КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

АРТ-ВІЗУАЛІЗАЦІЯ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав студент 4-го курсу
Ігор СЕЛЕЗЕНЬ

(підпис)

Науковий керівник:
доцент кафедри теорії та технології програмування
Олексій ТКАЧЕНКО (підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри теорії та технології програмування
«____» _____ 201_ р.,
протокол № ___
Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 38 сторінок, 29 ілюстрацій, 10 джерел посилань.

TELEGRAM BOT, АРТ-ВІЗУАЛІЗАЦІЯ ЗА ОПИСОМ, JAVA, SPRING, SPRING BOOT, TELEGRAM API, OPENAI

Об’єктом дослідження**:** даної роботи є растрові зображення.

Предметна область**:** технології генерування зображень за допомогою штучного інтелекту на основі природної мови.

Мета роботи**:** Розробка telegram бота для візуалізації наданого в природному стилі тексту, з можливістю обирати стилістику зображення.

Методи дослідження**:** загальнонаукові, метод моделювання, об’єктно-орієнтований підхід.

Задачі дослідження**:**

- дослідження основних аспектів растрових зображень
- дослідження існуючих методів генерування зображень
- моделювання та розробка телеграм бота для візуалізації опису за допомогою штучного інтелекту

Засоби розробки**:**

- інструменти: мова програмування Java (JDK 17), IDE IntelliJ IDEA, база даних PostgreSQl, Postman, Apache maven, сторонні API: telegram API, OpenAI API, залежності: Spring (Spring Boot, Spring Web, Spring Data JPA, Spring DevTools), OpenAI-Java, Lombok, PostgreSQL driver, Telegram Bots, OkHttp, Apache Commons.

Результати роботи**:** У результаті даної роботи було успішно здійснено проектування та розробка продукту з простим інтерфейсом та необхідним функціоналом. Застосунок, який було створено, дозволяє користувачу згенерувати зображення на основі наданого опису, з можливістю вибору стилістики зображення із набору доступних варіантів.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

IDE - Integrated Development Environment

API - Application Programming Interface

JDK - Java Development Kit

HTTP - HyperText Transfer Protocol

REST - Representational State Transfer

POJO - Plain Old Java Object

БД - база даних

ORM - Object-Relational Mapping

ВСТУП

Оцінка сучасного стану об’єкта розробки. Актуальність роботи та підстави для її виконання: На сьогоднішній день, генерування зображень штучним інтелектом є одним із актуальних напрямів дослідження в галузі комп'ютерного зору та штучного інтелекту. Згідно зі статистикою від Simiilarweb, кожного місяця провідними системами в даній галузі користуються десятки мільйонів користувачів.

Metric	Jan 2023	Feb 2023	Mar 2023	Apr 2023
Midjourney Website Traffic (Million)	29.1	31.6	41.4	42.7
Month Over Month Growth (%)		9%	31%	3%

Source: Similarweb

Рисунок - статистика трафіку користувачів сервісу MidJorney

Розвиток технологій машинного навчання та глибокого навчання, включаючи генеративні моделі, відкриває нові можливості для створення зображень, які мають великий потенціал у таких галузях, як мистецтво, дизайн, реклама та інші.

Зокрема, в даній роботі розроблений Telegram бот для візуалізації тексту з використанням штучного інтелекту є актуальним рішенням, оскільки забезпечує користувачам зручний та доступний інтерфейс для створення унікальних зображень за допомогою природної мови.

Популярність месенджерів, зокрема Telegram, значно зростає, що робить розробку такого бота досить актуальною та має потенціал залучити широке коло користувачів.

Таким чином, об'єкт розробки є актуальним у контексті сучасних тенденцій розвитку штучного інтелекту, машинного навчання та використання інтерактивних месенджерів.

Мета й завдання роботи**:** Ця кваліфікаційна робота має на меті розробку telegram бота для візуалізації наданого в природному стилі тексту, з можливістю обирати стилістику зображення. Об'єктом дослідження є генерування зображень штучним інтелектом за природної мовою, а предмет - арт-візуалізація за допомогою штучного інтелекту.

1. Вивчення сучасних підходів та методів генерування зображень штучним інтелектом, зокрема, за допомогою генеративних моделей та глибокого навчання.

2. Розробка архітектури та функціональності Telegram бота, що дозволяє користувачу надати текстовий опис та вибрати стилістику для генерування зображення.

3. Інтеграція з відповідними зовнішніми сервісами та API, зокрема з Telegram API для забезпечення взаємодії з ботом та OpenAI API для генерування зображень.

4. Розробка візуального інтерфейсу користувача, що забезпечує зручну та інтуїтивно зрозумілу взаємодію з ботом.

5. Тестування та оцінка продукту, включаючи точність генерування зображень та задоволення користувачів від використання Telegram бота.

Виконання поставлених завдань дозволить досягти поставленої мети і розробити функціональний та ефективний Telegram бот для генерування зображень за текстовим описом з можливістю вибору стилістики.

Можливі сфери застосування:

- Мистецтво та творчість: Бот може бути корисним інструментом для митців, дизайнерів та творчих осіб, які шукають натхнення та швидкого створення власних унікальних зображень на основі текстових описів.

- Реклама та маркетинг: Бот може бути використаний для створення ефектних зображень для рекламних кампаній, брендування продуктів, створення візуального контенту для соціальних медіа та інших маркетингових цілей.

- Ігрова індустрія: Завдяки можливості генерування зображень на основі природної мови, бот може бути використаний у розробці комп'ютерних ігор для створення унікальних графічних елементів, персонажів та світів.

- Освіта та навчання: Бот може слугувати інструментом для візуалізації текстових матеріалів, навчальних концепцій та ідей, що полегшує сприйняття та розуміння навчального матеріалу.

- Графічний дизайн та веб-розробка: Бот може бути корисним для дизайнерів та веб-розробників, які шукають швидкий інструмент для генерування зображень для веб-сайтів, ілюстрацій, графічного оформлення та інших проектів.

- Персоналізовані послуги: Бот може бути використаний в різних сферах, що пропонують персоналізовані послуги, такі як створення індивідуальних портретів, гравюр, логотипів тощо.

Це лише декілька прикладів можливих сфер застосування, і потенціал використання розробленого бота може бути ще ширшим, залежно від конкретних потреб та креативності користувачів.

РОЗДІЛ 1. ПЕРЕДУМОВИ СТВОРЕННЯ ІНСТРУМЕНТА

1.1 Загальні відомості про генерування зображень за допомогою штучного інтелекту:

Генерування зображень за допомогою штучного інтелекту є активною галуззю досліджень, яка займається створенням алгоритмів та моделей, що можуть автоматично створювати нові, реалістичні зображення з мінімальною або навіть без участі людини. Ця галузь знаходить застосування в різних областях, включаючи мистецтво, дизайн, комп'ютерну графіку та рекламу.

Штучний інтелект, зокрема глибокі нейронні мережі, використовуються для генерування зображень шляхом вивчення статистичних закономірностей у великому обсязі вхідних даних, таких як фотографії, малюнки або текстові описи. Моделі глибокого навчання, такі як генеративні згорткові мережі (GANs) та автокодувальні мережі (Autoencoders), є основними інструментами для генерування зображень.

Процес генерування зображень може включати кілька етапів, включаючи підготовку даних, навчання моделей та саме генерування. Після навчання модель може генерувати нові зображення на основі заданих вхідних параметрів, таких як текстові описи або стилістика.

Головною перевагою генерування зображень з використанням штучного інтелекту є можливість створення унікальних, неповторних зображень, які не існують в реальному світі. Це відкриває нові можливості для творчості, дизайну та інновацій. Однак, використання штучного інтелекту для генерування зображень також супроводжується викликами, такими як забезпечення якості, контроль над процесом генерування та етичні питання, пов'язані з використанням створених зображень.

В даній роботі використовуватиметься DALL-E, яка є моделлю генерації зображень за допомогою штучного інтелекту, що використовує комбінацію архітектурних елементів з метою створення різноманітних та якісних візуальних результатів. Вона базується на трансформерах та використовує глибоке навчання для створення відповідних зображень з текстових описів.

Основним принципом роботи DALLE є зв'язування тексту та зображень. Модель навчається на великому наборі зображень та текстових пар, що описують ці зображення. Вона навчається встановлювати взаємозв'язок між текстовими вказівками та візуальними ознаками, а потім використовує ці знання для генерації нових зображень з новими текстовими описами.

Один з основних внесків DALLE полягає в тому, що вона дозволяє генерувати різноманітні зображення з одного текстового опису. Вона використовує метод згортки відображень, щоб створити варіації візуальних результатів, враховуючи різні шляхи інтерпретації текстових вказівок.

Однак, для успішної роботи DALLE потрібно мати велику кількість обчислювальних ресурсів, так як модель має значну кількість параметрів і вимагає інтенсивного обчислення. Це може бути викликом при застосуванні моделі в реальних виробничих середовищах з обмеженими обчислювальними можливостями але не завадить у написанні даної кваліфікаційної роботи.

1.2 Огляд та аналіз наявних на ринку систем:

На ринку вже існують рішення для арт-візуалізації за описом, серед яких найпопулярнішими є MidJorney, DALL-E та Stable Diffusion. Однак, кожне з цих рішень має свої проблеми:

- MidJorney: Незважаючи на свою популярність, MidJorney має напрочуд незручний інтерфейс. Для використання цієї системи користувачам потрібно писати вказівки (далі prompts) у спеціальному форматі на діскорд сервері. Цей процес може бути вимогливим та неінтуїтивним для користувачів, особливо для тих, хто не володіє технічними навичками

- DALL-E: DALL-E є іншим розвиненим рішенням для арт-візуалізації. Вона використовує глибокі нейромережі, щоб генерувати зображення з текстових описів. Однак, однією з проблем DALL-E є її обмежена доступність, оскільки ця система є пропрієтарною та недоступною для широкої публіки.

