

АНОТАЦІЯ

Дана магістерська кваліфікаційна робота містить опис проектування та розробки системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту.

У вступі обґрунтовано актуальність теми дипломної роботи, сформульовано мету та основні задачі досліджень, визначено наукову новизну і описано практичне значення отриманих результатів.

У першому розділі роботи описаний огляд сучасного стану процесу автоматизації створення контенту. Проведено аналіз 20 наукових джерел, що стосуються генеративного штучного інтелекту та соціальних мереж. На основі даного аналізу визначено, що пошук ефективних способів генерування контенту для соціальних мереж є перспективним завданням для вирішення. На підставі аналізу уточнено формулювання мети та задач дослідження.

Другий розділ дипломної роботи присвячений розробці методу інтеграції системи для створення контенту з генеративними моделями штучного інтелекту та соціальними мережами. Наведений короткий огляд OpenAI API та способу його інтеграції в проект. Також описано процес інтеграції в проект API соціальних мереж та можливості взаємодії з електронною поштою через мережеві протоколи. Описано революцію трансформерів у машинному навчанні. Проведено огляд моделей глибинного навчання на основі дифузії. Результатом другого розділу також є розроблена специфікація вимог до програмного забезпечення.

У третьому розділі детально розглянутий процес програмної реалізації веб-застосунку для створення контенту. Описана архітектура застосунку, включаючи використання клієнт-серверної моделі та MVC підходу, що забезпечує ефективне розділення бізнес-логіки та користувацького інтерфейсу. Особлива увага приділена інтеграції з генеративними моделями штучного інтелекту від OpenAI, такими як: GPT-4 та DALL-E-3, для автоматизації процесу створення контенту та його оптимізації під різні платформи соціальних мереж.

У четвертому розділі описано розробку ефективних підказок для OpenAI API, адаптацію контенту під різні платформи та методології тестування веб-додатку. Проведена оцінка ефективності використання веб-застосунку для генерації контенту. Розкрито практичне застосування системи в комерційній та особистій сферах, а також її значення для наукових досліджень у галузі машинного навчання та обчислювальної лінгвістики.

Список публікацій автора, що відносяться до даної роботи:

Веренько А. Система управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту [Електронний ресурс] / Артем Веренько // international scientific-practical conference science, education, technology and society: problems and prospects. – 2023. – С. 54–57. – Режим доступу: <http://www.economics.in.ua/2023/10/12-2023.html> (дата звернення: 16.11.2023).

ABSTRACT

This thesis describes the design and development of a content management system for social networks with the integration of modern generative artificial intelligence models. The introduction justifies the relevance of the thesis topic, formulates the purpose and main tasks of the research, defines the scientific novelty, and describes the practical significance of the obtained results.

The first chapter provides an overview of the current state of the art in content creation automation. An analysis of 20 scientific sources related to generative artificial intelligence and social networks is carried out. Based on this analysis, it is concluded that discovering effective methods for generating content for social networks is a promising task to address. The aim and objectives of the research are refined accordingly.

The second chapter of the thesis is dedicated to the development of a method for integrating a content creation system with generative artificial intelligence models and social networks. A brief overview of the OpenAI API and its integration method into the project is given. The process of integrating social network APIs into the project and the possibilities of interacting with email via network protocols are also described. The revolution of transformers in machine learning is discussed, along with a review of deep learning models based on diffusion. The result of the second chapter is also the developed software requirements specification.

The third chapter examines in detail the process of software implementation of a web application for content creation. The architecture of the application is described, including the use of the client-server model and the MVC approach, which ensures effective separation of business logic and user interface. Special attention is given to the integration with generative artificial intelligence models from OpenAI, such as GPT-4 and DALL-E-3, to automate the content creation process and optimize it for different social media platforms.

The fourth chapter describes the development of effective prompts for the OpenAI API, adaptation of content for different platforms, and methodologies for testing a web application. The effectiveness of using a web application for content

generation is evaluated. The practical application of the system in commercial and personal spheres, as well as its significance for scientific research in the fields of machine learning and computational linguistics, is also discussed.

List of the author's publications related to this work:

Verenko, A. "Content Management System for Social Networks with Integration of Modern Generative Artificial Intelligence Models" [Electronic resource] / Artem Verenko // International scientific-practical conference science, education, technology, and society: problems and prospects. – 2023. – P. 54–57. – Available at: <http://www.economics.in.ua/2023/10/12-2023.html> (accessed: 16.11.2023).

ЗАГАЛЬНА ХАРАКТЕРИСТИКА МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Дана магістерська кваліфікаційна робота містить опис проектування та розробки системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Розробка такої системи покликана відповісти на виклики сучасного інформаційного простору, де швидкість, обсяг та якість контенту відіграють ключову роль.

Робота включає аналіз сучасних тенденцій розвитку генеративних моделей штучного інтелекту, вибір конкретних моделей штучного інтелекту для їх інтеграції в проект, розробку концептуальної моделі системи, створення функціоналу для генерування та розповсюдження контенту а також оцінку ефективності розробленої системи.

У результаті розроблений веб-застосунок має принести користь не тільки науковій спільноті, пропонуючи нові знання в області застосування генеративних моделей в соціальних медіа, але й широкому колу користувачів, чийм завданням є створення контенту.

МЕТА І ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Мета даної магістерської кваліфікаційної роботи полягає у розробці оригінальної системи створення контенту для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту.

Для досягнення цієї мети в роботі вирішені наступні завдання:

- вивчення сучасних тенденцій в області генеративних моделей штучного інтелекту та їх застосування в соціальних мережах;
- розробка концептуальної моделі системи створення контенту, яка включає інтеграцію генеративних моделей;
- вибір для впровадження конкретних моделей штучного інтелекту;
- розробка функціоналу для взаємодії з користувачами, генерування, оптимізації та розповсюдження контенту;
- тестування та оцінка ефективності розробленої системи.

Об'єкт дослідження – процеси інтеграції сучасних генеративних моделей штучного інтелекту.

Предмет дослідження – метод і технологія інтеграції сучасних генеративних моделей штучного інтелекту з оригінальною системою створення контенту для соціальних мереж.

АКТУАЛЬНІСТЬ, ПРОБЛЕМАТИКА ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Поруч з більшістю галузей, на які так чи інакше вплинуло виникнення та розвиток генеративних моделей штучного інтелекту, особливої трансформації зазнає сфера соціальних медіа. Оскільки соціальні мережі за своєю суттю є платформами для генерування контенту від користувачів, сучасні генеративні моделі відкривають нові горизонти для автоматизації та оптимізації цього процесу. Вони здатні автоматично створювати текстовий та візуальний контент, пропонувати персоналізовані рекомендації контенту на основі індивідуальних інтересів користувачів, тим самим значно покращуючи його користувацький досвід. Це вказує на те, що розробка системи для створення контенту для соціальних мереж з інтеграцією генеративних моделей штучного інтелекту є актуальним завданням з широкими перспективами.

Система використовує інтеграцію з генеративними моделями штучного інтелекту, такими як: GPT-4 та DALL-E-3 від OpenAI, для створення текстового та візуального контенту, та Twitter, Telegram, Gmail API для його поширення.

Ключовим аспектом системи є її здатність генерувати контент за допомогою штучного інтелекту. Користувачі можуть створювати привабливі та оригінальні пости, використовуючи прості підказки. Це включає автоматизацію генерації заголовків, текстів, посилань та навіть візуального контенту.

Система має вбудовані механізми для адаптації контенту до різних соціальних мереж, зокрема: Twitter, Telegram та Gmail. Вона автоматично налаштовує контент під особливості кожної платформи, наприклад, короткі та

лаконічні твіти для Twitter та більш детальні повідомлення для Telegram та Gmail.

Система інтегрується з різними соціальними мережами та генеративними моделями штучного інтелекту, що дозволяє з легкістю публікувати контент на різних платформах. Це не лише спрощує процес розповсюдження контенту, але й забезпечує його відповідність стандартам та форматам кожної соціальної мережі.

Система має високі стандарти безпеки для захисту даних користувачів, включаючи шифрування даних та політики доступу.

Інтерфейс системи розроблено з акцентом на інтуїтивність та простоту використання, що робить процес створення та публікації контенту доступним для широкого кола користувачів.

Описані особливості системи відкривають широкі можливості її комерційного застосування. Так система може бути використана в маркетингових та рекламних стратегіях компаній, де допомагатиме автоматизувати та оптимізувати процеси створення контенту. Це сприятиме підвищенню залучення аудиторії та популяризації бренду. Для блогерів та інфлюенсерів система є зручним інструментом для підтримки регулярної активності у соціальних мережах. Малі та середні підприємства можуть значно виграти від її використання, оскільки вона дозволяє самостійно керувати присутністю в соціальних мережах без необхідності наймати додаткових спеціалістів.

Розроблена система також має значний потенціал для дослідницьких цілей. Її компоненти, такі як: адаптери контенту та налаштовані API соціальних мереж, можуть бути використані для вивчення специфічних аспектів цифрової взаємодії та комунікації. Це відкриває можливості для експериментування в областях обчислювальної лінгвістики, аналізу соціальних медіа та дослідження впливу штучного інтелекту на медіа-контент.

Таким чином, розроблена система стає не лише інструментом для

ефективного управління контентом у соціальних мережах, але й цінним ресурсом для наукових досліджень та інноваційного розвитку. Вона демонструє як практичну цінність в комерційному використанні, так і значущість для наукової спільноти, відкриваючи нові горизонти для досліджень та розвитку в галузі штучного інтелекту та цифрової комунікації.

ОСНОВНИЙ ЗМІСТ ТА РЕЗУЛЬТАТИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Вибір моделей штучного інтелекту для створення контенту

Під час огляду наявних провайдерів генеративних моделей штучного інтелекту, опираючись на критерії: зручність імплементації та відповідність поставленим завданням, вибір зупинився на OpenAI. Цей провайдер надає зручний високорівневий API до широкого спектру моделей різного призначення. Виклики даного API є платними, але взамін отримуємо високу якість та гнучкість, що передбачено донною роботою. Використовуються моделі: GPT-4 – для генерації природніх текстів та DALL-E-3 – для генерації зображень.

2. Опис інтеграції в проект API соціальних мереж

API соціальних мереж дозволяють інтегрувати функції соціальних платформ у різні програми, поліпшуючи залученість користувачів та персоналізацію контенту. Інтеграція починається з розуміння документації API та реєстрації додатку для отримання ключів доступу. Важливі етапи включають встановлення системи авторизації (зазвичай через OAuth), формування та обробку запитів до API і ретельне тестування.

Інтеграція з Twitter API включає реєстрацію додатку, отримання ключів, використання OAuth для публікації твітів та обробку відповідей. Інтеграція з Telegram API має схожі кроки, але відрізняється можливостями публікації контенту в чатах і каналах.

Ефективне використання API соціальних мереж відкриває можливості для автоматизованого розповсюдження контенту та оптимізації маркетингових стратегій.

3. Розробка методу системи для створення контенту

Розробка методу системи для створення контенту є важливим кроком у використанні штучного інтелекту для ефективної роботи з соціальними мережами. Діаграма послідовності розкриває ключові етапи цього процесу від управління профілями до публікації постів і демонструє здатність системи до генерації, редагування та адаптації контенту.

Процеси включають в себе керування профілем користувача, створення та генерування контенту, зберігання чернеток постів і публікацію постів в соціальних мережах. Кожен з цих процесів має ряд взаємодій між різними компонентами системи.

У розділі "Управління профілем користувача", користувач взаємодіє з системою управління контентом для зміни налаштувань свого профілю. CMS зберігає ці зміни в базі даних і повідомляє користувачу про успішне оновлення профілю.

У розділі "Редактор постів та Генерування контенту" користувач створює новий пост в редакторі постів. Редактор постів взаємодіє з моделлю штучного інтелекту для генерування контенту і повертає цей контент користувачу.

У розділі "Зберігання чернеток постів" користувач просить систему зберегти пост як чернетку. Система зберігає чернетку в базу даних і повідомляє користувачу про успішне збереження.

У розділі "Публікація постів" користувач просить систему опублікувати пост. Система використовує адаптер контенту для адаптації контенту до вимог соціальної мережі, відправляє адаптований контент у соціальну мережу і повідомляє користувачу про успішну публікацію.

4. Реалізація та тестування програмного продукту

У процесі розробки системи створення контенту для соціальних мереж з інтегрованими сучасними генеративними моделями штучного інтелекту від OpenAI було задіяно передові технології для досягнення найкращої

продуктивності та адаптивності. В основу системи лягла платформа .NET 7 від Microsoft, обрана через її високу продуктивність та крос-платформеність, що сприяє створенню масштабованих та легко підтримуваних веб-застосунків.

Архітектура системи побудована на основі MVC моделі, яка дозволяє чітко розділити бізнес-логіку від користувацького інтерфейсу. Важливу роль у цій архітектурі відіграють репозиторії, які слугують абстракцією для доступу до даних, ізолюючи бізнес-логіку від низькорівневих операцій з базою даних. Це полегшує роботу з базою даних та забезпечує ефективну реалізацію CRUD операцій. Entity Framework Core використовується для зручного доступу до даних. Аспекти безпеки обробляються через ASP.NET Core Identity.

Інтеграція з зовнішніми сервісами, такими як: Telegram, Twitter, та API OpenAI, значно розширює функціонал системи, надаючи можливості для публікації та розповсюдження контенту у соціальних мережах, а також для генерації унікального контенту.

Фронтенд частина реалізована за допомогою технологій Bootstrap та Razor Views, які дозволили створити інтерактивні користувацькі інтерфейси.

Таким чином, розроблена система представляє собою сучасний веб-застосунок, який відповідає сучасним вимогам веб-розробки та забезпечує безпечний та ефективний досвід користувача.

5. Оцінка ефективності веб-застосунку та його практичного і наукового потенціалу

Оцінка ефективності розробленого веб-застосунку для генерації контенту в соціальних мережах виявила високу продуктивність та універсальність. Аналіз показав, що швидкість генерації контенту становить у середньому 4.5 секунди на запит, що відповідає високим стандартам ефективності. Якість згенерованого контенту оцінена як висока, задовольняючи більшість вимог користувачів. Зручність інтерфейсу застосунку була класифікована як "дуже зручна", забезпечуючи легкість у навігації та використанні. Дотримання тематики згенерованого контенту оцінено як високе, що свідчить про точне відтворення заданих тем та

контекстів. Гнучкість налаштувань системи дозволяє користувачам легко адаптувати застосунок під свої індивідуальні потреби.

У практичному застосуванні система використовується в маркетингових та рекламних стратегіях, сприяючи збільшенню залученості аудиторії та популяризації брендів. Застосунок ефективно використовується блогерами та інфлюенсерами, а також малими та середніми підприємствами для управління присутністю у соціальних мережах, зменшуючи необхідність у додаткових спеціалістах.

З наукової точки зору, застосунок відкриває шляхи для досліджень у сферах машинного навчання, обчислювальної лінгвістики, аналізу соціальних медіа та взаємодії людини з комп'ютером. Він надає можливості для вивчення персоналізації контенту та його впливу на аудиторію. Презентація методики розробки системи на науковій конференції підкреслює її значущість та потенціал для наукових досліджень у сфері цифрової комунікації та технологій.

Відповідно до поставлених завдань у магістерській кваліфікаційній роботі були досягнуті наступні результати:

- Було проведено дослідження сучасних технологій та підходів до генерування контенту для соціальних мереж за допомогою генеративних моделей штучного інтелекту;
- Створено модель системи для генерації контенту у соціальних мережах, що інтегрує генеративні моделі штучного інтелекту;
- Вибрано та інтегровано моделі GPT-4 та DALL-E-3 від OpenAI для оптимізації процесу створення та редагування контенту;
- Розроблено веб-застосунок з інтуїтивно зрозумілим інтерфейсом, інтегрованими інструментами для генерації, оптимізації та розповсюдження контенту;
- Успішно проведено тестування системи методом «чорної скриньки», що підтвердило її високу продуктивність та ефективність;

– Розроблено підказки для OpenAI API та оцінено ефективність веб-застосунку в контексті генерації контенту;

– Визначено практичні можливості застосування розробленої системи та проаналізовано потенційні напрямки для майбутніх досліджень у цій сфері.

Ці результати свідчать про успішне вирішення поставлених завдань.

ВИСНОВКИ

Отже, у даній магістерській кваліфікаційній роботі була розроблена система управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. В роботі було проведено глибокий аналіз сучасних тенденцій у розвитку генеративних моделей. Розроблено концептуальну модель системи, що включає генерацію, оптимізацію та розповсюдження контенту.

Важливим результатом роботи є створення функціонального веб-застосунку з інтуїтивно зрозумілим інтерфейсом, що забезпечує ефективну взаємодію з користувачами та можливість легкого управління контентом. Також були успішно інтегровані GPT-4 та DALL-E-3 моделі для створення текстового та візуального контенту, що дозволило підвищити якість та різноманітність генерованого матеріалу.

Оцінка ефективності розробленого веб-застосунку показала його високу продуктивність, демонструючи високу якість згенерованого матеріалу. Виявлено, що система має значний потенціал як у комерційному застосуванні так і в наукових дослідженнях, відкриваючи нові можливості для маркетингових стратегій, автоматизації та оптимізації створення контенту.

ПЕРЕЛІК ГРАФІЧНОГО МАТЕРІАЛУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

UML-діаграми послідовності, варіантів використання, діяльності системи, класів інтегрованих сторонніх сервісів і класів фреймворку тестування проекту. Екрани реалізованої програми.

OVERVIEW OF THE MASTER'S QUALIFICATION WORK

This master's thesis presents the design and development of a content management system for social networks with the integration of modern generative artificial intelligence models. The development of such a system aims to address the challenges of the modern information space, where speed, volume, and quality of content play a crucial role.

The work includes an analysis of current trends in generative artificial intelligence models, the selection of specific AI models for their integration into the project, the development of a conceptual model of the system, the creation of functionality for generating and distributing content, and the evaluation of the effectiveness of the developed system.

As a result, the developed web application is expected to benefit not only the scientific community, offering new knowledge in the field of applying generative models in social media but also a wide range of users whose task is content creation.

GOAL AND OBJECTIVE OF MASTER'S QUALIFICATION WORK

The aim of this master's thesis is to develop an original content creation system for social networks with the integration of modern generative artificial intelligence models.

To achieve this goal, the following tasks were accomplished:

- studying current trends in generative artificial intelligence models and their application in social networks;
- developing a conceptual model of the content creation system, which includes the integration of generative models;
- selecting specific artificial intelligence models for implementation;
- developing functionality for interaction with users, generating, optimizing, and distributing content;
- testing and evaluating the effectiveness of the developed system.

The object of research is the processes of integrating modern generative artificial intelligence models.

The subject of research is the method and technology of integrating modern generative artificial intelligence models with the original content creation system for social networks.

ACTUALITY, PROBLEMS AND PRACTICAL VALUE OF THE MASTER'S QUALIFICATION WORK

Alongside most sectors impacted by the emergence and development of generative artificial intelligence models, the social media sphere is undergoing a particularly transformative change. As social networks are inherently platforms for user-generated content, modern generative models open new horizons for automating and optimizing this process. They are capable of automatically creating textual and visual content, offering personalized content recommendations based on individual user interests, significantly enhancing the user experience. This indicates, that developing a content creation system for social networks with the integration of generative artificial intelligence models is a relevant task with broad prospects.

The system utilizes integration with generative artificial intelligence models such as OpenAI's GPT-4 and DALL-E-3 for creating textual and visual content and uses Twitter, Telegram, and Gmail APIs for content distribution.

A key aspect of the system is its ability to generate content using artificial intelligence. Users can create attractive and original posts using simple prompts. This includes the automation of generating titles, texts, hashtags and even visual content.

The system has built-in mechanisms for adapting content to various social networks, specifically Twitter, Telegram, and Gmail. It automatically adjusts the content to suit each platform's unique features, such as concise tweets for Twitter and more detailed messages for Telegram and Gmail.

The system integrates with various social networks and generative artificial intelligence models, allowing for easy content publication across different platforms. This not only simplifies the content distribution process but also ensures its compliance with the standards and formats of each social network.

The system maintains high security standards to protect user data, including data encryption and access policies.

The interface of the system is designed with a focus on intuitiveness and ease of use, making the process of creating and publishing content accessible to a wide range of users.

The described features of the system open wide possibilities for its commercial application. For instance, the system can be used in marketing and advertising strategies of companies, helping to automate and optimize content creation processes. This will contribute to increasing audience engagement and brand popularization. For bloggers and influencers, this system is a convenient tool for maintaining regular activity on social networks. Small and medium enterprises can significantly benefit from its use, as it allows them to manage their social media presence independently without the need to hire additional specialists.

The developed system also has significant potential for research purposes. Its components, such as content adapters and customized social network APIs, can be used to study specific aspects of digital interaction and communication. This opens opportunities for experimentation in the fields of computational linguistics, social media analysis and research on the impact of artificial intelligence on media content.

Thus, the developed system becomes not only a tool for effective content management in social networks but also a valuable resource for scientific research and innovative development. It demonstrates practical value in commercial use and significance for the scientific community, opening new horizons for research and development in the field of artificial intelligence and digital communication.

MAIN CONTENT AND RESULTS OF THE MASTER'S QUALIFICATION WORK

1. Selection of Artificial Intelligence Models for Content Creation

During the review of available generative artificial intelligence model providers, based on criteria such as ease of implementation and alignment with set objectives, OpenAI was chosen. This provider offers a convenient high-level API to a wide range of models for various purposes. Although the API calls are paid, they

offer high quality and flexibility, which aligns with the objectives of this work. Models used include GPT-4 for generating natural texts and DALL-E-3 for image generation.

2. Description of Social Network API Integration in the Project

Social network APIs allow the integration of social platform functionalities into various applications, improving user engagement and content personalization. Integration begins with understanding the API documentation and registering the application to obtain access keys. Key steps include establishing an authorization system (usually through OAuth), formulating and processing API requests and thorough testing.

Integration with the Twitter API includes app registration, key acquisition, using OAuth for tweet publication, and response handling. Integration with the Telegram API follows similar steps but differs in content publication capabilities in chats and channels.

Effective use of social network APIs opens opportunities for automated content distribution and optimization of marketing strategies.

3. Development of the System Method for Content Creation

Developing the system method for content creation is a crucial step in using artificial intelligence effectively with social networks. A sequence diagram reveals key stages of this process, from profile management to posting publications, demonstrating the system's ability to generate, edit and adapt content.

Processes include user profile management, content creation and generation, saving draft posts, and publishing posts on social networks. Each of these processes involves a series of interactions between different system components.

In the "User Profile Management" section, the user interacts with the content management system to change profile settings. The CMS stores these changes in the database and notifies the user of the successful profile update.

In the "Post Editor and Content Generation" section, the user creates a new post in the post editor. The post editor interacts with the AI model to generate content and returns this content to the user.

In the "Saving Draft Posts" section, the user requests the system to save a post as a draft. The system stores the draft in the database and informs the user of successful saving.

In the "Post Publication" section, the user requests the system to publish a post. The system uses a content adapter to adapt the content to the social network's requirements, sends the adapted content to the social network and informs the user about the successful publication.

4. Implementation and Testing of the Software Product

In developing the content creation system for social networks with integrated modern generative artificial intelligence models from OpenAI, cutting-edge technologies were employed to achieve optimal performance and adaptability. The system is built on the Microsoft .NET 7 platform, chosen for its high performance and cross-platform capabilities, facilitating scalable and easily maintainable web applications.

The system's architecture is based on the MVC model, which allows a clear separation of business logic from the user interface. Repositories play a crucial role in this architecture, serving as an abstraction for data access, isolating business logic from low-level database operations. This simplifies working with databases and ensures efficient implementation of CRUD operations. Entity Framework Core is used for convenient data access. Security aspects are handled through ASP.NET Core Identity.

Integration with external services like Telegram, Twitter, and OpenAI's API significantly expands the system's functionality, providing capabilities for publishing and distributing content on social networks and generating unique content.

The frontend is implemented using Bootstrap and Razor Views technologies, enabling the creation of interactive user interfaces.

Thus, the developed system represents a modern web application that meets contemporary web development requirements and provides a secure and effective user experience.

5. Evaluation of the Web Application's Efficiency and its Practical and Scientific Potential

The effectiveness of the developed web application for content generation in social networks was assessed to be highly productive and versatile. Analysis revealed that the average content generation speed is 4.5 seconds per request, meeting high efficiency standards. The quality of the generated content was rated as high, satisfying most user requirements. The convenience of the application's interface was classified as "very user-friendly", ensuring ease of navigation and usage. The adherence to the themes of the generated content was rated highly, indicating accurate reproduction of the given themes and contexts. The system's flexible settings allow users to easily adapt the application to their individual needs.

In practical application, the system is used in marketing and advertising strategies, enhancing audience engagement and brand popularization. The application is effectively utilized by bloggers and influencers, as well as small and medium-sized enterprises for managing their social media presence, reducing the need for additional specialists.

From a scientific perspective, the application paves the way for research in the fields of machine learning, computational linguistics, social media analysis and human-computer interaction. It provides opportunities for studying personalized content and its impact on audiences. The presentation of the system development methodology at a scientific conference underscores its significance and potential for scientific research in the field of digital communication and technology.

According to the objectives set in the master's thesis, the following results were achieved:

- a study of contemporary technologies and approaches to content generation for social networks using artificial intelligence generative models was conducted;
- a system model for content generation in social networks integrating artificial intelligence generative models was created;
- GPT-4 and DALL-E-3 models from OpenAI were selected and integrated to optimize the process of content creation and editing;

- a web application with an intuitive interface and integrated tools for generation, optimization and dissemination of content was developed;
- successful "black box" testing of the system confirmed its high productivity and efficiency;
- prompts for the OpenAI API were developed and the efficiency of the web application in the context of content generation was evaluated;
- the practical applications of the developed system were identified and potential directions for future research in this area were analyzed.

These results indicate the successful resolution of the set objectives.

CONCLUSIONS

Therefore, in this master's thesis, a content management system for social networks with the integration of modern artificial intelligence generative models was developed. A deep analysis of current trends in the development of generative models was conducted. A conceptual model of the system, including generation, optimization and dissemination of content, was developed.

An important result of the work is the creation of a functional web application with an intuitive interface that provides effective user interaction and easy content management. GPT-4 and DALL-E-3 models were successfully integrated for the creation of textual and visual content, which enhanced the quality and diversity of the generated material.

The assessment of the developed web application's effectiveness demonstrated its high productivity, showcasing the high quality of the generated material. It was found that the system has significant potential both in commercial application and scientific research, opening new possibilities for marketing strategies, automation and optimization of content creation.

LIST OF GRAPHICAL MATERIAL FOR THE MASTER'S QUALIFICATION WORK

UML sequence diagrams, use case diagrams, system activity diagrams, integrated third-party services class diagram and framework testing project class diagram. Screens of the implemented software.

ЗМІСТ

ВСТУП.....	25
РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЦЕСУ СТВОРЕННЯ КОНТЕНТУ ДЛЯ СОЦІАЛЬНИХ МЕРЕЖ.....	27
1.1. Оцінка потенціалу інтеграції генеративних моделей штучного інтелекту в систему для створення контенту.....	27
1.2. Критичний аналіз літературних джерел	30
1.3. Постановка проблеми та її обґрунтування	36
1.4. Формулювання мети і задач дослідження.....	37
1.5. Висновки	38
РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ІНТЕГРАЦІЇ СИСТЕМИ ДЛЯ СТВОРЕННЯ КОНТЕНТУ З ГЕНЕРАТИВНИМИ МОДЕЛЯМИ ШТУЧНОГО ІНТЕЛЕКТУ ТА СОЦІАЛЬНИМИ МЕРЕЖАМИ	39
2.1 Революція трансформерів у машинному навчанні	39
2.2 Огляд моделей глибинного навчання на основі дифузії	41
2.3 Порівняння доступних API генеративних моделей та оцінка можливостей їх інтеграції у проект.....	43
2.4 Опис архітектури веб-додатку для створення контенту.....	44
2.5 Опис структури веб-застосунку для створення контенту	47
2.6 Огляд OpenAI API та способу його інтеграції в проект	49
2.7 Опис інтеграції в проект API соціальних мереж.....	51
2.8 Застосування Інтернет-протоколів для управління електронною поштою	53
2.9 Розробка методу системи для створення контенту.....	55
2.10 Специфікація вимог до програмного забезпечення.....	56
2.11 Висновки	59
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ СТВОРЕННЯ КОНТЕНТУ	61

3.1 Проектування програмного рішення.....	61
3.2 Опис технологій, використаних у проекті	62
3.3 Програмна реалізація основних компонентів платформи.....	63
3.4 Програмна реалізація інтеграцій зі сторонніми сервісами	68
3.5 Тестування розробленого веб-застосунку.....	72
3.6 Висновки	79
РОЗДІЛ 4. ДОСЛІДЖЕННЯ ПЛАТФОРМИ СТВОРЕННЯ КОНТЕНТУ ДЛЯ СОЦІАЛЬНИХ МЕРЕЖ З ІНТЕГРАЦІЄЮ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ	80
4.1 Розробка підказок для генерування контенту з використанням OpenAI API	80
4.2 Оцінка ефективності веб-застосунку для генерації контенту	83
4.3 Можливості практичного застосування	87
4.4 Наукові перспективи	88
4.5 Результати виконання кваліфікаційної роботи.....	89
4.6 Висновки	91
ВИСНОВКИ	92
СПИСОК ЛІТЕРАТУРИ	93
ДОДАТКИ	96
Додаток А. UML діаграми	96
Додаток Б. PlantUML код UML діаграм	99
Додаток В. Лістинг базового функціоналу веб-застосунку	101
Додаток Д. Лістинг інтеграції сервісів.....	115
Додаток Е. Лістинг фреймворку автоматизованого тестування.....	120
Додаток Ж. Екрани реалізованої програми	125

ВСТУП

Сьогодні неможливо заперечити значний прогрес технологій в області штучного інтелекту. Вирішення завдань, які раніше вважалися такими, що погано піддаються автоматизації, особливо тих, які включають елементи творчості, такі як: створення текстів, зображень, музики або навіть відео – тепер стають все більш доступними завдяки стрімкому розвитку генеративних моделей різного рівня складності та якості. Використовуючи методологію глибинного навчання та доступ до обширних наборів даних, ці моделі здатні генерувати високоякісний контент, який у багатьох випадках складно відрізнити від людської творчості. Це підкреслює значний потенціал, які дані моделі представляють для різноманітних сфер застосування. Інтеграція цих передових технологій у власні проекти сьогодні є пріоритетним завданням як для найбільших корпорацій, таких як: Google, Microsoft, Meta чи Tesla, так і для інноваційних стартапів, що намагаються демонструвати передовий функціонал, незважаючи на обмежені ресурси.

Поруч з більшістю галузей, на які так чи інакше вплинуло виникнення та розвиток генеративних моделей штучного інтелекту, особливої трансформації зазнає сфера соціальних медіа. Оскільки соціальні мережі за своєю суттю є платформами для генерування контенту від користувачів, сучасні генеративні моделі відкривають нові горизонти для автоматизації та оптимізації цього процесу. Вони здатні автоматично створювати текстовий, візуальний та відеоконтент, пропонувати персоналізовані рекомендації контенту на основі індивідуальних інтересів користувачів, тим самим значно покращуючи його користувацький досвід.

Враховуючи це, мета даної магістерської кваліфікаційної роботи полягає у розробці оригінальної системи створення контенту для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Розробка такої системи покликана відповісти на виклики сучасного інформаційного простору, де швидкість, обсяг та якість контенту відіграють ключову роль. Генеративні моделі – це потужний інструмент, що дозволяє не

лише спрощувати процес створення контенту, а й забезпечує можливість його автоматизації, персоналізації та оптимізації.

Для досягнення цієї мети в магістерській кваліфікаційній роботі будуть вирішені наступні завдання:

- вивчення сучасних тенденцій в області генеративних моделей штучного інтелекту та їх застосування в соціальних мережах;
- розробка концептуальної моделі системи створення контенту, яка включає інтеграцію генеративних моделей;
- вибір для впровадження конкретних моделей штучного інтелекту;
- розробка функціоналу для взаємодії з користувачами, генерування, оптимізації та розповсюдження контенту;
- тестування та оцінка ефективності розробленої системи.

Об'єкт дослідження – процеси інтеграції сучасних генеративних моделей штучного інтелекту.

