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

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-06

ДИПЛОМНИЙ ПРОЕКТ

здобувача освіти денної форми навчання

РП.06.20.000.ДП

СІЛАЄВА ІГОРЯ ГЕНАДІЙОВИЧА

м. Одеса
2023 р

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Освітня програма: «**Розробка програмного забезпечення»**

Група: **4РП-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Розробка програми для генерації повідомлень і зображень за допомогою штучного інтелекту

Проектний матеріал складається з пояснювальної записки на 65 сторінках та графічного (презентаційного) матеріалу на 11 аркушах (слайдах).

Дипломник _____ (Сілаєв І.Г.)

Керівник _____ (Кунуп Т.В.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кунуп Т.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «22» 06 2023 р. Протокол ДКК № 1

Оцінка ДКК 4 (добре)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР Беркань І.В.

"_____" _____ 2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Сілаєву Ігорю Геннадійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка програми для генерації повідомлень і зображень за допомогою штучного інтелекту

затверджена наказом по коледжу від "17" 10 2022 р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) 12.06.2023р

3. Вихідні данні до проекту (роботи)

1. Вивчення можливостей AI моделей;
2. Адаптація та налаштування мовних моделей
3. Бібліотекі TensorFlow, PyTorch або Keras

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

Вибір технологій та інструментів для реалізації програми; Розробка структури програми; Розробка технічної частини програми; Розробка бекенд частини програми; Розробка фронтенд частини програмного застосунку; Результати програмування; Економічна частина; Охорона праці

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

Скриншот функціональності програми; Скриншот структури проекту; Скриншот <template> проекту; Скриншот <script>; Скриншот вікна програми; Скриншот обробки заданних параметрів; Скриншот привітання.

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Технологічний	Кунуп Т.В.		
Економічна частина	Копайгородська Т.Г.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		

7. Дата видачі завдання _____

Керівник Кунуп Т.В. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	Вступ. Постановка мети та задач проектування	5.05.2023	Виконано
2	Розробка структура програми	7.05. 2023	Виконано
3	Адаптація та налаштування моделей	9.05.22023	Виконано
4	Розробка технічної частини	11.05.2023	Виконано
5	Вибір і опис інструментарію для реалізації програми	13.05.2023	Виконано
6	Розробка зображення	16.05.2023	Виконано
7	Розробка завершення	18.05.2023	Виконано
8	Розробка бекенд частини	20.05.2023	Виконано
9	Розробка фронтенд частини	23.05.2023	Виконано
10	Опис функціоналу програми	25.05.2023	Виконано
11	Тестування програми	27.05.2023	Виконано
12	Аналіз результатів, підготовка слайдів презентації	30.05.2023	Виконано
13	Економічні розрахунки та питання з охорони праці	3.06.2023	Виконано
14	Підготовка проекту до захисту та тестування ПП	8.06.2023	

Дипломник _____
(підпис)

Керівник _____
(підпис)

ЗМІСТ

3.4 Пожежна безпека

ВИСНОВКИ

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Додаток 1

ВСТУП

Штучний інтелект (ШІ) - це наука та технологія, що займаються розробкою інтелектуальних машин і комп'ютерних програм. Ці системи прагнуть розуміти та моделювати людський інтелект, але вони не обмежені виключно біологічно відтворюваними методами.

На 2021 рік дослідники використовували наступну класифікацію типів штучного інтелекту:

Артифіційний суперінтелект (Artificial Super Intelligence, ASI) - гіпотетичний штучний інтелект, який не лише здатен репродукувати найвищі здібності людини, але й перевершити їх. Прихильники ASI вважають, що він зможе проникнути в думки та почуття людини з метою підкорення її волі. Див. Суперрозум: наукова фантастика або реальне майбутнє штучного інтелекту?

Штучний загальний інтелект (Artificial General Intelligence, AGI) також є гіпотетичним і знаходиться на рівні розуму, нижчому за ASI. Прихильники AGI обмежені у своїх переконаннях можливістю створення машин, здатних принаймні виконувати ті самі дії, що і людина.

Слабий, або обмежений інтелект (Artificial Narrow Intelligence, ANI) виявляється у поведінці машин як слабкі натяки на розум (тому його називають слабким). Він призначений лише для виконання вузького кола визначених завдань (тому його називають вузьким). У випадку ANI неможливе жодне незалежне від людини автономне поведінка або самостійний розвиток. Системи, що використовують ANI, можуть існувати лише у тому вигляді, в якому їх створив людина і навіть теоретично не можуть вийти поза його контроль.

Інтелект (від лат. intellectus) або розум - це психологічна якість, яка включає здатність адаптуватися до нових ситуацій, вчитися і запам'ятовувати на основі досвіду, розуміти і застосовувати абстрактні концепції і використовувати свої знання для управління навколишнім середовищем. Інтелект є загальною здатністю пізнання та вирішення проблем, що поєднує всі пізнавальні здібності людини, такі як сприйняття, пам'ять, мислення та уяву.

1. ТЕХНОЛОГІЧНА ЧАСТИНА

Створення програми для генерації повідомлень і зображень за допомогою штучного інтелекту може включати декілька ключових етапів і структурних компонентів. Основна мета полягає в розробці алгоритмів та моделей, які можуть генерувати текстові повідомлення та зображення, що максимально наближаються до людського творчого процесу.

На початку 1980-х років вчені в галузі теорії обчислень, Барр та Файгенбаум, запропонували наступне визначення штучного інтелекту (ІІ):

"Штучний інтелект - це область інформатики, яка займається розробкою інтелектуальних комп'ютерних систем, що мають можливості, які традиційно пов'язуються з людським розумом, такими як розуміння мови, навчання, здатність до міркування, вирішення проблем і т.д."

Пізніше до і штучного інтелекту (ІІ) стали відносити ряд алгоритмів і програмних систем, які володіють здатністю розв'язувати завдання таким чином, як це робить людина, застосовуючи розумові процеси.

Основні властивості штучного інтелекту включають розуміння мови, навчання та здатність мислити та діяти.

Штучний інтелект - це комплекс споріднених технологій і процесів, що розвиваються якісно та стрімко. До таких технологій відносяться:

Обробка тексту природною мовою: це включає розпізнавання та розуміння текстової інформації, аналіз синтаксису і семантики мови, а також генерацію тексту з розумінням контексту.

Машинне навчання: це галузь штучного інтелекту, що вивчає алгоритми і моделі, які дозволяють комп'ютерам автоматично навчатися на основі даних і здобувати досвід, покращуючи свою продуктивність з часом.

Експертні системи: це програмні системи, які моделюють експертний досвід і знання в певній області, дозволяючи робити експертні висновки та рекомендації у відповідних ситуаціях.

Віртуальні агенти (чат-боти та віртуальні помічники): це програмні агенти, які виконують завдання комунікації з користувачами, використовуючи природну мову, і можуть відповідати на запитання, надавати інформацію та виконувати певні дії.

Системи рекомендацій: це системи, які здатні аналізувати інформацію про користувача і його поведінку, а потім робити рекомендації щодо продуктів, послуг, вмісту або дій, які можуть бути цікавими або корисними для користувача.

Ці технології і процеси в штучному інтелекті розвиваються швидко і дозволяють створювати інтелектуальні системи, які наближаються до здатностей людського розуму

1.1Загальна структура програми

Розглянемо загальну структуру програми для генерації повідомлень і зображень за допомогою штучного інтелекту. Орієнтовно програма має такі етапи розробки:

1. Збір та підготовка даних:

Збір великого обсягу текстових повідомлень і зображень з різних джерел. Перевірка і попередня обробка даних, включаючи очищення, токенізацію (розбиття на окремі слова або символи), нормалізацію тощо.

Важливо мати репрезентативний набір даних для тренування, що включає різноманітність стилів, тематик і якісних показників.

Збір та підготовка даних є важливим етапом у створенні програми для генерації повідомлень і зображень за допомогою штучного інтелекту. Нижче подано детальний опис цього процесу.

Збір даних: - визначення джерел, з яких будуть зібрані текстові повідомлення і зображення. Це можуть бути публічні датамережі, веб-скрейпінг зі сторінок, соціальні медіа, форуми або власні накопичені дані.

Розробка методів автоматичного збору даних, таких як веб-скрейпінг або використання API для отримання доступу до певних джерел.

Збір великого обсягу даних, які відповідають потребам проекту. Чим більше даних, тим більше можливостей для навчання моделі і поліпшення її результатів.

2. Вибір моделей:

Вибір відповідних моделей машинного навчання або глибокого навчання для генерації тексту і зображень. Наприклад, можна використовувати генеративні моделі, такі як глибокі нейронні мережі з архітектурою GPT (Generative Pre-trained Transformer) або VAE (Variational Autoencoder), залежно від потреб проекту. Налаштування параметрів моделей і алгоритмів для досягнення бажаного рівня генерації і творчості.

3. Тренування моделей:

Використання зібраних і підготовлених даних для тренування моделей. Застосування алгоритмів навчання, таких як зворотнє поширення помилки (backpropagation), для оптимізації параметрів моделі.

Тренування моделей на міцних обчислювальних ресурсах, таких як графічні процесори (GPU) або спеціалізовані обчислювальні кластери.

4. Валідація та покращення моделей:

Оцінка якості і творчого потенціалу згенерованих повідомлень і зображень.Використання метрик оцінки якості, таких як BLEU (Bilingual Evaluation Understudy) для тексту або SSIM (Structural Similarity Index Measure) для зображень. Покращення моделей шляхом зміни архітектури, параметрів чи додаткового тренування на нових даних.

5. Інтеграція та розгортання:

Інтеграція навчених моделей у програмну систему, що забезпечує взаємодію з користувачами або іншими системами.

Розгортання програми на веб-сервері або іншому середовищі, яке забезпечує доступ до генерації повідомлень і зображень.

Забезпечення масштабованості та ефективності роботи програми, особливо при великому обсязі запитів.

1. Перевірка і попередня обробка даних:

Очищення даних шляхом видалення небажаних символів, спеціальних символів, знаків пунктуації тощо.

2. Токенізація даних, тобто розбиття текстових повідомлень на окремі слова або символи. Це може виконуватися за допомогою простого пробілу, регулярних виразів або спеціалізованих бібліотек для обробки природної мови.

Нормалізація даних, яка включає приведення тексту до одного формату. Наприклад, приведення слів до нижнього регістру, видалення зайвих пробілів, стемінг (відсічення закінчень слів до базової форми) або лематизація (приведення слів до їх лематичної форми).

Виключення стоп-слів (незначущих слів, таких як артиклі, прийменники, сполучники), які не несуть інформаційної цінності.

Репрезентативний набір даних:

Важливо мати репрезентативний набір даних, що включає різноманітність стилів, тематик і якісних показників. Наприклад, якщо програма має генерувати тексти з різних тем, то набір даних повинен містити представників цих тем з різних джерел.

Збалансований розподіл класів або категорій даних, щоб уникнути перекосів у тренувальному процесі.

Врахування різноманітних стилів і генерування тексту або зображень, що відповідають різним вимогам, таким як стиль мистецтва, жанр, тощо.

Збір та підготовка даних є ітеративним процесом, який може вимагати внесення змін у методи збору, очищення та обробки даних, щоб досягти найкращих результатів під час тренування моделей генерації повідомлень і зображень.