- Stable Diffusion: Ще одним рішенням є Stable Diffusion, яка використовує методи дифузії для генерування зображень. Проте, проблемою з використанням Stable Diffusion є необхідність скачування та встановлення графічного ядра nVidia. Це створює обмеження для користувачів, які не мають доступу до відповідного обладнання.

Одна спільна проблема, що присутня у всіх цих рішень, полягає в складності написання якісних prompts. Користувачам потрібно мати глибокі знання алгоритмів та моделей, а також розуміння їхніх обмежень та принципу роботи, щоб відповідним чином сформулювати prompts. Цей процес вимагає досвіду та експертизи, і для багатьох користувачів може стати викликом.

Розроблений Telegram бот вирішує цю проблему, оскільки він дозволяє користувачам взаємодіяти з ботом за допомогою тексту, написаного в природному стилі. Користувачам не потрібно писати складні prompts або володіти технічними навичками. Замість цього, вони можуть зручно вибирати стилістику зображення, що значно полегшує процес отримання бажаних результатів.

1.3 Актуальність розробленого застосунку:

- Природна мова: Однією з основних переваг розробленого бота є його здатність сприймати природну мову. Використання простої розмови замість написання prompts робить його більш доступним для користувачів, які необов'язково володіють технічними навичками

- Вибір стилістики зображення: Розроблений бот надає користувачам можливість обрати стиль зображення. Це дозволяє їм впливати на візуальний аспект створюваних зображень і створює більш індивідуальний та персоналізований підхід.

- Зручність використання: Використання Telegram бота має свої переваги порівняно з входженням на веб-сайт або використанням Discord-каналу. Він забезпечує мобільний доступ та можливість використовувати його на різних платформах. Користувачам не потрібно встановлювати додаткові програми (враховуючи популярність месенджера telegram) або реєструватися на додаткових платформах. Це дозволяє зробити процес отримання зображень більш зручним та швидким.

- Інтерактивність і зручність навігації: Telegram боти надають можливість взаємодії з користувачами шляхом кнопок, меню та інших елементів управління. Це робить навігацію та взаємодію з ботом більш інтуїтивною та зручною, порівняно зі стандартним введенням команд або навігацією на веб-сайті.

РОЗДІЛ 2. РОЗРОБКА ЗАСТОСУНКУ

2.1 Модель проєкту:

В проєкті присутні три модулі (мікро-сервіси), детальніше з яким ви можете ознайомитись далі. Далі наведене схематичне зображення взаємодії модулів між собою

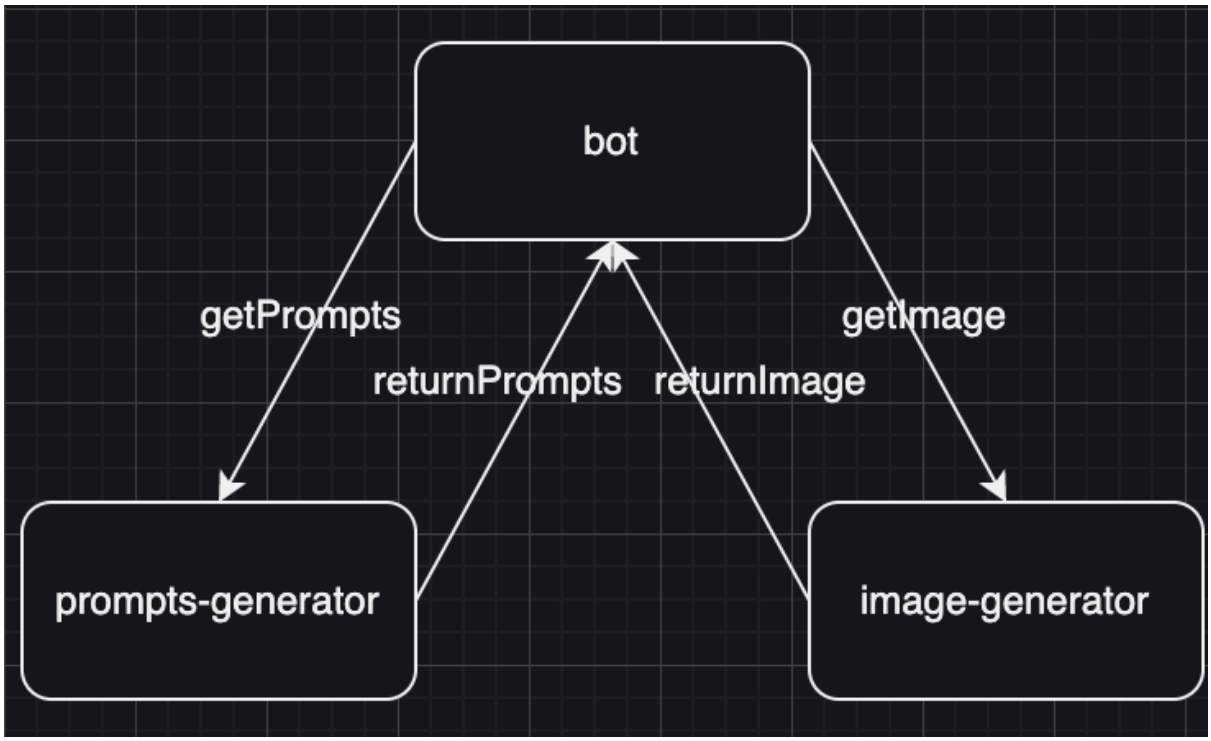

Рисунок 2.1.1 - Зображення взаємодії модулів

2.2 Проєктування програмної частини:

2.2.1 Набір програмних засобів розробки:

- Java (JDK 17)
- Apache maven
- PostgreSQl
- IDE IntelliJ IDEA
- Postman
- Telegram API (Telegram Bots)
- OpenAi API (OpenAI-Java)

- Spring (Spring Boot, Spring Web, Spring Data JPA, Spring DevTools)
- Lombok
- PostgreSQL driver
- OkHttp3
- Apache Commons

2.2.2 Архітектура програми:

проєкт був розроблений за допомогою мікро-сервісної архітектури та містить в собі 3 мікро-сервіси:

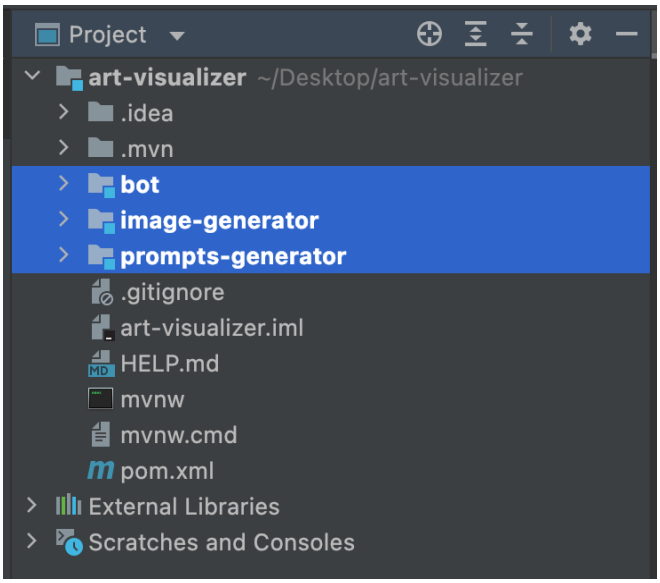

Рисунок 2.2.2.1 - структура проєкту

- мікро-сервіс prompts-generator:

  мікро-сервіс має наступну структуру

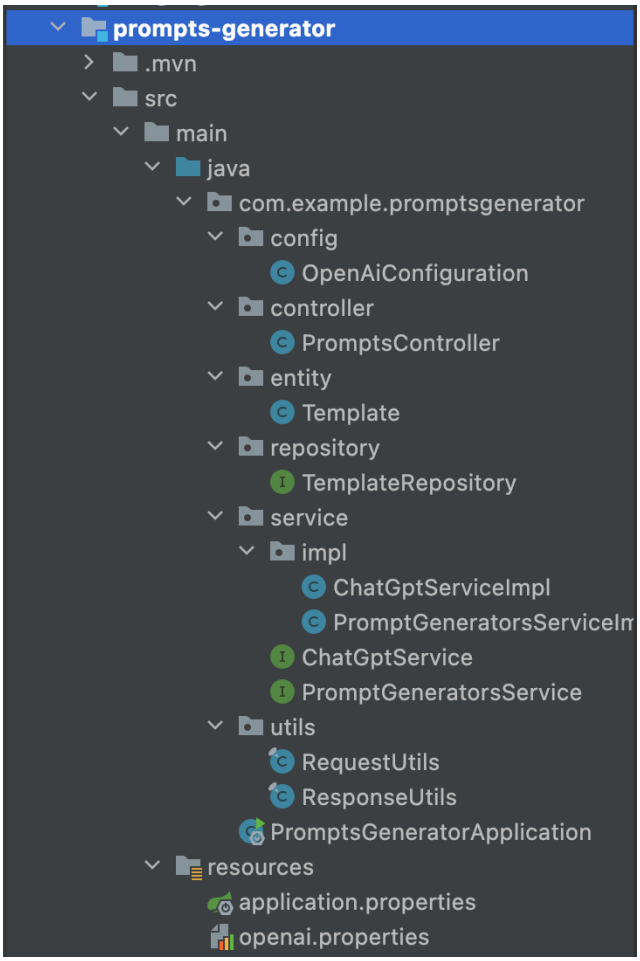

Рисунок 2.2.2.2 - Структура мікро-сервіса propmts-generator

така структура є стандартною для проєктів, написаних за допомогою фреймворку Spring Boot, в яких виділяються декілька основних модулів:

- Клас додатку (Application class): Це основний клас додатку, який містить метод main() і служить як вхідна точка для запуску додатку.
- Контролери (controller): Контролери виконують обробку HTTP-запитів, встановлюють маршрути і взаємодіють з користувачем. Вони містять анотації, такі як @RestController і @RequestMapping, для визначення маршрутів та обробки запитів.
- Сервіси (service): Сервіси містять бізнес-логіку додатку і виконують специфічні завдання. Вони виконуються в контексті контролера і зазвичай виконують операції з базою даних, зовнішніми сервісами або іншими компонентами.
- Репозиторії (repository): Репозиторії виконують операції з базою даних, такі як збереження, вибірка, оновлення і видалення даних. Вони забезпечують взаємодію з базою даних і можуть використовувати JPA (Java Persistence API) або інші технології доступу до даних.
- Моделі (entity): Моделі представляють об'єкти даних, що використовуються в додатку. Вони можуть бути аннотовані за допомогою анотацій JPA для мапування на базу даних або використовуватися для передачі даних між контролерами і сервісами.
- Конфігурація (config): Конфігураційні класи використовуються для налаштування різних аспектів додатку, таких як налаштування бази даних, бінів, залежностей та інших параметрів.
- Ресурси (resources): містить файли з розширенням .properties, які використовуються для зберігання конфігураційних параметрів програми.