Предмет дослідження – метод і технологія інтеграції сучасних генеративних моделей штучного інтелекту з оригінальною системою створення контенту для соціальних мереж.

У результаті розробка такої системи має принести користь не тільки науковій спільноті, пропонуючи нові знання в області застосування генеративних моделей в соціальних медіа, але й широкому колу користувачів соціальних медіа, які отримають покращені можливості для взаємодії з контентом.

Слід зазначити, що це дослідження є спробою систематизованого вивчення процесу впровадження генеративних моделей штучного інтелекту в систему для створення контенту для соціальних мереж, що демонструє актуальність та новизну даної роботи. Розроблений програмний продукт дозволить полегшити створення контенту надаючи користувачу інструменти для автоматизації дано процесу.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЦЕСУ СТВОРЕННЯ КОНТЕНТУ ДЛЯ СОЦІАЛЬНИХ МЕРЕЖ

1.1. Оцінка потенціалу інтеграції генеративних моделей штучного інтелекту в систему для створення контенту

Сучасний світ неможливо уявити без невпинно зростаючої кількості різноманітного контенту, що наповнює світовий інформаційний простір, стимулює розвиток бізнесу, освіти, науки або культури та містить цінність у вигляді спроможності інформувати, розважати, навчати чи об'єднувати людей. Контент є інструментом залучення та утримання аудиторії, популяризації брендів та стимулювання продажів, а отже, відіграє ключову роль у формуванні глобальних комунікацій та взаємодій.

Контент часто оцінюється на основі характеристик, серед яких ключовими є: якість, достовірність та актуальність. Ці показники стають віддзеркаленням інформаційної цінності контенту та вказують на ступінь його відповідності потребам цільової аудиторії. Дотримання високих стандартів не лише збільшує довіру користувачів до джерела інформації, але й допомагає виділитися серед конкурентів на насиченому ринку інформаційних послуг.

Хоча створення якісного контенту ще донедавна було досить трудомістким та вартісним процесом, сьогодні спостерігається глобальний тренд до зниження витрат на його виробництво, обробку та розповсюдження. Така тенденція є наслідком стрімкого прогресу в області генеративних моделей штучного інтелекту, які розширюють можливості автоматизації та оптимізації творчих процесів. Інновації в цій сфері сприяють створенню нових інструментів, які дозволяють швидше та ефективніше генерувати якісний контент, знижуючи при цьому затрати часу та ресурсів.

Відповідно до аналітичних даних, представлених на рис. 1.1, можна спостерігати динаміку зростання інтересу до генеративного штучного інтелекту в пошукових запитах Google з лютого 2022 року до лютого 2023 року. Зокрема, різкий підйом активності на початку 2023 року може свідчити

про значні прориви або популяризацію генеративних технологій серед широкого кола користувачів.

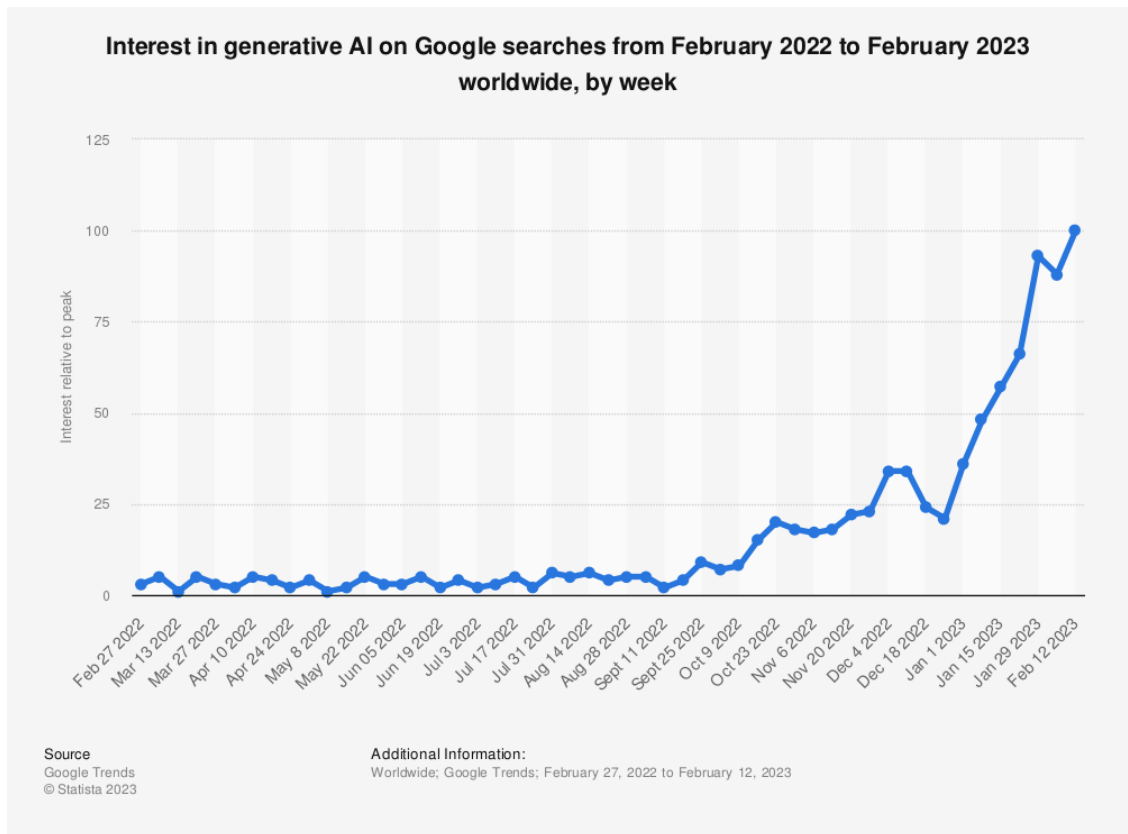


Рис. 1.1. Графік відображення зростання суспільного інтересу до генеративних моделей штучного інтелекту

Вчені та інженери по всьому світу демонструють небачений досі прогрес у розробці алгоритмів, здатних імітувати та доповнювати людську креативність. Ці генеративні системи, що використовують машинне навчання для створення нових образів, текстів, аудіо та відеоматеріалів, змінюють пейзаж цифрового медіапростору, відкриваючи двері до необмежених можливостей у виробництві контенту. Зростання інтересу до цих технологій не лише в академічних колах, а й серед звичайних користувачів і комерційних компаній свідчить про їхнє бачення потенціалу для інновацій та ефективності, зумовлюючи таким чином зростання їхньої популярності та доступності на ринку.

Розглянувши рис. 1.2, можна побачити динаміку впровадження генеративного штучного інтелекту у робочі процеси різних галузей. Лідерство

демонструє сектор маркетингу та реклами, за ним слідують технологічні компанії та консультативні служби. Навіть більш консервативні сфери, як освіта та охорона здоров'я, активно впроваджують новітні ШІ-рішення, свідчаючи про всеохоплюючий імпульс інновацій у бізнес-екосистемі.

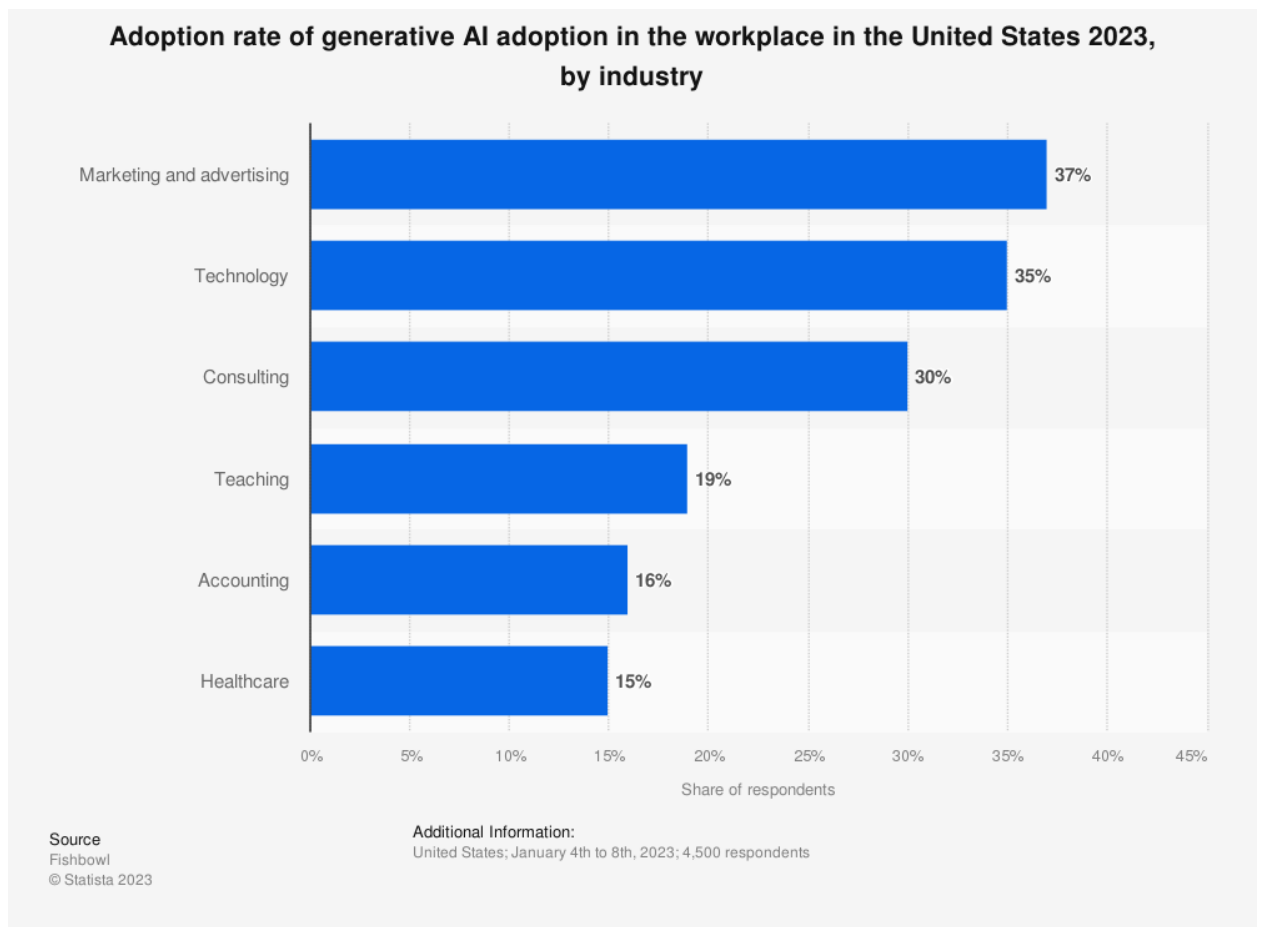


Рис. 1.2. Діаграма стану впровадження генеративних моделей штучного інтелекту в різні галузі США

Поруч з більшістю галузей, на які так чи інакше вплинули виникнення та розвиток генеративних моделей, особливої трансформації зазнає сфера соціальних медіа. Оскільки соціальні мережі за своєю суттю є платформами для генерування контенту від користувачів, сучасні генеративні моделі відкривають нові горизонти для автоматизації та оптимізації цього процесу. Звертаючись до сервісів генеративного штучного інтелекту, сьогодні користувачі здатні автоматично створювати та вдосконалювати найрізноманітніший контент, тим самим значно покращуючи власний

користувачький досвід. Це вказує на те, що розробка системи для створення контенту для соціальних мереж з інтеграцією генеративних моделей штучного інтелекту є актуальним завданням з широкими перспективами.

1.2. Критичний аналіз літературних джерел

У [1] описано модель Transformer, яка стала основою багатьох сучасних моделей генеративного штучного інтелекту, включаючи Generative Pre-trained Transformer (GPT). У даній статті автори вводять концепцію механізму уваги, який дозволяє моделі фокусуватись на важливих частинах вхідних даних при генерації виходу. Це особливо важливо для обробки послідовностей даних, як-от тексту, де контекст та залежності між елементами є критично важливими. Стаття також демонструє ефективність та гнучкість механізму уваги, показуючи, що модель Transformer перевершує традиційні моделі рекурентних та згорткових нейронних мереж в задачах машинного перекладу. Розуміння концепцій, що представлені в [1], допоможе поліпшити здатність розроблюваної системи аналізувати, генерувати та персоналізувати контент для користувачів у реальному часі.

У [2] автори створюють модель, яка генерує мему на основі коментарів користувачів. Вони використовують архітектуру Transformer, зокрема модель GPT-2, з невеликими модифікаціями для адаптації моделі до специфіки їхньої задачі. В статті також обговорюють використання різних метрик для оцінки гумористичного тексту, що генерується нейронною мережею у вигляді підписів до мемів. Результатом дослідження є розроблене програмне забезпечення для генерування нових мемів та подальшого публікування їх на платформі Reddit. Розглянуте дослідження є досить близьким до обраної теми даної магістерської кваліфікаційної роботи. З-поміж його недоліків можна виокремити використання застарілих на даний момент моделей. Зокрема, сьогодні є доступними наступні версії GPT-3,5 та GPT 4, використання яких має значно покращити кінцевий результат генерації контенту.

Автори [3] досягнули декількох важливих результатів, що стосуються використання великих моделей обробки природної мови, таких як GPT-2. Підтверджено ефективність початкового навчання моделі GPT-2. Вона виявилася здатною виконувати завдання обробки природної мови, такі як: відповідь на питання, машинний переклад, генерація висновків без будь-якого спеціального навчання на цих завданнях. Досліджено залежність покращення моделі з її розміром. Дане покращення відбувається за логарифмічною функцією. Визначено, що модель все ще недонавчена на наборі даних WebText. Це свідчить про те, що існує потенціал для подальшого покращення продуктивності за допомогою навчання більш великих моделей на цих даних. В контексті даної магістерської кваліфікаційної роботи обрана стаття важлива для розуміння можливостей подібних моделей та способів їх оцінки.

Дослідження [4] є зосередженим на розширенні розуміння того, як краще взаємодіяти з генеративними моделями для створення бажаних і узгоджених зображень із текстових підказок. Автори використовували такі моделі, як: VQGAN+CLIP, DALL-E та BERT, щоб створити візуальне поєднання конкретних предметів та образних стилів. Дослідження є актуальним для даної роботи, оскільки розкриває процес ефективної взаємодії з моделями перетворення тексту в зображення.

У [5] автори зосереджуються на аналізі існуючих способів оцінки результатів роботи NLG. У статті критикується їхня недосконалість через залежність від системи даних. Пропонується альтернативний підхід оцінки шляхом впровадження метрик на основі граматики. Дана стаття корисна з точки зору розуміння підходу до оцінки якості згенерованого контенту.

У [6] подано опис фреймворку StyleNet. Даний фреймворк призначений для створення привабливих візуальних підписів із різними стилями. У документі розглядаються проблеми створення підписів із певними стилями за рахунок факторизованої моделі LSTM, яка може відокремити фактичні та стильові фактори від речень за допомогою багатозадачного навчання. Запропонована структура StyleNet оцінюється на новому наборі даних

стилізованих підписів Flickr. Результати демонструють, що вона може створювати візуально доречні та привабливі підписи з різними стилями. Даний фреймворк буде розглянутий на предмет можливості імплементації в проект, що розробляється.

У [7] проведено дослідження можливостей GPT-3 синтезувати та пояснювати інформацію. Автори обговорюють роль штучного інтелекту в різних аспектах академічних досліджень. Зокрема, особливу увагу приділено тому, як узгоджується згенерований контент з загальними вимогами до авторства. Дана стаття розширює уявлення про академічну доброчесність використання генеративних моделей для генерації контенту.

У [8] розглядається набір програм, написаних мовою Python, які використовують великі мовні моделі для ідентифікації та аналізу даних із платформ соціальних медіа, що мають відношення до групи інтересів. Метою дослідження є представлення продукту інтеграції GPT-3 OpenAI з соціальною мережею Twitter. В контексті даної роботи важливо дослідити принципи подібної інтеграції.

У [9] розглядається дослідження використання GPT-3 для генерування та трансформації вмісту без ручної допомоги з боку людей. Автори також звертають увагу на можливі зловживання інструментом і його основні можливості та обмеження. Розуміння описаних концепцій дозволить знайти правильний підхід для інтеграції генеративних моделей в розроблюване програмне забезпечення.

У [10] розглянуто появу та еволюцію соціальних ботів – програмних алгоритмів, що автоматично генерують контент і взаємодіють з людьми на соціальних медіа платформах. Автори розглядають різні варіанти соціальних ботів, їхній потенційний вплив на суспільство, а також методи виявлення та аналізу ботів. Вони також описують ряд інструментів для виявлення ботів, включаючи "Bot or Not?", "Network Hashtag Graph", інструменти аналізу настроїв, інструменти часового аналізу, аналіз поведінки користувачів, захист від Sybil атак через соціальні мережі та детектор Sybil Renren. Дана стаття

допоможе зрозуміти особливості сприйняття згенерованого контенту соціальними мережами.

У [11] представлено модель, розроблену для класифікації коротких текстів українською мовою за їхньою науковою тематикою з використанням методики передавального навчання, NLP та BERT. Використовуючи Python та кілька його бібліотек машинного навчання, автори навчили багатомовну модель BERT на українських текстах шкільних предметів та досягли точності класифікації від 85 до 98% після однієї епохи навчання. Однак, BERT виявився відносно повільним і вимогливим до обсягу пам'яті. Рішення на основі BERT виявилось конкурентоспроможним порівняно з іншими підходами до проблеми класифікації текстів українською мовою.

У [12] оцінено зміст, згенерований моделлю обробки мови ChatGPT, за шістьма ключовими журналістськими стандартами: актуальністю, надійністю, балансом думок, відокремленням фактів від думок, точністю та повнотою інформації. Дослідження виявило, що ChatGPT має тенденцію генерувати упереджений зміст, який більше відповідає контексту запиту користувача, ніж дотримується балансу. Відзначено і інші недоліки, включаючи відсутність найновішої інформації, непрозорість джерел даних та схильність до вигадкування фактів. Проте, незважаючи на обмеження, наявність та можливості ChatGPT вказують на далекосяжні перспективи використання AI технологій у цифрових медіа, хоча і вимагають додаткових досліджень та уточнень.

Стаття [13] пропонує всеосяжний огляд історії недавніх досягнень в області генеративних моделей штучного інтелекту, з особливим акцентом на унімодальні та мультимодальні генеративні моделі. Крім того, обговорюються недавні застосування генеративних моделей AI, часто використовувані техніки в AIGC. На завершення автори досліджують відкриті проблеми та майбутні напрямки для AIGC, висвітлюючи потенційні шляхи для інновацій та прогресу. Основна мета цього огляду – надати читачам всеосяжне розуміння недавніх розвитків та майбутніх викликів у генеративному AI.

Автори [14] зосереджуються на вивченні технологій AI-генерованого контенту, таких як ChatGPT, які набули широкого застосування і призводять до парадигми зміни у створенні контенту та представленні знань. Автори проводять детальне дослідження принципів роботи генеративних моделей, їхньої архітектури, ключових характеристик, а також потенційних загроз безпеці, приватності, етичних і суспільних наслідків. Окрім того, в статті розглядаються сучасні методи захисту контенту та невирішені проблеми. Основна мета дослідження – надати всеосяжний огляд принципів роботи, потенційних загроз безпеці, приватності та етиці, можливих контрзаходів у галузі генеративного контенту, а також стимулювати додаткові інноваційні ініціативи в цій області.

Автори [15] надають всеосяжний огляд штучно створеного контенту, розглядаючи його визначення, основні умови, передові можливості та функції. Використовуючи машинне навчання, моделі здатні доповнити або замінити ручне створення контенту на основі ключових слів користувача або вимог. Основною ціллю статті є дослідження поточних можливостей та недоліків, аналіз великомасштабних попередньо навчених моделей та промислового ланцюга. Автори також розглядають різницю між допоміжною та автоматичною генерацією, потенційну інтеграцію моделей з віртуальним всесвітом. В кінці статті акцентуються існуючі проблеми та пропонуються майбутні напрямки для застосування.

У [16] описано особливості використання Інтернет-протоколів для управління електронною поштою. Розглянуто принципи роботи таких протоколів як SMTP, IMAP та POP, а також наведено приклади їх використання для управління електронною поштою. Це важливо для даної роботи в якості створення ще одного каналу розповсюдження інформації через електронні листи.

У [17] описано процес вибору архітектури програмної компоненти платформи для монетизації навчальних курсів. Програмна реалізація веб-

додатку, що розробляється, базується на схожих принципах з деякими вдосконаленням.

Автори [18] обговорюють методи покращення взаємодії з великими мовними моделями, такими як GPT, через експериментальне порівняння стандартного підходу до формулювання запитань (промптів) та підходу "Chain-of-Thought". Останній підкреслює переваги детальнішого та структурованого викладу думок, що сприяє більш ефективній взаємодії з моделями штучного інтелекту. В розвиток теми, [19] розглядає ще складніший механізм взаємодії з мовними моделями, знаний як "Tree of Thoughts". Дана стаття вважається фундаментальною у контексті розробки агентів штучного інтелекту, оскільки вона розширює поняття лінійного викладу думок до більш складної, деревоподібної структури, що дозволяє глибше зануритися у проблематику та забезпечити більш комплексне рішення.

Стаття [20] акцентує увагу на зростаючій важливості перевірки фактів у часи, коли поширення дезінформації перевищує можливості людської перевірки. Вона висвітлює роль Великих Мовних Моделей, таких як GPT-4, у верифікації інформації та написанні академічних статей, судових позовів та новинних статей, підкреслюючи їх значення у відрізненні правди від неправди та важливість перевірки їхніх результатів. Результати показують, що ефективність значно зростає при використанні контекстуальної інформації. Дослідження вимагає подальших досліджень для глибшого розуміння, коли моделі досягають успіху і коли вони зазнають невдачі.

В даному підрозділі проаналізовано провідні публікації, що стосуються тематики генеративних моделей ШІ. Це допомогло зрозуміти поточний стан даної області дослідження та перспективні напрямки для продовження роботи. Висновки, що можна зробити після аналізу, полягають в тому, що генеративні моделі є перспективною сферою з надзвичайно швидким впровадженням інновацій. Вони мають широку сферу застосування та підходять для імплементації в систему створення контенту для соціальних мереж.

1.3. Постановка проблеми та її обґрунтування

Прогрес у галузі штучного інтелекту, зокрема в розвитку генеративних моделей, відкриває нові перспективи для створення контенту у соціальних мережах. Цей прогрес підштовхує до розробки систем, здатних автоматизувати та оптимізувати процеси створення контенту, включаючи тексти, зображення, музику, та навіть відео. Постійно зростаюча потреба в швидкому та якісному контенті робить генеративні моделі штучного інтелекту ключовими технологіями у задоволенні цих потреб.

Водночас, з'являється виклик збереження людської креативності та унікальності в контенті, генерованому машинами. Створення системи, яка гармонійно поєднуватиме можливості штучного інтелекту з людською творчістю, є суттєвим кроком у напрямку інновацій у соціальних мережах. Це дозволить не тільки підвищити ефективність створення контенту, але й забезпечити його релевантність та персоналізацію для кінцевих користувачів.

Важливим аспектом є також забезпечення якості та оригінальності контенту. В епоху цифрових медіа, де кожен користувач може бути одночасно і споживачем, і творцем контенту, виникає потреба у створенні контенту, який не лише високоякісний, але й відображає індивідуальність користувачів.

Наявність системи, яка ефективно інтегрує штучний інтелект для створення контенту, відкриває широкий спектр можливостей, від автоматизації рутинних завдань до розробки новаторських способів взаємодії з аудиторією. Це становить вирішальний фактор для бізнесів, стартапів та інших організацій, які прагнуть використовувати потенціал соціальних медіа для розширення своєї аудиторії та залучення клієнтів.

Основним завданням цієї роботи є розробка системи створення контенту для соціальних мереж, яка інтегрує передові генеративні моделі штучного інтелекту. Це має на меті не тільки вирішити існуючі виклики, але й відкрити нові можливості для персоналізації та автоматизації створення контенту.

1.4. Формулювання мети і задач дослідження

Незважаючи на значний прогрес, досягнутий в дослідженнях у галузі генеративних моделей штучного інтелекту та способів їх інтеграції, обрана тема має великий потенціал для подальшого розвитку досліджень. Мета даної магістерської роботи полягає у розробці оригінальної системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Розробка такої системи покликана відповісти на виклики сучасного інформаційного простору, де швидкість, обсяг та якість контенту відіграють ключову роль.

Для досягнення цієї мети в магістерській роботі будуть вирішені наступні завдання:

- вивчення сучасних тенденцій в області генеративних моделей штучного інтелекту та їх застосування в соціальних мережах;
- розробка концептуальної моделі системи управління контентом, яка включає інтеграцію генеративних моделей;
- вибір конкретних моделей для впровадження в систему;
- розробка функціоналу для взаємодії з користувачами, генерування, оптимізації та розповсюдження контенту;
- тестування та оцінка ефективності розробленої системи.

Об'єкт дослідження – процеси інтеграції сучасних генеративних моделей штучного інтелекту.

Предмет дослідження – метод і технологія інтеграції сучасних генеративних моделей штучного інтелекту з оригінальною системою управління контентом соціальних мереж.

Отже, у підрозділі 1.4 було сформульовано мету і задачі магістерської кваліфікаційної роботи. Метою є розробка оригінальної системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Це дослідження спрямоване на відповідь на

виклики сучасного інформаційного простору, в якому швидкість, обсяг і якість контенту відіграють вирішальну роль.

1.5. Висновки

У першому розділі магістерської роботи розглянуто ключові аспекти та виклики, пов'язані з інтеграцією генеративних моделей штучного інтелекту в процеси створення контенту для соціальних мереж. Аналіз наукових джерел показав, що область прикладного застосування генеративних моделей розвивається шаленими темпами, тому дана тема має значний потенціал для подальших досліджень.

Однією з основних проблем є необхідність збереження балансу між автоматизацією процесів та збереженням людської креативності та унікальності в контенті. Хоча генеративні моделі штучного інтелекту здатні ефективно генерувати різноманітний контент, важливо забезпечити, щоб цей контент був не лише технічно правильним, але й цікавим, привабливим та оригінальним для кінцевих користувачів.

Іншим важливим викликом є забезпечення високої якості та оригінальності контенту в умовах постійно зростаючого обсягу інформації, яку необхідно обробляти. Це вимагає розробки ефективних технік, які дозволяють адаптувати контент до індивідуальних потреб та інтересів користувачів.

Нарешті, розробка оригінальної системи управління контентом для соціальних мереж з інтеграцією генеративних моделей штучного інтелекту має потенціал значно вплинути на наукову спільноту та комерційний сектор, відкриваючи нові можливості для інновацій та розвитку. Така система може сприяти автоматизації рутинних завдань, розробці новаторських способів взаємодії з аудиторією, та, в кінцевому підсумку, підвищенню загальної ефективності процесів створення контенту.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ІНТЕГРАЦІЇ СИСТЕМИ ДЛЯ СТВОРЕННЯ КОНТЕНТУ З ГЕНЕРАТИВНИМИ МОДЕЛЯМИ ШТУЧНОГО ІНТЕЛЕКТУ ТА СОЦІАЛЬНИМИ МЕРЕЖАМИ

2.1 Революція трансформерів у машинному навчанні

Трансформер представляє собою порівняно новий концепт в машинному навчанні, теорію якого вперше було описано у науковій роботі «Attention is All You Need» у 2017 році дослідниками з Google Brain. Ця модель глибинного навчання, що пропонує ефективний підхід до виконання завдань у сфері обробки природньої мови, здатна контекстуально обробляти текст та визначати взаємозв'язки між словами. Вона відрізняється від традиційних рекурентних нейронних мереж, адже може одночасно обробляти всю послідовність вхідних даних, що робить її значно ефективнішою.

Завдяки механізму «self-attention», трансформери з легкістю виявляють взаємозв'язки між словами, навіть якщо вони розташовані далеко одне від одного в тексті, що підвищує точність і глибину розуміння контексту. Наприклад, в аналізі речень, таких як «Який чудовий день, щоб прогулятися! Я обожаю, коли сонце світить», слова «чудовий день» і «сонце світить» розташовані далеко одне від одного, але «self-attention» дозволяє моделі зрозуміти, що ці два елементи мають сильний взаємозв'язок у контексті цілого речення. Модель вагомо враховує цей зв'язок, вибудовуючи більш точне та глибоке розуміння загального сенсу та емоційного підтексту.

Цей підхід у технології машинного навчання дозволив створити такі моделі, як BERT від Google, GPT від OpenAI, Llama від Meta та багато інших з найрізноманітнішими характеристиками, які вплинули на різноманітні сфери, включаючи пошукові системи та автоматизацію обслуговування клієнтів. Їх застосування в розумінні природньої мови та генерації тексту змінило підходи в області штучного інтелекту, підкреслюючи роль трансформерів як ключового елемента у цьому напрямку. Суттєвим фактором можна розглядати випуск деяких передових моделей у формі відкритого

програмного забезпечення, що значно розширює доступність подібних технологій. Огляд деяких популярних моделей наведено в таблиці 2.1.

Таблиця 2.1

Огляд попередньо навчених моделей на основі Трансформер

Назва	Компанія	Параметри	Відкритий код	Дата виходу
GPT 4	Open AI	1.76 трлн.	Ні	Березень 2023
GPT 3.5	Open AI	175 млрд.	Ні	Листопад 2022
PaLM 2	Google	340 млрд.	Ні	Травень 2023
Claude 2	Anthropic	137 млрд.	Ні	Липень 2022
Cohere	Cohere	410 млн. 52 млрд.	Так	Травень 2021
Falcon	ТІІ	40 млрд. 180 млрд.	Так	Вересень 2023
LLaMA 2	Meta AI	7 млрд. 70 млрд.	Так	Липень 2023
Vicuna	LMSYS	7 млрд. 33 млрд.	Так	Березень 2023
MPT-30B	MosaicML	30 млрд.	Так	Червень 2023
Guanaco	UW nlp group	7 млрд. 65 млрд.	Так	Травень 2023

Трансформери допомогли підвищити можливості штучного інтелекту в завданнях роботи з природньою мовою, ефективно працюючи в різних сценаріях: переклад, генерація, класифікація, підсумовування текстів, відповідь на питання, аналіз емоцій, розпізнавання іменованих сутностей тощо. Майбутнє трансформерів є досить перспективним. Повсякчас здійснюються спроби покращення їхньої ефективності шляхом скорочення даних для тренування та подолання викликів з інтерпретацією.

Сьогодні трансформери все активніше застосовуються в мультимодальних системах штучного інтелекту, що означає їх використання у системах, які обробляють та інтегрують кілька типів даних, таких як текст, зображення, звук і відео. Ця технологія дозволяє створювати більш комплексні та інтерактивні системи, що можуть розуміти та аналізувати інформацію з різних джерел. Це відкриває нові можливості для розвитку штучного інтелекту в таких сферах, як автоматичний переклад, розуміння мультимедійного контенту, інтерактивні помічники та ін.

2.2 Огляд моделей глибинного навчання на основі дифузії

Дифузійні моделі глибинного навчання представляють собою передову сферу досліджень, яка демонструє вражаючі результати в різноманітних областях, включаючи усунення шуму та артефактів, колоризацію, підвищення роздільної здатності та генерацію зображень. Ці моделі, що використовують ланцюги Маркова для дифузійних процесів, здійснюють перехід від чистого шуму до цільового зображення, де кожен крок впливає лише на наступний.

Одним із ключових підходів у цій області є імовірнісні моделі дифузії з усуненням шуму. Вони працюють шляхом інтеграції шуму в зображення, після чого за допомогою нейронних мереж відбувається відновлення оригінального зображення з ефективним усуненням шуму.

Ще один підхід – це мережі балів з умовою шуму, де шум використовується як умова для генерації нових даних. Такі моделі використовують два шумові вектори: один для створення зображення шуму і інший для контролю процесу дифузії.

Третім значним напрямком є стохастичні диференціальні рівняння, які є математичними моделями для відтворення систем, які розвиваються з часом з урахуванням випадкових змін. У контексті дифузійних моделей вони використовуються для моделювання процесу дифузії в часовому аспекті.

