Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі

Виконала: студентка 2-го курсу, групи 3КН-23м
спеціальності 122 – «Комп'ютерні науки»
Верба Г. Р.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. АІІТ
Богач І. В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця – 2024 року

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
_____
(підпис)
«11» 10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Вербі Ганні Романівній
(прізвище, ім'я, по батькові)

1. Тема роботи: «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі»
керівник роботи к.т.н., доц., доц. каф. КН, Крилик Л. В.
затверджені наказом вищого навчального закладу від «17» 09 2024 року №310

2. Строк подання студентом роботи 11.11.2024 2024 року

3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована; формат вхідних даних – текст; вхідні параметри – не меньше 2 шт.; роздільна здатність зображення не менше 800×600 пікселів; формат вихідних даних – .zip, .png, .jpg; інтерфейс користувача має бути інтуїтивно зрозумілим.

4. Зміст текстової частини: вступ, обґрунтування доцільності розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі; моделювання інформаційної технології створення зоображень за допомогою генеративної нейронної мережі; програмна реалізації інформаційної технології створення зображень за допомогою генеративної нейронної мер економічна частина, висновки, список використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма послідовності взаємодії головних модулів інформаційної технології створення зображень за допомогою генеративної нейронної мережі; структурна схема програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі; схема алгоритму роботи інформаційної технології створення зображень за допомогою генеративної нейронної мережі; діаграма діяльності серверу під час процесу генерації зображення; діаграма діяльності серверу під час процесу отримання зображень користувача для галереї; діаграма діяльності серверу під час процесу скачування користувачем архіву; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1–3	Крилик Л. В., к.т.н., доц. каф. КН		
4	Адлер О. О., к.т.н., доц. каф. ЕПВМ		

7. Дата видачі завдання 02.09. 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обгрунтування доцільності розробки	02.09.24-03.09.24	Розділ 1
2	Моделювання інформаційної технології створення зображень за допомогою генеративної нейронної мережі	03.09.24-23.09.24	Розділ 2
3	Програмна реалізація інформаційної технології створення зображень за допомогою генеративної нейронної мережі	23.09.24-28.10.24	Розділ 3
4	Підготовка економічної частини	29.10.24-04.11.24	Розділ 4
5	Апробація результатів дослідження	17.10.24	тези доповіді, авторське право на твір
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24-11.11.24	Пояснювальна записка, графічний матеріал, презентація

Студентка _____ Верба Г.
(підпис) (прізвище та ініці

Керівник роботи _____ Крилик Л.
(підпис) (прізвище та ініці

АНОТАЦІЯ

УДК 004.42

Верба Г. Р. Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 123 с.

Укр. мовою. Бібліогр.: 28 назв; рис.: 22; табл.: 12.

У магістерській кваліфікаційній роботі проведено аналіз сфери створення зображень за допомогою нейронних мереж. Досліджено основні підходи до створення зображень за допомогою цих технологій, здійснено аналіз аналогів, розглянуто методи та технології для вирішення схожих завдань та обґрунтовано доцільність розробки. Розроблено структурну модель інформаційної технології та математичну модель для створення зображень за допомогою нейронних мереж, що сприяє розширенню функціональних можливостей інформаційної технології, а також забезпечує можливість генерувати зображення використовуючи індивідуальні налаштування користувача.

Ця робота дає можливість користувачу отримати новий інструмент для створення зображень за допомогою нейронних мереж в якому поєднано зручний зрозумілий дизайн, можливість налаштувати бажане зображення та довготривале збереження згенерованих зображень.

В роботі приведені результати досліджень та аналіз роботи розробленої інформаційної технології створення зображень за допомогою генеративної нейронної мережі. Доведена економічна доцільність розробки.

Ілюстративна частина складається з 7 плакатів із результатами моделювання.

Ключові слова: генеративні нейронні мережі, інформаційні технології, генерація зображень, штучний інтелект, творчість.

ABSTRACT

Verba H.R. Information technology for image generation using a generative neural network. Master's thesis in the specialty 122 «Computer Science», educational program «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 123 pages.

In Ukrainian language. Bibliography: 28 titles; figures: 22; tables: 12.

In the master's qualification thesis, an analysis of the field of image creation using neural networks was carried out. The main approaches to creating images with the help of these technologies were studied, the analysis of analogs was carried out, the methods and technologies for solving similar tasks were considered, and the feasibility of the development was substantiated. A structural model of information technology and a mathematical model for creating images using neural networks have been developed, which contributes to the expansion of the functional capabilities of information technology, and also provides the ability to generate images using individual user settings.

This work enables the user to get a new tool for creating images with the help of neural networks, which combines a convenient and understandable design, the ability to adjust the desired image and long-term storage of the generated images.

The paper presents research results and analysis of the developed information technology for creating images using a generative neural network. The economic feasibility of the development has been proven.

The illustrative part consists of 7 posters with simulation results.

Keywords: generative neural networks, information technology, image generation, artificial intelligence, creativity.

ЗМІСТ

ВСТУП

Актуальність теми. Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі вирішує завдання створення креативних та унікальних графічних витворів за допомогою сучасних технологій штучного інтелекту. У наш час, коли важливість швидкого та ефективного створення високоякісних зображень стає все більш актуальною, використання алгоритмів AI Horde, які засновані на генеративній нейронній мережі, надає можливість широкому колу користувачів творити графіку за допомогою простого та зрозумілого інтерфейсу. Застосування AI Horde як кластера, що використовує генеративну нейронну мережу в інформаційній технології створення зображень, може знайти своє застосування в різних сферах − від мистецтва до дизайну, створюючи унікальні та захоплюючі візуальні елементи.

Доцільність розробки інформаційної технології для створення зображень за допомогою генеративних нейронних мереж базується на потребі універсального та інтуїтивно зрозумілого інструменту для створення візуального контенту. Це забезпечується наданням користувачам можливості перегляду попередньо згенерованих зображень та введених слів, а також можливості генерації нових зображень. Інформаційна технологія сприяє не лише полегшенню творчого процесу, але й збільшенню продуктивності та креативності користувачів. Інтеграція генеративної нейронної мережі в існуючі інформаційні технології створює потужний інструмент, який допомагає задовольняти потреби користувачів у візуальному контенті, а також підтримує їхню творчість та інноваційний потенціал [1].

Розробка інформаційної технології створення зображень за допомогою генеративної нейронної мережі є актуальною натепер і полягає в можливості зробити процес створення графіки більш доступним, креативним та ефективним для широкого кола користувачів.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових

досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем».

Мета і завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей інформаційної технології створення зображень на основі генеративної нейронної мережі за рахунок використання AiHorde API. Це дозволить розробити зручний інструмент для генерації та менеджменту згенерованих зображень.

Для досягнення поставленої мети потрібно розв’язати такі **завдання**:

– Провести аналіз технічного рівня існуючих систем генерації зображень та обґрунтувати доцільність розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

– Розробити математичну модель інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

– Розглянути методи та технології створення зображень нейронними мережами.

– Спроектувати архітектуру та структуру програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

– Проаналізувати та обґрунтувати вибір інструментів для розробки.

– Виконати програмну реалізацію програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

– Виконати тестування програми та провести аналіз отриманих результатів.

– Економічно обґрунтувати доцільність розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

Об’єктом дослідження є процес створення зображень за допомогою генеративної нейронної мережі.

Предметом дослідження є програмні засоби проектування та реалізації інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

Методи дослідження. У процесі дослідження використовувались: дослідження сучасних методів генерації зображень, методи та підходи до розробки WEB-орієнтованих програмних додатків, методи об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

- удосконалено математичну модель інформаційної технології створення зображень за допомогою генеративної нейронної мережі, яка відрізняється від існуючих здатністю налаштовувати стиль і зміст зображень на основі вхідних параметрів, що дозволяє ефективно налаштовувати генеративну модель під індивідуальні потреби користувача;

- удосконалено інформаційну технологію створення зображень за допомогою генеративної нейронної мережі, яка відрізняється від існуючих розширеним функціоналом, а саме можливістю налаштовувати параметри майбутнього зображення на свій смак та довготривалим збереженням згенерованих користувачем зображень, це дозволить користувачу повертатися до них з плином часу.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм функціонування інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

2. Розроблено діаграму послідовності взаємодії модулів інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

3. Здійснено програмну реалізацію інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів під час доведення наукових положень, порівнянням результатів із

відомими та збіжністю результатів моделювання з результатами, які отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Результати магістерської кваліфікаційної роботи отримані самостійно. В публікації у співавторстві здобувачу належить дослідження особливостей розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі та обгрунтована доцільність її застосування [1].

Апробація результатів роботи. Результати досліджень було апробовано на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» м. Вінниці у 2024 р. [1].

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тези доповіді на науково-технічній конференції [1] та отримано свідоцтво про реєстрацію авторського права на твір у формі комп'ютерної програми – «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» [2].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЕНЕРАТИВНОЇ НЕЙРОННОЇ МЕРЕЖІ

1.1 Аналіз предметної галузі створення зображень за допомогою генеративної нейронної мережі

Використання нейронних мереж для створення зображень почалося з простих архітектур, таких як багатошарові перцептрони, але з часом еволюціонувало в складні моделі, здатні генерувати зображення з високою роздільною здатністю. Зокрема, технології, такі як згорткові нейронні мережі (CNN), стали основою для багатьох систем, що працюють з візуальним контентом.

Розглянемо застосування в різних сферах:

– Мистецтво та дизайн: нейронні мережі активно використовуються для створення цифрового мистецтва. Художники можуть генерувати нові стилі, а також комбінувати елементи з різних картин для отримання унікальних творів.

– Реклама та маркетинг: генерація зображень для рекламних кампаній, ілюстрацій для WEB-сайтів та соціальних мереж допомагає компаніям швидко отримувати потрібний контент без необхідності залучення художників.

– Мода: у модній індустрії нейронні мережі використовуються для створення дизайнів одягу, передбачення трендів та навіть для віртуальних подіумів, де моделі демонструють одяг.

– Медичні зображення: нейронні мережі можуть генерувати та покращувати медичні зображення, такі як рентгенівські знімки чи знімки МРТ, це допомагає лікарям у діагностиці.

– Ігрова індустрія: генерація персонажів, ландшафтів та навіть цілого світу в іграх стала можливою завдяки нейронним мережам, які створюють унікальний контент на основі заданих параметрів.

Хоча технології створення зображень нейронними мережами мають значний потенціал, вони також стикаються з рядом недоліків:

– Якість результатів: іноді згенеровані зображення можуть бути недостатньо реалістичними або містити артефакти, що робить їх непридатними для використання.

– Обчислювальні ресурси: процес навчання нейронних мереж вимагає значних обчислювальних ресурсів, що може бути перешкодою для окремих користувачів або малих компаній.

– Етичні питання: використання зображень, що порушують авторські права, або створення контенту, який може бути неправильно витлумачений, викликає етичні та правові питання.

Сфера створення зображень за допомогою нейронних мереж активно розвивається. Наразі існує велика кількість досліджень, присвячених вдосконаленню моделей, підвищенню їхньої точності та швидкості генерації. Нові архітектури, такі як нейронні мережі на основі трансформерів, обіцяють підвищити якість та можливості генерації зображень.

Крім того, інтеграція нейронних мереж в процеси візуалізації, такі як анімація або віртуальна реальність, відкриває нові горизонти для створення інтерактивного контенту. З часом можна очікувати на зростання попиту на такі технології в різних сферах, включаючи освітні програми, розваги та промисловість.

1.2 Огляд відомих програмних систем-аналогів

Один з потенційних аналогів проекту − Midjorney. Це нейромережа, що перетворює написаний користувачем текст на зображення, які потім можна використовувати навіть у комерційних цілях. Звичайно, поки вона не може замінити людського дизайнера, але з простими завданнями, які не вимагають високої точності та відповідності справляється легко [3].

Midjourney − це платна нейромережа, яка має безкоштовний демо-доступ, в якому доступна генерація всього 25 зображень (рис. 1.1). Спочатку Midjourney функціонувала як чат-бот у Discord, де користувачі могли взаємодіяти з нейромережею, відправляючи свої запити прямо в загальному чаті (рис. 1.2).

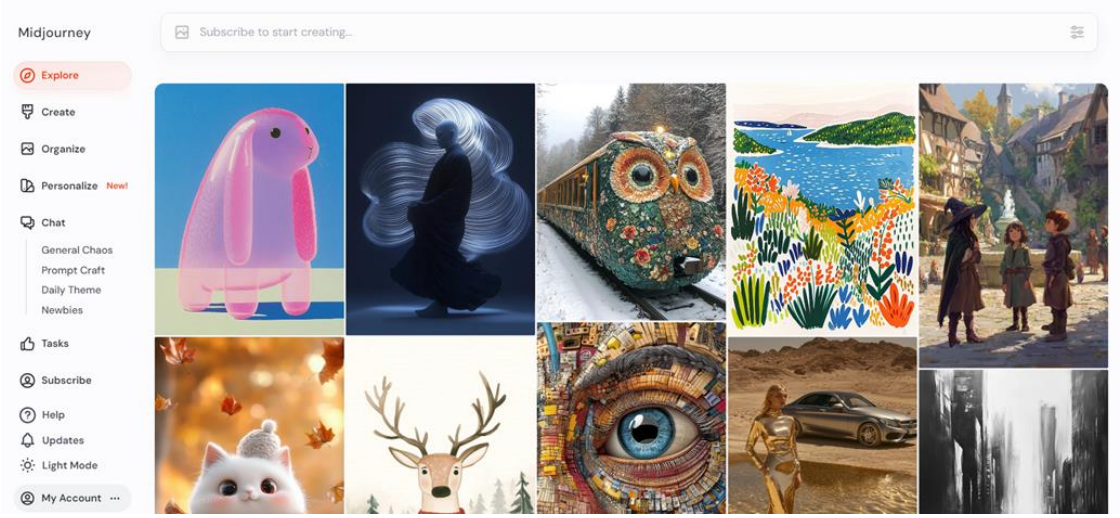

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна Midjorney

Ця форма спілкування дозволяла користувачам легко генерувати зображення, проте мала свої недоліки. По-перше, всі запити та результати роботи нейромережі були видимі для всіх учасників чату, що іноді ускладнювало процес взаємодії. По-друге, через величезну кількість запитів від користувачів, які спілкувалися в цьому загальному чаті, час очікування та швидкість генерації зображень могли суттєво варіюватися, що ускладнювало адекватне користування сервісом.

З часом, на основі цього успішного досвіду, Midjourney була розроблена платформа, що пропонує більш зручний та ефективний інтерфейс для генерації зображень. Крім базового безкоштовного користування, Midjourney пропонує також три тарифних плани на вибір: Basic, Standard та Pro. Ці плани відрізняються між собою ціною, кількістю доступних генерацій у режимах Relax (коли нейромережа обробляє запити по черзі) та Fast (обробляє запит із найвищим пріоритетом у режимі реального часу). Користувачі можуть вибирати

план, що найкраще відповідає їхнім потребам, щоб отримати максимальну вигоду від використання сервісу.

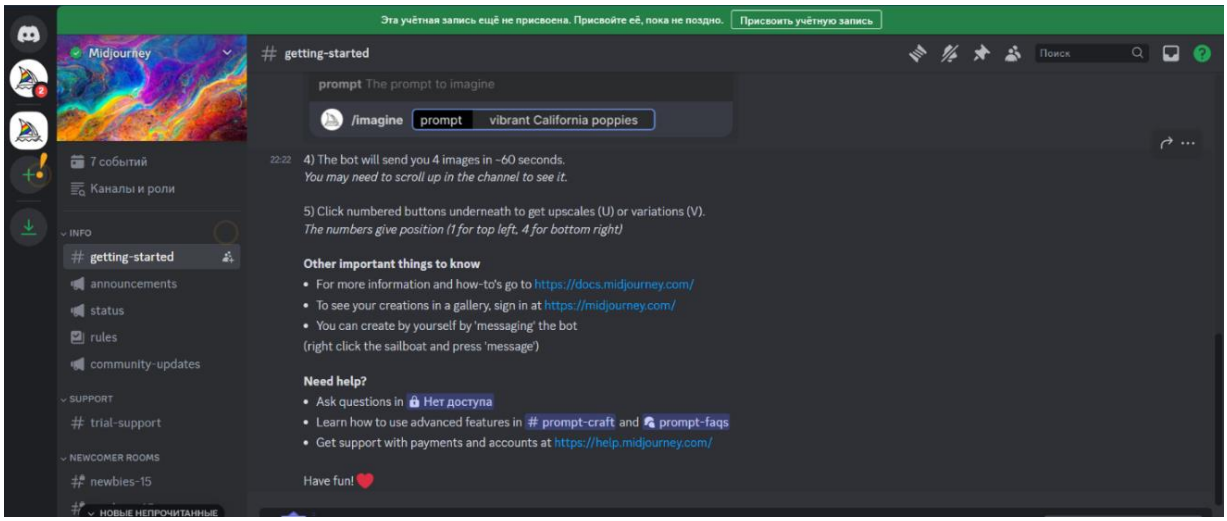

Рисунок 1.2 – Загальний вигляд чат-бота Midjorney в Discord

Переваги Midjorney:

– Висока якість генерованих зображень;

– Широкий спектр можливостей для творчості та налаштувань;

– Зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Недоліки Midjorney:

– Обмеження вибору параметрів генерації;

– Обмеження в кількості генерованих картинок яке залежить від вашого тарифного плану;

– Залежність від якості вхідних даних та параметрів.

Інший важливий гравець у галузі генерації зображень − DeepArt. Це інноваційна програма, яка використовує передову технологію штучного інтелекту для створення мистецтва з тексту, пропонуючи користувачам можливість перетворювати прості описи на приголомшливі візуальні твори, натхненні різними стилями мистецтва [4].

Технологія DeepArt базується на передових нейронних алгоритмах, які не лише відтворюють стилістичні елементи, але й зберігають цілісність оригінального зображення під час трансформації. Це забезпечує детальну

обробку зображень, що дозволяє отримувати високу якість навіть після застосування художнього стилю. DeepArt демонструє, як технології можуть об’єднувати традиційні та сучасні форми мистецтва, відкриваючи нові можливості для творчого самовираження.

Розгляннкмо переваги та недоліки даної платформи:

Переваги:

– Потужна обробка зображень;
– Використовує штучний інтелект;
– Масштабує та розфарбовує зображення;
– Доступно на комп’ютері та мобільному пристрої;
– Пропонує API для інтеграції;
– Дозволяє створювати власні художні фільтри.

Недоліки:

– Потрібно затратити багато часу, щоб навчитися з ним працювати;
– Без безкоштовного плану.

Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.3 та рисунку 1.4.

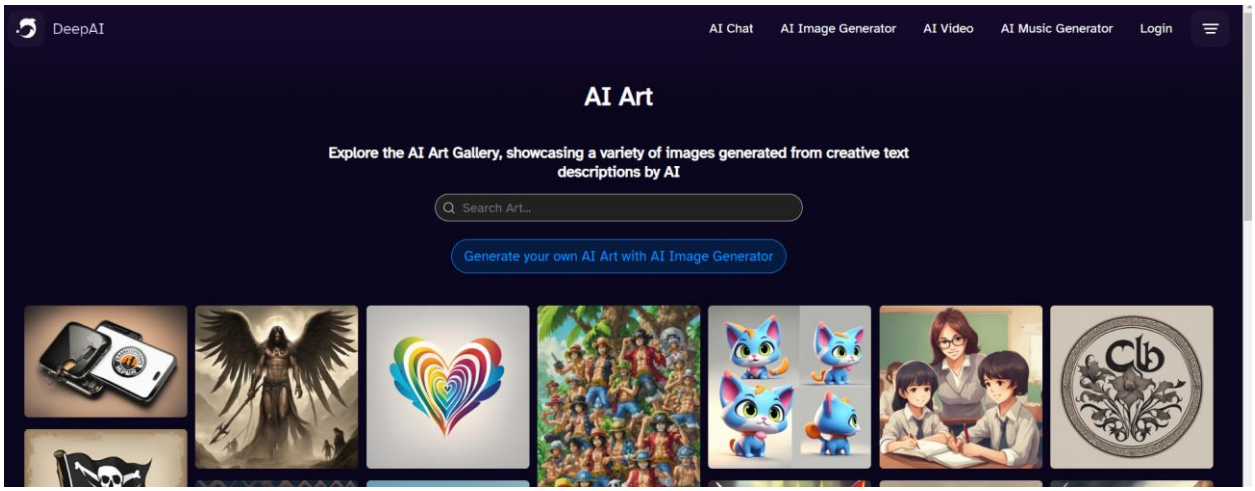

Рисунок 1.3 – Загальний вигляд інтерфейсного вікна сайту DeepArt

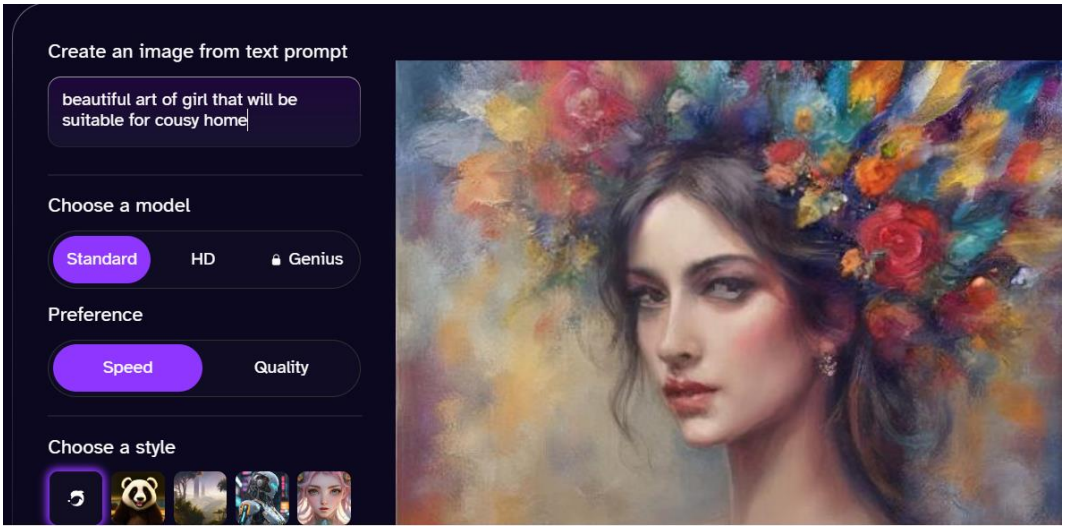

Рисунок 1.4 – Загальний вигляд інтерфейсного вікна сайту DeepArt та згенерованого зображення

Наступним сайтом для розгляду є OpenArt. OpenArt AI це зручна платформа, яка використовує передові алгоритми, такі як DALL-E 3, для створення вражаючих, різноманітних і високоякісних ілюстрацій із текстових описів (рис. 1.5, 1.6). OpenArt підтримує такі функції, як генерація зображень з тексту, конвертація зображень у відео, редагування зображень (inpainting), а також налаштування моделей для досягнення персоналізованих результатів.