Вибір відповідних моделей машинного навчання або глибокого навчання є ключовим етапом у створенні програми для генерації тексту і зображень за допомогою штучного інтелекту. Нижче наведено подробний опис цього процесу:

Аналіз потреб проекту:

Ретельний аналіз вимог і цілей проекту щодо генерації тексту і зображень.

Визначення типу даних, які потрібно генерувати (текст, зображення, або комбінація обох).

Встановлення характеристик, якостей і критеріїв, які має мати модель (наприклад, зрозумілість, творчість, реалістичність тощо).

Дослідження наявних моделей:

Вивчення наукових досліджень, літератури та існуючих робіт у сфері генерації тексту і зображень за допомогою штучного інтелекту.

Вибір відомих і успішно застосованих моделей, таких як глибокі нейронні мережі GPT (Generative Pre-trained Transformer), VAE (Variational Autoencoder), LSTM (Long Short-Term Memory), GAN (Generative Adversarial Network) і т.д.

Оцінка переваг і недоліків кожної моделі з точки зору вимог проекту, таких як якість генерації, швидкодія, ресурсоємність, можливість контролювати творчість тощо.

1.2 Адаптація та налаштування моделей.

Визначення необхідної архітектури моделі залежно від вимог проекту і типу даних, що мають бути згенеровані.

Встановлення параметрів моделі, таких як розмірності шарів, кількість прихованих одиниць, функції активації тощо.

Попереднє навчання моделі на підготовлених даних для попереднього налаштування ваг моделі.

Навчання моделі:

Використання підготовлених даних для тренування моделі.

Застосування методів навчання, таких як зворотне поширення помилки (backpropagation) для оновлення ваг моделі.

Налаштування гіперпараметрів навчання, таких як швидкість навчання (learning rate), розмір пакета (batch size), кількість епох навчання тощо, для досягнення оптимальних результатів.

Оцінка моделей:

Використання метрик оцінки якості генерації, таких як BLEU (Bilingual Evaluation Understudy) для тексту або SSIM (Structural Similarity Index Measure) для зображень.

Порівняння результатів моделей і вибір найкращої моделі, яка задовольняє вимоги проекту.

Пост-обробка результатів:

Застосування пост-обробки до згенерованого тексту або зображень для поліпшення їх якості, реалізму або відповідності вимогам проекту.

Налагодження параметрів моделі або пост-обробки для досягнення найкращих результатів.

Вибір моделей є процесом, який може вимагати експериментування та ітераційного підходу залежно від вимог і характеристик проекту. Розуміння сутності різних моделей і їх потенціалу допоможе забезпечити ефективну реалізацію програми для генерації тексту і зображень з використанням штучного інтелекту.

3.Тренування моделей є важливим етапом у розробці програми для генерації тексту і зображень за допомогою штучного інтелекту. Нижче подано детальний опис цього процесу:

Використання зібраних і підготовлених даних:

Використання набору даних, який був зібраний та підготовлений на попередньому етапі.

Розділення набору даних на тренувальну, перевірочну і тестову вибірки. Тренувальна вибірка використовується для навчання моделі, перевірочна - для налаштування гіперпараметрів та вибору найкращої моделі, а тестова - для оцінки остаточної якості моделі.

Застосування алгоритмів навчання:

Використання алгоритмів навчання, зокрема зворотнього поширення помилки (backpropagation), для оптимізації параметрів моделі.

Обчислення градієнтів, які вказують напрямок, яким необхідно змінити ваги моделі, щоб мінімізувати помилку між прогнозами моделі і дійсними значеннями.

Використання обчислювальних ресурсів:

Тренування моделей вимагає значних обчислювальних ресурсів через обробку великої кількості даних та складність алгоритмів.

Використання міцних обчислювальних ресурсів, таких як графічні процесори (GPU) або спеціалізовані обчислювальні кластери, що дозволяють прискорити процес тренування моделей.

Параметри тренування:

Встановлення гіперпараметрів навчання, таких як швидкість навчання (learning rate), кількість епох навчання, розмір пакета (batch size) тощо.

Відстеження процесу тренування, таких як збіжність моделі, зменшення помилки, збільшення якості генерації.

Оптимізація моделі:

Використання методів оптимізації моделі, таких як регуляризація, dropout, ансамблювання моделей тощо, для зменшення перенавчання та покращення загальної роботи моделі.

Тренування моделей є ітеративним процесом, який може вимагати кількох епох навчання, зміну гіперпараметрів і методів оптимізації, а також вибір кращої моделі на основі оцінки її якості і відповідності вимогам проекту.

Валідація та покращення моделей

Після тренування моделей для генерації тексту і зображень за допомогою штучного інтелекту проводиться їх оцінка та покращення. Основні кроки цього процесу включають:

Оцінка якості і творчого потенціалу згенерованих повідомлень і зображень:

Застосування згенерованих повідомлень і зображень до відповідних сценаріїв або контекстів.

Аналіз результатів генерації, включаючи зрозумілість, логічність, реалістичність, креативність та відповідність поставленим вимогам.

Використання метрик оцінки якості:

Використання метрик, таких як BLEU (Bilingual Evaluation Understudy) для тексту або SSIM (Structural Similarity Index Measure) для зображень, для кількісної оцінки якості згенерованих даних порівняно з оригінальними даними.

Порівняння метрик між різними моделями або конфігураціями моделей для вибору найкращого варіанта.

Покращення моделей:

Зміна архітектури моделей, включаючи різні шари, рекурентні або згорткові блоки, або використання пре-тренованих моделей для покращення якості генерації.

Зміна гіперпараметрів моделей, таких як розмір шарів, кількість прихованих одиниць, швидкість навчання тощо.

Додаткове тренування моделей на нових або розширених наборах даних для поліпшення їх здатності генерувати бажані повідомлення і зображення.

Цей процес вимагає експериментування та ітеративного підходу. Оцінка якості, використання метрик та покращення моделей допомагають досягти кращої якості генерації та відповідності потребам проекту.

Інтеграція та розгортання навчених моделей для генерації повідомлень і зображень в програмну систему включає наступні кроки:

Інтеграція навчених моделей:

Розробка інтерфейсу або API, який дозволяє взаємодіяти з навченими моделями.

Інтеграція моделей у програмну систему, забезпечуючи їх доступність і функціональність для користувачів або інших систем.

Розгортання програми:

Вибір середовища для розгортання програми, наприклад, веб-сервер, хмарна платформа або контейнеризація.

Налаштування необхідних залежностей та інфраструктури для розгортання програми.

Забезпечення масштабованості та ефективності:

Використання технологій, таких як горизонтальне масштабування, балансування навантаження, кешування результатів для забезпечення швидкості та ефективності роботи програми.

Оптимізація коду та алгоритмів, щоб забезпечити швидку генерацію повідомлень і зображень.

Тестування та контроль якості:

Проведення ретельного тестування програми, включаючи модульні, інтеграційні та функціональні тести, для переконання в її коректності та стабільності.

Використання методів контролю якості, які дозволяють оцінити роботу програми під різними умовами та вимогами.

Управління версіями:

Використання систем управління версіями, таких як Git, для контролю та керування версіями програмного коду та навчених моделей.

Забезпечення можливості відкату до попередніх версій або впровадження нових функцій з мінімальними ризиками.

Ці кроки допомагають інтегрувати навчені моделі в програмну систему та забезпечити їх доступність, ефективність та масштабованість. Однак, важливо враховувати безпеку даних та конфіденційність, роблячи відповідні заходи для їх захисту під час розгортання і використання програми.

1.3 Розробка техничної частини програми «Вітабот»

1.3.1 Опис API OpenAI.

Для розробки програми, заснованої на штучному інтелекті, необхідно вивчити різні моделі та АПІ (апаратний програмний інтерфейс). Найпопулярнішим і найдоступнішим виявилося АПІ (апаратний програмний інтерфейс) від компанії OpenAi (https://platform.openai.com/docs/api-reference), яка розробила і представила ChatGPT. У цьому підрозділі буде описано кінцеві точки (endpoint), які будуть використовуватися в програмі.

1.3.2 Створення завершення (Create completion)

Цей кінцевий пункт API використовується для генерації тексту на основі введеного вами початкового тексту. Ви надаєте "промпт" (початковий текст), а модель "text-davinci-003" продовжує його в найбільш ймовірний спосіб. Ви можете контролювати довжину вихідного тексту, його температуру (випадковість) та інші параметри. Простіше, ця кінцева точка поверне текст чи вірші залежно від підказки, яку клієнт відправить. Наприклад якщо надіслати підказку (prompt) " Напиши привітання до Дня Народження для мого колеги Сергія. Побажай здоров'я та благополуччя", то отримаємо у відповідь:

“ Дорогий Сергію,

Вітаю тебе з Днем народження! Бажаю тобі найщирішого здоров'я, яке буде супроводжувати тебе на кожному кроці твого життя. Нехай благополуччя завжди супроводжує тебе, принесаючи радість, успіх і задоволення у всіх твоїх починаннях. Щиро вітаю з цим особливим днем і бажаю, щоб весь твій новий рік був наповнений радістю та незабутніми моментами!”

POST: https://api.openai.com/v1/completions

Ось основні параметри, які ми встановлюємо при використанні цього кінцевого пункту:

model: Це ID моделі, яку ми хочемо використовувати. Наприклад, "text-davinci-003".

prompt: Це початковий текст, який ви хочете, щоб модель продовжила. Наприклад, "Жив-був дракон".

max_tokens: Це максимальна кількість токенів у вихідному тексті. Якщо ви встановите це значення, скажімо, 100, модель зупинить генерацію тексту після того, як вона виробить 100 токенів.

temperature: Це параметр, який контролює випадковість вихідного тексту. Вище значення, наприклад 1.0, робить вихід більш випадковим, тоді як нижче значення, наприклад 0.2, робить його більш детермінованим.

Приклад запиту на NodeJs:

```
const { Configuration, OpenAIApi } = require("openai");
const configuration = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);
const response = await openai.createCompletion({
  model: "text-davinci-003",
  prompt: " Напиши привітання до Дня Народження для мого колеги Сергія.",
  max_tokens: 100,
  temperature: 1,
});
```

Відповідь буде у такій формі:

```
{
  "id": "cmpl-uqkvlQyYK7bGYrRHQ0eXlWi7",
  "object": "text_completion",
  "created": 1589478378,
  "model": "text-davinci-003",
  "choices": [
    {
      "text": " Дорогий Сергію, \n Вітаю тебе з Днем народження! Бажаю тобі неймовірного свята, сповненого радості, сміху та незабутніх вражень. Хай твій новий рік буде наповнений успіхами і досягненнями в усіх твоїх справах ",
      "index": 0,
      "logprobs": null,
      "finish_reason": "length"
    }
  ],
}
```

1.3.3 Створення зображення (Create image).

Цей кінцевий пункт API використовується для генерації зображень за допомогою моделі DALL·E. Ми надаємо текстовий опис, а модель створює зображення, яке відповідає цьому опису. Згенеровані зображення можуть мати розмір 256×256, 512×512 або 1024×1024 пікселів. Менші розміри генеруються швидше. За допомогою параметра *n* можна запитувати 1-10 зображень за раз.