- мікро-сервіси image-generator та bot мають схожу структуру:

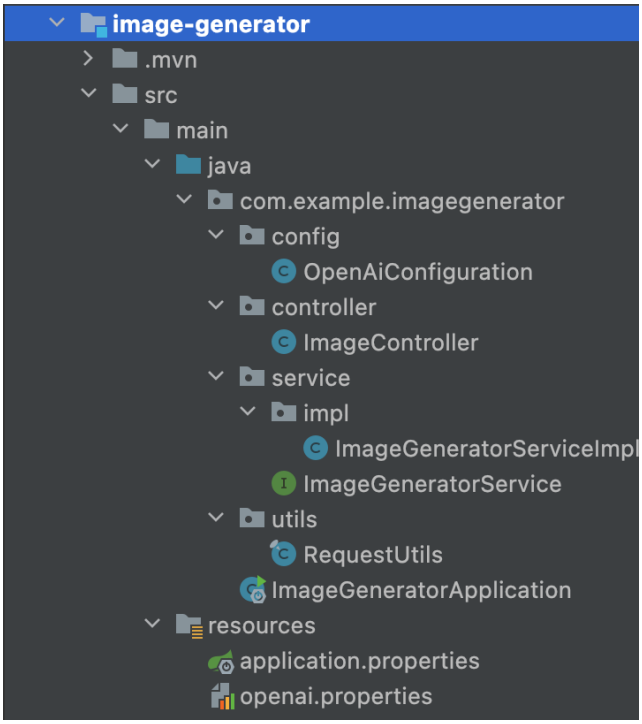

Рисунок 2.2.2.3 - Структура мікро-сервіса image-generator

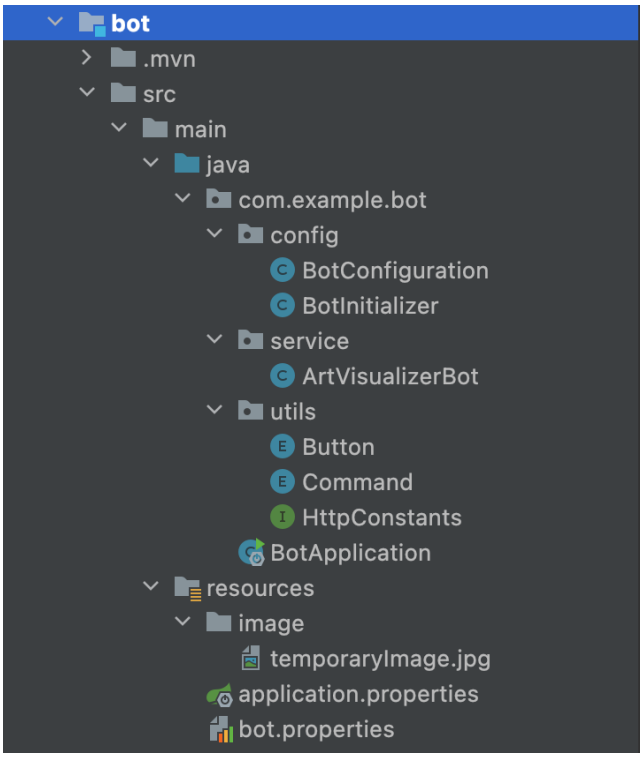

Рисунок 2.2.2.4 - Структура мікро-сервіса bot

2.2.3 “Спілкування” мікросервісів:

Оскільки програма поділена на 3 мікро-сервіси, має існувати спосіб обміну даними між ними. При розробці даного проєкту, був обраний спосіб спілкування через REST, в основі якого лежить набір принципів та обмежень, які дозволяють створювати легкі, масштабовані та стійкі до змін веб-додатки.

Зі схематичної моделі, представленої в пункті 3.1 видно, що модулі prompts-generator та image-generator мають вміти приймати запити, а bot - відправляти їх. З цієї причини, в перших двох наявні контроллери, що містять endpoint’и (адреси, заа якими можна звернутись до цього контроллеру), а другому - OkHttpClient, за допомогою якого можна відправляти запити на відповідні контроллери та отримувати відповіді. Детальніше про реалізацію читайте в наступних пунктах.

2.3 Бізнес-логіка проєкту:

2.3.1 Загальна логіка:

Зрозуміти загальну логіку проєкту буде легше, зрозумівши кінцевий результат, що в нашому випадку є отриманням згенерованого зображення. Для генерування зображення використовувалась нейронна мережа DALL-E, звертатись до якої дозволяє API OpenAi, і, як вже було зазначено раніше, для генерування якісних зображень за допомогою цієї мережі, потрібно писати якісні, з технічної точки зору, вказівки (prompts). Отже, ці вказівки потрібно звідкись брати, цим займається мікро-сервіс propmpts-generator, з нього і почнемо:

2.3.2 Prompts-generator:

мікро-сервіс prompts-generator має контроллер PromptsController

```
@RestController
@RequestMapping(value = "/prompt")
@RequiredArgsConstructor
public class PromptsController {
    private final PromptGeneratorsService promptGeneratorsService;

    @GetMapping("/{templateType}")
    public String generatePrompts(@RequestParam String message,
                                  @PathVariable String templateType) {
        return promptGeneratorsService.generatePrompts(message, templateType);
    }
}
```

Рисунок 2.3.2.1 - Контроллер PromptsController

який надає можливість мікро-сервісу bot звертатись до нього. Далі, ми потрапляємо в клас PromptGeneratorsServiceImpl, який містить в собі метод generatePrompts(), в якому є три етапи.

```
@Service
@RequiredArgsConstructor
public class PromptGeneratorsServiceImpl implements PromptGeneratorsService {
    private final ChatGptService chatGptService;
    private final TemplateRepository templateRepository;

    @Override
    public String generatePrompts(String message, String templateType) {
        Template template = templateRepository.getTemplateByType(templateType);
        String preparedMessage = buildMessage(message, template.getTemplate());
        return chatGptService.createCompletion(preparedMessage);
    }

    private String buildMessage(String message, String template) {
        return String.format(
            "answer only in format 'prompt: <generate prompts>' and nothing else "
                + "apply this template '%s' to the following description %s "
                + "and create a prompt for dall-e, "
                + "you need to be specific and try to put an answer in 20 words ",
            template,
            message);
    }
}
```

Рисунок 2.3.2.2 - Сервіс PromptsGeneratorServiceImpl

Спочатку, на основі того, який стиль обире юзер (яку кнопку натисне в телеграм боті), prompts-generator обирає відповідний шаблон (template) для запиту на генерацію вказівок (prompts), усі шаблони зберігаються в базі, та дістаються за унікальним ідентифікатором, який в свою чергу вшитий у CallBack відповідної кнопки в телеграм бота, що дає змогу легко співставляти відповідні кнопки з обраними стилями.

Далі, на основі опису, який ввів юзер, та шаблону, витягнутого з бази даних, формується повідомлення (метод buildMessage()), яке в результаті відправляється на API OpenAi, після чого валідується та повертається юзеру. Сам запит виглядає наступним чином:

```
@UtilityClass
public class RequestUtils {
    private static final String GPT_VERSION = "gpt-3.5-turbo";
    private static final int CHAT_COMPLETION_CHOICES_AMOUNT = 1;
    private static final int MAX_TOKENS_AMOUNT = 100;

    public ChatCompletionRequest buildTextRequest(String message) {
        List<ChatMessage> messages = new ArrayList<>();
        ChatMessage systemMessage = new ChatMessage(ChatMessageRole.SYSTEM.value(), message);
        messages.add(systemMessage);
        return ChatCompletionRequest
            .builder()
            .model(GPT_VERSION)
            .messages(messages)
            .n(CHAT_COMPLETION_CHOICES_AMOUNT)
            .maxTokens(MAX_TOKENS_AMOUNT)
            .logitBias(new HashMap<>())
            .build();
    }
}
```

Рисунок 2.3.2.3 - Утильний клас RequestUtils

В даному модулі міститься додаткова логіка (формування запитів, валідація відповідей, взаємодія з базою даних, мапинг POJO об’єктів до таблиць в БД за допомогою ORM властивостей фреймворку Spring Data JPA тощо) детальніше з якою ви можете ознайомитись на GitHub репозиторії з даним проєктом

2.3.3 Image-generator:

Після отримання вказівок (prompts) у нас є все для безпосереднього генерування зображення, нею займається image-generator.