Кожен із цих підходів робить свій внесок у розвиток та розширення можливостей дифузійних моделей, пропонуючи унікальні переваги та можливості для специфічних застосувань. Їхні різноманітність та гнучкість роблять дифузійні моделі глибинного навчання однією з найбільш обладійливих та інноваційних областей сучасних досліджень у сфері штучного інтелекту.

Дифузійні моделі глибинного навчання знайшли широке застосування у різних галузях, від медичної візуалізації до розваг. Наприклад, у медицині ці моделі використовуються для покращення якості зображень МРТ та КТ, де вони допомагають усувати шум та артефакти, значно підвищуючи чіткість

діагностичних зображень. У сфері цифрового мистецтва та графічного дизайну дифузійні моделі використовуються для автоматичного колоруювання чорно-білих зображень та відео, надаючи їм живість та реалістичність. Крім того, в галузі розваг та відеоігор, ці моделі застосовуються для генерації реалістичних текстур та ландшафтів, а також для створення динамічних сцен і персонажів.

Порівняємо найвідоміші програмні реалізації моделей глибинного навчання на основі дифузії (Таблиця 2.2).

Таблиця 2.2

Програмні реалізації моделей глибинного навчання на основі дифузії

Модель	Stable Diffusion	DALL-E-3	Midjourney
Доступність	Відкритий код на GitHub	Комерційний	Комерційний
Ціноутворення	Безкоштовно, запускається локально	Платний API	Платна підписка
Якість	Висока, реалістична	Висока, деталізована	Висока, деталізована, реалістична
Креативність	Висока	Висока	Висока
Підтримка спільноти	Активна	Обмежена	Обмежена
Ліцензування	Вільне для некомерційного використання	Комерційне	Комерційне

Підсумовуючи, Stable Diffusion – це хороший варіант для користувачів, які хочуть створювати високоякісні зображення без плати за підписку. Однак дана модель не така креативна, як Midjourney, і не така швидка, як DALL-E-3. Модель DALL-E-3 – найкращий варіант для користувачів, які хочуть створювати високоякісні зображення з широким набором стилів. Це також найшвидший варіант, який має зручний API для інтеграції у власні проекти. Midjourney – хороший варіант з найширшими можливостями генерації якісних реалістичних зображень.

Для реалізації завдань генерування контенту в даній роботі буде використано DALL-E-3 API.

2.3 Порівняння доступних API генеративних моделей та оцінка можливостей їх інтеграції у проект

Сьогодні існує велика кількість провайдерів технологій генеративного штучного інтелекту. Проведемо огляд деяких з них в таблиці 2.3.

Таблиця 2.3

Провайдери технологій генеративного штучного інтелекту

API	Призначення	Ціна	Можливості
OpenAI	Творчий текст, генерування зображень, програмний код	Оплата лише за використані ресурси, наявні тестові кредити	Надає перелік різних моделей, що дозволяють з високою якістю виконати завдання генерації контенту.
Amazon Generative AI API	Текст, зображення, аудіо, програмний код	Безкоштовне використання до 10000 запитів на місяць; платні плани, доступні для більшого використання	Легко інтегрується з іншими проектами за допомогою API або з іншими службами Amazon, такими як: AWS Lambda та Amazon S3
Google Generative AI API	Текст, зображення, програмний код	Безкоштовне використання до 100 000 запитів на місяць; платні плани, доступні для більшого використання	Легко інтегрується з іншими проектами за допомогою API або з іншими службами Google, такими як: Google Cloud Platform і Google Workspace
Meta Generative AI API	Текст, зображення	Безкоштовне використання до 1000 запитів на місяць; платні плани, доступні для більшого використання	Може бути інтегрований з іншими проектами за допомогою API, але не так добре інтегрований з іншими службами Meta
IBM Watson Assistant	Побудова віртуальних агентів/чат-ботів	Корпоративний план для свого продукту або безкоштовний план із дуже обмеженими функціями	Є хорошим вибором для генерування тексту для різноманітних цілей, оскільки він навчається на масивному наборі тексту, що дозволяє генерувати текст, який є одночасно творчим і фактичним.

На підставі проведеного порівняльного аналізу прийнято рішення інтегрувати до проекту саме OpenAI API. Обраний продукт найбільш точно відповідає вимогам проекту та дозволить реалізувати необхідний функціонал.

До ключових пріоритетів належить забезпечення ефективної взаємодії з користувачами через інтерпретацію природньої мови, а також створення змісту, що відповідає контексту та інтересам користувачів.

OpenAI API надає можливість використовувати високоякісні моделі для генерації тексту та зображень, що дозволяє впровадити підтримку розуміння мови і забезпечити її автоматичну інтерпретацію в реальному часі. Це включає можливість створювати автоматичні відповіді на запити користувачів, генерувати оригінальний зміст для постів, а також допомагати в аналізі та адаптації контенту.

2.4 Опис архітектури веб-додатку для створення контенту

Кожного разу, вводячи адресу сайту або натискаючи на гіперпосилання, користувач ініціює взаємодію між браузером і веб-сервером здебільшого через протокол HTTP. У сучасних веб-застосунках сервер виконує усю логіку програми, що пов'язана з взаємодією з базою даних, обробкою запитів, генеруванням відповідей на запити тощо, тоді як клієнт, представлений браузером, слугує інтерфейсом для кінцевого користувача. Схему клієнт-серверної архітектури можна побачити на рис. 2.1.

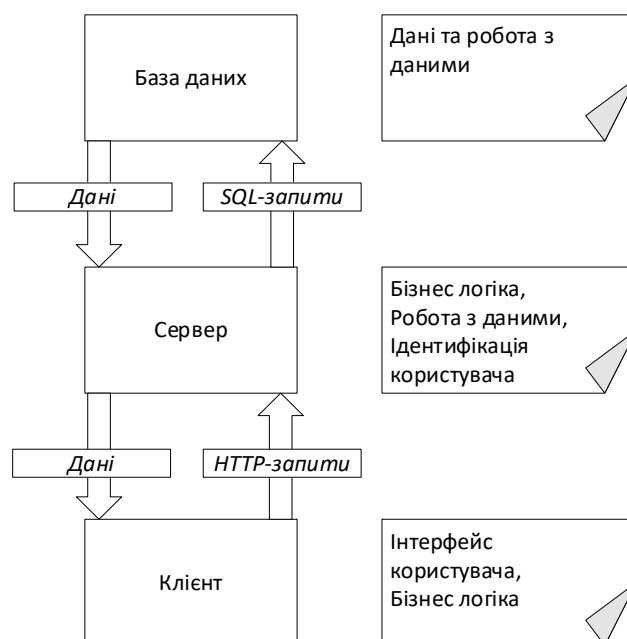


Рис. 2.1. Клієнт-серверна архітектура веб-застосунку

Розглядаючи архітектурні компоненти веб-застосунку, можна визначити, що сервер виступає як центральний елемент, що об'єднує всі інші частини системи. Це не просто місце, де зберігається бізнес-логіка; це складний механізм, який забезпечує оптимальну взаємодію між користувачем, застосунком та базою даних. Однією з ключових функцій сервера є механізм ідентифікації користувача, що не лише визначає особистість користувача, але й забезпечує безпеку даних, обмежуючи доступ до ресурсів. Інтеграція з базою даних реалізована таким чином, що прямий доступ до даних зі сторони клієнта є неможливим. Це є критично важливим для забезпечення конфіденційності та цілісності інформації.

Одиним з найбільш ефективних та універсальних підходів до розробки веб-застосунків є архітектурний шаблон Модель-Представлення-Контролер. Цей шаблон вирішує критичну проблему розробки програмного забезпечення, забезпечуючи розділення бізнес-логіки від її візуального представлення. У цьому шаблоні, Модель відповідає за управління даними та бізнес-правилами системи, Представлення перетворює ці дані у формат, зрозумілий та доступний для користувача, а Контролер діє як посередник між Моделлю та Представленням, управляючи потоком інформації та взаємодією між ними. Такий підхід дозволяє розробникам зосередитись на окремих аспектах застосунку, забезпечуючи більш високу модульність, легкість у тестуванні та підтримці, а також сприяє більш ефективній командній роботі.

Microsoft зі своїм фреймворком ASP.NET MVC розширює можливості цього шаблону, пропонуючи інструменти для автоматичної генерації коду та інші функціональні засоби, що спрощують процес розробки та дозволяють розробникам зосередитися на більш важливих аспектах застосунку, таких як бізнес-логіка.

Рис. 2.2 ілюструє взаємодію між моделлю, представленням і контролером. Представлення, що бачить користувач платформи є згенерованим відображенням певної моделі. Обробка запитів, рендер представлення та керування відбувається через контролер.

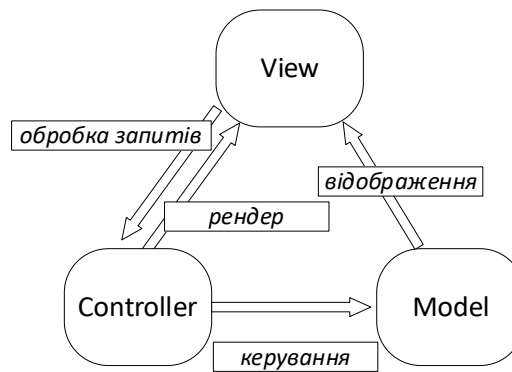


Рис. 2.2. Архітектура MVC веб-застосунку

Після вводу користувачем URL-адреси в пошукову стрічку браузера, вона передається на веб-сервер, де направляється на контролер. Контролер взаємодіє з пов'язаними представленнями та моделями для цього запиту, створює відповідь і надсилає її назад до браузера.

Враховуючи ефективність MVC у розділенні бізнес-логіки та її візуальної презентації, багаторівнева архітектура програмного забезпечення стає логічним наступним кроком (рис. 2.3). Ця архітектура розширює описані принципи, розподіляючи функціональність на декілька рівнів: Web-рівень для інтерфейсу користувача, сервісний рівень для бізнес-логіки та обробки даних, шар репозиторію для доступу та управління даними, та найнижчий, доменний модель, для представлення бізнес-сутностей. Такий мультирівневий підхід забезпечує додаткову гнучкість і масштабованість та спрощує підтримку програмних систем, ефективно інтегруючи та розширюючи базові принципи, закладені в MVC.

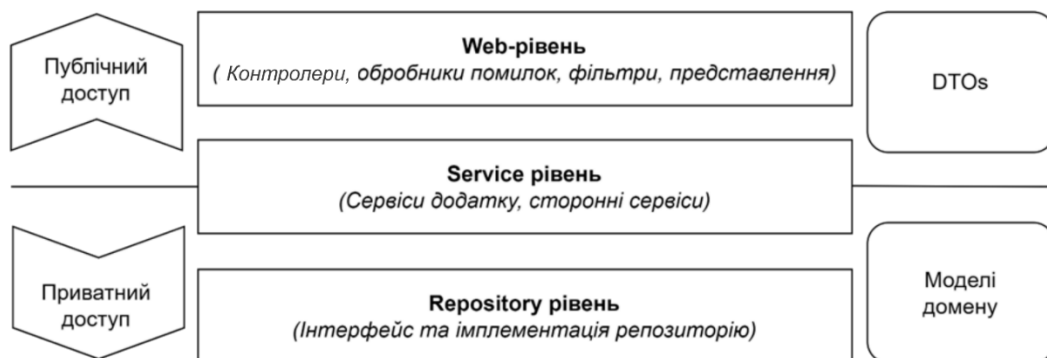


Рис. 2.3. Багаторівнева архітектура застосунку

Для розробки системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту обрано ASP.NET MVC багаторівневу архітектуру програмного компоненту. Вона найкраще підходить для реалізації поставлених завдань, дозволяючи створити необхідний для платформи функціонал.

2.5 Опис структури веб-застосунку для створення контенту

У сучасному цифровому середовищі, структура веб-сайтів відіграє вирішальну роль у забезпеченні інтуїтивності та зручності для користувачів. Незважаючи на зовнішню різноманітність, більшість сайтів поділяють спільні структурні особливості, що засновані на перевірених шаблонах. Це включає лінійні структури, які керують користувачами за визначеним маршрутом, ієрархічні структури з деревоподібним розгалуженням, а також більш хаотичні веб-структури. Крім того, існують гнучкі, адаптивні структури, що дозволяють розробникам втілювати унікальні бізнес-процеси.

Для розробки система управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту обрано довільну структуру веб-сайту. Така структура дозволить найкраще реалізувати поставлені перед розробкою завдання, сприяючи ефективності керування увагою клієнта під час перебування на сайті.

Схема розробленої онлайн-платформи зображена на рис. 2.4.

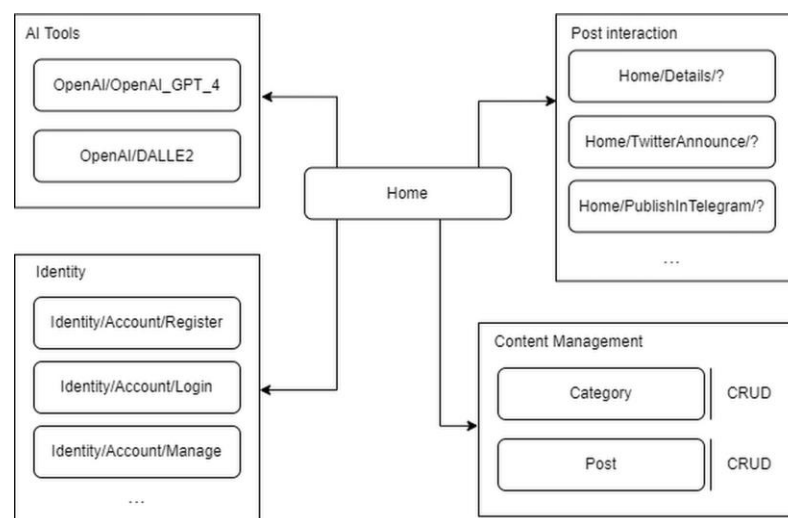


Рис. 2.4. Структурна схема система управління контентом

Онлайн платформа включає такі сторінки:

- Home – головна сторінка сайту, що містить меню сайту, панель вибору категорії публікацій та сам список публікацій;
- Home/Details – сторінка відображення детальної інформації про окрему публікацію;
- Home/TwitterAnnounce – сторінка для підготовки анонсу про окрему публікацію для платформи X (Містить інтегровані API від OpenAI);
- Home/PublishInTelegram – сторінка для підготовки публікації для платформи Telegram (Містить інтегровані API від OpenAI);
- OpenAI/OpenAI_GPT_4 – сторінка для взаємодії з GPT4 OpenAI API;
- OpenAI/ DALL-E-3 – сторінка для взаємодії з DALL-E-3 OpenAI API;
- Category – набір сторінок для створення та керування категоріями публікацій;
- Post – набір сторінок для створення та керування публікацій (Містить інтегровані API від OpenAI);
- Identity – набір сторінок для керування авторизацією користувачів.

Керування основними сторінками веб-платформи відбувається з допомогою навігаційної панелі сайту. Розглядаючи існуючі види навігаційних панелей, можна виділити декілька їх видів. Горизонтальна навігація, розташована у верхній частині, забезпечує швидкий доступ до ключових розділів з лаконічними назвами. Вертикальна панель, зазвичай розміщена зліва, підходить для сайтів з обширними назвами розділів. Прихована навігація, активована значком "гамбургера", оптимізована для мобільних пристроїв. При проектуванні структури важливо враховувати цілі сайту та очікувані дії користувача для забезпечення оптимального досвіду.

Для переходу між сторінками сайту розроблено звичайну горизонтальну навігаційну панель (рис. 2.5).



Рис. 2.5. Навігаційна панель сайту

На навігаційній панелі розміщено посилання на наступні розділи сайту: головна сторінка або сторінка публікацій платформи, розділ менеджменту контенту, який доступний тільки адміністратору сайту, розділ редагування особистої сторінки користувача, службові кнопки: вхід, вихід і реєстрація.

Отже, в підрозділі розглянуто існуючі структури веб-сайту та обрано довірливу структуру. Наведена структурна схема сторінок онлайн платформи. Розроблена навігаційна панель сайту.

2.6 Огляд OpenAI API та способу його інтеграції в проект

OpenAI API представляє собою передовий інтерфейс програмування застосунків, що забезпечує доступ до низки потужних моделей штучного інтелекту, розроблених компанією OpenAI. Ця технологія є ключовим інструментом для розробників, що бажають інтегрувати можливості штучного інтелекту у свої програмні рішення. OpenAI API відкриває доступ до таких моделей як GPT-3.5, GPT-4 для обробки природної мови, DALL-E для генерації зображень та Codex для автоматизації написання коду.

Основною перевагою OpenAI API є його гнучкість і універсальність. Розробники можуть використовувати цей API для створення рішень, що варіюються від простого текстового аналізу до складних інтерактивних ботів, від обробки мовних запитів до створення візуального контенту. Цей інструмент відкриває можливості для різноманітних застосувань, включаючи персоналізовані рекомендаційні системи, автоматизацію відповідей на запитання клієнтів, генерацію контенту для медіа та багато іншого.

OpenAI API також вирізняється простотою інтеграції. Розробники можуть використовувати стандартні HTTP-запити та працювати з API через звичні мови програмування. Це дозволяє легко впроваджувати функціонал штучного інтелекту в існуючі системи без необхідності глибоких знань у галузі машинного навчання.

OpenAI API є цілісною екосистемою різноманітних продуктів, що пов'язані з генеративними моделями (рис. 2.6).

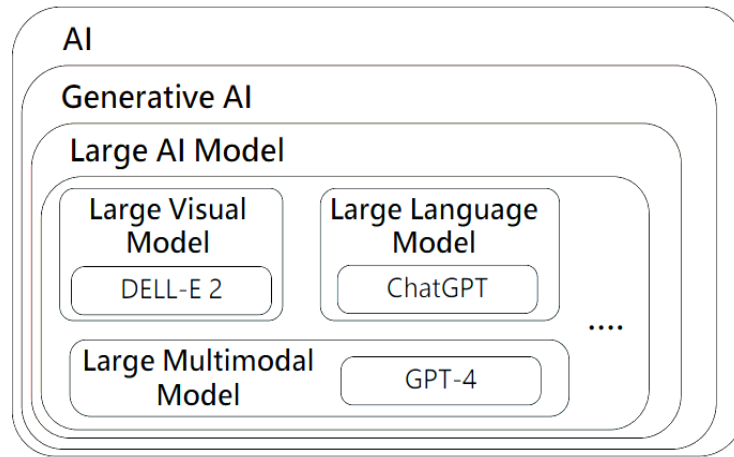


Рис. 2.6. Екосистема програмних рішень OpenAI

Процес інтеграції OpenAI API зазвичай включає такі кроки:

- реєстрація та отримання API-ключа: для використання API OpenAI, потрібно зареєструватися на офіційному веб-сайті і отримати унікальний API-ключ для доступу до сервісів;
- встановлення бібліотеки OpenAI: залежно від мови програмування, використовуйте відповідний менеджер пакетів для встановлення клієнтської бібліотеки OpenAI;
- написання коду для використання API: слідуючи документації OpenAI, реалізуйте код, який здійснює звернення до ендпоінтів API, створюючи об'єкт API з використанням отриманого ключа;
- тестування та налагодження: переконайтеся, що ваш код коректно взаємодіє з API, проводячи відповідні тести.

Важливо також звернути увагу на безпеку та ефективність використання API. Забезпечте, що ваша система адекватно обробляє чутливі дані, і вживайте заходів для захисту API-ключа від несанкціонованого доступу. Крім того, оптимізуйте запити до API, щоб уникнути надмірного навантаження або перевищення обмежень квот. Нарешті, постійно оновлюйте ваше розуміння можливостей і обмежень API, адже OpenAI регулярно оновлює свої функції та політику використання.

Процес інтеграції OpenAI API зображено на рис. 2.7.

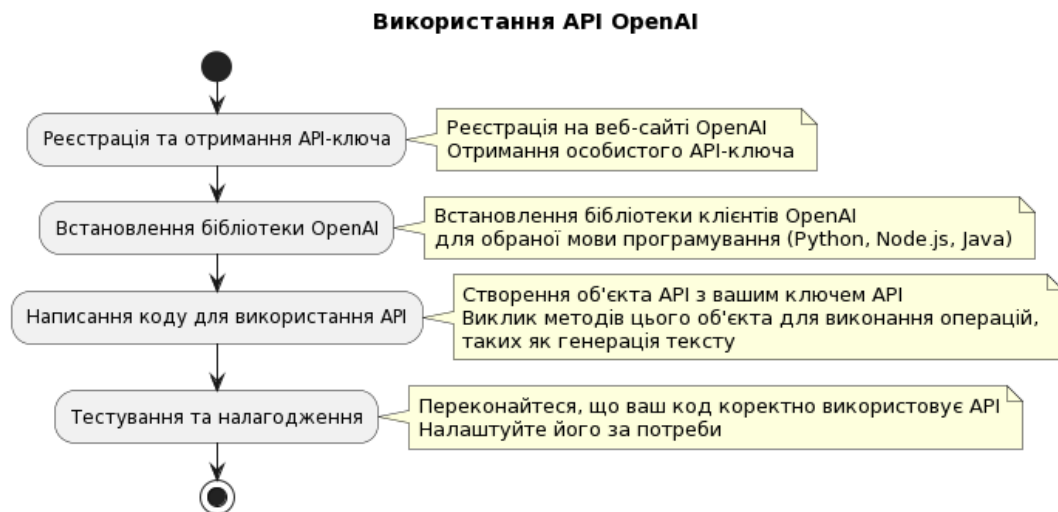


Рис. 2.7. Блок-схема процесу інтеграції OpenAI API

Отже, було розглянуто основні аспекти та процес інтеграції OpenAI API, включаючи його гнучкість та універсальність для різноманітних застосувань. Також були описані ключові кроки інтеграції, від реєстрації для отримання API-ключа до написання коду та його тестування, підкреслюючи важливість безпеки та ефективності при роботі з цим інструментом.

2.7 Опис інтеграції в проект API соціальних мереж

API соціальних мереж – це потужні інструменти, які дозволяють розробникам інтегрувати функціонал соціальних платформ у свої програмні рішення. Ці API надають доступ до широкого спектру можливостей, включаючи взаємодію з профілями користувачів, публікацію та управління контентом, інтеграцію соціальних функцій та аналіз даних соціальних мереж. Основні переваги використання API соціальних мереж включають збільшення залученості користувачів, покращення персоналізації контенту, автоматизацію соціального маркетингу та збір цінних даних по поведінку користувачів. Ці інструменти можуть бути використані в різноманітних застосуваннях, від мобільних застосунків до веб-сайтів та комерційних систем.

Інтеграція API соціальних мереж – це процес, який починається з ретельного вивчення документації API для розуміння його можливостей і обмежень. Розробники повинні зареєструвати свій застосунок на платформі соціальної мережі, щоб отримати унікальний ключ або ідентифікатор для авторизації запитів. Ключовим етапом є встановлення надійної системи авторизації та аутентифікації, часто з використанням OAuth або інших механізмів безпеки. Після цього розробники створюють запити до API для отримання, відправлення чи модифікації даних, які можуть включати в себе доступ до інформації профілю, публікації статусів чи завантаження фотографій. Важливою частиною процесу є обробка відповідей від API, управління помилками та пагінацією даних. Розробники повинні також провести ретельне тестування інтеграції, щоб гарантувати її надійність та ефективність. Вони також мають забезпечити, щоб інтеграція відповідала всім політикам конфіденційності та безпеки, встановленим соціальними мережами, а також міжнародним стандартам захисту даних. Завершальним кроком є розгортання інтеграції в продуктивне середовище та забезпечення її постійного моніторингу для виявлення та вирішення можливих проблем.

У контексті створення системи для генерації контенту для соціальних мереж з використанням сучасних генеративних моделей штучного інтелекту, інтеграція з API таких платформ, як Twitter та Telegram, є ключовою. Розглянемо детальніше процес інтеграції з Twitter, що є продовженням попереднього опису загальної методології.

Процес взаємодії з Twitter API починається з реєстрації застосунку на платформі Twitter для отримання авторизаційних ключів. Використовуючи OAuth для авторизації, система отримує доступ до публікації постів через API. Це включає формування запитів для розміщення твітів, які можуть включати текст або зображення, згенеровані за допомогою моделей штучного інтелекту. Важливим етапом є обробка відповідей від API та управління помилками, що забезпечує правильне публікування контенту.

Інтеграція з Telegram API для системи генерації контенту для соціальних мереж відрізняється від інтеграції з Twitter головним чином у способі взаємодії з платформою. Починаючи з реєстрації застосунку на Telegram для отримання авторизаційних токенів, процес включає розробку механізму для надсилання запитів до API з метою публікації контенту. На відміну від Twitter, де взаємодія зосереджена на публікації у публічних твітах, Telegram надає можливість публікувати контент у приватних або групових чатах, а також у каналах. Це забезпечує більшу гнучкість у розповсюдженні контенту, дозволяючи досягти різних сегментів аудиторії. Обробка відповідей від Telegram API та управління помилками є схожими на Twitter API.

Залучення потенціалу соціальних медіа через ефективне використання їхніх API відкриває широкі можливості для автоматизованого розповсюдження контенту, інтерактивності з аудиторією та оптимізації маркетингових стратегій. Інтеграція з різними платформами, такими як: Twitter та Telegram, демонструє гнучкість та адаптивність цього підходу, підкреслюючи його значення у створенні ефективних цифрових рішень.

2.8 Застосування Інтернет-протоколів для управління електронною поштою

Розробка систем для створення контенту для соціальних мереж, що базуються на генеративних моделях штучного інтелекту, вимагає не лише інтеграції з соціальними мережами, але й використання інших каналів комунікації. Один з таких каналів – це електронна пошта, яка управляється через протоколи, такі як SMTP, IMAP та POP. Ці протоколи надають можливість ефективно інтегрувати різноманітні системи та платформи, дозволяючи організувати комплексний підхід до доставки контенту.

SMTP використовується для надсилання електронних повідомлень. Він передбачає використання команд MAIL, RCPT та DATA для встановлення адреси відправника, одержувача та початку тіла повідомлення, відповідно. Це

дозволяє системі генерації контенту автоматизувати розсилку повідомлень, наприклад, для інформування користувачів про новий контент або оновлення.

Що стосується протоколів IMAP та POP, вони використовуються для отримання повідомлень. Ці протоколи дозволяють системі збирати вхідні повідомлення, що може бути використано для аналізу зворотного зв'язку від користувачів або збору даних для подальшої персоналізації контенту.

Схема використання описаних протоколів зображена на рис. 2.8.

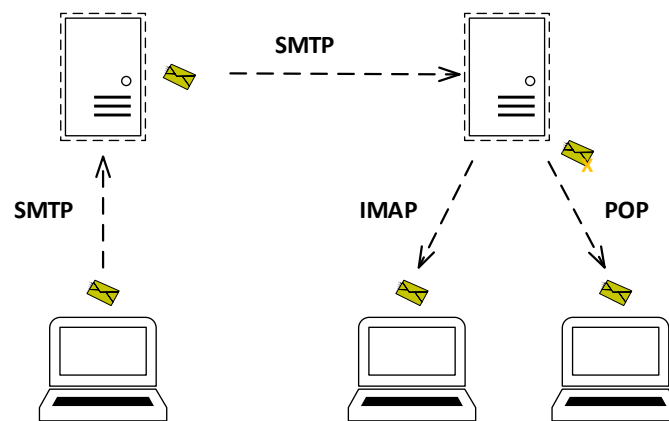


Рис. 2.8. Схема використання протоколів передачі повідомлень

Використання таких бібліотек як AE.Net.Mail спрощує роботу з цими протоколами. За її допомогою можна створити IMAP клієнт для взаємодії з поштовим сервісом, що дозволяє автоматично отримувати та обробляти вхідні повідомлення. Це включає ініціалізацію клієнта з параметрами, такими як: хост, ім'я користувача, пароль, порт, метод автентифікації та використання SSL, та використання методів для витягування та обробки повідомлень.

У підсумку, інтеграція електронної пошти як каналу розповсюдження контенту в системі розширює можливості комунікації з аудиторією. Це не лише забезпечує більшу гнучкість у доставці контенту, але й відкриває шляхи для збору та аналізу даних, що можуть бути використані для оптимізації стратегій маркетингу та підвищення ефективності комунікаційних кампаній. Описані технології буде застосовано при налаштуванні розсилки повідомлень на платформі.

2.9 Розробка методу системи для створення контенту

У сучасних умовах швидкого прогресу технологій генеративних моделей штучного інтелекту постає завдання створення комплексної системи для генерації адаптованого контенту для соціальних мереж. Як відповідь на дану необхідність пропонується система, яка використовує можливості штучного інтелекту для автоматизації цього процесу. Діаграма послідовності візуалізує взаємодії між користувачами та системою, розкриває цілісний процес управління контентом – від моменту створення нового посту користувачем до його публікації в соціальній мережі (Додаток А). Вона також враховує можливість редагування та персоналізації контенту з використанням інтелектуальних алгоритмів штучного інтелекту, що забезпечує не лише автоматизацію процесу, але й підвищення його якості для кінцевого споживача.

Процеси, відображені на діаграмі, включають в себе керування профілем користувача, створення та генерування контенту, зберігання чернеток постів і публікацію постів в соціальних мережах. Кожен з цих процесів включає в себе ряд взаємодій між різними компонентами системи. Через візуалізацію цих взаємодій діаграма послідовності допомагає краще зрозуміти, як система функціонує.

У розділі "Управління профілем користувача", користувач взаємодіє з системою управління контентом для зміни налаштувань свого профілю. CMS зберігає ці зміни в базі даних і повідомляє користувачу про успішне оновлення профілю.

У розділі "Редактор постів та Генерування контенту" користувач створює новий пост в редакторі постів. Редактор постів взаємодіє з моделлю штучного інтелекту для генерування контенту і повертає цей контент користувачу.

У розділі "Зберігання чернеток постів" користувач просить систему зберегти пост як чернетку. Система зберігає чернетку в базу даних і повідомляє користувачу про успішне збереження.

У розділі "Публікація постів" користувач просить систему опублікувати пост. Система використовує адаптер контенту для адаптації контенту до вимог соціальної мережі, відправляє адаптований контент у соціальну мережу і повідомляє користувача про успішну публікацію.

Розробка методу системи для створення контенту є важливим кроком у використанні штучного інтелекту для ефективної роботи з соціальними мережами. Діаграма послідовності розкриває ключові етапи цього процесу, від управління профілями до публікації постів, і демонструє здатність системи до генерації, редагування та адаптації контенту. Представлення описаного методу у вигляді поданих тез на науковій конференції засвідчило її значення та ефективність [21].

2.10 Специфікація вимог до програмного забезпечення

Призначення та мета програмного забезпечення

Це програмне забезпечення призначене для автоматизації процесу створення, редагування та публікації контенту в соціальних мережах. Це досягається шляхом інтеграції генеративних моделей штучного інтелекту, які можуть створювати високоякісний контент на основі вхідних даних користувача.

Головною метою цього програмного забезпечення є полегшення процесу створення та дистрибуції контенту в соціальних мережах, зменшуючи час і ресурси, які потрібні для генерації оригінального та цікавого контенту.

Функціональні вимоги

– Менеджмент профілів користувачів: Система може надавати можливість користувачам керувати своїми профілями, змінювати налаштування, управляти повідомленнями та налаштувати параметри конфіденційності;

- Редактор постів: Система має надавати редактор для створення типових постів для соціальних мереж. Пости повинні складатися з компонентів: заголовок, фотографія, текст, хештеги;
- Генерування контенту: Система має мати можливість генерувати контент для різних компонентів постів (наприклад, заголовок, текст, хештеги), використовуючи генеративні моделі штучного інтелекту, такі як: ChatGPT-3.5 та DALL-E-3. Система повинна запитувати у користувача вхідні дані (промпти) для генерування контенту.
- Зберігання чернеток постів: Система має надавати можливість зберігати незавершені пости як чернетки.
- Публікація постів: Система має надавати можливість публікувати готові пости в різних соціальних мережах, включаючи Twitter та Telegram.
- Адаптація контенту: Система має мати можливість автоматично адаптувати контент до вимог та стандартів різних соціальних мереж.