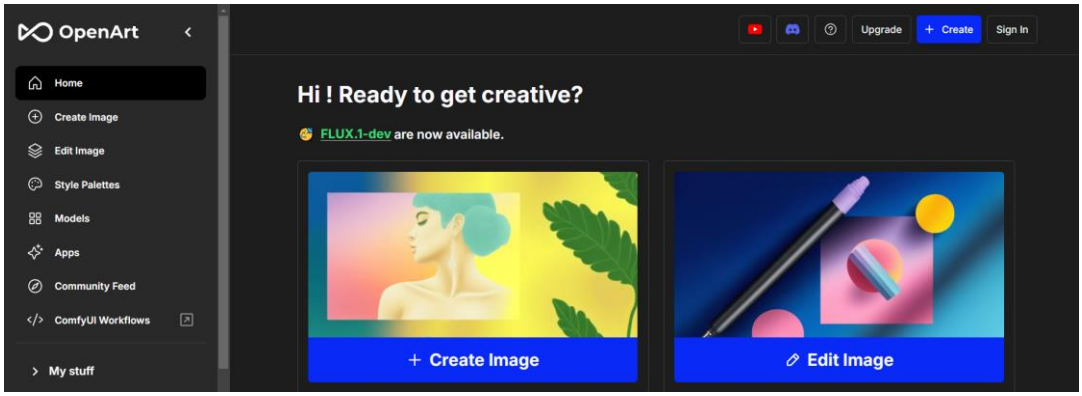

Рисунок 1.5 – Загальний вигляд інтерфейсного вікна сайту OpenArt

Платформа також інтегрована з хмарними сервісами, що полегшує збереження та обмін створеними роботами. Незважаючи на деякі обмеження безкоштовної версії та випадкові спотворення результатів, широкі інструменти

редагування та безмежні можливості для творчості роблять її найкращим вибором для художників усіх рівнів, які прагнуть втілити свої бачення в життя [5].

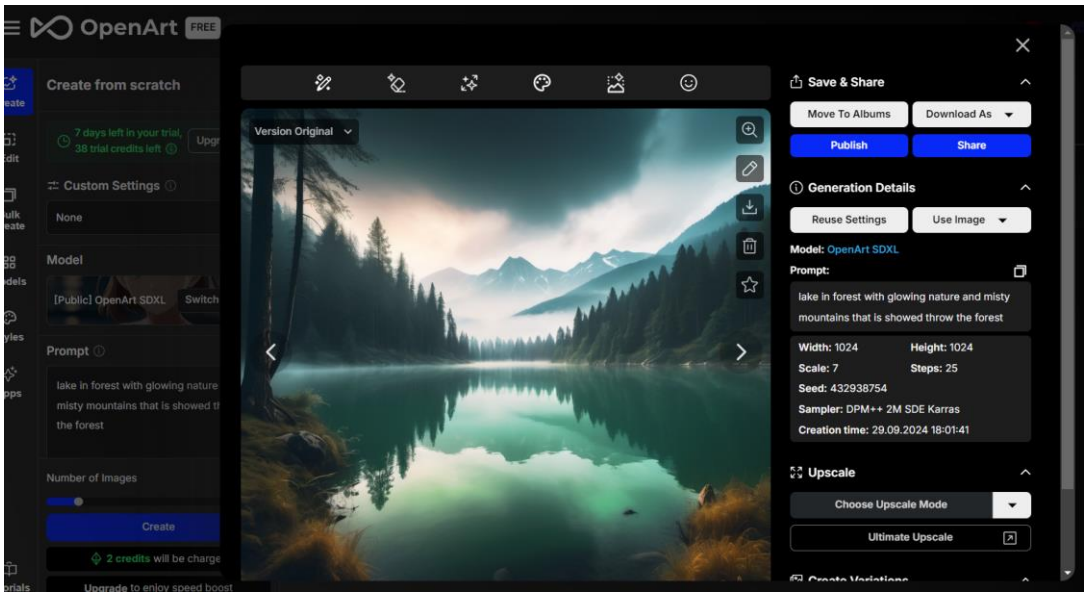

Рисунок 1.6 – Загальний вигляд інтерфейсного вікна сайту OpenArt

Переваги:

– Зручний інтерфейс для будь-якого рівня кваліфікації;

– Безкоштовний план із 500 кредитами, щоб спробувати всі функції;

– Понад 100 моделей і стилів на вибір для багатьох мистецьких досліджень;

– Тренуйте моделей, щоб створити свій унікальний стиль;

– Багато ресурсів, наприклад, відеоуроки, книга підказок і шаблони підказок;

– Галерея прикладів для конкретних інструментів від спільноти OpenArt AI.

Недоліки:

– Безкоштовна версія має обмеження;

– Деякі картинки можуть бути спотворені, що може дратувати.

1.3 Постановка задачі дослідження

Швидкий розвиток технологій штучного інтелекту та його інтеграція в повсякденне життя створюють нові можливості для різних сфер, включаючи творчі галузі. Генеративні нейронні мережі, здатні створювати зображення за текстовим запитом, забезпечують як професіоналів, так і любителів доступом до інструментів автоматизованого генерування зображень, що дозволяє реалізувати складні творчі проєкти з мінімальними витратами ресурсів. Зокрема, можливості таких систем, як Midjourney, OpenArt, DeepArt, демонструють значний попит на подібні технології, хоча вони мають обмеження у гнучкості та масштабованості.

Проблема полягає в необхідності створення ефективного, доступного та масштабованого інструменту для генерації зображень за текстовим запитом, який би відповідав високим стандартам якості зображень та забезпечував зручний і персоналізований досвід для користувача.

Загальний опис функціоналу інформаційної технології для генерації зображень на основі текстових запитів:

– Інтеграція сучасних генеративних моделей: використання провідних нейронних мереж для забезпечення якості та різноманітності зображень.

– Розробка зручного інтерфейсу: можливість вибору стилю, кольорової гами та інших параметрів зображення.

– Інтерактивний вибір параметрів: гнучкість у налаштуванні та збереженні попередніх налаштувань для повторного використання.

– Можливість комерційного використання зображень: забезпечення користувача правами на використання зображень в комерційних цілях.

У цій роботі потрібно розробити інформаційну технологію для генерації зображень на основі текстових запитів, яка забезпечить користувача ефективним інструментом для створення візуального контенту. Це дозволить автоматизувати процеси, пов'язані з дизайном, візуалізацією та мистецтвом, і розширити

можливості користувача, надаючи доступ до технологій штучного інтелекту для творчої роботи.

Таким чином, вирішення поставлених задач дозволить розробити платформу яка буде орієнтована на якість та зручність генерації зображень. Зокрема, технологія буде орієнтована на проектування архітектури для інтеграції та ефективного використання генеративних моделей, оптимізацію якості та параметрів зображень, розробку інтерфейсу для взаємодії користувача із системою, включаючи налаштування та вибір стилю зображень, тестування та порівняльний аналіз технології щодо аналогів з урахуванням параметрів продуктивності та якості.

1.4 Висновок до розділу 1

У першому розділі було розглянуто основні концепції генеративних нейронних мереж та проаналізовано актуальні технологічні рішення для створення зображень на основі текстових запитів. Проведено огляд відомих систем-аналогів, таких як Midjourney, OpenArt, DeepArt та висвітлено особливості їх функціонування, що включають обмеження щодо доступу, комерційного використання та гнучкості налаштувань для користувачів. Розглянуті аналоги підтверджують необхідність у розробці універсальної інформаційної технології, що забезпечить користувачам високий рівень персоналізації та простоту використання для генерування візуального контенту.

Аналіз відомих рішень виявив певні обмеження у їхньому функціоналі, що робить розробку нової інформаційної технології для генерації зображень за текстовими запитами актуальною. Таке рішення має забезпечувати ширші можливості для налаштування та використання генеративних моделей, зокрема можливість вибору стилю, параметрів зображення та інтеграції в різноманітні процеси користувача.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЇНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЕНЕРАТИВНОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обгрунтування вибору методу створення зображень за допомогою генеративної нейронної мережі

Вибір методів реалізації інформаційної технології є критично важливим етапом у процесі розробки системи для генерації зображень за допомогою штучного інтелекту. Оскільки технології створення графічних елементів за допомогою генеративних нейронних мереж стрімко розвиваються, правильний підхід до архітектури та вибору методів є ключовим для забезпечення високої якості результатів і продуктивності системи. Цей розділ присвячений аналізу та обґрунтуванню вибору технологій, зокрема генеративних нейронних мереж, що лягли в основу розроблюваної технології, а також застосуванню AI Horde для забезпечення синергетичної роботи різних генеративних моделей.

AI Horde використовує кластерний підхід, що дозволяє підключати різні генеративні моделі (GAN, VAE, Diffusion Models) для отримання широкого спектру результатів. Розглянемо ці моделі:

Автокодувальники. Цей тип моделей складається з двох штучних нейронних мереж: кодувальника та декодувальника. Посередині знаходиться код (який часто також називають bottleneck, з англ. «вузьке місце») − це шар нейронів невеликої розмірності (у порівнянні з іншими шарами). Основна задача моделі − це зменшення розмірності даних, тобто кодування вхідних даних у вектор ознак меншої розмірності (код) ніж вони були спочатку та подальше відтворення (декодування) початкових даних з цього вектора [6].

Навчання автокодувальників відбувається за допомогою методу зворотного поширення помилки, а в основі функції втрат використовується різниця між оригінальними даними (наприклад, зображеннями), що подаються на вхід кодувальника та відновленими декодувальником.

З погляду генерації даних найбільшим недоліком класичних автокодувальників є те, що мінімізуючи функцію втрат, автокодувальник може різко змінювати прихований простір (навіть незважаючи на регуляризаційні механізми) [7]. В результаті точки, що знаходяться поруч, можуть відповідати за вкрай несхожі між собою за ознаками дані, або можуть існувати такі області в просторі між двома точками даних, які не дозволять відтворити щось схоже з реальним розподілом даних. Цей недолік має суттєвий вплив на вибір звичайних автокодувальників як моделей генерування даних.

Варіаційні автокодувальники. Цей тип генеративних моделей не має вищеописаного недоліку, притаманного звичайним автокодувальникам. Замість визначення для кожного прикладу вхідних даних фіксованих значень коду, варіаційний автокодувальник вивчає параметри його розподілу, стискаючи таким чином вхідну інформацію до обмеженого ймовірнісного латентного простору. А на вхід декодувальника надходить вектор, що випадково вибирається з цього розподілу.

У варіаційних автокодувальниках використовується так званий трюк перепараметрування, який під час тренування моделі уможливлює використання методу зворотного поширення помилки, попри випадковий характер вибору значень вектора кодування з розподілу.

Суть трюку перепараметрування полягає у відокремленні стохастичності вибору значень цього вектора (він фактично стає окремим входом) від основного шляху поширення сигналів у системі, що дозволяє виключити цю складову з процесу зворотного поширення помилки.

Проте може виникнути ситуація, коли кодувальник генерує розподіли з нульовою дисперсією або значно віддаленими середніми значеннями, що є еквівалентним у різних масштабах. Це може призвести до виникнення проблеми, характерної для звичайного автокодувальника. Щоб запобігти цьому, у функцію втрат включається міра розходження Кульбака–Лейблера як регуляризаційний параметр, яка порівнює згенеровані розподіли зі стандартним нормальним розподілом. Це дозволяє групувати дані якомога ближче до стандартного

нормального розподілу, що, у свою чергу, дозволяє брати випадкові вектори з нього та генерувати нові дані. Крім того, забезпечується можливість знаходити проміжні вектори між двома вибраними векторами за допомогою інтерполяції.

Особливості застосування автокодувальників для візуалізації даних:

– Згладжування шуму;

– Зменшення розмірності.

З урахуванням певних обмежень щодо розмірності та розрідженості, автокодувальники здатні досліджувати проекції даних, які є більш цікавими, ніж традиційні методи, такі як PCA (метод головних компонент) та інші основні техніки. Автокодувальники здійснюють навчання на наборах даних автоматично, що означає, що їхні складові можна легко навчити для ефективної роботи з конкретною топологією даних без потреби у впровадженні нових технік − достатньо просто мати відповідні дані для навчання.

Проте автокодувальники не завжди ефективні у стисненні зображень. Під час навчання на заданому наборі даних вони можуть досягати прийнятних результатів у стисненні даних, схожих на ті, що використовуються для навчання, але виникають проблеми, коли їх намагаються використовувати як загальні компресори. Наприклад, компресія JPEG демонструє значно кращі результати. Автокодувальники навчаються як зберігати інформацію, так і надавати нові властивості, використовуючи різні типи архітектур. Однак завжди існує компроміс між зміщенням і дисперсією. З одного боку, важливо мати декодер, який зможе якісно відновлювати вихідні дані, а з іншого − прагнуть зменшити латентний простір до мінімуму.

Типи авктодувальників:

– Класична реалізація автокодувальника;

– Багатошаровий автокодувальник;

– Роздріджений автокодувальник;

– Згортковий автокодувальник;

– Автодувальники для видалення шуму.

Генеративні змагальні мережі (GAN) є типами архітектури нейронної мережі здатні генерувати нові дані, що відповідає вивченим шаблонам. GAN можна використовувати для створення зображень людських облич чи інших об’єктів, для виконання перекладу тексту в зображення, для перетворення одного типу зображення в інший, а також для підвищення роздільної здатності зображень (суперроздільності) серед інших програм. Оскільки GAN можуть генерувати абсолютно нові дані, вони стоять на чолі багатьох передових систем ШІ, програм і досліджень [8].

GAN (генеративна змагальна мережа) складається з дискримінатора та генератора. Дискримінатор має завдання відрізнити реальні зразки від тих, що були згенеровані, тоді як генератор намагається імітувати реальні зразки, спираючись на інформацію, отриману від дискримінатора. GAN вирізняється серед багатьох інших генеративних моделей. Замість того щоб явно вибирати з розподілу ймовірностей, GAN використовує глибоку нейронну мережу як генератор, який створює зразки на основі випадкового шуму.

Дослідники з OpenAI вважають, що генеративні моделі є одним із найперспективніших підходів для наділення комп’ютерів здатністю розуміти навколишній світ. У 2018 році команда OpenAI створила генеративне попереднє навчання (GPT), яке є авторегресійною нейронною мовною моделлю (NLM). Ця модель була навчена на широкому спектрі немаркованого тексту та потім точно налаштована для виконання конкретних завдань, що забезпечило значне підвищення її продуктивності.

Основна ідея GAN полягає у визначенні гри між двома конкурентними мережами. Генератор перетворює джерело шуму в зразки у вхідному просторі, тоді як дискримінатор отримує або згенерований, або справжній зразок даних і має визначити, який з них є справжнім. Генератор тренується на те, щоб обманювати дискримінатора.

Концепція змагальності в GAN часто пояснюється через метафори. Наприклад, генеративну мережу можна порівняти з фальшивомонетником або

підроблювачем картин, тоді як дискримінатор виступає в ролі експерта, який намагається виявити підробку.

Незважаючи на значні успіхи в точності та швидкості завдяки використанню більш швидких і глибоких згорткових нейронних мереж, одна ключова проблема залишається невирішеною: як відновити деталі тонких текстур при значному масштабуванні зображень? Ефективність методів супер-роздільної здатності на основі оптимізації в основному визначається вибором цільової функції. Генеративні змагальні мережі (GAN) знаходять широке застосування у створенні різноманітного контенту, що включає не лише зображення для онлайн-магазинів та аватари для комп'ютерних ігор. Також вони здатні створювати віртуальних ведучих для телевізійних програм, що відкриває нові горизонти у світі медіа та розваг. Синтезовані дані, які генерують ці мережі, можуть служити основою для навчання інших систем, завдяки чому стає можливим покращення алгоритмів штучного інтелекту.

Крім того, генеративні змагальні мережі активно використовуються в автоматичному редагуванні, що вже знайшло своє місце в сучасних смартфонах і спеціалізованих програмах. Цей інноваційний підхід дозволяє вносити різноманітні зміни в фотографії: коригувати риси обличчя, зменшувати зморшки, змінювати зачіски, а також перетворювати денні зображення на нічні.

Не менш важливим є й застосування нейронних мереж GAN для генерації реалістичного відео міського середовища. Це особливо актуально при створенні фільмів, комп'ютерних ігор та віртуальної реальності, де важливо передати атмосферу і деталі візуального середовища.

Переваги та недоліки розглянутих технологій генерування зображень можна представити за допомогою таблиці 2.1.

Вибір генеративних змагальних мереж (GAN) для розробки був обґрунтований їхньою здатністю генерувати високу якість та реалістичні зображення, що є ключовими вимогами для досягнення успіху в проектах, пов’язаних з генерацією зображень. Незважаючи на певні виклики, пов'язані з процесом навчання, такі як нестабільність та ризик «mode collapse», GAN

демонструють значну адаптивність, що дозволяє їх застосовувати у різних сферах, від мистецтва до реалістичних візуалізацій.

Таблиця 2.1 – Переваги та недоліки технологій генерування зображень

Модель	Переваги	Недоліки
Генеративні змагальні мережі (GAN)	Висока якість створених зображень. Реалістичність та унікальність згенерованих зображень. Широка адаптивність до різних завдань	Нестабільність під час навчання. Можливе утворення «mode collapse» (модель генерує лише одне або кілька схожих зображень). Високі вимоги до обчислювальних ресурсів.
Автокодувальники (VAE)	Простота в реалізації та стабільність процесу навчання. Можливість контролювати генерацію через латентний простір. Швидка генерація зображень.	Менш деталізовані зображення порівняно з GAN. Нижча якість створених зображень.
Диференціальні перетворювачі (Diffusion Models)	Висока якість та деталізація згенерованих зображень. Стабільність навчання без проблем з «mode collapse». Ефективність при генеруванні зображень з великою кількістю деталей.	Довший процес генерації через ітеративність. Потреба у великих обчислювальних потужностях.

У порівнянні з альтернативними моделями, такими як автокодувальники (VAE) та диференціальні перетворювачі (Diffusion Models), GAN забезпечують кращу деталізацію та унікальність результатів. Це робить їх особливо придатними для проектів, де важлива висока якість зображень. Вибір GAN дозволяє використовувати їхні сильні сторони та досягати бажаних результатів у сфері генерації зображень, що робить їх оптимальним вибором для розробки

інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

2.2 Математична модель створення зображень за допомогою генеративної нейронної мережі

Математичні моделі відіграють ключову роль у сучасних наукових дослідженнях і технологічних розробках, оскільки вони забезпечують структурований підхід до аналізу складних систем та явищ. У їх основі лежить формалізація реальних процесів за допомогою математичних рівнянь та структур, що дозволяє зрозуміти, проаналізувати та передбачити поведінку об'єктів в умовах змінюваних параметрів. Це особливо важливо в епоху інформаційних технологій, коли зростає обсяг даних і ускладнюється середовище, в якому ці дані функціонують.

Одним із основних напрямків використання математичних моделей є аналіз, який дозволяє дослідникам і розробникам отримати чітке уявлення про те, як різні параметри впливають на результати експериментів чи системних процесів. Застосування математичних моделей у аналізі дозволяє виявити кореляції між змінними, з'ясувати причинно-наслідкові зв'язки та ідентифікувати критичні точки, на які варто звернути увагу при проведенні подальших досліджень або реалізації проектів.

Наприклад, у фінансових системах моделі можуть допомогти виявити, як зміни в облікових ставках впливають на економічне зростання або ринок акцій. Аналіз, що базується на математичних моделях, дозволяє формулювати обґрунтовані гіпотези, які згодом можуть бути перевірені експериментально або через спостереження.

Наступним важливим аспектом є передбачення. Математичні моделі можуть бути використані для прогнозування майбутніх подій або поведінки систем, що є надзвичайно важливим у багатьох сферах, таких як економіка, екологія та технології. Наприклад, у сфері екології моделі можуть бути

використані для прогнозування змін клімату, оцінки впливу забруднення на навколишнє середовище або управління природними ресурсами. У фінансових ринках моделі можуть передбачати коливання цін на акції, валюту чи інші активи, що дозволяє інвесторам приймати обґрунтовані рішення. В медицині моделі можуть прогнозувати поширення захворювань або ефективність лікування, що може врятувати життя.

Останнім, але не менш важливим аспектом є оптимізація. Використання математичних моделей дозволяє знаходити оптимальні рішення для складних задач, що зменшує витрати часу та ресурсів. Це особливо актуально в промисловості, де потрібно оптимізувати виробничі процеси, логістику, управління запасами та інші аспекти діяльності підприємства. Моделі можуть бути використані для визначення найбільш ефективного розподілу ресурсів, зменшення витрат на виробництво або підвищення продуктивності праці. Наприклад, у сфері транспорту моделі можуть оптимізувати маршрути доставки, що зменшує витрати на паливо та покращує обслуговування клієнтів. В аграрному секторі математичні моделі можуть допомогти у визначенні оптимальних умов для вирощування культур, що збільшує врожайність і зменшує використання ресурсів. Таким чином, оптимізація на основі математичних моделей сприяє підвищенню ефективності та конкурентоспроможності організацій у різних галузях.

Математична модель створення зображень за допомогою генеративної нейронної мережі (GAN) є складним процесом, що передбачає перетворення випадкових вхідних даних або певних параметрів у візуальні об'єкти. Основною моделлю, яка лежить в основі цього процесу, є Generative Adversarial Network (GAN). Ця модель складається з двох основних компонентів: генератора (G) та дискримінатора (D). Генератор відповідальний за створення нових зображень, тоді як дискримінатор оцінює їх реалістичність, намагаючись відрізнити згенеровані зображення від справжніх.

У контексті використання AI Horde як генеративної нейронної мережі, математична модель базується на архітектурі GAN та її специфічних

особливостях. AI Horde інтегрує різні генеративні моделі, що дозволяє комбінувати різноманітні підходи для створення зображень, що підвищує якість та різноманітність результатів.