Так само як і попередній модуль, image-generator містить контроллер, із одним end-point'ом на Get запит, і єдиний query parameter, який йому потрібний це String prompt, що являє собою вказівку до генерування зображення

```java
@RestController
@RequestMapping("/image")
@RequiredArgsConstructor
public class ImageController {
    private final ImageGeneratorService imageGeneratorService;

    @GetMapping
    public String generateImage(@RequestParam String prompt) {
        return imageGeneratorService.createImage(prompt);
    }
}
```

Рисунок 3.3.3.1 - Контроллер ImageeController

Далі, запит відправляється на те саме API OpenAi, де в якості відповіді ми отримуємо посилання на щойно згенероване зображення

```
    @Override
    public String createImage(String prompt) {
        return getOpenAiService().createImage(
                    RequestUtils.buildImageRequest(prompt))
                .getData().get(0).getUrl();
    }

    private OpenAiService getOpenAiService() { return new OpenAiService(openAiConfiguration.getToken()); }
}
```

Рисунок 2.3.3.2 - Сервіс ImageGeneratorServiceImpl

Загалом, тут відбувається, приблизно, те саме, що і в попередньомуу модулі, детальніше з кодом ви можете ознайомитись на GitHub

2.3.4 bot:

Найбільша частина бізнес-логіки цього модуля зосереджена в класі ArtVisualizerBot.

Почнемо з методу, який виконує роль event-listener'а: onUpdateReceived(), цей клас наслідує абстрактний клас TelegramLongPollingBot, тому цей метод раз в певний, дуже малий час перевіряє, чи не надійшло нове повідомлення

```
    @Override
    public void onUpdateReceived(Update update) {
        if (Objects.nonNull(update) && update.hasMessage()) {
            Message message = update.getMessage();
            String text = message.getText();
            if (Objects.equals(text, Command.START.getCommandText())) {
                handleStartCommand(message);
            } else {
                handleTextMessage(message.getChatId(), text);
            }
        } else if (Objects.nonNull(update) && update.hasCallbackQuery()) {
            handleCallback(update);
        }
    }
```

Рисунок 2.3.4.1 - Метод onUpdateReceived

В цьому методі ми обробляємо 3 типи вхідних повідомлень:
- команда “/start”: у відповідь на команду /start ми відправляємо юзеру заготовлений текст, з проханням надіслати опис для зображення, в який вшивається username користувача, що дістається з повідомлення

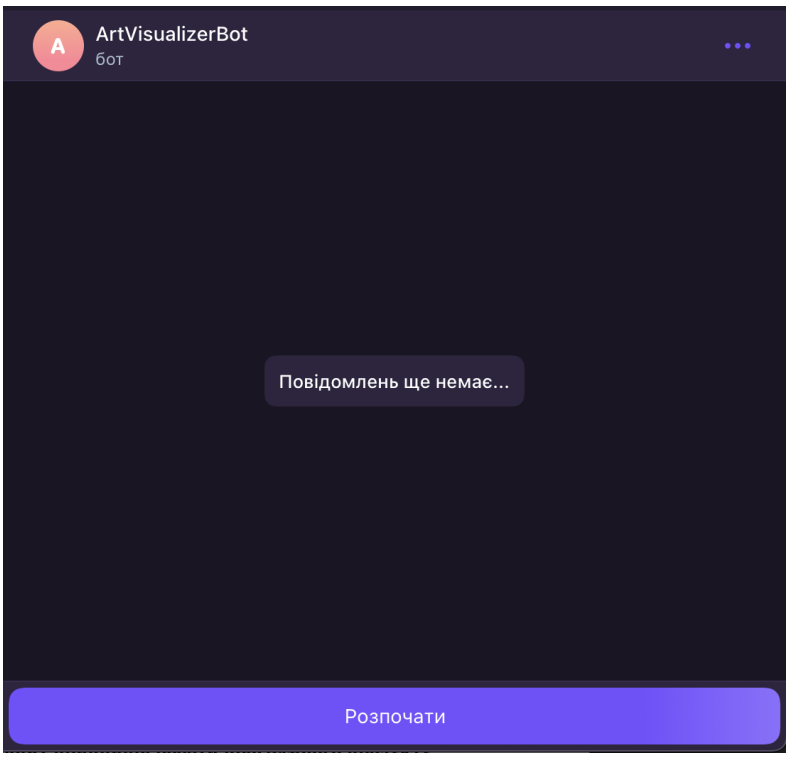

Рисунок 2.3.4.2 - Стратова сторінка взаємодії з ботом

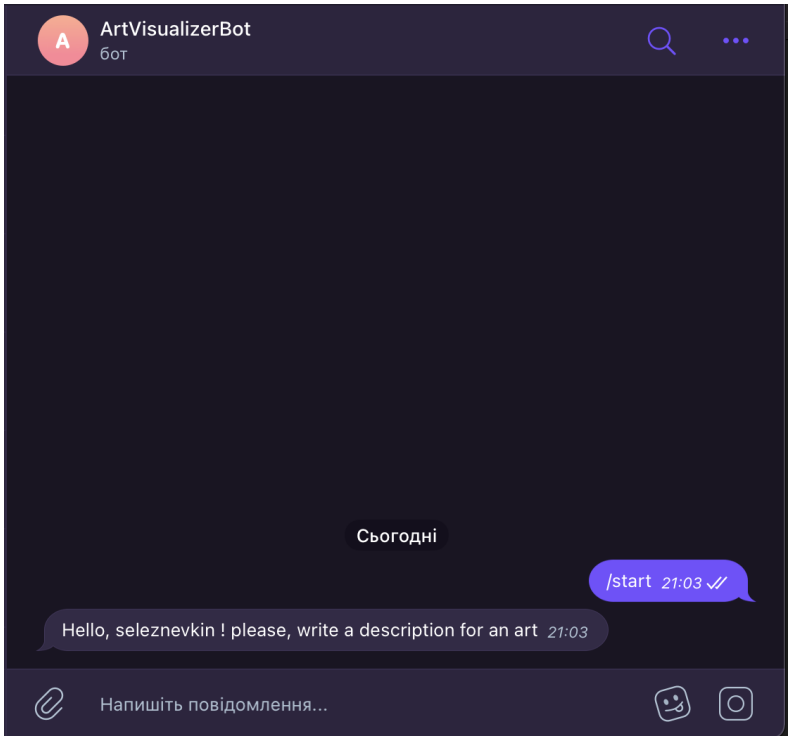

Рисунок 3.3.4.3 - Реакція на команду /start

- текстові поввідомлення:

у відповідь на текстове повідомлення з описом зображення, юзеру надсилається повідомлення з прикрипленим InlineKeyboard зі списком стилів на вибір. Після натискання кнопки, надсилається новий event з CallBack'ом кнопки

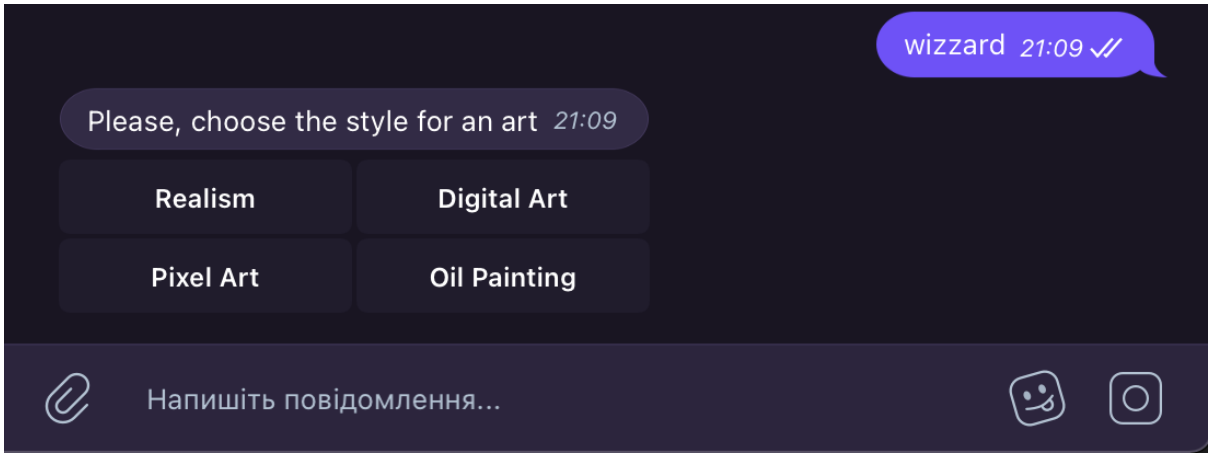

Рисунок 2.3.4.4 - Реакція на введеня опису

- надходження події CallBackQuery (повертається при натисканні юзером на кнопку)

```
@RequiredArgsConstructor
@Getter
public enum Button {
    REALISM( buttonText: "Realism",  buttonCallback: "realism"),
    DIGITAL_ART( buttonText: "Digital Art",  buttonCallback: "digital_art"),
    PIXEL_ART( buttonText: "Pixel Art",  buttonCallback: "pixel_art"),
    OIL_PAINTING( buttonText: "Oil Painting",  buttonCallback: "oil_painting");

    private final String buttonText;
    private final String buttonCallback;
}
```

Рисунок 2.3.4.5 - Усі доступні кнопки

усі наявні варіанти зберігаються в Enum Button, тож для масштабування програми потрібно лише додати нову кнопку в цю Enum і новий стиль в таблицю template

```
    private void handleCallback(Update update) {
        Long chatId = update.getCallbackQuery().getMessage().getChatId();
        sendMessage(chatId,
             message: "please wait, it will take some time to generate your image");
        String urlForArt = generateArtByDescription(update);
        sendMessage(chatId,
             message: "image is ready, downloading it");
        sendPhoto(urlForArt, chatId);
    }
```

Рисунок 2.3.4.6 - Метод обробки Callback події

далі викликається метод handleCallBack(), який відправляє юзер інформацію про те, що зображення почало генеруватись, кидає запит на prompts-generator для генерування опису, а потім цей опис кидається на image-generator для генерування зображення, в результаті чого ми отримуємо посилання на зображення, яке лишаються скачати та відправити юзеру

запити на отримання вказівок та зображення відправляються за допомогою OkHttp3:

```
private Request buildPromptsRequest(Update update) {
    HttpUrl.Builder urlBuilder
        = Objects.requireNonNull(HttpUrl.parse(
            BASE_URL
                + PROMPT_PORT
                + PROMPT_ENDPOINT
                + "/"
                + update.getCallbackQuery().getData()))
        .newBuilder();
    urlBuilder.addQueryParameter( name: "message", description);

    String url = urlBuilder.build().toString();
    return new Request.Builder()
        .url(url)
        .build();
}
```

