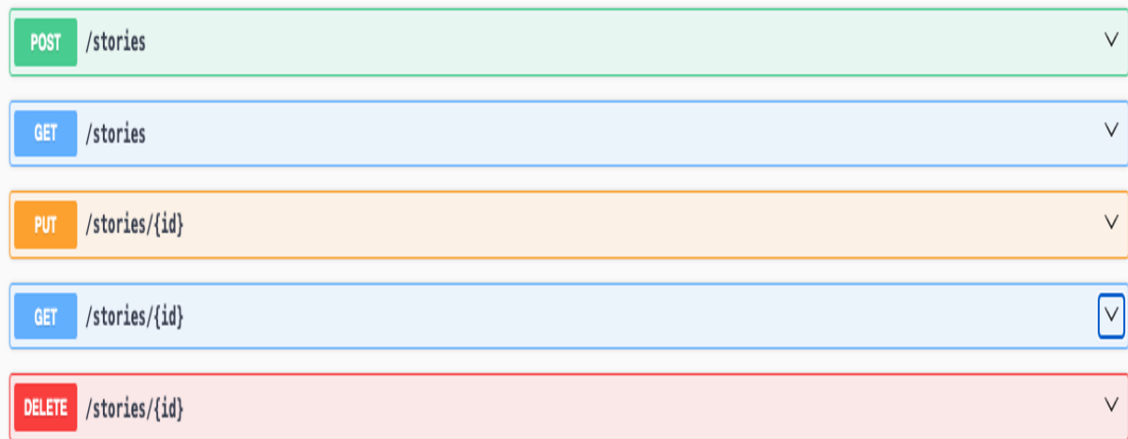
API використовує OpenAI для аналізу та створення відповідей



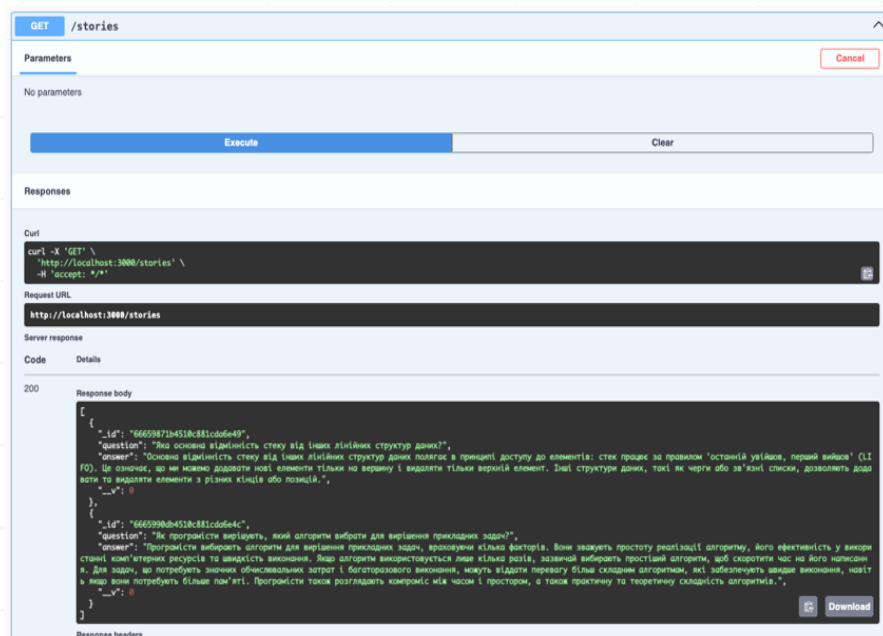
Скриншот зберігання питання та відповіді в MongoDB



Скриншот CRUD операції над питаннями та відповідями в базі даних



Скриншот усіх збережених питань і відповідей в бд.



Висновки

- Результатом проєкту є готовий до впровадження серверний застосунок, який може бути використаний у різних галузях — освіті, бізнесі, консалтингу — як основа для створення інтелектуального цифрового помічника.

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти
відділення комп'ютерних систем

Романенка Микиту Олександровича

(прізвище, ім'я та по батькові)

Спеціальність *123 Комп'ютерна інженерія*

Освітньо-професійна програма *Комп'ютерна графіка і Web-дизайн*

Керівник дипломного проекту (роботи) *Кунуп Тетяна Василівна*

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) *Створення web-застосунку цифрового помічника з використанням Open AI*

Обсяг розрахунково-пояснювальної записки *76* сторінок

Обсяг графічної (презентаційної) частини *13* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

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

*Робота відповідає технічному завданню до дипломного проекту.
Виконана у відповідності з вимогами*

б) характеристика виконання кожного розділу дипломного проекту (роботи)

При виконанні дипломного проекту здобувач продемонстрував уміння використовувати останні досягнення науки та техніки, уміння працювати з літературою. Так, здобувач грамотно дослідив та проаналізував усі етапи проектування web-застосунку цифрового помічника з використанням Open AI

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) *Пояснювальна записка та графічна частина відповідає вимогам, виконана якісно та відображає основні елементи створення web-застосунку цифрового помічника з використанням Open AI*

г) перелік позитивних якостей дипломного проекту (роботи)

Розроблено web-застосунок цифрового помічника з використанням Open AI із використанням мови програмування JavaScript, створено архітектурну модель серверного застосунку на основі Node.js та NestJS, яка включає взаємодію з базами даних MongoDB та Pinescone, використання OpenAI API, документацію Swagger та розширену підтримку інтеграції з клієнтськими додатками.