Опишемо процес генерації зображення по крокам:

1) Вхідний простір: AI Horde приймає на вхід латентний вектор z або певний набір параметрів, що впливають на стиль і вміст зображення. Це може бути випадковий вектор, текстове описання або інші характеристики. Різноманіття вхідних даних є критично важливим для генерації зображень, оскільки воно впливає на креативність і різноманіття результатів. Розподіл латентних векторів або параметрів:

$$z \sim Pz(z), \tag{2.1}$$

де $Pz(z)$ − розподіл латентних векторів або параметрів.

2) Функція генерації: генеративний модуль AI Horde $\mathrm{G}(\mathrm{z};\theta_H)$ перетворює вхідний вектор z у зображення. Важливо зазначити, що генератор навчений на великій кількості зображень, тому він може відтворювати складні патерни та деталі. Ознайомимось з функцією генерації:

$$x_{fake} = \mathrm{G}(\mathrm{z};\theta_H), \tag{2.2}$$

де $\mathrm{G}(\mathrm{z};\theta_H)$ − генеративний модуль,

x_{fake} − згенероване зображення,

θ_H − параметри генеративної моделі.

3) Кластеризація моделей: Кластеризація передбачає розподіл об'єктів на раніше невідомі групи на основі подібності або близькості значень певних ознак [9]. У AI Horde кластеризація проявляється через розподіл запитів між вузлами з різними моделями, що дозволяє оптимально використовувати ресурси мережі та забезпечувати різноманітність результатів.

$$x_{fake} = \sum_{i=1}^{N} \alpha_i G_i(z; \theta_i)\,, \tag{2.3}$$

де G_i − генератори окремих нейронних мереж,

α_i − ваги, що визначають внесок кожної моделі в кінцевий результат.

4) Функція втрат: AI Horde оптимізує генерацію через спільну роботу різних моделей. Можна використовувати модифіковану функцію втрат, що дозволяє адаптувати ваги для кожної окремої моделі:

$$L_H = -E_z \sim P_z(z) \left[log \left(\sum_{i=1}^{N} \alpha_i D_i(\, G_i(z) \right) \right], \tag{2.4}$$

де D_i − дискримінатор, що оцінює реалістичність зображень, згенерованих моделями.

5) Алгоритм навчання: Навчання AI Horde здійснюється через ітеративну оптимізацію параметрів кожної моделі θ_i та ваг α_i. Кожна генеративна модель навчається окремо, але синергізує з іншими через спільний вихід [10].

6) Вибір моделі користувачем: AI Horde може дозволити користувачам обирати параметри для зображення або конкретні моделі, що впливає на кінцевий результат. Це додає ще один параметр β_i, який враховує користувацькі преференції:

$$x_{custom} = \sum_{i=1}^{N} \beta_i \alpha_i G_i(z; \theta_i), \tag{2.5}$$

де β_i − параметри, що задаються користувачем для налаштування зображення,

α_i − ваги моделей.

Таким чином, математична модель створення зображень за допомогою генеративної нейронної мережі AI Horde базується на принципах GAN та

використовує можливості кількох генеративних моделей. Це дозволяє ефективно генерувати високоякісні зображення відповідно до заданих параметрів, без необхідності змінювати базові алгоритми моделі.

2.3 Розробка UML-діаграми послідовності інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Для кращого розуміння роботи модуля створення зображень за допомогою нейронної мережі побудуємо UML-діаграму.

UML (англ. Unified Modeling Language) − уніфікована мова моделювання, яка широко використовується в розробці програмного забезпечення для опису, візуалізації, проектування й документування систем. Вона підтримує об'єктно-орієнтовані концепції і є стандартом для моделювання різноманітних аспектів системи за допомогою графічних позначень. UML допомагає створити абстрактне представлення системи, що сприяє кращому розумінню її структури та поведінки, а також взаємодії між компонентами [11].

UML складається з діаграм, які поділяються на структурні й поведінкові. Структурні діаграми, такі як діаграма класів (Class diagram), компонента (Component diagram) і діаграма розгортання (Deployment diagram), показують статичні аспекти системи: її структуру, класи, модулі та залежності між ними. Поведінкові діаграми зосереджуються на динамічних аспектах системи — її поведінці під час виконання.

Для моделювання модуля, пов'язаного з генеруванням зображень, особливо важливо зосередитися на поведінкових діаграмах, оскільки вони допомагають зрозуміти основні процеси й сценарії взаємодії між користувачем та системою.

Діаграма діяльності ілюструє процес роботи модуля, демонструючи потік дій від однієї операції до іншої, а також загальну структуру роботи системи. Це розширений варіант блок-схем, проте, на відміну від блок-схем, вона

використовує стандартизовану нотацію, яка дозволяє візуалізувати паралельні дії та синхронізацію.

Діаграма прецедентів показує, які ролі взаємодіють із системою, а також зв’язки між ними та системою, хоча й не вказує точну послідовність дій. Вона відображає функціональні вимоги (можливості системи) з точки зору користувача і може бути представлена текстово або у вигляді графічної діаграми.

Діаграма послідовності фокусується на відправленні та отриманні повідомлень об'єктами протягом часу, що показує послідовність дій у часі. Ця діаграма обрана для представлення повного циклу роботи модуля генерації зображень, оскільки саме вона ілюструє, як елементи системи взаємодіють у процесі виконання завдання.

Діаграма послідовності включатиме компоненти, що описують цикл роботи програмного модуля: WEB-клієнт, сервер, постачальник даних і база даних. Також буде додано об’єкт користувача для відображення його взаємодії з іншими модулями програми.

Розроблену діаграму послідовностей наведено на рисунку 2.1.

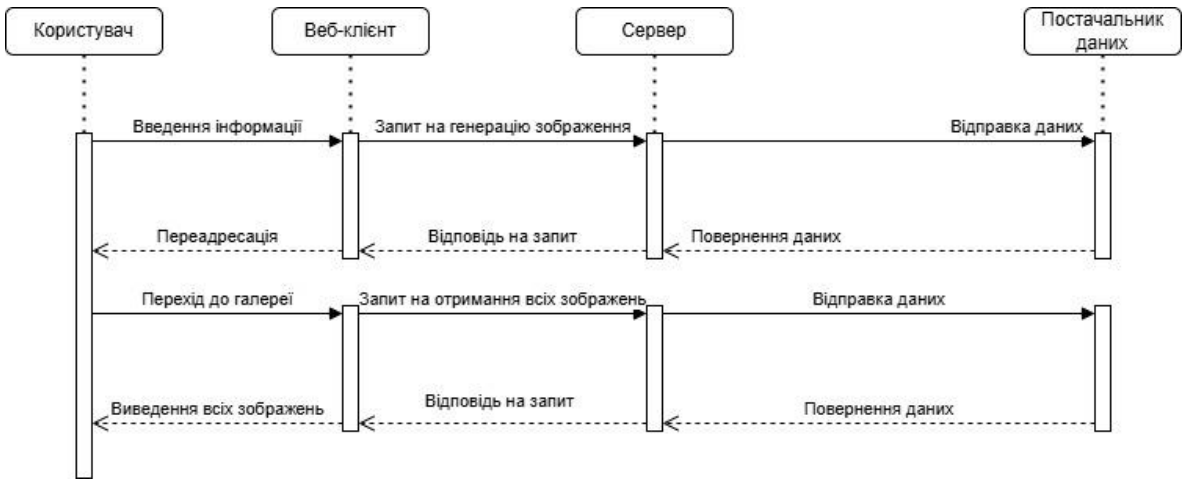

Рисунок 2.1 – UML-діаграма послідовності взаємодії головних модулів інформаційної технології створення зображень за допомогою генеративної нейронної мережі

2.4 Висновок до розділу 2

У другому розділі було детально розглянуто основні аспекти створення зображень за допомогою генеративних нейронних мереж (GAN). Спочатку обґрунтовано вибір методу GAN, який виявився оптимальним для реалізації проекту завдяки своїй здатності генерувати високоякісні зображення з вражаючою деталізацією та стилістичною різноманітністю. Використання GAN дозволяє не лише створювати нові зображення, а й вивчати властивості існуючих, що робить цей метод надзвичайно потужним у художній генерації.

Удосконалено математичну модель, яка лежить в основі роботи GAN. Ця модель описує ключові етапи навчання мережі, включаючи функції втрат та алгоритми оптимізації. Розуміння цих математичних основ є критично важливим для подальшої розробки, оскільки вони забезпечують основу для вдосконалення роботи мережі та покращення якості згенерованих зображень.

Розроблена UML-діаграма послідовності ілюструє процес генерації зображень і демонструє взаємодію різних компонентів системи. Ця діаграма є важливим інструментом для візуалізації робочих процесів, що сприяє кращому розумінню архітектури проекту.

Таким чином, детальний аналіз вибору методу, опис математичної моделі та UML-діаграми послідовності формує чітке уявлення про підходи до реалізації технології генерації зображень, це основа для наступних етапів розробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЕНЕРАТИВНОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Обґрунтування вибору архітектури інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Архітектура програмного забезпечення − це структура програми, системи чи модуля, що визначає взаємодію між її елементами, їхні властивості та розподіл функціональних обов'язків. Проектування архітектури є першим кроком у розробці програмного забезпечення [11].

При розробці модуля, орієнтованого на технічні вимоги, важливо створити оптимальну архітектуру, враховуючи продуктивність майбутнього проєкту, взаємозв'язки між основними компонентами системи та забезпечуючи можливість її масштабування. Архітектура програмного забезпечення встановлює обмеження для написання коду, визначає єдиний стиль розробки та закріплює ключові рішення, прийняті на початкових етапах розробки [12].

Технологія створення зображень на основі генеративної нейронної мережі використовує клієнт-серверну архітектуру, що є ефективним способом обробки даних. У цій архітектурі сервер виступає як потужний обчислювальний ресурс, призначений для зберігання та обробки інформації, а також забезпечення передачі даних між користувачами. Це дозволяє масштабувати систему та працювати з великими обсягами інформації.

Клієнтська частина системи відповідає за прийом та обробку інформації, введеної користувачем, а також за візуалізацію існуючих даних. Принцип роботи такої архітектури полягає в тому, що клієнт здійснює безпосередню взаємодію з користувачем через зручний та інтуїтивно зрозумілий інтерфейс. Клієнтська програма збирає необхідну інформацію від користувача, обробляє та, за потреби, модифікує її, перш ніж відправити на сервер. Для цього вона використовує популярні протоколи передачі даних, такі як HTTP або TCP.

Протокол HTTP, який широко використовується для передачі даних в Інтернеті, дозволяє клієнту надсилати запити на сервер і отримувати відповіді. Водночас, протокол TCP має суттєву перевагу − він забезпечує створення постійного з’єднання між клієнтом і сервером. Це дозволяє здійснювати динамічну передачу змін у даних, що робить систему більш ефективною та інтерактивною. У випадку, коли дані в одному з елементів системи змінюються, зміни можуть бути миттєво передані іншим елементам.

Сервер виконує ключову роль у цій архітектурі, беручи на себе відповідальність за обробку та збереження інформації, надісланої клієнтом. Він не тільки зберігає дані, а й здійснює їх аналіз, обробку та підготовку до подальшої передачі назад до клієнта.

Сервер здатен обслуговувати кількох клієнтів одночасно, а також система може включати кілька серверів, кожен із яких виконує різні завдання залежно від архітектури і взаємодіє між собою через протоколи HTTP або TCP. Сервер має доступ до бази даних системи, відповідає за контроль доступу до неї та здійснює операції з даними. В архітектурі можна визначити пріоритети запитів до сервера, завдяки чому критично важливі для функціонування системи завдання можуть оброблятися першочергово [13]. На рисунку 3.1 подано схему алгоритму роботи клієнт-серверної архітектури.

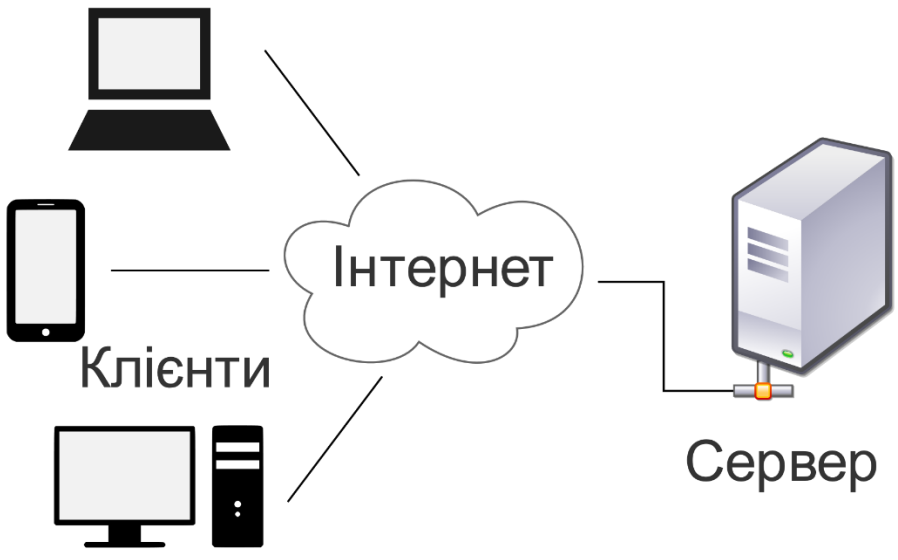

Рисунок 3.1 – Схема алгоритму роботи клієнт-серверної архітектури

Функції сервера:

– зберігання, доступ, захист і резервне копіювання даних. Сервер відповідає за централізоване зберігання інформації, що дозволяє користувачам легко отримувати доступ до необхідних даних. Забезпечення їхньої безпеки є пріоритетом, тому сервер реалізує заходи захисту даних, включаючи шифрування та контроль доступу. Крім того, резервне копіювання інформації гарантує, що дані не будуть втрачені в разі збоїв або інших непередбачуваних обставин;

– обробка клієнтського запиту. Сервер отримує запити від клієнтів, аналізує їх і виконує відповідні дії. Це може включати виконання обчислень, доступ до бази даних або підготовку даних для відправлення назад клієнту. Швидкість та ефективність обробки запитів є критично важливими для забезпечення високої продуктивності системи;

– відправлення результату (відповіді) клієнту. Після завершення обробки запиту сервер формує відповідь та надсилає її назад до клієнта. Цей процес є важливим для зворотного зв’язку, що дозволяє клієнту отримувати актуальну інформацію та дані, необхідні для подальшої роботи;

– моніторинг та управління системними ресурсами. Сервер виконує функції моніторингу, контролюючи використання ресурсів, таких як пам'ять, процесор та дисковий простір. Це допомагає забезпечити стабільну роботу системи та уникнути перевантажень;

– логування та обробка помилок. Сервер веде журнали (логи) всіх запитів і відповідей, а також фіксує помилки, які виникли. Це дозволяє системі аналізувати та діагностувати проблеми, що можуть виникати під час роботи, і забезпечує можливість їх виправлення;

– аутентифікація та авторизація користувачів. Сервер відповідає за перевірку особи користувача, що намагається отримати доступ до системи, а також за визначення його прав на виконання певних дій. Це є критично важливим для забезпечення безпеки даних.

Функції клієнта:

– надання користувальницького інтерфейсу. Клієнт відповідає за створення зручного та інтуїтивно зрозумілого інтерфейсу, який дозволяє користувачам взаємодіяти з системою. Інтерфейс має бути адаптованим до потреб користувача, щоб забезпечити легкість використання та доступність всіх необхідних функцій;

– формулювання запиту до сервера і його відправка. Клієнт створює запит на основі дій користувача, формулює його у зрозумілому для сервера форматі і надсилає через обраний протокол зв’язку. Цей етап є критично важливим, оскільки від точності формулювання запиту залежить успішність отримання необхідної інформації;

– отримання результатів запиту і відправка додаткових команд. Після отримання відповіді від сервера клієнт обробляє результати запиту та виводить їх на екран користувача. У разі необхідності клієнт може надсилати додаткові команди, наприклад, запити на додавання, оновлення або видалення даних, що забезпечує динамічну взаємодію з сервером та актуальність інформації;

– валідація введених даних. Клієнт відповідає за перевірку введених даних перед їх відправкою на сервер. Це допомагає уникнути надходження некоректної інформації, що може призвести до помилок на сервері;

– забезпечення відображення повідомлень про помилки та успішні операції. Клієнтська частина системи має надавати користувачеві інформацію про статус виконання запитів, включаючи повідомлення про помилки або підтвердження успішного завершення операцій;

– збереження налаштувань користувача. Клієнт може зберігати персоналізовані налаштування користувача, такі як переваги у відображенні інформації, теми оформлення тощо, що робить досвід користування більш зручним та індивідуалізованим.

3.2 Проектування структури та розробка алгоритму функціонування інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Модульний підхід дозволяє розробникам структурувати програмне забезпечення на окремі, чітко визначені частини − модулі, кожен з яких має свою конкретну функціональність. Це забезпечує можливість легкого тестування, оновлення та підтримки кожного модуля окремо, що значно полегшує процес розробки, особливо у великих проєктах. Завдяки такому підходу можна мінімізувати помилки та підвищити гнучкість системи, дозволяючи масштабувати її та додавати нові функції [14].

Програмний модуль включатиме в себе такі компоненти:

1. Модуль навігації;
2. Модуль генерації картинок;
3. Модуль перегляду згенерованих картинок;
4. Модуль опцій.

Модуль навігації забезпечить користувачеві зручний та зрозумілий доступ до всіх доступних функцій системи. Завдяки інтерактивному меню, користувач зможе легко переходити між різними модулями та отримувати доступ до необхідних інструментів для виконання своїх завдань. Це меню буде простим у використанні і допоможе орієнтуватися в системі, що підвищить загальну ефективність роботи користувача.

Модуль генерації картинок виконує важливу функцію, надсилаючи запити до штучного інтелекту для створення унікальних зображень. Користувач матиме можливість вказувати певні параметри або теми, за якими буде виконуватись генерація, це дозволить створювати зображення, які відповідають їхнім вимогам. Завдяки цьому модулю, користувачі зможуть швидко отримувати візуальний контент, необхідний для їхніх проектів.

Модуль перегляду згенерованих картинок надасть можливість користувачам з легкістю переглядати всі раніше згенеровані зображення. Це

стане в нагоді для тих, хто хоче швидко знайти та повторно використовувати вже створений контент. Модуль буде оснащений функціями фільтрації та пошуку, щоб користувач міг швидко знаходити потрібні зображення за ключовими словами, датами або іншими критеріями.

Модуль опцій надасть користувачам можливість змінювати характеристики картинок, які вони очікують отримати. Цей модуль дозволить налаштовувати різні параметри, такі як розмір, стиль, колірна гамма та інші аспекти, що впливають на кінцевий вигляд зображення. Гнучкість цього модуля забезпечить користувачам можливість отримувати результати, які найкраще відповідають їхнім потребам і перевагам.

Структурна схема програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі подана на рисунку 3.2.

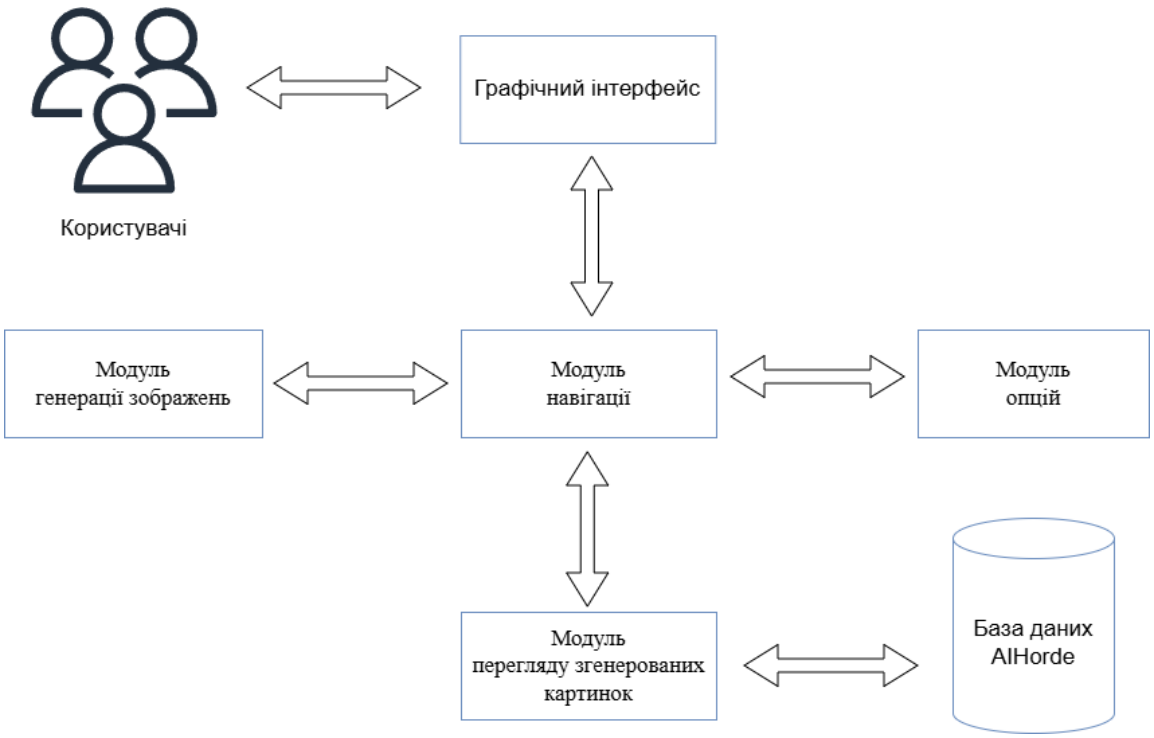

Рисунок 3.2 – Структурна схема програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Алгоритм функціонування інформаційної технології створення зображень за допомогою генеративної нейронної мережі є важливим елементом в розробці програмного забезпечення, оскільки він допоможе визначити послідовність процесів. Алгоритми, що представляють собою чіткі набори інструкцій для вирішення різних задач, спрощують виконання навіть найскладніших операцій, розбиваючи їх на зрозумілі етапи. Вони забезпечують ефективність і надійність роботи системи, адже дозволяють уникнути помилок та оптимізувати процеси.

Розглянемо основні властивості алгоритмів:

1. Зрозумілість. Кожна інструкція має бути доступною та зрозумілою.