POST **https://api.openai.com/v1/images/generations**

Ось основні параметри, які ми можемо встановити при використанні цього кінцевого пункту:

prompt: текстовий опис потрібного(их) зображення(й). Максимальна довжина – 1000 символів.

n: кількість зображень для створення. Має бути від 1 до 10. За замовчуванням 1.

size: розмір згенерованих зображень. Має бути один із 256x256, 512x512 або 1024x1024. За замовчуванням 1024x1024

Приклад запиту на NodeJs:

```
const openai = new OpenAIApi(configuration);
const response = await openai.createImage({
  prompt: " біла сіамська кішка",
  n: 1,
  size: "1024x1024",
});
```

У відповідь отримуємо масив із посиланнями на згенеровані зображення:

```
{
  "created": 1589478378,
  "data": [
    {
      "url": " https://oaidalleapiprodscus.blob.core.windows.net/private/org-J1TxeLsuOXF6lItCt1RYmqqj/user-xtUCyyh8q3xU2ZmxB1dGq0S5/img-pAW3A8ErFhP5s0vTqqPlU9X3.png?st=2023-05-28T15%3A07%3A16Z&se=2023-05-28T17%3A07%3A16Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2023-05-28T09%3A30%3A19Z&ske=2023-05-29T09%3A30%3A19Z&sks=b&skv=2021-08-06&sig=lW/ZwgYrfidWdHO9udC5%2Bp0e6IpX/L/f5s3vjRNMWZ4%3D "
    }
  ]
}
```

1.3.4 Створення редагування зображення (Create image edit).

Цей кінцевий пункт API дозволяє нам редагувати вже створені зображення. Ви можете вказати, які зміни ви хочете внести, і модель виконає ці зміни. Наприклад, ви можете змінити колір об'єкта на зображенні або додати новий елемент на зображення.

Можливо редагувати та розширювати зображення, завантажуючи маску. Прозорі області маски вказують, де потрібно відредагувати зображення, а підказка має описувати нове зображення в цій області.

Завантажене зображення та маска мають бути квадратними зображеннями PNG розміром менше 4 МБ, а також мають однакові розміри.

POST https://api.openai.com/v1/images/edits

Ось основні параметри, які ми можемо встановити при використанні цього кінцевого пункту:

image: зображення для редагування. Має бути дійсний файл у форматі PNG розміром менше 4 МБ і квадратний. Якщо маска не вказана, зображення повинно мати прозорість, яка буде використовуватися як маска.

mask: додаткове зображення, повністю прозорі області якого (наприклад, де альфа дорівнює нулю) вказують, де зображення слід редагувати. Має бути дійсний файл у форматі PNG, розміром менше 4 МБ і такий самий розмір, як зображення.

prompt: текстовий опис потрібного(их) зображення(й). Максимальна довжина – 1000 символів.

n: кількість зображень для створення. Має бути від 1 до 10. За замовчуванням 1

size: розмір згенерованих зображень. Має бути один із 256x256, 512x512 або 1024x1024. За замовчуванням 1024x1024.

Приклад запиту на NodeJs:

```
const openai = new OpenAIApi(configuration);
const response = await openai.createImageEdit(
  fs.createReadStream("otter.png"),
  " біла сіамська кішка ",
  fs.createReadStream("mask.png"),
  2,
  "1024x1024"
```

);

У відповідь отримуємо масив із посиланнями на відредаговані зображення:

```
{
  "created": 1589478378,
  "data": [
    {
      "url": " https://oaidalleapiprodscus.blob.core.windows.net/private/..."
    }
  ]
}
```

1.4 Мета та опис функціональності програми «Вітабот»

Вітабот (VitaBot) - поєднання "Вітає" (привітання) і "Бот" (бот), що підкреслює AI-аспект програми.

Вітабот має на меті надати користувачам зручний і креативний спосіб висловити свої добрі побажання чи подяку за допомогою ШІ.

Основна мета Вітабот - запропонувати зручну платформу для створення індивідуальних привітань і супровідних фотографій для різноманітних випадків, таких як дні народження, ювілеї та інші особливі події.

Додаток використовуватиме алгоритми штучного інтелекту для аналізу введених користувачем даних і створення персоналізованого контенту із зображеннями та текстом на основі події, одержувача та бажаного тону, використовуючи OpenAi GPT для тексту та DALL·E для генерації зображень.

Вибір мови: українська та англійська

Введення імені отримувача: Введіть ім'я людини, яку ви бажаєте привітати.

Вибір події: Оберіть тему привітання (день народження, річниця тощо).

Вибір стилю привітання: Виберіть між неофіційним, офіційним або нейтральним стилем.

Вибір типу привітання: Оберіть тип привітання (текст, тост, рима, пісня).

Підказка для тексту привітання: Надайте підказку для написання привітання штучним інтелектом.

Генерація фото на основі підказок.

Редагування фото на основі підказок

Модальне вікно зі згенерованим результатом: Перегляд згенерованого привітання в модальному вікні.

Відображення зображення та тексту: Перегляд згенерованого зображення та супроводжуючого тексту.

Функція редагування: Редагування згенерованого привітання за бажанням.

Опція повторної генерації: Генерація нової версії привітання.

Функція завантаження: Завантаження згенерованого привітання для подальшого використання.

Інтеграція з соціальними мережами: Поділ привітання безпосередньо у соціальних мережах за допомогою наданих посилань.

1.5 Опис роботи програми

Користувач обирає мову, ім'я людини, яку хоче привітати, тему привітання (день народження, ювілей, стиль привітання (неформальний, офіційний, нейтральний), тип (текст, тост, віршик, пісня), текст-підказку для привітання та фото з сгенерованої картинки. Після відправки ми бачимо модальне вікно з результатом генерації ШІ, що містить зображення, текст, деякі сервіси (редагувати, регенерувати, завантажити) та посилання на соціальну мережу.

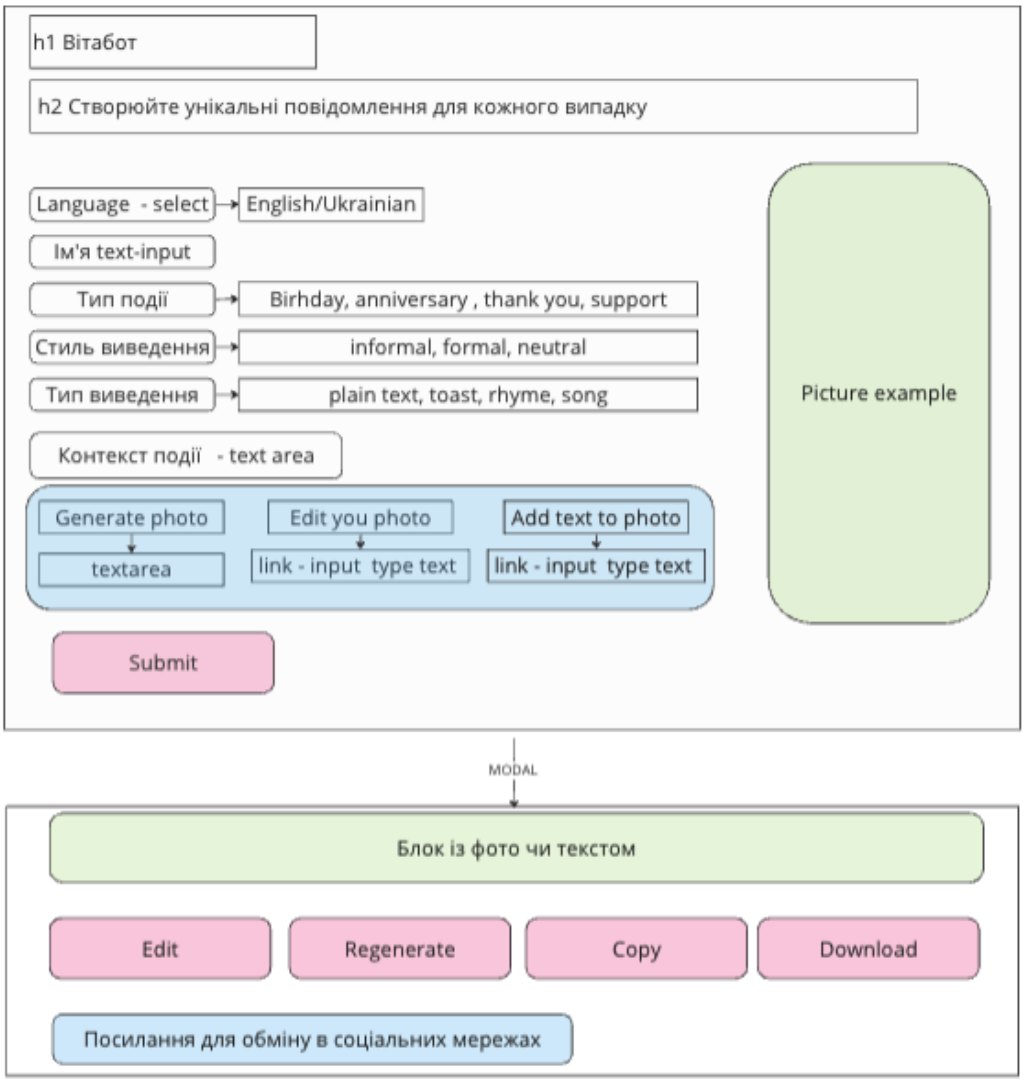

Рисунок 1. Скриншот функціональності програми

1.6 Технології, які використовуються в розробці програми

1.6.1 Опис технології розробки фронтенд частини

Для розробки фронтенд частини використовуватиметься Vue 3 та Vite. Vue 3 та Vite - це сучасні технології, що використовуються для розробки веб-додатків.

1. Vue 3: Vue 3 - це остання версія популярного фреймворку для розробки веб-додатків, який називається Vue.js. Vue 3 пропонує ряд важливих поліпшень, зокрема:

- Новий API Composition, який дозволяє розбити логіку додатка на менші, легкі для розуміння частини.
- Поліпшену продуктивність, завдяки реалізації дерева рендерування на основі статичного аналізу.
- Підтримку TypeScript з коробки.
- Кращу інтеграцію з іншими інструментами і фреймворками, такими як Vuex та Vue Router.

2. Vite: Vite (що у французькій мові означає "швидко") - це новий інструмент для розробки веб-додатків, створений Еваном Ю (творцем Vue.js). Vite пропонує швидку та ефективну сбірку вихідного коду. Основні особливості Vite включають:
    - Швидку "гарячу" заміну модулів (HMR), що дозволяє розробникам бачити зміни в реальному часі без перезавантаження сторінки.
    - Підтримку TypeScript, JSX, CSS, JSON та інших модулів з коробки.
    - Підтримку Vue 3, React, Preact, LitElement і інших сучасних фреймворків.
    - Зручність налаштування і перевірки додатку в режимі розробки і продакшена.

Разом Vue 3 та Vite створюють потужний та простий стек для розробки веб-додатків, що використовуються розробниками по всьому світу.

1.6.2 Опис технології розробки бекенд частини

Бекенд частина буде зроблена на платформі NodeJs та фрейфорке Express.

Node.js - це відкрита платформа, створена на двигуні JavaScript V8 від Google Chrome. Вона дозволяє виконувати JavaScript-код на сервері, що робить JS не тільки мовою для клієнтської (браузерної) частини, але й для серверної.