д) основні недоліки дипломного проекту (роботи)

Обмежене тестування на реальних кейсах. Бракує конкретних сценаріїв атак і опису заходів реагування, що робить розгляд захисту доволі поверхневим. Вибір технологій обґрунтований переважно без оцінки конкурентних рішень, тому не до кінця зрозуміло, чому Pinescone або NestJS обрані на користь інших популярних рішень. Наявні помилки оформлення пояснювальної записки

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента к.т.н. Шибасєва Наталя Олегівна

Місце роботи і посада рецензента Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій

Підпис: _____

« 20 » червня 2025р.



ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Романенка Микити Олександровича

(прізвище, ім'я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

Тема дипломного проєкту: Створення web-застосунку цифрового
помічника з використанням Open AI

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 76 сторінки. У пояснювальній записці виконано опис етапів, відповідно до індивідуального завдання, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у дипломному проєкті. Графічна частина складається з 13 слайдів мультимедійної презентації. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проєктом: Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Романенко Микита Олександрович 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 Романенко Микита Олександрович під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

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

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проектування здобувач освіти Романенко М. О. розробив
web-застосунку цифрового з використанням Open AI. Дипломник володіє
навичками написання веб-застосунку за допомогою таких технологій як
мова програмування JavaScript, NestJS, фреймворк LangChain, OpenAi,
технологія Swagger.

Оцінка розрахункової частини _____

Відмінно

Оцінка графічної частини _____

Відмінно

Загальна оцінка _____

Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту _____

Кунун Тетяна Василівна

Місце роботи і посада керівника дипломного проекту _____

НУ "Одеська політехніка" к.т.н. ст. викладач кафедри "Інформаційні
технології".

Підпис _____



« 16 » 06 2025 р.

**ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»**

Ми, що нижче підписалися,

Романенко Микита Олександрович

здобувач освіти гр. 4КГ-08, та

Кунуп Тетяна Василівна,

керівник дипломного проекту,

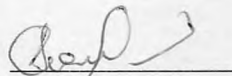
не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Створення web-застосунку цифрового помічника з використанням Open AI» (автор роботи – Романенко М.О., керівник роботи – Кунуп Т.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 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



/ Романенко М.О. /

Керівник



/ Кунуп Т.В. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти ІV курсу
відділення комп'ютерних систем групи 4КГ-08

Романенка Микити Олександровича

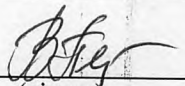
на тему Створення web-застосунку цифрового помічника
з використанням Open AI

Висновок відповідальної особи за проведення нормоконтролю:

пояснювальна записка до дипломного проєкту виконана з деякими

порушеннями ДСТУ та оформлена відповідно до вимог Положення про

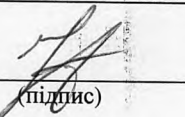
дипломне проєктування


(підпис)

18.06.2025
(дата)

Петрашова В.І.
(П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного
плагиату згідно звіту про перевірку від 29.05.2025 р. значення коефіцієнту
подібності в роботі становить 16,60%, коефіцієнт цитування – 1,23%.


(підпис)

18.06.2025
(дата)

Краснокутська К.Г.
(П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти

Романенка М.О.
(П.І.Б.)

проведена « 18 » червня 2025 р.

Висновки Пояснювальна записка до дипломного проєкту виконана у повному
обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає
вимогам Положення про дипломне проєктування та рекомендована до
захисту.

Голова ЦК КТ та ПІ


(підпис)

Кривченко Ю.В.
(П.І.Б.)

Звіт подібності

метадані

Назва організації

Odesa Technical Professional College of Odesa National University of Technology

Заголовок

Створення web-застосунку цифрового помічника з використанням Open AI

Автор

Науковий керівник / Експерт

Романенко Микита ОлександровичКунуп Тетяна Василівна

Підрозділ

Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25

Довжина фраз для коефіцієнта подібності 2

10707

Кількість слів

83892

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		19
Інтервали		0
Мікропробіли		3
Білі знаки		0
Парафрази (SmartMarks)		78

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Копію тексту означає в якому джерелі він був знайдений. Ці джерела і значення коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Копію тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА (URL (НАЗВА БАЗИ))	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	87 0.81 %
2	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	87 0.81 %
3	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	79 0.74 %
4	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efb0c8/download	77 0.72 %
5	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	74 0.69 %

6	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	71 0.66 %
7	https://ifnm.edu.ua/wp-content/uploads/2024/11/%D1%96%D0%BD%D1%81%D1%82%D1%80-%D0%BF%D1%80%D0%BE-%D0%B7%D0%B0%D1%85-%D0%BF%D0%BE%D0%B6-%D0%B1%D0%B5%D0%B7%D0%BF-%E2%84%9612.pdf	70 0.65 %
8	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	55 0.51 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	49 0.46 %
10	https://studopedia.su/5_17511_pravila-zabezpechennya-pozhezhnoi-bezpeki-v-ofisnih-primishchennyah.html	46 0.42 %