2. Визначеність. Кожна дія в алгоритмі має бути точно описана та не викликати сумнівів.

3. Дискретність. Алгоритм має складатися з чітко окреслених кроків, які виконуються в установленому порядку.

4. Універсальність. Алгоритм може бути застосований для вирішення схожих завдань.

5. Ефективність. Після завершення алгоритму має бути конкретний результат.

Блок-схеми, в свою чергу, є графічними представленнями алгоритмів, які полегшують сприйняття логіки та послідовності дій. Використання блок-схем дозволяє візуально уявити структуру алгоритму та взаємозв’язки між його частинами. Це особливо корисно в командній роботі, адже візуалізації допомагають з легкістю пояснити ідеї та концепції іншим учасникам проекту.

Таким чином, алгоритми та блок-схеми спрощують процес розробки, забезпечуючи уніфікацію підходів, документування виконаних дій, а також сприяють співпраці між розробниками. Вони дозволяють не лише зрозуміти, як працює система, але й аналізувати алгоритми для пошуку ефективніших рішень, що є ключовим для успішної реалізації програмних проектів.

Розглянемо алгоритм, який реалізує інформаційну технологію створення зображень за допомогою генеративної нейронної мережі (рис. 3.3):

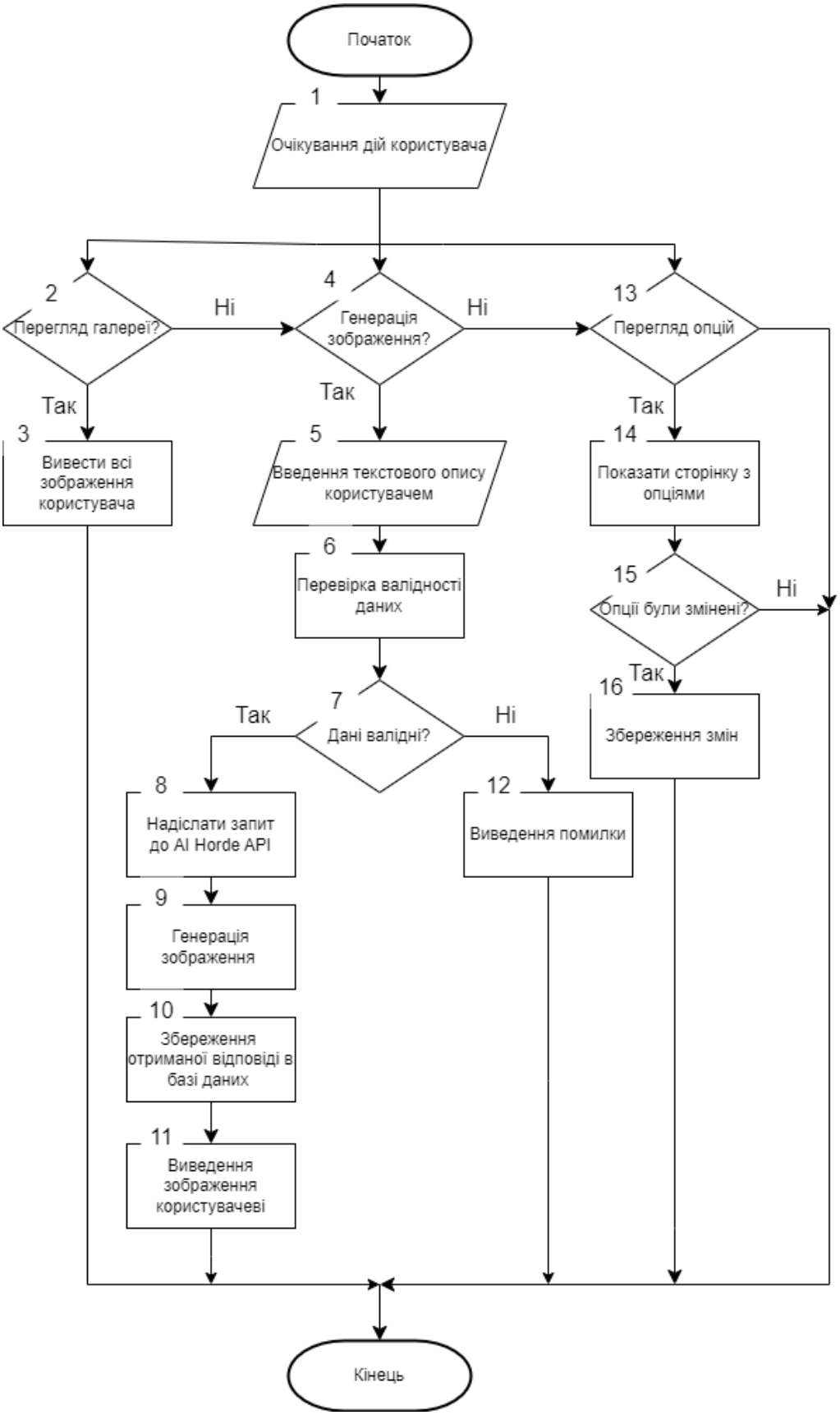

Рисунок 3.3 – Схема алгоритму роботи інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Опишемо наведений алгоритм роботи інформаційної технології створення зображень за допомогою генеративної нейронної мережі (рис. 3.3).

1. Очікування дій користувача.

2. Якщо користувач обрав перегляд галереї перейдемо до пункту 3.

3. Виведення всіх зображень, що були раніше згенеровані користувачем.

4. Якщо користувач обрав генерацію зображення перейдемо до пункту 5.

5. Користувач вводить текстовий опис зображення, яке він бажає згенерувати.

6. Введені користувачем дані перевіряються на валідність.

7. Якщо після перевірки даних вони виявилися не валідними, наприклад користувач ввів пустий рядок, відбувається перехід на крок 12.

8. Надсилається запит на AI Horde Api.

9. AI Horde Api обробляє запит та генерує зображення.

10. Зображення та інформація про нього, яку повертає AI Horde Api зберігається в базі даних.

11. Зображення виводиться користувачеві й перехід до кроку 17.

12. Виведення помилки й перехід до кроку 17.

13. Перевірка чи користувач обрав перегляд опцій, якщо ні то переходимо на крок 17.

14. Виведення сторінки опцій.

15. Якщо користувач вніс зміни до опцій переходимо на крок 16, якщо ні, то на крок 17.

16. Зберігаємо нові значення для опцій в базі даних

17. Завершення алгоритму

3.3 Обґрунтування вибору мови програмування

Обґрунтування вибору мови програмування є важливим етапом у розробці будь-якого програмного продукту, зокрема у проекті. Цей етап визначає не лише якість та ефективність кінцевого продукту, але і загальний процес розробки.

Python є однією з найпопулярніших мов програмування у світі завдяки своїй простоті, читаємості та потужним можливостям для реалізації різноманітних завдань, включаючи WEB-розробку [15]. Ця мова високого рівня має чітку синтаксичну структуру, що робить її доступною для новачків, але при цьому вона залишається потужним інструментом для досвідчених розробників.

Переваги мови Python:

– Простота – лаконічний та інтуїтивно зрозумілий синтаксис робить Python доступним для новачків і дозволяє швидко писати зрозумілий код.

– Крос-платформеність – Python працює на різних операційних системах (Windows, macOS, Linux), що дозволяє легко переносити програми між платформами.

– Багатофункціональність – завдяки великій кількості стандартних бібліотек та розширень Python підходить для розв'язання різноманітних задач, від обробки даних до машинного навчання.

– Інтеграція з іншими мовами – Python легко взаємодіє з кодом, написаним іншими мовами, що розширює його можливості.

– Інтерактивний режим розробки – можливість виконувати швидко код та одразу бачити результат, це дає змогу ефективно тестувати та налагоджувати програми.

Недоліки мови Python:

– Швидкодія – Python може бути повільнішим порівняно з компільованими мовами, особливо для великих обчислювальних задач.

– Обмежена підтримка мобільних застосунків – Python не є основною мовою для розробки мобільних додатків, порівняно з такими мовами як Swift чи Kotlin.

– Контроль над пам’яттю – автоматичне управління пам'яттю може призводити до невеликого витоку пам'яті в деяких випадках.

– Обмежена підтримка ООП – незважаючи на те, що Python підтримує об’єктно-орієнтоване програмування, його реалізація може бути менш гнучкою порівняно з іншими мовами.

Однією з основних причин вибору Python для WEB-розробки є потужні фреймворки, такі як Django і Flask, які значно спрощують процес створення WEB-застосунків. Django, зокрема, є повноцінним фреймворком, що включає в себе все необхідне для створення сучасних WEB -ресурсів, включаючи систему адміністрування, вбудовану підтримку роботи з базами даних, а також інструменти для забезпечення безпеки. Flask, в свою чергу, є легким мікрофреймворком, що дає більше свободи для розробника, дозволяючи вибирати окремі компоненти для реалізації конкретних задач.

Python також широко використовується для роботи з великими даними, аналітики та машинного навчання. Завдяки таким бібліотекам, як NumPy, Pandas та TensorFlow, Python є ідеальним вибором для розробки додатків, що потребують обробки великих обсягів даних або інтеграції з технологіями штучного інтелекту. Це робить Python вкрай зручним для розробки додатків, які використовують генеративні нейронні мережі [16].

Простота синтаксису Python дозволяє швидко розпочати розробку і забезпечує високу швидкість написання коду, що є важливим аспектом при розробці стартапів або прототипів продуктів. Крім того, Python має велику кількість бібліотек і пакетів, що дозволяє легко реалізувати різноманітні функції, від роботи з WEB-сервісами до обробки зображень або відео.

Вибір Python для WEB-розробки також має перевагу завдяки його кросплатформеності. Python працює на різних операційних системах, що дозволяє створювати додатки, які можуть бути використані як на серверній, так і на клієнтській стороні без суттєвих змін у коді.

Загалом, Python є гарним вибором для розробки WEB-ресурсів завдяки своїй простоті, потужним фреймворкам та широким можливостям для інтеграції з сучасними технологіями, такими як штучний інтелект та обробка даних. Ця

мова дозволяє ефективно вирішувати різноманітні завдання та створювати високоякісні, масштабовані WEB-додатки.

Go, також відомий як Golang − це мова програмування, створена компанією Google з метою забезпечити простоту і ефективність при розробці великих і високопродуктивних додатків. Завдяки своїй простоті, швидкості виконання та потужним можливостям для роботи з багатозадачністю, Go є гарним вибором для створення WEB-ресурсів, зокрема у високонавантажених системах [17].

Переваги Go:

– Легкість в освоєнні – мова проста для навчання, але має високу швидкість і потужність, що робить її ідеальною для мережевих продуктів.

– Багата бібліотека – містить численні бібліотеки для роботи в різних галузях.

– Простий синтаксис – синтаксис Go є інтуїтивно зрозумілим і зручним для програмістів.

– Статична типізація – гарантує високу безпеку, зменшуючи можливість помилок під час компіляції.

– Сумісність з іншими мовами – дозволяє легко взаємодіяти з іншими мовами програмування.

– Автоматизація підтримується – ідеально підходить для застосувань у штучному інтелекті та науці про інформацію.

– Гарно підходить для створення односторінкових WEB-додатків.

Недоліки Go:

– Великі витрати часу на повторне написання коду – на переробку існуючого коду може знадобитись більше часу.

– Молода екосистема – мова ще не має такої великої спільноти та ресурсів, як більш зрілі мови.

– Відсутність дженериків – не підтримує узагальнення, що може ускладнити написання універсальних функцій.

– Відсутність віртуальної машини – не має вбудованої віртуальної машини для виконання коду.

Go має вбудовану підтримку для роботи з мережами, що робить її гарним вибором для розробки сервісів, що працюють із запитами HTTP, зокрема REST API або WebSocket-сервіси. Бібліотеки Go забезпечують високу ефективність обробки запитів і дозволяють швидко розробляти високопродуктивні WEB-системи.

Однією з особливостей Go є її простий і зрозумілий синтаксис, що дозволяє швидко освоїти мову навіть тим розробникам, які мають досвід роботи з іншими мовами. Go також забезпечує високий рівень безпеки і стабільності завдяки статичній типізації та чітким правилам для управління пам'яттю.

Загалом, Go є належним вибором для розробки високопродуктивних, масштабованих WEB-ресурсів, зокрема для додатків, які потребують обробки великої кількості запитів або одночасних з'єднань. Її простота, швидкість і ефективність обробки паралельних задач роблять Go одним з найкращих інструментів для розробки сучасних WEB-додатків.

Розглянемо також JavaScript (рис. 3.4), вона є динамічною, високорівневою, інтерпретованою мовою програмування, що використовується для створення WEB-ресурсів і розширення функціональності WEB-сторінок [18].

Рисунок 3.4 – Логотип мови програмування JavaScript

JavaScript є потужним та гнучким інструментом для розробки WEB-ресурсів, і його основні особливості та переваги значно сприяють цьому. Однією з ключових характеристик JavaScript є його кросплатформеність, яка дозволяє працювати як на клієнтській стороні (у браузері), так і на серверній (в середовищі Node.js). Це робить JavaScript універсальним інструментом, який підтримує розробку як фронтенду, так і бекенду. Така інтеграція спрощує WEB-розробку, оскільки дозволяє використовувати одну й ту ж мову для створення різних компонентів додатка, що зменшує кількість необхідних знань і полегшує взаємодію між системами.

Динамічність JavaScript також є важливою перевагою. Як динамічна мова, вона дозволяє автоматично визначати типи змінних під час виконання програми, що забезпечує гнучкість у роботі з даними. Це спрощує обробку різноманітних даних та дозволяє швидко реагувати на зміни у вимогах або умовах роботи програми, що особливо корисно у швидкозмінних проектах [19].

Крім того, JavaScript підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам створювати класи та об'єкти, а також використовувати наслідування. Це значно спрощує організацію та структурування коду, що робить його більш зрозумілим і легким для підтримки. ООП також сприяє повторному використанню коду, що зменшує час на розробку та тестування.

Асинхронність − ще одна важлива характеристика JavaScript. Ця можливість дозволяє виконувати операції без блокування основного потоку виконання програми, що підвищує продуктивність додатків, особливо під час взаємодії з сервером або при обробці великих обсягів даних. Асинхронне програмування дозволяє створювати більш чутливі до дій користувачів WEB-додатки, оскільки вони можуть продовжувати виконувати інші задачі, поки чекають завершення певних операцій.

Загалом, завдяки своїм численним перевагам, JavaScript є вигідним вибором для реалізації WEB-ресурсів, зокрема для проектів, що включають генерацію зображень за допомогою генеративних нейронних мереж. Він

пропонує зручність інтеграції, гнучкість у роботі з даними та можливість створення високопродуктивних додатків, що відповідають сучасним вимогам WEB-розробки. Ці фактори роблять JavaScript незамінним інструментом в арсеналі кожного WEB-розробника.

Є кілька важливих причин, чому варто розглянути використання JavaScript для створення WEB-ресурсу:

1. Широка підтримка: JavaScript − це одна з найвідоміших мов програмування, яка підтримується всіма основними WEB-браузерами, такими як Chrome, Firefox, Safari та Edge. Це означає, що ваш WEB-ресурс зможе працювати на різних пристроях − комп'ютерах, ноутбуках, планшетах та смартфонах. Таке охоплення дає змогу залучити більшу кількість користувачів, що є дуже важливим для успіху вашого проекту.

2. Велика спільнота розробників: JavaScript має величезну та активну спільноту розробників. Це означає, що ви зможете знайти безліч ресурсів, включаючи документацію, відеоуроки, форуми та приклади коду. Якщо у вас виникнуть проблеми під час розробки, ви завжди зможете отримати допомогу від досвідчених програмістів. Спільнота постійно ділиться своїм досвідом, що дозволяє швидше вирішувати питання та покращувати навички.

3. Легкість використання: JavaScript має зрозумілий і простий синтаксис, що робить його ідеальним для новачків у програмуванні. Ви можете швидко навчитися основам і почати розробку, не витрачаючи багато часу на вивчення. Крім того, багато онлайн-курсів та посібників допоможуть вам швидко піднятися на наступний рівень. Це також дозволяє досвідченим розробникам економити час на реалізації нових ідей.

4. Розширюваність: JavaScript має велику кількість сторонніх бібліотек та інструментів, які можна використовувати для покращення функціональності вашого WEB-ресурсу. Наприклад, ви можете скористатися бібліотеками для обробки зображень, аналізу даних або створення графіки. Це дозволяє вам легко додавати нові можливості до вашого проекту, роблячи його більш привабливим та корисним для користувачів.

5. Відповідність сучасним стандартам: JavaScript постійно розвивається та адаптується до нових вимог. Зараз існують багато нових технологій, таких як фреймворки (наприклад, Vue.js, React, Angular), які полегшують розробку складних додатків. Це означає, що використовуючи JavaScript, ви завжди будете на крок попереду, зможете використовувати сучасні підходи та забезпечувати високу якість вашого продукту [20].

Інтерактивність: JavaScript дозволяє створювати інтерактивні елементи на WEB-сторінках, такі як анімації, спливаючі вікна та динамічні списки. Це підвищує залученість користувачів і покращує загальний досвід взаємодії з сайтом. Наприклад, ви можете реалізувати функцію, яка дозволяє користувачам взаємодіяти з вмістом сторінки без її перезавантаження, що робить використання WEB-ресурсу більш зручним.

3.4 Обгрунтування вибору фрейморків та бібліотек для розробки

Проведемо порівняльну характеристику найпопулярніших фреймоворків: Vue JS, Angular та бібліотеки React для реалізації інформаційної технології.

Angular − це відкритий JavaScript-фреймворк з модульною архітектурою, розроблений компанією Google. Його основна мета полягає в запровадженні архітектурних обмежень, яких не має у звичайному JavaScript, а також у впровадженні суворої типізації через використання TypeScript (JavaScript з підтримкою об’єктно-орієнтованого програмування). Це заохочує розробників використовувати об’єктно-орієнтований підхід, хоча можливий також функціональний стиль.

Angular створений для спрощення процесу розробки та управління JavaScript-додатками. Фреймворк побудований на основі архітектури MVC і особливо корисний для створення односторінкових WEB-додатків. Завдяки тому, що він базується на звичайному JavaScript і HTML, Angular автоматично обробляє маніпуляції з DOM і AJAX-запити, це зазвичай вимагає ручної реалізації з боку розробників [21].

Angular надає модульні компоненти JavaScript-коду, які можна легко комбінувати та тестувати, що дозволяє швидко інтегрувати їх на будь-яку HTML-сторінку за допомогою простих тегів. Фреймворк забезпечує ефективний контроль над даними на стороні клієнта, звільняючи розробників від необхідності вручну вибирати елементи з DOM і впроваджувати в них значення. Завдяки зручним прив'язкам даних зміни, внесені в JavaScript, автоматично відображаються в HTML-частині, і навпаки: якщо користувач змінює щось у HTML, JavaScript отримує нові значення.

Крім того, фреймворк має потужну підтримку ін’єкцій залежностей, що включає готові класи для виконання AJAX-запитів та управління маршрутами [22]. Розглянемо переваги використання фреймворку Angular:

– Є повноцінним фреймворком, що має все необхідне для написання сучасних WEB-застосунків без завантаження додаткових пакетів.

– Компонентний підхід. Цей підхід дозволяє розробку додатку у якості великої кількості структурних одиниць – компонентів. Використання компонентного підходу дозволяє створити додатковий рівень абстракції. З практичної сторони, компонентний підхід дозволяє створити універсальні компоненти, які будуть перевикористовуватись в різних місцях додатку, що пришвидшує розробку та впровадження нового функціоналу.

– Модульний підхід. Завдяки наявності модулів програма чудово масштабується, якщо попередньо чітко планувати структуру застосунку й слідувати їй.

– Висока швидкість роботи при великій кількості функціоналу. Використання Angular дозволяє створювати комплексні рішення з великою кількістю функціоналу та має високу потужність при роботі з високими навантаженнями.

– Строга типізація. Використання TypeScript дозволяє розробнику краще описувати код, полегшує та покращує підтримку і збільшує безпеку написаного коду.

Недоліки Angular:

– Високий поріг входу. Складність структури Angular вимагає від розробників значних знань і навичок, що створює круту криву навчання. Це, в свою чергу, призводить до підвищення складності використання порівняно з іншими фреймворками та бібліотеками.

– Висока завантаженість. Angular постачається з численними інструментами для розробки, з яких деякі виявляються рідко затребуваними або мають кращі аналоги. Наявність непотрібних модулів може негативно вплинути на продуктивність системи. У контексті розробки невеликих проектів Angular демонструє обмежену ефективність.

– Низька швидкість роботи. Продуктивність Angular може бути легко знижена, але з правильним розумінням механізму виявлення змін (Change Detection) та оптимізації швидкості роботи та завантаження WEB-додатків, можна досягти конкурентоспроможних показників продуктивності.

React.js – це фреймворк і бібліотека JavaScript з відкритим вихідним кодом, розроблені та підтримувані компаніями Facebook і Instagram. Він призначений для швидкого та ефективного створення інтерактивних користувацьких інтерфейсів і WEB-додатків, використовуючи значно менше коду в порівнянні зі звичайним JavaScript. У React розробка застосунків базується на створенні повторно використовуваних компонентів, які можна уявити як незалежні блоки конструктора Lego. Ці компоненти є окремими елементами остаточного інтерфейсу, які разом формують весь користувацький інтерфейс програми.

Основною функцією React у додатку є управління відображенням, аналогічно букві V у шаблоні «модель-представлення-контролер» (MVC), що забезпечує оптимізацію та ефективність рендерингу. Замість того, щоб розглядати весь користувацький інтерфейс як єдине ціле, React.js пропонує розробникам розподілити складні інтерфейси на окремі повторно використовувані компоненти, які становлять будівельні блоки всього інтерфейсу. При цьому React поєднує швидкість та ефективність JavaScript з

більш досконалим методом маніпулювання DOM, що дозволяє швидко рендерити WEB-сторінки та створювати високодинамічні WEB-додатки [23].

Переваги React.js:

1. Компонентний підхід: дозволяє створювати повторно використовувані компоненти, що полегшує управління великими проектами.

2. Швидкість рендерингу: використання віртуального DOM забезпечує високу продуктивність, зменшуючи витрати на оновлення реального DOM.

3. Гнучкість: React можна інтегрувати з іншими бібліотеками та фреймворками, що дозволяє використовувати його в різних архітектурах.

4. Широка екосистема: Багато доступних плагінів, бібліотек та інструментів для розширення функціональності.