Нефункціональні вимоги

- Надійність: Система має бути надійною, забезпечувати стабільну роботу і мінімізувати випадки відмов. Система повинна мати відмінні показники наявності та мінімальний час простою;
- Продуктивність: Система має мати високу продуктивність, здатну впоратися з великим навантаженням;
- Безпека: Система має мати високі стандарти безпеки для захисту даних користувачів;
- Використання ресурсів: Система має ефективно використовувати ресурси, включаючи обчислювальні ресурси для генеративних моделей AI, мережеві ресурси для з'єднання з соціальними мережами та зберігання для збереження постів користувачів;
- Інтеграція: Система має бути інтегрованою з різними соціальними медіа платформами та AI моделями;

– Користувацький досвід: Система має бути легкою у використанні, з інтуїтивно зрозумілим інтерфейсом та коротким часом відгуку на дії користувача.

Вимоги до інтерфейсу користувача

- Інтуїтивність: Інтерфейс має бути зрозумілим та легким для користувачів;
- Простота: Дизайн інтерфейсу має бути простим і чистим, з мінімумом непотрібних елементів, що відволікають увагу;
- Пристосування: Інтерфейс повинен бути гнучким і адаптивним, здатним пристосовуватися до різних типів пристроїв та розмірів екрану;
- Естетика: Інтерфейс має бути привабливим і приємним на вигляд;
- Консистентність: Всі елементи інтерфейсу має бути узгоджені за дизайном та поведінкою.

Вимоги до інтеграції

- Інтеграція з соціальними медіа: Система повинна підтримувати публікацію контенту в різних соціальних мережах, зокрема Twitter та Telegram. Це включає в себе не тільки публікацію, але й адаптацію контенту до кожної платформи;
- Інтеграція з генеративними моделями AI: Система повинна підтримувати інтеграцію з моделями AI, такими як: ChatGPT та DALL-E-3, для генерації контенту. Це включає можливість виклику цих моделей та обробки отриманих від них результатів;
- Інтеграція з системами аутентифікації: Система повинна підтримувати аутентифікацію користувачів через їхні облікові записи у соціальних мережах, таких як Twitter та Telegram;
- Інтеграція з системами зберігання даних: Система повинна підтримувати інтеграцію з базами даних для зберігання чернеток, історії постів та інформації про користувачів.

Вимоги до даних

- Управління даними: Система повинна мати інструменти для управління даними, включаючи можливості видалення, модифікації, бекапу та відновлення даних;
- Нормалізація та стандартизація даних: Для забезпечення консистентності дані повинні бути нормалізовані та стандартизовані відповідно до визначених форматів та структур;
- Безпека даних: Дані повинні зберігатися та передаватися безпечно. Це може включати в себе шифрування даних та застосування політик доступу до даних.

Отже, вказана специфікація вимог до програмного забезпечення надає чітку та детальну картину того, що необхідно для розробки ефективного інструменту для автоматизації створення, редагування та публікації контенту в соціальних мережах. Призначення програми полягає у полегшенні процесу створення та дистрибуції контенту, що значно знижує час та ресурси, потрібні для генерації оригінального та привабливого контенту. У цілому, ця специфікація визначає основу для розробки надійної та ефективної системи створення контенту для соціальних медіа.

2.11 Висновки

У другому розділі проведено детальний огляд предметної області процесу інтеграції генеративних моделей штучного інтелекту в систему управління контентом. Досліджено генеративні моделі побудовані на основі «Трансформер», що використовуються для генерації тексту, та генеративні моделі побудовані на основі дифузії, що використовуються для генерування зображень. В результаті було вирішено для інтеграції в проект обрати OpenAI API, оскільки даний програмний продукт досить просто інтегрується в сторонні застосунки, має помірну вартість та надає гнучкий функціонал.

Ключовим етапом даного розділу стала розробка архітектури та структури веб-додатку для створення контенту. Визначено, що сервер відіграє центральну роль у системі, забезпечуючи взаємодію між користувачем,

застосунком та базою даних, а також безпеку та конфіденційність даних. Згадано про використання архітектурного шаблону Модель-Представлення-Контролер, який забезпечує розділення бізнес-логіки від її візуального представлення, а також про мультирівневу архітектуру, що забезпечує додаткову гнучкість і масштабованість.

Важливою складовою даного розділу є розробка методу системи для створення контенту, яка демонструє комплексний підхід до управління різними аспектами створення контенту, включаючи генерацію, редагування та публікацію. Даний метод буде програмно реалізований в наступних розділах.

Специфікація вимог до програмного забезпечення, включена у цей розділ, визначає критерії та стандарти, які будуть використовуватися при розробці цієї системи.

Загалом, представлені у цьому розділі теоретичні рішення та розробки стануть фундаментом для подальшого наукового дослідження та практичної реалізації системи для створення контенту.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ СТВОРЕННЯ КОНТЕНТУ

3.1 Проектування програмного рішення

Відповідно до розробленого в підрозділі 2.9 методу системи для створення контенту специфікації, що описана в підрозділі 2.10, спроектуємо програмне рішення.

Діаграма варіантів використання демонструє різні способи, якими актори можуть взаємодіяти з розроблюваною системою (Додаток А).

На діаграмі видно, що система для створення контенту для соціальних мереж поділена на три основні підсистеми: "Управління профілем", "Створення постів" та "Публікація постів". У підсистемі "Управління профілем" користувач може керувати своїм профілем. У підсистемі "Створення постів" користувач може відкрити редактор постів та створити пост вручну або надати промпт AI моделям. AI моделі на основі промпту генерують текст, фото та хештеги для поста. Користувач також може зберегти пост як чернетку. У підсистемі "Публікація постів" користувач обирає пост для публікації та надає його AI моделям. AI моделі адаптують пост до стандартів різних соціальних мереж. Після адаптації соціальні мережі публікують оновлений пост.

Далі побудуємо діаграму діяльності (Додаток А). Вона представляє робочий процес системи для створення контенту. Користувач може керувати своїм профілем, використовувати редактор постів для автоматичного генерування контенту або ручного створення, зберігати пости як чернетки, а також публікувати адаптований до вимог конкретної соціальної мережі контент.

У рамках проектування програмного рішення розроблена система для створення контенту, що інтегрує AI для генерації, адаптації та публікації контенту в соціальних мережах. Діаграми варіантів використання та діяльності відображають ключові процеси системи, забезпечуючи ефективне

управління контентом від створення до публікації. PlantUML код UML діаграм наведений у додатку Б.

3.2 Опис технологій, використаних у проекті

Розроблений програмний код системи для створення контенту для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту є поєднанням багатьох технологій. Список та відсоток використання мов програмування, застосованих у проекті, зображений на рис. 3.1.

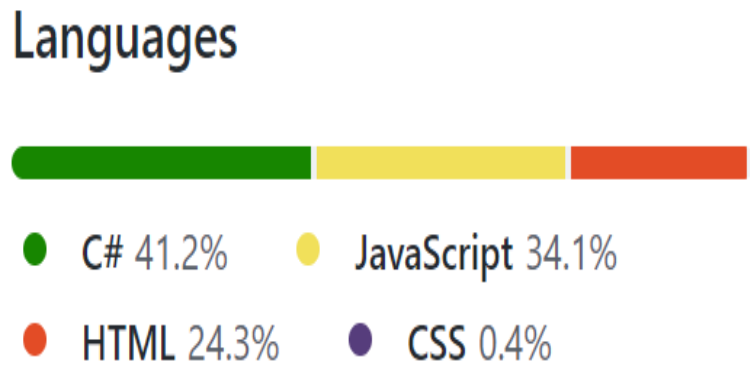


Рис. 3.1. Статистика використання мов програмування

Веб-застосунок інтегрує ряд сучасних технологій та підходів у програмуванні задля створення надійного, функціонального та інтуїтивно зрозумілого користувацького досвіду. Основою для реалізації застосунку вибрано .NET 7, останню версію відомого та високоефективного фреймворку від Microsoft. Він забезпечує оптимальну продуктивність, безпеку та підтримку крос-платформного розроблення, що є важливими факторами для сучасних веб-застосунків.

За архітектурою застосунку вибрано модель MVC, яка дозволяє ефективно відокремити бізнес-логіку від користувацького інтерфейсу, забезпечуючи чистоту коду та спрощуючи процес розробки та підтримки. Для взаємодії з базами даних використовується Entity Framework Core, потужний ORM інструмент, який спрощує роботу з даними, автоматизуючи численні операції. Це забезпечує гнучкість та ефективність при роботі з базами даних.

Значну увагу приділено безпеці та управлінню ідентифікацією користувачів, використовуючи ASP.NET Core Identity для автентифікації та авторизації. Це гарантує надійний захист даних користувачів та системи в цілому.

Інтеграція зі сторонніми сервісами, такими як: Telegram та Twitter відкриває можливості для розширення функціональності та взаємодії з іншими платформами. У проекті також реалізована інтеграція з OpenAI API, що є важливим аспектом для включення функцій штучного інтелекту та машинного навчання.

У проекті використовуються Razor Views, потужний інструмент для генерування HTML на стороні сервера. Razor забезпечує зручний синтаксис для вбудовування C# коду безпосередньо в HTML, що дозволяє створювати динамічні веб-сторінки. Фронтенд розроблено з використанням Bootstrap, що забезпечує адаптивність та сучасний дизайн інтерфейсу.

У цілому, вибір технологій та їхня конфігурація в проекті забезпечують міцну основу для розробки надійного, ефективного та зручного у використанні веб-застосунку, відповідного сучасним вимогам та очікуванням користувачів.

3.3 Програмна реалізація основних компонентів платформи

Онлайн-платформа реалізована на основі .NET 7 MVC. Вибір цього фреймворку не випадковий, оскільки він надає широкі можливості для створення сучасних, високопродуктивних веб-застосунків та дозволяє розробникам платформи реалізувати її основні компоненти у відповідності з сучасними тенденціями веб-розробки. Структура програмного коду застосунку повністю відповідає накладеним фреймворком обмеженням та його логіці.

Головним файлом, що є вхідною точкою програми, є Program.cs. Цей файл представляє собою конфігураційний файл для налаштування та запуску веб-застосунку на платформі ASP.NET Core. Розглянемо його елементи:

– Маршрутизація та точки завершення:

```
1. app.MapControllerRoute(
2.     name: "default",
3.     pattern: "{controller=Home}/{action=Index}/{id?}");
4. app.MapRazorPages();
5.
```

Цей код додає маршрут з ім'ям default. Маршрут має шаблон {controller=Home}/{action=Index}/{id?};

– Налаштування доступу до бази даних, реєстрація репозиторіїв та реєстрація класу ConfigurationBuilder, що використовується для створення об'єкта конфігурації, який містить налаштування застосунку:

```
1. var configuration = new ConfigurationBuilder()
2.     .AddJsonFile("appsettings.json").Build();
3. builder.Services.AddDbContext<ApplicationDbContext>
4. (options => {
5.     options.UseSqlServer(configuration.GetConnectionString("DefaultConnection"
6. ));
7. });
8. builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
9. builder.Services.AddScoped<IPostRepository, PostRepository>();
10.
```

– Налаштування ідентифікації та авторизації, а також конфігурація сеансу для веб-застосунку:

```
1. builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>
2.     options.SignIn.RequireConfirmedAccount = false)
3.     .AddDefaultTokenProviders().AddDefaultUI().AddEntityFrameworkStores<Applic
4.     ationDbContext>();
5. builder.Services.AddSession(options => {
6.     options.IdleTimeout = TimeSpan.FromMinutes(10);
7.     options.Cookie.HttpOnly = true;
8.     options.Cookie.IsEssential = true;
9. });
```

У програмній архітектурі репозиторій є ключовим елементом для розділення бізнес-логіки та доступу до даних. Це дозволяє працювати з даними без знання про їхнє фізичне зберігання, полегшує тестування за допомогою замінених репозиторіїв, сприяє повторному використанню коду через універсальний клас Repository<T>, і централізує логіку доступу до даних, що спрощує внесення змін у способи зберігання та отримання даних. В

інтерфейсі класу Repository визначені основні методи, які можуть бути використані для отримання, додавання, видалення та зберігання даних.

```

1. public interface IRepository<T> where T : class
2. {
3.     Task<T> FindAsync(int id);
4.     Task<IEnumerable<T>> GetAllAsync(
5.         Expression<Func<T, bool>> filter = null,
6.         Func<IQueryable<T>, IOOrderedQueryable<T>> orderBy = null,
7.         string includeProperties = null,
8.         bool isTracking = true
9.     );
10.    Task<T> FirstOrDefaultAsync(
11.        Expression<Func<T, bool>> filter = null,
12.        string includeProperties = null,
13.        bool isTracking = true
14.    );
15.    void Add(T entity);
16.    void Remove(T entity);
17.    void RemoveRange(IEnumerable<T> entity);
18.    Task SaveAsync();
19.}

```

Розглянемо один з реалізованих методів класу Repository GetAllAsync(). Цей метод повертає колекцію сутностей типу T. Він дозволяє фільтрувати результати, сортувати їх, включати зв'язані сутності та контролювати перебіг запитів.

```

1. public async Task<IEnumerable<T>> GetAllAsync(Expression<Func<T, bool>> filter
2.     = null, Func<IQueryable<T>, IOOrderedQueryable<T>> orderBy = null, string
3.     includeProperties = null, bool isTracking = true)
4. {
5.     IQueryable<T> query = dbSet;
6.     if (filter != null)
7.     {
8.         query = query.Where(filter);
9.     }
10.    if (includeProperties != null)
11.    {
12.        foreach (var includeProp in includeProperties.Split(new char[] { ',' }
13.            , StringSplitOptions.RemoveEmptyEntries)){
14.            query = query.Include(includeProp);
15.        }
16.    }
17.    if (orderBy != null)
18.    {
19.        query = orderBy(query);
20.    }
21.    query = query.AsNoTracking();
22.    return await query.ToListAsync();
23.}

```

Розглянемо реалізацію MVC патерну на прикладі окремого компоненту платформи. Основним її компонентом можна вважати публікацію. Її можна створювати, редагувати та поширювати в соціальних мережах.

```

1. public class Post
2. {
3.     [Key]
4.     public int Id { get; set; }
5.     [Required]
6.     public string Name { get; set; }
7.     [AllowNull]
8.     public string? ShortDesc { get; set; }
9.     [AllowNull]
10.    public string? PostText { get; set; }
11.    public string Image { get; set; }
12.    [Display(Name = "Category Type")]
13.    public int? CategoryId { get; set; }
14.    [ForeignKey("CategoryId")]
15.    public virtual Category Category { get; set; }
16. }

```

Головна логіка обробки моделі описана в PostController. Це клас унаслідований від базового класу MVC Controller. У ньому описані HTTP запити для взаємодії. Клас має конструктор, де ініціалізуються IPostRepository, IWebHostEnvironment, IConfiguration та HttpClient.

```

1. public class PostController : Controller
2. {
3.     private readonly IPostRepository _postRepo;
4.     private readonly IWebHostEnvironment _webHostEnvironment;
5.     private readonly IConfiguration _configuration;
6.     private readonly HttpClient _httpClient;
7.     public PostController(IPostRepository prodRepo, IWebHostEnvironment
webHostEnvironment, IConfiguration configuration, HttpClient httpClient)
8.     {
9.         _postRepo = prodRepo;
10.        _webHostEnvironment = webHostEnvironment;
11.        _configuration = configuration;
12.        _httpClient = httpClient;
13.    }

```

Детально розглянемо HTTP запити PostController. HttpGet Index() повертає View зі списком всіх постів.

```

1. [HttpGet]
2. public async Task<IActionResult> Index()
3. {
4.     IEnumerable<Post> objList = await
_postRepo.GetAllAsync(includeProperties: "Category");
5.     return View(objList);
6. }

```

Останнім нерозглянутим компонентом MVC є представлення. Для його реалізації використовується спеціальний синтаксис, що є поєднанням кодів HTML та C#. Для прикладу розглянемо сторінку Post.Index. На ній розміщується список усіх публікацій платформи, а також кнопки для операцій CRUD.

```

1. @model IEnumerable<Diploma_Model.Models.Post>
2. <div class="container p-3">
3.     <div class="row pt-4">
4.         <div class="col-6"><h2 class="text-primary">Post List</h2></div>
5.         <div class="col-6 text-right"><a asp-action="Upsert" class="btn btn-primary">
6.             <i class="fas fa-plus"></i> Create New Post </a> </div></div>
7.     <br /><br />
8.     @if (Model.Count() > 0)
9.     { <table class="table table-bordered table-striped" style="width:100%">
10.         <thead><tr><th> Post Name </th> <th>Category</th><th></th></tr>
11.         </thead>
12.         <tbody>
13.             @foreach (var obj in Model)
14.             {
15.                 <tr><td width="30%">@obj.Name</td>
16.                 <td width="20%">@obj.Category.Name</td>
17.                 <td class="text-center">
18.                     <div class="w-75 btn-group" role="group">
19.                         <a asp-route-Id="@obj.Id" asp-action="Upsert" class="btn btn-primary mx-2">
20.                             <i class="fas fa-edit"></i> </a>
21.                         <a asp-route-Id="@obj.Id" asp-action="Delete" class="btn btn-danger mx-2">
22.                             <i class="far fa-trash-alt"></i> </a> </div></td></tr>
23.             }
24.         </tbody>
25.     </table>
26.     }
27.     else
28.     {
29.         <p> No product exists.</p>
30.     }
31. </div>

```

Кінцевий вигляд сторінки Post зображено на рис. 3.2.

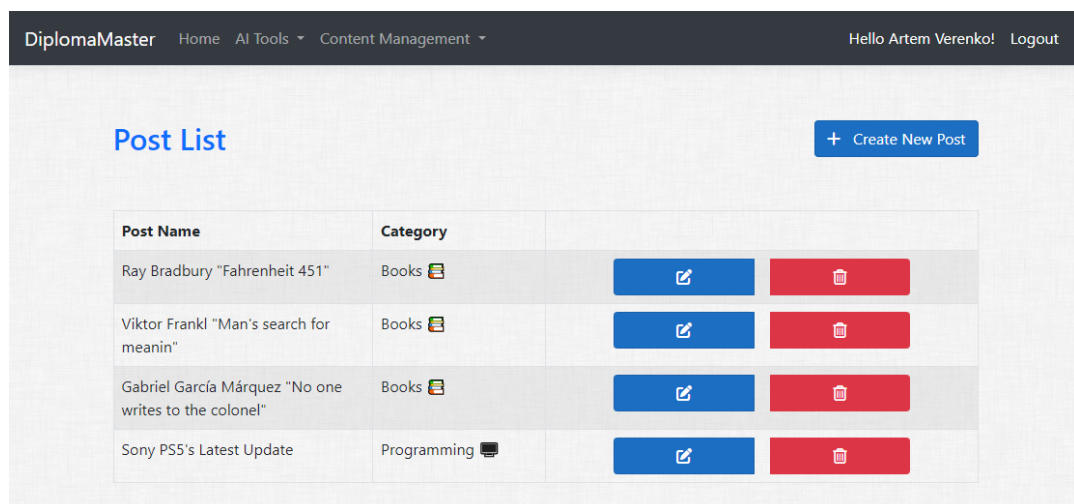


Рис. 3.2. Вигляд сторінки Post

Сторінка створення та редагування публікацій зображена на рис. 3.3.

The screenshot shows the 'Edit Post' interface of the DiplomaMaster application. The top navigation bar includes 'DiplomaMaster', 'Home', 'AI Tools', 'Content Management', and a user greeting 'Hello Artem Verenko! Logout'. The main form is titled 'Edit Post' and contains the following elements:

- Name:** A text input field containing 'Gabriel García Márquez "No one writes to the colo'.
- ShortDesc:** A text input field containing 'But... Someday, war will end.'
- PostText:** A rich text editor with a toolbar (bold, italic, underline, link, unlink, list, indent, outdent, code, help) and a text area containing a paragraph about Gabriel García Márquez and a list of hashtags: #gabriel_garcia_marquez, #writing, #poem, #philosophy, #psychology, #Meaning, #bookstagram, #booksofinstagram.
- Image Source:** Two radio buttons: 'Upload Locally' (selected) and 'Generate Image'.
- Choose Image:** A button labeled 'Choose Files' and a text field showing 'No file chosen'.
- Category Type:** A dropdown menu currently set to 'Books'.
- Buttons:** 'Update' (blue) and 'Back' (green) buttons at the bottom.
- Preview:** A small image of a book cover for 'No One Writes to the Colonel' by Gabriel García Márquez.

The footer of the application shows '© 2023 - DiplomaMaster'.

Рис. 3.3. Вигляд сторінки Post/Upsert

Отже в даному підрозділі було розглянуто програмну реалізацію основних компонентів системи управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Лістинг базового функціоналу веб-застосунку наведено у додатку В.

3.4 Програмна реалізація інтеграцій зі сторонніми сервісами

Реалізація інтеграцій зі сторонніми сервісами є важливою частиною онлайн платформи. У контексті даного проекту, інтеграції включають взаємодію з соціальними мережами та використання генеративних моделей штучного інтелекту, що надають додаткові можливості для генерування контенту та його розповсюдження. Платформа надає можливість публікувати пости в Twitter та Telegram, а також надсилати їх через Gmail. Реалізована

інтеграція з API OpenAI, зокрема з моделями GPT-4 та DALL-E-3, для генерування текстового та візуального контенту.

Сторонні API реалізовані в проєкті як окремі сервіси. Це видно на діаграмі класів розроблених сервісів (Додаток А). Розглянемо програмний код, що відправляє POST-запит до API OpenAI, передаючи підказку до OpenAI сервісу. Запит повертає результат користувачу у вигляді представлення.

```

1. [HttpPost]
2. public async Task<IActionResult> OpenAI_GPT_4(string prompt)
3. {
4.     var result = await _openAIService.GenerateTextAsync(prompt);
5.     ViewBag.Prompt = prompt;
6.     ViewBag.Result = result.IsSuccess ?
7.         result.Content : "An error occurred";
8.     return View("OpenAI_GPT_4");
9. }

```

Описуючи метод GenerateTextAsync, звернемо увагу на наступні особливості. Підготовка Запиту відбувається шляхом створення об'єкту payload, що містить дані для запиту до OpenAI API. Це включає модель AI ("gpt-4-1106-preview"), текстовий промпт від користувача та налаштування, як temperature та max_tokens. Поле messages формує відправлення двох повідомлень: системне повідомлення та користувацьке, де користувацьке повідомлення містить вхідний текст-підказку.

```

1. var payload = new
2. {
3.     model = "gpt-4-1106-preview",
4.     messages = new object[]
5.     {
6.         new { role = "system", content = $"Hi"},
7.         new {role = "user", content = prompt}
8.     },
9.     temperature = 0.3,
10.    max_tokens = _max_tokens
11. };

```

Об'єкт payload серіалізується у формат JSON за допомогою JsonConvert.SerializeObject. Створено HttpContent з JSON рядком, який використовується для відправки HTTP POST запиту. Запит відправляється до OpenAI API за допомогою _openAIClient.PostAsync. В разі успішного виконання повертається результат.

```

1. OpenAIResponseGPT4 response = null;
2. string jsonPayload = JsonConvert.SerializeObject(payload);
3. HttpContent httpContent = new StringContent(jsonPayload, Encoding.UTF8,
   "application/json");
4.
5. //Send the request
6. var responseMessage = await _openAIClient.PostAsync("https://api.openai.com/v1/chat/completions", httpContent);
7. if (responseMessage.IsSuccessStatusCode)
8. {
9.     var responseMessageJson = await responseMessage.Content.ReadAsStringAsync();
10.
11.     //Return a response
12.     response = JsonConvert.DeserializeObject<OpenAIResponseGPT4>(responseMessageJson);
13. }
14.
15. return new TextGenerationResult
16. {
17.     IsSuccess = responseMessage.IsSuccessStatusCode,
18.     Content = response.Choices[0].Message.Content
19. };

```

На платформі реалізований веб-інтерфейс пісочниця, де користувач може експериментувати з запитам до GPT 4 та DALL-E-3 (рис. 3.4). Окрім цього згадані моделі доступні безпосередньо під час створення чи редагування публікацій на сторінці. Результат генерації одразу можна зберегти як публікацію. Ще одною особливістю є кастомні адаптери контенту під конкретну соціальну мережу, де користувачеві пропонуються наперед підготовлені підказки, які користувач може модифікувати перед використанням.

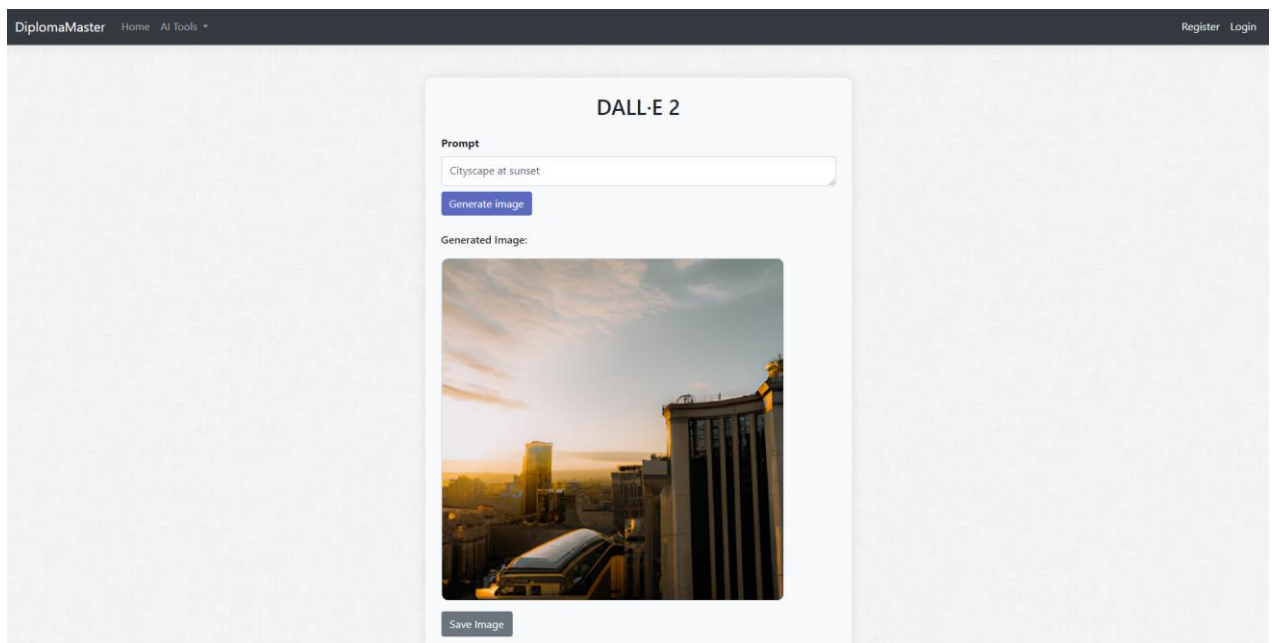


Рис. 3.4. Вигляд представлення OpenAI_GPT_4

Розглянемо реалізацію описаних адаптерів контенту на прикладі підготовки публікації до її анонсу в соціальній мережі Twitter. За взаємодію з Twitter відповідає `TwitterController`. `TwitterPostGenerate` запит, який використовує API OpenAI для генерації твіту за заданою підказкою. Він спочатку отримує пост з бази даних та створює об'єкт `TwitterAnnounceVM`. Потім він відправляє POST-запит до GPT та десеріалізує відповідь у тимчасовому сховищі.

```

1. [HttpPost]
2. public async Task<IActionResult> TwitterPostGenerate(int id, [FromForm]
   TwitterAnnounceVM model)
3. { var postFromDb = await _db.Post.Include(u => u.Category).FirstOrDefaultAsync(u =>
   u.Id == id);
4.     TwitterAnnounceVM twitterAnnounce = new TwitterAnnounceVM(postFromDb);
5.     string promptFromForm = model.Prompt;
6.     if (!string.IsNullOrEmpty(promptFromForm) && promptFromForm !=
       twitterAnnounce.DefaultPrompt)
7.     { twitterAnnounce.Prompt = promptFromForm; }
8.     var openAIResult = await _openAIService.GenerateTextAsync(promptFromForm, 65);
9.     var textResult = openAIResult.IsSuccess ? openAIResult.Content : "An error occurred
   during text generation.";
10.    TempData["Result"] = textResult;
11.    ViewBag.Prompt = model.Prompt;
12.    ViewBag.Result = textResult;
13.    return View("Index", twitterAnnounce);
14. }
```

Наступний код використовує API Twitter для публікації твіту. Запит отримує текст твіту з тимчасового сховища, створює об'єкт `TwitterClient` з використанням ключів API Twitter, отриманих з конфігурації застосунку. Далі він викликає метод `Execute.AdvanceRequestAsync()` для публікації твіту, встановлює тимчасове повідомлення про успішну публікацію твіту.

```

1. [HttpPost]
2. public async Task<IActionResult> TwitterPostAnnounce(int id)
3. {
4.     var textResult = TempData["Result"]?.ToString();
5.     if (string.IsNullOrEmpty(textResult))
6.     { TempData["TweetMessage"] = "There is no message to announce.";
7.       return RedirectToAction(nameof(Index), new { id = id });
8.     }
9.     var tweetResult = await _twitterService.PostTweetAsync(textResult);
10.    if (tweetResult.IsSuccess)
11.    {
12.        TempData["TweetMessage"] = "Tweet was sent successfully!";
13.    }
14.
15.    return RedirectToAction("Index", "Home");
16. }
```

Головне представлення Twitter Announcement зображено на рис. 3.5.

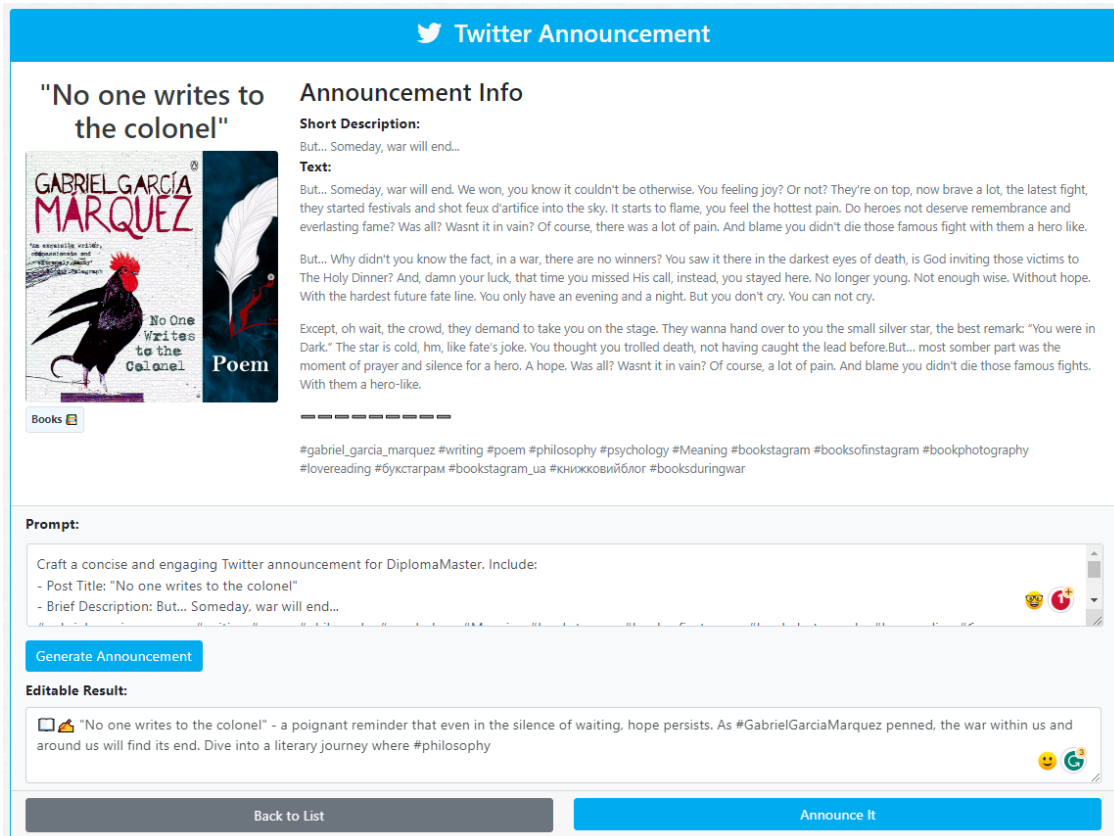


Рис. 3.5. Вигляд представлення Twitter Announcement

Отже, в підрозділі описано програмну реалізацію інтеграції сторонніх API сервісів від OpenAI. Продемонстровано лістинг програмного коду методів, а також деякі представлення. Лістинг наведено в додатку Д.