Особливості Node.js включають:

- Асинхронне та неблокуюче введення/виведення (I/O): Node.js використовує асинхронні API, що дозволяє обробляти багато запитів одночасно, без блокування основного потоку.
- Подієво-орієнтована модель: Node.js використовує подієво-орієнтовану модель, що допомагає управляти асинхронними операціями.
- npm (Node Package Manager): Це потужний менеджер пакетів, що дозволяє встановлювати та управляти бібліотеками JavaScript, що використовуються в проекті.

Express - це легкий та гнучкий веб-фреймворк для Node.js. Він робить розробку веб-додатків на Node.js більш простою та організованою.

Особливості Express включають:

- Маршрутизація: Express пропонує просту та інтуїтивно зрозумілу систему маршрутів, яка дозволяє визначати, як ваш додаток відповідає на різні HTTP запити на конкретні URL.
- Middleware: Express дозволяє використовувати middleware, функції, які мають доступ до об'єкту запиту (req), об'єкту відповіді (res) та наступної функції обробки в системі обробки запитів. Middleware можуть виконувати такі завдання, як перевірка користувача, логування та декодування запитів.
- Шаблонізація: Express підтримує різні двигуни шаблонів, які дозволяють генерувати HTML насервері перед відправкою на клієнт.

В комбінації Node.js та Express створюють потужний стек технологій для розробки серверної частини веб-додатків. Node.js надає можливість виконувати JavaScript на сервері, а Express додає ряд корисних функцій, які полегшують розробку веб-додатків.

Для розробки програми використовуватиметься редактор коду VScode.

Visual Studio Code - це безкоштовний редактор коду від Microsoft, який підтримує багато мов програмування та має велику кількість розширень. Він працює на всіх основних операційних системах: Windows, macOS та Linux.

Особливості VSCode включають:

- Підтримка багатьох мов: VSCode підтримує велику кількість мов програмування з коробки, включаючи JavaScript, TypeScript, Python, C#, PHP, Go, Java та інші.
- Розширення: VSCode має велику кількість розширень, які дозволяють налаштувати редактор під конкретні потреби. Це можуть бути розширення для підсвічування синтаксису, форматування коду, інтеграції з Git та іншими інструментами.
- Інтегрована термінал: VSCode має вбудований термінал, що дозволяє виконувати команди без виходу з редактора.
- Інтелектуальне завершення коду: VSCode використовує IntelliSense для автоматичного завершення коду, що полегшує написання коду та зменшує шанси на помилки.
- Підтримка Git: VSCode має вбудовану підтримку Git, що дозволяє виконувати команди Git безпосередньо з редактора коду.

Для слідкування за змінами в програмному коді в процесі розробки дуже зручно використовувати программу Git.

Git - це система контролю версій, яка допомагає командам ефективно керувати змінами в коді. Вона була створена Лінусом Торвальдсом, творцем Linux.

Особливості Git включають:

- Розподілена архітектура: Кожен розробник має свою власну локальну копію репозиторію, що дозволяє працювати незалежно і відправляти зміни, коли вони готові.
- Гнучкість у використанні гілок: Git дозволяє легко створювати, зливати та переключатися між гілками коду. Це дозволяє розробникам працювати над різними функціями або виправленнями помилок одночасно.
- Журнал змін: Git зберігає детальний журнал змін, який дозволяє відслідковувати, хто зробив зміни, коли вони були зроблені, і що саме було змінено.
- Захист від втрати даних: Завдяки своїй розподіленій архітектурі, Git дозволяє відновити втрачені дані з інших копій репозиторію.

- Інтеграція з робочими процесами: Git може бути інтегрований з різними системами стеження за помилками, неперервною інтеграцією/доставкою (CI/CD) та іншими інструментами для розробки програмного забезпечення.

Разом VSCode та Git створюють потужний набір інструментів для розробки програмного забезпечення, який дозволяє розробникам ефективно створювати, тестувати та управляти кодом.

Також використаний пакетний менеджер NPM

NPM (Node Package Manager) - це пакетний менеджер для Node.js, який використовується для керування залежностями проекту. За його допомогою можна встановлювати, оновлювати та видаляти пакети, що є частиною вашого проекту. Пакети - це модулі коду, які містять функціональність, яку можна використовувати в вашому проекті.

NPM має власний онлайн-репозиторій пакетів, які можна використовувати в вашому проекті. Це означає, що вам не потрібно писати весь код від початку - ви можете використовувати пакети, які створили інші розробники.

Деякі команди NPM, які нам знадобляться:

- npm install: встановлює всі залежності, вказані в файлі package.json.
- npm install <package-name>: встановлює конкретний пакет.
- npm uninstall <package-name>: видаляє пакет з проекту.
- npm update: оновлює всі пакети до останніх версій, відповідно до вказаних версій в package.json.
- npm run <script-name>: виконує скрипт, вказаний в package.json під іменем <script-name>.

Для використання NPM потрібно встановити Node.js, оскільки NPM входить до його стандартного пакету. Після встановлення Node.js ми використовуємо NPM в командному рядку для керування залежностями вашого проекту.

Node.js - це оточення виконання JavaScript, яке працює на сервері. Воно було розроблене на двигуні V8 від Google, який використовується в браузері Chrome.

Node.js розширює можливості JavaScript за межі браузера, дозволяючи виконувати скрипти на сервері або в системі користувача.

1.7 Розробка програми «Вітабот»

1.7.1 Розробка Фронтенду

Щоб почати новий проект на Vue 3 з використанням Vite, спершу потрібно встановити Node.js і npm. Після цього ви можете використовувати командний рядок для створення нового проекту.

Ось основні кроки:

1. Встановлюємо Vite глобально на нашому комп'ютері за допомогою npm:
   npm install -g create-vite
2. Створюємо новий проект Vue 3 за допомогою Vite:
   create-vite my-vue-project --template vue
   Ця команда створить нову директорію з ім'ям нашого проекту, яка міститиме всі необхідні файли для проекту Vue 3.
3. Переходимо до директорії нашого проекту:
   cd my-vue-project
4. Встановлюємо всі залежності проекту:
   npm install
5. Запускаємо сервер розробки:
   npm run dev

Після цього ваш проект Vue 3 з Vite повинен бути доступний в браузері за адресою `http://localhost: 5173 `.

Звернемо увагу, на те що це базова конфігурація. В залежності від наших потреб ми, можливо, захочемо додати інші плагіни або бібліотеки.

Структура проекту представлена на рисунку :

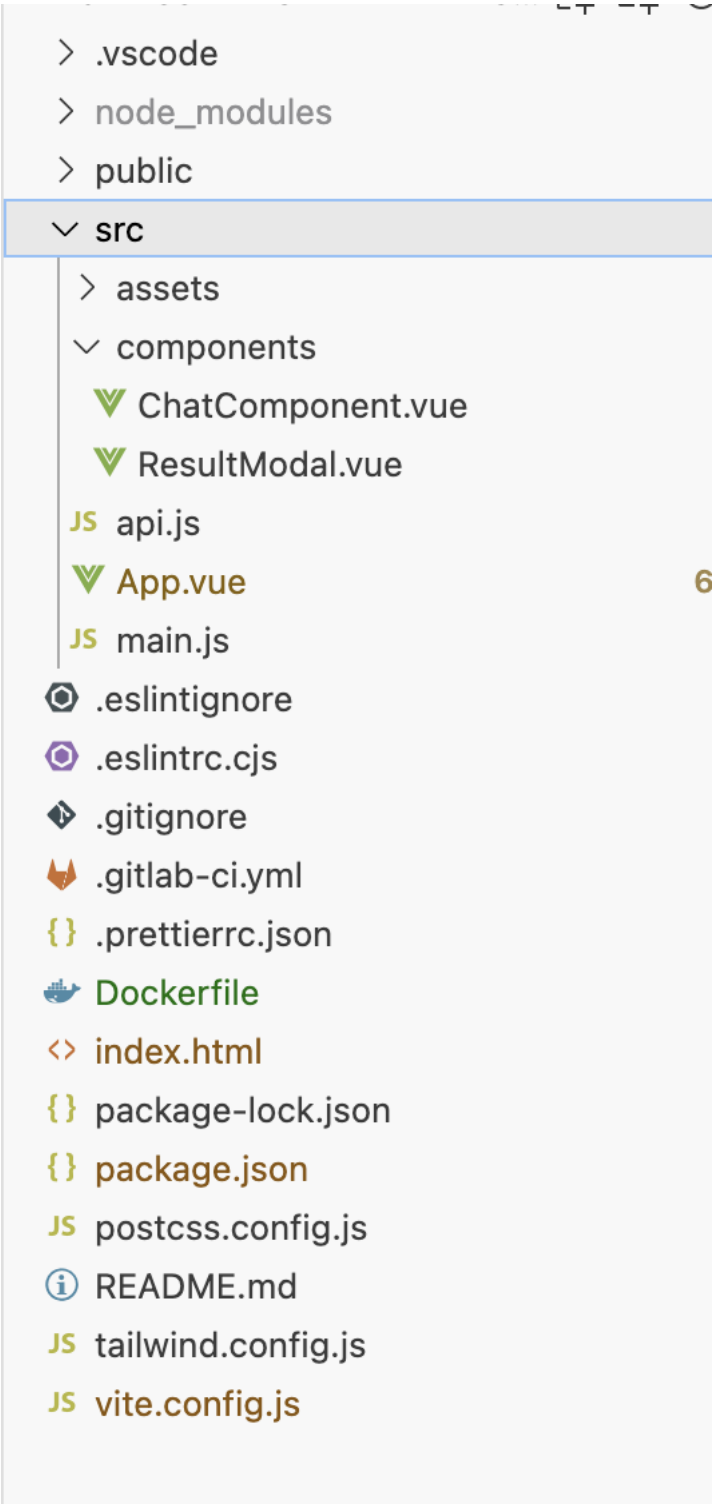

Рисунок 2. Скриншот структури проекту

Розглянемо структуру проекту

package.json - це важливий файл в проектах Node.js, який зберігає метадані про проект і конфігурує різні аспекти середовища. Він використовується npm для встановлення модулів, що залежать від вашого проекту, і включає інформацію про проект та його залежності.

Файл package.json:

```
{
  "name": "vue-latest",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "vite",
    "dev": "vite --open",
    "build": "vite build",
    "preview": "vite preview",
    "lint": "eslint . --ext .vue,.js,.jsx,.cjs,.mjs --fix --ignore-path .gitignore",
    "format": "prettier --write src/"
  },
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^6.4.0",
    "@fortawesome/free-brands-svg-icons": "^6.4.0",
    "@fortawesome/free-regular-svg-icons": "^6.4.0",
    "@fortawesome/free-solid-svg-icons": "^6.4.0",
    "@fortawesome/vue-fontawesome": "^3.0.3",
    "@headlessui/vue": "^1.7.13",
    "@heroicons/vue": "^2.0.18",
    "@tailwindcss/forms": "^0.5.3",
    "axios": "^1.4.0",
    "vue": "^3.2.47",
    "vue3-toast": "^0.0.1",
    "vue-social-sharing": "^4.0.0-alpha4"
  },
  "devDependencies": {
    "@rushstack/eslint-patch": "^1.2.0",
    "@vitejs/plugin-vue": "^4.2.1",
    "@vue/eslint-config-prettier": "^7.1.0",
    "autoprefixer": "^10.4.14",
    "eslint": "^8.39.0",
    "eslint-plugin-vue": "^9.11.0",
    "postcss": "^8.4.23",
    "prettier": "^2.8.8",
    "sass": "^1.62.1",
    "tailwindcss": "^3.3.2",
    "vite": "^4.3.4"
  }
}
```

Ось деякі ключові елементи, які можуть бути в файлі `package.json`:

name: Ім'я нашого проекту.

version: Поточна версія нашого проекту.

scripts: Об'єкт, який включає команди, які можна виконати з npm. Наприклад:

npm run start - запускає проект ;

npm run dev - запускає проект девелоп режимі;

npm run lint – виконує лінтер на вашому коді. Лінтер - це інструмент, що аналізує код на предмет помилок, проблем зі стилем коду, неправильного форматування, потенційних помилок та інших проблем;

npm run format - використовує інструмент Prettier для форматування коду вашого проекту. Prettier - це інструмент форматування коду, який забезпечує консистентність стилю кодування через весь проект

devDependencies: Об'єкт, що містить список пакетів, які використовуються тільки під час розробки, а не в продакшені.