Рисунок 2.3.4.7 - Метод побудови запиту

```
private Response makeRequest(Request request) {
    OkHttpClient client = new OkHttpClient.Builder()
        .connectTimeout(DEFAULT_CONNECTION_TIMEOUT,
            TimeUnit.MILLISECONDS)
        .readTimeout(DEFAULT_READ_TIMEOUT,
            TimeUnit.MILLISECONDS)
        .build();
    Call call = client.newCall(request);
    try {
        return call.execute();
    } catch (IOException e) {
        throw new RuntimeException("error occurred during a call", e);
    }
}
```

Рисунок 2.3.4.8 - Метод з відправкою запиту

значення для побудови запиту зберігаються в утильному інтерфейсі:

```
public interface HttpConstants {
    String BASE_URL = "http://localhost:";
    String PROMPT_ENDPOINT = "/prompt";
    String PROMPT_PORT = "8081";
    String IMAGE_ENDPOINT = "/image";
    String IMAGE_PORT = "8082";
    int DEFAULT_CONNECTION_TIMEOUT = 60000;
    int DEFAULT_READ_TIMEOUT = 60000;
}
```

Рисунок 2.3.4.9 - Константи, що використовуються при запитах

останнє що залишається зробити - викачати згенероване зображення за посиланням, що повернулося з image-generator модуля. Відбувається це в методі downloadImage() робиться це за допомогою бібліотеки Apache commons

```
    private void downloadImage(File temporaryImage, String url) {
        try {
            FileUtils.copyURLToFile(
                new URL(url),
                temporaryImage,
                DEFAULT_CONNECTION_TIMEOUT,
                DEFAULT_READ_TIMEOUT);
        } catch (IOException e) {
        throw new RuntimeException("cannot download an image", e);
        }
    }
```

Рисунок 2.3.4.10 - Метод скачування зображення

тепер лишається лише відправити фото, робиться це у методі sendPhoto()

```
private void sendPhoto(String url, Long chatId) {
    try {
        File temporaryImage = new File(TEMPORARY_IMAGE_PATH);
        downloadImage(temporaryImage, url);

        SendPhoto sendPhoto = new SendPhoto();
        sendPhoto.setPhoto(new InputFile(temporaryImage));
        sendPhoto.setChatId(chatId);

        execute(sendPhoto);
    } catch (TelegramApiException e) {
        throw new RuntimeException("cannot send the message", e);
    }
}
```

Рисунок 2.3.4.11 - Метод відправки зображення

З повним кодом програми можна ознайомитись на GitHub репозиторії https://github.com/issugy/art-visualizer/tree/master

РОЗДІЛ 3. ГЕНЕРУВАННЯ ЗОБРАЖЕНЬ

3.1 Генерування вказівок (Prompts):

Як було зазначено раніше, для вдалого генерування зображення потрібно вказувати структуровані prompts, тому для даної програми було згенеровано декілька шаблонів, які зберігаються в таблиці template

	id	type	template
1	1	realism	photography, detailed,…
2	2	digital_art	A digital Illustration…
3	4	pixel_art	pixel art
4	5	oil_painting	van gogh style oil pai…

Рисунок 3.1.1 - Таблиця templates

приклад шаблону для стилю “realism” - “photography, detailed, realistic, photo-realistic, 8k, highly detailed, full length frame, piercing, diffused soft lighting, shallow depth of field, sharp focus, hyperrealism, cinematic lighting”

після отримання запиту на генерування вказівок створюється запит, який відправляється до OpenAi з використанням gpt3.5, а той в свою чергу повертає вказівку

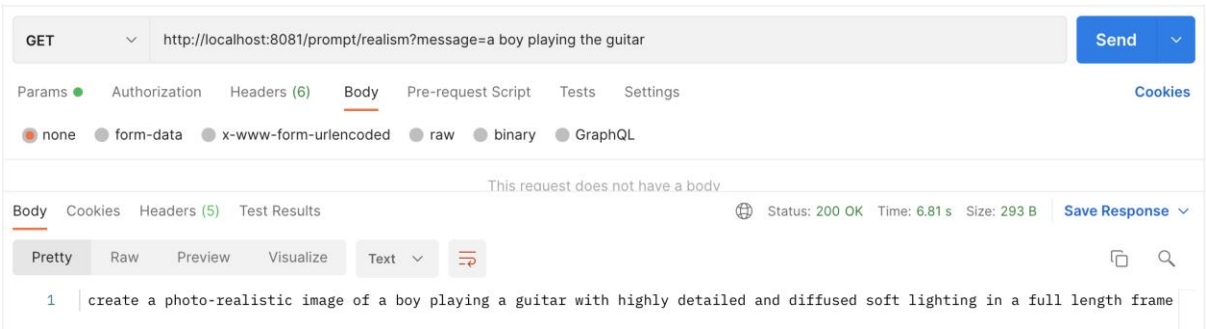

Рисунок 3.1.2 - Запит на генерування вказівок

3.2 Генерування зображення:

Відбувається за допомогою API OpenAi. Для цього потрібен тільки опис, який був попередньо згенерований prompts-generator. OpenAi розробляє DALL-E - нейронну мережу для генерування зображень, тому використовуючи їх API можна генерувати зображення. В результаті генерування, користувачу повертається посилання на згенероване зображення

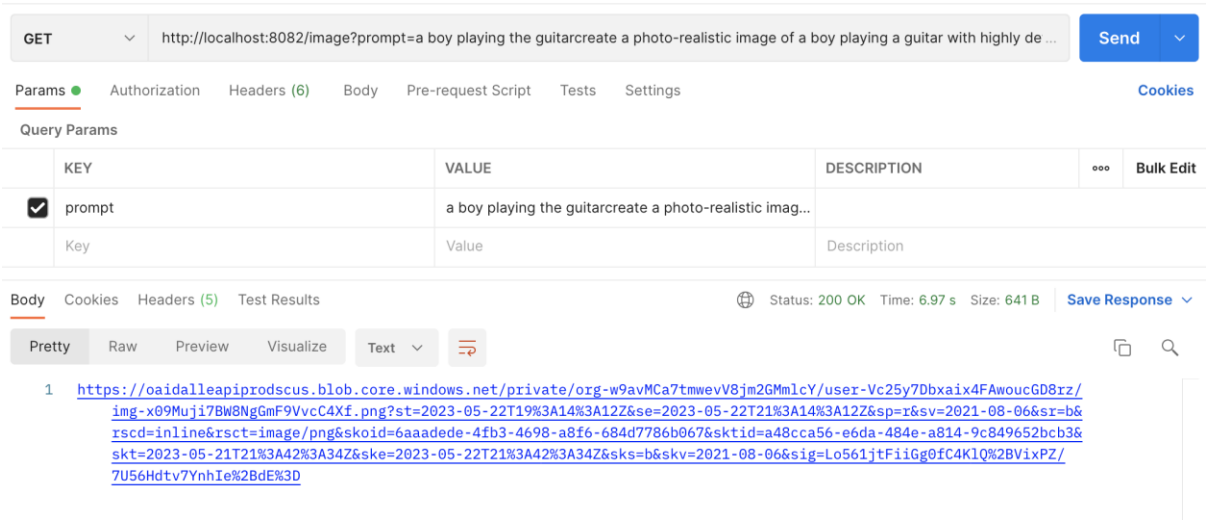

Рисунок 3.2.1 - Запит на генерування зображення

3.3 Аналітика:

дивитись актуальну інформацію про всі запити на OpenAi можна в особистому кабінеті на сайті OpenAi

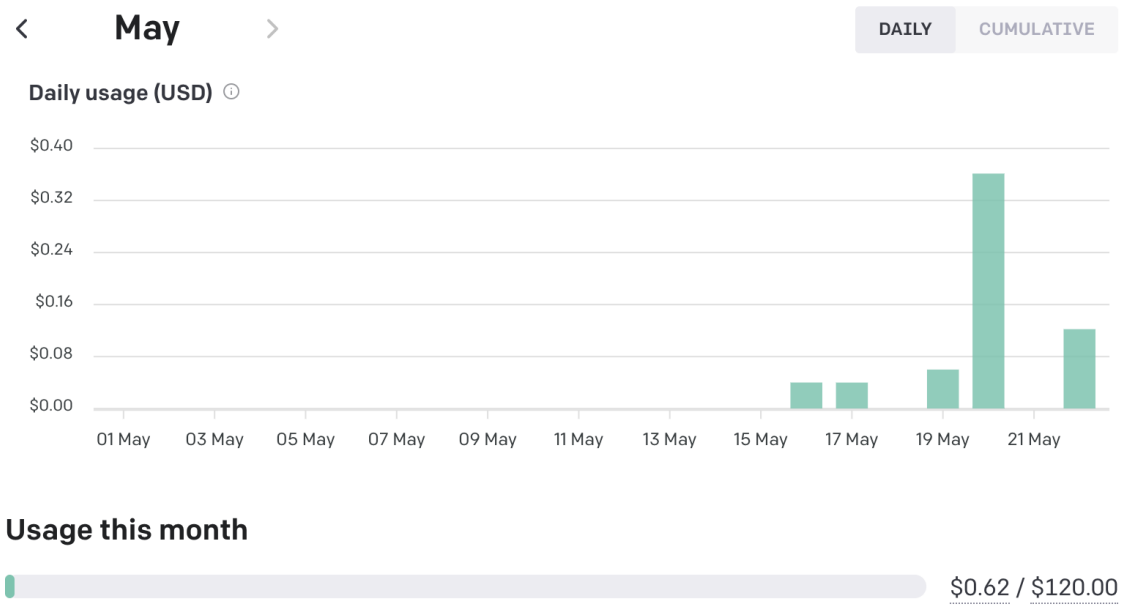

Рисунок 3.3.1 - Графік запитів на OpenAi

3.4 Огляд результатів:

В цьому розділі буде продемонстровано роботу написаного додатку:

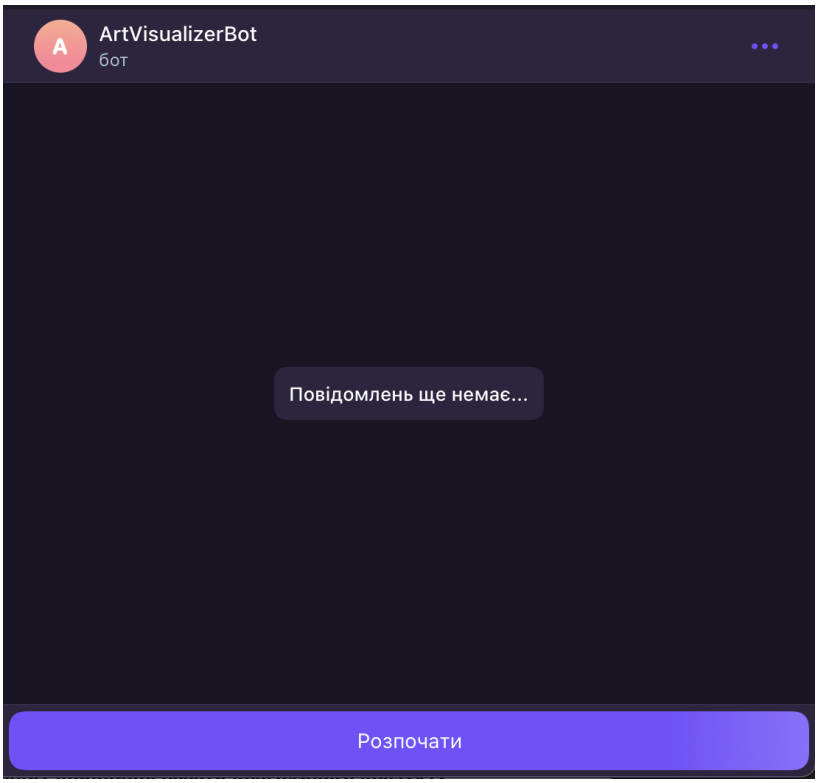

Рисунок 3.4.1 - Стратова сторінка взаємодії з ботом

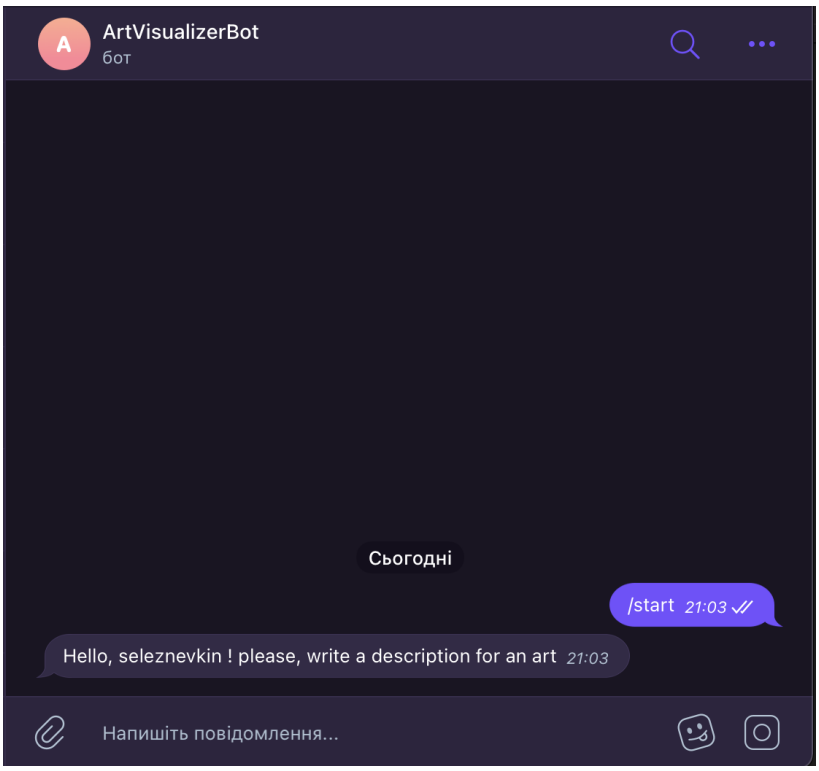

Рисунок 3.4.2 - Реакція на команду /start

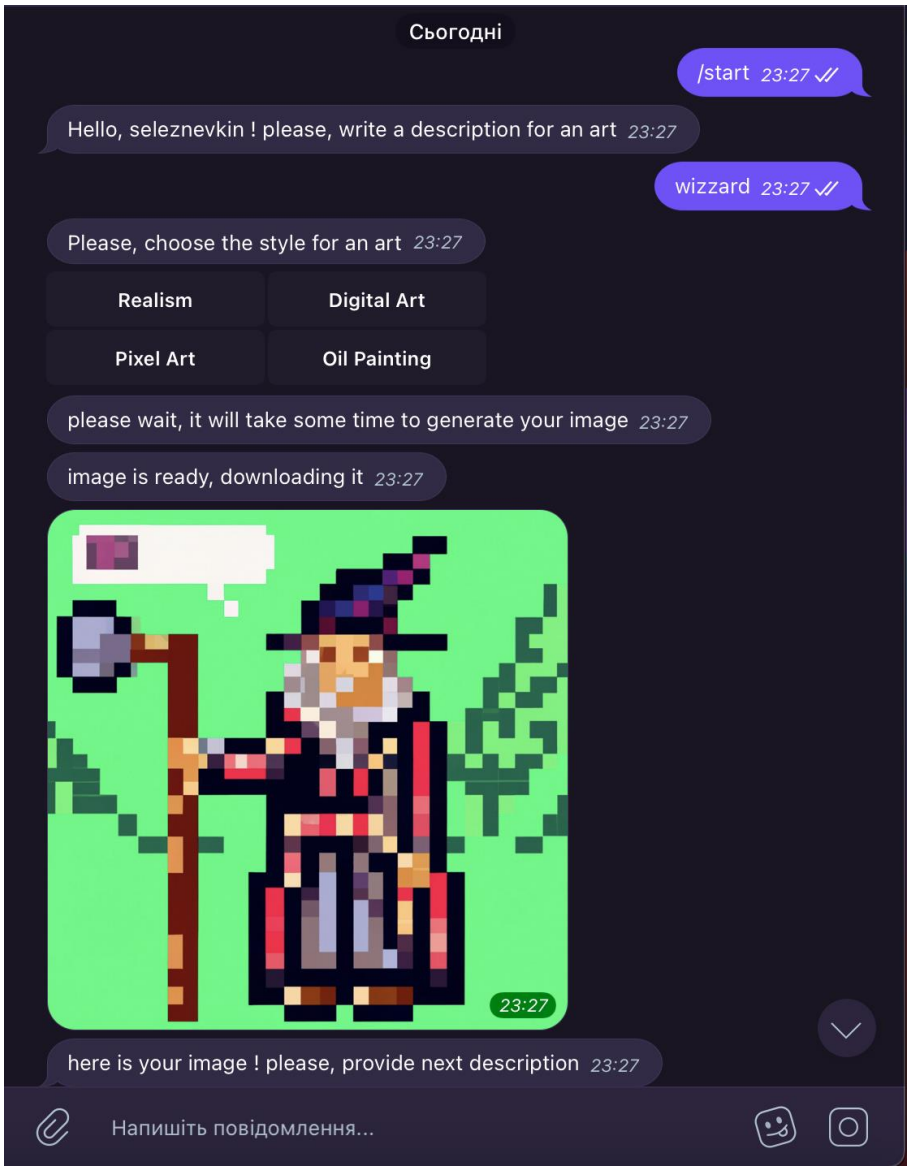

Рисунок 3.4.3 - Зображення за описом “wizzard” та стилем pixel art

Для прикладу: зображення згенероване за тим самим описом але зі стилем oil painting

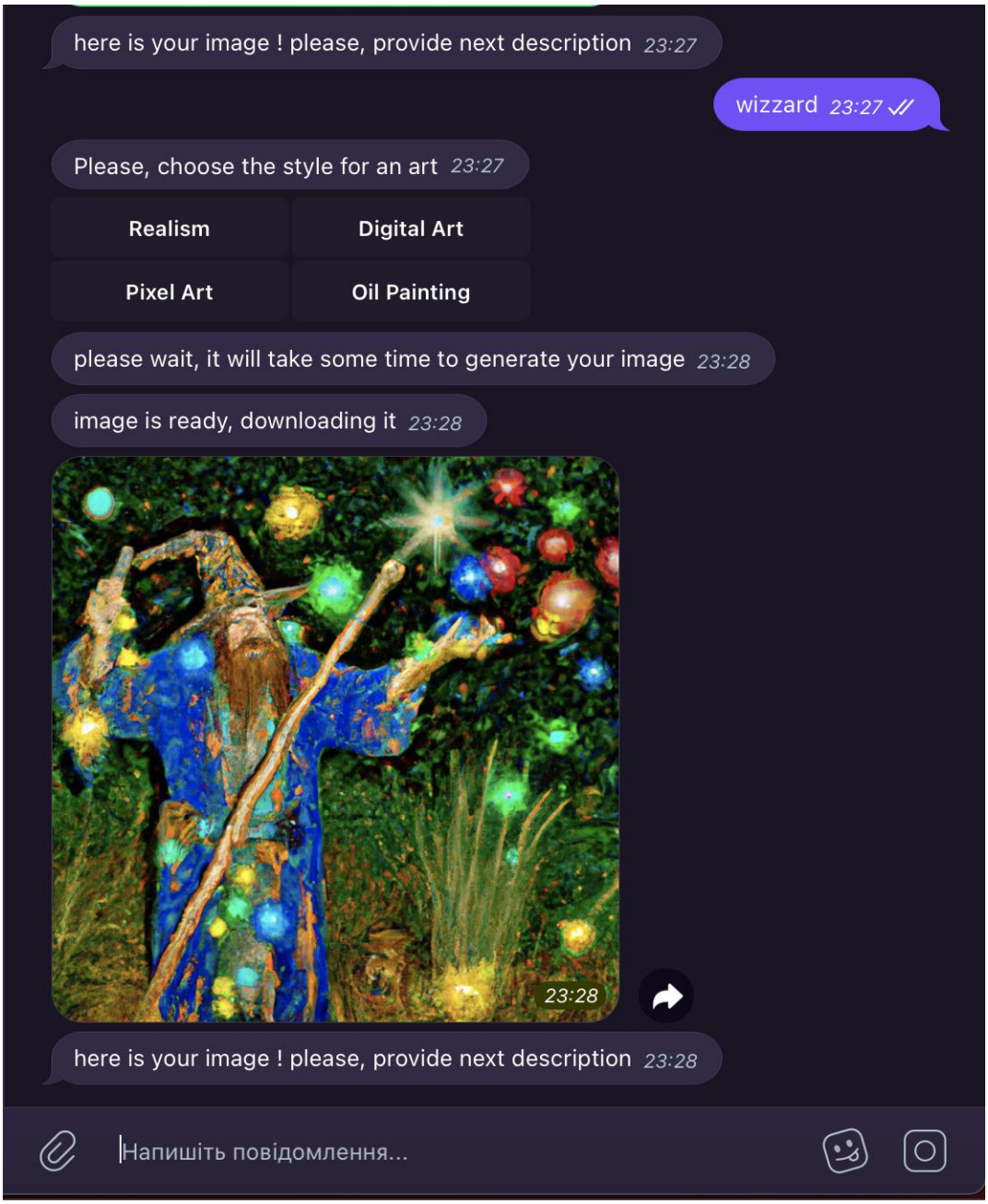

Рисунок 3.4.4 - Зображення за описом “wizzard” та стилем oil painting

Для наступного прикладу введемо однакові описи в розробленого бота та користувацький інтерфейс від DALL-E:

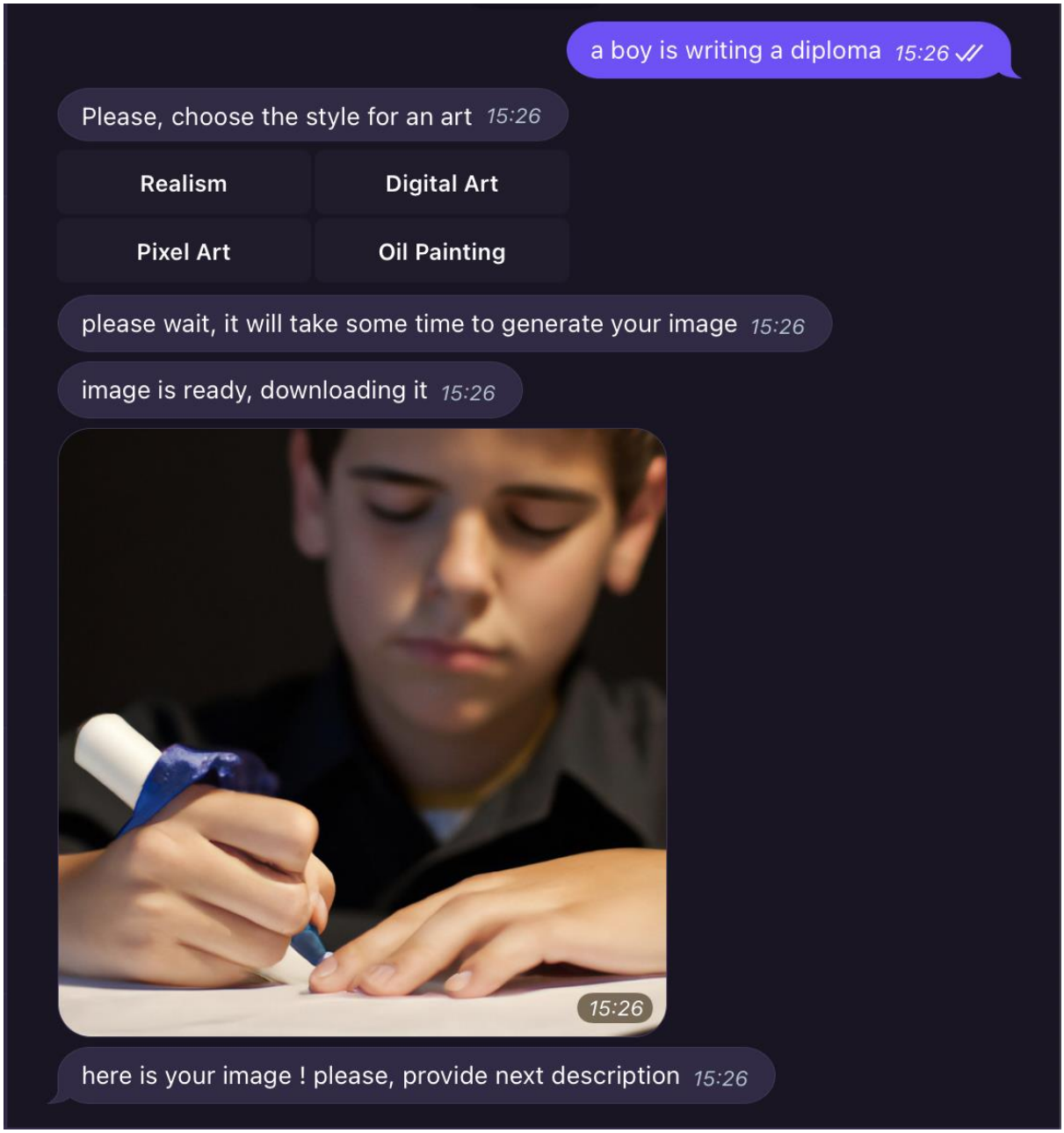

Рисунок 3.4.5 - Зображення згенероване з використанням бота за описом “a boy is writing a diploma” та стилем realism

Як бачимо, бот виконав поставлену задачу, без вимоги до написання відповідних вказівок зі сторони користувача

Тепер введемо той самий опис в готовий інтерфейс від DALL-E

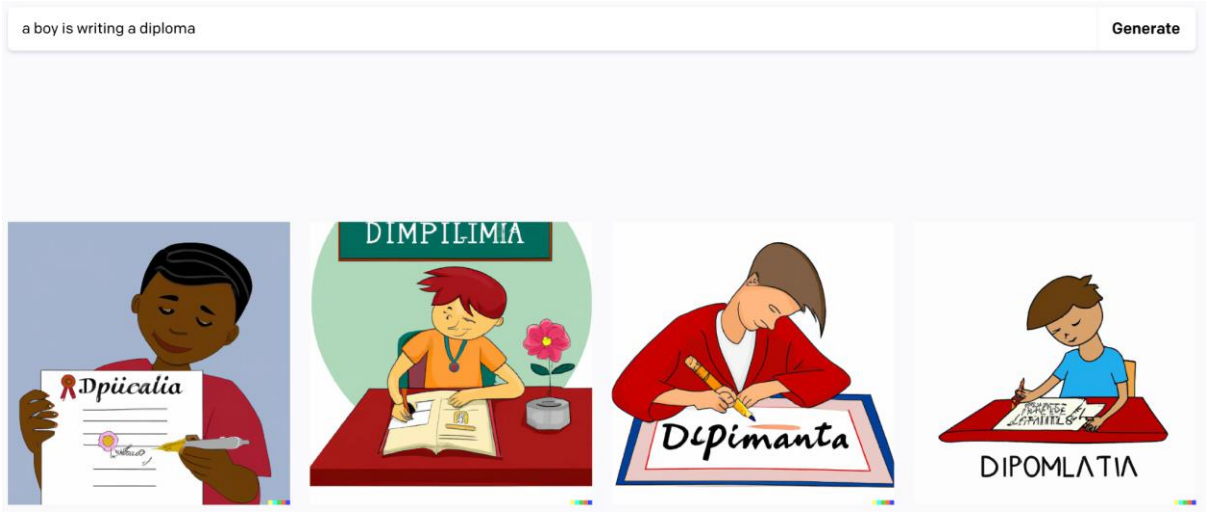

Рисунок 3.4.6 - Зображення згенероване з використанням готовий інтерфейс від DALL-E за описом “a boy is writing a diploma”

Як видно з результатів, в обох випадках користувачу були згенеровані зображення, що за змістом відповідають його запиту, але в першому випадку, задля генерації зображення в стилі “реалізм”, користувачу достатньо було натиснути одну кнопку, в той час як в другому випадку, користувачу довелось би писати вказівки самому, що є не тривіальною задачею.