5. Активна спільнота: велика кількість ресурсів, документування та підтримка з боку спільноти та компанії Facebook.

6. Простота у навчанні: концепції, такі як JSX, полегшують розуміння, навіть для новачків у JavaScript.

7. Підтримка TypeScript: можливість використовувати TypeScript для статичної типізації, що покращує безпеку коду.

Недоліки React.js:

1. Високий поріг входу: нові користувачі можуть зіткнутися з труднощами через необхідність вивчення додаткових концепцій, таких як JSX, пропси та стан.

2. Складність налаштування: хоча базова установка може бути простою, великі проекти часто вимагають налаштування складних систем для управління станом, маршрутизації тощо.

3. Велика бібліотека: потребує завантаження численних бібліотек та інструментів, що може ускладнити початкову конфігурацію.

4. Можливі проблеми з продуктивністю: неправильне використання компонентів може призвести до погіршення продуктивності через часті оновлення DOM.

5. Часті зміни: швидкий розвиток технологій і часті оновлення можуть призвести до необхідності постійного навчання та адаптації.

6. Керування станом: ускладнене управління глобальним станом, яке може вимагати використання додаткових бібліотек, таких як Redux або MobX.

7. SEO обмеження: погіршена підтримка SEO для односторінкових застосунків без належної оптимізації [24].

Vue.js − це фреймворк, призначений для розробки WEB-додатків. На відміну від інших відомих бібліотек та фреймворків, Vue.js не має корпоративного походження, а розробка його почалася як стартап і згодом трансформувалася в самостійний проект. Vue.js забезпечує можливість створення односторінкових додатків (SPA) і може бути використаний як бібліотека для розширення функціоналу вже існуючих рішень або як повноцінний фреймворк для розробки нових застосунків [25].

Vue.js, як прогресивний фреймворк для створення інтерфейсів користувача, надає розробникам широкі можливості для створення динамічних та інтерактивних WEB-ресурсів (рис. 3.5).

Рисунок 3.5 – Логотип фреймворку Vue.js

Однією з основних переваг Vue.js є його простота у вивченні та використанні. Завдяки зрозумілому синтаксису, новачки можуть швидко освоїти основи фреймворка, а досвідчені розробники отримують можливість ефективно реалізовувати складні функціональні можливості.

Також, Vue.js підтримує концепцію компонентного підходу, що дозволяє розробникам розділяти код на менші, автономні частини − компоненти. Це значно спрощує управління проектом, полегшує його тестування та підтримку.

Кожен компонент може містити свій власний шаблон, стилі та логіку, що дозволяє створювати модульні та реюзабельні елементи інтерфейсу [26].

Фреймворк дозволяє автоматично оновлювати інтерфейс користувача у відповідь на зміни даних. Це забезпечує високий рівень інтерактивності, що особливо важливо для сучасних WEB-ресурсів, які мають забезпечувати позитивний досвід користувача.

Vue.js має розгалужену екосистему, що включає в себе безліч бібліотек та інструментів, таких як Vue Router для маршрутизації та Vuex для управління станом. Це дає змогу розробникам легко інтегрувати різні компоненти та додаткові функції, що підвищує продуктивність процесу розробки. Наприклад, при роботі з великими даними або складними формами, Vuex може значно спростити управління станом додатка.

Також варто зазначити, що Vue.js активно підтримується спільнотою, що робить його популярним вибором серед розробників. Наявність безлічі ресурсів, таких як документація, форуми, онлайн-курси та приклади проектів, дозволяє легко отримати необхідну інформацію та допомогу в разі виникнення труднощів.

Отже, вибір мови програмування для розробки з використанням Vue.js базується на врахуванні цих факторів, а також специфічних вимог проекту. Важливо провести всебічний аналіз можливостей, щоб забезпечити максимальну ефективність та якість реалізації проекту. Тільки таким чином можна досягти оптимального результату і створити продукт, що відповідає сучасним вимогам та очікуванням користувачів.

3.5 Реалізація програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Першим, що має бути доступним користувачеві є мен, яке дозволить переміщатися між сторінками сайту. Запишемо код головного модуля, який закріпить меню до всіх інших сторінок.

```
<template>
  <div :class="{ 'menu-container': !isMobile }">
    <el-menu
      :default-active="route.path"
      mode="horizontal"
      :router="true"
      :ellipsis="!isMobile"
      :class="isMobile ? 'mobile-menu' : 'menu'"
      ref="menuRef"
    >
      <el-menu-item class="remove-item-styling center-vertical" v-if="!isMobile">
        <template #title>
          <div style="font-size: 20px;">IGenerator</div>
        </template>
      </el-menu-item>
      <MainMenuItem :isMobile="isMobile" index="/">
        <template #icon>
          <div class="generator-icons">
            <el-icon><operation /></el-icon>
            <el-icon v-if="uiStore.showGeneratorBadge" class="generator-badge"
:size="10"><circle-filled /></el-icon>
          </div>
        </template>
        <template #title>Generate</template>
      </MainMenuItem>
      <MainMenuItem :isMobile="isMobile" index="/images" >
        <template #icon><el-icon><icon-menu /></el-icon></template>
        <template #title>Images</template>
      </MainMenuItem>

      <MainMenuItem :isMobile="isMobile" index="/options">
        <template #icon><el-icon><options /></el-icon></template>
        <template #title>Options</template>
      </MainMenuItem>
    </el-menu>

  </div>
  <ImageDialog />
</template>
```

На сторінці генерації користувач вводить необхідні дані для створення зображення. Підключення до AI Horde здійснюється в анонімному режимі за допомогою ключа "0000000000". Анонімне підключення надає трохи обмежений функціонал, але його достатньо для роботи інформаційної системи.

Діаграма діяльності серверу під час процесу автентифікації користувача зображена на рисунку 3.6.

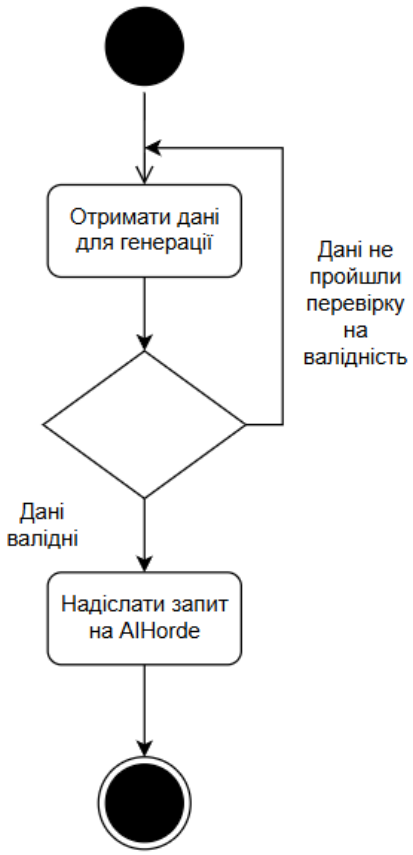

Рисунок 3.6 – Діаграма діяльності серверу під час процесу генерації зображення

Нижче наведено код для підготовки даних для генерації зображення за допомогою AI Horde.

```
async function generateImage(type: typeof generatorType["value"]) {
    if (!validGeneratorTypes.includes(type)) return [];

    if (prompt.value === "") return generationFailed("Failed to generate: No prompt submitted.");
```

```
        for (const multi of Object.values(multiSelect.value)) {
            if (multi.enabled && multi.selected.length === 0) return
generationFailed(multi.noneMessage);
        }

        const canvasStore = useCanvasStore();
        const optionsStore = useOptionsStore();
        const uiStore = useUIStore();

        canvasStore.saveImages();
        const { sourceImage, maskImage, sourceProcessing } = getImageProps(type);

        let model = [selectedModel.value];
        const realModels = filteredAvailableModels.value.filter(el => el.value !==
"Random!" && el.value !== "All Models!");
        if (selectedModel.value === "Random!") {
            model = [realModels[Math.floor(Math.random() * realModels.length)].value];
        }
        if (selectedModel.value === "All Models!") {
            model = realModels.map(el => el.value);
        }

        pushToPromptHistory(prompt.value);

        // Cache parameters so the user can't mutate the output data while it's generating
        const paramsCached: GenerationInputStable[] = [];
        const multiSelectEnabled = Object.values(multiSelect.value).filter(el =>
el.enabled);

        const getMultiSelect = <T>(item: IMultiSelectItem<T>, defaultValue: any): T[]
=> item.enabled ? item.selected : defaultValue;
        const prompts      = promptMatrix();
        const models       = getMultiSelect(multiSelect.value.model,       model);
        const guidances    = getMultiSelect(multiSelect.value.guidance,
[params.value.cfg_scale]);
        const steps        = getMultiSelect(multiSelect.value.steps,
[params.value.steps]);
        const clipSkips    = getMultiSelect(multiSelect.value.clipSkip,
[params.value.clip_skip]);
        const samplers     = getMultiSelect(multiSelect.value.sampler,
[params.value.sampler_name]);
```

```
        const hiResFix     = getMultiSelect(multiSelect.value.hiResFix,
[params.value.hires_fix]);
        const karras       = getMultiSelect(multiSelect.value.karras,
[params.value.karras]);
        const controlTypes = getMultiSelect(multiSelect.value.controlType,
[params.value.control_type]);

        for (let i = 0; i < (params.value.n || 1); i++) {
            for (const currentControlType of controlTypes) {
                for (const currentHiResFix of hiResFix) {
                    for (const currentKarras of karras) {
                        for (const currentModel of models) {
                            for (const currentGuidance of guidances) {
                                for (const currentSteps of steps) {
                                    for (const currentClipSkip of clipSkips) {
                                        for (const currentPrompt of prompts) {
                                            for (const currentSampler of (
                                                currentModel.includes("stable_diffusion_2.0") ?
                                                    ["dpmsolver"] :
                                                    samplers
                                            ) as ModelGenerationInputStable["sampler_name"][]) {
                                                paramsCached.push({
                                                    prompt: currentPrompt,
                                                    params: {
                                                        ...params.value,
                                                        seed_variation: params.value.seed === "" ? 1000 :
1,
                                                        post_processing: postProcessors.value,
                                                        sampler_name: currentSampler,
                                                        control_type:
sourceGeneratorTypes.includes(type) && currentControlType !== "none" ?
currentControlType : undefined,
                                                        cfg_scale: currentGuidance,
                                                        steps: currentSteps,
                                                        clip_skip: currentClipSkip,
                                                        karras: currentKarras,
                                                        hires_fix: currentHiResFix,
                                                        n: 1
                                                    },
                                                    nsfw: nsfw.value,
                                                    censor_nsfw: !nsfw.value,
```

```
                                    trusted_workers: trustedOnly.value,
                                    source_image: sourceImage?.split(",")[1],
                                    source_mask: maskImage,
                                    source_processing: sourceProcessing,
                                    workers: optionsStore.useWorker === "None" ?
undefined : [optionsStore.useWorker],
                                    models: [currentModel],
                                    shared: useOptionsStore().shareWithLaion ===
"Enabled",
                                    r2: true,
                                });
                            }
                        }
                    }
                }
            }
        }
    }

    if (DEBUG_MODE) console.log("Using generation parameters:",
paramsCached)

    generating.value = true;
    uiStore.showGeneratedImages = false;

    // Push each item in the parameters array to the queue
    for (let i = 0; i < paramsCached.length; i++) {
        queue.value.push({
            ...paramsCached[i],
            jobId: "",
            index: i,
            gathered: false,
            failed: false,
        })
    }

    // Reset variables
    outputs.value = [];
```

```
        gatheredImages.value = 0;

        function getMaxRequests(arr: GenerationInputStable[]) {
          return Math.min(arr.length, MAX_PARALLEL_REQUESTS);
        }

        // Loop until queue is done or generation is cancelled
        let secondsElapsed = 0;
        while (!queue.value.every(el => el.gathered || el.failed) && !cancelled.value) {
          if (queueStatus.value.done) await sleep(200);

          const availableQueue = queue.value.filter(el => !el.gathered && !el.failed);
          const t0 = performance.now() / 1000;
          await Promise.all(availableQueue.slice(0,
getMaxRequests(availableQueue)).map(async (queuedImage, i) => {
            await sleep(i * 100);
            if (cancelled.value) return;
            if (queuedImage.waitData?.done) return;

            if (!queuedImage.jobId) {
              const resJSON = await fetchNewID(queuedImage);
              if (!resJSON) {
                generationFailed(undefined, queuedImage);
                queuedImage.failed = true;
                return;
              }
              queuedImage.jobId = resJSON.id as string;
            }

            const status = await checkImage(queuedImage.jobId);
            if (!status) {
              generationFailed(undefined, queuedImage);
              queuedImage.failed = true;
              return;
            }

            if (status.faulted) {
              generationFailed("Failed to generate: Generation faulted.",
queuedImage);
              queuedImage.failed = true;
```

```
                return;
            }

            if (status.is_possible === false) {
                generationFailed("Failed to generate: Generation not possible.",
queuedImage);
                queuedImage.failed = true;
                return;
            }
            queuedImage.waitData = status;

            if (status.done) {
                const finalImages = await getImageStatus(queuedImage.jobId);
                if (!finalImages) {
                    generationFailed(undefined, queuedImage);
                    queuedImage.failed = true;
                    return;
                }
                processImages(finalImages.map(image => ({...image,
...queuedImage})))
                    .then(() => queuedImage.gathered = true);
            }
        }))
        await sleep(500);
        const t1 = performance.now() / 1000;
        secondsElapsed += t1 - t0;

        queueStatus.value = getQueueStatus();
        uiStore.updateProgress(queueStatus.value, secondsElapsed);
        if (DEBUG_MODE) console.log("Checked all images:", queueStatus.value);
    }

    if (DEBUG_MODE) console.log("Images done/cancelled");

    if (cancelled.value) {
        // Retrieve final images that were cancelled
        for (const queuedImage of queue.value) {

            if (queuedImage.gathered || queuedImage.jobId === "") continue;
            const finalImages = cancelled.value ? await
cancelImage(queuedImage.jobId) : await getImageStatus(queuedImage.jobId);
```

```
            if (!finalImages) {
                generationFailed(undefined, queuedImage);
                queuedImage.failed = true;
                continue;
            }
            if (finalImages.length === 0) continue;
            await processImages(finalImages.map(image => ({...image,
...queuedImage})));
          }
          if (DEBUG_MODE) console.log("Got cancelled images");
        }

        return generationDone(multiSelectEnabled);
    }
```

В поданому фрагменті коду відбувається головна частина роботи інформаційної технології, а саме:

1. Перевірка вхідних даних:

– Функція generateImage приймає параметр type, що визначає тип генератора;

– Спочатку перевіряється, чи є вибраний тип у списку допустимих (validGeneratorTypes). Якщо ні, функція повертає порожній масив;

– Далі перевіряється, чи порожній prompt.value. Якщо користувач не ввів опис (prompt), викликається функція generationFailed з відповідним повідомленням про помилку;

– Перевіряються мультивибіркові параметри (якщо вони активовані), щоб гарантувати, що в них вибрані елементи.

2. Збереження зображень:

– Зберігаються зображення з canvasStore перед генерацією, щоб мати доступ до них на наступних етапах.

3. Налаштування моделей для генерації:

– Визначається модель для генерації. Якщо обраний варіант "Random!", вибирається випадкова модель з доступних, а якщо "All Models!" — вибираються всі доступні моделі для паралельного використання.

4. Кешування параметрів:

– Історія введених користувачем промптів зберігається в pushToPromptHistory.

– Основні параметри для генерації зображень кешуються в paramsCached для подальшого використання. Це дозволяє уникнути змінення параметрів користувачем під час генерації.

– Для кожного активованого мультивибіркового параметра (модель, керівництво, кроки тощо) виконується ітерація, щоб створити всі можливі комбінації налаштувань для генерації.

5. Підготовка запитів:

– Кожен варіант з paramsCached додається в чергу queue.value для обробки. Кожен елемент має унікальний jobId та інші атрибути для контролю процесу генерації.

6. Генерація зображень:

– Поки черга не порожня і генерація не скасована, кожен запит обробляється паралельно.

– Якщо queuedImage.jobId не встановлено, викликається fetchNewID для отримання нового ідентифікатора роботи.

– Далі перевіряється статус генерації за допомогою checkImage. Якщо статус вказує на завершення, викликається getImageStatus для отримання фінальних зображень.

– У разі успішного отримання зображень вони обробляються за допомогою processImages.

7. Оновлення інтерфейсу:

– Прогрес генерації оновлюється через updateProgress в uiStore, а queueStatus відображає поточний стан черги.

8. Обробка скасування:

– Виконується обробка кожного зображення, яке можна отримати навіть після скасування запиту.

9. Завершення генерації:

– Після завершення всіх процесів повертається результат генерації за допомогою generationDone, що включає всі успішно згенеровані зображення.

Діаграма діяльності серверу під час процесу отримання користувачем галереї зображена на рисунку 3.7.

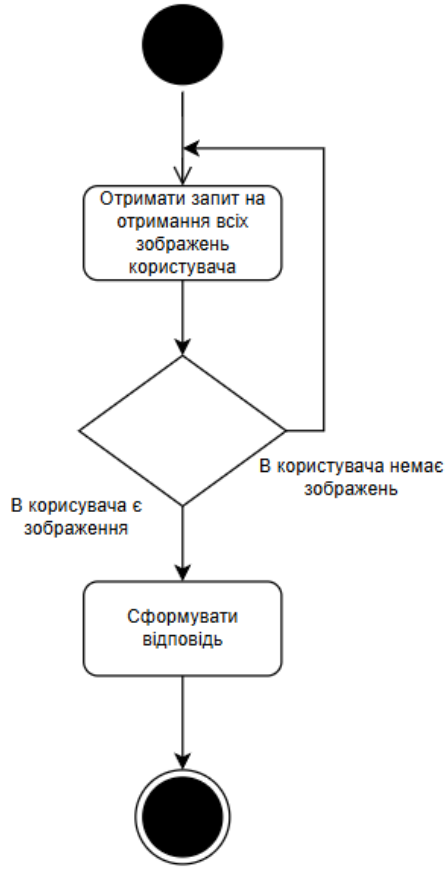

Рисунок 3.7 – Діаграма діяльності серверу під час процесу отримання зображень користувача для галереї

Зображений вище алгоритм (рис. 3.7) можна записати у вигляді коду який має включати панель інструментів для керування сторінками, налаштування

перегляду зображень у різних режимах, можливість вибору та завантаження декількох зображень, а також функціонал для видалення вибраних зображень:

```
<template>
  <div class="images-top-bar">
    <div class="options">

    </div>
    <el-pagination
      layout="prev, pager, next"
      :total="store.outputsLength"
      :page-size="optionStore.pageSize"
      :current-page="store.currentPage"
      @update:current-page="(val: number) => store.currentPage = val"
      hide-on-single-page
      v-if="optionStore.pageless === 'Disabled'"
    />
    <div class="center-both" v-if="uiStore.multiSelect" style="gap: 12px">
      <div>{{ uiStore.selected.length }} selected</div>
      <el-button type="danger" @click="confirmDelete" :icon="Delete"
plain>Delete</el-button>
      <el-button type="success" @click="bulkDownload" :icon="Download" plain
style="margin: 0">Download</el-button>
    </div>
  </div>
  <div v-if="store.outputsLength != 0">
    <div style="display: flex; gap: 8px" v-if="store.currentLayout === 'dynamic'">
      <div v-for="(items, index) in splitList" :key="index" style="flex: 1 1 0%">
        <CustomImage
          v-for="image in items"
          :key="image.id"
          :image-data="image"
          style="margin-bottom: 8px;"
        />
      </div>
    </div>
    <div class="images" v-if="store.currentLayout === 'grid'">
      <CustomImage
        v-for="image in store.currentOutputs"
        :key="image.id"
        :image-data="image"
```

```
            style="width: 200px; height: 200px"
          />
        </div>
    </div>
    <div v-if="store.outputsLength == 0">
      <el-empty description="No Images Found" />
    </div>
</template>
```

Останній етап розробки − це створення сторінки опцій, яка дозволить користувачеві вибирати параметри завантаження, експорту та імпорту зображень. Цей код реалізує базову структуру сторінки налаштувань.

Діаграма діяльності серверу під час процесу архівації зображень користувача зображена на рисунку 3.8.

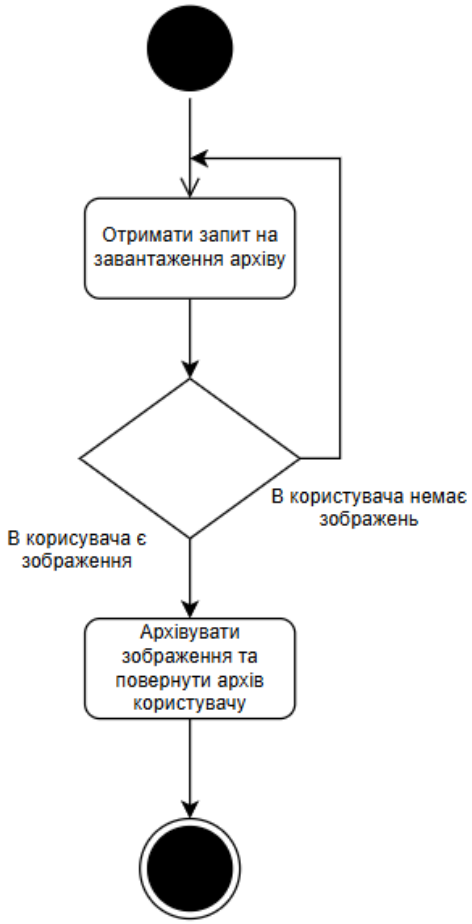

Рисунок 3.8 – Діаграма діяльності серверу під час процесу скачування користувачем архіву

За цією діаграмою (рис. 3.8) напишемо код програми:

```
<template>
  <h1>Settings</h1>
  <el-form
    label-position="top"
    :model="store.options"
    @submit.prevent
  >
    <el-tabs type="border-card" style="min-height: 50vh;">

      <el-tab-pane label="Images">
        <h2>Image Settings</h2>
        <form-radio label="Image Format" prop="downloadFormat" v-
model="store.imageDownloadFormat" :options="['WEBP', 'PNG', 'JPG']" />

        <el-form-item label="Export Images as ZIP">
          <el-button :icon="Download" @click="startDownload()" v-
if="!isDownloading">
            Download {{outputsStore.outputsCount}} image(s)
          </el-button>
          <el-button :icon="Download" disabled v-else>
            Downloading... ({{downloadProgress}} /
{{outputsStore.outputsCount}} image(s))
          </el-button>
        </el-form-item>
        <el-form-item label="Upload Images (ZIP File)">
          <el-upload
            drag
            ref="imageUpload"
            @change="handleFileChange"
            :file-list="fileQueue"
            :limit="1"
            multiple
          >                </el-upload>
        </el-form-item>
      </el-tab-pane>
      <el-tab-pane disabled>
        <!-- Placeholder for future settings -->
      </el-tab-pane>
```

```
      </el-tabs>
    </el-form>
</template>
```

3.6 Тестування інформаційної технології створення зображень за допомогою генеративної нейронної мережі

Тестування розробленої інформаційної технології є важливим етапом, що забезпечує її якість, надійність, безпеку та сумісність з різними пристроями. Воно дозволяє виявити помилки на ранніх етапах і зменшити витрати на їх виправлення в майбутньому. Тестування також підвищує задоволеність користувачів і гарантує відповідність продукту вимогам і специфікаціям. Було проведено комплексне тестування, яке включало перевірку головної сторінки та інтерфейсу, що забезпечило коректну роботу системи. Загалом виконано понад 100 тестових запусків, що дозволило ретельно оцінити всі функціональні можливості та продуктивність продукту.

Почати тестування можна з будь якого пристрою, який забезпечує доступ в Інтернет. Першим, що побачить користувач розробленої інформаційної технології буде головна сторінка на якій і відбувається генерація (рис. 3.9). У верхній частині інтерфейсного вікна розташоване меню, яке допоможе перемикатися між сторінками.

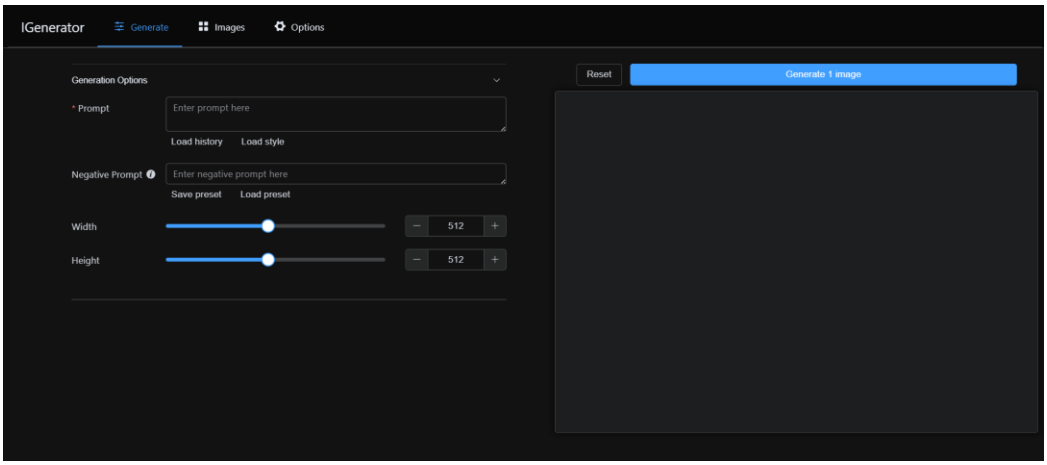

Рисунок 3.9 – Загальний вигляд інтерфейсного вікна сторінки генерації

Введемо текст, опціонально також опишемо що ми не хочемо бачити на зображенні в блоці Negative Prompt і розміри бажаного зображення, та очікуємо на завершення генерації (рис. 3.10).

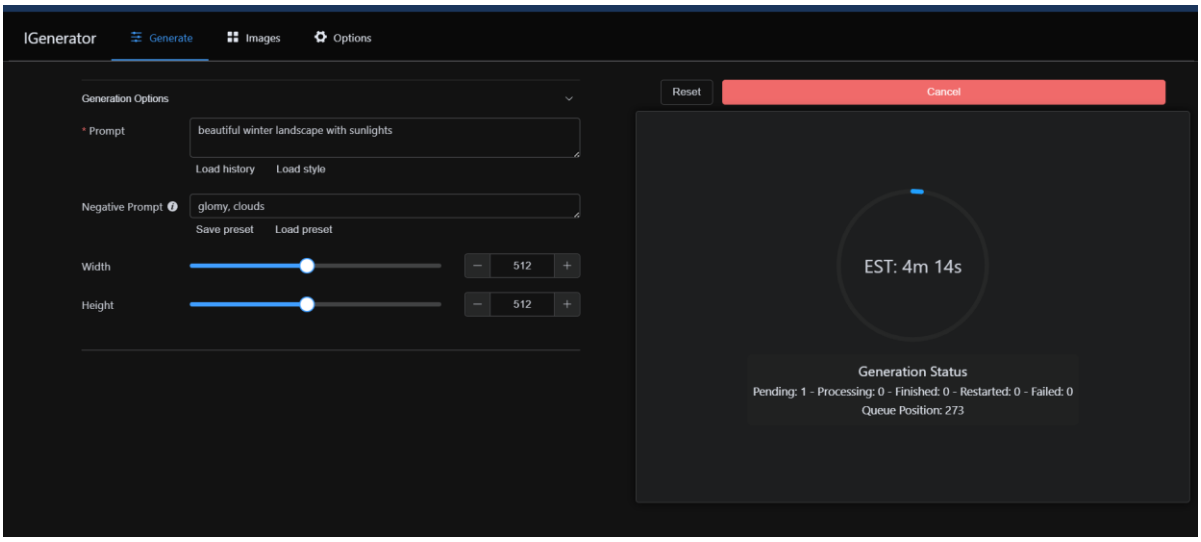

Рисунок 3.10 – Загальний вигляд інтерфейсного вікна сторінки генерації під час генерації

Отримуємо готове зображення що відповідає введеному тексту та параметрам (рис. 3.11).

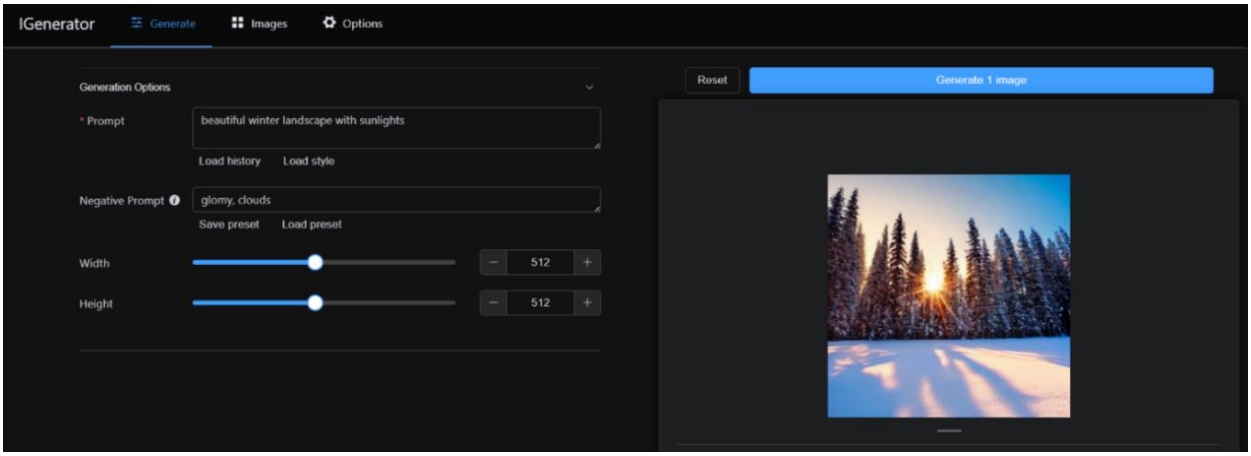

Рисунок 3.11 – Загальний вигляд інтерфейсного вікна сторінки генерації з готовим зображенням

Зайдемо на сторінку галереї з раніше генерованими зображеннями (рис. 3.12).

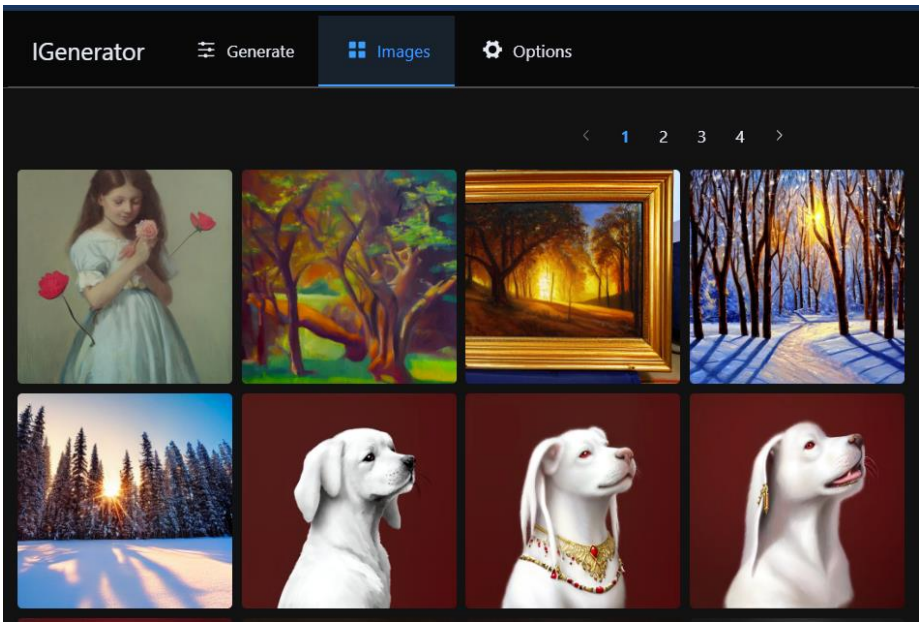

Рисунок 3.12 – Загальний вигляд інтерфейсного вікна галереї

Переглянемо сторінку Options, тут можна завантажити раніше згенеровані зображення в бажаному форматі, на майбутнє тут можна розмістити різноманітні налаштування для AIHord, це дозволить користувачеві персоналізувати роботу генеративної мережі (рис. 3.13).

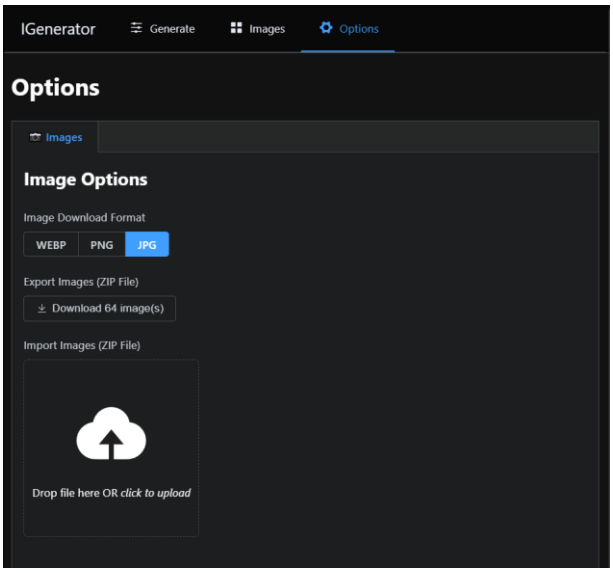

Рисунок 3.13 – Загальний вигляд інтерфейсного вікна сторінки налаштувань

Можна завантажити всі згенеровані зображення в zip форматі (рис. 3.14), або лише одне зображення кнопкою Download (рис. 3.15).

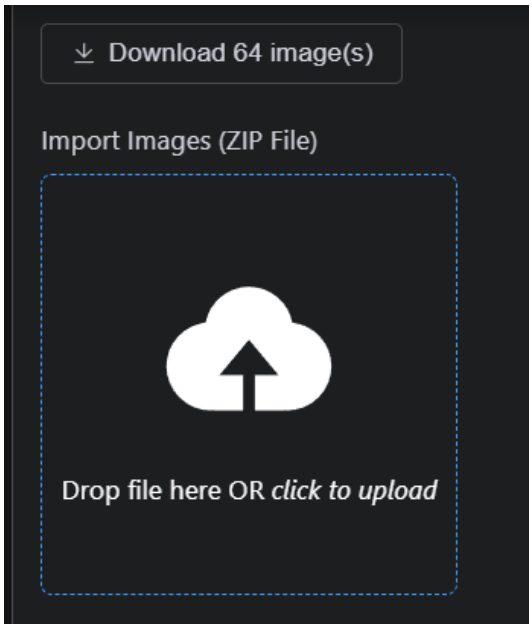

Рисунок 3.14 – Завантаження власних згенерованих зображень

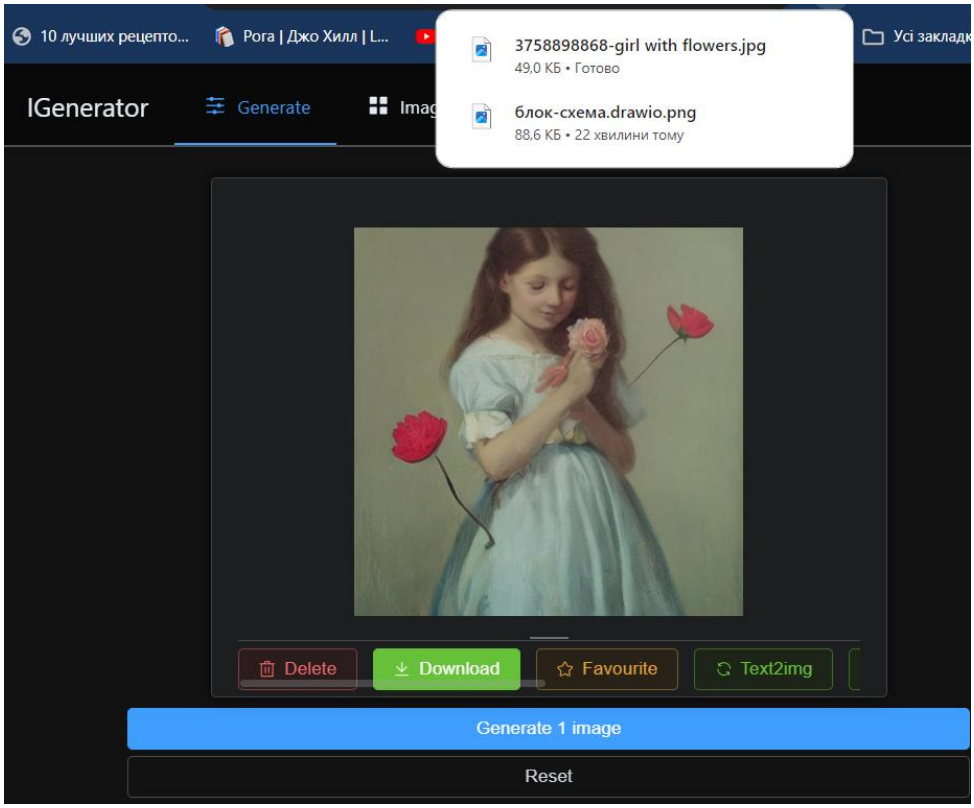

Рисунок 3.15 – Загальний вигляд інтерфейсного вікна під час скачування картинки

Розроблений програмний модуль генерації зображень з використанням генеративної нейронної мережі успішно пройшов тестування та відповідає всім заданим вимогам.

В таблиці 3.1 подано порівняння результатів тестування розробленого програмного продукту та аналогів.

В результаті розробки було реалізовано функціонал, який був відсутній або частково відсутній в системах-аналогах, а саме:

- галерея;
- завантаження зображень у вигляді архіву;
- встановлення розмірів майбутнього зображення;
- підтримка мобільної версії.

Таблиця 3.1 – Результати тестування розробленого програмного продукту та аналогів

Функціонал	Інформаційна технологія	Midjorney	DeepArt	OpenArt
Наявність галереї	+	-	+	-
Завантаження зображень архівом	+	-	-	-
Завантаження зображень в бажаному форматі	+	+	-	-
Встановлення розмірів бажаного зображення	+	-	-	+
Підтримка мобільної версії	+	+	+	-

Отже, проаналізувавши та протестувавши розроблений програмний продукт можна дійти висновку, що поставлених цілей було досягнуто. З поданої інформації таблиці 3.1 можна побачити, що розроблена інформаційна технологія створення зображень за допомогою генеративної нейронної мережі має покращений функціонал у порівнянні з системами аналогами щонайменше на 3

можливості: наявність галереї для перегляду зображень, можливість завантаження зображень у вигляді архіву, а також встановлення бажаних розмірів для генерованих зображень. Розширений функціонал забезпечить зручність користування та полегшить творчий процес.

3.7 Висновок до розділу 3

У цьому розділі було розглянуто різні підходи до генерації зображень за допомогою генеративних нейронних мереж, зокрема архітектуру GAN. Оцінивши переваги та недоліки встановлено, що розробка здатна забезпечити високу якість та різноманітність результатів генерації, оскільки може генерувати різні типи зображень, залежно від архітектури та налаштувань. Тестування розробленого програмного модуля показало, що він виконує всі заявлені функції та відповідає вимогам. Під час тестування було виявлено, що розроблений програмний продукт має кілька суттєвих переваг у порівнянні з аналогами. Реалізувано такі можливості, яких не було в інших системах: наявність галереї для перегляду зображень, можливість завантаження зображень у вигляді архіву, а також функцію встановлення бажаних розмірів для генерованих зображень, тобто розробка має розширений функціонал на 3 можливості.

У порівнянні з такими системами таких, як Midjorney, DeepArt та OpenArt, розроблена технологія виграє за кількома важливими критеріями. Наприклад, має галерею для збереження і перегляду зображень, можливість завантаження архівів з кількома зображеннями одночасно та підтримує встановлення розмірів майбутнього зображення. Крім того, інформаційна технологія підтримує мобільну версію, чого не мають деякі аналоги.

Отже, на основі проведеного тестування можна зробити висновок, що розроблена інформаційна технологія створення зображень за допомогою генеративної нейронної мережі має покращений функціонал і забезпечує кращий досвід користувача, а також полегшує процес творчості завдяки додатковим функціям, які не були реалізовані в аналогах.

4 ЕКОНОМІЧНА ЧАСТИНА

Ефективне впровадження нових науково-технічних рішень у галузі генеративних нейромереж є можливим за умови відповідності сучасним вимогам технічного прогресу та врахування економічних аспектів. Важливим елементом цього процесу є оцінка економічної ефективності результатів наукових досліджень. Дана магістерська робота, присвячена розробці та дослідженню «Інформаційної технології створення зображень за допомогою генеративної нейронної мережі», орієнтована на подальший вихід на ринок. Прийняття рішення про комерціалізацію результатів може відбуватися в ході самої розробки, що відкриває можливість для її практичного застосування.

Цей напрямок визначений як пріоритетний завдяки потенційній корисності для широкого кола зацікавлених сторін та можливості отримання економічної вигоди. Для успішного впровадження важливо знайти інвестора, який зацікавиться реалізацією проєкту та зрозуміє його економічну обґрунтованість.

Для досягнення цієї мети визначено наступні етапи:

− Здійснено комерційний аудит розробки, який охоплює аналіз науково-технічного рівня та комерційного потенціалу.

− Розраховано витрати на реалізацію розробки.

− Визначено економічну ефективність розробки у разі її впровадження та обґрунтовано доцільність комерціалізації для потенційного інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного та технологічного аудиту дослідження за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» є оцінка науково-технічного рівня та комерційного потенціалу розробки, отриманої в результаті науково-дослідної діяльності. Рекомендується оцінювати науково-технічний рівень та комерційний

потенціал технології за допомогою п’ятибальної системи оцінювання за 12-ма критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Ринкові перспективи					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведені до таблиці 4.2. Для опитування було залучено експертів з ВНТУ, кафедри комп'ютерних наук: к.т.н., доцент Крилик Людмила Вікторівна, к.т.н., доцент Озеранський Володимир Сергійович, к.т.н., проф. Колесницький Олег Костянтинович.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Крилик Л.В.	Озеранський В.С.	Колесницький О. К.
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	4	3	4
5. Ринкові переваги (експлуатаційні витрати)	4	4	2
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	4	3
Сума балів	$СБ_1$=43	$СБ_2$=43	$СБ_3$=43
Середньоарифметична сума балів $СБ_c$	(43+43+43)/3 = 43		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 4.3 [27].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» оцінюється у 43 бали, що, відповідно до таблиці 4.3, свідчить про високу комерційну значущість цього дослідження. Такий високий рівень потенціалу обґрунтовує доцільність комерційного використання розробки.

Ця технологія орієнтована на користувачів, які цікавляться створенням зображень за допомогою генеративних нейронних мереж і мають доступ до інтернету. Для її використання користувачам потрібно відвідати сайт ресурсу та придбати доступ до продукту за фіксованою ціною.

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [28]:

$$q_i = \frac{P_i}{P_{базі}}, \quad (4.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
Якість зображення	65	80	1,23	60%
Швидкість генерації	4	3	0,75	40%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає. Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [28]:

$$I_{НП} = \prod_{i=1}^{n} q_i, \quad (4.2)$$

де $I_{нп}$ – загальний показник конкурентоспроможності;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому Інп = 1.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [28]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{4.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками;

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{тп} = 1{,}22 \cdot 0{,}6 + 0{,}75 \cdot 0{,}4 = 1{,}032.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [28]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{4.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{ЕП} = 0{,}82 \cdot 0{,}5 + 0{,}68 \cdot 0{,}5 = 0{,}75.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [28]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{4.5}$$

$$K_{ІНТ} = 1 \cdot 1{,}032 / 0{,}75 = 1{,}376.$$

Інтегральний показник конкурентоспроможності $K_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з виконанням науково-дослідної роботи на тему «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі», групуються за відповідними статтями під час планування, обліку та розрахунку собівартості проєкту. Це дозволяє ефективніше контролювати бюджет і забезпечувати економічну прозорість у процесі дослідження.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» включаються витрати на основну та додаткову заробітну плату керівників відділів, лабораторій, секторів і груп, наукових та інженерно-технічних працівників, конструкторів, технологів, креслярів, копіювальників, лаборантів, робітників, студентів, аспірантів та інших співробітників, що безпосередньо зайняті реалізацією конкретної теми. Зарплата розраховується на основі посадових окладів, відрядних розцінок та

тарифних ставок згідно з чинними в організаціях системами оплати праці. Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [28]:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$=21 дні.

$$З_о = 30000 \cdot 40 / 21 = 28571 \text{ (грн.)}.$$

Проведені розрахунки зведемо до таблиці 4.5.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	34000	1619,27	20	28571
Інженер-програміст	30000	1428,5	40	57142
Всього				82713

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{4.7}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{4.8}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати, приймемо $Мм$=8000 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [28]:

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$$С_1 = 8000 \cdot 1{,}1 \cdot 1{,}65 / (21 \cdot 8) = 86{,}42 \text{ (грн.)},$$
$$З_{р1} = 86{,}42 \cdot 8 = 691{,}36 \text{ (грн.)}.$$

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	8	2	86,42	691,36
2.Налагоджувальні	10	4	117,85	1178,5
3.Випробувальні	2	3	106,07	212,14
Всього				2082

Додаткову заробітну плату дослідників та робітників розраховуємо як 10 … 12% від суми основної заробітної плати за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_{дод} = (82713 + 2082) \cdot 11 / 100\% = 9327{,}45 \text{ (грн.)}.$$

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від основної та додаткової заробітної плати робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$$Зн = (82713 + 2082 + 9327{,}45) \cdot 22 / 100\% = 20706{,}939 \text{ (грн.)}.$$

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» входять витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби праці, придбані у сторонніх підприємств, установ і організацій, які використовуються для проведення досліджень за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (4.11)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір А 4	169	1	169
Ручка	20	1	20
Диск оптичний OPTIMA CD	15	1	15
Flesh-пам'ять GOODRAM 64 C10A	410	1	410
Всього			614
З врахуванням коефіцієнта транспортування			675,4

4.3.4 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$$B_{спец} = 30000,00 \cdot 1 \cdot 1,11 = 33000 \ (грн.).$$

Отримані результати зведемо до таблиці 4.8:

Таблиця 4.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
1. Ноутбук Lenovo	1	30000	33000
3. Миша Vinga	1	300	330
4. Крісло офісне	1	3076	3383,6
5. Стіл офісний	1	4000	4400
Всього			41113,6

4.3.5 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.13)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$$B_{прг} = 1600 \cdot 1 \cdot 1,11 = 1776 \text{ (грн.)}.$$

Отримані результати зведемо до таблиці 4.9:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Деплоймент додатку на AWS	1	1600	1776
AWS S3 Bucket storage	1	400	440
Всього			2216

4.3.6 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.14)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$A_{обл} = (33000 \cdot 1) / (6 \cdot 12) = 1011{,}11 \text{ (грн.)}.$$

Проведені розрахунки по амортизаційних відрахуваннях зведемо до таблиці 4.10.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук	33000	6	1	458,33
Приміщення лабораторії	170000	20	1	708,33
Всього				1166,63

4.3.7 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (Ве) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.15)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,32 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$$B_e = 0{,}7 \cdot 260{,}0 \cdot 4{,}32 \cdot 0{,}75 / 0{,}99 = 595{,}63 \text{ (грн.)}.$$

4.3.8 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» належать витрати на відрядження штатних співробітників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, що займаються розробленням та дослідженнями, а також відрядження, пов'язані з проведенням випробувань обладнання та пристроїв. Крім того, сюди входять витрати на відрядження для участі в наукових конференціях, нарадах, з’їздах, що стосуються виконання конкретного дослідження.

Витрати за статтею «Службові відрядження» розраховуються як 20-25% від суми основної заробітної плати дослідників і працівників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.16)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$$В_{св} = (82713 + 2082) \cdot 20 / 100\% = 16959 \text{ (грн.)}.$$

4.3.9 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо цей параметр як $Н_{ів}$ = 50%.

$$І_в = (82713 + 2082) \cdot 50 / 100\% = 42397{,}5 \text{ (грн.)}.$$

4.3.10 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» у дослідженні на тему «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» відносяться такі витрати: витрати на управління організацією, витрати на винахідницьку та раціоналізаторську діяльність,

витрати на підготовку, перепідготовку та навчання персоналу, витрати на наймання робочої сили, оплату послуг банків, освоєння нового виробництва, а також витрати на науково-технічну інформацію та рекламу.

Розмір витрат за статтею «Накладні (загальновиробничі) витрати» розраховується як 100÷150% від суми основної заробітної плати дослідників і працівників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (4.18)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$В_{нзв} = (82713 + 2082) \cdot 100 / 100\% = 84795 \text{ (грн.)}.$$

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{спец} + В_{прг} + \\ + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}, \quad (4.19)$$

$В_{заг}$= 82713+ 2082+ 9327,45 + 20706,939 + 675,4 + 41113,6+ 2216+ 1166,63+ +595,63 + 16959 + 16959+ 42397,5 + 84795= 321707,149 (грн.).

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.20)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

$$ЗВ = 321707,149 / 0,9 = 357452,387 \text{ (грн.)}.$$

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 4000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 470,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів

від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [28]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.21)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 0,4$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 0,18$;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1 \cdot 470 + 4000 \cdot 2000) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18) = 2178047,95 \text{ (грн.)}.$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1 \cdot 470 + 4000 \cdot (2000 + 1500)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18) = 3811487,95 \text{ (грн.)}.$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1 \cdot 470 + 4000 \cdot (2000 + 1500 + 650)) \cdot 0,83 \cdot 0,4 \cdot (1 - 0,18) = 4519311,95 \text{грн.}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.22)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 2178047{,}95/(1+0{,}18)^1 + 3811487{,}95/(1+0{,}18)^2 + 4519311{,}95/(1+0{,}18)^3 =$$
$$= 7333747.42 \text{ (грн.)}.$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.23)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 357452,387 (грн.).

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 357452{,}387 = 714904{,}774 \text{ (грн.)}.$$

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.24}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 7333747.42 (грн.);

PV – теперішня вартість початкових інвестицій, 714904,774 (грн.).

$$E_{абс} = ПП - PV = 7333747.42 - 714904,774 = 6618842.646 \text{ (грн.)}.$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.25}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 6618842.646 \ / 714904,774)^{1/3} - 1 = 1.17.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.26}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$$\tau_{мін} = 0,1+0,25 = 0,35 < 1.17,$$

$\tau_{мін} < E_в$ свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності.

Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.27}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = 1 / 1.17 = 0,85 \text{ р.}$$

$T_{ок} <$ 3-х років, що може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновок до розділу 4

Згідно з проведеними дослідженнями, комерційний потенціал розробки за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» оцінюється в 43 бала, що вказує на високу комерційну значущість таких досліджень. При оцінці конкурентоспроможності, науково-технічна розробка перевершує існуючі аналоги приблизно в 1,376 разів за узагальненим коефіцієнтом конкурентоспроможності. Крім того, період окупності складає 0,85 року, що є меншим за 3 роки, підтверджуючи комерційну привабливість розробки та її потенційну привабливість для інвесторів, які можуть профінансувати впровадження та вихід цієї технології на ринок. Таким чином, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі».

ВИСНОВКИ

Всі завдання, поставлені для реалізації інформаційної технології створення зображень за допомогою генеративної нейронної мережі виконані в повному обсязі, а саме: проведено аналіз технічного рівня існуючих систем генерації зображень та обґрунтувано доцільність розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі; розроблено математичну модель інформаційної технології; розглянуто методи та технології створення зображень нейронними мережами; спроектовано архітектуру та структуру програмного модуля інформаційної технології; проаналізувано та обґрунтувано вибір інструментів для розробки; виконано програмну реалізацію програмного модуля інформаційної технології; виконано тестування програми та проведено аналіз отриманих результатів; економічно обґрунтувано доцільність розробки інформаційної технології створення зображень за допомогою генеративної нейронної мережі.

При виконанні магістерської кваліфікаційної роботи реалізовано інформаційну технологію створення зображень за допомогою генеративної нейронної мережі.

Проведено аналіз технічних рішень програм для генерації зображень та доведено актуальність розробки інформаційної технології генерації зображень за допомогою генеративної нейронної мережі. Розглянуті системи-аналоги є корисними та доволі актуальними, проте усі вони або не мають достатнього функціоналу або надають цей функціонал лише за платну підписку, обмежуючи користувача в творчому процесі.

На основі аналізу існуючих моделей, що застосовуються для вирішення поставленої задачі обґрунтовано використання в інформаційній технології математичної моделі, яка описує алгоритми роботи генеративної нейронної мережі (GAN), ця модель відповідає за генерацію зображень на основі поданих користувачем даних.

Спроектовано архітектуру та структуру програмного модуля. Розроблено UML-діаграму послідовності, що ілюструє основні етапи інформаційної технології створення зображень. Ця діаграма слугує важливим інструментом для візуалізації процесу та взаємодії компонентів системи. Також створено схему алгоритму роботи інформаційної технології та діаграми процесів, які відбуваються під час її роботи.

Обґрунтовано вибір мови програмування та інструментів для розробки. Реалізацію інформаційної технології проведено з використанням мови програмування JavaScript та фреймворку Vue.js.

Розроблена інформаційна технологія демонструє високий рівень зручності та продуктивності порівняно з існуючими аналогами. Додано 3 функціональних можливості, яких не мають аналоги, а саме: галерея зображень користувача, завантаження зображень архівом, надання користувачеві можливості обрати розміри майбутнього зображення. Проведене тестування підтвердило працездатність системи, яка забезпечує швидку та якісну генерацію зображень і відповідає сучасним вимогам ринку.

Економічно обґрунтувано доцільність розробки аналізом витрат і прогнозуванням прибутковості проекту. Було спрогнозовано орієнтовану величину витрат сумою 357452,387 гривень. Шляхом аналізу розрахунків було доведено що розроблена інформаційна технологія переважає існуючі аналоги приблизно в 1,376 рази. Період окупності інформаційної технології створення зображень за допомогою генеративної мережі складе близько 0,85 років.

Поставлена мета, а саме розширення функціональних можливостей інформаційної технології створення зображень за допомогою генеративної нейронної мережі була досягнута за рахунок використання фреймворку для програмування Vue.js та відкритої API AIHorde, що дозволило створити зручний інтерфейс користувача та отримати такі функціональні можливості як галерея користувача, функція завантаження зображень у вигляді архіву, можливість вибору користувачем розмірів зображення перед генерацією. Це сприяло створенню зручного додатку, який може конкурувати з системами-аналогами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Верба Г. Р., Крилик Л. В. Обґрунтування доцільності розробки інформаційної технології для генерації зображень за допомогою генеративної нейронної мережі. *Матеріали Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» : збірник доповідей*. [Електронний ресурс]. Вінниця: ВНТУ, 2024. С.1189 – 1190. URL: https://press.vntu.edu.ua/index.php/vntu/catalog/view/836/1459/2733-1. (дата звернення 29.09.24).

2. Свідоцтво про реєстрацію авторського права на твір № 130789 від 17.10.2024 р. Комп'ютерна програма «Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі» / Крилик Л. В., Верба Г. Р. ДО «Український національний офіс інтелектуальної власності та інновацій» (УКРНОІВІ). – 2 с.

3. Midjourney: огляд штучного інтелекту для генерації зображень [Електронний ресурс]. URL: https://root-nation.com/ua/soft-ua/web-services-ua/ua-midjourney-images-ai-review/?amp (дата звернення 29.09.2024).

4. App Store. DeepArt: генератор мистецтва за допомогою штучного інтелекту [Електронний ресурс]. URL: https://apps.apple.com/ua/app/deepart-ai-art-generator/id6502173348?l=uk (дата звернення 29.09.2024).

5. Unite AI. OpenArt: огляд штучного інтелекту [Електронний ресурс]. URL: https://www.unite.ai/uk/openart-ai-review/ (дата звернення 29.09.2024).

6. Kramer, M. Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal, 1991. V. 37(2). P. 233-243. DOI: 10.1002/aic.690370209.

7. Kingma, D., & Welling, M. Auto-Encoding Variational Bayes. arXiv e-prints. [Електронний ресурс]. URL: https://arxiv.org/pdf/1312.6114.pdf (дата звернення 29.09.2024).

8. Unite AI. Що таке генеративна змагальна мережа (GAN)? [Електронний ресурс]. URL: https://www.unite.ai/uk/what-is-a-generative-adversarial-network-gan/ (дата звернення 29.09.2024).

9. Яровий А. А., Крилик Л. В., Козловський А. В. Сучасні інформаційні технології у сфері штучного інтелекту : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс]. Вінниця : ВНТУ, 2023. 145 с.

10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley & Bengio, Y. Generative Adversarial Nets. Advances in Neural Information Processing Systems, 2014. V. 27. [Електронний ресурс]. URL: https://arxiv.org/abs/1406.2661 (дата звернення 29.09.2024).

11. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. Вінниця : ВНТУ, 2023. (PDF, 58 с.)

12. JavaScript: що таке JavaScript та як функціонує [Електронний ресурс]. URL: http://ruszura.in.ua/uncategorized/scho-take-javascript yakfunktsionuyu javascript.html (дата звернення 30.09.2024).

13. Wezom. Архітектура програмного забезпечення [Електронний ресурс]. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення 30.10.2024).

14. VPN Unlimited. Переваги модульного програмування [Електронний ресурс]. URL: https://www.vpnunlimited.com/uk/help/terms/modular-programming. (дата звернення 30.10.2024).

15. QA TestLab. Архітектура клієнт-сервер [Електронний ресурс]. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 30.10.2024).

16. Sigma Software University. Що таке Python [Електронний ресурс]. URL: https://university.sigma.software/what-is-python (дата звернення 30.10.2024).

17. Merehead. Мови програмування 2023 року [Електронний ресурс]. URL: https://merehead.com/ua/blog/most-in-demanding-programming-languages-2023 (дата звернення 30.10.2024).

18. Wikipedia. JavaScript [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/JavaScript (дата звернення 29.09.2024).

19. GOIT. Що таке JavaScript і для чого він потрібен [Електронний ресурс]. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення 30.10.2024).

20. DOU. Форуми: обговорення JavaScript [Електронний ресурс]. URL: https://dou.ua/forums/topic/35184 (дата звернення 30.10.2024).

21. Босько В. В., Константинова Л. В. Аналіз та дослідження фреймворку Angular як засобу розробки вебсайтів. Центральноукраїнський науковий вісник. Технічні науки. 2022. Вип. 5(36). С. 3.

22. DOU. Порівнюємо React, Angular і Vue − найпопулярніші бібліотеки й фреймворки у 2022 році [Електронний ресурс]. URL: https://dou.ua/forums/topic/39933 (дата звернення 06.11.2024).

23. Cases Media. Що таке React.js і як почати вивчати [Електронний ресурс]. URL: https://cases.media/en/article/sho-take-react-js-yak-pochati-vivchati-reakt-navichki-dlya-react-developer (дата звернення 06.11.2024).

24. DAN IT. Що таке React.js [Електронний ресурс]. URL: https://dan-it.com.ua/uk/blog/chto-takoe-react-js-i-dlja-chego-on-nuzhen/ (дата звернення 06.11.2024).

25. Vue.js: вступ. [Електронний ресурс]. URL: https://vuejs.org/guide/introduction.html (дата звернення 06.11.2024).

26. Vue.js (українською): вступ. [Електронний ресурс]. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення 06.11.2024).

27. Технологічний аудит [Електронний ресурс]. URL: https://studfile.net/preview/7336428/ (дата звернення 27.11.2024).

28. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад.: В. О. Козловський, О. Й. Лесько,В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

Додаток А (обов'язковий)
Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія створення зображень за допомогою генеративної нейронної мережі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 92,00% Схожість 8,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи Верба Г.Р.

Керівник роботи Крилик Л.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов’язковий)