Програмний код файлу index.html представлено нижче.

Файл index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <link rel="icon" href="/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://rsms.me/inter/inter.css">
    <title>Вимабот</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

Цей код представляє собою базовий HTML-файл для Vue.js-додатку. Давайте розглянемо кожну стрічку окремо:

- `<!DOCTYPE html>`: це декларація типу документа, яка повідомляє браузеру, що це сучасний HTML5-документ.

- `<html lang="en">`: це початковий тег HTML-документа. Атрибут `lang` вказує, що основна мова документа - англійська.

- `<head>`: це частина HTML-документа, що містить метадані, такі як кодування символів, посилання на іконку сайту, вигляд на мобільних пристроях і посилання на стилі.

- `<meta charset="UTF-8">`: це метатег, що вказує на кодування символів документа.

- `<link rel="icon" href="/favicon.ico">`: це посилання на іконку сайту, яка відображається в вкладці браузера.

- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: це метатег, що контролює, як сайт відображається на мобільних пристроях.

- `<link rel="stylesheet" href="https://rsms.me/inter/inter.css">`: це посилання на зовнішній CSS-файл, який використовується для стилізації веб-сторінки.

- `<title>Вітабот</title>`: це назва веб-сторінки, яка відображається в вкладці браузера.

- `<body>`: це частина HTML-документа, що містить вміст веб-сторінки.

- `<div id="app"></div>`: це контейнер для Vue.js-додатку. Vue.js буде "монтувати" додаток до цього елемента.

- `<script type="module" src="/src/main.js"></script>`: це посилання на основний JavaScript-файл додатку Vue.js. Значення `type="module"` дозволяє використовувати модулі ECMAScript в браузері.

- `</body>` та `</html>`: це закриваючі теги для елементів `body` та `html`.

Файл main.js стартовий файл для проекту:

```
import './assets/main.scss'
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'
import VueSocialSharing from 'vue-social-sharing'
import { createApp } from 'vue'
import { library } from '@fortawesome/fontawesome-svg-core'
import App from './App.vue'
import { faUserSecret } from '@fortawesome/free-solid-svg-icons'
import {
  faTwitter,
  faFacebook,
```

```
    faLinkedin,
    faTelegram
} from "@fortawesome/free-brands-svg-icons";
import {fas} from "@fortawesome/free-solid-svg-icons";
const app = createApp(App);
library.add(faUserSecret, faTwitter, faFacebook, faLinkedin, faTelegram, fas);
app.component("font-awesome-icon", FontAwesomeIcon);
app.use(VueSocialSharing);
app.mount('#app');
```

Даний код імпортує різні модулі та іконки, створює Vue.js додаток, реєструє компонент іконок FontAwesome та плагін соціального обміну, а потім монтує додаток на HTML елемент з id "app". Давайте розглянемо кожну стрічку окремо:

1. **import './assets/main.scss'** - Імпортує файл стилів SCSS, який знаходиться у директорії "assets".
2. **import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'** - Імпортує компонент FontAwesomeIcon з бібліотеки vue-fontawesome, що дозволяє використовувати іконки FontAwesome в Vue.js додатку.
3. **import VueSocialSharing from 'vue-social-sharing'** - Імпортує модуль vue-social-sharing, який дозволяє додати кнопки соціального обміну в додаток Vue.js.
4. **import { createApp } from 'vue'** - Імпортує функцію createApp з бібліотеки vue, що дозволяє створити новий екземпляр Vue.js додатку.
5. **import { library } from '@fortawesome/fontawesome-svg-core'** - Імпортує об'єкт library з бібліотеки fontawesome-svg-core, що дозволяє додавати іконки до бібліотеки FontAwesome, які будуть використовуватися в додатку.
6. **import App from './App.vue'** - Імпортує головний компонент додатку Vue.js з файлу App.vue.
7. **import { faUserSecret } from '@fortawesome/free-solid-svg-icons'** - Імпортує конкретну іконку (faUserSecret) з бібліотеки free-solid-svg-icons від FontAwesome.
8. Імпорт іконок з бібліотеки брендів FontAwesome.

9. **const app = createApp(App);** - Створює новий екземпляр додатку Vue.js з компонентом App.

10. **library.add(faUserSecret, faTwitter, faFacebook, faLinkedin, faTelegram, fas);** - Додає іконки до бібліотеки FontAwesome.

11. **app.component("font-awesome-icon", FontAwesomeIcon);** - Реєструє компонент FontAwesomeIcon в додатку Vue.js під ім'ям "font-awesome-icon".

12. **app.use(VueSocialSharing);** - Реєструє плагін VueSocialSharing в додатку Vue.js.

13. **app.mount('#app');** - Монтує (запускає) додаток Vue.js на HTML елемент з id "app". Після цього кроку додаток Vue.js повністю ініціалізується і починає працювати.

Файл App.vue є головним компонентом у проектах Vue.js, який слугує коренем (root) для всіх інших компонентів. Він зазвичай слугує місцем, де інші компоненти імпортуються та використовуються. App.vue може містити три основні секції:

<template>: - це секція, де ми описуємо HTML-структуру нашого додатку. Можна використовувати інші Vue компоненти в цій секції, а також директиви Vue, такі як v-if, v-for, v-bind та інші для динамічного контролю елементів.

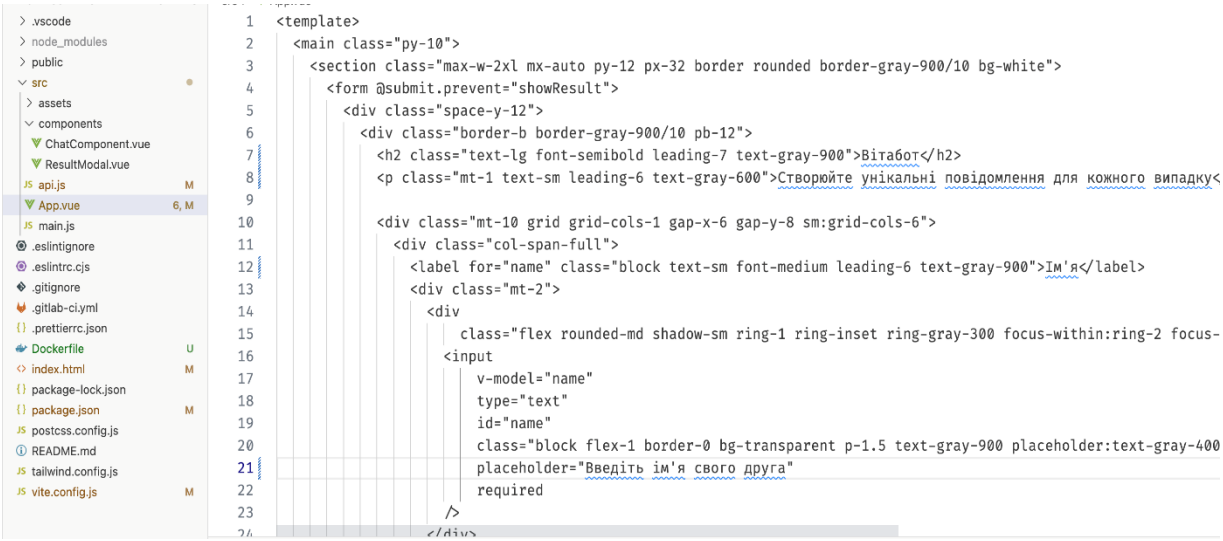

Рисунок 3. – Скриншот <template>

<script>: - це секція, де ми пишемо JavaScript або TypeScript код для нашого компонента. Тут експортуємо об'єкт Vue, який може містити різні опції, такі як data, methods, computed, watch, components та інші.

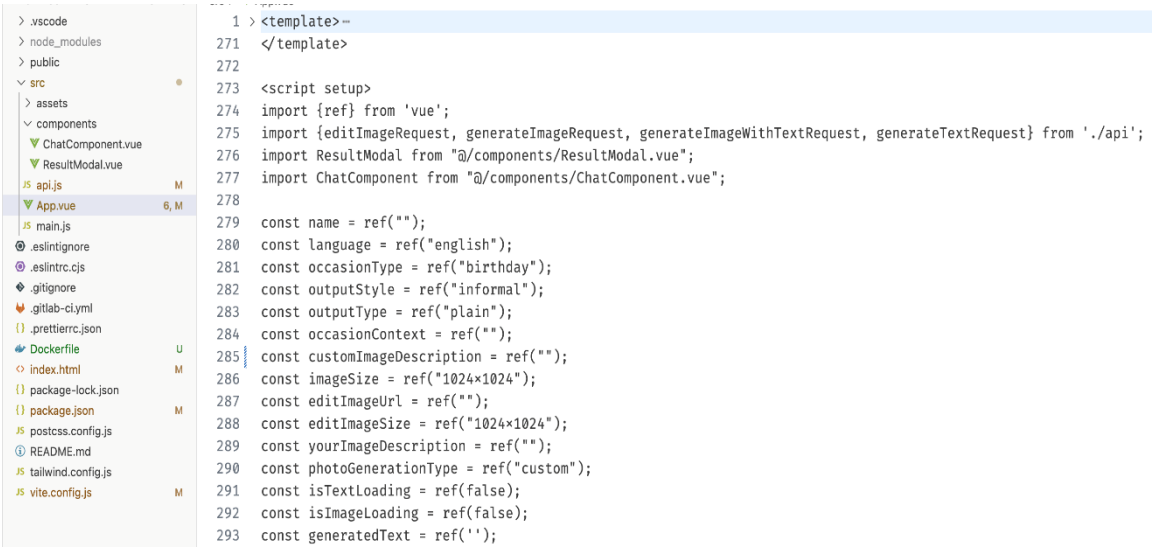

Рисунок 4. – Скриншот <script>

Дізайн програми

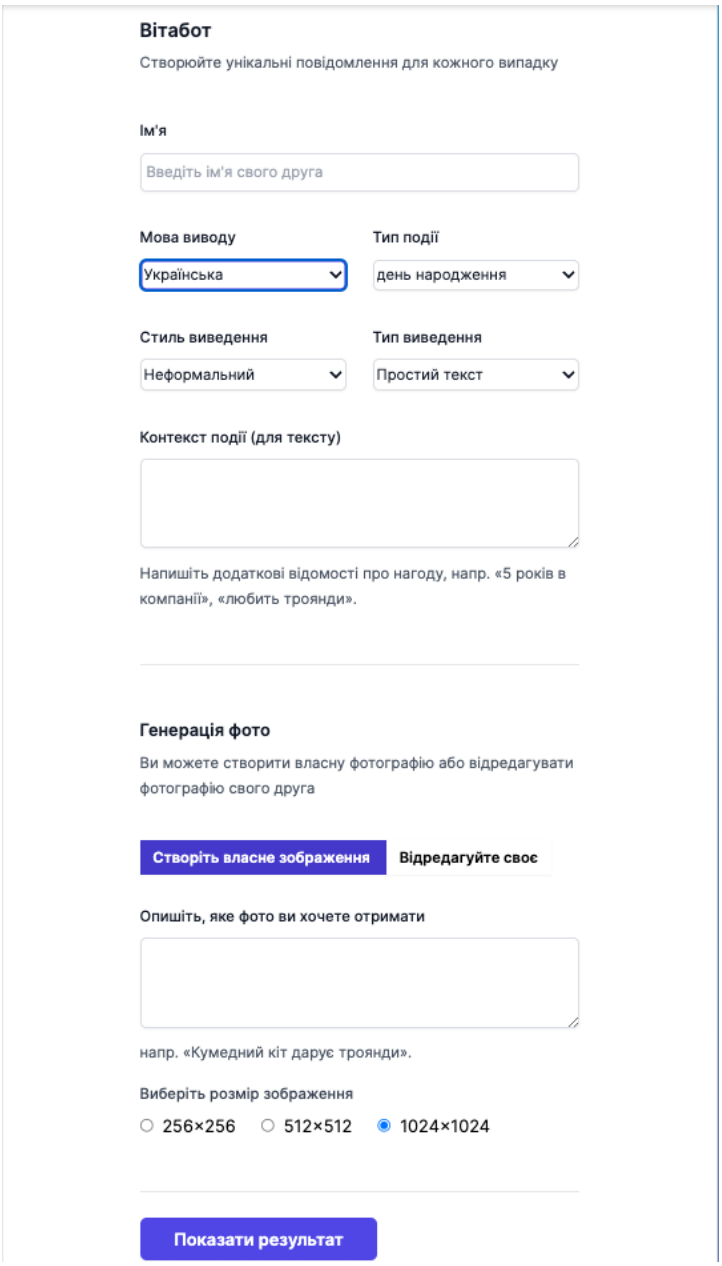

Рисунок 5– Скриншот роботи програми

У файлі api.js опис у взаємодії з бекенд частиною

```
1  import axios from 'axios'; 53.6k (gzipped: 19.8k)
2
3  const API_BASE_URL = 'http://localhost:3000/api';
4
5  export const generateTextRequest = async (name, occasionType, language, outputStyle, outputType,
   occasionContext) => {…
38 }
39
40 export const generateImageRequest = async (customImageDescription, imageSize) => {…
51 }
52
53 export const editImageRequest = async (editImageUrl, yourImageDescription, editImageSize) => {…
67 }
68
69
70
```

Рисунок 6 – Скриншот взаємодії з з бекенд частиною

generateTextRequest - запит на кінцеву точку (endpoint) бекенда для генерації тексту за підказкою (prompt)

generateImageRequest - запит на кінцеву точку генерації зображення за підказкою

editImageRequest - запит на Редагування зображення за підказкою

1.7.2 Розробка бекенд частини

Опишемо, як створити сервер на Node.js та Express з трьома маршрутами **generateText, generateImage, editImage**:

1. Встановлення Node.js

Переконуємося, що на нашому комп'ютері встановлено Node.js. Якщо ні, то ми можемо завантажити його та встановити з офіційного веб-сайту Node.js: https://nodejs.org

2. Створення нового проекту

Створіть нову папку для проекту. Наприклад, ви можете назвати його "greetings".

3. Ініціалізація проекту

Відкриваємо командний рядок (термінал) і перейдіть до папки "greetings". Потім виконуємо наступну команду, щоб ініціалізувати проект та створити файл package.json: *npm init -y*

Ця команда створить файл package.json, який містить інформацію про наш проект та його залежності.

4. Встановлення залежностей

Встановимо фреймворк Express.js в свій проект. Виконуємо наступну команду в командному рядку: *npm install express*

Ця команда завантажить та встановить Express.js та всі його залежності в папку "node_modules" вашого проекту.

5. Створення сервера

Створюємо новий файл у папці "grettings" і називаємо його, "server.js".

Відкриваємо його у редакторі коду і додаємо наступний код:

```
const express = require('express');

const app = express();
const port = 3000;
// Маршрут 1: POST https://api.openai.com/v1/completions
app.post('/api/completions, (req, res) => {
  res.send('Це маршрут completions!');
});

// Маршрут 2: POST https://api.openai.com/v1/images/generations
app.post('/api/images/generations, (req, res) => {
  res.send('Це маршрут images/generations!');
});

// Маршрут 3: POST https://api.openai.com/v1/images/edits
app.post('/api/images/edits, (req, res) => {
  res.send('Це маршрут images/edits!');
});

app.listen(port, () => {
  console.log(`Сервер запущений на порті ${port}`);
});
```

6. Запуск сервера

Відкриваємо командний рядок (термінал), перейдимо до папки "my-server" і виконуємо наступну команду:

node server.js

ми побачимо повідомлення "Сервер запущений на порті 3000", що означає, що наш сервер працює.

7. Перевірка сервера

Відкрийте веб-браузер і введіть у адресному рядку

`http://localhost:3000/api/completions`, `http://localhost:3000/api/images/generations` або `http://localhost:3000/api/images/edits`.

Ми повинні побачити відповідно текст "Це маршрут completions!", "Це маршрут images/generations!" або "Це маршрут images/edits!", що означає успішну відповідь від вашого сервера.

У нас вже є сервер, тепер потрібно реалізувати логіку взаємодії з API openai. Для роботи з openai необхідно спочатку зареєструватися та отримати токен.

Щоб токен не зберігався в git, створимо файл .env, і за допомогою бібліотеки dotenv зможемо отримувати його значення в програмі.

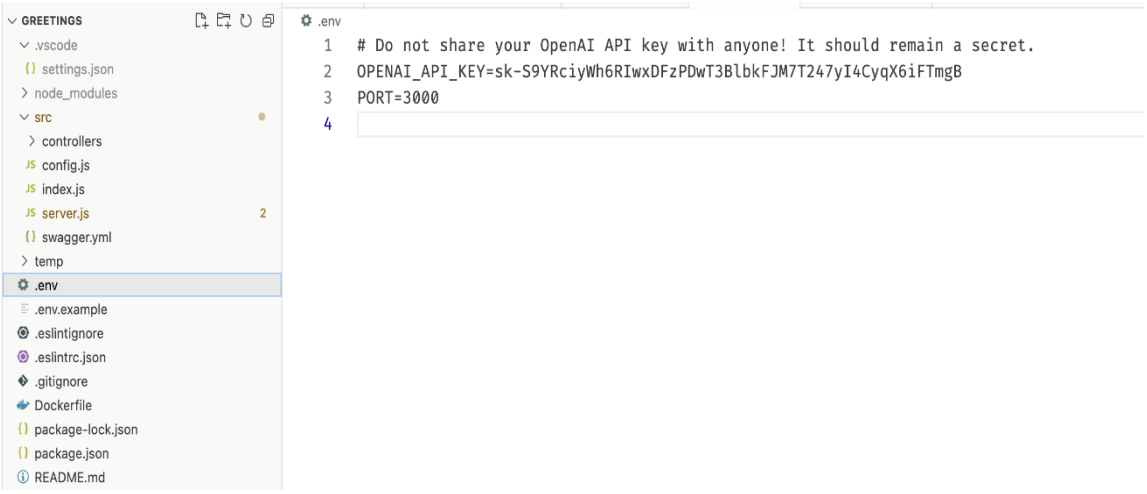

Рисунок 7 – Скриншот

Розглянемо код обробку маршрута створення тексту:

```
app.post('/api/completions', async (req, res) => {
  try {
    const { prompt } = req.body;
    const result = await getCompletions({ prompt });
    res.json(result);
  } catch (error) {
    res.status(500).send(String(error));
  }
});
```

Коли робиться POST-запит до цієї кінцевої точки, код витягує дані 'prompt' з тіла запиту. Потім код викликає функцію з назвою 'getCompletions()' і передає 'prompt' як аргумент. Після завершення виконання функція 'getCompletions()' повертає результат.

Результат, повернутий функцією 'getCompletions()', надсилається назад клієнту у вигляді JSON-об'єкта за допомогою методу 'res.json()'.

З іншого боку, якщо під час виконання виникла помилка, об'єкт-відповідь матиме статус 500 (внутрішня помилка сервера), а повідомлення про помилку буде надіслано назад клієнту за допомогою методу 'res.status().send()'.

Розглянемо метод getCompletions:

```
import { openai } from '../config.js';
export const getCompletions = async ({ prompt }) => {
  try {
    const response = await openai.createCompletion({
      model: 'text-davinci-003',
      prompt,
      max_tokens: 1000,
      temperature: 0.7,
    });
    return response.data.choices.map((choice) => ({ text: choice.text, index:
choice.index }));
  } catch (error) {
    throw new Error(error?.response?.data?.error?.message ||
error?.message);
  }
};
```

Цей код імпортує об'єкт openai з конфігураційного файлу *../config.js*, а потім експортує асинхронну функцію getCompletions. Ця функція отримує об'єкт як параметр, який має ключ підказки. Функція getCompletions використовує значення підказки для створення API-запиту до моделі GPT-3 OpenAI за допомогою функції openai.createCompletion().

API-запит містить назву моделі, prompt, max_tokens та параметри температури. Параметр model вказує, яку модель GPT-3 використовувати, а параметр prompt задає вхідний текст для заповнення. Параметр max_tokens визначає максимальну кількість токенів (слів або символів), які може згенерувати модель. Параметр temperature задає значення для керування випадковістю виведення згенерованого тексту.

Функція повертає масив об'єктів, кожен з яких має властивість text, що містить згенерований текст, та властивість index, що визначає індекс цього завершення.

Якщо виникає помилка, функція генерує помилку з повідомленням про помилку, включеним в об'єкт відповіді, що повертається з API.

Розглянемо конфігураційний файл *config.js*

Рисунок 8. - Скриншот конфігураційний файл config.js

```
import { Configuration, OpenAIApi } from 'openai';
import dotenv from 'dotenv';
dotenv.config();

const apiKey = process.env.OPENAI_API_KEY;
const configuration = new Configuration({ apiKey });
const openai = new OpenAIApi(configuration);
const port = process.env.PORT;

export {
   port,
   openai,
};
```

Код імпортує дві бібліотеки 'openai' та 'dotenv'. Бібліотека 'dotenv' завантажує змінні оточення з файлу '.env' в об'єкт process.env. Бібліотека 'openai' надає API для моделі обробки природної мови OpenAI GPT-3.

Потім код використовує об'єкт process.env для отримання значення 'OPENAI_API_KEY' і встановлює його як змінну apiKey.

За допомогою apiKey створюється новий об'єкт конфігурації. Цей об'єкт конфігурації потім використовується для створення нового екземпляра класу OpenAIApi, доступного у бібліотеці 'openai'.

Нарешті, код експортує порт і об'єкт openai, які можуть бути використані іншими частинами програми. Змінна port використовується для прослуховування вхідних запитів, в той час як об'єкт openai може бути використаний для виклику методів і використання моделі OpenAI GPT-3.