з домашньої бази даних (0.06 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка моделі оптимізації та безпеки передачі даних за допомогою механізму NBAR 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.06 %

з програми обміну базами даних (0.17 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Веб-застосунок для продажу права власності на віртуальні об'єкти з використанням свідоцтва створеного на основі технології блокчейн. Веб-застосунок підтримки процесів продажу віртуальних об'єктів 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	13 (1) 0.12 %
2	2022_60730000_Pik_Vitalii_Iurilovich_96626 10/26/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	5 (1) 0.05 %

з Інтернету (16.37 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	626 (19) 5.85 %
2	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	199 (9) 1.86 %
3	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	104 (3) 0.97 %
4	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbcb/download	77 (1) 0.72 %
5	https://ifnm.edu.ua/wp-content/uploads/2024/11/%D1%96%D0%BD%D1%81%D1%82%D1%80-%D0%BF%D1%80%D0%BE-%D0%B7%D0%B0%D1%85-%D0%BF%D0%BE%D0%B6-%D0%B1%D0%B5%D0%B7%D0%BF-%E2%84%9612.pdf	70 (1) 0.65 %
6	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	62 (4) 0.56 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	59 (3) 0.55 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	57 (2) 0.53 %

9	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	50 (5) 0.47 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content	45 (5) 0.42 %
11	https://studopedia.su/5_17511_pravila-zabezpechennya-pozhezhnoi-bezpeki-v-ofisnih-primishchennyah.html	45 (1) 0.42 %
12	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bffd-1d5ed04c1ce4/content	35 (2) 0.33 %
13	https://ep3.nuwm.edu.ua/18432/1/06-08-16.pdf	24 (2) 0.22 %
14	https://yur-gazeta.com/publications/practice/trudove-pravo/ohrona-praci-v-ofisi-.html	23 (2) 0.21 %
15	https://cyberleninka.ru/article/n/otsenka-funktsionalnyh-harakteristik-programnyh-podsystem-ucheta-tovaryh-zapasov	23 (1) 0.21 %
16	https://repository.lnup.edu.ua/ispu/bitstream/123456789/6171/1/%D0%A1%D1%83%D0%BB%D1%8F%D1%82%D0%B8%D1%86%D1%8C%D0%BA%D0%B8%D0%B9_bach.pdf	22 (2) 0.21 %
17	https://ukrbukva.net/page.9.17108-lzgotovleniye-zhelezobetonnyh-izdeliy.html	21 (2) 0.20 %
18	http://eir.nuos.edu.ua/xmlui/bitstream/handle/123456789/3576/Kovalov-magister.pdf.pdf?sequence=1&isAllowed=y	21 (1) 0.20 %
19	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	21 (2) 0.20 %
20	https://card-file.ontu.edu.ua/bitstreams/341a820e-d025-42f3-b7dc-27e831d6c66f/download	19 (1) 0.18 %
21	https://biz.ligazakon.net/news/185362_dopustim-normi-shumu-v-zbillovikh-ta-gromadskikh-primishchennyakh-onovli	19 (1) 0.18 %
22	https://ukrtxtbook.com/oxorona-praci-moskalova-v-m/oxorona-praci-moskalova-v-m-4-11-pervinni-zasobi-pozhezhhogasinnya.html	18 (1) 0.17 %
23	https://essuir.sumdu.edu.ua/bitstream/123456789/96514/1/Hrytsenko_bak_rob.pdf	18 (2) 0.17 %
24	https://pppb.com.ua/news/pererahuvti-svoji-vognegasnivky-leper-vih-kilkist-zalezhyt-vid-ploshchi-poverhu-budynku	16 (2) 0.15 %
25	https://ekmair.ukma.edu.ua/server/api/core/bitstreams/cda54b2d-2ab9-43d6-a781-928a4fe1bbd1/content	14 (2) 0.13 %
26	https://card-file.ontu.edu.ua/bitstreams/8edf4be9-2a31-4d01-821f-c284e2bfc606/download	12 (1) 0.11 %
27	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	9 (1) 0.08 %
28	https://card-file.ontu.edu.ua/bitstreams/8999d5af-6274-44f4-ae76-d23e08048d38/download	9 (1) 0.08 %
29	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	8 (1) 0.07 %
30	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bdc670e308b/download	7 (1) 0.07 %
31	https://card-file.ontu.edu.ua/bitstreams/63ee88cb-a3d0-4005-9cf2-0cfff89f23c0d/download	7 (1) 0.07 %
32	https://highload.tech/uk/shho-take-crud-prostimi-slovmi-funktsiyi-perevagi-ta-prikladi/	7 (1) 0.07 %
33	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffb7-4469-b6a1-fe84a1fe21cd/download	6 (1) 0.06 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-------	---------------------------------------

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП **ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ**