МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Особливості використання API OpenAI у сучасних Web-розробках**

Виконав: студент 2 курсу, групи 8.1213-іпз-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
М.С. Лавров
(ініціали та прізвище)
Керівник доцент А.І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
"_02_" __вересня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лавров Микола Сергійович
(прізвище, ім'я, по батькові)

1.Тема роботи Особливості використання API OpenAI у сучасних Web-розробках
керівник роботи доцент кафедри електроніки, інформаційних систем та програмного забезпечення, Безверхий Анатолій Ігорович, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___28.02.2024р. № 361-е____

2. Строк подання студентом кваліфікаційної роботи_____30.11.2024_____
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми розпізнавання мов та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 02.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
	Аналіз предметної області	02.09-10.09.24	виконано
	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.24	виконано
	Аналіз проблематики	13.09-14.09.24	виконано
	Дослідження засобів та технологій забезпечення безпеки	15.09-20.09.24	виконано
	Дослідження основних типів існуючих нейронних мереж	21.09-26.09.24	виконано
	Узгодження подальших дій з науковим керівником	27.09-28.09.24	виконано
	Збір тренувального набору для нейронної мережі з відкритих джерел та його попередня підготовка	29.09-13.10.24	виконано
	Генерація коду Веб-застосунку	14.10-16.10.24	виконано
	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	17.10-19.10.23	виконано
0	Обробка отриманих результатів	20.10-09.11.24	виконано
1	Аналіз отриманих результатів	10.11-17.11.24	виконано
2	Оформлення звіту	17.11-27.11.24	виконано

Студент _____ Лавров М.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Безверхий А.І.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 72

Рисунків: 11

Джерел: 27

Лавров М. С. Особливості використання API OpenAI у сучасних Web-розробках: кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник А. І. Безверхий. Запоріжжя : ЗНУ, 2024. 72 с.

Метою дослідницької роботи є систематичний аналіз застосування OpenAI API у процесі розробки веб-застосунків, ідентифікація основних сфер та можливостей використання, оцінка впливу на продуктивність розробки та якість кінцевих продуктів. Робота спрямована на обґрунтування теоретичних та практичних аспектів інтеграції AI інструментів у веб-розробку, з акцентом на інноваційні рішення та вдосконалення існуючих методик. Актуальність теми полягає в тому, що API OpenAI дозволяє значно розширити функціонал додатків, інтегруючи з ними можливості штучного інтелекту, а також підвищити ефективність, швидкість та якість самого процесу розробки. Таке інтегрування удосконалює взаємодію з кінцевим користувачем, забезпечити більшу персоналізацію сервісу, підвищити ефективність обробки даних, покращити досвід розробника.Дослідження охоплює аналіз потенційних можливостей, вивчення практичних застосувань, та формулювання рекомендацій щодо ефективного використання OpenAI API. Ця робота важлива для розробників, студентів, а також компаній, що прагнуть використовувати новітні технології для підвищення конкурентоспроможності їхніх продуктів.

Ключові слова: штучний інтелект, OpenAI API, генеративна нейронна мережа, web-розробка.

SUMMARY

Pages: 72

Figures:11

Sources: 27

Lavrov M.S. Peculiarities of Using the OpenAI API in Modern Web Development: qualification work of the master of specialty 121 "Software Engineering" / scientific adviser A.I. Bezverkhy. Zaporizhzhia : ZNU, 2024. 72 p.

The purpose of this research work is to systematically analyze the application of the OpenAI API in the process of web application development, identify the main areas and opportunities for use, assess the impact on development productivity and the quality of final products. The work is aimed at justifying the theoretical and practical aspects of integrating AI tools into web development, with a focus on innovative solutions and the improvement of existing methodologies. The relevance lies in the fact that the OpenAI API allows you to significantly expand the functionality of add-ons, integrating with them the capabilities of artificial intelligence, as well as increasing the efficiency, fluidity and fluidity of the development process itself. Such integration improves the interaction with the end user, ensures greater personalization of the service, increases the efficiency of data processing, and improves the data of the distributor. analysis of potential possibilities, analysis of practical problems, and formulation of recommendations for effective development of the OpenAI API. This work is important for developers, students, and also companies that want to use new technologies to improve the competitiveness of their products.

Keywords: artificial intelligence, OpenAI API, generative neural network, web-development.

ЗМІСТ

ВСТУП.....8
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ВИКОРИСТАННЯ OPENAI API У ВЕБ-РОЗРОБЦІ 17
1.1Огляд проблеми використання OpenAI API у процесі розробки веб-застосунків 17
1.2Технології використання OpenAI API у процесі розробки веб-застосунків 18
1.3Сфери застосування OpenAI API у процесі розробки веб-застосунків 19
1.4Рішення для використання OpenAI API у процесі розробки веб-застосунків 21
1.5Сучасні підходи використання OpenAI API у процесі розробки веб-застосунків 23
1.6Аналіз програмного забезпечення для використання OpenAI API у процесі розробки веб-застосунків 24
1.7 Висновки з розділу 25
РОЗДІЛ 2 ТЕОРЕТИЧНІ АСПЕКТИ ВИКОРИСТАННЯ OPENAI API У ВЕБ-РОЗРОБЦІ 27
2.1 Поняття та роль мовних мод 27
2.2 Принцип роботи API як сервісу 28
2.3 Chat Completion API 29
2.4 Висновки з розділу 2 30
РОЗДІЛ 3 РОЗРОБКА ЗАСТОСУНКІВ ЩО ВИКОРИСТОВУЮТЬ OPENAI API ДЛЯ ЕФЕКТИВНОЇ РОЗРОБКИ WEB-ЗАСТОСУНКІВ 32
3.1 Основні поняття і терміни 32
3.2 Особливості управління історією запитів у ChatGPT 35

3.3 Особливості управління історією запитів у OpenAI Completion API 36

3.4 Особливості управління історією запитів у OpenAI Assistant API .. 36

3.5 Механізм управління контекстом 37

3.6 Висновки з розділу 3 38

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ РОЗРОБЛЕНИХ ЗАСТОСУНКІВ 39

4.1 Опис функціоналу web-застосунку. 39

4.2 Інструменти для розробки web-застосунку 48

4.3 Опис процесів розробки web-застосунку з використанням OpenAI API 48

4.3.1 Використання ChatGPT для генерації коду проекту веб застосунку 49

4.3.2 Використання OpenAI Assistant API для генерації коду проекту веб застосунку 58

4.3.3 Використання Chat Completion API для генерації коду проекту веб застосунку 63

4.4 Висновки з розділу 4 67

ВИСНОВКИ 69

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 70

ВСТУП

Актуальність теми

Інтеграція штучного інтелекту (AI) і машинного навчання (ML) у веб-розробку змінила цифровий ландшафт, створивши інноваційні, ефективні та інтелектуальні програми. Одним із найбільш революційних досягнень цієї інтеграції є API OpenAI, яке надає розробникам доступ до потужних моделей штучного інтелекту, таких як Chat GPT, для різноманітних додатків, починаючи від створення контенту до автоматизованої підтримки клієнтів. У цій дипломній роботі досліджуються особливості використання OpenAI API в сучасній веб-розробці, зосереджуючись на його практичному застосуванні, перевагах і проблемах.

Актуальність теми полягає в тому, що API OpenAI дозволяє значно розширити функціонал додатків, інтегруючи з ними можливості штучного інтелекту, а також підвищити ефективність, швидкість та якість самого процесу розробки. Таке інтегрування може удосконалити взаємодію з кінцевим користувачем, забезпечити більшу персоналізацію сервісу, підвищити ефективність обробки даних, покращити досвід розробника.

Впровадження пропозицій OpenAI у веб-розробку знаменує значну еволюцію в тому, як розробники підходять до дизайну та функціональності веб-сайтів і веб-додатків. Здатність API розуміти та генерувати людський текст, відповідати на запитання, узагальнювати вміст і навіть генерувати фрагменти коду відкриває безліч можливостей для покращення взаємодії з користувачем, оптимізації створення контенту та автоматизації завдань. Це дослідження не тільки заглиблюється в технічну інтеграцію API OpenAI у веб-проекти, але й вивчає ширші наслідки для розробників, компаній і кінцевих користувачів.

Завдяки детальному аналізу тематичних досліджень і додатків у реальному світі ця робота має на меті надати вичерпний огляд того, як OpenAI API можна використовувати в сучасній веб-розробці для створення більш

динамічного, інтерактивного та інтелектуального цифрового досвіду. Крім того, розглядаються етичні міркування та потенційні проблеми при впровадженні рішень на основі штучного інтелекту, враховуючи, що розробники та зацікавлені сторони поінформовані про найкращі практики відповідального використання штучного інтелекту.

По суті, ця дипломна робота служить основоположним дослідженням ролі OpenAI API у формуванні майбутнього веб-розробки, розумінні його можливостей, обмежень і інноваційних рішень, які воно дає.

Актуальність дослідження особливостей використання OpenAI API в сучасній веб-розробці підкреслюється кількома ключовими подіями та трендами в технологічній індустрії:

Швидкий розвиток технологій штучного інтелекту: галузь штучного інтелекту розвивається безпрецедентними темпами, і OpenAI є лідером багатьох із цих досягнень. API надає доступ до найдосконаліших доступних моделей штучного інтелекту, що робить його дуже актуальним для розробників, які прагнуть включити передові функції ШІ у свої програми.

Підвищений попит на персоналізовану роботу в інтернеті: сучасні користувачі інтернету очікують дуже персоналізованої та інтерактивної роботи. OpenAI API дозволяє розробникам створювати більш привабливу та індивідуальну взаємодію з користувачем, від персоналізованих рекомендацій контенту до інтелектуальних чат-ботів, які можуть розуміти запити користувачів і відповідати на них природною мовою.

Автоматизація та ефективність у створенні контенту: здатність штучного інтелекту генерувати, узагальнювати та перекладати текст може значно оптимізувати процеси створення контенту. Це особливо актуально в епоху, коли контент є королем, а попит на високоякісний і різноманітний контент постійно зростає.

Підвищення доступності та інклюзивності: технології штучного інтелекту можуть грати ключову роль у тому, щоб зробити інтернет більш доступним для людей з обмеженими можливостями. Наприклад, інструменти

на основі штучного інтелекту можуть автоматично генерувати альтернативний текст для зображень або забезпечувати мовний переклад у реальному часі, долаючи бар'єри та створюючи більш інклюзивний цифровий простір.

Етичні та суспільні наслідки: оскільки штучний інтелект стає все більш інтегрованим у веб-розробку, питання щодо конфіденційності, безпеки даних та етичного використання стають дедалі актуальнішими. Обговорення відповідального використання штучного інтелекту, включно з потенційними упередженнями в моделях штучного інтелекту та їхнім впливом на суспільство, є дуже актуальним для розробників, компаній і політиків.

Регуляторні аспекти та аспекти відповідності. Зі збільшенням використання штучного інтелекту все більше уваги приділяється нормам, що регулюють використання даних, конфіденційність і етику штучного інтелекту. Навігація цими законодавчими рамками є важливою для розробників і компаній, щоб забезпечити дотримання вимог і отримати довіру користувачів.

Конкурентні переваги на ринку: компанії, які використовують штучний інтелект у своїх стратегіях веб-розробки, можуть отримати конкурентну перевагу, пропонуючи інноваційні функції та покращуючи залучення користувачів. Розуміння особливостей використання OpenAI API має вирішальне значення для розробників і компаній, які прагнуть залишатися попереду у швидко мінливому цифровому ландшафті.

Враховуючи ці фактори, вивчення застосування OpenAI API у веб-розробці є не тільки своєчасним, але й необхідним для того, щоб йти в ногу з технологічним прогресом, відповідати очікуванням користувачів і вирішувати етичні проблеми, пов'язані з інтеграцією ШІ в цифрові послуги.

Новизна і корисність дослідження полягають в тому, що, впроваджуючи API OpenAI, розробники отримують доступ до найсучасніших досягнень у сфері машинного навчання та штучного інтелекту, що дозволяє створювати продукти нового покоління. Це не лише підвищує конкурентоспроможність розроблених рішень, але й відкриває нові можливості для інновацій.

Інтеграція API OpenAI у веб-розробку пропонує кілька переваг порівняно з традиційними підходами, змінюючи те, як розробники створюють, оптимізують і постачають веб-додатки. Ось деякі з ключових переваг:

Контроль якості: автоматизація тестування та валідації даних з допомогою AI значно покращує якість продукту.

Деплоймент: швидкість та ефективність впровадження нових функцій збільшуються завдяки можливостям швидкого навчання та адаптації.

Безпека: інструменти штучного інтелекту дозволяють краще протистояти кібератакам, аналізуючи загрози в реальному часі.

Вартість: значне зниження витрат на розробку та підтримку за рахунок оптимізації процесів та ресурсів.

Покращена взаємодія з користувачем: моделі OpenAI можуть взаємодіяти з користувачами природною розмовною мовою, роблячи веб-додатки більш інтуїтивно зрозумілими та привабливими. Це значний крок у порівнянні зі статичними формами та попередньо визначеними відповідями, що забезпечує динамічний та персоналізований досвід користувача.

Автоматизація та ефективність створення контенту та керування ним: API може генерувати, узагальнювати та змінювати текст, автоматизуючи процеси створення та контролю контенту. Це скорочує час і зусилля, необхідні для створення високоякісного контенту, дозволяючи розробникам і створювачам контенту зосередитися на стратегії та дизайні.

Генерація коду та налагодження: такі інструменти, як Codex, можуть розуміти програмні запити та генерувати фрагменти коду, допомагати у налагодженні або навіть пояснювати складні блоки коду, прискорюючи процеси розробки та навчання.

Персоналізація та індивідуальний досвід: аналізуючи дані та взаємодію користувачів, моделі штучного інтелекту можуть адаптувати контент, рекомендації та послуги для окремих користувачів, підвищуючи задоволеність і залученість користувачів.

Швидке створення прототипів і експериментування: розробники можуть використовувати API для швидкого створення прототипів нових функцій і експериментувати з функціональними можливостями, керованими штучним інтелектом, сприяючи інноваціям і створюючи гнучкі методи розробки.

Доступ до передових досліджень штучного інтелекту: OpenAI постійно оновлює свої моделі за допомогою останніх досягнень у галузі досліджень штучного інтелекту, надаючи розробникам доступ до найсучасніших технологій без потреби у великому досвіді чи ресурсах у сфері штучного інтелекту.

Мовний переклад і спеціальні можливості: можливості API можна використовувати для автоматичного перекладу вмісту або створення описів для зображень, що робить веб-додатки більш доступними для різноманітної глобальної аудиторії та людей з обмеженими можливостями.

Зменшення часу та витрат на розробку: автоматизуючи повторювані завдання та використовуючи штучний інтелект для створення контенту та підтримки клієнтів, компанії можуть скоротити час на розробку та операційні витрати, дозволяючи перерозподіл ресурсів в інші стратегічні сфери.

Інноваційні функції та послуги: включення функціональних можливостей штучного інтелекту може виділити веб-додатки на переповненому ринку, пропонуючи унікальні функції, які покращують залучення та утримання користувачів.

Обґрунтоване прийняття рішень: API OpenAI, зосереджене на етичній розробці штучного інтелекту, заохочує використання штучного інтелекту з урахуванням впливу на суспільство, конфіденційності та безпеки, спрямовуючи розробників до відповідального використання штучного інтелекту.

У порівнянні з традиційною веб-розробкою, використання OpenAI API дозволяє створювати більш інтелектуальні, ефективні та персоналізовані веб-додатки, які можуть краще задовольняти потреби сучасних користувачів і

компаній. Це являє собою зміну парадигми в бік більш інтерактивного, автоматизованого та інноваційного підходу до веб-розробки.

Перспективи використання API OpenAI у web-розробках надзвичайно широкі. Це включає в себе створення більш інтуїтивно зрозумілих та "розумних" інтерфейсів, автоматизацію рутинних задач, підвищення безпеки додатків та зниження вартості їх розробки та обслуговування.

Мета і завдання дослідження

Метою цієї дослідницької роботи є систематичний аналіз застосування OpenAI API у процесі розробки веб-застосунків, ідентифікація основних сфер та можливостей використання, оцінка впливу на продуктивність розробки та якість кінцевих продуктів. Робота спрямована на обґрунтування теоретичних та практичних аспектів інтеграції AI інструментів у веб-розробку, з акцентом на інноваційні рішення та вдосконалення існуючих методик.

Дослідження охоплює аналіз потенційних можливостей, вивчення практичних застосувань, та формулювання рекомендацій щодо ефективного використання OpenAI API. Ця робота важлива для розробників, студентів, а також компаній, що прагнуть використовувати новітні технології для підвищення конкурентоспроможності їхніх продуктів.

Об’єкт дослідження

Об'єкт дослідження даної роботи – процес розробки веб-застосунків з використанням інструментів штучного інтелекту. Об'єкт охоплює загальні підходи та методики, що застосовуються у веб-розробці, з акцентом на інтеграцію і використання AI для автоматизації, оптимізації та поліпшення якості розроблюваних рішень.

Предмет дослідження

Предмет дослідження – інтеграція та використання API від OpenAI в процесі розробки веб-застосунків. Предмет включає детальний аналіз

можливостей OpenAI API, як засобу для покращення ефективності розробки, зменшення часу на реалізацію проектів та збільшення функціональності продуктів. Розглядаються конкретні приклади використання, переваги та можливі обмеження цих інструментів.

Методи дослідження

У рамках даної дослідницької роботи використовуються різні методи наукового дослідження, які дозволяють глибоко аналізувати використання OpenAI API у веб-розробці. Основні методи включають:

1. Аналітичний метод: Використання аналітичного підходу для розгляду теоретичних матеріалів, технічної документації, і джерел, що описують API OpenAI. Це дозволяє систематизувати наявну інформацію і визначити основні напрямки використання та потенційні можливості.

2. Синтез: Комбінування отриманих даних для формування узагальнених висновків щодо впливу OpenAI API на процеси розробки, їх ефективності та результативності.

3. Порівняльний аналіз: Співставлення можливостей OpenAI API з іншими доступними інструментами AI у веб-розробці для оцінки їх переваг та недоліків.

4. Кейс-стаді метод: Дослідження конкретних прикладів використання OpenAI API на реальних проектах, що дозволяє виявити практичні аспекти застосування та результативність.

5. Метод експертних оцінок: Залучення думок та аналізів від фахівців у галузі веб-розробки, що мають досвід роботи з OpenAI API, для збору кваліфікованих висновків і рекомендацій.

Наукова новизна

Наукова новизна даної дипломної роботи полягає у комплексному аналізі використання OpenAI API як інструменту для розробки веб-застосунків, з акцентом на наступні аспекти:

1. Вивчення інтеграції AI технологій у веб-розробці: Дослідження розглядає як OpenAI API може бути інтегровано у різні стадії розробки веб-застосунків, від проектування до тестування та деплойменту, що є порівняно новою темою в академічних дослідженнях.

2. Аналіз ефективності та продуктивності розробки: Проведення глибинного аналізу з використанням кількісних та якісних метрик для оцінки впливу OpenAI API на швидкість розробки та якість кінцевих продуктів.

3. Формулювання методик оптимізації використання AI у веб-розробці: Розробка та пропозиція методик використання OpenAI API для максимізації ефективності та мінімізації потенційних ризиків, зокрема, з питань безпеки та приватності.

4. Розробка рекомендацій для практичного впровадження: Основа роботи включає розробку практичних рекомендацій для розробників і компаній, які прагнуть інтегрувати OpenAI API у свої проекти, з урахуванням специфіки веб-розробки та бізнес-вимог.

Ці аспекти демонструють наукову значимість дослідження, сприяють розвитку галузі веб-розробки та вносять вклад у теоретичну базу використання AI технологій у програмуванні.

Практичне значення одержаних результатів

Практична значимість цієї кваліфікаційної роботи полягає у внеску в реальне застосування OpenAI API у веб-розробці, забезпечуючи значні вигоди для розробників, команд та організацій, які впроваджують новітні технології. Основні аспекти практичної значимості включають:

1. Покращення продуктивності розробки: Використання OpenAI API дозволяє автоматизувати рутинні завдання, знижувати час на розробку, і збільшувати ефективність кодування, що сприяє більш швидкому випуску продуктів на ринок.

2. Зменшення витрат: Автоматизація допомагає скоротити витрати на трудові ресурси, зменшити кількість помилок на ранніх етапах розробки, що в

подальшому веде до зниження загальних витрат на тестування та виправлення помилок.

3. Підвищення якості продуктів: OpenAI API сприяє розробці більш точного та надійного коду, підвищуючи функціональність і надійність веб-застосунків, що покращує загальний досвід користувача.

4. Інноваційний підхід до розробки: Впровадження штучного інтелекту в процес розробки відкриває нові можливості для створення інноваційних функцій і вдосконалення існуючих веб-застосунків, сприяючи технологічному прогресу.

5. Розширення можливостей для розробників: Навчання та впровадження методів роботи з OpenAI API збільшує кваліфікацію розробників, розширюючи їх можливості у кар'єрному зростанні та розвитку.

6. Підтримка прийняття рішень: Аналітичні можливості OpenAI API можуть бути використані для підтримки прийняття рішень на основі даних, що є критично важливим для бізнес-планування та стратегічного розвитку компаній.

Ці практичні вигоди демонструють, що дослідження має значний потенціал для впровадження в реальні проекти і може служити основою для подальшого розвитку інструментів та підходів у веб-розробці.

Висновок

Розглядаючи особливості використання API OpenAI у сучасних web-розробках, можна зробити висновок, що інтеграція інноваційних технологій штучного інтелекту відкриває нові горизонти для розвитку галузі. Використання таких інструментів не лише сприяє покращенню якості та безпеки продуктів, але й значно знижує витрати на їх розробку та підтримку, що робить тему надзвичайно актуальною і перспективною для дослідження.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ВИКОРИСТАННЯ OPENAI API У ВЕБ-РОЗРОБЦІ

1.1 Огляд проблеми використання OpenAI API у процесі розробки веб-застосунків

OpenAI API почало використовуватись з моменту його випуску, ставши популярним серед розробників завдяки своїй здатності автоматизувати та покращувати процеси розробки. Від великих технологічних компаній до стартапів та індивідуальних розробників, OpenAI API знайшло своє застосування у різних сферах веб-розробки.

OpenAI API офіційно було випущено у червні 2020 року, з тих пір його популярність постійно зростала.

API використовують різні категорії користувачів, включаючи великі корпорації, середні компанії, стартапи та індивідуальні розробники.

OpenAI API інтегрується у інструменти розробника.

Основна причина використання OpenAI API - це його здатність значно підвищити продуктивність розробників, зменшити час розробки та забезпечити більш високу якість продукту. API дозволяє автоматизувати рутинні завдання, генерувати код, проводити тестування та багато іншого.

Переваги використання OpenAI API включають:

- Зниження витрат часу та ресурсів на розробку
- Підвищення якості коду та продукту
- Автоматизація рутинних завдань
- Можливість створення інноваційних функцій та рішень

Недоліки включають:

- Залежність від зовнішнього сервісу
- Потенційні питання безпеки та конфіденційності даних
- Необхідність адаптації та навчання розробників для ефективного використання API

- Можливі обмеження у масштабуванні та продуктивності в залежності від використаного плану API

1.2 Технології використання OpenAI API у процесі розробки веб-застосунків

OpenAI API пропонує зручний інтерфейс та бібліотеки для інтеграції у процес розробки веб-застосунків, забезпечуючи гнучкість та універсальність для різних програмних рішень.

CLI app

CLI (Command Line Interface) додатки дозволяють розробникам взаємодіяти з OpenAI API через командний рядок. Це корисно для автоматизації завдань програмування, написання скриптів для генерації коду, тестування або виправлення помилок без необхідності створювати графічний інтерфейс.

Desktop app

Десктопні додатки можуть інтегрувати OpenAI API для покращення функціональних можливостей IDE та текстових редакторів. Наприклад, OpenAI API може використовуватись для автодоповнення коду, рефакторингу, генерації документації або пошуку помилок у реальному часі.

Web app

Веб-додатки використовують OpenAI API для різних завдань у програмуванні, таких як генерація коду, автоматизація рутинних завдань, створення інтерактивних інструментів для розробників або інтеграція з іншими API для більш ефективного управління кодом та даними.

IDE Extensions or Plugins

Інтеграція OpenAI API у середовища розробки (IDE) через розширення або плагіни дозволяє розробникам отримувати допомогу під час написання коду, виправляти помилки, генерувати приклади та оптимізувати код, що підвищує продуктивність та якість програмного забезпечення.

Mobile app

Мобільні додатки можуть використовувати OpenAI API для покращення функціональних можливостей розробки, таких як автоматичне генерування коду, тестування, відлагодження або створення допоміжних інструментів для мобільних розробників.

1.3 Сфери застосування OpenAI API у процесі розробки веб-застосунків

OpenAI API пропонує широкий спектр застосувань у програмуванні, які охоплюють різні аспекти розробки та автоматизації коду.

Генерація окремих частин коду

API може генерувати окремі функції, класи, моделі, цикли та елементи UI на основі наданих інструкцій, що значно знижує навантаження на розробників.

Запити до API та баз даних

API дозволяє автоматизувати створення запитів до інших API та баз даних, полегшуючи інтеграцію різних сервісів і управління даними.

Генерація коду окремих файлів

OpenAI API може генерувати цілі файли коду на основі специфікацій або шаблонів, що прискорює розробку і зменшує можливість людських помилок.

Генерація коду повноцінних проектів

За допомогою API можна створювати основні компоненти проекту, такі як налаштування серверів, маршрутизація, обробка запитів тощо, забезпечуючи швидкий старт для розробників.

Генерація тестів та тестових сценаріїв

API дозволяє автоматично генерувати тести для перевірки функціональності, що покращує якість коду та знижує витрати на тестування.

Генерація документації

API може створювати документацію для коду, пояснюючи його функції, параметри та приклади використання, що робить проект більш доступним для інших розробників.

Пошук та виправлення помилок

OpenAI API допомагає знаходити і виправляти помилки у коді, пропонуючи можливі рішення та оптимізації.

Рефакторинг

API автоматично оптимізує код, роблячи його більш читабельним та ефективним, без зміни функціональності.

Оптимізація алгоритмів

API може покращувати існуючі алгоритми, підвищуючи їх ефективність та продуктивність.

Модифікація функціоналу коду

API дозволяє змінювати функціональність існуючого коду, додаючи нові можливості або покращуючи існуючі.

Конвертація коду у інші мови програмування

API може перекладати код з однієї мови програмування на іншу, що полегшує міграцію проектів між різними технологіями.

Генерація коду на основі зображення

API може створювати код на основі зображень або дизайнів з графічних редакторів, таких як Figma, що пришвидшує процес створення інтерфейсів.

Пояснення коду

API пояснює функціональність та логіку існуючого коду, що допомагає розробникам зрозуміти та модифікувати його.

Code review

API автоматизує процес перевірки коду, виявляючи помилки, недоліки та пропонуючи покращення.

Автодоповнення при написанні коду

API допомагає розробникам писати код швидше і з меншими помилками, пропонуючи автодоповнення.

Генерація конфігураційних файлів

API автоматично створює конфігураційні файли для CI/CD систем, лінтерів та інших інструментів, спрощуючи налаштування проекту.

Генерація службових скриптів

API може створювати скрипти для автоматизації різних задач, таких як збірка проекту, деплоймент, резервне копіювання тощо.

Використання у контексті роботи з системами контролю версій

API автоматизує завдання, пов'язані з системами контролю версій, такі як генерація назв для коммітів, описів для pull request, та інше.

Генерація діаграм класів чи архітектури проекту

API може створювати діаграми, що ілюструють структуру та архітектуру проекту, полегшуючи його розуміння та документування.

Відповідь на питання зі сфери програмування

API допомагає відповідати на питання розробників, надаючи пояснення та приклади коду.

Допомога у навчанні та освоєнні нових технологій

API сприяє навчанню, надаючи навчальні матеріали, приклади та пояснення нових технологій та інструментів.

1.4 Рішення для використання OpenAI API у процесі розробки веб-застосунків

Існує багато рішень, що використовують OpenAI API для покращення та автоматизації процесу розробки веб-застосунків. Ось кілька популярних прикладів:

Chat GPT

Це інтерактивний чат-бот, заснований на OpenAI GPT, який допомагає розробникам знаходити рішення проблем, надає приклади коду та відповідає на питання щодо програмування.

GitHub Copilot

Інструмент, інтегрований в IDE, який використовує OpenAI API для автоматичного доповнення коду на основі контексту, що дозволяє швидше писати код та знижує кількість помилок.

Pieces desktop app

Додаток для управління фрагментами коду, який використовує OpenAI API для аналізу та оптимізації коду, полегшуючи його організацію та повторне використання.

16x Prompt desktop app

Інструмент для створення та управління запитами до OpenAI API, що допомагає розробникам легко генерувати та тестувати кодові фрагменти.

SpiritAI

Інструмент для створення інтерактивних персонажів та чат-ботів, який використовує OpenAI API для надання більш природного та інтуїтивного спілкування.

Aider (aider-chat) CLI app

Застосунок командного рядка, який використовує OpenAI API для допомоги в розробці, генеруючи та аналізуючи код безпосередньо з командного рядка.

Devin

Інструмент, що використовує OpenAI API для автоматизації процесу тестування коду, допомагаючи знаходити та виправляти помилки.

Codeium

Інструмент для автодоповнення коду, що використовує OpenAI API, підвищуючи швидкість та точність написання коду.

BlackBox AI

Рішення для автоматизації рутинних завдань у розробці, яке використовує OpenAI API для генерації та оптимізації коду.

OpenUI

Інструмент для створення користувацьких інтерфейсів, який використовує OpenAI API для генерації коду UI компонентів на основі опису.

Factory.AI Code Droids

Інструмент, який використовує OpenAI API для створення повноцінних програмних рішень, автоматизуючи процес розробки від початку до кінця.

Vercel v0.dev

Інструмент для автоматизації деплойменту та управління хостингом веб-застосунків, який використовує OpenAI API для оптимізації та налаштування процесів CI/CD.

1.5 Сучасні підходи використання OpenAI API у процесі розробки веб-застосунків

Сучасні підходи до використання OpenAI API пропонують різноманітні методи та технології для підвищення ефективності та автоматизації процесу розробки.

Completion API

Цей API дозволяє генерувати текст або код на основі наданого початку, допомагаючи розробникам автоматично завершувати функції, класи або інші програмні конструкції.

Assistant API

Assistant API надає розробникам можливість створювати інтерактивних помічників, які можуть допомагати у написанні коду, виправленні помилок та оптимізації алгоритмів.

LangChain

LangChain дозволяє об'єднувати декілька мов програмування у одному проекті, використовуючи OpenAI API для перекладу та інтеграції різних мов у єдине рішення.

RAG (Retrieval-Augmented Generation)

Цей підхід комбінує генерацію тексту з пошуком інформації, дозволяючи створювати більш точні та інформативні відповіді на запити розробників.

Agents

Агенти використовують OpenAI API для автоматизації різних завдань у розробці, таких як написання коду, тестування, рефакторинг та управління проектами.

Micro Agents

Мікро агенти - це спеціалізовані інструменти, що виконують конкретні завдання у процесі розробки, використовуючи OpenAI API для швидкого та ефективного вирішення задач.

1.6 Аналіз програмного забезпечення для використання OpenAI API у процесі розробки веб-застосунків

Аналізуючи програмне забезпечення для використання OpenAI API у процесі розробки веб-застосунків, слід звернути увагу на декілька ключових аспектів:

Функціональність

Програмне забезпечення, що використовує OpenAI API, пропонує широкий спектр функцій, включаючи автодоповнення коду, генерацію коду, рефакторинг, тестування та документацію. Ці функції значно спрощують процес розробки та підвищують продуктивність розробників.

Інтеграція

Багато інструментів пропонують глибоку інтеграцію з популярними середовищами розробки (IDE) та платформами, такими як Visual Studio Code, JetBrains IDEs та GitHub. Це забезпечує безперебійний робочий процес і зручність використання.

Продуктивність

Інструменти, що використовують OpenAI API, покращують продуктивність, дозволяючи розробникам швидше писати, тестувати та виправляти код. Вони також знижують ймовірність помилок і підвищують якість кінцевого продукту.

Зручність використання

Більшість програмного забезпечення, що використовує OpenAI API, мають інтуїтивно зрозумілий інтерфейс і зручні функції, які легко освоїти навіть новачкам. Це дозволяє розробникам швидко адаптуватися та ефективно використовувати інструменти у своїй роботі.

Вартість

Вартість використання інструментів, що базуються на OpenAI API, може варіюватися залежно від функціональності та рівня інтеграції. Багато з них пропонують безкоштовні плани або пробні періоди, що дозволяє розробникам оцінити їх ефективність перед покупкою.

Підтримка та спільнота

Наявність активної спільноти та підтримки з боку розробників інструментів є важливим аспектом, що впливає на зручність використання OpenAI API. Багато інструментів мають активні форуми, документацію та підтримку, що допомагає розробникам швидко вирішувати проблеми.

1.7 Висновки з розділу 1

У цьому розділі було досліджено різні аспекти використання OpenAI API у процесі розробки веб-застосунків. Висновки можна сформулювати наступним чином:

Актуальність та популярність: OpenAI API почало використовуватись з моменту його випуску і здобуло популярність серед розробників завдяки своїй здатності автоматизувати та покращувати процеси розробки.

Технології та форми використання: OpenAI API інтегрується у різні форми програмного забезпечення, включаючи CLI, десктопні додатки, веб-додатки, мобільні додатки та розширення для IDE, що забезпечує гнучкість і універсальність для розробників.

Сфери застосування: OpenAI API пропонує широкі можливості для генерації коду, тестування, рефакторингу, оптимізації, конвертації між мовами програмування, пояснення коду, автодоповнення та багатьох інших завдань, що підвищує продуктивність і якість розробки.

Рішення на основі OpenAI API: Різноманітні інструменти, такі як Chat GPT, GitHub Copilot, Pieces, 16x Prompt, SpiritAI, Aider, Devin, Codeium, BlackBox AI, OpenUI, Factory.AI Code Droids та Vercel v0.dev, демонструють ефективність використання OpenAI API для автоматизації та покращення процесу розробки.

Сучасні підходи: Сучасні методи, такі як Completion API, Assistant API, LangChain, RAG, Agents та Micro Agents, показують інноваційні способи використання OpenAI API для підвищення ефективності розробки.

Аналіз програмного забезпечення: Використання OpenAI API у різних інструментах демонструє покращення продуктивності, зручність використання та широкі функціональні можливості, що сприяє оптимізації процесу розробки веб-застосунків.

РОЗДІЛ 2 ТЕОРЕТИЧНІ АСПЕКТИ ВИКОРИСТАННЯ OPENAI API У ВЕБ-РОЗРОБЦІ

2.1Поняття та роль мовних моделей

Теоретична основа OpenAI API спирається на концепцію глибинного навчання та трансформерних архітектур. Мовні моделі, які лежать в основі OpenAI API, є статистичними моделями для обробки природної мови (Natural Language Processing, NLP). Вони навчаються на великих масивах текстових даних і здатні унаочнювати складні контексти, смислові взаємозв'язки та семантичні тонкощі. Brown et al. (2020). "Language Models are Few-Shot Learners." arXiv:2005.14165 – у цій роботі автори представили GPT-3 та продемонстрували, що модель може виконувати безліч завдань без додаткового спеціального навчання, використовуючи лише нульовий, одноразовий чи незначний (few-shot) контекст запитів.

Великомасштабне попереднє тренування мовних моделей, таких як GPT та Codex, забезпечує моделям широкий спектр знань про синтаксис, семантику, архітектуру та практики розробки програмного забезпечення. Завдяки цьому, моделі можуть виконувати завдання програмування та веб-розробки — наприклад, генерувати код, пояснювати фрагменти, оптимізувати алгоритми, створювати шаблони веб-застосунків — без необхідності додаткового глибинного навчання під конкретний проєкт. Іншими словами, модель вже «побачила» достатній контекст у своєму корисному наборі даних, щоб розуміти загальні патерни, фреймворки, мовні конструкції та часто використовувані бібліотеки, що дозволяє їй ефективно застосовувати ці знання у нових середовищах.

Ключові аспекти впливу великомасштабного тренування на здібності моделі:

1. **Засвоєння широкого контексту:**
   Під час попереднього тренування моделі аналізують великі корпуси тексту, включно з репозиторіями коду (GitHub тощо). Це дає моделі уявлення про типові мовні конструкції (HTML/CSS/JS для веб, Python, JS, C#, інше), фреймворки (React, Angular, Django, Node.js), патерни та найкращі практики.
2. **Few-shot та zero-shot навчання:**
   Великомасштабне тренування робить модель здатною виконувати завдання без спеціального «донавчання» на нових даних. Просте надання інструкцій у prompt (few-shot) або навіть один приклад коректної відповіді дозволяють моделі швидко адаптувати свої знання до нового завдання (скажімо, генерації коду для конкретного елемента веб-сторінки).
3. **Кодування семантичного зв'язку між технічними поняттями:**
   Модель завдяки великій вибірці даних вчиться встановлювати семантичні зв'язки між запитами (наприклад, «створи форму реєстрації з валідацією email») та відповідними рішеннями (HTML-форма, скрипт валідації на JavaScript, CSS-стилізація). Це стає можливим без необхідності спеціального тренування, оскільки модель вже «бачила» подібні приклади у наборах даних.

2.2Принцип роботи API як сервісу

OpenAI API функціонує як віддалений сервіс, з яким веб-застосунки взаємодіють через програмні інтерфейси. Теоретично це означає, що веб-розробник може не перейматися побудовою та підтримкою складних мовних моделей локально. Замість цього він звертається до вже підготовленого та оптимізованого мовного середовища через мережевий запит. Така парадигма

«API-first» забезпечує швидку інтеграцію потужних когнітивних можливостей абстрагуючи від громіздкого інфраструктурного навантаження.

Теоретично, веб-розробники отримують змогу абстрагуватися від низькорівневих складнощів машинного навчання. Це дозволяє:

- Зосередитися на логіці бізнес-процесів та інтерфейсі користувача.
- Легко масштабувати проєкт, оскільки потужні серверні ресурси та оптимізації (кешування, шардінг моделей, оновлення) лежать на стороні OpenAI.
- Використовувати готові концептуальні будівельні блоки — мовні моделі, їх оновлення та модифікації, не заглиблюючись у складну архітектуру трансформерів та оптимізацію процесів тренування.

На практиці для підтримки моделями ШІ знань з оновлень або випуску нових бібліотек чи фреймворків вимагає “донавчання” моделей або використання Embedding (математичне уявлення тексту або іншого типу даних, яке кодує семантичні значення у вигляді числового вектора, Завдяки цьому комп’ютеру легше порівнювати, кластеризувати та знаходити семантичну подібність між різними уривками тексту або коду) та баз знань таких як RAG (Retrieval Augmented Generation),.

2.3Chat Completion API

Chat Completion API — це частина OpenAI API, яка дозволяє взаємодіяти з мовними моделями, такими як GPT-4 або GPT-3.5, у форматі діалогу. Вона спеціально розроблена для роботи з чатами, де модель може враховувати контекст попередніх повідомлень, щоб генерувати більш релевантні та природні відповіді.

Застосування Chat Completion APIу веб-розробці дозволяє створювати інтерактивні чат-боти, віртуальних асистентів або інші інструменти, які використовують можливості мовних моделей GPT.

Переваги Chat Completion API:

1. Контекстність: Модель враховує попередні повідомлення, що дозволяє створювати природні діалоги.
2. Гнучкість: Ви можете налаштувати поведінку моделі під свої потреби.
3. Простота інтеграції: API легко інтегрується в будь-який додаток через HTTP-запити.
4. Широкий спектр застосувань:
   - Чат-боти для підтримки клієнтів.
   - Віртуальні асистенти.
   - Автоматизація рутинних завдань.
   - Освітні інструменти.

Обмеження Chat Completion API:

1. Вартість: Використання API платне, і вартість залежить від кількості токенів.
2. Контекстний ліміт: Моделі мають обмеження на кількість токенів у контексті (наприклад, 8k або 32k токенів для GPT-4).
3. Чутливість до інструкцій: Модель може іноді неправильно інтерпретувати інструкції.
4. Етичні обмеження: Модель може генерувати небажаний або некоректний контент, якщо її неправильно використовувати.

2.4 Висновки з розділу 2

1. Теоретичні аспекти використання OpenAI API у веб-розробці охоплюють принципи роботи, архітектурні підходи, безпеку, етичні питання та оптимізацію.
2. Розуміння цих аспектів дозволяє створювати ефективні, безпечні та етичні веб-додатки, які використовують можливості штучного інтелекту.

РОЗДІЛ 3 РОЗРОБКА ЗАСТОСУНКІВ ЩО ВИКОРИСТОВУЮТЬ OPENAI API ДЛЯ ЕФЕКТИВНОЇ РОЗРОБКИ WEB-ЗАСТОСУНКІВ

Через те, що використання ChatGPT проявляється у формі чату, тобто у формі “спілкування” людською мовою, через можливості кастомізації та налаштовування “сервісу” під деякі аспекти користувача, через етику, мораль, інтелігентність, варто, наразі, відзначити деяку невизначеність, як для третіх осіб правильно називати те, з “чим” саме відбувається спілкування, а також як зробити це спілкування найефективнішим для досягнення мети (у даному випадку – генерації коду проекту веб застосунку). Адже є рекомендації не використовувати у промптах ввічливі фрази “будь ласка …” або “чи не міг би ти …”, бо в них немає сенсу, проте фідбек про гарні чи погані відповіді у формі виразів людською мовою може надати моделі корисну інформацію для формування подальших правильних відповідей. Далі у контексті “співбесідника” він буде називатись “модель”.

3.1 Основні поняття і терміни

У контексті даної роботи слід підкреслити такі поняття, як модель, промпт (prompt), промпт інжинірінг (prompt engineering), контекст, контекстне вікно (context window).

Модель у контексті штучного інтелекту та машинного навчання — це математична або статистична репрезентація, створена для виконання певних завдань на основі навчання з даних. Вона використовується для розпізнавання шаблонів, класифікації, прогнозування або інших завдань обробки даних. Модель навчається на основі вхідних даних (тренувального набору), щоб потім застосовувати свої знання для аналізу нових, невідомих даних. Моделі можуть бути простими (лінійна регресія) або складними (нейронні мережі, GPT-3).

Промпт (prompt) у контексті штучного інтелекту та машинного навчання – це вхідний запит або інструкція, яку користувач надає моделі, щоб отримати бажану відповідь або виконання завдання. Промпт може бути у формі тексту, коду, зображення, запитання або будь-яких інших вхідних даних, які спрямовують модель на виконання конкретної задачі. Промпт визначає, яку інформацію або дію очікує користувач від моделі. Наприклад, в OpenAI's GPT-3, промпт може бути фразою або питанням, на яке модель має згенерувати відповідь.

Промпт інжиніринг (prompt engineering) – це процес створення, налаштування та оптимізації промптів (вхідних запитів) для моделей штучного інтелекту, таких як GPT-3.5 або GPT-4o, щоб отримати від них найбільш релевантні та корисні відповіді. Цей процес включає розробку ефективних запитів, які враховують специфіку роботи моделі та цільові задачі. Основні аспекти промпт інжинірингу включають:

1. Формулювання чітких та конкретних запитів.
2. Використання контексту для отримання точних відповідей.
3. Тестування та коригування промптів для покращення результатів.
4. Створення спеціалізованих промптів для конкретних завдань.

Контекст у розробці штучного інтелекту і природної мови — це інформація, яка оточує вхідний запит (промпт) і допомагає моделі краще зрозуміти запит та надати відповідну відповідь. Контекст може включати попередні запити та відповіді, інформацію про конкретну задачу, дані про користувача, або інші релевантні дані, що допомагають моделі "зрозуміти" ситуацію. У контексті промпт інженірінгу, правильне надання контексту може значно покращити якість і точність відповідей моделі.

Контекстне вікно (context window) у машинному навчанні та обробці природної мови — це обсяг тексту або даних, які модель може враховувати під час генерації відповіді або виконання завдання. Контекстне вікно визначає максимальну кількість токенів (слів, частин слів або символів), які модель

може обробити одночасно. Наприклад, для моделей GPT-3 контекстне вікно може становити до 4096 токенів, а для GPT-4 — до 8192 токенів або більше залежно від версії.

Контекстне вікно включає:

1. **Вхідний текст:** Поточний запит або інструкції, що надані моделі.

2. **Попередній контекст:** Попередні запити та відповіді в межах поточної сесії або розмови.

3. **Системні повідомлення:** Інструкції або налаштування, які впливають на поведінку моделі.

4. **Метадані:** Додаткова інформація, така як метадані про користувача або специфічні налаштування сесії.

Ці елементи разом формують весь контекст, який модель використовує для генерації відповіді.

Коли історія запитів та відповідей перевищує розмір контекстного вікна, відбувається наступне:

1. **Обрізання контексту:** Модель відкидає найстаріші запити та відповіді, щоб вмістити нові дані у межах контекстного вікна.

2. **Втрата важливої інформації:** Важливі деталі з початку розмови можуть бути втрачені, що може вплинути на точність та релевантність відповідей моделі.

3. **Зниження якості відповідей:** Відсутність повного контексту може призвести до менш точних або менш інформативних відповідей.

4. **Модель може почати працювати повільніше**, коли історія запитів та відповідей перевищує розмір контекстного вікна. Це пояснюється тим, що моделі доводиться обробляти більше даних і вирішувати, які частини інформації потрібно зберегти, а які відкинути, що збільшує обчислювальні ресурси та час обробки. Управління контекстом і оптимізація історії запитів допомагає зменшити ці затримки та підтримувати швидкість і ефективність роботи моделі.

Правильне управління історією запитів та вибір важливих даних для збереження у контекстному вікні є ключовим для підтримання високої якості відповідей.

Управління контекстом в рамках використання моделей штучного інтелекту включає кілька важливих аспектів:

1. **Обмеження контекстного вікна:** Розуміння максимального розміру контекстного вікна, яке модель може обробити, щоб уникнути втрати важливої інформації.

2. **Управління історією запитів: Історію запитів та відповідей можна редагувати** для більш ефективного та раціонального використання контекстного вікна. Це включає **вибір найбільш релевантних запитів і відповідей, видалення повторюваної або менш важливої інформації, а також оптимізацію формату даних**, щоб забезпечити найбільш точні та корисні результати від моделі. Цей підхід дозволяє максимально використовувати обмежену кількість токенів у контекстному вікні, покращуючи якість взаємодії з моделлю.

3. **Розумне скорочення:** Оптимізація введених даних для включення найважливіших елементів у контекстне вікно.

4. **Налаштування системних повідомлень:** Конфігурація інструкцій та налаштувань для моделі, щоб забезпечити правильне розуміння завдання.

5. **Контекстні підказки:** Використання спеціальних запитів для налаштування контексту, що допомагає моделі зрозуміти специфічні вимоги або завдання.

3.2 **Особливості управління історією запитів у ChatGPT**

У ChatGPT користувачі можуть редагувати свої повідомлення. Це дозволяє виправити помилки або уточнити запити. При редагуванні повідомлення в середині чату всі наступні повідомлення та відповіді видаляються, що може призвести до втрати важливого контексту.

Особливості та обмеження:

- Видалення наступних повідомлень: Після редагування повідомлення всі наступні повідомлення та відповіді видаляються.

- Швидкість та ефективність: Редагування середніх повідомлень може знизити швидкість роботи моделі через повторну обробку контексту.

Приклад:

1. Користувач надсилає повідомлення.

2. Модель відповідає.

3. Користувач редагує своє повідомлення.

4. Модель перераховує контекст, видаляючи всі наступні відповіді.

3.3 **Особливості управління історією запитів у OpenAI Completion API**

У Completion API контекстом є лише окремий запит та відповідь. Для підтримки контексту у нових запитах потрібно зберігати, редагувати та додавати попередні повідомлення до нового запиту вручну.

Особливості:

- Користувацьке управління контекстом: Користувач має самостійно зберігати та редагувати історію запитів.

- Код для зберігання: Потрібен додатковий код для збереження та обробки історії запитів.

Приклад:

1. Зберігайте всі попередні запити та відповіді у масив.

2. При новому запиті додайте цей масив до контексту.

3.4 **Особливості управління історією запитів у OpenAI Assistant API**

У Assistant API контекстом є весь потік повідомлень (Thread). Користувач може редагувати кожне повідомлення у Playground

(https://platform.openai.com/playground/chat?models=gpt-4o) або писати код для редагування історії.

Особливості:

- Редагування у Playground: Користувач може вручну редагувати повідомлення у веб-інтерфейсі Playground.

- Код для автоматизації: Можна писати код для автоматизованого редагування історії.

Приклад:

1. Відкрийте Playground.

2. Виберіть потрібне повідомлення для редагування.

3. Внесіть зміни і збережіть їх.

3.5 **Механізм управління контекстом**

1. Збір даних: Зберігайте всі запити та відповіді.

2. Редагування: Додавайте зміни до історії за потреби.

3. Форматування: Оптимізуйте формат даних для забезпечення релевантного контексту.

4. Інтерфейс користувача: Надання зручного інтерфейсу для перегляду та редагування історії.

При розробці застосунків, що мають бути інструментами для розробки веб застосунків з використанням OpenAI API, для кращого DX (Developer Experience) буде корисно приділити увагу наступному:

1. Зручність: Інтуїтивно зрозумілий інтерфейс для перегляду та редагування історії (контексту).

2. Простота використання: Легкі інструменти для внесення змін до контексту.

3. Візуальні підказки: Ясні індикатори для розуміння, які частини контексту було змінено.

4. Автоматизація: Опції для автоматичного збереження та оновлення контексту.

Управління контекстом є критичним для забезпечення точності та ефективності роботи моделей штучного інтелекту, особливо у складних та довготривалих взаємодіях.

Окремо слід зазначити, що з технічних причин більш ефективним спілкування з моделлю буде англійською мовою.

3.6 **Висновки з розділу 3**

1. У цьому розділі було розглянуто ключові аспекти взаємодії зі штучним інтелектом, зокрема поняття моделі, промпту, промпт інжинірингу, контексту та контекстного вікна. Описано особливості управління контекстом у ChatGPT, OpenAI Completion API, OpenAI Assistant API, важливість ефективного редагування історії запитів та відповідей, щоб забезпечити точність та релевантність роботи моделі.
2. Розуміння та ефективне управління контекстом є ключовими для досягнення високої якості взаємодії зі штучним інтелектом, особливо у складних та довготривалих проектах.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ РОЗРОБЛЕНИХ ЗАСТОСУНКІВ

4.1 Опис функціоналу web-застосунку.

Для використання OpenAI API у процесі розробки web-застосунку було обрано фреймворк Next.js, база даних MySQL, Redis для зебрігання сесій користувачів та наступний функціонал застосунку:

1. Реєстрація користувачів з ролями “user” та “admin”;
2. Автентифікація та авторизація користувачів з використанням sessions та cookies;
3. Для авторизованого користувача відображення профілю та функціонал “Logout”;
4. Для авторизованих користувачів функціонал гри в «Тетріс» зі збереженням стану гри у базі даних з можливістю подальшого продовження гри через деякий час. Гра зберігається для кожного користувача окремо.
5. Для авторизованих користувачів відображення рейтингу найкращих гравців (наразі не реалізовано за браком часу);
6. Для авторизованих користувачів можливість бачити список користувачів, запрошувати інших користувачів у друзі, бачити запрошення у друзі, погоджувати чи відмовляти у встановленні дружніх відносин за запитами (наразі не реалізовано за браком часу);
7. Для авторизованих користувачів відображення стрічки новин, повідомлень, постів користувачів, з можливістю коментувати, відповідати на коментарі, додавати реакції на головній сторінці (наразі не реалізовано за браком часу);
8. Для авторизованих користувачів відображення приватних чатів з друзями або адміністратором (наразі не реалізовано за браком часу);

9. Повідомлення у публічній чи приватній стрічці має динамічну структуру і може містити такі елементи: title, text, image, audio, video, code, pdf, file, link (наразі не реалізовано за браком часу).

Також окремими умовами були:

1. Використання ORM (Object-Relational Mapping) бібліотеки Sequelize;
2. Схема бази даних має включає види відношень між сутностями такі як “один до одного”, “багато до одного”, “багато до багатьох”;
3. Використання міграцій для створення та модифікації таблиць бази даних;
4. Застосування бібліотеки Passport.js для автентифікації;
5. Створення “middleware” для автентифікації та авторизованих запитів;
6. Використання Redux для роботи зі станом клієнтської частини застосунку;
7. Використання бібліотеки Tailwindcss;
8. Mobile First and Responsive design.

На рисунку 1 зображено сторінку логіну:

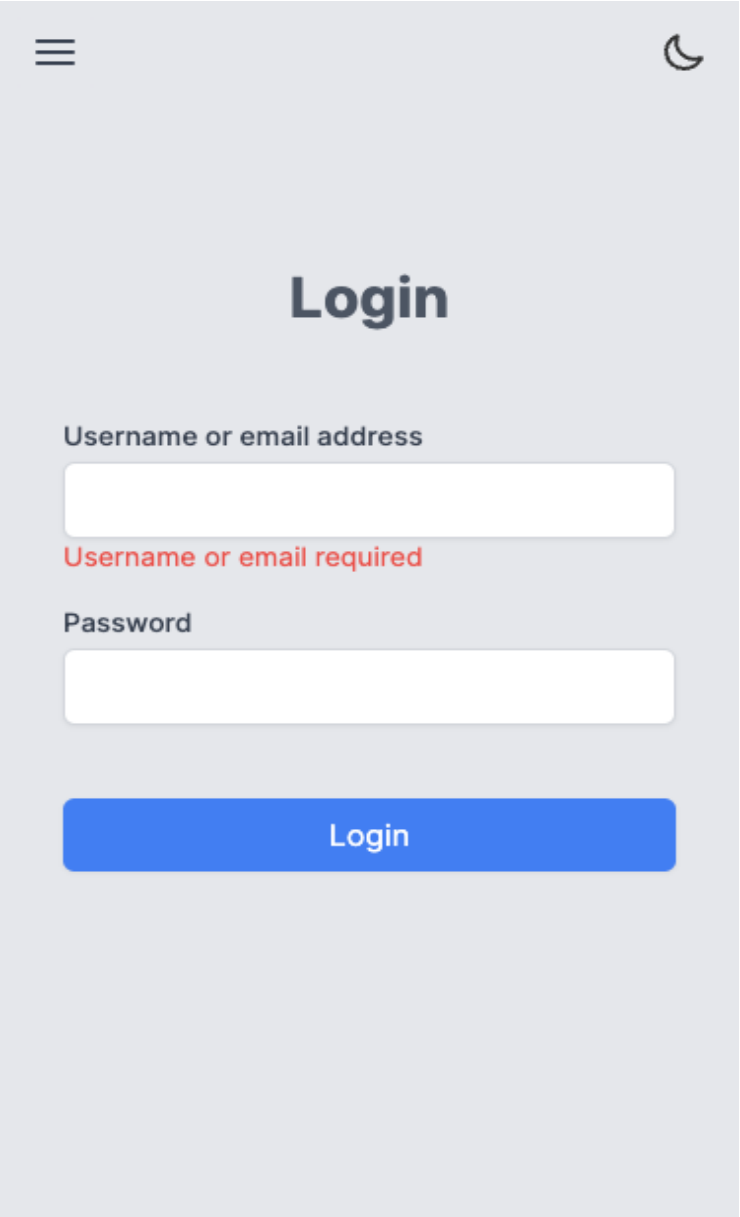


Рисунок 1 – Сторінка логіну

На рисунку 2 зображено сторінку профілю:

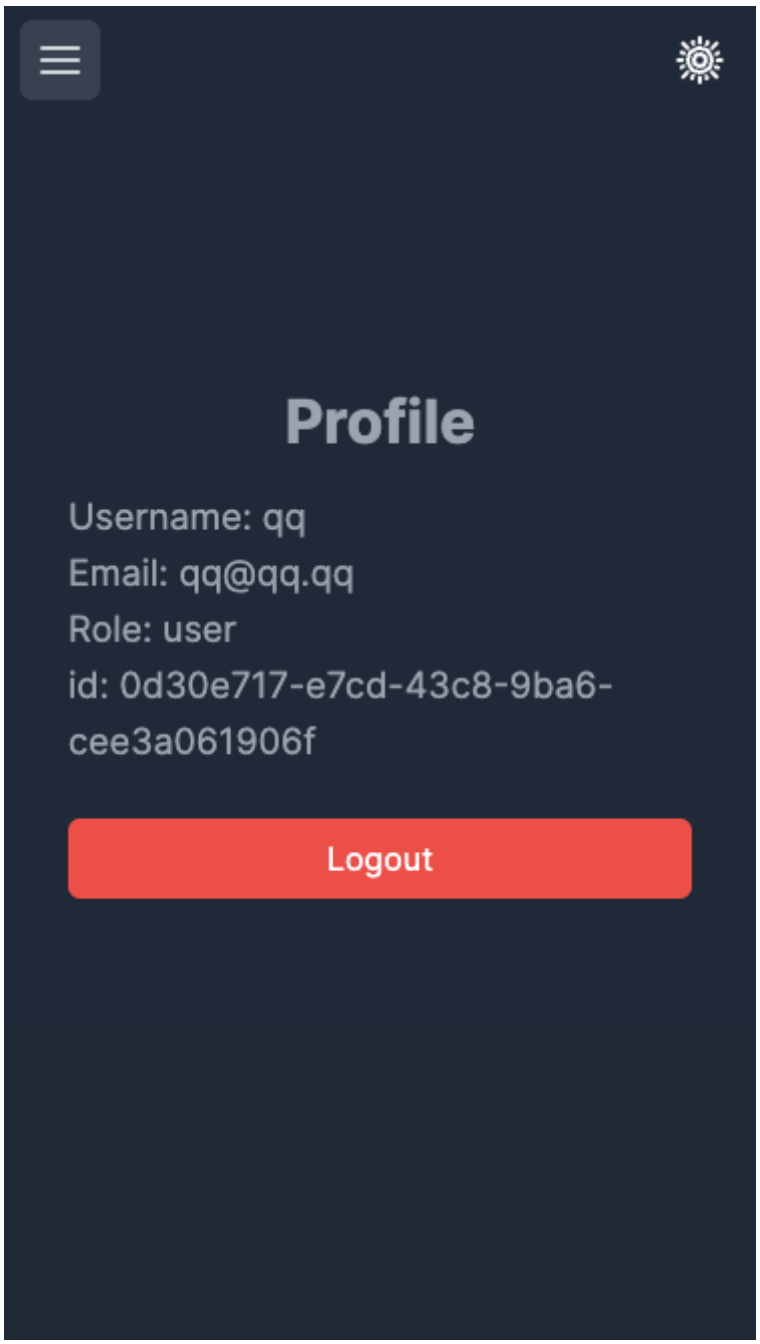


Рисунок 2 – Сторінка профілю

На рисунку 3 зображена сторінка гри в “Tetris”:

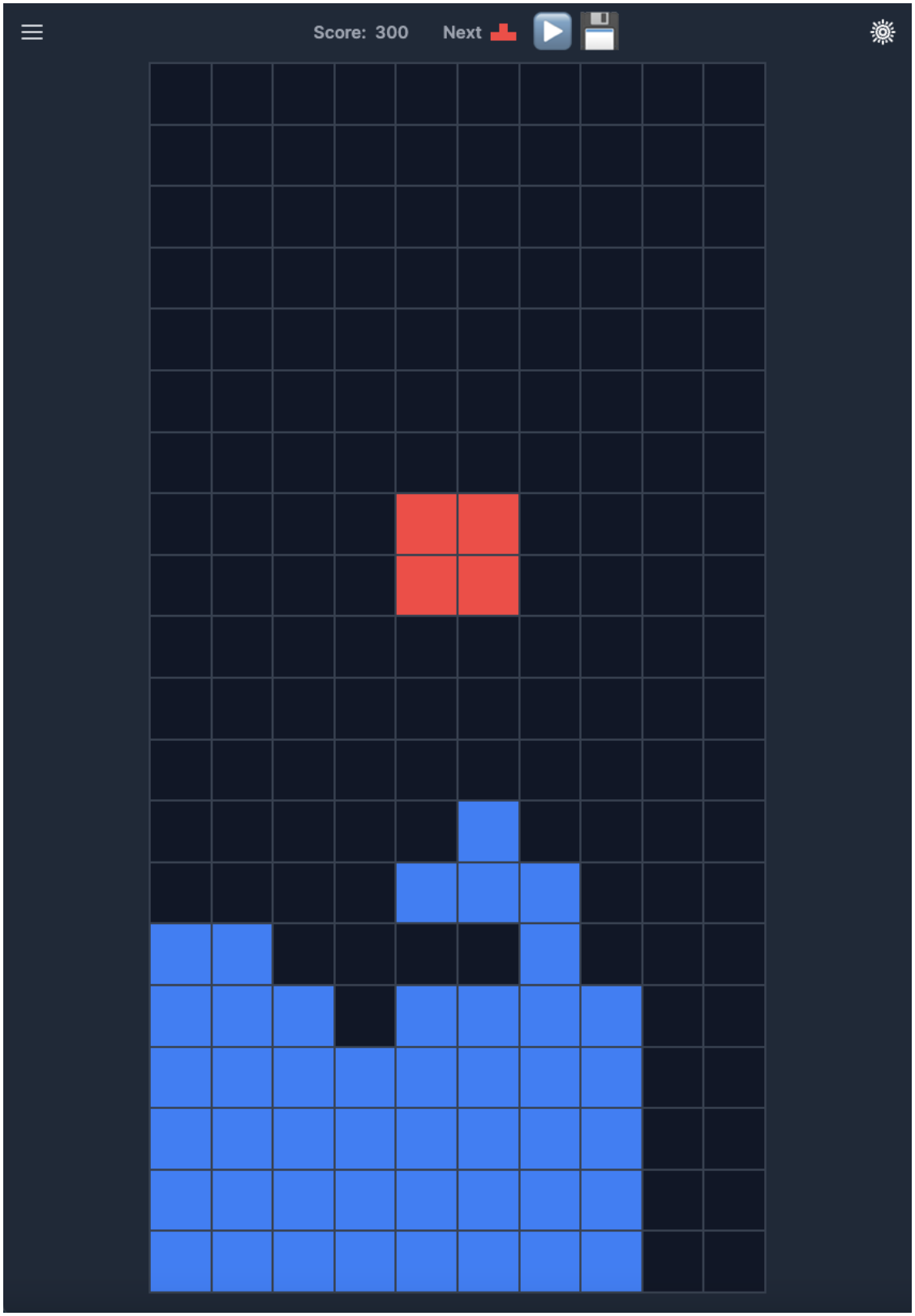


Рисунок 3 – Сторінка гри в “Tetris”

На рисунку 4 зображена розроблена схема бази даних:

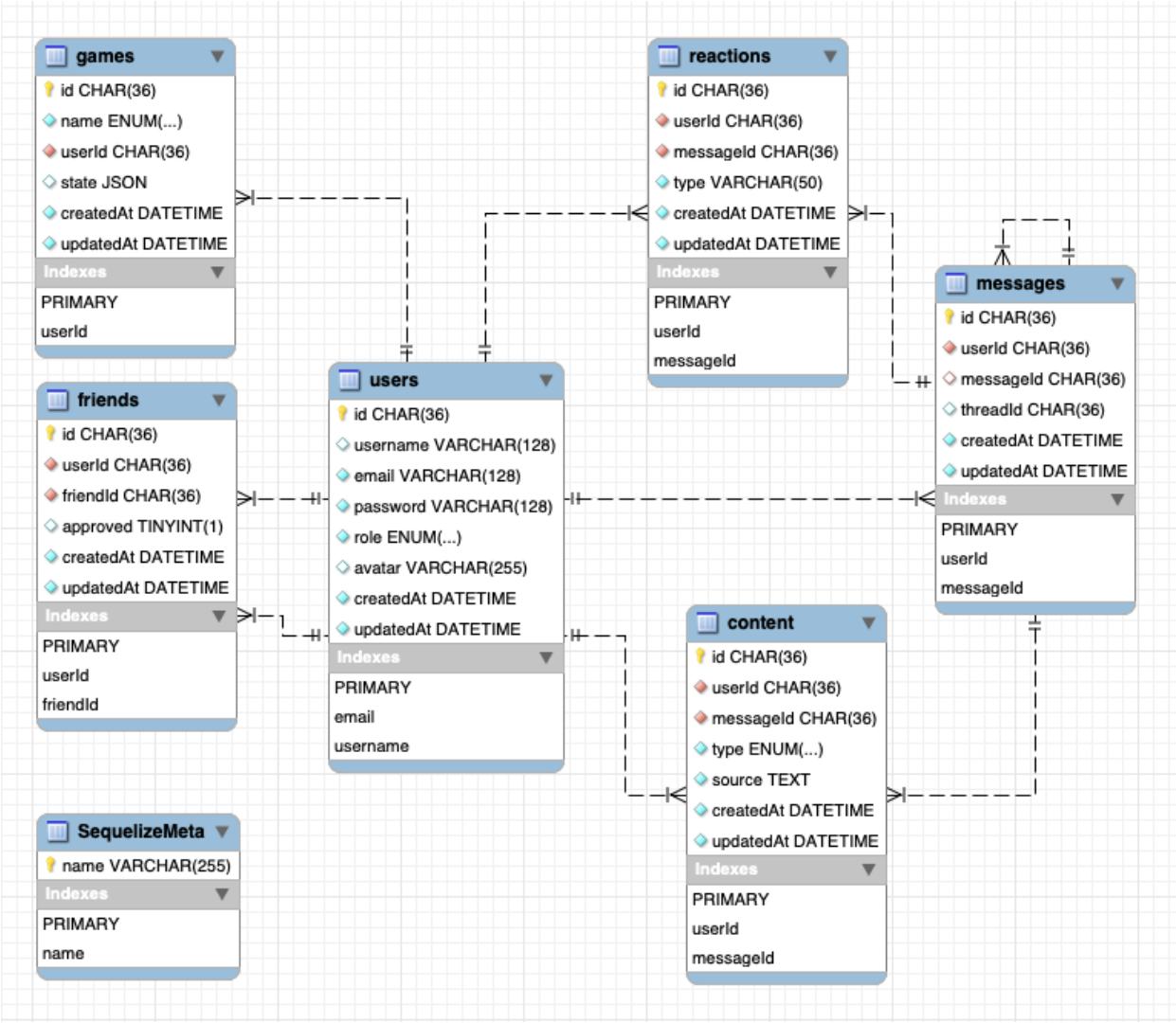


Рисунок 4 – Схема бази даних веб застосунку

На лістингу 1 представлен список залежностей проекту веб застосунку.

Лістинг 1 – Файл package.json веб застосунку

```
{
  "name": "aifans",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "db:create": "ts-node -r tsconfig-paths/register ./scripts/create-db.ts",
    "db:migrate": "npx sequelize-cli db:migrate",
    "db:migrate:undo": "npx sequelize-cli db:migrate:undo",
    "dev": "next dev",
    "build": "next build",
```

```
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@reduxjs/toolkit": "^2.2.5",
    "@tailwindcss/custom-forms": "^0.2.1",
    "autoprefixer": "^10.4.19",
    "axios": "^1.7.2",
    "bcrypt": "^5.1.1",
    "connect-redis": "^7.1.1",
    "cookies-next": "^4.2.1",
    "dotenv": "^16.4.5",
    "express-session": "^1.18.0",
    "express-validator": "^7.1.0",
    "formik": "^2.4.6",
    "helmet": "^7.1.0",
    "mysql2": "^3.10.0",
    "next": "14.2.3",
    "next-connect": "^1.0.0",
    "next-session": "^4.0.5",
    "passport": "^0.7.0",
    "passport-local": "^1.0.0",
    "react": "^18",
    "react-dom": "^18",
    "react-redux": "^9.1.2",
    "redis": "^4.6.14",
    "redux": "^5.0.1",
    "redux-saga": "^1.3.0",
    "sequelize": "^6.37.3",
    "sequelize-cli": "^6.6.2",
    "validator": "^13.12.0",
    "xss": "^1.0.15",
    "yup": "^1.4.0"
  },
  "devDependencies": {
    "@types/bcrypt": "^5.0.2",
    "@types/express-session": "^1.18.0",
    "@types/node": "^20.13.0",
    "@types/passport": "^1.0.16",
    "@types/passport-local": "^1.0.38",
    "@types/react": "^18",
    "@types/react-dom": "^18",
    "@types/yup": "^0.32.0",
    "eslint": "^8",
    "eslint-config-next": "14.2.3",
    "postcss": "^8.4.38",
    "tailwindcss": "^3.4.4",
    "ts-node": "^10.9.2",
    "tsconfig-paths": "^4.2.0",
```

```
    "typescript": "^5.4.5"
  }
}
```

На рисунку 5 зображена структура файлів та папок проекту веб застосунку:

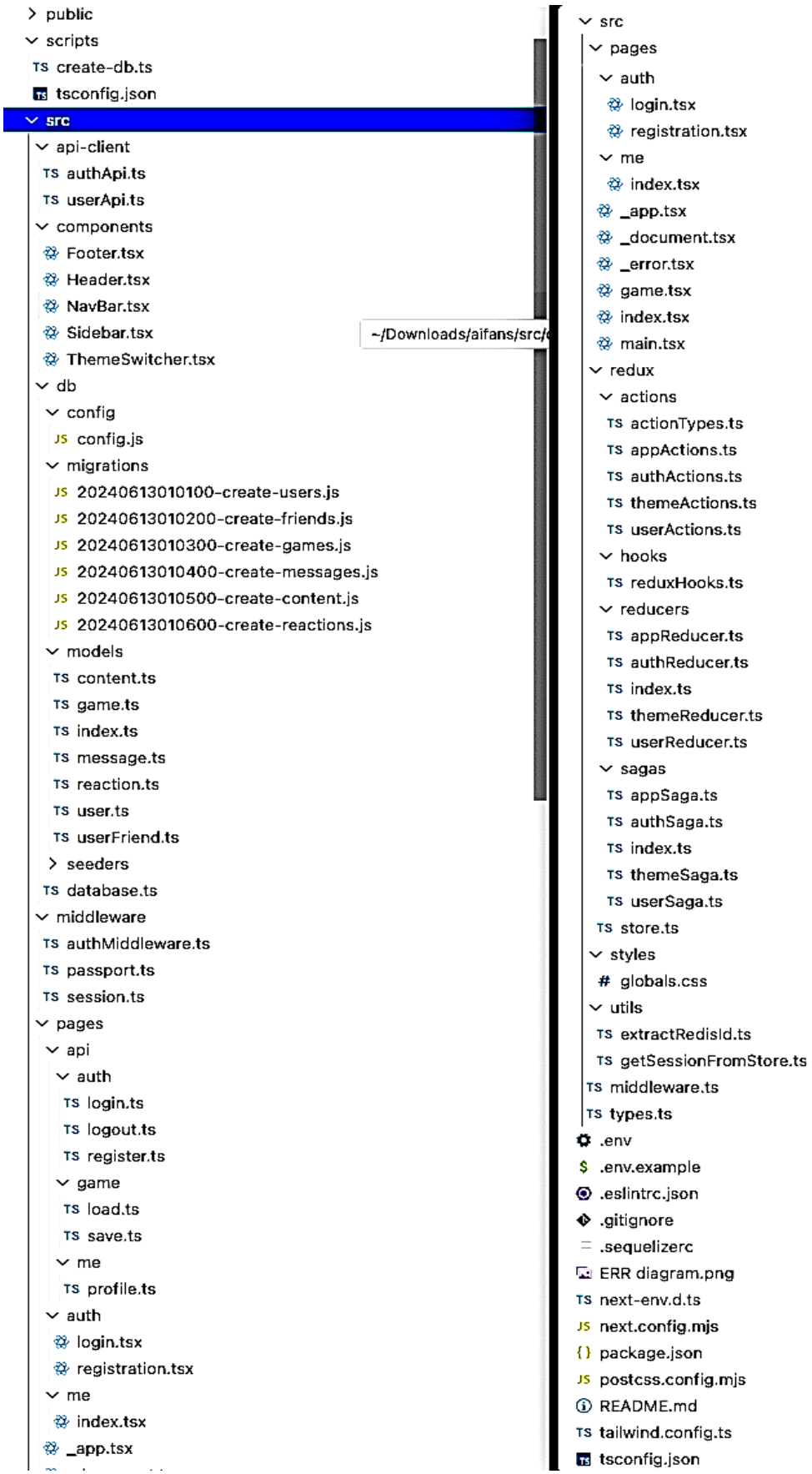


Рисунок 5 – Структура файлів та папок проекту веб застосунку

4.2 Інструменти для розробки web-застосунку

Розробка веб застосунку проводилася на комп’ютері MacBook Pro з наступними характеристиками:

1. Процессор – Intel Core i9 2,3 GHz 8 ядер.
2. Оперативна пам’ять – 16Gb 2400MHz DDR4.
3. Операційна система – Mac OS Sonoma 14.5.

Встановлене необхідне програмне забезпечення:

– Node.js v21.7.1;

– NPM 10.5.0.

В якості IDE використовувалась VSCode з плагіном GitHub Copilot.

Для розробки веб застосунку було використано такі інструменти штучного інтелекту:

1. ChatGPT – використовувався для генерації усього проекту в цілому;

2. OpenAI Assistant API – використовувалось для генерації усього проекту в цілому;

3. GitHub Copilot – використовувався для автодоповнення коду при друкуванні, для опису проблем та виправлення помилок.

Слід зазначити, що і ChatGPT, і Assistant API використовувалися з останньою на момент розробки моделлю GPT-4o. Для ChatGPT використовувався акаунт з платною підпискою (20-22 $/місяць). Для використання Assistant API була передплачена сума у розмірі до 10$.

4.3 Опис процесів розробки web-застосунку з використанням OpenAI API

Оскільки наразі ні ChatGPT ні OpenAI API не надає файли з кодом в буквальному значенні (як посилання на завантаження файлу), але надаються

code snippets з вказівкою мови програмування коду, у подальшому тексті такі code snippets для зручності будуть називатись файлами.

В ході даної роботи було розроблено декілька методик генерації коду проекту веб застосунку використовуючи ChatGPT та використовуючи OpenAI Assistant API. Далі буде приведено опис деяких з цих методик.

4.3.1 Використання ChatGPT для генерації коду проекту веб застосунку

Для генерації коду проекту веб застосунку була розроблена наступна стратегія, яка втілювалась через формулювання мети, умов та інструкцій у першому запиті (промпті):

1. Зазначення ролі моделі у контексті даного чату: “Your role is full stack Team Lead”;
2. Зазначення мети якої потрібно досягти у контексті даного чату: “...and you help me with onboarding to the company projects which are mostly with the same technologies and dependencies. So I need to develop basic project using this technologies and dependencies.”;
3. Зазначення інструкцій для моделі - “You must gave me suggested solutions step by step or file by file per message.”. Не вказавши таку інструкцію швидше за все модель видасть усі можливі файли проекту, що є не зручним для опрацювання та дуже затратним по використанню токенів у разі якщо буде потрібно вносити якісь зміни, після чого модель знову швидше за все відповість з усіма файлами проекту. Опрацювання поступово файл за файлом не тільки дає можливість сконцентруватися на перевірці кожного файлу окремо, але також ефективніше економить контекст;
4. Додавання прикладів поведінки моделі: “For example I ask you: "What should I start from?", and you must respond with only `package.json` file and the cli command.”;

5. Корисно попросити надати структуру файлів та папок проекту: “Then I ask you: "Describe me the files and folders structure", and you must only respond with detailed pseudo-graphical files and folders structure.”. Це дуже зручно для уявлення обсягу файлів проекту та подальшого контролю. Звісно в ході розробки файлова структура може змінюватись і варто періодично запитувати оновлену структуру;

6. Корисно просити додавати у кожному файлі проекту на першій стрічці коментар з відносним шляхом до цього файлу: “Then I may ask you: "Write the Landing page code", and you must respond only with the code for this page with commented relative file path at the top line.”;

7. Забороніть деякі дії моделі у разі якщо ви самі не попросите про них: “Don't respond with many files and long description if I didn't ask about it.”;

8. Додавання до чату детального опису проекту веб застосунку: “So here is the project 'gammy' description: Help me to develop the Next.js application which will have Landing page for not authenticated user, Registration page with username (unique), email (unique, not required) and password form fields, Login page with username and password form fields, Home page for authenticated users, Profile page for authenticated users with ability to change password, change username and add avatar. Need to implement Sessions and Cookies. Also (if it is not conflicting with sessions and cookies) it needs to use JWT tokens with expiration and refresh token (but don't implement JWT if it conflicting with sessions and cookies). It needs to implement caching with Redis.”;

9. Додавання до чату окремих вимог до проекту: “Other requirements and required dependencies: 1) Page Router 2) typescript 3) tailwindcss, @tailwindcss/custom-forms, postcss 4) Responsive and mobile first design 5) yup 6) formik 7) validator 8) redux, redux-saga, next-redux-wrapper, react-redux 9) passport, passport-local 10) next-session 11) express-session 12) next-connect 13) redis, connect-redis 14) sequelize, sequelize-cli 15) mysql2 16) bcrypt 17) Use `src` folder”;

10. Усі попередні пункти є складовими частинами першого промпту який варто завершити проханням спершу надати структуру файлів та папок: “So firstly I'm asking you to suggest files and folders structure”.

Послідуючі дії розробника в основному зводяться до того, щоб послідовно опрацьовувати надані файли – візуально перевіряти, копіювати code snippet, створювати відповідний файл за вказаним шляхом, вставляти код у створений файл, перевіряти в IDE наявність помилок, по можливості запускати проект і перевіряти функціонал, в разі потреби просити модель внести зміни чи виправлення. У разі самостійного виправлення помилок чи внесення змін до файлу варто відправити оновлений файл з коментарем моделі для підтримки актуальності контексту. Завершивши з опрацюванням файлу слід запросити новий файл опціонально вказавши ім’я файлу. Також у нових повідомленнях розробник може додавати нові умови та інструкції чи редагувати існуючі.

На рисунку 6 зображено перше повідомлення (промпт), що містить умови та інструкції для генерації коду проекту:

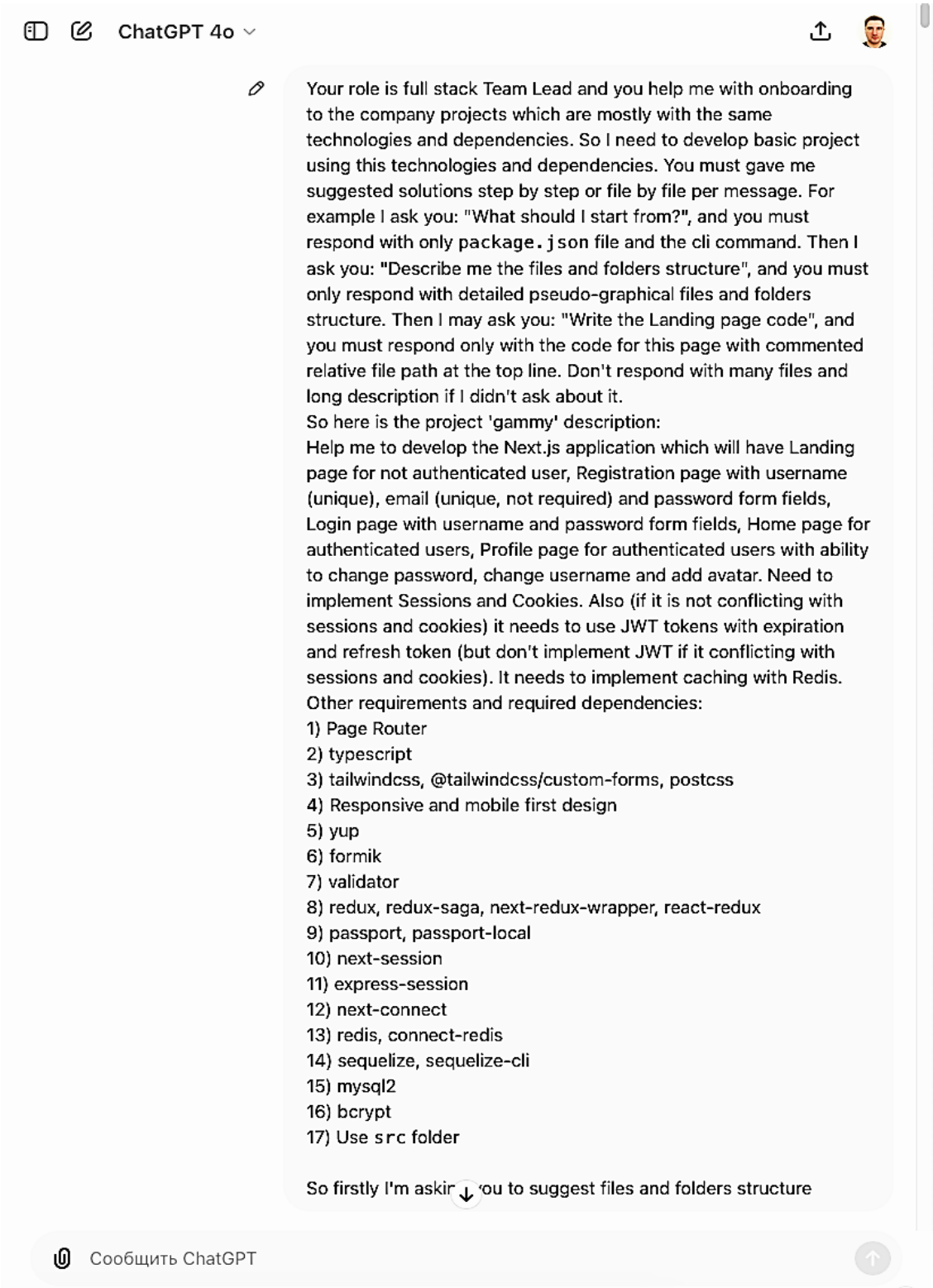


Рисунок 6 – Перше повідомлення у ChatGPT з умовами та інструкціями до генерації коду проекту

На рисунку 7 зображено першу відповідь моделі, що відображає структуру файлів та папок проекту:

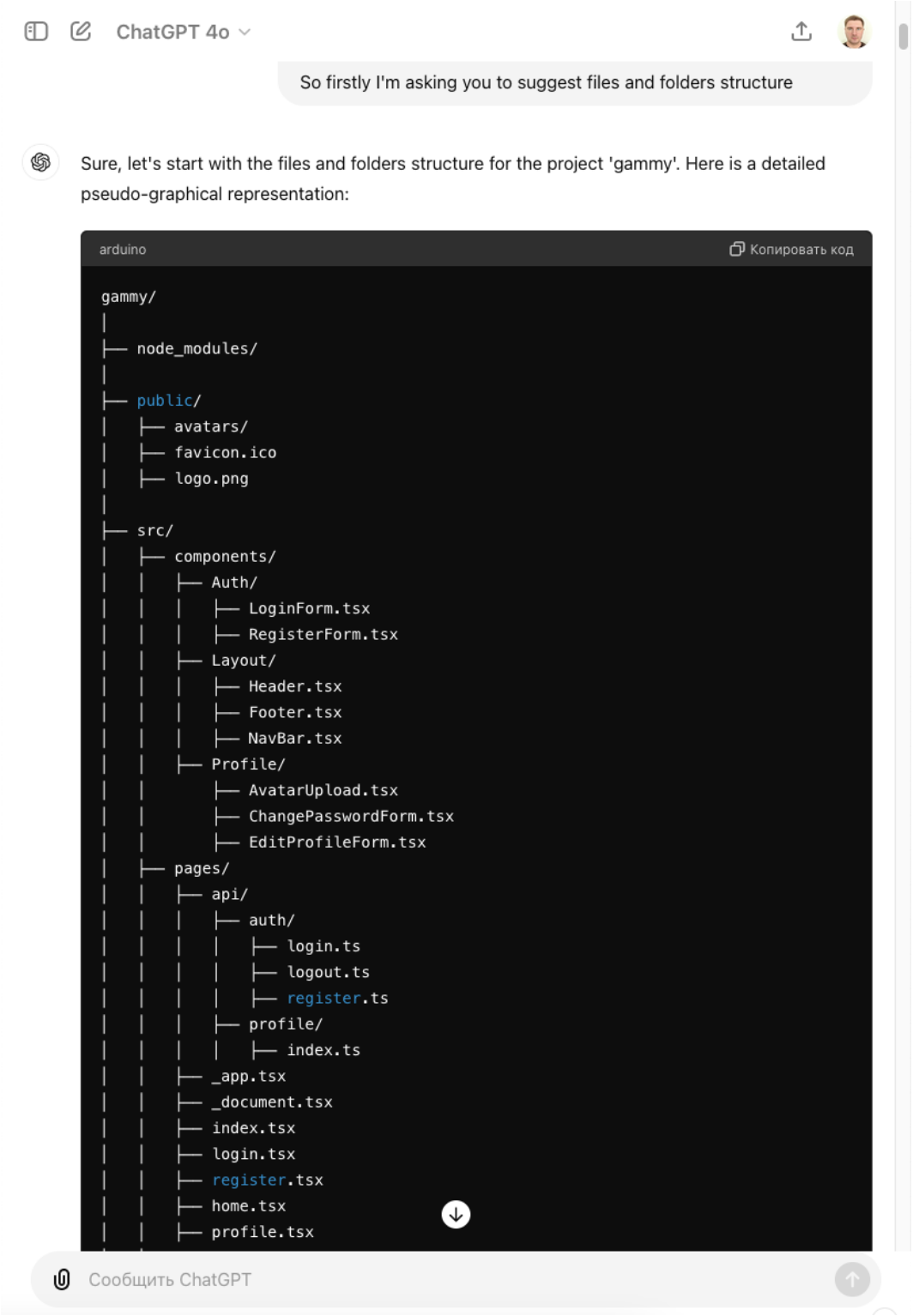


Рисунок 7 – Перша відповідь моделі з відображенням структури файлів та папок проекту

На рисунку 8 зображено повідомлення з додатковими умовами та відповідь моделі:

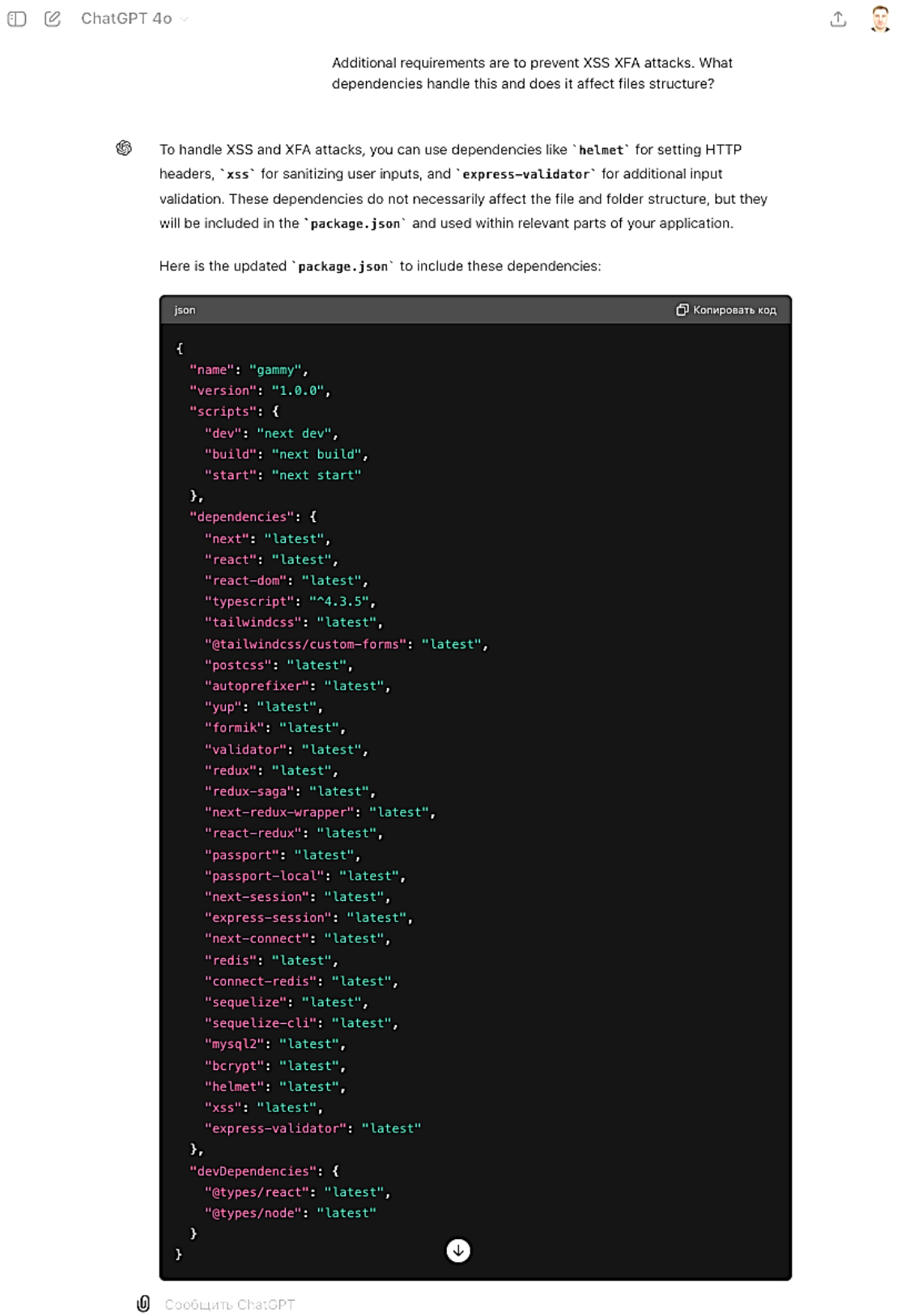


Рисунок 8 – Повідомлення з додатковими умовами та відповідь моделі

На рисунку 9 зображені повідомлення з запитом .env файлу, запитом початкової сторінки та відповідними відповідями моделі:

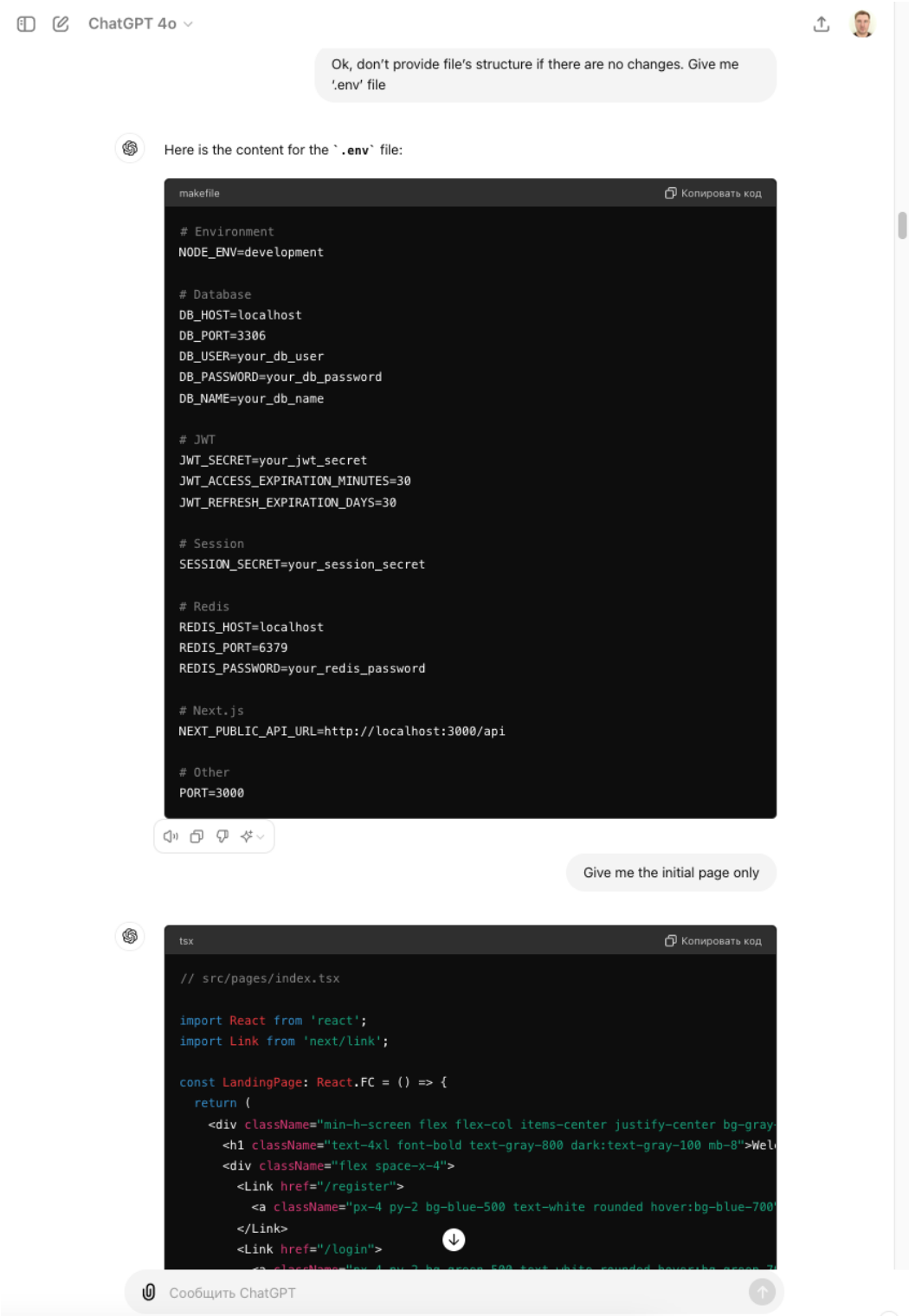


Рисунок 9 – Запити файлів та відповіді моделі

На рисунку 10 зображено повідомлення з запитанням до файлу та відповідь моделі:

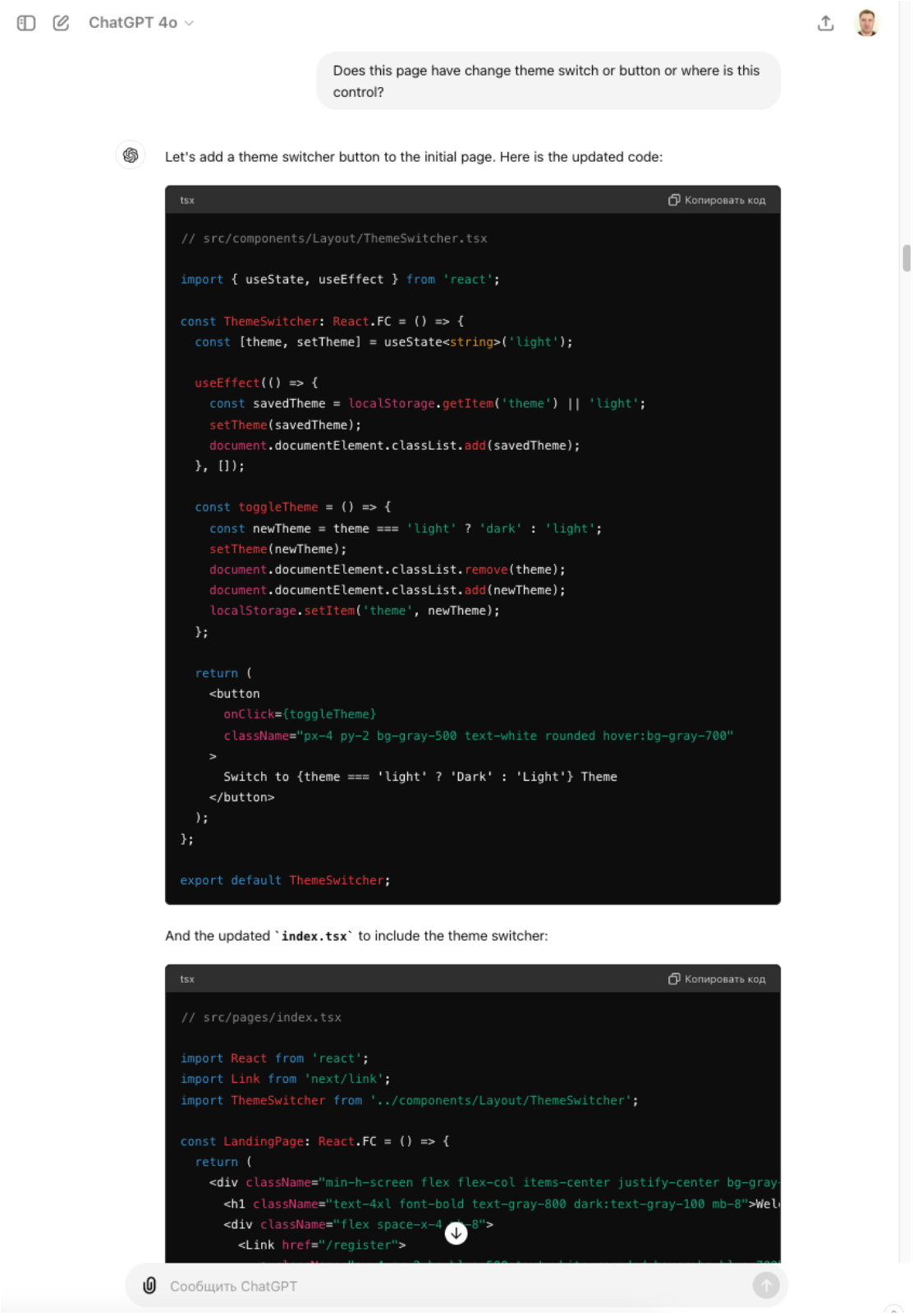


Рисунок 10 – Повідомлення з запитанням до файлу та відповідь моделі

На рисунку 11 зображено повідомлення з запитом написати скрипт встановлення залежностей та відповідь моделі:

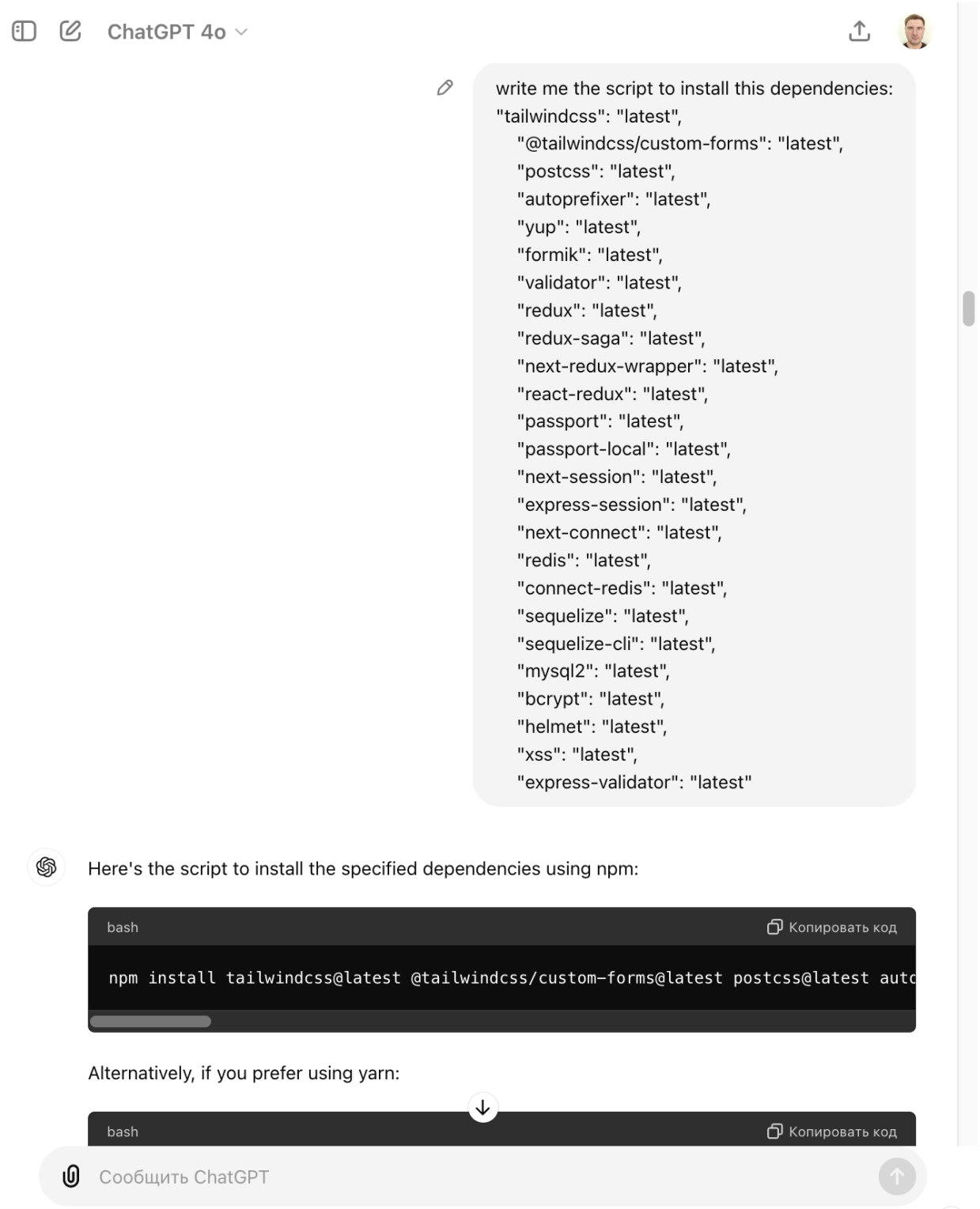


Рисунок 11 – Повідомлення з запитом написати скрипт встановлення залежностей та відповідь моделі

4.3.2 Використання OpenAI Assistant API для генерації коду проекту веб застосунку

Використання OpenAI Assistant API для генерації коду веб-застосунку є потужним інструментом, який дозволяє автоматизувати створення базового коду, оптимізувати процес розробки та зосередитися на бізнес-логіці. Ось як це можна зробити покроково:

OpenAI Assistant API (наприклад, Chat Completion API) дозволяє взаємодіяти з мовними моделями GPT для виконання різних завдань, включаючи генерацію коду. Ви можете використовувати його для створення компонентів веб-застосунку, таких як:

- Backend (серверна частина).
- Frontend (інтерфейс користувача).
- Бази даних.
- API-інтерфейси.
- Тестування.

2. Підготовка до роботи

2.1. Отримайте доступ до OpenAI API

1. Зареєструйтесь на OpenAI Platform.
2. Отримайте API-ключ у розділі API Keys.

2.2. Встановіть SDK або бібліотеку

Для інтеграції API у ваш проект використовуйте офіційні бібліотеки:

- Python: openai
- Node.js: openai

Встановлення:

- Python:

bash

pip install openai

- Node.js:

bash

npm install openai

3. Генерація коду для веб-застосунку

3.1. Архітектура проекту

Перед генерацією коду визначте архітектуру вашого веб-застосунку:

- Frontend: HTML, CSS, JavaScript (або фреймворки, як React, Vue, Angular).
- Backend: Node.js (Express), Python (Flask/Django), Ruby, PHP тощо.
- База даних: MySQL, PostgreSQL, MongoDB.

3.2. Приклад використання OpenAI API для генерації коду

Python: Генерація backend-коду

Ось приклад, як за допомогою OpenAI API згенерувати код для backend на Flask:

Лістинг 2 – Генерація коду OpenAI API для backend на Flask

```
python
import openai

# Ініціалізація OpenAI API
openai.api_key = "YOUR_API_KEY"

# Запит до API для генерації коду
response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[
        {"role": "system", "content": "Ти є досвідченим розробником веб-застосунків."},
        {"role": "user", "content": "Згенеруй backend-код для Flask, який реалізує API для управління списком завдань (CRUD)."}
    ],
    temperature=0.7
)

# Виведення згенерованого коду
print(response['choices'][0]['message']['content'])
```

Лістинг 3 – Результат (приклад згенерованого коду):

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

app = Flask(__name__)

# Список завдань
tasks = []

# Отримати всі завдання
@app.route('/tasks', methods=['GET'])
def get_tasks():
    return jsonify(tasks)

# Додати нове завдання
@app.route('/tasks', methods=['POST'])
def add_task():
    data = request.get_json()
    task = {"id": len(tasks) + 1, "title": data['title'],
"completed": False}
    tasks.append(task)
    return jsonify(task), 201

# Оновити завдання
@app.route('/tasks/<int:task_id>', methods=['PUT'])
def update_task(task_id):
    data = request.get_json()
    for task in tasks:
        if task['id'] == task_id:
            task['title']      =      data.get('title',
task['title'])
            task['completed']  =  data.get('completed',
task['completed'])
            return jsonify(task)
    return jsonify({"error": "Task not found"}), 404

# Видалити завдання
@app.route('/tasks/<int:task_id>', methods=['DELETE'])
def delete_task(task_id):
    global tasks
    tasks = [task for task in tasks if task['id'] !=
task_id]
    return jsonify({"message": "Task deleted"})

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

Лістинг 4 – Генерація frontend-коду для React-компонента:

```
javascript
const { Configuration, OpenAIApi } = require("openai");

const configuration = new Configuration({
  apiKey: "YOUR_API_KEY",
});
const openai = new OpenAIApi(configuration);

async function generateCode() {
  const response = await openai.createChatCompletion({
    model: "gpt-4",
    messages: [
      { role: "system", content: "Ти є досвідченим frontend-розробником." },
      { role: "user", content: "Згенеруй React-компонент для списку завдань із можливістю додавання та видалення завдань." }
    ],
    temperature: 0.7,
  });


console.log(response.data.choices[0].message.content);
}

generateCode();
```

Лістинг 5 – Результат (приклад згенерованого коду):

```
javascript
import React, { useState } from 'react';

function TaskList() {
  const [tasks, setTasks] = useState([]);
  const [task, setTask] = useState('');

  const addTask = () => {
    if (task.trim()) {
      setTasks([...tasks, task]);
      setTask('');
    }
  };

  const deleteTask = (index) => {
    const newTasks = tasks.filter((_, i) => i !== index);
    setTasks(newTasks);
  };
```

```
    return (
      <div>
        <h1>Task List</h1>
        <input
          type="text"
          value={task}
          onChange={(e) => setTask(e.target.value)}
          placeholder="Enter a task"
        />
        <button onClick={addTask}>Add Task</button>
        <ul>
          {tasks.map((t, index) => (
            <li key={index}>
              {t}       <button       onClick={()       =>
deleteTask(index)}>Delete</button>
            </li>
          ))}
        </ul>
      </div>
    );
  }

  export default TaskList;
```

3.3. Генерація бази даних

Ви також можете використовувати OpenAI API для створення SQL-запитів або схем бази даних.