Якщо у випадку “реалізму” DALL-E може випадковим чином генерувати зображення в цьому стилі:

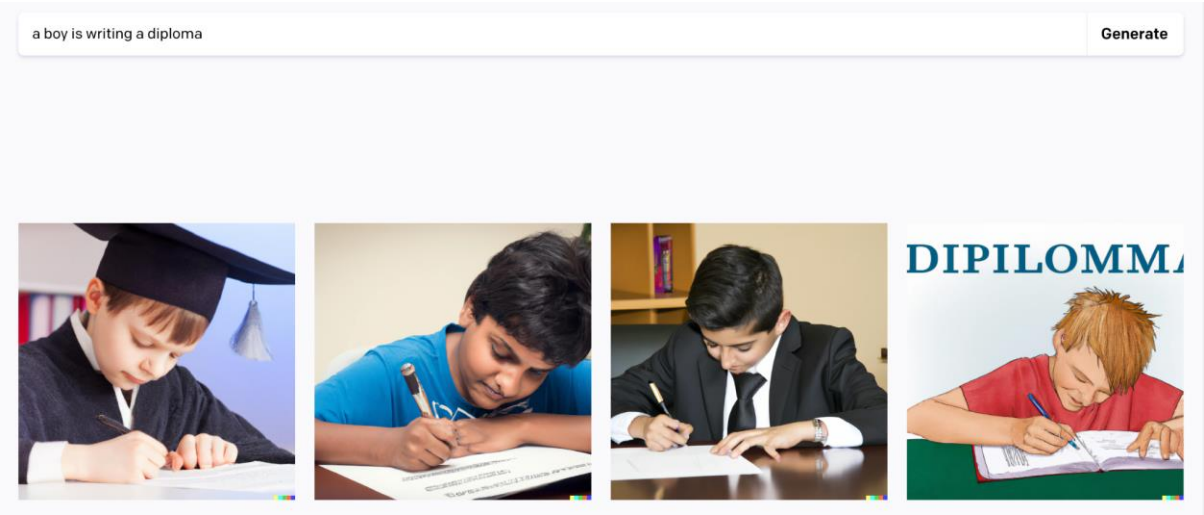

Рисунок 3.4.7 - Зображення згенероване з використанням готовий інтерфейс від DALL-E за описом “a boy is writing a diploma”

То, з більш конкретними запитами на стилістику без вичерпних вказівок не обійтись. Для прикладу розглянемо генерування зображення з однаковими описами, з наміром отримати арт у стилістиці “oil painting”

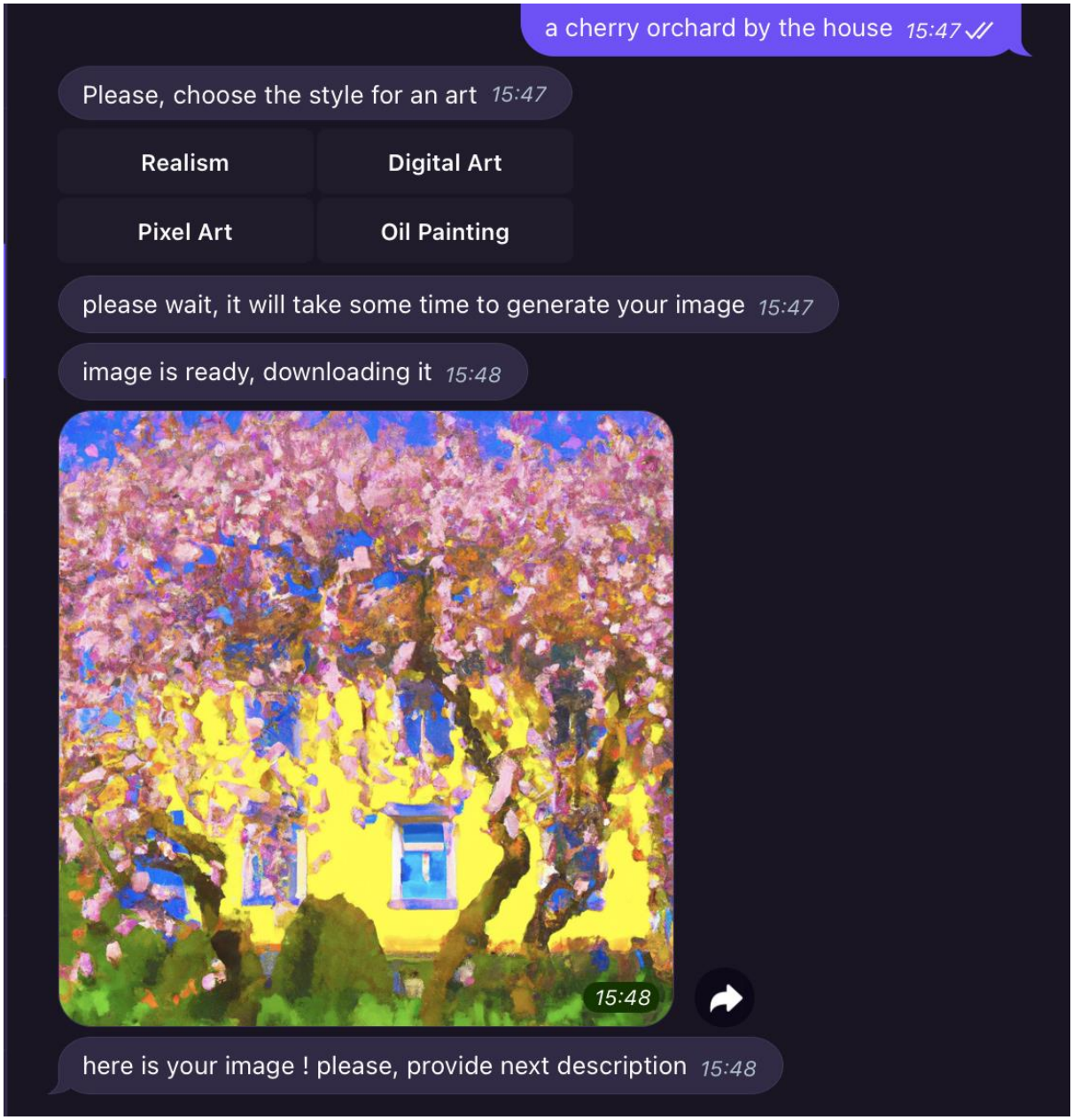

Рисунок 3.4.8 - Зображення згенероване з використанням бота за описом “a cherry orchard by the house” та стилем oil painting

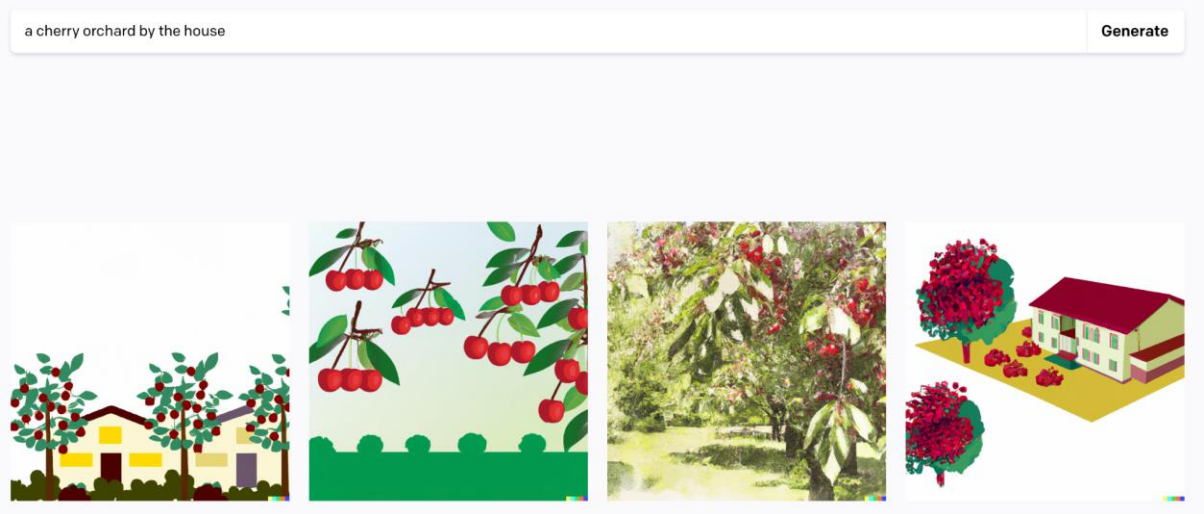

Рисунок 3.4.9 - Зображення згенероване з використанням готовий інтерфейс від DALL-E за описом “a cherry orchard by the house”

При повторних спробах згенерувати зображення за тим самим описом, результат жодного разу не виявився бажаним, з чого ми вкотре переконуємося, що без конкретних описів, тяжко, а, подекуди неможливо досягти бажаного результату в генеруванні зображень

ВИСНОВКИ

Проведено дослідження у галузі генерації зображень за допомогою штучного інтелекту з використанням розмовної мови.

Розроблено програмний продукт (Telegram бот) для візуалізації описового тексту у формі зображень з можливістю вибору стилістики, у тому числі, реалізовано інтеграцію з відповідними зовнішніми сервісами та API, зокрема, з Telegram API для забезпечення взаємодії з ботом та OpenAI API для генерування зображень, розроблено інтерфейсу користувача, що забезпечує зручну та інтуїтивно зрозумілу взаємодію з ботом, проведено тестування розробленої системи.

Розроблена система дозволяє користувачам генерувати зображення, описуючи їх у розмовній мові та обираючи бажану стилістику.

Перспективи подальшої розробки: впровадження нової стилістики зображень, вдосконалення алгоритмів генерації та покращення інтерфейсу користувача. Також можливе розширення функціоналу бота для підтримки інших платформ комунікації, що відкриває додаткові перспективи розвитку застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Baeldung [Електронний ресурс] - Режим доступу до ресурсу: https://www.baeldung.com/
2. OpenAi API documentation [Електронний ресурс] - Режим доступу до ресурсу: https://platform.openai.com/docs/introduction
3. Telegram API documentation [Електронний ресурс] - Режим доступу до ресурсу: https://core.telegram.org/api
4. OkHttp Documentation [Електронний ресурс] - Режим доступу до ресурсу: https://square.github.io/okhttp/
5. OpenAI-Java README.md by TheoKanning [Електронний ресурс] - Режим доступу до ресурсу: https://github.com/TheoKanning/openai-java#readme
6. MidJorney documentation [Електронний ресурс] - Режим доступу до ресурсу: https://docs.midjourney.com/
7. DALL-E documentation [Електронний ресурс] - Режим доступу до ресурсу: https://openai.com/product/dall-e-2
8. Stable Diffusion documentation [Електронний ресурс] - Режим доступу до ресурсу: https://stablediffusionapi.com/docs/
9. Apache common documentation [Електронний ресурс] - Режим доступу до ресурсу: https://commons.apache.org/
10. Кей Хортсман Core Java 12th edition, Oracle press, 2021, 944 с.
11. Крейг Воллс Spring in action 5th edition, 2018, 520 с.