Розглянемо кінцеву точку створення зображення:

```
app.post('/api/images/generations', async (req, res) => {
  try {
    const { prompt, n, size } = req.body;
    const result = await generateImages({ prompt, n, size });
    res.json(result);
  } catch (error) {
    res.status(500).send(String(error));
  }
});

export const generateImages = async ({ prompt, n = 1, size = '1024x1024' })
=> {
  try {
    const response = await openai.createImage({ prompt, n, size });
    return response.data.data;
  } catch (error) {
    throw new Error(error?.response?.data?.error?.message ||
error.message);
  }
};
```

Функція приймає об'єкт як аргумент з трьома необов'язковими властивостями: prompt (підказка), n (число зображень, яке за замовчуванням дорівнює 1) і size (розмір зображеня, який за замовчуванням дорівнює "1024x1024").

Функція здійснює асинхронний виклик OpenAI API за допомогою методу openai.createImage. В якості параметрів методу ми передаємо prompt (підказка), size (розмір зображеня), Ми також передаємо), n (число зображень) і кількість зображень, які ми хочемо згенерувати (за допомогою параметра n).

Відповідь від API повертається тому, хто викликав. Відповідь містить масив зображень. Кожне зображення має url та підпис. Підпис - це текст, який ми передали в якості підказки для цього зображення. URL - це посилання на згенероване зображення.

Якщо є помилка, функція видає помилку. Якщо помилка пов'язана з OpenAI API, то в якості повідомлення про помилку використовується повідомлення з API. В іншому випадку в якості повідомлення про помилку використовується повідомлення з помилки.

1.8 Опис роботи програми.

1.8.1 Привітальна листівка.

Хочемо поздоровити з ювілеєм колегу на ім’я Лілія, побажати їй здоров'я, мирного неба, приємних подій.

Фон листівкі привітальної має бути таким:

Гарні троянди.

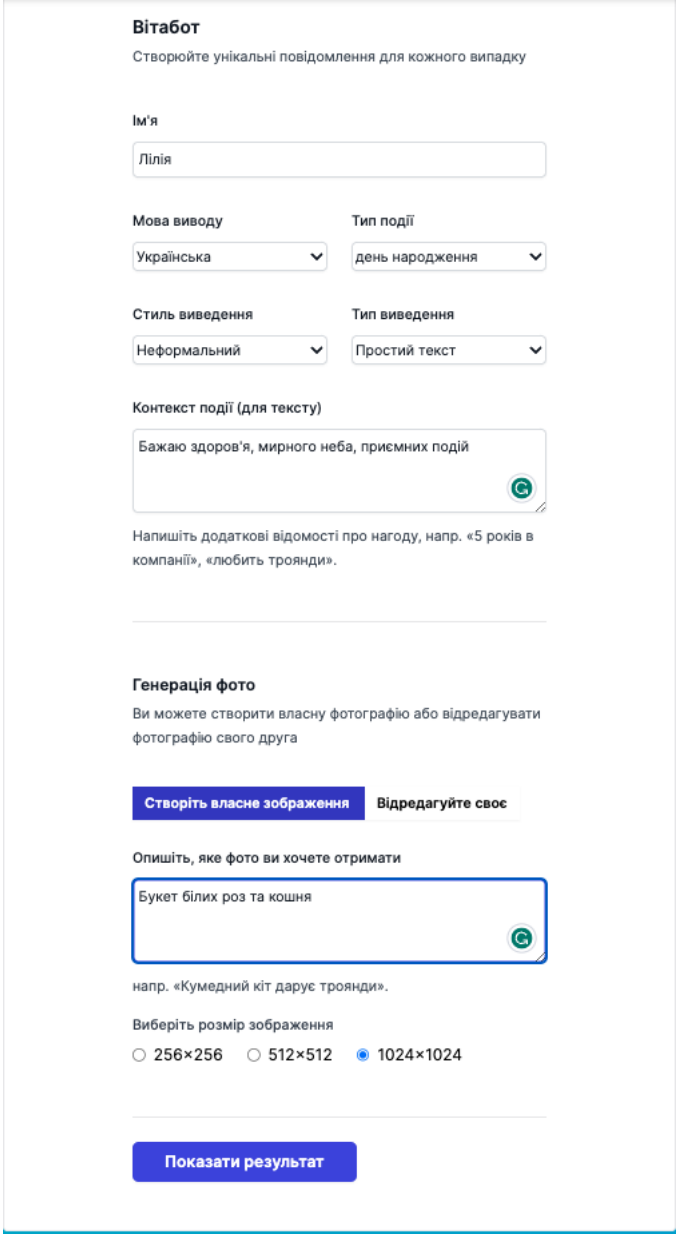

Рисунок 9.– Скриншот вікна програми

Рисунок 10. – Скриншот обробки заданних параметрів

Також ми можемо відправити вітальну листівку через такі месенджери як фейсбук, твитер, телеграм

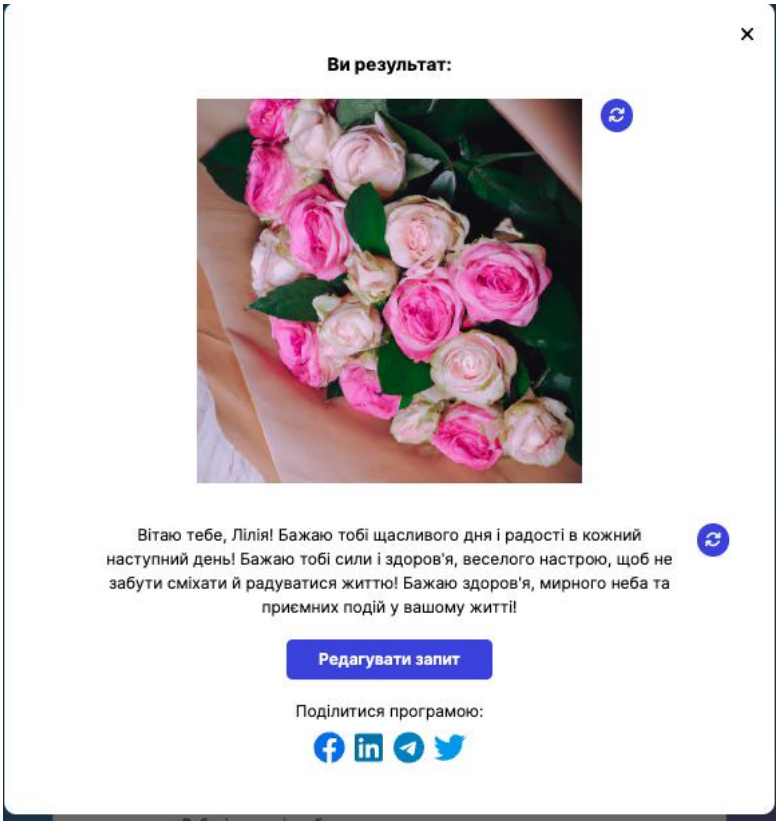

Рисунок 11. – Скриншот привітання

Якщо нам не сподобався результат ми можемо повторити обробку декілька раз, поки не отримуємо результат, який нам довподоби.

1.8.2 Редагування свого фото

Це ще одна з функцій нашої програми. Можна взяти любе фото та вставити до щось. Наприклад, я взяв фото головного корпусу коледжа з сайту та бажаю висловити слова подяки колективу коледжу.

Вставили фото :

https://scontent.fods8-1.fna.fbcdn.net/v/t39.30808-6/294030682_433653342109016_4447528365081357310_n.jpg?_nc_cat=111&c

cb=1-

7&_nc_sid=e3f864&_nc_ohc=5DpI_I92j4QAX8Qee19&_nc_ht=scontent.fods8-1.fna&oh=00_AfDcd2OIxOLMMnTA9tWATqv_cTNlTWnhTOWXGrv5LPgt2A&oe=64797983

Рисунок 12. - Фото головного корпусу коледжу для обробки

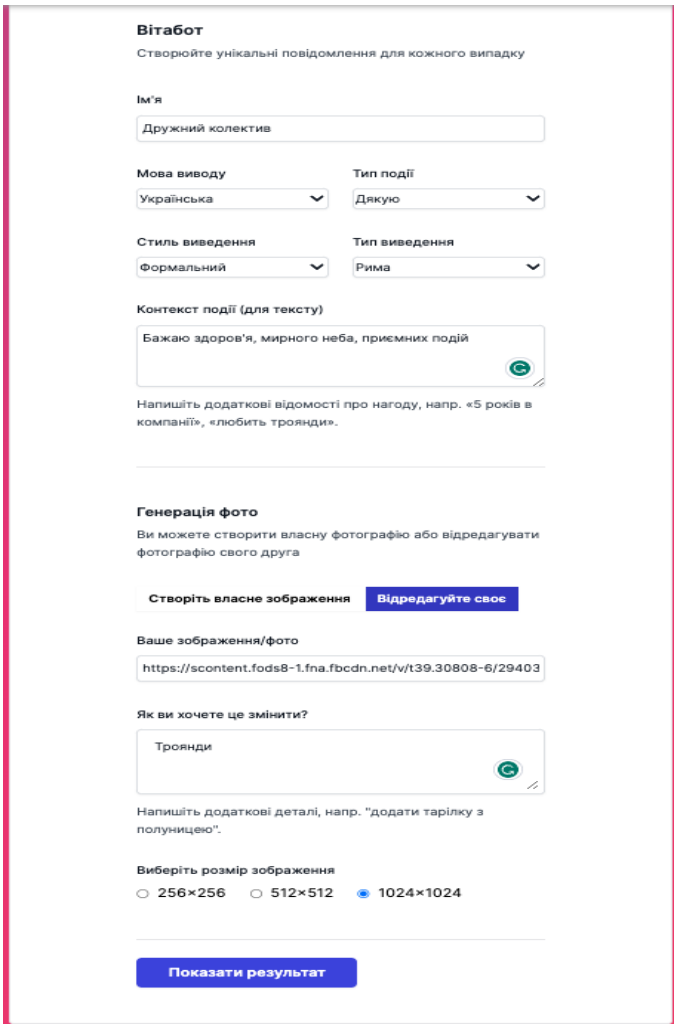

Рисунок 13. Скриншот вікна програми для обробки фото

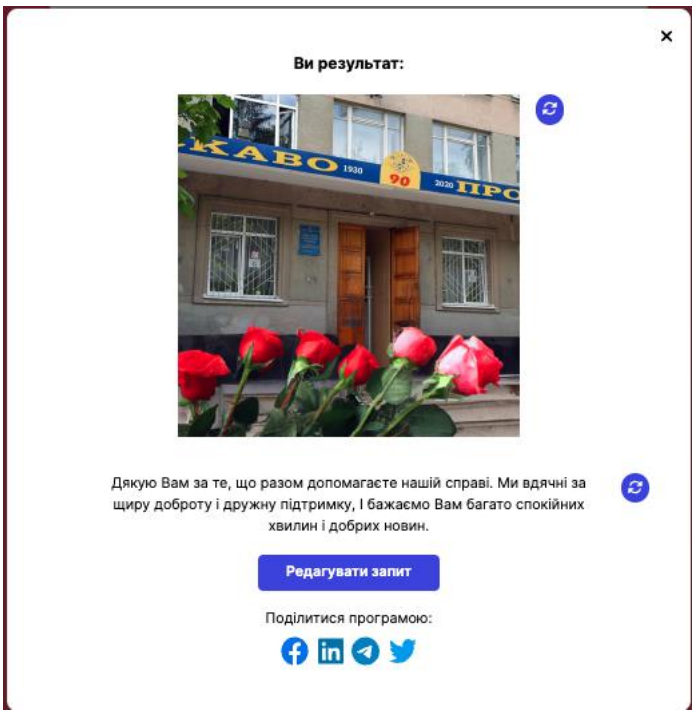

Рисунок14. Скриншот вікна результата обробки програми після обробки фото

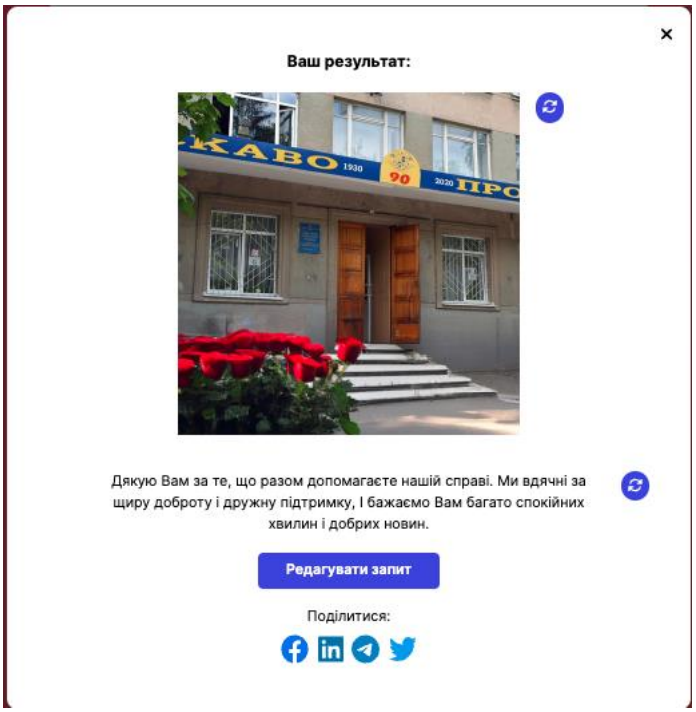

Рисунок 15. Скриншот вікна результата обробки програми після другої обробки фото

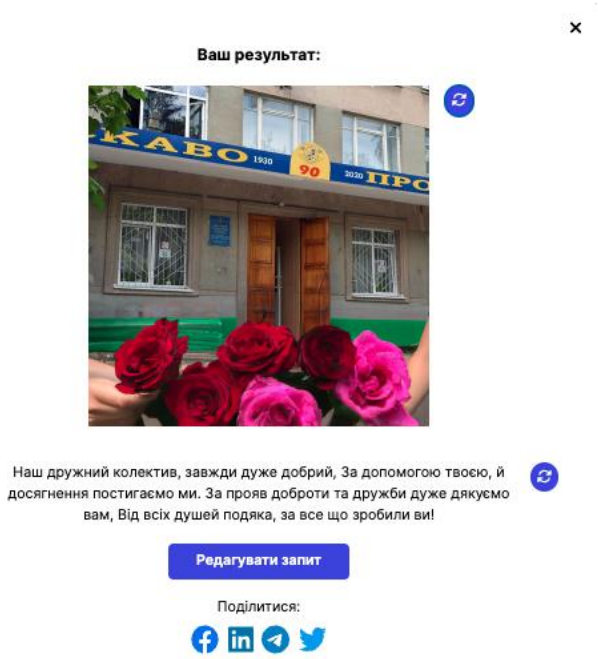

Рисунок 16. Скриншот вікна результата обробки програми після третьої обробки фото

2.ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

Темою даного дипломного проекту є «Розробка програми для генерації повідомлень і зображень за допомогою штучного інтелекту». Програма дає нам можливість формувати та надсилати привітальні листівки та редагувати різні малюнки та фото.

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

2.2. Визначення трудомісткості розробки програмного забезпечення.

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

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

Каталог аналогів

Таблиця 2.1

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

Для нашого варіанта виділено сірим кольором.

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

Таблиця.2.2

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

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

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

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

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

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

Для розрахунку необхідні наступні коефіцієнти:

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

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

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

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

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

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

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

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

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

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

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

Трудомісткість розробки технічного проекту

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

Трудомісткість розробки робочого проекту

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

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

Таблиця 2.5. Розрахунок трудомісткості ПП

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

2.3 Розрахунок ціни програмного продукту.

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

Таблиця 2.6 Розрахунок основної заробітної плати виконавців.

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

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

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПО

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

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

Таблиця 2.8. Розрахунок статей витрат планової собівартості

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

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

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

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

Оптова ціна (кошторисна вартість) визначається по формулі:

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

Податок на додану вартість визначаємо по наступній формулі:

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

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

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

3 ОХОРОНА ПРАЦІ

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

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

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

3.1 Аналіз та безпека умов праці працівника на робочому місці

Під час будь-якого виду роботи за комп'ютером, на працівника можуть мати дію небезпечні фактори виробничого середовища, а саме: фізичні та психофізіологічні небезпечні й шкідливі виробничі фактори.

Серед фізичних небезпечних факторів, найпоширеніші це підвищена температура повітря робочої зони, підвищений рівень шуму, знижена вологість повітря – це звичайні фактори, які виникають при роботі у приміщеннях з комп'ютерами, через їх роботу на робочому місті підіймається температура та знижається вологість повітря. 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.

Психофізіологічні виробничі небезпечні фактори поділяються на фізичні перевантаження та нервово-психічні перевантаження, при роботі з комп'ютером найчастіше друге. У нервово-психічних перевантаженнях програміст зазнає перенапругу аналізаторів та монотонність праці, інколи, ще й розмовну перенапругу, коли розробнику потрібно складати технічне завдання разом з клієнтом.

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).

Мікроклімат

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

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

3.3 Організація робочого місця користувача ПК

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

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

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

3.4 Пожежна безпека

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

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

Первинні засоби пожежогасіння застосовуються для боротьби з пожежами на початковій стадії. До них належать: пожежні кран-комплекти, вогнегасники,

пожежний інвентар (резервуари з водою, ящики з піском, пожежні відра, лопати), а також різний переносний пожежний інструмент (кирки, сокири, багри, ломи і т. ін.).

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

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

Згідно Правил, на кожному поверсі будинку адміністративного призначення повинно знаходитися не менше двох вогнегасників з масою заряду вогнегасної речовини 5 кг і більше. Експлуатація вогнегасників без призначення відповідального за організацію цієї роботи не допускається.

Забороняється палити на підприємстві, крім спеціально виведених для цього місцях, забороняється зберігати легкозаймисті матеріали, такі як папір ближче ніж 1 метр від електрощитів, 0,15 м від приладів центрального водяного опалення та 0,6 м від сповіщувачів автоматичної пожежної сигналізації, також документація повинна зберігатися у спеціально відведених для цього шафах.

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

Підприємство крім установки пожежної сигналізації на своєму об'єкті, має укласти договір на обслуговування даної системи з фірмою, що має на це ліцензію. В обслуговування входить проведення встановлених нормами регламентних робіт, а так само усунення несправностей в роботі системи. Періодичність

перевірки узгоджується з замовником, але повинна бути не рідше ніж один раз на місяць.

У разі, якщо пожежі не вдалось уникнути, необхідно:

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

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Сілаєва Ігоря Геннадійвича

(прізвище, ім`я та по батькові)

Спеціальність **121 «Програмна інженерія»**

Освітня програма **«Розробка програмного забезпечення»**

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

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи)

Розробка програми для генерації повідомлень за допомогою штучного інтелекту

Обсяг розрахунково-пояснювальної записки 65 сторінок

Обсяг графічної (презентаційної) частини 14 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

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

Представлений на рецензію робота відповідає затверджений темі та виконаний відповідно технічному завданню.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Пояснювальна записка роботу виконана якісно, з дотриманням усіх норм та стандартів. В дипломному проекті проаналізовано існуючі рішення, проведено аналіз та систематизовано порядок появи на ринку сучасного програмного забезбечення штучного інтелекту та його застосування. Проведено аналіз та прозроблено програмне забезпечення. Розроблена структура програми та виконана програмна реалізація генерації повідомлень.

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

Графічна частина складається з 11 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми та схеми алгоритму, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки висока, розробку виконано у повному обсязі

г) перелік позитивних якостей дипломного проекту (роботи)_____

Застосування сучасних комп'ютерних програм для розробки програми длягенерації повідомлень за допомогою штучного інтелекту є актуальним завданням. Також інтерес представляє висновки та рекомендації щодо подальшого їх застосування адаптації під відповідні завдання.

д) основні недоліки дипломного проекту (роботи) _____

1. У пояснювальній записці не зазначено для якої конкретного застосування та варіанта запропонованого сучасного програмни продукти та платформ.

2. У тексті пояснювальної записки зустрічаються друкарські помилки та неточності.

3. Відсутнє посилання літературу.

Оцінка розрахункової частини *відмінно*

Оцінка графічної частини *відмінно*

Загальна оцінка *відмінно*

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, к.ф.н., доцент кафедри КБ та ТЗІ, пом.декану факультету інформаційних технологій та кібербезпеки

Підпис: _____

«16» 06 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

ВІДДІЛ КАДРІВ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Сілаєва Ігоря Геннадійовича

(прізвище, ім'я та по батькові)

Спеціальність: 121 «Інженерія програмного забезпечення»"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Розробка програми для генерації повідомлень і зображень за допомогою штучного інтелекту

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 65 сторінок. У пояснювальній записці виконано опис етапів розробки програми для генерації повідомлень і зображень за допомогою штучного інтелекту. Графічна частина складається з 12 слайдів мультимедійної презентації, які також містять креслення, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом:

Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Сілаєв І.Г. 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 Сілаєв І.Г. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

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

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Сілаєв І.Г. мав змогу самостійно приймати окремі рішення з реалізації програмного продукту, а саме структурної схеми проекту та показав вміння організовано працювати над поставленим завданням, розробити програмний продукт за допомогою мов программування, таких як JawaScript, PHP.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Кунуп Тетяна Вісилівна

Місце роботи і посада керівника дипломного проекту

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

Підпис

«09» 06 2023 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Сілаєв Ігор Геннадійович,
здобувач освіти гр. 4РП-06, та

Кунуп Тетяна Василівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту молодшого спеціаліста на тему:

«Розробка автоматизованої системи перевірки якості JavaScript-коду» (автор роботи – Сілаєв І.Г., керівник роботи – Кунуп Т.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 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 _____ / Сілаєв І.Г. /

Керівник _____ / Кунуп Т.В. /

«12» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015546589

Дата перевірки:
11.06.2023 12:50:09 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2023 12:53:03 EEST

ID користувача:
100011688

Назва документа: 4РП-06 Сілаєв І.Г

Кількість сторінок: 57 Кількість слів: 9073 Кількість символів: 68525 Розмір файлу: 4.56 MB ID файлу: 1015199122

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

10%
Схожість

Найбільша схожість: 1.06% з Інтернет-джерелом (https://www.bibliofond.ru/view.aspx?id=891894)

10% Джерела з Інтернету	805	..... Сторінка 59

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	32

Підозріле форматування	15 сторінок