Лістинг 6 – Генерація запиту до бази даних

```
  python
  response = openai.ChatCompletion.create(
      model="gpt-4",
      messages=[
          {"role": "system", "content": "Ти є експертом з
баз даних."},
          {"role": "user", "content": "Згенеруй SQL-схему
для таблиці 'tasks' з полями id, title, completed."}
      ]
  )
  print(response['choices'][0]['message']['content'])


  Resalt:

  sql
  CREATE TABLE tasks (
      id SERIAL PRIMARY KEY,
```

```
    title VARCHAR(255) NOT NULL,
    completed BOOLEAN DEFAULT FALSE
);
```

4.3.3 Використання Chat Completion API для генерації коду проекту веб застосунку

Покрокова інструкція, як інтегрувати Chat Completion API у Web-додаток:

1. Отримайте API-ключ

- Зареєструйтесь на OpenAI Platform.
- Увійдіть у свій обліковий запис і створіть API-ключ у розділі API Keys.
- Збережіть ключ у безпечному місці (він знадобиться для запитів).

---

2. Виберіть технології для веб-додатку

Для інтеграції API можна використовувати будь-який стек веб-розробки. Наприклад:

- **Frontend**: HTML, CSS, JavaScript (або фреймворки, як React, Vue, Angular).
- **Backend**: Node.js, Python (Flask/Django), PHP або будь-який інший серверний фреймворк.

---

3. Налаштуйте Backend

API-запити до OpenAI повинні виконуватись із серверної частини (backend), щоб захистити ваш API-ключ. Ось приклад реалізації на Node.js.

Встановіть бібліотеку OpenAI:

npm install openai

Лістнг 7 –– Приклад коду для Node.js

```
const express = require('express');
const bodyParser = require('body-parser');
const { Configuration, OpenAIApi } = require('openai');
const app = express();
```

```
const port = 3000;
// Налаштування OpenAI API
const configuration = new Configuration({
  apiKey: 'YOUR_API_KEY', // Замініть на ваш API-ключ
});
const openai = new OpenAIApi(configuration);

// Middleware для обробки JSON
app.use(bodyParser.json());

// Маршрут для обробки запитів до Chat Completion API
app.post('/chat', async (req, res) => {
  const { messages } = req.body; // Отримуємо
повідомлення з frontend

  try {
    const response = await openai.createChatCompletion({
      model: 'gpt-4', // Або gpt-3.5-turbo
      messages: messages,
      temperature: 0.7,
    });

    res.json({
      reply: response.data.choices[0].message.content,
// Відповідь моделі
    });
  } catch (error) {
    console.error(error);
    res.status(500).send('Щось пішло не так!');
  }
});

// Запуск сервера
app.listen(port, () => {
  console.log(`Сервер запущено на
http://localhost:${port}`);
});
```

4. **Налаштуйте Frontend**

Frontend відповідає за інтерфейс користувача. Ви можете використовувати простий HTML/JavaScript або фреймворки, як React чи Vue.

Лістинг 8 – Простий приклад на HTML + JavaScript:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <title>Chatbot</title>
      <style>
        body { font-family: Arial, sans-serif; margin:
20px; }
        #chat { max-width: 600px; margin: auto; }
        .message { margin: 10px 0; }
        .user { text-align: right; color: blue; }
        .bot { text-align: left; color: green; }
      </style>
    </head>
    <body>
      <div id="chat">
        <div id="messages"></div>
        <input type="text" id="userInput"
placeholder="Напишіть повідомлення..." />
        <button onclick="sendMessage()">Відправити</button>
      </div>

      <script>
        async function sendMessage() {
          const userInput =
document.getElementById('userInput').value;
          const messagesDiv =
document.getElementById('messages');

          // Додати повідомлення користувача до чату
          messagesDiv.innerHTML += `<div class="message
user">${userInput}</div>`;
          document.getElementById('userInput').value = '';

          // Надіслати запит до backend
          const response = await fetch('/chat', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
              messages: [
                { role: 'system', content: 'Ти є корисним
асистентом.' },
                { role: 'user', content: userInput }
              ]
            }),
          });

          const data = await response.json();

          // Додати відповідь бота до чату
```

```
          messagesDiv.innerHTML += `<div class="message
bot">${data.reply}</div>`;
        }
      </script>
    </body>
    </html>
```

5. Запустити додаток

1. Запустіть сервер backend:

   node server.js
2. Відкрийте HTML-файл у браузері або налаштуйте сервер для обслуговування frontend.
3. Введіть повідомлення у поле вводу, і чат-бот відповість

6. Оптимізація та розширення

- Додайте історію діалогу: Зберігайте всі повідомлення між користувачем і ботом у масиві messages, щоб модель враховувала контекст.
- Додайте авторизацію: Захистіть API-ключ, використовуючи токени доступу або інші методи аутентифікації.
- Дизайн: Використовуйте CSS-фреймворки (наприклад, TailwindCSS або Bootstrap) для покращення інтерфейсу.
- Деплой: Розгорніть додаток на хостингу (наприклад, Vercel, Netlify для frontend і Render, Heroku для backend).

  Щоб зберігати історію діалогу, модифікуйте frontend:

Лістинг 9 – Приклад збереження контексту

```
    let chatHistory = [
      { role: 'system', content: 'Ти є корисним
асистентом.' }
    ];

    async function sendMessage() {
      const userInput =
document.getElementById('userInput').value;
      const messagesDiv =
document.getElementById('messages');
```

```
      // Додати повідомлення користувача до історії
      chatHistory.push({ role: 'user', content:
userInput });
      messagesDiv.innerHTML += `<div class="message
user">${userInput}</div>`;
      document.getElementById('userInput').value = '';

      // Надіслати запит до backend
      const response = await fetch('/chat', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ messages: chatHistory }),
      });

      const data = await response.json();

      // Додати відповідь бота до історії та чату
      chatHistory.push({ role: 'assistant', content:
data.reply });
      messagesDiv.innerHTML += `<div class="message
bot">${data.reply}</div>`;
    }
```

---

7. Додаткові можливості

- Локалізація: Додайте підтримку кількох мов.
- Кнопки швидких відповідей: Додайте кнопки для стандартних запитів.
- Інтеграція з базою даних: Зберігайте історію діалогів у базі даних (наприклад, MongoDB, PostgreSQL).

4.4 Висновки з розділу 4

1. Переваги використання OpenAI API

- Швидкість розробки: Генерація базового коду займає лічені секунди.
- Гнучкість: Можна адаптувати запити для різних мов програмування та фреймворків.
- Автоматизація: Зменшує рутинну роботу, дозволяючи зосередитися на бізнес-логіці.
- Навчання: Можливість отримати приклади коду для вивчення нових технологій.

2.Обмеження

- Якість коду: Згенерований код може потребувати оптимізації або виправлення.
- Контекст: Модель не завжди враховує специфіку вашого проекту.
- Вартість: Використання OpenAI API платне, і вартість залежить від кількості токенів.

3 Рекомендації

- Використовуйте OpenAI API для створення базового коду, але завжди перевіряйте його якість.
- Інтегруйте API у ваші інструменти розробки для автоматизації рутинних завдань.
- Використовуйте чіткі інструкції для отримання релевантного коду.

ВИСНОВКИ

1. OpenAI API дозволяє автоматизувати створення базового коду, таких як backend, frontend, бази даних, або навіть тестування. Це значно скорочує час розробки, особливо для типових завдань, таких як CRUD-операції, генерація інтерфейсів чи написання SQL-запитів.

2. API OpenAI підтримує різні мови програмування та фреймворки. Ви можете генерувати код для Python (Flask/Django), JavaScript (Node.js, React, Vue), PHP, Ruby тощо. Це робить API універсальним інструментом для розробників із різними технічними уподобаннями.

3. Розробники-початківці можуть використовувати OpenAI API для навчання та створення веб-застосунків. Генерація прикладів коду допомагає зрозуміти структуру проекту, синтаксис мови та принципи роботи з фреймворками.

4. API дозволяє автоматизувати рутинні завдання, такі як генерація REST API, створення шаблонів інтерфейсу, написання документації. Це звільняє час для вирішення складніших завдань, пов'язаних із бізнес-логікою.

5. OpenAI API може використовуватися як інструмент для швидкого прототипування. Розробники можуть створювати MVP (мінімально життєздатний продукт) за короткий час, що особливо корисно для стартапів або проектів із обмеженими ресурсами.

6. Згенерований код не завжди ідеальний. Він може містити помилки, не враховувати специфіку проекту або бути неоптимізованим. Тому важливо перевіряти та тестувати код перед використанням у продакшн-середовищі.

7. Етичні та безпекові аспекти:

- Безпека: API-ключі повинні бути захищені, а конфіденційні дані користувачів не повинні передаватися до OpenAI API.
- Етика: Важливо уникати використання API для створення шкідливого або неетичного контенту. Також слід інформувати користувачів, що відповіді або функціонал створені за допомогою штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lokesh Dudhat. OpenAI in Web App Development: Benefits, Future And Limitation. Last updated on: November 03, 2023/\. URL: https://solguruz.com/blog/openai-in-web-app-development-benefits-future (дата звернення 05.05.2024).

2. SAMRAGYI CHAMOLI. New Features You Can Integrate Into Your Web App Using OpenAI. URL: https://www.openxcell.com/blog/new-features-you-can-integrate-into-your-web-app-using-openai-api (дата звернення 05.05.2024).

3. OpenAI developer platform documentation: URL: https://platform.openai.com/docs/overview (дата звернення 05.05.2024).

4. Samragyi Chamoli. New Features You Can Integrate Into Your Web App Using OpenAI API, 2024. URL: https://www.openxcell.com/blog/new-features-you-can-integrate-into-your-web-app-using-openai-api/ (дата звернення 20.11. 2024).

5. Exploring the Capabilities of OpenAI API URL : https://decos.com/en/blog/exploring-capabilities-openai-api (дата звернення 20.11. 2024).

6. Official OpenAI API documentation. URL: https://platform.openai.com/docs/introduction (дата звернення 20.11. 2024).

7. OpenAI API GitHub repository URL : https://github.com/openai/openai-node (дата звернення 20.11. 2024).

8. OpenAI API Tutorial on Dev.to. URL: https://dev.to/aaronksaunders/getting-started-with-openai-api-using-node-js-1lge (дата звернення 20.11. 2024).

9. Medium article "Building Web Applications with OpenAI API" URL : https://medium.com/better-programming/building-web-applications-with-openai-api-a-comprehensive-guide-c1959ee4ea7e (дата звернення 20.11. 2024).

10. FreeCodeCamp tutorial. URL: https://www.freecodecamp.org/news/how-to-build-an-ai-powered-app-with-openai-api/ (дата звернення 20.11. 2024).

11. YouTube video course on OpenAI API integration. URL: https://www.youtube.com/watch?v=JJ9fkYX7q4A (дата звернення 20.11. 2024).

12. OpenAI API Best Practices. URL: https://platform.openai.com/docs/guides/best-practices (дата звернення 20.11. 2024).

13. Hashnode article on integrating OpenAI with React. URL: https://hashnode.com/post/integrating-openai-api-with-react-a-step-by-step-guide (дата звернення 20.11. 2024).

14. OpenAI API Reference. URL : https://platform.openai.com/docs/api-reference (дата звернення 20.11. 2024).

15. LogRocket blog article. URL : https://blog.logrocket.com/ultimate-guide-using-openai-api-node-js/ (дата звернення 20.11. 2024).

16. Web Development with OpenAI API on JavaScript Mastery. URL : https://www.jsmastery.pro/blog/web-development-with-openai-api (дата звернення 20.11. 2024).

17. Smashing Magazine how-to guide. URL: https://www.smashingmagazine.com/2023/02/building-web-apps-openai-api/

18. OpenAI API Examples repository. URL: https://github.com/openai/openai-cookbook (дата звернення 20.11. 2024).

19. Article on DEV Community. URL : https://dev.to/codingleo/how-to-use-openai-api-in-your-web-applications-4o5m (дата звернення 20.11. 2024).

20. Tutorial on Digital Ocean. URL: https://www.digitalocean.com/community/tutorials/how-to-use-openai-api (дата звернення 20.11. 2024).

21. Web Developer Monthly on OpenAI API. URL: https://webdevelopermonthly.com/using-openai-api-web-development (дата звернення 20.11. 2024).

22. Article on Towards Data Science. URL: https://towardsdatascience.com/implementing-openai-api-in-web-applications-8e49c3745d1e (дата звернення 20.11. 2024).

23. OpenAI API Security Best Practices. UR: https://platform.openai.com/docs/guides/security-best-practices (дата звернення 20.11. 2024).

24. Practical Guide on CSS-Tricks. URL : https://css-tricks.com/getting-started-with-the-openai-api/ (дата звернення 20.11. 2024).

25. Full Stack Open course on OpenAI API. URL: https://fullstackopen.com/en/part9/using_openai_api (дата звернення 20.11. 2024).

26. Лавров М.С., Безверхий А. І. Інноваційні можливості API OpenAI у сфері веб-розробки. Інженерні інновації та розбудова національної економіки : матеріали І Міжнародної науково-практичної конференції Інженерного навчально-наукового інституту ім. Ю. М. Потебні (09–10 травня 2024 року, м. Запоріжжя) / наук. ред. Н. Г. Метеленко ; Інженерний навчально-науковий інститут ім. Ю. М. Потебні Запорізького національного університету. – Одеса : Видавничий дім «Гельветика», 2024. С. 618-620.

27. Лавров Микола магістрант 1 курсу ІННІ ім. Ю. М. Потебні Наук. кер.: доц. Безверхий А.І. ОСОБЛИВОСТІ ВИКОРИСТАННЯ API OPENAI У СУЧАСНИХ WEB-РОЗРОБКАХ. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «МОЛОДА НАУКА-2024», Запоріжжя : Запорізький національний університет, 2024. С. 188-189.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Лавров Микола Сергійович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти se121-02@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Особливості використання API OpenAI у сучасних Web-розробках**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 29.11.2024 Лавров Микола Сергійович
(студент)

Дата 29.11.2024 Безверхий Анатолій Ігорович
(науковий керівник)