3.5 Тестування розробленого веб-застосунку

Тестування програмного забезпечення є ключовим етапом у розробці будь-якої програми. Цей процес включає технічний аналіз, який виявляє якість програмного продукту відповідно до вимог замовника. Залежно від ступеня знань про систему, тестування поділяється на кілька основних типів: тестування «білої скриньки», «чорної скриньки» та «сірої скриньки». Метод «білої скриньки» детально аналізує внутрішню структуру програми, включно з кодом, тоді як метод «чорної скриньки» сфокусований на зовнішній поведінці програми, тестуючи її через інтерфейс, як це робить кінцевий користувач.

У контексті розробки платформи створення контенту для соціальних мереж з інтеграцією генеративних моделей штучного інтелекту, було обрано метод «чорної скриньки», оскільки він дозволяє виявити помилки в логіці функціонування програми та потенційні неполадки у роботі основних алгоритмів, що є критично важливим для забезпечення якості кінцевого продукту.

Тестування платформи за методикою «чорної скриньки» передбачає перевірку правильності функціонування додатку при виконанні основних алгоритмів його використання та порівняння фактичного результату виконання із очікуваним. Такі алгоритми використання програмного додатку називають тест-кейсами. Тому для тестування було розроблено наступні тест-кейси:

Тест-кейс №1 – Увійти в систему як адміністратор:

1. Завантажити платформу;
2. Натиснути кнопку увійти на головному меню сайту;
3. Ввести існуючий логін адміністратора;
4. Ввести існуючий пароль адміністратора;
5. Натиснути кнопку увійти, що підтверджує вхід на сторінку;
6. Перевірити чи доступні на сайті інструменти адміністратора.

Очікуваним результатом даного тест-кейсу є успішна авторизація адміністратора. Це дає доступ до панелі керування на сайті (рис. 3.6).

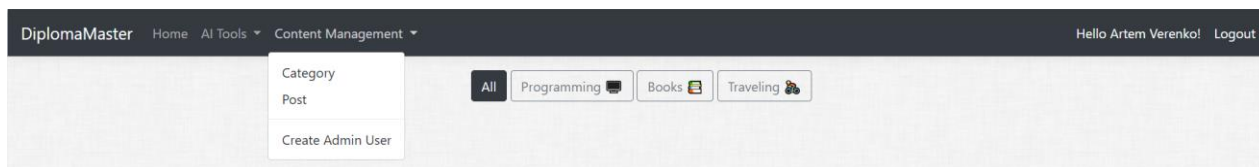


Рис. 3.6. Результат виконання тест-кейсу авторизації адміністратора

Тест-кейс №2 – Увійти в систему як звичайний користувач:

1. Завантажити платформу;
2. Натиснути кнопку вхід на головному меню сайту;
3. Ввести існуючий логін користувача сайту;

4. Ввести існуючий пароль користувача сайту;
5. Натиснути кнопку вхід, що підтверджує вхід на сторінку;
6. Перевірити чи вдалося авторизуватися.

Цей тест-кейс відрізняється від попереднього тим, що авторизація відбувається на акаунт звичайного користувача сайту. Результат тест-кейсу зображено на рис. 3.7.

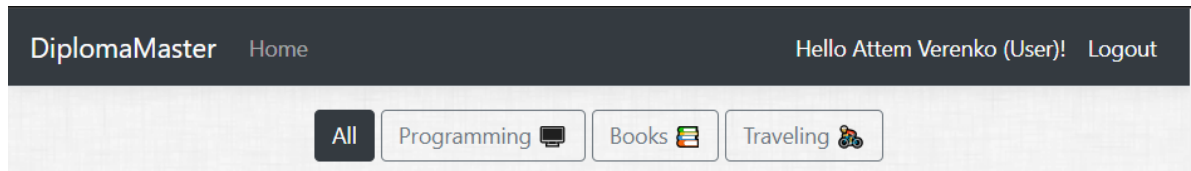


Рис. 3.7. Результат виконання тест-кейсу авторизації користувача

Тест-кейс №3 – Додати нову категорію публікацій:

1. Виконати дії тест-кейсу №1 (ввійти в систему як адміністратор);
2. Натиснути кнопку Category, що доступна на головному меню сайту;
3. Натиснути на кнопку Create New Category;
4. Ввести ім'я нової категорії;
5. Ввести порядковий номер відображення категорії;
6. Натиснути на кнопку Create.
7. Перевірити чи створена категорія

Даний тест кейс перевіряє можливість додавання категорії на сайт з допомогою панелі адміністратора. В даному випадку спробуємо додати нову категорію публікацій. Натиснувши на пункт меню Category ми перейдемо на сторінку списку вже існуючих категорій. Тут можна переглянути категорії, редагувати або видаляти їх, а також створювати нові. Натиснувши на кнопку Create New Category ми переходимо на сторінку створення категорії. Тут потрібно ввести назву категорії і порядковий номер за яким вона буде відображатися в списку. Після заповнення даних натискаємо на кнопку Create. Останнім кроком потрібно перевірити, чи категорія успішно створилася. Список доступних категорій на сайті рис. 3.8.

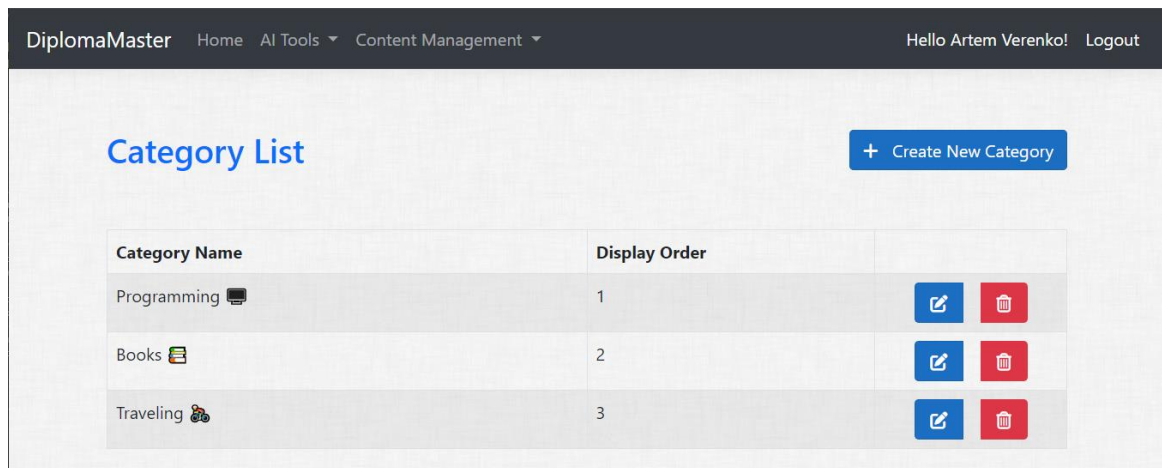


Рис. 3.8. Список наявних категорій на сайті

На рис. 3.9 відображена сторінка для створення нової категорії де користувач має ввести ім'я категорії та її порядковий номер відображення.

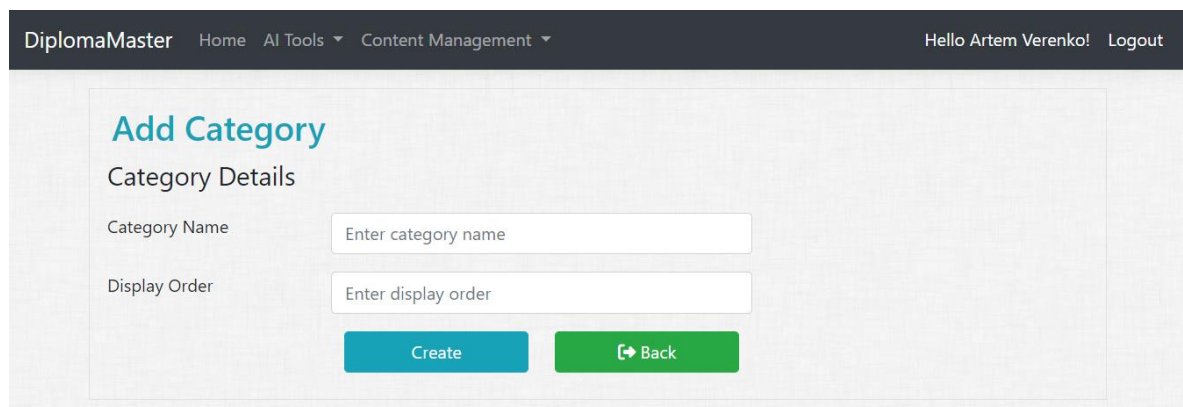


Рис. 3.9. Сторінка створення нової категорії

На рис. 3.10 ми бачимо нову категорію на сайті, що є підтвердженням успішного виконання тест-кейсу.

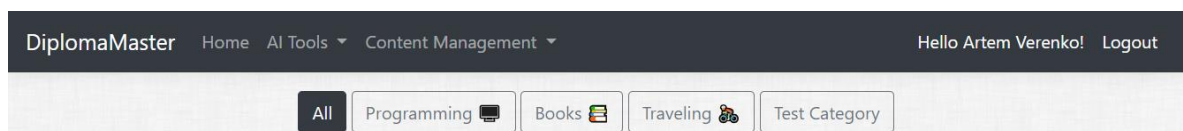


Рис. 3.10. Наявність нової категорії на головній сторінці

Тест-кейс №4 – Додати нову публікацію:

1. Виконати дії тест-кейсу №1 (ввійти в систему як адміністратор);
2. Натиснути кнопку Post, що доступна на головному меню сайту;

3. Натиснути на кнопку Create New Post;
4. Заповнити форму;
5. Вибрати опцію Generate Image;
6. Згенерувати обкладинку публікації ввівши необхідний промпт;
7. Натиснути на кнопку Create;
8. Перевірити чи створена нова публікація.

Даний тест кейс перевіряє можливість додавання контенту на сайт з допомогою панелі адміністратора. Загалом процес схожий на додавання нової категорії. Форма для створення публікації зображена на рис. 3.11. Для успішного створення публікації користувач має заповнити обов'язкові поля форми.

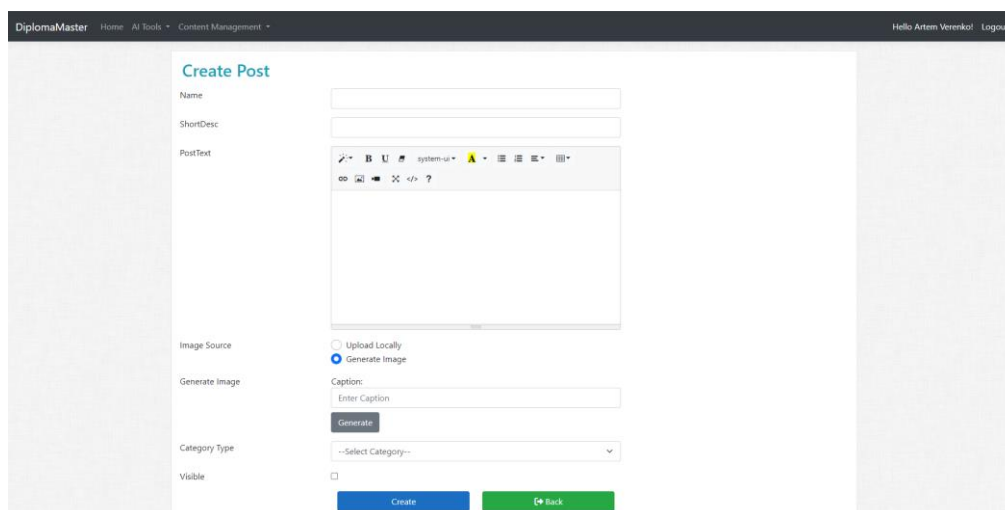


Рис. 3.11. Сторінка створення нової публікації

Для додавання обкладинки публікації користувачеві доступно дві опції: завантажити файл з локального сховища або згенерувати зображення звернувшись через інтерфейс платформи до OpenAI API.

На рис. 3.12 ми бачимо нову публікацію на сайті, що є підтвердженням успішного виконання тест-кейсу.

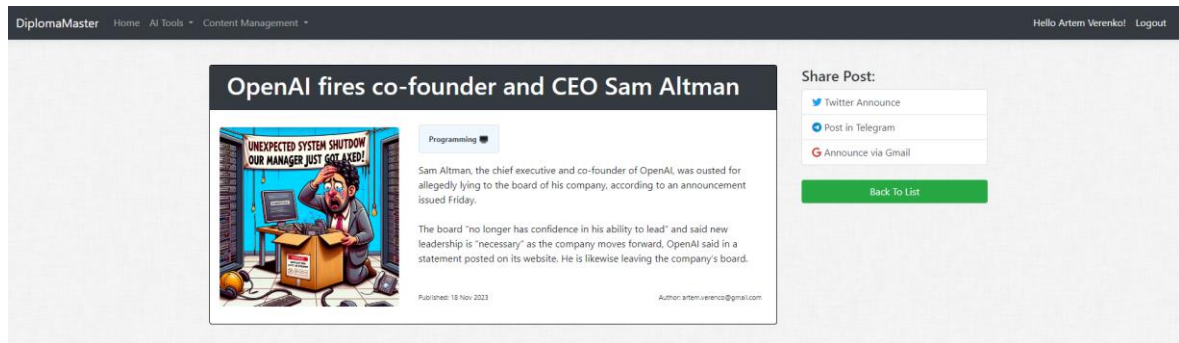


Рис. 3.12. Створена публікація Тест-кейс №4

Тест-кейс №5 – Анонс публікації в Twitter:

1. Виконати дії тест-кейсу №1 (ввійти в систему як адміністратор);
2. Натиснути кнопку View Details на публікації;
3. Натиснути на кнопку Twitter Announce;
4. Натиснути на кнопку Generate Announcement;
5. Переглянути згенерований анонс;
6. Натиснути на кнопку Announce It;
7. Перевірити чи опублікувався анонс в Twitter.

Після виконання описаних кроків отримуємо опублікований анонс в Twitter (рис. 3.13).

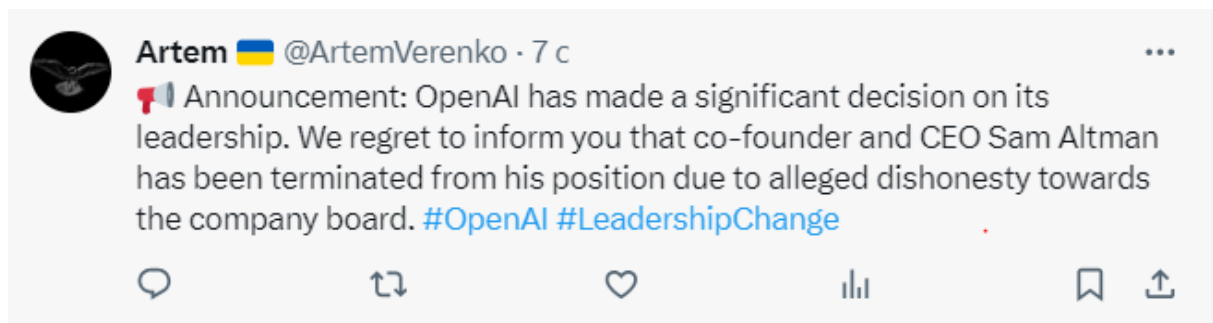


Рисунок 3.13. Опублікувався анонс в Twitter Тест-кейс №5

Для автоматизації даних тестів була реалізований фреймворк автоматичного тестування. Його архітектурні складові: фреймворк автоматизації та проект з тестами видно на діаграмі класів (Додаток А). Для його реалізації використаний Selenium, що є стандартом у сфері автоматизованого тестування веб-додатків. Selenium дозволяє тестувальникам

взаємодіяти з різними компонентами веб-інтерфейсу, імітуючи користувача, що дає можливість виявлення помилок у реальних умовах користування .

Структура фреймворку, розробленого згідно з патерном «page object», забезпечує ефективну організацію тестового коду, розділяючи визначення елементів веб-сторінки та дій з ними на окремі модулі. Це дозволяє легко модифікувати та обслуговувати тестові скрипти при змінах у веб-додатку, що є критично важливим для динамічно розвиваючихся проєктів.

Запропоновані тест-кейси виконано успішно (рис. 3.14).

Test	Duration	Traits	Error Message
▲ ✓ Dipoma.Automation.Tests ...	5.3 sec		
▲ ✓ Dipoma.Automation.Tes...	4.1 sec		
▲ ✓ ContentTest (1)	4.1 sec		
✓ AddCategory	4.1 sec		
▲ ✓ Dipoma.Automation.Tests	1.2 sec		
▲ ✓ UserLifeCycleTest (2)	1.2 sec		
✓ LoginUserAsAdmin	459 ms		
✓ LoginUser	768 ms		

Рис. 3.14. Успішне виконання автоматичних тестів

Отже, у розділі детально описано процес тестування програмного забезпечення для платформи створення контенту для соціальних мереж, з акцентом на використання методу "чорної скриньки". Описано виконання різноманітних тест-кейсів, від авторизації адміністратора до публікації контенту та анонсування у Twitter. Використання Selenium та патерну "page object" для автоматизації тестування забезпечує ефективне виявлення та виправлення помилок, що значно підвищує надійність та якість кінцевого продукту. Успішне виконання всіх тестів підтверджує, що розроблена платформа відповідає встановленим вимогам та готова до експлуатації.

Код даного тестового фреймворку розмішений в додатку Е.

3.6 Висновки

У висновках можна підкреслити, що було успішно розроблено веб-застосунок, який інтегрує передові підходи та технології у відповідності до сучасних стандартів. Обрано .NET 7 і ASP.NET Core MVC як основу для створення надійної, гнучкої та високопродуктивної архітектури, що забезпечує чистоту коду та спрощення розробки.

Ефективне розділення бізнес-логіки та користувацького інтерфейсу досягнуто за допомогою шаблону MVC, що важливо для підтримки та розвитку програмного продукту. Окрім того, враховано безпеку та управління ідентифікацією користувачів за рахунок впровадження ASP.NET Core Identity для забезпечення надійного захисту даних користувачів.

Інтеграція з генеративними моделями штучного інтелекту OpenAI відкрила нові можливості для автоматизації створення контенту, збільшуючи ефективність роботи з контентом та надаючи інноваційні інструменти для креативного використання.

Тестування розробленого веб-додатку, проведене за методом «чорної скриньки» підтвердило відповідність системи встановленим вимогам та її готовність до експлуатації.

Розроблений веб-застосунок виявився універсальним рішенням, придатним для використання у різноманітних сферах, включаючи системи управління контентом та соціальні мережі, завдяки своїй адаптивності, масштабованості та інтеграції з передовими технологіями. Таким чином, успішно реалізовано програмний продукт, що відповідає сучасним вимогам та очікуванням ринку.

Екрани реалізованої програми представлені в додатку Ж.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ПЛАТФОРМИ СТВОРЕННЯ КОНТЕНТУ ДЛЯ СОЦІАЛЬНИХ МЕРЕЖ З ІНТЕГРАЦІЄЮ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

4.1 Розробка підказок для генерування контенту з використанням OpenAI API

Розробка ефективних підказок для генерації контенту за допомогою OpenAI API є ключовим аспектом у створенні застосунків, що використовують можливості штучного інтелекту для автоматизації та оптимізації процесів створення контенту в соціальних мережах. Підказки, або вхідні дані, слугують важливою інструкцією для моделей штучного інтелекту, які генерують текстовий чи візуальний контент, тому їх розробка вимагає особливої уваги та розуміння основних принципів роботи цих технологій.

Перш за все, важливо зрозуміти, що ефективність підказок залежить від їх здатності чітко та конкретно формулювати запити до AI-систем. Це означає, що користувачі повинні вказувати не лише тему, але й контекст, стиль, обсяг та інші специфічні вимоги до генерованого контенту. Наприклад, якщо потрібно створити текст для посту в соціальній мережі, користувач повинен вказати не лише тему, але й бажаний стиль, тональність, і, можливо, ключові слова, які мають бути включені.

Другим важливим аспектом є здатність підказок адаптуватися до різних форматів та стандартів соціальних мереж. Кожна соцмережа має свої унікальні характеристики та вимоги до контенту, тому підказки повинні враховувати ці особливості. Наприклад, текст для Twitter має бути коротким і лаконічним, в той час як пост для блогу може бути більш детальним і розгорнутим.

У випадку з Twitter, через обмеження у кількості символів на один твіт, система розроблена таким чином, щоб публікувати лише анонси постів з посиланням на повний контент. Це дозволяє залучити увагу читачів і спонукати їх перейти за посиланням для отримання детальнішої інформації.

Розроблений промпт є шаблоном, куди підставляється інформація про конкретний пост.

1. DefaultPrompt = \$"Create an announcement for publication on Twitter using the following information: Platform : DiplomaMaster, Post Name : {Post?.Name}, Brief description : {Post?.ShortDesc}. Use the twitter format";

В результаті користувач отримує згенерований анонс, який він може реагувати та публікувати (рис 4.1).

Prompt:

Create an announcement for publication on Twitter using the following information: Platform : DiplomaMaster, Post Name : "No one writes to the colonel", Brief description : But... Someday, war will end.... Use the twitter format

Generate Announcement

Editable Result:

Exciting news from #DiplomaMaster! 📰
 Our latest post "No one writes to the colonel" is now live!
 Dive into a narrative where hope lingers - "Someday, war will end...."
 Read, reflect, and join the conversation!

Back to List **Announce It**

Рис. 4.1. Вигляд процесу генерування анонсу для Twitter

Опублікований пост зображено на рис. 4.2.



Рис. 4.2. Вигляд опублікованого анонсу в Twitter

Для Telegram та Gmail система адаптує контент так, щоб він відображався цілісно та відповідав форматам цих платформ. У Telegram пости публікуються в повному обсязі, забезпечуючи просте та зручне читання. Для Gmail, враховуючи можливості цієї платформи відображати HTML-контент, система адаптує пости до HTML формату, що дозволяє інтегрувати в них

різноманітні візуальні елементи, такі як: зображення, різні стилі тексту та інтерактивні компоненти.

Ключовою особливістю системи є її гнучкість у роботі з промптами. Користувачам пропонуються базові шаблони промптів, які вони можуть використовувати для швидкого створення контенту. Проте, система також надає можливість користувачам модифікувати ці промпти або створювати власні з нуля відповідно до їхніх унікальних потреб та цілей. Це дозволяє користувачам забезпечити більшу персоналізацію контенту та адаптувати його до специфіки їхньої аудиторії чи бренду.

Розроблені промпти для Telegram та Gmail відображають унікальні можливості кожної платформи, забезпечуючи оптимальний візуальний та текстовий формат для кожного типу контенту:

1. DefaultPrompt = \$"Prepare a full post for publication on Telegram using the following details:\r\n\r\nPlatform: DiplomaMaster\r\nPost Title: {Post.Name}\r\nDescription: {Post.ShortDesc}\r\nFull Text: {Post.PostText}\r\n\r\nPlease ensure the post is formatted for clear readability and includes all necessary details for a comprehensive update to the audience. Include relevant hashtags and format it appropriately for the Telegram platform.";
2. DefaultPrompt = \$"Create a clean, responsive HTML email for Gmail, dedicated to announcing a post on the DiplomaMaster platform. The email should strictly contain the following elements and no other text:\r\n\r\nEmail Subject: {Post.Name}\r\nBrief Description: {Post.ShortDesc}\r\nFull Text: {Post.PostText}\r\n\r\nEnsure the email is formatted for clear readability and uses inline CSS for the layout of the header and body. The email should only display the information provided above, ready for publication.";

Вигляд представлень адаптерів контенту для Telegram та Gmail зображено на рис. 4.3.

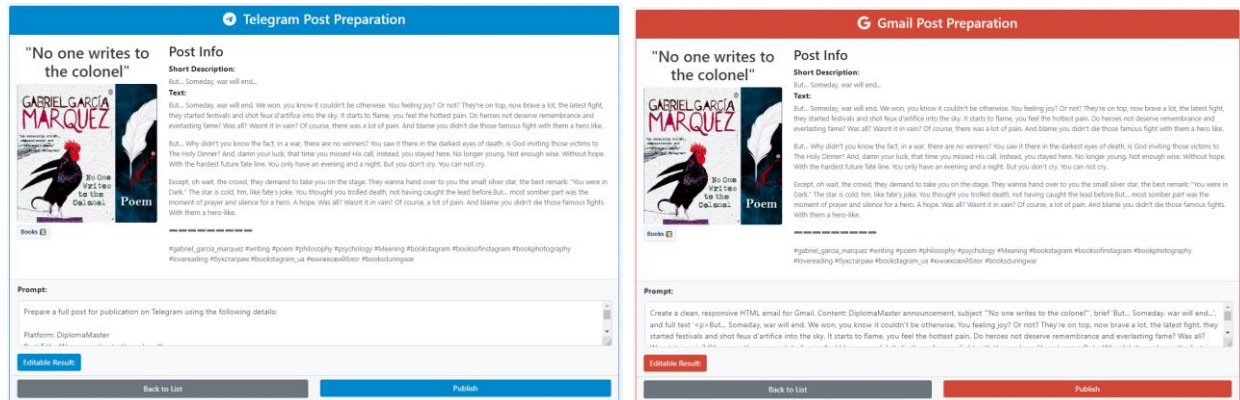


Рис. 4.3. Вигляд представлень адаптерів контенту для Telegram та Gmail

Вигляд опублікованого посту в Telegram зображено на рис. 4.4.

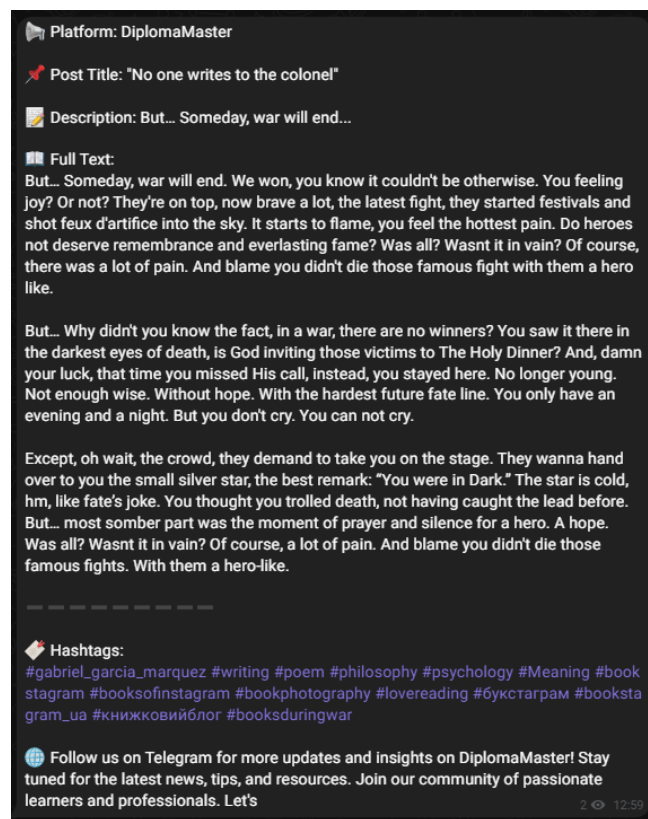


Рис. 4.4. Вигляд опублікованого посту в Telegram

Таким чином, розроблена система представляє собою комплексне рішення для ефективного управління контентом в соціальних мережах, яке не лише спрощує процес генерації контенту, але й забезпечує його високу адаптивність до різноманітних платформ і потреб користувачів.

4.2 Оцінка ефективності веб-застосунку для генерації контенту





Оцінка ефективності веб-застосунку для генерації контенту у соціальні мережі є важливою для розуміння його вартості, практичної придатності та впливу на робочі процеси користувачів. Вона допомагає виявити сильні сторони веб-застосунку та потенційні можливості для його вдосконалення, забезпечуючи цінну інформацію для його подальшого розвитку та оптимізації.

Для об'єктивної оцінки спочатку розглянемо приклади згенерованого або адаптованого контенту, що створений за допомогою інструментів веб-застосунку. Так в таблиці 4.1 можна наявно побачити якість згенерованих

зображень за допомогою API DALL-E-3, що слугують обкладинками для публікацій.

Таблиця 4.1

Огляд можливостей генерації зображень розробленим веб-застосунком













































Публікація	Згенерована обкладинка
<p>1) Review of the book «Brave New World» by Aldous Huxley</p> <p>"But I don't want comfort. I want God, I want poetry, I want real danger, I want freedom, I want goodness. I want sin."</p> <p>✍️ You are a developer, and you can create any reality whatever you want. Think about it. It's not simply enough. You write a line, one or many. Your world is static. But you want a move, a growth, some kind of self-improvement. How can you reach it? ...</p>	
<p>2) Review of the book «Fahrenheit 451» by Ray Bradbury</p> <p>Another dystopian novel. Another experiment with people's world.</p> <p>📖 Which thing has the greatest value? Important question without a simple answer. Maybe, it will be close to say that is the common culture what unconsciously spread between all of us.</p> <p>📖 Now we speak about "Fahrenheit 451" - one of the most powerful and popular book on the world. Reading it you can really touch to the fire. ...</p>	
<p>3) Review of the book «The Practice and Theory of Individual Psychology» by Alfred Adler</p> <p>The point where we start is Frankl's quote about the meaning of life.</p> <p>🔗 "Life is not primarily a quest for pleasure, as Freud believed, or a quest for power, as Alfred Adler taught, or a quest for economic, supplant Marx, but a quest for meaning. The greatest task for any person is to find it in his or her life."</p> <p>...</p>	
<p>4) Post about OpenAI's Kotlin library</p> <p>🚀 Exciting News for Android Developers!</p> <p>Last week, we delved into OpenAI's latest feature - Assistants - in the web playground. Today, we're thrilled to announce that the OpenAI Kotlin library is swiftly being updated to include these new APIs. Android developers, it's time to get your hands on these cutting-edge tools with snapshot package builds!</p> <p>...</p>	

Зображення, створені за допомогою API DALL-E-3, демонструють вражаючу деталізацію, глибину та відповідаючи тематиці текстів публікацій.

Наступними розглянемо результати роботи адаптерів контенту на прикладі створень анонсів для платформи Twitter про вихід нової публікації на платформі (Таблиця 4.2).

Таблиця 4.2

Опубліковані згенеровані анонси в Twitter

1	<div><div></div><div><div>Artem </div><div>@ArtemVerenko · Зараз</div></div><div><div>Рекламувати</div><div>...</div></div><div><div></div><div>New insights on #DiplomaMaster: Dive into Aldous Huxley's "Brave New World." A quest beyond comfort—seeking God, poetry, danger, and freedom in a static world. Uncover the pursuit of self-improvement in our latest review!   #BraveNewWorld #BookReview</div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>
2	<div><div></div><div><div>Artem </div><div>@ArtemVerenko · 2 хв</div></div><div>...</div><div><div></div><div>#DiplomaMaster unveils the power of "Fahrenheit 451"! Explore Ray Bradbury's fiery dystopia where culture's value burns at the heart of society. Can you feel the heat? Read our review and join the conversation!   #Fahrenheit451 #DystopianNovel</div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>
3	<div><div></div><div><div>Artem </div><div>@ArtemVerenko · 3 хв</div></div><div>...</div><div><div></div><div>On #DiplomaMaster: Alfred Adler's quest for meaning in "The Practice and Theory of Individual Psychology." Beyond pleasure and power—find life's greatest task with us. Delve into the profound review now!   #Adler #Psychology</div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>
4	<div><div></div><div><div>Artem </div><div>@ArtemVerenko · 3 хв</div></div><div>...</div><div><div></div><div>#DiplomaMaster celebrates #AndroidDev evolution! OpenAI's Kotlin library gets a breakthrough with new API updates. Ready your code for a leap into the future with our latest post on these tech marvels!   #OpenAIKotlin #TechUpdate</div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>

Згенеровані анонси для платформи Twitter відзначаються своєю здатністю лаконічно та влучно передавати суть публікацій, враховуючи обмеження соціальної мережі. Ефективно використовуються хештеги для збільшення видимості та привертання уваги аудиторії а також наліпки для

більшого емоційного резонансу. Анонси створені таким чином, щоб заохочувати користувачів до взаємодії, стимулюючи перехід на платформу #DiplomaMaster для повного ознайомлення з матеріалом.

В рамках оцінки ефективності веб-застосунку для генерації контенту у соціальних мережах, окрім аналізу якості візуального контенту та ефективності створення анонсів у Twitter, важливо також оцінити інші ключові аспекти системи (таблиця 4.3). Це включає швидкість генерації контенту, якість, зручність інтерфейсу, дотримання тематики та гнучкість налаштувань. Такий підхід дозволить отримати всебічне уявлення про продуктивність та універсальність веб-застосунку.

Таблиця 4.3

Характеристики ефективності веб-застосунку для генерації контенту

Критерій Оцінки	Опис Критерію	Результати Оцінки	Примітки
Швидкість генерації	Час відправки запиту до отримання результату	4.5 сек	Швидкість генерації в межах очікуваних показників
Якість контенту	Відповідність згенерованого контенту очікуванням користувача	Висока	Контент відповідає більшості вимог користувачів
Зручність інтерфейсу	Зручність та інтуїтивність користувацького інтерфейсу	Дуже зручний	Інтерфейс є легким для навігації
Дотримання тематики	Ступінь відповідності згенерованого контенту заданій темі або контексту	Висока	Модель добре відтворює задану тематику
Гнучкість налаштувань	Можливість користувача адаптувати налаштування під свої потреби	Гнучкі	Система дозволяє користувачам легко змінювати налаштування

Отримані характеристики свідчать про високу ефективність веб-застосунку у генерації контенту для соціальних мереж.

4.3 Можливості практичного застосування

Розглянемо практичне застосування розробленої системи автоматизації створення, редагування та публікації контенту в соціальних мережах. Система використовує інтеграцію з генеративними моделями штучного інтелекту, такими як: GPT-4 та DALL-E-3 від OpenAI, для створення текстового та візуального контенту та Twitter, Telegram, Gmail API для його поширення.

Ключовим аспектом системи є її здатність генерувати контент за допомогою штучного інтелекту. Користувачі можуть створювати привабливі та оригінальні пости, використовуючи прості вхідні промпти. Це включає автоматизацію генерації заголовків, текстів, хештегів та навіть візуального контенту.

Система має вбудовані механізми для адаптації контенту до різних соціальних мереж, зокрема Twitter, Telegram та Gmail. Вона автоматично налаштовує контент під особливості кожної платформи, наприклад, короткі та лаконічні твіти для Twitter та більш детальні повідомлення для Telegram та Gmail.

Система інтегрується з різними соціальними мережами та генеративними моделями штучного інтелекту, що дозволяє з легкістю публікувати контент на різних платформах. Це не лише спрощує процес розповсюдження контенту, але й забезпечує його відповідність стандартам та форматам кожної соціальної мережі.

Система має високі стандарти безпеки для захисту даних користувачів, включаючи шифрування даних та політики доступу.

Інтерфейс системи розроблено з акцентом на інтуїтивність та простоту використання, що робить процес створення та публікації контенту доступним для широкого кола користувачів.

Описані особливості системи відкривають широкі можливості її комерційного застосування. Так система може бути використана в маркетингових та рекламних стратегіях компаній, де допомагатиме

автоматизувати та оптимізувати процеси створення контенту. Це сприятиме підвищенню залучення аудиторії та популяризації бренду. Для блогерів та інфлюенсерів, така система є зручним інструментом для підтримки регулярної активності у соціальних мережах. Малі та середні підприємства можуть значно виграти від її використання, оскільки вона дозволяє самостійно керувати присутністю в соціальних мережах без необхідності наймати додаткових спеціалістів.

Розроблена система також має значний потенціал для дослідницьких цілей. Її компоненти, такі як: адаптери контенту та налаштовані API соціальних мереж, можуть бути використані для вивчення специфічних аспектів цифрової взаємодії та комунікації. Це відкриває можливості для експериментування в областях обчислювальної лінгвістики, аналізу соціальних медіа та дослідження впливу штучного інтелекту на медіа-контент.

Таким чином, розроблена система стає не лише інструментом для ефективного управління контентом у соціальних мережах, але й цінним ресурсом для наукових досліджень та інноваційного розвитку. Вона демонструє як практичну цінність в комерційному використанні, так і значущість для наукової спільноти, відкриваючи нові горизонти для досліджень та розвитку в галузі штучного інтелекту та цифрової комунікації.

4.4 Наукові перспективи

Розглядаючи наукові перспективи розробленої системи для автоматизації створення, редагування та публікації контенту в соціальних мережах, важливо відзначити кілька ключових напрямків, які відкривають нові можливості для досліджень та інновацій. По-перше, використання генеративних моделей штучного інтелекту у цій системі становить значний інтерес для сфери машинного навчання та штучного інтелекту. Вивчення їх ефективності, адаптивності та здатності до удосконалення може привести до розвитку більш точних і надійних алгоритмів. Також система може слугувати важливим інструментом для аналізу соціальних мереж, дозволяючи глибше

зрозуміти поведінку користувачів, тенденції контенту та їх вплив на суспільство та культуру.

Важливим аспектом є внесок системи у галузь обчислювальної лінгвістики, оскільки вона обробляє та генерує текстовий контент, що відкриває нові можливості для вивчення мовних моделей та їх здатності створювати природній, зрозумілий текст. Це може значно сприяти розвитку цієї галузі. Крім того, система може бути використана для дослідження персоналізації контенту та його впливу на аудиторію, що є актуальним у контексті взаємодії у цифровому просторі.

Нарешті, дослідження у сфері взаємодії людини та комп'ютера можуть отримати новий імпульс завдяки аналізу того, як користувачі взаємодіють з системою та її інтерфейсом. Це дозволить розробити більш інтуїтивні та зручні інтерфейси, забезпечуючи кращу взаємодію між людиною та цифровими технологіями.

Важливо також відзначити, що метод цієї системи був представлений у вигляді тез на науковій конференції. Це свідчить про потенціал системи як предмету наукових досліджень та її значимість у контексті сучасних тенденцій у цифровій комунікації та технологіях.

У сукупності, наукові перспективи розробленої системи охоплюють широкий спектр досліджень, від машинного навчання до етики та безпеки даних, відкриваючи нові горизонти для інновацій та розвитку знань у сфері цифрових технологій та їх застосування у різноманітних аспектах сучасного життя.

4.5 Результати виконання кваліфікаційної роботи

Відповідно до поставлених завдань у магістерській кваліфікаційній роботі було досягнуто наступні результати:

– Було проведено глибокий аналіз сучасних технологій та підходів у сфері генеративних моделей штучного інтелекту. Це включало огляд

найновіших досліджень та розробок в області штучного інтелекту з акцентом на їх застосування для створення контенту в соціальних мережах;

- Була створена концептуальна модель системи створення контенту для соціальних мереж, що інтегрує генеративні моделі штучного інтелекту. Модель охоплює всі ключові аспекти від ідей до публікації кінцевого контенту забезпечуючи ефективність давного процесу;

- В рамках роботи були вибрані та впроваджені конкретні моделі штучного інтелекту: GPT-4 та DALL-E-3 від OpenAI. Вони найкраще відповідали вимогам проекту. Ці моделі були інтегровані для оптимізації процесів створення та редагування контенту;

- Розроблений веб-застосунок включає інструменти для генерування, оптимізації та розповсюдження контенту, а також засоби для персоналізації і адаптації контенту під різні соціальні мережі. Система була оснащена інтуїтивно зрозумілим інтерфейсом для забезпечення легкої взаємодії з користувачами;

- Було проведено ретельне тестування розробленої системи, використовуючи методику «чорної скриньки», що дозволило оцінити її функціональність, стабільність та ефективність. Результати тестування підтвердили високу продуктивність системи та її придатність для використання в реальних умовах.

- Було проведено розробку підказок для генерування контенту з використанням OpenAI API, дано оцінку ефективності веб-застосунку для генерації контенту, визначені можливості практичного застосування та вірогідні наукових перспективи.

Загалом, отримані результати демонструють успішне вирішення поставлених завдань та внесок у розвиток технологій створення контенту для соціальних мереж, використовуючи можливості сучасних генеративних моделей штучного інтелекту.

4.6 Висновки

У четвертому розділі, який присвячений дослідженню платформи для створення контенту для соціальних мереж з інтеграцією генеративних моделей штучного інтелекту, розглянуто ряд важливих аспектів та досягнень роботи.

Описано процес розробки ефективних підказок для OpenAI API, що є ключовим елементом у створенні адаптивного контенту. Адаптивність контенту до різних платформ, таких як: Twitter, Telegram та Gmail є ключовою особливістю системи. Користувач має можливість модифікувати базові шаблони промптів або створювати власні для більшої персоналізації контенту.

Оцінка ефективності розробленого веб-застосунку підтвердила його високу продуктивність і практичність. Ключові показники, таких як: швидкість генерації, якість контенту, зручність інтерфейсу та дотримання тематики дозволяють всебічно оцінити роботу системи. Результати вказують на ефективність використаних моделей штучного інтелекту, зручність користувацького інтерфейсу та гнучкість системи в адаптації до різних користувацьких вимог. Це підкреслює потенціал веб-застосунку як цінного інструменту для генерації контенту в широкому спектрі застосувань.

Опис практичного застосування системи виявив її значний потенціал, як у комерційному використанні, так і у дослідницькій сфері

У науковій перспективі, система відкриває нові можливості для досліджень у сфері штучного інтелекту, обчислювальної лінгвістики та взаємодії людини з комп'ютером. Метод системи, який був представлений на науковій конференції, підкреслює її значення для сучасних тенденцій у цифровій комунікації та технологіях.

Система відповідає сучасним вимогам та потребам, водночас відкриваючи нові горизонти для подальших досліджень і розвитку в галузі цифрових технологій.

ВИСНОВКИ

Магістерська кваліфікаційна робота демонструє успішну розробку інноваційної системи створення контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту. Система виявилася ефективною у спрощенні процесу створення контенту, зменшуючи потребу в часі та ресурсах для його генерації.

Незважаючи на успіхи, дипломна робота також висвітлює важливість збереження балансу між автоматизацією та людською креативністю, що є ключовим викликом у цій галузі.

Світові тенденції та можливості використання:

- Робота актуальна у контексті світових тенденцій у сфері цифрових технологій, особливо у галузі штучного інтелекту та обчислювальної лінгвістики;

- Система може бути використана у різноманітних сферах, включаючи маркетинг у соціальних мережах, особисте використання для креативного контенту;

- Знання та технології, розроблені в рамках цієї роботи, мають потенціал для подальших інновацій та розвитку в галузі штучного інтелекту.

Соціальна значимість полягає у здатності системи полегшити та автоматизувати процеси створення контенту.

Перспективними напрямками подальших досліджень є розробка додаткових функцій та інструментів для підвищення точності та релевантності автоматично генерованого контенту та дослідження впливу автоматизації створення контенту на поведінку користувачів та взаємодію у соціальних мережах.

СПИСОК ЛІТЕРАТУРИ

1. Attention is all you need [Електронний ресурс] / Ashish Vaswani [та ін.] // Computation and Language. – 2017. – Режим доступу: <https://doi.org/10.48550/arXiv.1706.03762> (дата звернення: 12.06.2023).
2. Kurochkin A. Generation of Memes to Engage Audience in Social Media [Електронний ресурс] / Andrew Kurochkin, Kostiantyn Bokhan. – 2019. – Режим доступу: <http://ceur-ws.org/Vol-2566/MS-AMLV-2019-paper11-p010.pdf>. (дата звернення: 12.06.2023).
3. Language Models are Unsupervised Multitask Learners [Електронний ресурс] / Alec Radford [та ін.] // Computer Science. – [Б. м.], 2019. – Режим доступу: <https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe> (дата звернення: 21.05.2023).
4. Liu V. Design Guidelines for Prompt Engineering Text-to-Image Generative Models [Електронний ресурс] / Vivian Liu, Lydia B. Chilton // CHI '22: CHI Conference on Human Factors in Computing Systems. – New Orleans LA USA, 2022. – С. 1–23. – Режим доступу: <https://doi.org/10.1145/3491102.3501825> (дата звернення: 28.03.2023).
5. Why We Need New Evaluation Metrics for NLG [Електронний ресурс] / Jekaterina Novikova [та ін.]. – Copenhagen, Denmark, 2017. – Режим доступу: <https://doi.org/10.18653/v1/D17-1237> (дата звернення: 21.05.2023).
6. StyleNet: Generating Attractive Visual Captions with Styles [Електронний ресурс] / Chuang Gan [та ін.] // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – Honolulu, HI, 2017. – С. 955–964. – Режим доступу: <https://doi.org/10.1109/CVPR.2017.108> (дата звернення: 21.05.2023).
7. Polonsky M. J. Should Artificial Intelligent Agents be Your Co-author? [Електронний ресурс] / Michael Jay Polonsky, Jeffrey D. Rotman // Australasian Marketing Journal. – 2023. – Т. 31, № 2. – С. 91–96. – Режим доступу: <https://doi.org/10.1177/14413582231167882> (дата звернення: 22.05.2023).

8. Feldman P. The Keyword Explorer Suite: A Toolkit for Understanding Online Populations [Електронний ресурс] / Philip Feldman, James R. Foulds. – Режим доступу: <http://arxiv.org/abs/2301.05198> (дата звернення: 22.05.2023).
9. Saravanan S. GPT-3 Powered System for Content Generation and Transformation [Електронний ресурс] / Shruti Saravanan, K. Sudha // 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT). – Sonapat, India, 2022. – С. 514–519. – Режим доступу: <https://doi.org/10.1109/CCICT56684.2022.00096> (дата звернення: 22.05.2023).
10. The rise of social bots [Електронний ресурс] / Emilio Ferrara [та ін.] // Communications of the ACM. – 2016. – Т. 59, № 7. – С. 96–104. – Режим доступу: <https://doi.org/10.1145/2818717> (дата звернення: 23.05.2023).
11. Katerynych L. Transformer-based Model for Text Classification in Ukrainian [Електронний ресурс] / Larysa Katerynych – 2022. – Режим доступу: https://ceur-ws.org/Vol-3179/Short_6.pdf (дата звернення: 12.06.2023).
12. Zagorulko D. I. Chatgpt in newsrooms: adherence of ai-generated content to journalism standards and prospects for its implementation in digital media [Електронний ресурс] / D. I. Zagorulko // "Scientific notes of V. I. Vernadsky Taurida National University", Series: "Philology. Journalism". – 2023. – Т. 2, №1. – С. 319–325. – Режим доступу: <https://doi.org/10.32782/2710-4656/2023.1.2/50> (дата звернення: 11.06.2023).
13. A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT [Електронний ресурс] / Yihan Cao [та ін.] // Artificial Intelligence. – 2023. – Режим доступу: <https://doi.org/10.48550/ARXIV.2303.04226> (дата звернення: 11.06.2023).
14. A Survey on ChatGPT: AI-Generated Contents, Challenges, and Solutions [Електронний ресурс] / Yuntao Wang [та ін.] // Computers and Society. – 2023. – Режим доступу: <https://doi.org/10.48550/ARXIV.2305.18339> (дата звернення: 11.06.2023).

15. AI-Generated Content (AIGC): A Survey [Електронний ресурс] / Jiayang Wu [та ін.] // Artificial Intelligence. – 2023. – Режим доступу: <https://doi.org/10.48550/ARXIV.2304.06632> (дата звернення: 11.06.2023).
16. Веренько А. Особливості використання інтернет-протоколів для управління електронною поштою [Електронний ресурс] / Артем Веренько // Стан, досягнення та перспективи інформаційних систем і технологій. – 2022. – Режим доступу: <https://card-file.ontu.edu.ua/handle/123456789/21689> (дата звернення: 16.11.2023).
17. Веренько А. Вибір архітектури програмної компоненти платформи для монетизації навчальних курсів [Електронний ресурс] / Артем Веренько // LI Науково-технічна конференція факультету ІТКІ. – 2022. – Режим доступу: <https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/16044> (дата звернення: 16.11.2023).
18. Wei J. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models [Electronic resource] / Jason Wei, Xuezhi Wang, Dale Schuurmans // Computer Science. Computation and Language. – 2022. – Mode of access: <https://arxiv.org/pdf/2201.11903.pdf> (date of access: 16.11.2023).
19. Yao S. Tree of Thoughts: Deliberate Problem Solving with Large Language Models [Electronic resource] / Shunyu Yao, Dian Yu, Jeffrey Zhao // Computer Science, Computation and Language. – 2023. – Mode of access: <https://arxiv.org/pdf/2305.10601.pdf> (date of access: 16.11.2023).
20. Quelle D. The Perils & Promises of Fact-checking with Large Language Models [Electronic resource] / Dorian Quelle // Computer Science. – 2023. – Mode of access: <https://arxiv.org/pdf/2310.13549.pdf> (date of access: 25.11.2023).
21. Веренько А. Система управління контентом для соціальних мереж з інтеграцією сучасних генеративних моделей штучного інтелекту [Електронний ресурс] / Артем Веренько // International scientific-practical conference science, education, technology and society: problems and prospects. – 2023. – С. 54–57. – Режим доступу: <http://www.economics.in.ua/2023/10/12-2023.html> (дата звернення: 16.11.2023).

ДОДАТКИ

Додаток А. UML діаграми

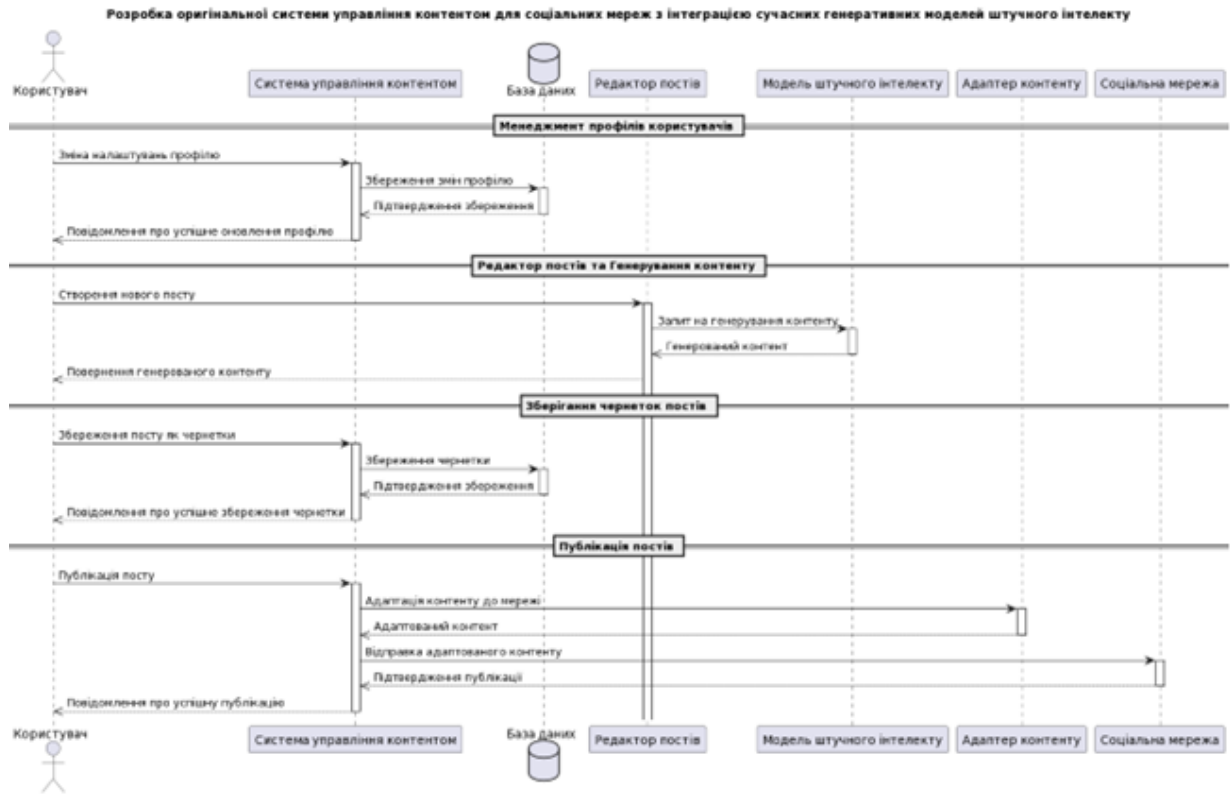


Рис. А. 1. Діаграма послідовності



Рис. А. 2. Діаграма варіантів використання системи для створення контенту

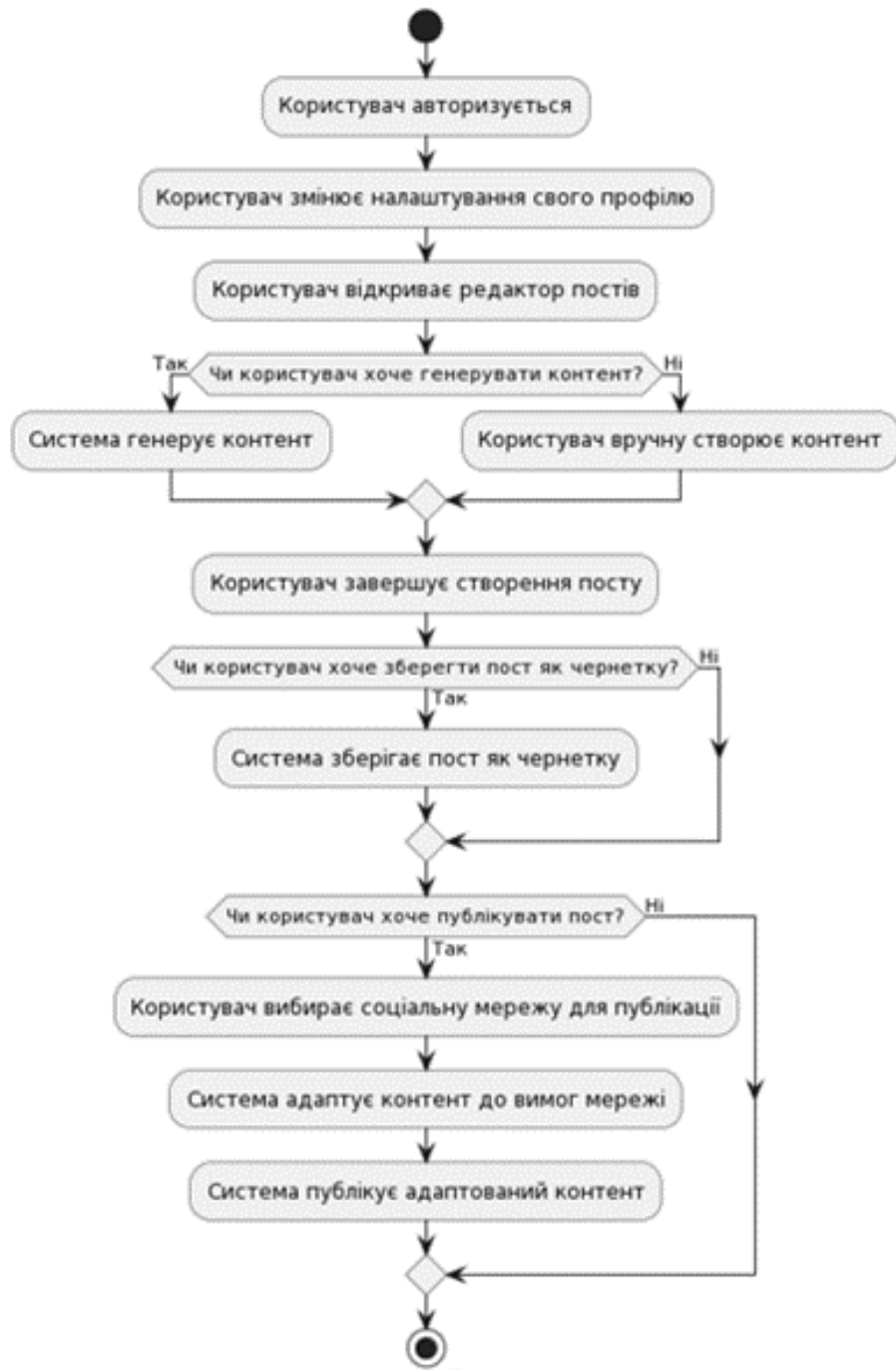


Рис. А. 3. Діаграма діяльності системи для створення контенту

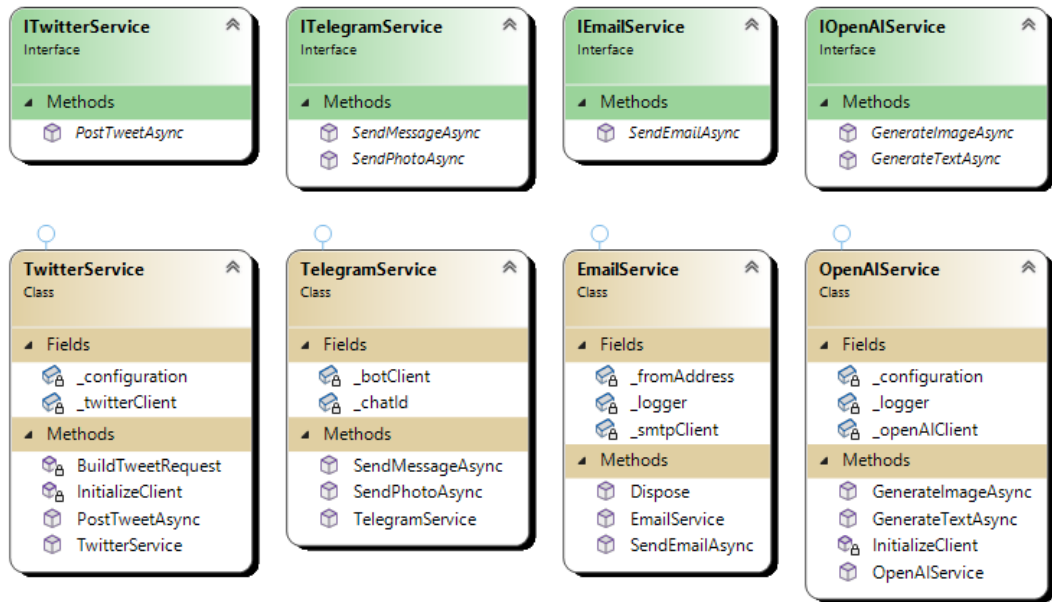


Рис. А. 4. Діаграма класів інтегрованих сторонніх сервісів

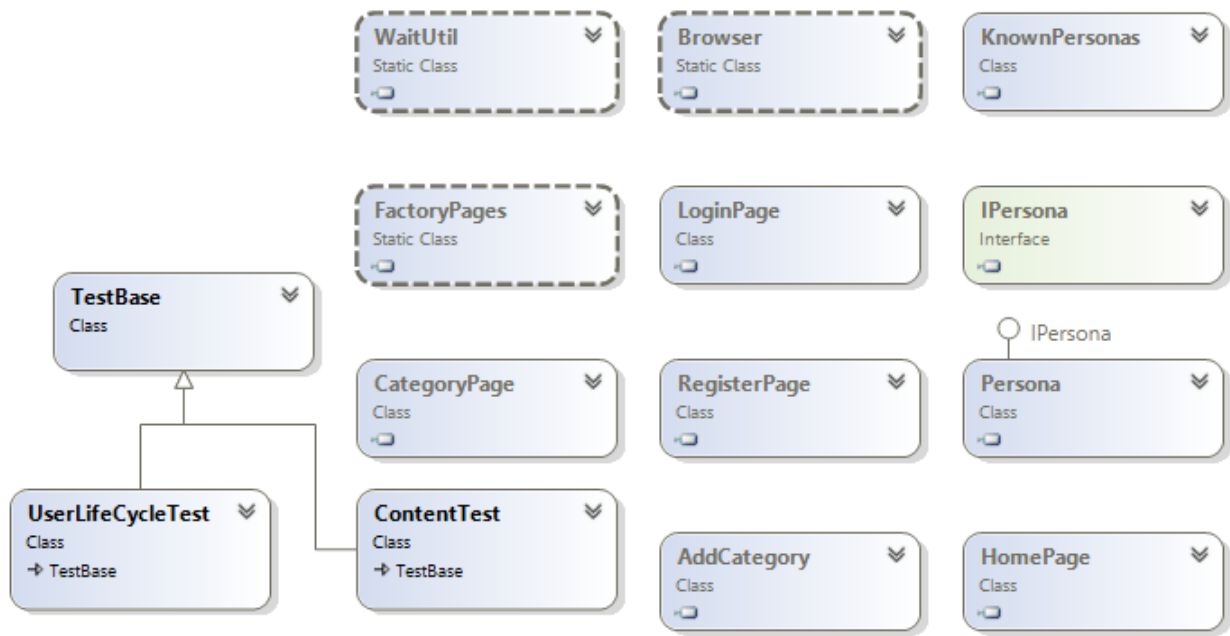


Рис. А. 5. Діаграма класів фреймворку тестування проекту

Додаток Б. PlantUML код UML діаграм

Лістинг діаграми послідовності:

```

1. @startuml
2. title Розробка оригінальної системи управління контентом для соціальних мереж з
   інтеграцією сучасних генеративних моделей штучного інтелекту
3.
4. actor Користувач as user
5. participant "Система управління контентом" as CMS
6. database "База даних" as DB
7. participant "Редактор постів" as Editor
8. participant "Модель штучного інтелекту" as AI
9. participant "Адаптер контенту" as Adapter
10. participant "Соціальна мережа" as SocialMedia
11.
12. == Менеджмент профілів користувачів ==
13. user -> CMS: Зміна налаштувань профілю
14. activate CMS
15. CMS -> DB: Збереження змін профілю
16. activate DB
17. DB -->> CMS: Підтвердження збереження
18. deactivate DB
19. CMS -->> user: Повідомлення про успішне оновлення профілю
20. deactivate CMS
21.
22. == Редактор постів та Генерування контенту ==
23. user -> Editor: Створення нового посту
24. activate Editor
25. Editor -> AI: Запит на генерування контенту
26. activate AI
27. AI -->> Editor: Генерований контент
28. deactivate AI
29. Editor -->> user: Повернення генерованого контенту
30.
31. == Зберігання чернеток постів ==
32. user -> CMS: Збереження посту як чернетки
33. activate CMS
34. CMS -> DB: Збереження чернетки
35. activate DB
36. DB -->> CMS: Підтвердження збереження
37. deactivate DB
38. CMS -->> user: Повідомлення про успішне збереження чернетки
39. deactivate CMS
40.
41. == Публікація постів ==
42. user -> CMS: Публікація посту
43. activate CMS
44. CMS -> Adapter: Адаптація контенту до мережі
45. activate Adapter
46. Adapter -->> CMS: Адаптований контент
47. deactivate Adapter
48. CMS -> SocialMedia: Відправка адаптованого контенту
49. activate SocialMedia
50. SocialMedia -->> CMS: Підтвердження публікації
51. deactivate SocialMedia
52. CMS -->> user: Повідомлення про успішну публікацію
53. deactivate CMS
54.
55. @enduml

```


Лістинг діаграми варіантів використання:

```

1. @startuml
2. title Діаграма варіантів використання для системи управління контентом
3. actor User as "Користувач"
4. actor AI as "AI моделі"
5. actor SocialNetworks as "Соціальні мережі"
6.
7. package "Управління профілем" {
8.     User -- (Керувати профілем)
9. }
10. package "Створення постів" {
11.     User -- (Відкрити редактор постів)
12.     User -- (Створити пост вручну)
13.
14.     User --> AI : надати промпт
15.     AI --> User : генерувати текст, фото, хештеги
16.     User -- (Зберегти пост як чернетку)
17. }
18. package "Публікація постів" {
19.     User -- (Обрати пост для публікації)
20.     User --> AI : надати пост
21.     AI --> SocialNetworks : адаптувати пост до стандартів
22.     SocialNetworks --> User : опублікувати адаптований пост
23. }
24. @enduml

```

Лістинг діаграми діяльності:

```

1. @startuml
2. title Розробка оригінальної системи управління контентом для соціальних мереж з
  інтеграцією сучасних генеративних моделей штучного інтелекту
3.
4. start
5. :Користувач авторизується;
6. :Користувач змінює налаштування свого профілю;
7. :Користувач відкриває редактор постів;
8. if (Чи користувач хоче генерувати контент?) then (Так)
9.     :Система генерує контент;
10. else (Hi)
11.     :Користувач вручну створює контент;
12. endif
13. :Користувач завершує створення посту;
14. if (Чи користувач хоче зберегти пост як чернетку?) then (Так)
15.     :Система зберігає пост як чернетку;
16. else (Hi)
17. endif
18. if (Чи користувач хоче публікувати пост?) then (Так)
19.     :Користувач вибирає соціальну мережу для публікації;
20.     :Система адаптує контент до вимог мережі;
21.     :Система публікує адаптований контент;
22. else (Hi)
23. endif
24.
25. stop
26. @enduml

```

Додаток В. Лістинг базового функціоналу веб-застосунку

Program.cs

```

1. using Autofac.Core;
2. using Diploma_DataAccess.Data;
3. using Diploma_DataAccess.Data.Repository;
4. using Diploma_DataAccess.Data.Repository.IRepository;
5. using Diploma_DataAccess.DTOS;
6. using DiplomaMaster.Services;
7. using Microsoft.AspNetCore.Identity;
8. using Microsoft.EntityFrameworkCore;
9. using Microsoft.OpenApi.Models;
10. using System.Configuration;
11.
12. var builder = WebApplication.CreateBuilder(args);
13. var configuration = new ConfigurationBuilder().AddJsonFile("appsettings.json").Build();
14. builder.Services.AddDbContext<ApplicationDbContext>(options =>
15. {
16.     options.UseSqlServer(configuration.GetConnectionString("DefaultConnection"));
17. });
18. builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>
19.     options.SignIn.RequireConfirmedAccount = false)
20.     .AddDefaultTokenProviders().AddDefaultUI().AddEntityFrameworkStores<ApplicationDbCo
21.     ntext>());
22. builder.Services.AddControllers();
23. builder.Services.AddEndpointsApiExplorer();
24. builder.Services.AddSwaggerGen(c =>
25. {
26.     c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });
27. });
28. builder.Services.AddHttpClient();
29. builder.Services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
30. builder.Services.AddDistributedMemoryCache();
31. builder.Services.Configure<TelegramSettings>(configuration.GetSection("TelegramSettings
32.     "));
33. builder.Services.AddSingleton<ITelegramService, TelegramService>();
34. builder.Services.AddSingleton<ITwitterService, TwitterService>();
35. builder.Services.AddSingleton<IOpenAIService, OpenAIService>();
36. builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
37. builder.Services.AddScoped<IPostRepository, PostRepository>();
38. builder.Services.AddScoped<IEmailService, EmailService>();
39. builder.Services.AddControllersWithViews();
40. var app = builder.Build();
41. if (!app.Environment.IsDevelopment())
42. {
43.     app.UseExceptionHandler("/Home/Error");
44.     app.UseHsts();
45. }
46. app.UseHttpsRedirection();
47. app.UseStaticFiles();
48. app.UseRouting();
49. app.UseAuthentication();
50. app.UseAuthorization();
51. app.UseSession();
52. app.MapControllerRoute(name:
53.     "default", pattern: "{controller=Home}/{action=Index}/{id?}");
54. app.MapRazorPages();
55. app.UseSwagger();
56. app.UseSwaggerUI(c =>
57. {
58.     c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
59. });
60. app.Run();

```

Post.cs

```

1. using System.ComponentModel.DataAnnotations.Schema;
2. using System.ComponentModel.DataAnnotations;
3. using System.Diagnostics.CodeAnalysis;
4.
5. namespace Diploma_Model.Models
6. {
7.     public class Post
8.     {
9.         [Key]
10.        public int Id { get; set; }
11.
12.        [Required]
13.        public string Name { get; set; }
14.
15.        [AllowNull]
16.        public string? ShortDesc { get; set; }
17.
18.        [AllowNull]
19.        public string? PostText { get; set; }
20.
21.        public string Image { get; set; }
22.        public DateTime PublishedDate { get; set; } = DateTime.Now;
23.        public string? Author { get; set; }
24.        public bool Visible { get; set; }
25.
26.        [Display(Name = "Category Type")]
27.        public int? CategoryId { get; set; }
28.
29.        [ForeignKey("CategoryId")]
30.        public virtual Category Category { get; set; }
31.    }
32. }

```

Category.cs

```

1. using System.ComponentModel.DataAnnotations;
2. using System.ComponentModel;
3.
4. namespace Diploma_Model.Models
5. {
6.     public class Category
7.     {
8.         [Key]
9.        public int Id { get; set; }
10.
11.        [Required]
12.        public string Name { get; set; }
13.
14.        [DisplayName("Display Order")]
15.        [Required]
16.        [Range(1, int.MaxValue, ErrorMessage = "Display Order for category must be greater than 0")]
17.        public int DisplayOrder { get; set; }
18.    }
19. }

```

PostVM.cs

```

1. using Microsoft.AspNetCore.Mvc.Rendering;
2. using System.Diagnostics.CodeAnalysis;
3.
4. namespace Diploma_Model.Models.ViewModels
5. {
6.     public class PostVM
7.     {
8.         [AllowNull]
9.         public Post? Post { get; set; }
10.
11.         [AllowNull]
12.         public IEnumerable<SelectListItem>? CategorySelectList { get; set; }
13.
14.         [AllowNull]
15.         public IEnumerable<SelectListItem>? ApplicationTypeSelectList { get; set; }
16.     }
17. }

```

CategoryController.cs

```

1. using Diploma_Utility;
2. using Diploma_Utility;
3. using Diploma_DataAccess.Data;
4. using Diploma_Model.Models;
5. using Microsoft.AspNetCore.Authorization;
6. using Microsoft.AspNetCore.Mvc;
7. using Diploma_DataAccess.Data.Repository.IRepository;
8.
9. namespace DiplomaMaster.Controllers
10. {
11.     [Authorize(Roles = WC.AdminRole)]
12.     public class CategoryController : Controller
13.     {
14.         private readonly ICategoryRepository _catRepo;
15.
16.         public CategoryController(ICategoryRepository cetRepo)
17.         {
18.             _catRepo = cetRepo;
19.         }
20.
21.         [HttpGet]
22.         public async Task<IActionResult> Index()
23.         {
24.             IEnumerable<Category> objList = await _catRepo.GetAllAsync();
25.             objList = objList.OrderBy(c => c.DisplayOrder);
26.             return View(objList);
27.         }
28.
29.         [HttpGet]
30.         public IActionResult Create()
31.         {
32.             return View();
33.         }
34.
35.         [HttpPost]
36.         [ValidateAntiForgeryToken]
37.         public async Task<IActionResult> Create(Category obj)
38.         {
39.             if (ModelState.IsValid)
40.             {
41.                 _catRepo.Add(obj);
42.                 await _catRepo.SaveAsync();

```

```

43.         return RedirectToAction("Index");
44.     }
45.     return View(obj);
46. }
47.
48. [HttpGet]
49. public async Task<IActionResult> Edit(int? id)
50. {
51.     if (id == null || id == 0)
52.     {
53.         return NotFound();
54.     }
55.     var obj = await _catRepo.FindAsync(id.GetValueOrDefault());
56.     if (obj == null)
57.     {
58.         return NotFound();
59.     }
60.     return View(obj);
61. }
62.
63. [HttpPost]
64. [ValidateAntiForgeryToken]
65. public async Task<IActionResult> Edit(Category obj)
66. {
67.     if (ModelState.IsValid)
68.     {
69.         await _catRepo.UpdateAsync(obj);
70.         await _catRepo.SaveAsync();
71.         return RedirectToAction("Index");
72.     }
73.     return View(obj);
74. }
75.
76. [HttpGet]
77. public async Task<IActionResult> Delete(int? id)
78. {
79.     if (id == null || id == 0)
80.     {
81.         return NotFound();
82.     }
83.     var obj = await _catRepo.FindAsync(id.GetValueOrDefault());
84.     if (obj == null)
85.     {
86.         return NotFound();
87.     }
88.     return View(obj);
89. }
90.
91. [HttpPost]
92. [ValidateAntiForgeryToken]
93. public async Task<IActionResult> DeletePost(int? id)
94. {
95.     var obj = await _catRepo.FindAsync(id.GetValueOrDefault());
96.     if (obj == null)
97.     {
98.         return NotFound();
99.     }
100.     _catRepo.Remove(obj);
101.     await _catRepo.SaveAsync();
102.     return RedirectToAction("Index");
103. }
104. }
105. }

```

PostController.cs

```

1. using Diploma_DataAccess.Data.Repository.IRepository;
2. using Diploma_Model.Models;
3. using Diploma_Model.Models.ViewModels;
4. using Diploma_Utility;
5. using DiplomaMaster.Services;
6. using Microsoft.AspNetCore.Authorization;
7. using Microsoft.AspNetCore.Mvc;
8.
9. namespace DiplomaMaster.Controllers
10. {
11.     [Authorize(Roles = WC.AdminRole)]
12.     public class PostController : Controller
13.     {
14.         private readonly IPostRepository _postRepo;
15.         private readonly IWebHostEnvironment _webHostEnvironment;
16.         private readonly IOpenAIService _openAIService;
17.
18.         public PostController(IPostRepository prodRepo, IWebHostEnvironment
webHostEnvironment, IOpenAIService openAIService)
19.         {
20.             _postRepo = prodRepo;
21.             _webHostEnvironment = webHostEnvironment;
22.             _openAIService = openAIService;
23.         }
24.
25.         [HttpGet]
26.         public async Task<IActionResult> Index()
27.         {
28.             IEnumerable<Post> objList = await _postRepo.GetAllAsync(includeProperties:
"Category");
29.             return View(objList);
30.         }
31.
32.
33.         [HttpGet]
34.         public async Task<IActionResult> Upsert(int? id)
35.         {
36.             PostVM postVM = new PostVM()
37.             {
38.                 Post = new Post(),
39.                 CategorySelectList = _postRepo.GetAllDropDownList(WC.CategoryName),
40.             };
41.
42.             if (id == null)
43.             {
44.                 //this is for create
45.                 return View(postVM);
46.             }
47.             else
48.             {
49.                 postVM.Post = await _postRepo.FindAsync(id.GetValueOrDefault());
50.                 if (postVM.Post == null)
51.                 {
52.                     return NotFound();
53.                 }
54.                 return View(postVM);
55.             }
56.         }
57.
58.         [HttpPost]
59.         [ValidateAntiForgeryToken]
60.         public async Task<IActionResult> Upsert(PostVM postVM, string imageSource,
string imageCaption)
61.         {

```

```

62.         if (ModelState.IsValid)
63.         {
64.             var files = HttpContext.Request.Form.Files;
65.             string webRootPath = _webHostEnvironment.WebRootPath;
66.             string extension;
67.             string upload = webRootPath + WC.ImagePath;
68.             string fileName = Guid.NewGuid().ToString();
69.
70.             if (postVM.Post.Id == 0)
71.             {
72.                 //Creating
73.                 if (imageSource == "local")
74.                 {
75.                     extension = Path.GetExtension(files[0].FileName);
76.
77.                     using (var fileStream = new FileStream(Path.Combine(upload,
file
fileName + extension), FileMode.Create))
78.                     {
79.                         files[0].CopyTo(fileStream);
80.                     }
81.                 }
82.                 else if (imageSource == "generate")
83.                 {
84.                     extension = ".png";
85.
86.                     var result = await
_openAIService.GenerateImageAsync(imageCaption);
87.
88.                     string generatedImageUrl = result.IsSuccess ? result.ImageUrl :
"An error occurred";
89.
90.                     using (HttpClient client = new HttpClient())
91.                     {
92.                         var response = await client.GetAsync(generatedImageUrl);
93.                         response.EnsureSuccessStatusCode();
94.
95.                         byte[] imageBytes = await
response.Content.ReadAsByteArrayAsync();
96.
97.                         string savePath = Path.Combine(upload, fileName +
extension);
98.
99.                         await System.IO.File.WriteAllBytesAsync(savePath,
imageBytes);
100.
101.                     }
102.                     postVM.Post.Image = fileName + extension;
103.                 }
104.                 else
105.                 {
106.                     TempData[WC.Error] = "Invalid image source selected.";
107.                     postVM.CategorySelectList =
_postRepo.GetAllDropDownList(WC.CategoryName);
108.                     return View(postVM);
109.                 }
110.
111.                 postVM.Post.Image = fileName + extension;
112.
113.                 postVM.Post.PublishedDate = DateTime.Now;
114.
115.
116.                 postVM.Post.Author = User.Identity.Name;
117.
118.                 _postRepo.Add(postVM.Post);
119.             }
120.             else
121.             {

```

```

        var objFromDb = await _postRepo.FirstOrDefaultAsync(u => u.Id ==
postVM.Post.Id, isTracking: false);

122.         var oldFile = Path.Combine(upload, objFromDb.Image);
123.
124.         if (imageSource == "local" && files.Count > 0)
125.         {
126.             extension = Path.GetExtension(files[0].FileName);
127.
128.             if (System.IO.File.Exists(oldFile))
129.             {
130.                 System.IO.File.Delete(oldFile);
131.             }
132.
133.             using (var fileStream = new
FileStream(Path.Combine(upload, fileName + extension), FileMode.Create))
134.             {
135.                 files[0].CopyTo(fileStream);
136.             }
137.             postVM.Post.Image = fileName + extension;
138.         }
139.         else if (imageSource == "generate")
140.         {
141.             extension = ".png";
142.
143.             if (System.IO.File.Exists(oldFile))
144.             {
145.                 System.IO.File.Delete(oldFile);
146.             }
147.
148.             var result = await
_openAIService.GenerateImageAsync(imageCaption);
149.
150.             string generatedImageUrl = result.IsSuccess ?
result.ImageUrl : "An error occurred";
151.
152.             using (HttpClient client = new HttpClient())
153.             {
154.                 var response = await
client.GetAsync(generatedImageUrl);
155.                 response.EnsureSuccessStatusCode();
156.
157.                 byte[] imageBytes = await
response.Content.ReadAsByteArrayAsync();
158.
159.                 string savePath = Path.Combine(upload, fileName +
extension);
160.
161.                 await System.IO.File.WriteAllBytesAsync(savePath,
imageBytes);
162.
163.             }
164.             postVM.Post.Image = fileName + extension;
165.         }
166.         else
167.         {
168.             postVM.Post.Image = objFromDb.Image;
169.         }
170.
171.         postVM.Post.PublishedDate = DateTime.Now;
172.
173.         postVM.Post.Author = User.Identity.Name;
174.
175.         _postRepo.Update(postVM.Post);
176.     }
177.
178.     await _postRepo.SaveAsync();

```



```

179.         return RedirectToAction("Index");
180.     }
181.
182.     postVM.CategorySelectList =
183.     _postRepo.GetAllDropDownList(WC.CategoryName);
184.     return View(postVM);
185. }
186. [HttpGet]
187. public async Task<IActionResult> Delete(int? id)
188. {
189.     if (id == null || id == 0)
190.     {
191.         return NotFound();
192.     }
193.     Post product = await _postRepo.FirstOrDefaultAsync(u => u.Id == id,
includeProperties: "Category");
194.     if (product == null)
195.     {
196.         return NotFound();
197.     }
198.     return View(product);
199. }
200.
201. [HttpPost, ActionName("Delete")]
202. [ValidateAntiForgeryToken]
203. public async Task<IActionResult> DeletePost(int? id)
204. {
205.     var obj = await _postRepo.FindAsync(id.GetValueOrDefault());
206.     if (obj == null)
207.     {
208.         return NotFound();
209.     }
210.
211.     string upload = _webHostEnvironment.WebRootPath + WC.ImagePath;
212.     var oldFile = Path.Combine(upload, obj.Image);
213.
214.     if (System.IO.File.Exists(oldFile))
215.     {
216.         System.IO.File.Delete(oldFile);
217.     }
218.     _postRepo.Remove(obj);
219.     await _postRepo.SaveAsync();
220.     TempData[WC.Success] = "Action completed successfully";
221.     return RedirectToAction("Index");
222. }
223. }
224. }

```

HomeController.cs

```

1. using Diploma_DataAccess.Data;
2. using Diploma_Model.Models;
3. using Diploma_Model.Models.ViewModels;
4. using Microsoft.AspNetCore.Mvc;
5. using Microsoft.EntityFrameworkCore;
6. using System.Diagnostics;
7.
8. namespace DiplomaMaster.Controllers
9. {
10.     public class HomeController : Controller
11.     {
12.         private readonly ILogger<HomeController> _logger;
13.         public readonly ApplicationDbContext _db;
14.
15.         public HomeController(ILogger<HomeController> logger, ApplicationDbContext db)

```

```

16.     {
17.         _logger = logger;
18.         _db = db;
19.     }
20.
21.     [HttpGet]
22.     public IActionResult Index()
23.     {
24.         ViewBag.TweetMessage = TempData["TweetMessage"]?.ToString();
25.         HomeVM homeVM = new HomeVM()
26.         {
27.             Posts = _db.Post.Where(p => p.Visible).Include(u => u.Category),
28.             Categories = _db.Category.OrderBy(c => c.DisplayOrder)
29.         };
30.         return View(homeVM);
31.     }
32.
33.     [HttpGet]
34.     public IActionResult Details(int id)
35.     {
36.
37.         DetailsVM DetailsVM = new DetailsVM()
38.         {
39.             Post = _db.Post.Include(u => u.Category).Where(u => u.Id ==
id).FirstOrDefault()
40.         };
41.
42.         return View(DetailsVM);
43.     }
44.
45.     [HttpGet]
46.     public IActionResult TwitterAnnounce(int id)
47.     {
48.         return RedirectToAction("Index", "Twitter", new { id = id });
49.     }
50.
51.     [HttpGet]
52.     public IActionResult TelegramAnnounce(int id)
53.     {
54.         return RedirectToAction("Index", "Telegram", new { id = id });
55.     }
56.
57.     [HttpGet]
58.     public IActionResult GmailAnnounce(int id)
59.     {
60.         return RedirectToAction("Index", "Email", new { id = id });
61.     }
62.
63.     [HttpGet]
64.     [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
65.     public IActionResult Error()
66.     {
67.         return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
68.     }
69. }
70. }

```

OpenAIController.cs

```

1. using DiplomaMaster.Services;
2. using Microsoft.AspNetCore.Mvc;
3.
4. namespace DiplomaMaster.Controllers
5. {

```

```

6.     public class OpenAIController: Controller
7.     {
8.         private readonly IOpenAIService _openAIService;
9.         public OpenAIController(IOpenAIService openAIService)
10.        {
11.            _openAIService = openAIService;
12.        }
13.
14.        [HttpGet]
15.        public IActionResult OpenAI_GPT_4()
16.        {
17.            return View("OpenAI_GPT_4");
18.        }
19.
20.        [HttpPost]
21.        public async Task<IActionResult> OpenAI_GPT_4(string prompt)
22.        {
23.            var result = await _openAIService.GenerateTextAsync(prompt);
24.            ViewBag.Prompt = prompt;
25.            ViewBag.Result = result.IsSuccess ? result.Content : "An error occurred";
26.            return View("OpenAI_GPT_4");
27.        }
28.
29.        [HttpGet]
30.        public IActionResult DALLE2()
31.        {
32.            return View("DALLE2");
33.        }
34.
35.        [HttpPost]
36.        public async Task<IActionResult> DALLE2(string prompt)
37.        {
38.            var result = await _openAIService.GenerateImageAsync(prompt);
39.            ViewBag.Prompt = prompt;
40.            ViewBag.Result = result.IsSuccess ? result.ImageUrl : "An error occurred";
41.            return View("DALLE2");
42.        }
43.
44.    }
45. }

```

TwitterAnnounceVM.cs

```

1. namespace Diploma_Model.Models.ViewModels
2. {
3.     public class TelegramAnnounceVM
4.     {
5.         public TelegramAnnounceVM ()
6.         {
7.         }
8.
9.         public TelegramAnnounceVM(Post post)
10.        {
11.            Post = post ?? new Post();
12.            DefaultPrompt = $"Prepare a full post for publication on Telegram using the
following          details:\r\n\r\nPlatform:          DiplomaMaster\r\n\r\nPost          Title:
{Post.Name}\r\n\r\nDescription:          {Post.ShortDesc}\r\n\r\nFull          Text:
{Post.PostText}\r\n\r\n\r\nPlease ensure the post is formatted for clear readability and
includes all necessary details for a comprehensive update to the audience. Include
relevant hashtags and format it appropriately for the Telegram platform.";
13.        }
14.
15.        public Post Post { get; set; }
16.
17.        public string DefaultPrompt { get; set; }
18.    }

```

```

19.         public string? Prompt { get; set; }
20.     }
21. }

```

DALLE2.cshtml

```

1. @{
2.     ViewData["Title"] = "DALLE2";
3. }
4.
5. <div class="container mt-5">
6.     <div class="row justify-content-center">
7.         <div class="col-md-6 p-4" style="background-color: #f8f9fa; border-radius:
10px; box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);">
8.             <h2 class="text-center mb-4">DALL·E 2</h2>
9.             <form method="post" action="@Url.Action("DALLE2")">
10.                 <div class="form-group">
11.                     <label class="pb-2" for="prompt" style="font-
weight:bold;">Prompt</label>
12.                     <textarea class="form-control rounded py-2" id="prompt"
name="prompt" placeholder="Enter your prompt" rows="4" style="border: 1px solid
#ced4da;">@ViewBag.Prompt</textarea>
13.                 </div>
14.                 <div class="pt-2">
15.                     <button type="submit" class="btn btn-primary" style="background-
color: #5c6bc0; border: none;">Generate image</button>
16.                 </div>
17.             </form>
18.
19.             @if (ViewBag.Result != null)
20.             {
21.                 <div class="mt-4 pb-2">
22.                     <p style="font-weight: 600;">Generated Image:</p>
23.                     
24.                 </div>
25.
26.                 <a href="@ViewBag.Result" target="_blank"
download="generated_image.png" class="btn btn-secondary mt-2">Save Image</a>
27.             }
28.         </div>
29.     </div>
30. </div>

```

OpenAI_GPT_4.cshtml

```

1. @{
2.     ViewData["Title"] = "GetGPTResponse";
3. }
4.
5. <div class="container mt-5">
6.     <div class="row justify-content-center">
7.         <div class="col-md-6 p-4" style="background-color: #f8f9fa; border-radius:
10px; box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);">
8.             <h2 class="text-center mb-4">GPT-4</h2>
9.             <form method="post" action="@Url.Action("OpenAI_GPT_4")">
10.                 <div class="form-group">
11.                     <label class="pb-2" for="prompt" style="font-
weight:bold;">Prompt</label>
12.                     @* <textarea class="form-control rounded py-2" id="prompt"
name="prompt" placeholder="Enter your prompt" rows="4" style="border: 1px solid
#ced4da;"></textarea> *@

```

```

13.         <textarea class="form-control rounded py-2" id="prompt"
name="prompt" placeholder="Enter your prompt" rows="4" style="border: 1px solid
#ced4da;">@ViewBag.Prompt</textarea>
14.     </div>
15.     <div class="pt-2">
16.         <button type="submit" class="btn btn-primary" style="background-
color: #5c6bc0; border: none;">Submit</button>
17.     </div>
18. </form>
19.
20. @if (ViewBag.Result != null)
21. {
22.     <div class="mt-4 pb-2">
23.         <p style="font-weight: 600;">Result:</p>
24.         <div class="alert alert-info" style="font-size:
16px;">@Html.Raw(ViewBag.Result)</div>
25.     </div>
26. }
27. </div>
28. </div>
29. </div>

```

Home Index.cshtml

```

1. @model Diploma_Model.Models.ViewModels.HomeVM
2.
3. @if (!string.IsNullOrEmpty(ViewBag.TweetMessage))
4. {
5.     <script>
6.         window.onload = function () {
7.             alert('@ViewBag.TweetMessage');
8.         };
9.     </script>
10. }
11.
12. <div class="container">
13.     <div class="text-center">
14.         <button class="btn btn-dark filter-button" data-filter="all">All</button>
15.         @foreach (var obj in Model.Categories)
16.         {
17.             <button class="btn btn-outline-secondary filter-button" data-
filter="@obj.Name.Replace(' ', '_')">
18.                 @obj.Name
19.             </button>
20.         }
21.     </div>
22.     <br />
23.
24.     <div class="row">
25.         @foreach (var prod in Model.Posts)
26.         {
27.             <partial name="_IndividualPostCard" model="prod" />
28.         }
29.     </div>
30.     <div class="container">
31.         
32.     </div>
33. </div>
34.
35.
36.
37. @section Scripts{
38.     <script>
39.         $(document).ready(function () {
40.
41.             $(".filter-button").click(function () {

```



```

41.         <button asp-action="TwitterPostGenerate"
class="btn btn-primary shadow-sm border" style="background-color: #1DA1F2; border:
none;">Generate an announcement</button>
42.     </div>
43. </div>
44. @if (ViewBag.Result != null)
45. {
46.     <div class="mt-4 pb-2">
47.         <p style="font-weight: 600;">Result:</p>
48.         <div class="alert alert-info" style="font-
size: 16px;">@Html.Raw(ViewBag.Result)</div>
49.     </div>
50. }
51. </div>
52. </div>
53. </div>
54. </div>
55. </div>
56. <div class="card-footer bg-dark">
57.     <div class="row">
58.         <div class="col-12 col-md-6 pb-1 ">
59.             <a asp-action="Index" class="btn btn-success btn-square
form-control btn-lg" style="height:50px;">Back To List</a>
60.         </div><div class="col-12 col-md-6 pb-1 ">
61.             <button asp-action="TwitterPostAnnounce" class="btn btn-
info btn-square form-control btn-lg" style="height:50px;">Announce It</button>
62.         </div>
63.     </div>
64. </div>
65. </div>
66. </div>
67. </div>
68. </form>
69. </div>

```

Додаток Д. Лістинг інтеграції сервісів

EmailService.cs

```

1. using Diploma_DataAccess.DTOS;
2. using DiplomaMaster.Services.Models;
3. using System.Net;
4. using System.Net.Mail;
5.
6. namespace DiplomaMaster.Services
7. {
8.     public class EmailService: IEmailService, IDisposable
9.     {
10.         private readonly SmtpClient _smtpClient;
11.         private readonly string _fromAddress;
12.         private readonly ILogger<EmailService> _logger;
13.
14.         public EmailService(IConfiguration configuration, ILogger<EmailService> logger)
15.         {
16.             _logger = logger;
17.             _smtpClient = new SmtpClient(configuration["EmailSettings:Host"])
18.             {
19.                 Port = int.Parse(configuration["EmailSettings:Port"]),
20.                 Credentials = new NetworkCredential(
21.                     configuration["EmailSettings:Username"],
22.                     configuration["EmailSettings:Password"]
23.                 ),
24.                 EnableSsl = bool.Parse(configuration["EmailSettings:EnableSsl"])
25.             };
26.             _fromAddress = configuration["EmailSettings:FromAddress"];
27.         }
28.         public async Task<GmailResult> SendEmailAsync(EmailMessage emailMessage)
29.         {
30.             try
31.             {
32.                 using (var mailMessage = new MailMessage
33.                 {
34.                     From = new MailAddress(_fromAddress),
35.                     Subject = emailMessage.Subject,
36.                     Body = emailMessage.Body,
37.                     IsBodyHtml = true
38.                 })
39.                 {
40.                     mailMessage.To.Add(emailMessage.To);
41.                     await _smtpClient.SendMailAsync(mailMessage);
42.                 }
43.                 return new GmailResult
44.                 {
45.                     IsSuccess = true,
46.                     Message = "Success: Message was sent successfully."
47.                 };
48.             }
49.             catch (Exception ex) {
50.                 _logger.LogError(ex, "Error occurred while sending email.");
51.                 return new GmailResult
52.                 {
53.                     IsSuccess = false,
54.                     Message = $"Error: Failed to send message"
55.                 };
56.             }
57.         }
58.         public void Dispose()
59.         {
60.             _smtpClient?.Dispose();
61.         }
62.     }
63. }

```


OpenAIService.cs

```

1. using Diploma_DataAccess.DTOS;
2. using DiplomaMaster.Services.Models;
3. using Newtonsoft.Json;
4. using System.Text;
5.
6. namespace DiplomaMaster.Services
7. {
8.     public class OpenAIService : IOpenAIService
9.     {
10.         private readonly HttpClient _openAIClient;
11.         private readonly IConfiguration _configuration;
12.         private readonly ILogger<OpenAIService> _logger;
13.
14.         public OpenAIService(IConfiguration configuration, HttpClient openAIClient,
15.             ILogger<OpenAIService> logger)
16.         {
17.             _configuration = configuration;
18.             _openAIClient = openAIClient;
19.             InitializeClient();
20.             _logger = logger;
21.         }
22.         private void InitializeClient()
23.         {
24.             var openAPIKey = _configuration["OpenAI:Key"];
25.             _openAIClient.DefaultRequestHeaders.Add("Authorization", $"Bearer
26. {openAPIKey}");
27.         }
28.         public async Task<TextGenerationResult> GenerateTextAsync(string prompt, int
29.             _max_tokens = 2000)
30.         {
31.             OpenAIResponseGPT4 response = null;
32.             var payload = new
33.             {
34.                 model = "gpt-3.5-turbo-16k",
35.                 messages = new object[]
36.                 {
37.                     new { role = "system", content = $"Hi"},
38.                     new { role = "user", content = prompt}
39.                 },
40.                 temperature = 0.3,
41.                 max_tokens = _max_tokens
42.             };
43.             string jsonPayload = JsonConvert.SerializeObject(payload);
44.             HttpContent httpContent = new StringContent(jsonPayload, Encoding.UTF8,
45.                 "application/json");
46.             //Send the request
47.             var responseMessage = await
48.                 _openAIClient.PostAsync("https://api.openai.com/v1/chat/completions", httpContent);
49.             if (responseMessage.IsSuccessStatusCode)
50.             {
51.                 var responseMessageJson = await
52.                     responseMessage.Content.ReadAsStringAsync();
53.                 //Return a response
54.                 response =
55.                     JsonConvert.DeserializeObject<OpenAIResponseGPT4>(responseMessageJson);
56.             }
57.             return new TextGenerationResult
58.             {

```

```

58.         IsSuccess = responseMessage.IsSuccessStatusCode,
59.         Content = response.Choices[0].Message.Content
60.     };
61. }
62.
63.     public async Task<ImageGenerationResult> GenerateImageAsync(string prompt)
64.     {
65.         OpenAIResponseDAALE2 response = null;
66.         var payload = new
67.         {
68.             model = "dall-e-3",
69.             size = "1024x1024",
70.             prompt = prompt,
71.             quality = "standard",
72.             n = 1,
73.         };
74.         string jsonPayload = JsonConvert.SerializeObject(payload);
75.         HttpContent httpContent = new StringContent(jsonPayload, Encoding.UTF8,
"application/json");
76.
77.         //Send the request
78.         var responseMessage = await
_openAIClient.PostAsync("https://api.openai.com/v1/images/generations", httpContent);
79.         if (responseMessage.IsSuccessStatusCode)
80.         {
81.             var responseMessageJson = await
responseMessage.Content.ReadAsStringAsync();
82.
83.             //Return a response
84.             response = JsonConvert.DeserializeObject<OpenAIResponseDAALE2>(responseMessageJson);
85.         }
86.
87.         return new ImageGenerationResult
88.         {
89.             IsSuccess = responseMessage.IsSuccessStatusCode,
90.             ImageUrl = response.Data[0].Url,
91.             Prompt = prompt
92.         };
93.     }
94. }
95. }

```

TelegramService.cs

```

1. using Diploma_DataAccess.DTOS;
2. using DiplomaMaster.Services.Models;
3. using Microsoft.Extensions.Options;
4. using Telegram.Bot;
5. using Telegram.Bot.Exceptions;
6. using Telegram.Bot.Types;
7.
8. namespace DiplomaMaster.Services
9. {
10.     public class TelegramService : ITelegramService
11.     {
12.         private readonly TelegramBotClient _botClient;
13.         private readonly string _chatId;
14.
15.         public TelegramService(IOptions<TelegramSettings> settings)
16.         {
17.             _botClient = new TelegramBotClient(settings.Value.BotToken);
18.             _chatId = settings.Value.CHAT_ID;
19.         }
20.
21.         public async Task<TelegramResult> SendMessageAsync(string message)

```

```

22.     {
23.         try
24.         {
25.             var result = await _botClient.SendTextMessageAsync(_chatId, message);
26.             return new TelegramResult
27.             {
28.                 IsSuccess = true,
29.                 Message = "Success: Message was sent successfully."
30.             };
31.         }
32.         catch (ApiRequestException apiEx)
33.         {
34.             return new TelegramResult
35.             {
36.                 IsSuccess = false,
37.                 Message = $"Error: Failed to send message - {apiEx.Message}"
38.             };
39.         }
40.     }
41.
42.     public async Task<TelegramResult> SendPhotoAsync(Stream photoStream, string
caption = null)
43.     {
44.         try
45.         {
46.             var inputPhoto = new InputFileStream(photoStream);
47.             var result = await _botClient.SendPhotoAsync(_chatId, inputPhoto, null,
caption);
48.             return new TelegramResult
49.             {
50.                 IsSuccess = true,
51.                 Message = "Success: Photo was sent successfully."
52.             };
53.         }
54.         catch (ApiRequestException apiEx)
55.         {
56.             return new TelegramResult
57.             {
58.                 IsSuccess = false,
59.                 Message = $"Error: Failed to send photo - {apiEx.Message}"
60.             };
61.         }
62.     }
63. }
64. }

```

TwitterService.cs

```

1. using Diploma_DataAccess.DTOS;
2. using DiplomaMaster.Services.Models;
3. using System.Text;
4. using Tweetinvi;
5. using Tweetinvi.Models;
6.
7. namespace DiplomaMaster.Services
8. {
9.     public class TwitterService : ITwitterService
10.    {
11.        private readonly IConfiguration _configuration;
12.        private TwitterClient _twitterClient;
13.
14.        public TwitterService(IConfiguration configuration)
15.        {
16.            _configuration = configuration;
17.            InitializeClient();
18.        }

```

```

19.     private void InitializeClient()
20.     {
21.         var consumerKey =
22.         _configuration.GetValue<string>("TwitterAPI:ConsumerKey");
23.         var consumerSecret =
24.         _configuration.GetValue<string>("TwitterAPI:ConsumerSecret");
25.         var accessToken =
26.         _configuration.GetValue<string>("TwitterAPI:AccessToken");
27.         var accessSecret =
28.         _configuration.GetValue<string>("TwitterAPI:AccessSecret");
29.         _twitterClient = new TwitterClient(consumerKey, consumerSecret,
30.         accessToken, accessSecret);
31.     }
32.
33.     public async Task<TweetResult> PostTweetAsync(string message)
34.     {
35.         var tweetRequest = BuildTweetRequest(new TwitterDTO { Text = message },
36.         _twitterClient);
37.         var result = await
38.         _twitterClient.Execute.AdvanceRequestAsync(tweetRequest);
39.
40.         return new TweetResult
41.         {
42.             IsSuccess = result.Response.IsSuccessStatusCode,
43.             Message = result.Response.IsSuccessStatusCode ? "Success: Tweet was
44.             sent successfully." : $"Error: Failed to send tweet - {result.Response.Content}"
45.         };
46.     }
47.
48.     private static Action<ITwitterRequest> BuildTweetRequest(
49.         TwitterDTO newTweet,
50.         TwitterClient userClient)
51.     {
52.         return (ITwitterRequest request) =>
53.         {
54.             var jsonBody = userClient.Json.Serialize(newTweet);
55.             var content = new StringContent(jsonBody, Encoding.UTF8,
56.             "application/json");
57.             request.Query.Url = "https://api.twitter.com/2/tweets";
58.             request.Query.HttpMethod = Tweetinvi.Models.HttpMethod.POST;
59.             request.Query.HttpContent = content;
60.         };
61.     }
62. }

```

Додаток Е. Лістинг фреймворку автоматизованого тестування

FactoryPages.cs

```

1. using Diploma.Automation.Suport;
2. using SeleniumExtras.PageObjects;
3.
4. namespace Diploma.Automation.Pages
5. {
6.     public static class FactoryPages
7.     {
8.         public static HomePage HomePage
9.         {
10.            get
11.            {
12.                var homePage = new HomePage();
13.                PageFactory.InitElements(Browser.Driver, homePage);
14.                return homePage;
15.            }
16.        }
17.        public static LoginPage LoginPage
18.        {
19.            get
20.            {
21.                var loginPage = new LoginPage();
22.                PageFactory.InitElements(Browser.Driver, loginPage);
23.                return loginPage;
24.            }
25.        }
26.
27.        public static RegisterPage RegisterPage
28.        {
29.            get
30.            {
31.                var registerPage = new RegisterPage();
32.                PageFactory.InitElements(Browser.Driver, registerPage);
33.                return registerPage;
34.            }
35.        }
36.
37.        public static CategoryPage CategoryPage
38.        {
39.            get
40.            {
41.                var categoryPage = new CategoryPage();
42.                PageFactory.InitElements(Browser.Driver, categoryPage);
43.                return categoryPage;
44.            }
45.        }
46.    }
47. }
48. }
```

Browser.cs

```

1. using OpenQA.Selenium;
2. using OpenQA.Selenium.Chrome;
3. using OpenQA.Selenium.Support.UI;
4. using System;
5.
6. namespace Diploma.Automation.Suport
7. {
8.     public static class Browser
```

```

9.      {
10.         private static IWebDriver webDriver;
11.         public static WebDriverWait wait;
12.         public static string BASEURL = "https://localhost:44390/";
13.
14.         public static void Init()
15.         {
16.             webDriver = new ChromeDriver(@"C:\CD");
17.             wait = new WebDriverWait(webDriver, TimeSpan.FromSeconds(10));
18.         }
19.
20.         public static string Title
21.         {
22.             get { return webDriver.Title; }
23.         }
24.
25.         public static IWebDriver Driver
26.         {
27.             get { return webDriver; }
28.         }
29.
30.         public static void Goto(string url)
31.         {
32.             webDriver.Url = url;
33.         }
34.
35.         public static void Close()
36.         {
37.             webDriver.Quit();
38.         }
39.
40.         public static string FullURL(string pageUrl)
41.         {
42.             return BASEURL + pageUrl;
43.         }
44.     }
45. }

```

WaitUtil.cs

```

1. using OpenQA.Selenium;
2. using OpenQA.Selenium.Support.UI;
3. using System;
4. using System.Collections.Generic;
5. using System.Linq;
6. using System.Text;
7. using System.Threading.Tasks;
8. using SeleniumExtras.WaitHelpers;
9.
10. namespace Diploma.Automation.Support
11. {
12.     public static class WaitUtil
13.     {
14.         public static void ShouldLocate(IWebDriver webDriver, string location)
15.         {
16.             try
17.             {
18.                 new WebDriverWait(webDriver, TimeSpan.FromSeconds(10)).Until(c => c.Url
19. == location);
20.             }
21.             catch (WebDriverTimeoutException ex)
22.             {
23.                 throw new NotFoundException($"Cannot find out that app in specific
24. location: {location}", ex);
25.             }
26.         }
27.     }
28. }

```

```

25.
26.     public static void WaitSomeInterval(int seconds = 10)
27.     {
28.         Task.Delay(TimeSpan.FromSeconds(seconds)).Wait();
29.     }
30.
31.     public static void WaitElementVisible(IWebDriver webDriver, By locator, int
seconds = 10)
32.     { try { new WebDriverWait(webDriver,
TimeSpan.FromSeconds(seconds)).Until(ExpectedConditions.ElementIsVisible(locator));
33.     }
34.     catch (WebDriverTimeoutException ex)
35.     {
36.         throw new NotFoundException($"Cannot find out that app in specific
location: {locator}", ex);
37.     }
38.
39.     }
40.     public static void WaitElementClickable(IWebDriver webDriver, By locator, int
seconds = 10)
41.     {
42.         try
43.         {
44.             new WebDriverWait(webDriver,
TimeSpan.FromSeconds(seconds)).Until(ExpectedConditions.ElementToBeClickable(locator));
45.         }
46.         catch (WebDriverTimeoutException ex)
47.         {
48.             throw new NotFoundException($"Cannot find out that app in specific
location: {locator}", ex);
49.         }
50.     }
51. }
52. }

```

HomePage.cs

```

1. using Diploma.Automation.Support;
2. using OpenQA.Selenium;
3.
4. namespace Diploma.Automation.Pages
5. {
6.     public class HomePage
7.     {
8.         public static string PageUrl = ("Index");
9.         private readonly By manageLocator = By.Id("manage");
10.        private      readonly      By      adminNavbarDropdownLocator      =
By.Id("navbarDropdown");
11.
12.        private      readonly      By      categoryDropdownLocator      =
By.XPath("//a[text() = 'Category']");
13.        public string GetCurrentUser()
14.        {
15.            return Browser.Driver.FindElement(manageLocator).Text;
16.        }
17.        public bool AdminNavbarIsVisible =>
18.            Browser.Driver.FindElement(adminNavbarDropdownLocator).Displ
ayed;
19.

```

```

20.         public void OpenCategoryPage()
21.         {
22.             Browser.Driver.FindElement(adminNavbarDropdownLocator).Click();
23.             Browser.Driver.FindElement(categoryDropdownLocator).Click();
24.         }
25.         public HomePage Goto() {
26.             Browser.Goto(Browser.FullURL(PageUrl));
27.             WaitUtil.ShouldLocate(Browser.Driver,
28. Browser.FullURL(PageUrl));
29.             return this; } }

```

LoginPage.cs

```

1. using Diploma.Automation.Suport;
2. using OpenQA.Selenium;
3.
4. namespace Diploma.Automation.Pages
5. {
6.     public class LoginPage
7.     {
8.         public static string PageUrl = ("Identity/Account/Login");
9.
10.        private readonly By EmailInputLocator = By.Id("Input_Email");
11.        private readonly By PasswordInputLocator = By.Id("Input_Password");
12.        private readonly By LoginButonLocator = By.CssSelector(".btn.btn-primary");
13.
14.        public void NewLogIn(string email, string password)
15.        {
16.            Browser.Driver.FindElement(EmailInputLocator).SendKeys(email);
17.            Browser.Driver.FindElement>PasswordInputLocator).SendKeys(password);
18.            Browser.Driver.FindElement(LoginButonLocator).Click();
19.        }
20.        public LoginPage Goto()
21.        {
22.            Browser.Goto(Browser.FullURL(PageUrl));
23.            WaitUtil.ShouldLocate(Browser.Driver, Browser.FullURL(PageUrl));
24.            return this;
25.        }
26.    }

```

CategoryPage.cs

```

1. using Diploma.Automation.Suport;
2. using OpenQA.Selenium;
3. using System;
4.
5. namespace Diploma.Automation.Pages
6. {
7.     public class CategoryPage
8.     {
9.         private readonly By addButton = By.CssSelector(".fas.fa-plus");
10.        public AddCategory pressAddButton()
11.        {
12.            Browser.Driver.FindElement(addButton).Click();
13.            return new AddCategory();
14.        }
15.    }

```


UserLifeCycleTest.cs

```

1. using Diploma.Automation.Pages;
2. using Diploma.Automation.Personas;
3. using Dipoma.Automation.Tests.TestSuport;
4. using NUnit.Framework;
5.
6. namespace Dipoma.Automation.Tests
7. {
8.     [TestFixture]
9.     public class UserLifeCycleTest : TestBase
10.    {
11.        [Test]
12.        public void LogInUserAsAdmin()
13.        {
14.            FactoryPages.LoginPage.Goto().NewLogIn(KnownPersonas.SystemAdmin.Username,
KnownPersonas.SystemAdmin.Password); ;
15.            Assert.AreEqual($"Hello {KnownPersonas.SystemAdmin.Username}!",
FactoryPages.HomePage.GetCurrentUser());
16.            Assert.IsTrue(FactoryPages.HomePage.AdminNavbarIsVisible);
17.        }
18.        [Test]
19.        public void LogInUser()
20.        {
21.            FactoryPages.LoginPage.Goto().NewLogIn(KnownPersonas.SystemAdmin.Username,
KnownPersonas.SystemAdmin.Password); ;
22.            Assert.AreEqual($"Hello {KnownPersonas.SystemAdmin.Username}!",
FactoryPages.HomePage.GetCurrentUser());
23.            Assert.IsTrue(FactoryPages.HomePage.AdminNavbarIsVisible);
24.        }
25.    }

```

ContentTest.cs

```

1. using Diploma.Automation.Pages;
2. using Diploma.Automation.Personas;
3. using Dipoma.Automation.Tests.TestSuport;
4. using NUnit.Framework;
5.
6. namespace Dipoma.Automation.Tests.Tests
7. {
8.     [TestFixture]
9.     public class ContentTest : TestBase
10.    {
11.        [Test]
12.        public void AddCategory()
13.        {
14.            FactoryPages.LoginPage.Goto().NewLogIn(KnownPersonas.SystemAdmin.U
sername, KnownPersonas.SystemAdmin.Password); ;
15.            Assert.AreEqual($"Hello {KnownPersonas.SystemAdmin.Username}!",
FactoryPages.HomePage.GetCurrentUser());
16.            Assert.IsTrue(FactoryPages.HomePage.AdminNavbarIsVisible);
17.            FactoryPages.HomePage.OpenCategoryPage();
18.            FactoryPages.CategoryPage.pressAddButton().AddNewCategory("Test",
"111");
19.        }

```

Додаток Ж. Екрани реалізованої програми

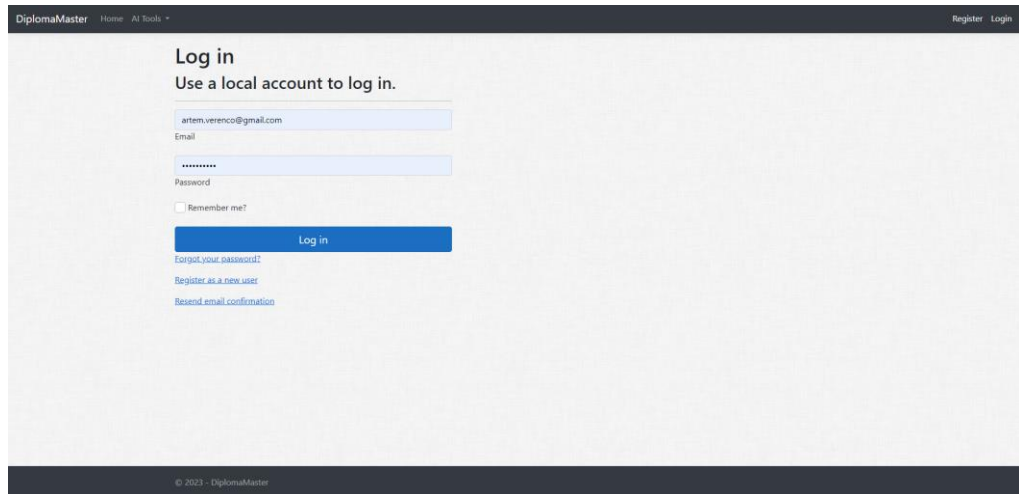


Рис. Д. 1. Вікно входу в систему для створення контенту

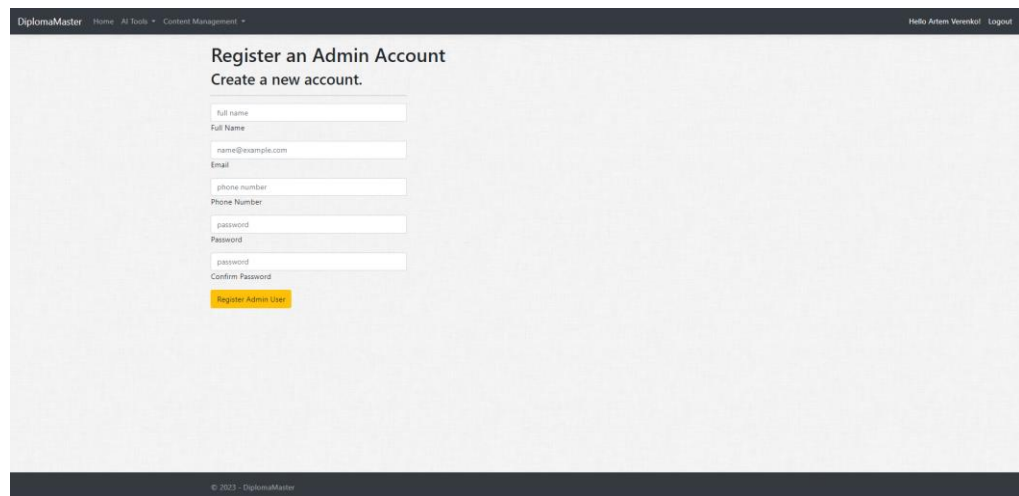


Рис. Д. 2. Вікно реєстрації нового адміністратора

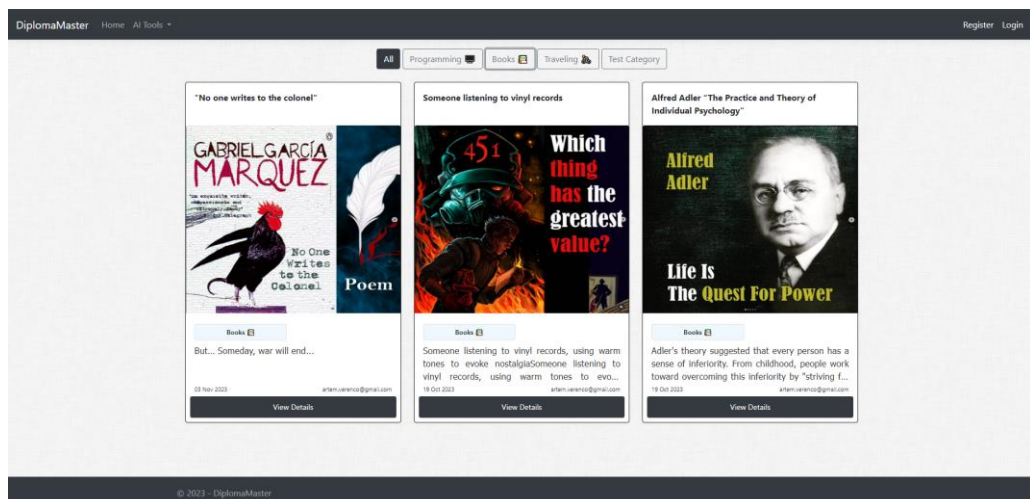


Рис. Д. 3. Головний екран

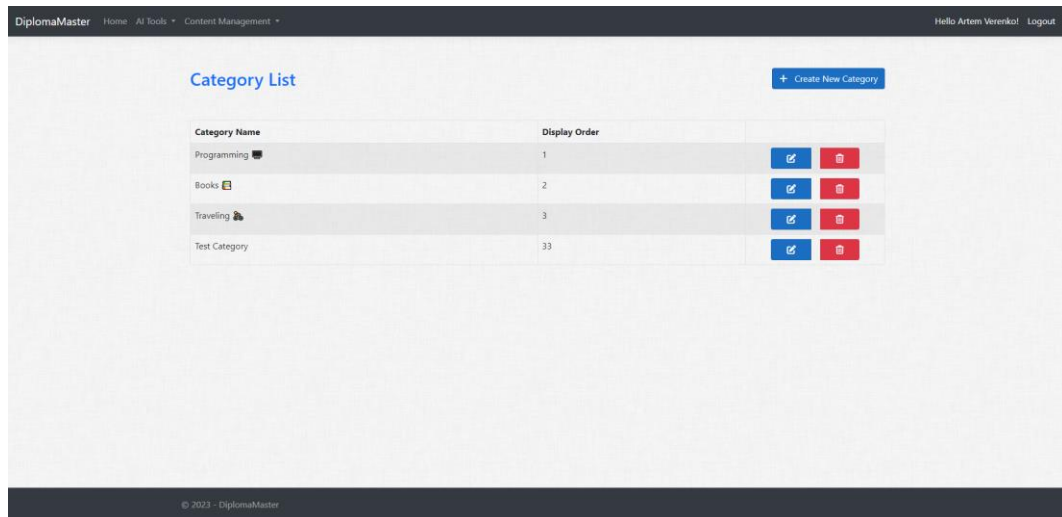


Рис. Д. 4. Вікно управління категоріями

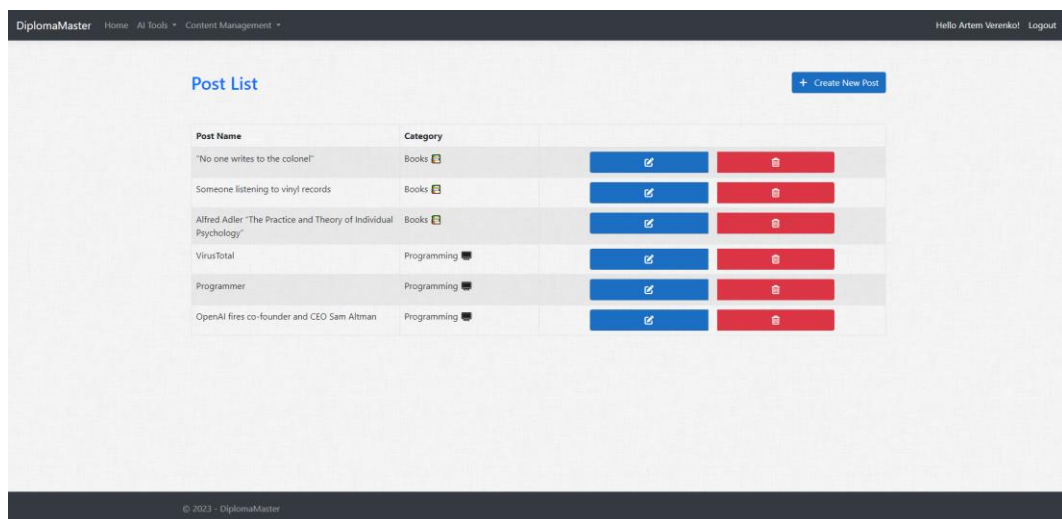


Рис. Д. 5. Вікно управління публікаціями

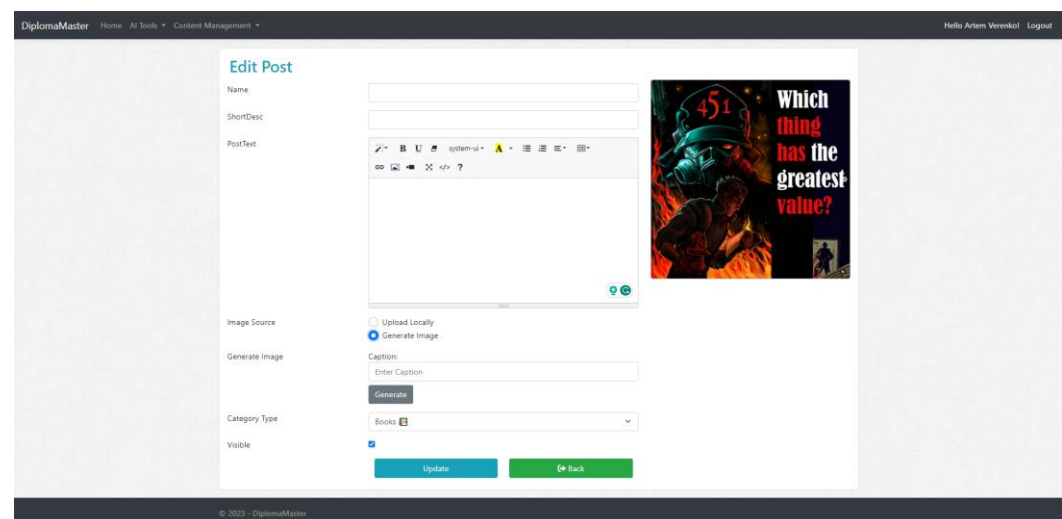


Рис. Д. 6. Вікно редагування публікації

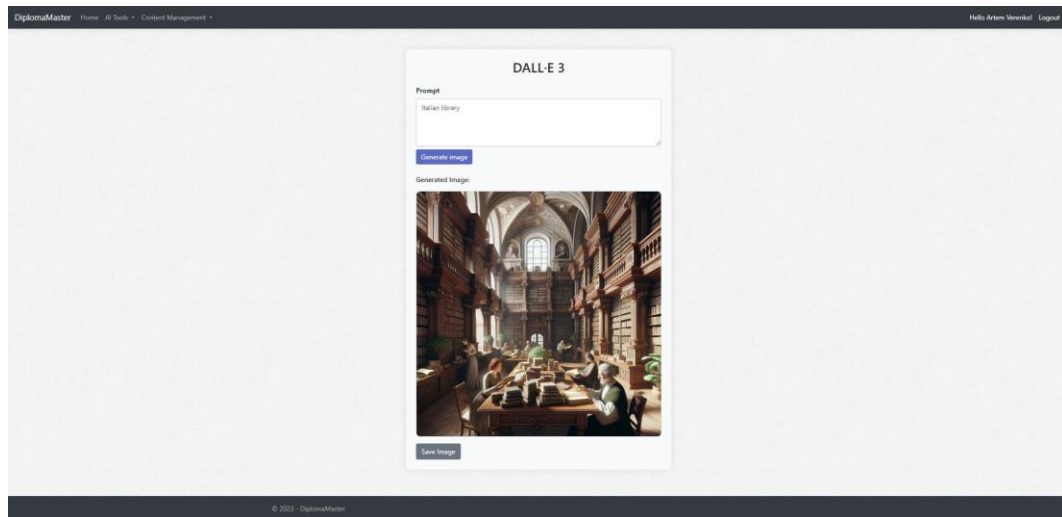


Рис. Д. 7. Вікно Dall E 3 пісочниця

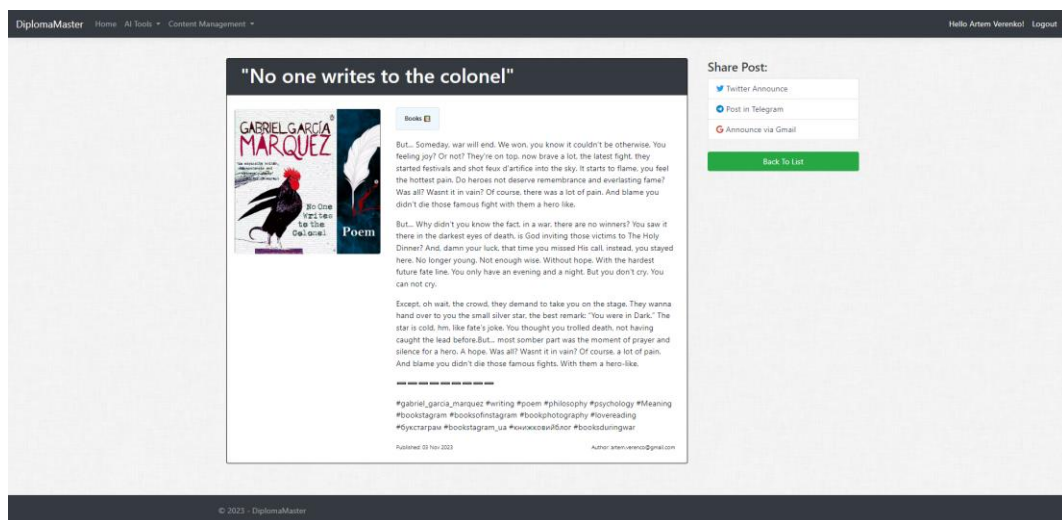


Рис. Д. 8. Вікно відображення публікації

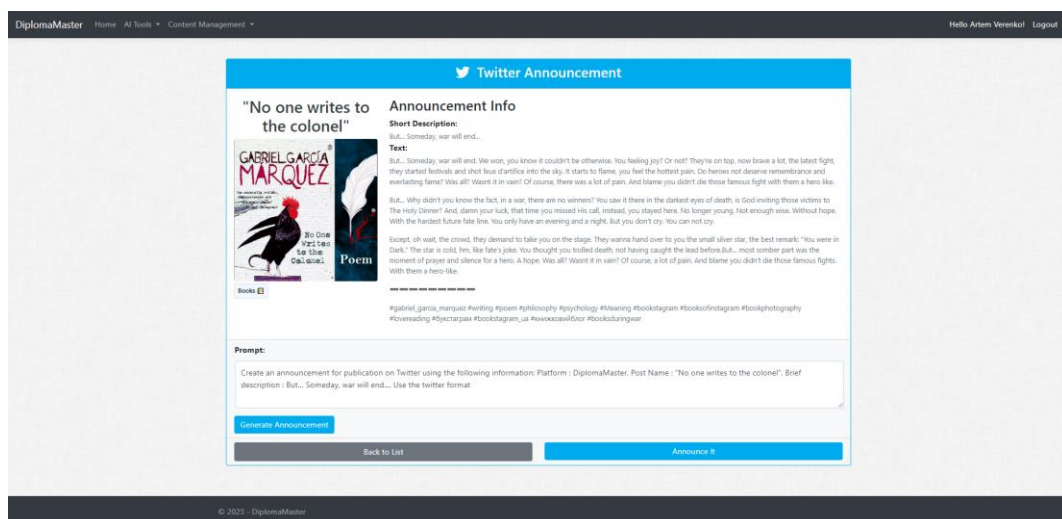


Рис. Д. 9. Вікно підготовки Twitter анонсу

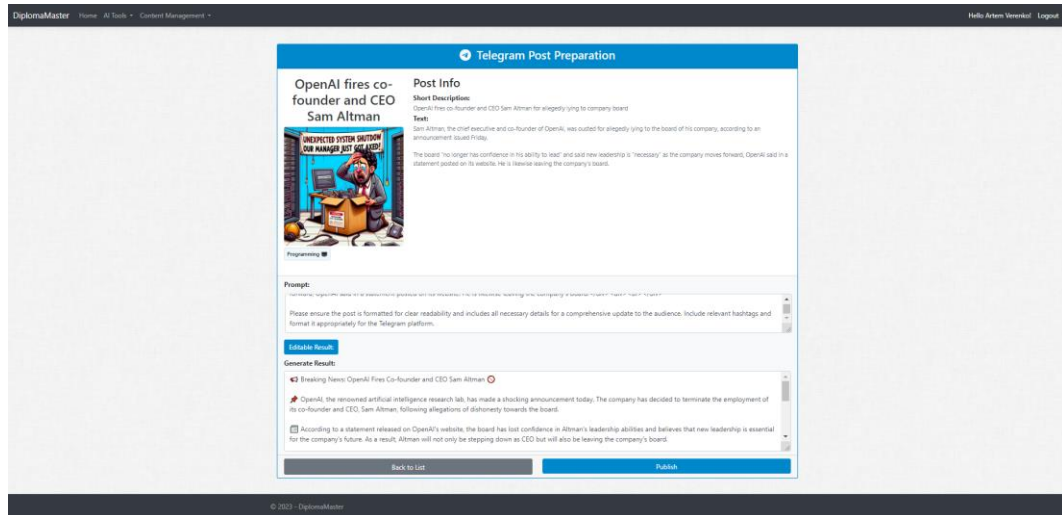


Рис. Д. 10. Вікно підготовки Telegram публікації

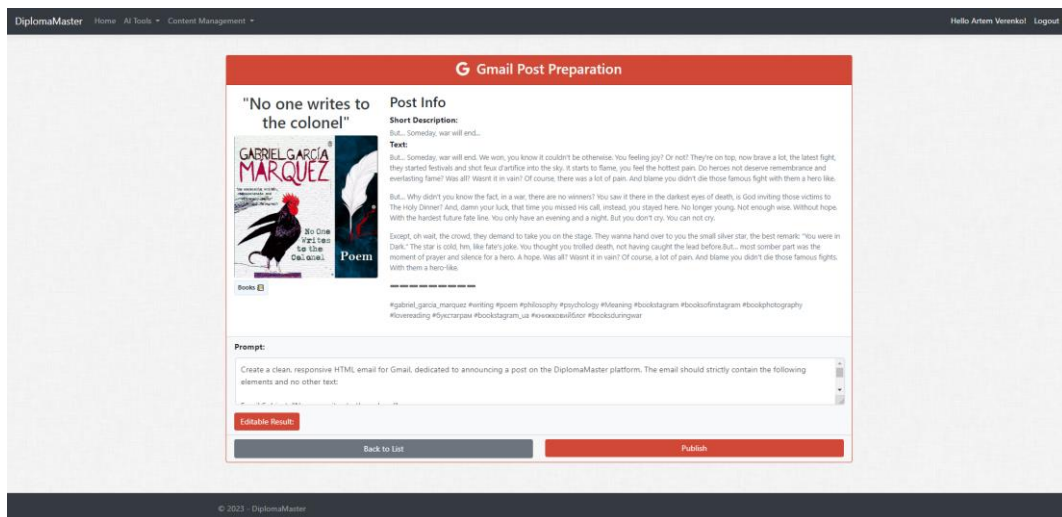


Рис. Д. 11. Вікно підготовки Gmail листа