Лістинг програми

```
<script setup lang="ts">
import { computed, reactive, ref } from 'vue';
import { useGeneratorStore } from '@/stores/generator';
import {
   type FormRules,
   ElCollapse,
   ElCollapseItem,
   ElForm,
   ElButton,
   ElCard,
   ElMenu,
   vLoading,
   ElLoading,
   ElTooltip,
   ElRow,
   ElCol
} from 'element-plus';
import {
   Comment,
   PictureFilled,
   MagicStick,
} from '@element-plus/icons-vue';
import BrushFilled from '../components/icons/BrushFilled.vue';
import StarEdit24Regular from '../components/icons/StarEdit24Regular.vue';
import ImageSearch from '../components/icons/ImageSearch.vue';
import ImageProgress from '../components/ImageProgress.vue';
import FormSlider from '../components/FormSlider.vue';
import FormSelect from '../components/FormSelect.vue';
import FormInput from '../components/FormInput.vue';
import FormSwitch from '../components/FormSwitch.vue';
import FormModelSelect from '../components/FormModelSelect.vue';
import FormPromptInput from '../components/FormPromptInput.vue';
import GeneratedCarousel from '../components/GeneratedCarousel.vue'
import CustomCanvas from '../components/CustomCanvas.vue';
import GeneratorMenuItem from '../components/GeneratorMenuItem.vue';
import DialogList from '../components/DialogList.vue';
import RatingView from '../components/RatingView.vue';
import BaseLink from '../components/BaseLink.vue';
import { useUIStore } from '@/stores/ui';
import { useCanvasStore } from '@/stores/canvas';
import { useOptionsStore } from '@/stores/options';
import { breakpointsTailwind, useBreakpoints } from '@vueuse/core';
import handleUrlParams from "@/router/handleUrlParams";
```

```
import { useRatingStore } from '@/stores/rating';
import InterrogationView from '@/components/InterrogationView.vue';
import { useEllipsis } from '@/utils/useEllipsis';
import { useVideoStore } from '@/stores/video';
import { formatSeconds } from '@/utils/format';

const breakpoints = useBreakpoints(breakpointsTailwind);
const isMobile = breakpoints.smallerOrEqual('md');

const store = useGeneratorStore();
const uiStore = useUIStore();
const canvasStore = useCanvasStore();
const optionsStore = useOptionsStore();
const ratingStore = useRatingStore();
const videoStore = useVideoStore();
const { ellipsis } = useEllipsis();

const negativePromptLibrary = ref(false);

const samplerListLite = ["k_lms", "k_heun", "k_euler", "k_euler_a", "k_dpm_2", "k_dpm_2_a",
"DDIM"]
const dpmSamplers = ['k_dpm_fast', 'k_dpm_adaptive', 'k_dpmpp_2m', 'k_dpmpp_2s_a',
'k_dpmpp_sde']


const pendingRequests = computed(() => store.queue.filter(el => el.jobId === '' || !el.waitData));

const rules = reactive<FormRules>({
   prompt: [{
      required: true,
      message: 'Please input prompt',
      trigger: 'change'
   }],
   apiKey: [{
      required: true,
      message: 'Please input API Key',
      trigger: 'change'
   }]
});

function updateCurrentSampler(newSamplers: string[]) {
   if (!store.params) return newSamplers;
   if (!store.params.sampler_name) return newSamplers;
   if (newSamplers.indexOf(store.params.sampler_name) === -1) {
      store.params.sampler_name = newSamplers[0] as any;
   }
   return newSamplers;
}
```

```
function formatEST(seconds: number) {
   return "EST: " + formatSeconds(seconds, true, { days: true, hours: true, minutes: true, seconds:
true })
}

function disableBadge() {
   if (!store.validGeneratorTypes.includes(store.generatorType)) uiStore.showGeneratorBadge =
false;
}

function onMenuChange(key: any) {
   store.generatorType = key;
   disableBadge();
   console.log(key)
}

function onDimensionsChange() {
   canvasStore.showCropPreview = true;
   canvasStore.updateCropPreview();
}

const availablePostProcessors = computed(() => {
   const upscalersDisabled =
      store.postProcessors.includes("RealESRGAN_x4plus_anime_6B") ||
      store.postProcessors.includes("RealESRGAN_x4plus") ||
      store.postProcessors.includes("NMKD_Siax") ||
      store.postProcessors.includes("4x_AnimeSharp");

   const upscalerDisabled = (name: string) => !store.postProcessors.includes(name as any) &&
upscalersDisabled;

   return [
      "GFPGAN",
      "CodeFormers",
      { label: "RealESRGAN_x4plus", value: "RealESRGAN_x4plus", disabled:
upscalerDisabled("RealESRGAN_x4plus") },
      { label: "RealESRGAN_x4plus_anime_6B", value: "RealESRGAN_x4plus_anime_6B",
disabled: upscalerDisabled("RealESRGAN_x4plus_anime_6B") },
      { label: "NMKD_Siax", value: "NMKD_Siax", disabled: upscalerDisabled("NMKD_Siax") },
      { label: "4x_AnimeSharp", value: "4x_AnimeSharp", disabled:
upscalerDisabled("4x_AnimeSharp") },
      "strip_background"
   ]
})

disableBadge();
handleUrlParams();
```

```
</script>

<template>
  <div class="form">
    <div v-if="store.generatorType === 'Rating'" style="padding-bottom: 50px;">
      <h1 style="margin: 0">Image Rating</h1>
      <div>Rate images based on aesthetics to gain kudos and help <BaseLink
href="https://laion.ai/">LAION</BaseLink> - the non-profit who helped train Stable Diffusion -
improve their datasets!</div>
      <div v-if="optionsStore.apiKey === '0000000000' || optionsStore.apiKey === ''">You have
rated a total of <strong>{{ ratingStore.imagesRated }}</strong> images! <BaseLink router
href="/options">Sign in</BaseLink> using your API key to start earning kudos.</div>
      <div v-else>From rating a total of <strong>{{ ratingStore.imagesRated }}</strong> images,
you have gained <strong>{{ ratingStore.kudosEarned }}</strong> kudos!</div>
      <el-button
        @click="() => ratingStore.updateRatingInfo()"
        v-if="!ratingStore.currentRatingInfo.id"
        :disabled="ratingStore.submitted"
        style="margin-top: 10px"
        size="large"
      >{{ ratingStore.submitted ? "Loading image..." : "Start rating!"}}</el-button>
      <RatingView
        :id="ratingStore.currentRatingInfo.id || ""
        :image-source="ratingStore.currentRatingInfo.url || ""
        :submitted="ratingStore.submitted"
        @onRatingSubmit="ratingStore.submitRating"
      />
    </div>
    <div v-else-if="store.generatorType === 'Interrogation'" style="padding-bottom: 50px;">
      <h1 style="margin: 0">Interrogation</h1>
      <div>Interrogate images to get their predicted descriptions, tags, and NSFW status.</div>
      <InterrogationView />
    </div>
    <el-form
      label-position="left"
      label-width="140px"
      :model="store"
      class="container"
      :rules="rules"
      @submit.prevent
      v-else
    >
      <div class="sidebar">
        <el-collapse v-model="uiStore.activeCollapse" style="margin-bottom: 24px">
          <el-collapse-item title="Generation Options" name="1">
            <form-prompt-input />
            <form-input
              label="Negative Prompt"
```

```
                    prop="negativePrompt"
                    v-model="store.negativePrompt"
                    :autosize="{ maxRows: 15 }"
                    resize="vertical"
                    type="textarea"
                    placeholder="Enter negative prompt here"
                    info="What to exclude from the image. Not working? Try increasing the
guidance."
                    label-position="top"
                  >
                    <template #inline>
                      <el-button class="small-btn" style="margin-top: 2px" @click="() =>
store.pushToNegativeLibrary(store.negativePrompt)" text>Save preset</el-button>
                      <el-button class="small-btn" style="margin-top: 2px" @click="() =>
negativePromptLibrary = true" text>Load preset</el-button>
                    </template>
                  </form-input>
            <form-slider label="Width"          prop="width"          v-model="store.params.width"
:min="store.minDimensions"    :max="store.maxDimensions" :step="64"
:change="onDimensionsChange" />
                  <form-slider label="Height"          prop="height"          v-
model="store.params.height"                  :min="store.minDimensions"
:max="store.maxDimensions" :step="64"   :change="onDimensionsChange" />
 </el-collapse-item>
            </el-collapse>
          </div>
          <div class="main">
            <el-button @click="() => store.resetStore()" class="reset-btn">Reset</el-button>
            <el-button
              v-if="!store.generating"
              type="primary"
              class="generate-cancel-btn"
              @click="() => store.generateImage(store.generatorType)"
            >
              <span>
                Generate {{ store.totalImageCount }} image{{ store.totalImageCount === 1 ? "" :
"s" }}
                <span v-if="store.totalImageCount > 3 && store.createVideo">
                  + {{ Math.round(store.totalImageCount / videoStore.initFramerate * 100) / 100
}}s video
                </span>
                <span v-if="optionsStore.apiKey !== '0000000000' && optionsStore.apiKey !== "">
                  ({{ optionsStore.allowLargerParams === 'Enabled' ? store.canGenerate ? '✅ ' :
'❌ ' : '' }}{{ store.kudosCost.toFixed(2) }} kudos{{ store.canGenerate ? '' : ' required' }})
                </span>
              </span>
            </el-button>
```

```
            <el-button
              v-if="store.generating"
              type="danger"
              class="generate-cancel-btn"
              :disabled="store.cancelled"
              @click="() => {
                store.cancelled = true;
                videoStore.cancelGeneration();
              }"
            >Cancel</el-button>
          </div>
          <div class="image center-horizontal">
            <el-card
              class="center-both generated-image"
              v-loading="store.generating && !store.generatingVideo && uiStore.progress === 0 ?
{
                text: `Waiting for request(s) to upload${ellipsis}${' '.repeat(3 -
ellipsis.length)}`,
                background: 'rgba(0, 0, 0, 0.5)'
              } : false"
            >
              <div v-if="!store.generating && store.outputs.length == 0">
                <CustomCanvas v-if="/Inpainting/.test(store.generatorType)" />
              </div>
              <image-progress
                :est="videoStore.generationData.est === Infinity ? `Loading${ellipsis}` :
formatEST(videoStore.generationData.est)"
                :progress="videoStore.generationData.ratio * 100"
                :total="store.queue.reduce((prev, curr) => prev + (curr.params?.n ?? 0), 0)"
                :gathered="store.gatheredImages"
                @show-generated="uiStore.showGeneratedImages = true"
                v-if="!uiStore.showGeneratedImages && store.generatingVideo"
              >
                <template #status>
                  <div style="font-size: 18px">Video Status</div>
                  <div v-if="videoStore.generationData.ratio">
                    <span>Completed: {{ Math.floor(videoStore.generationData.ratio * 100) }}% -
</span>
                    <span>Processed: {{ videoStore.generationData.time }}s / {{
videoStore.generationData.duration }}s</span>
                  </div>
                  <div v-else>
                    <span>Pending video generation{{ellipsis}}</span>
                  </div>
                </template>
              </image-progress>
              <image-progress
```

```
                    :est="formatEST(Math.round((store.queueStatus?.wait_time as number) *
(pendingRequests.length + 1)))"
                    :progress="uiStore.progress"
                    :failed="store.queue.filter(el => el.failed).reduce((prev, curr) => prev +
(curr.params?.n ?? 0), 0)"
                    :total="store.queue.reduce((prev, curr) => prev + (curr.params?.n ?? 0), 0)"
                    :gathered="store.gatheredImages"
                    :pending-requests="pendingRequests"
                    :queue-status="store.queueStatus"
                    @show-generated="uiStore.showGeneratedImages = true"
                    v-if="!uiStore.showGeneratedImages && uiStore.progress"
                  />
                  <generated-carousel v-if="uiStore.showGeneratedImages && store.outputs.length !==
0" />
              </el-card>
            </div>
        </el-form>
    </div>
    <DialogList
        v-model="negativePromptLibrary"
        title="Negative Prompts"
        :list="store.negativePromptLibrary"
        empty-description="No negative prompts found"
        search-empty-description="Found no matching negative prompt(s) from your search."
        search-text="Search by prompt"
        deleteText="Delete preset"
        useText="Use preset"
        @use="negPrompt => store.negativePrompt = negPrompt"
        @delete="() => store.removeFromNegativeLibrary"
    />
</template>

<style>
:root {
    --sidebar-width: 70px
}

.small-btn {
    padding: 6px 8px;
    height: unset;
}

.generated-image {
    aspect-ratio: 1 / 1;
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: center;
```

```
    align-items: center;
}

.generated-image > .el-card__body {
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}

.form {
    padding-left: 20px;
    margin-left: var(--sidebar-width);
}

.main {
    grid-area: main;
    display: flex;
    justify-content: center;
}


.generate-cancel-btn {
    width: 80%;
}

.sidebar {
    grid-area: sidebar;
    max-width: 90%;
}

.image {
    grid-area: image;
}

.container {
    display: grid;
    height: 75vh;
    grid-template-columns: 50% 50%;
    grid-template-rows: 40px 95%;
    grid-template-areas:
        "sidebar main"
        "sidebar image";
}

@media only screen and (max-width: 1280px) {
    .generated-image > .el-card__body {
```

```
        height: 100%;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .generated-image {
        width: 80%;
        height: 100%;
    }

    .container {
        display: grid;
        height: 110vh;
        grid-template-rows: minmax(400px, 45vh) 65px 60%;
        grid-template-columns: 100%;
        gap: 10px;
        grid-template-areas:
            "image"
            "main"
            "sidebar";
    }

    .sidebar {
        max-width: 100%;
    }

    .main {
        flex-wrap: wrap;
        gap: 5px;
    }

    .main > * {
        width: 100% !important;
        margin: 0 !important;
    }

    .reset-btn {
        order: 1;
    }

    .generate-cancel-btn {
        order: 0;
    }
}

@media only screen and (max-width: 768px) {
    .generated-image {
```

```
      width: 100%;
      height: 100%;
    }

    .container {
      grid-template-rows: minmax(400px, 50vh) 65px 60%;
    }

    .form {
      padding-top: 20px;
      padding-left: 0;
      margin-left: 0;
    }
}

</style>

<script setup lang="ts">
import CustomImage from '@/components/CustomImage.vue';
import { useOutputStore, type ImageData } from '@/stores/outputs';
import { useUIStore } from '@/stores/ui';
import {
    ElEmpty,
    ElButton,
    ElMessageBox,
    ElPagination,
    ElPopover,
    ElIcon,
} from 'element-plus';
import {
    Delete,
    Download,
    Filter,
    CircleCheck,
    CircleCheckFilled,
    Sort,
} from '@element-plus/icons-vue';
import { computed } from 'vue';
import { useOptionsStore } from '@/stores/options';
import { onKeyStroke } from '@vueuse/core';
import { downloadMultipleImages } from '@/utils/download';
import { db } from '@/utils/db';
import { useWindowSize } from '@vueuse/core'
```

```
import LayoutOutlined from '@/components/icons/LayoutOutlined.vue';

const { width } = useWindowSize();

const store = useOutputStore();
const optionStore = useOptionsStore();
const uiStore = useUIStore();

function selectPage() {
   uiStore.selected = uiStore.selected.filter(el => !store.currentOutputs.map(el =>
el.id).includes(el));
   uiStore.selected = [...uiStore.selected, ...store.currentOutputs.map(el => el.id)];
   uiStore.multiSelect = true;
}

async function selectAll() {
   const allKeys = await db.outputs
      .toCollection()
      .primaryKeys() as number[];
   uiStore.selected = allKeys;
   uiStore.multiSelect = true;
}

function deselectPage() {
   uiStore.selected = uiStore.selected.filter(el => !store.currentOutputs.map(el =>
el.id).includes(el));
   if (uiStore.selected.length === 0) uiStore.multiSelect = false;
}

function deselectAll() {
   uiStore.selected = [];
   uiStore.multiSelect = false;
}

const confirmDelete = () => {
   ElMessageBox.confirm(
      `This action will permanently delete ${uiStore.selected.length} images.
Continue?`,
      'Warning',
      {
         confirmButtonText: 'OK',
```

```
            cancelButtonText: 'Cancel',
            type: 'warning',
        }
    )
        .then(() => {
            store.deleteMultipleOutputs(uiStore.selected);
        })
}

onKeyStroke(['a', 'A', 'ArrowLeft'], uiStore.openModalToLeft)
onKeyStroke(['d', 'D', 'ArrowRight'], uiStore.openModalToRight)

async function bulkDownload() {
    downloadMultipleImages(uiStore.selected);
}

const splitList = computed(() => {
    let columns = 2;
    if (width.value > 1440) {
        columns = 6;
    } else if (width.value > 1280) {
        columns = 5;
    } else if (width.value > 768) {
        columns = 4;
    } else if (width.value > 480) {
        columns = 3;
    }
    const result = [];
    for (let i = 0; i < columns; i++) {
        const column = [];
        for (let j = i; j < store.currentOutputs.length; j += columns) {
            column.push(store.currentOutputs[j]);
        }
        result.push(column);
    }
    return result;
})
</script>

<template>
    <div class="images-top-bar">
```

```
      <div class="options">

      </div>
      <el-pagination
        layout="prev, pager, next"
        :total="store.outputsLength"
        :page-size="optionStore.pageSize"
        :current-page="store.currentPage"
        @update:current-page="(val: number) => store.currentPage = val"
        hide-on-single-page
        v-if="optionStore.pageless === 'Disabled'"
      />
      <div class="center-both" v-if="uiStore.multiSelect" style="gap: 12px">
        <div>{{ uiStore.selected.length }} selected</div>
        <el-button type="danger" @click="confirmDelete" :icon="Delete"
plain>Delete</el-button>
        <el-button type="success" @click="bulkDownload" :icon="Download" plain
style="margin: 0">Download</el-button>
      </div>
    </div>
    <div v-if="store.outputsLength != 0">
      <div style="display: flex; gap: 8px" v-if="store.currentLayout === 'dynamic'">
        <div v-for="(items, index) in splitList" :key="index" style="flex: 1 1 0%">
          <CustomImage
            v-for="image in items"
            :key="image.id"
            :image-data="image"
            style="margin-bottom: 8px;"
          />
        </div>
      </div>
      <div class="images" v-if="store.currentLayout === 'grid'">
        <CustomImage
          v-for="image in store.currentOutputs"
          :key="image.id"
          :image-data="image"
          style="width: 200px; height: 200px"
        />
      </div>
    </div>
    <div v-if="store.outputsLength == 0">
```

```
        <el-empty description="No Images Found" />
    </div>
</template>

<style scoped>
.images {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 8px;
    width: 100%;
}

.selected {
    color: var(--el-color-primary);
    text-decoration: underline;
    background-color: #262626;
}

.btn-select {
    width: 48px;
    height: 32px;
}

.options {
    display: flex;
    align-items: center;
    gap: 8px;
}

.options > * {
    margin: 0;
}

.images-top-bar {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 12px;
}
```

```
.images-top-bar > * {
   width: 100%;
   display: flex;
   justify-content: center;
   text-align: center;
   white-space: nowrap;
   flex-grow: 0;
}

.bottom-pagination {
   display: none;
}

@media only screen and (max-width: 768px) {
   .images-top-bar {
      flex-wrap: wrap;
   }

   .bottom-pagination {
      margin-bottom: 50px;
      display: flex;
   }
}
</style>
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СТВОРЕННЯ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЕНЕРАТИВНОЇ НЕЙРОННОЇ МЕРЕЖІ»

Виконала: студентка 2-го курсу, групи ЗКН-23м
спеціальності 122 – «Комп'ютерні науки»

Верба Г. Р.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)

«05» 12 2024 р.

Вінниця ВНТУ – 2024 рік

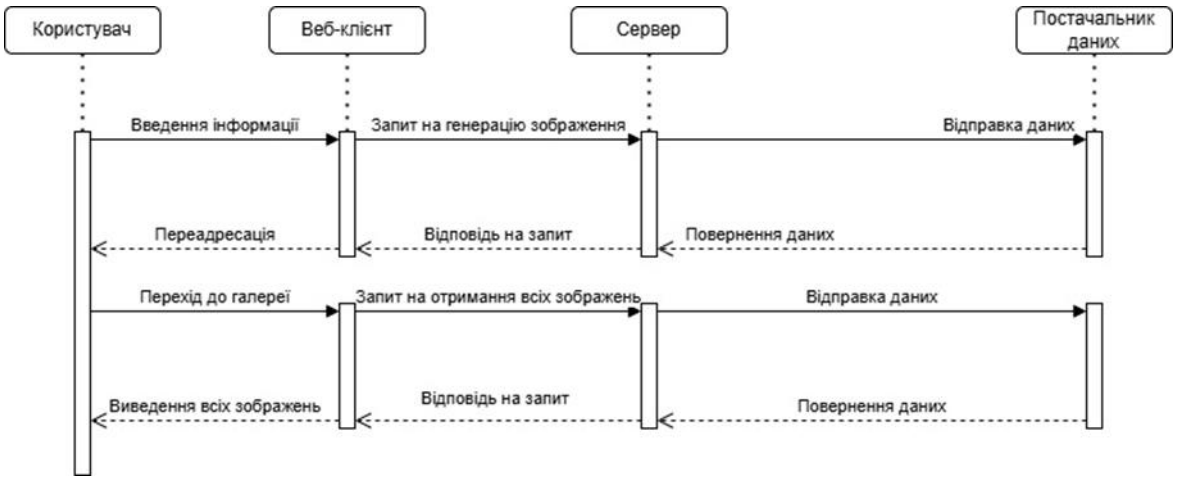

Рисунок В.1 – UML-діаграма послідовності взаємодії головних модулів інформаційної технології створення зображень за допомогою генеративної нейронної мережі

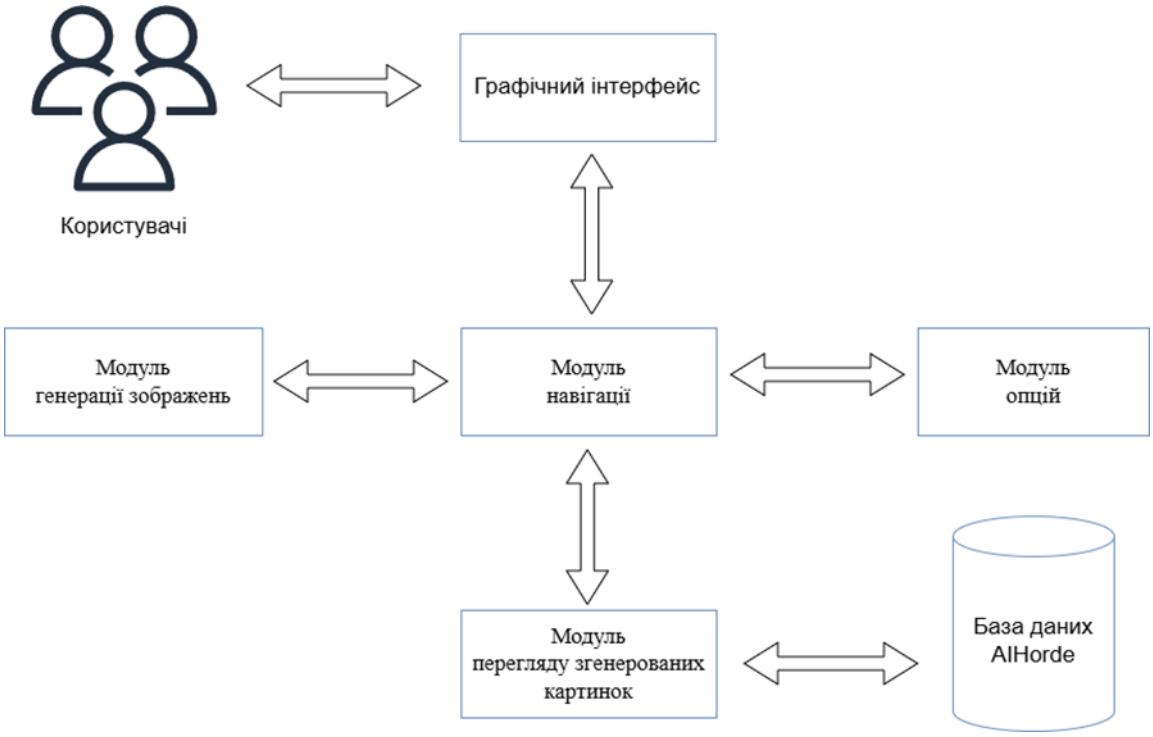

Рисунок В.2 – Структурна схема програмного модуля інформаційної технології створення зображень за допомогою генеративної нейронної мережі

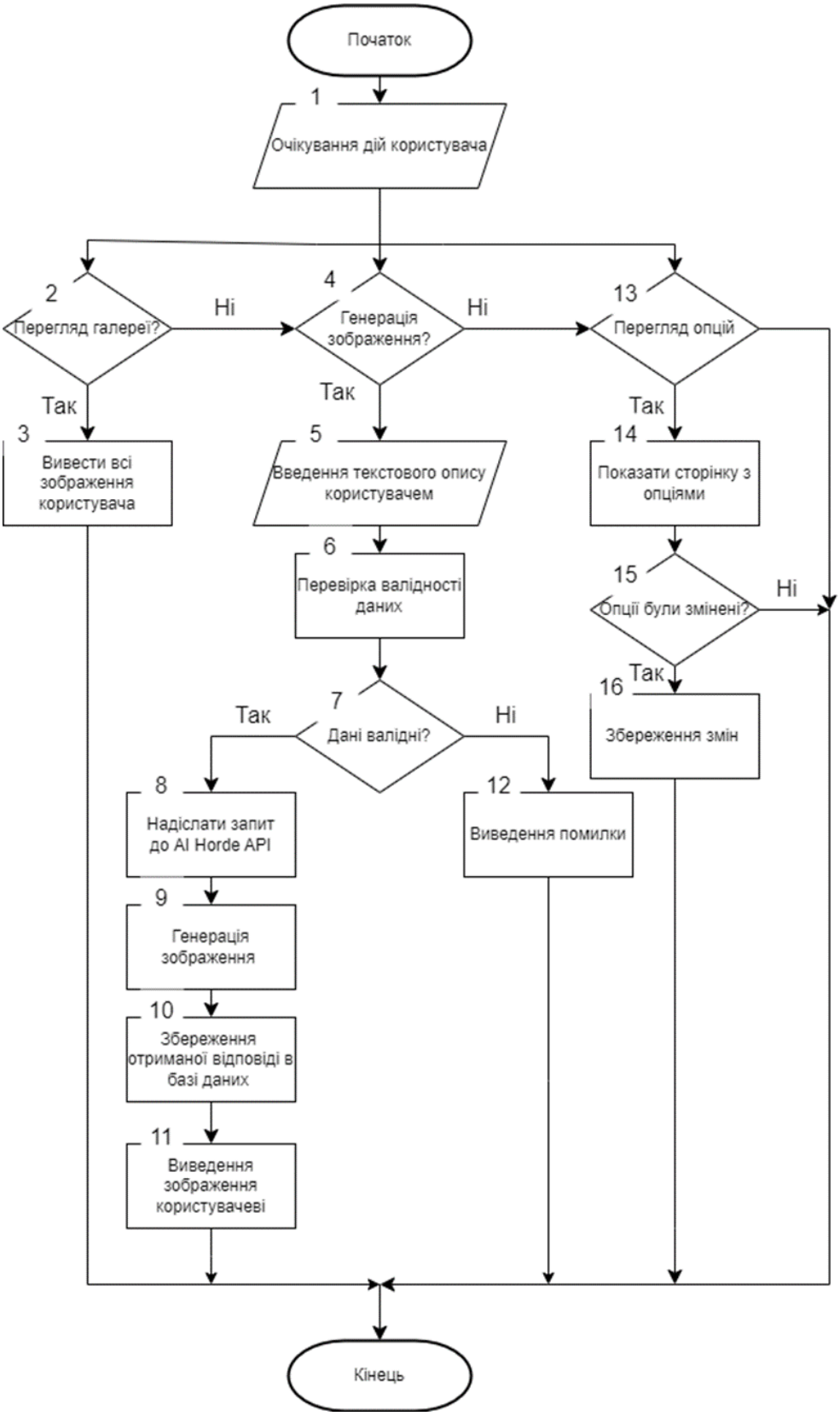

Рисунок В.3 – Схема алгоритму роботи інформаційної технології створення зображень за допомогою генеративної нейронної мережі

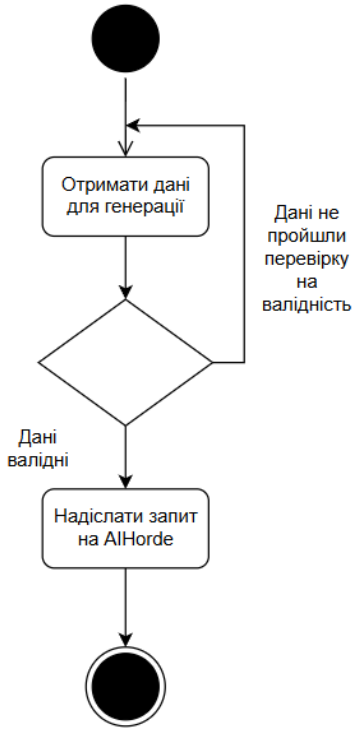

Рисунок В.4 – Діаграма діяльності серверу під час процесу генерації зображення

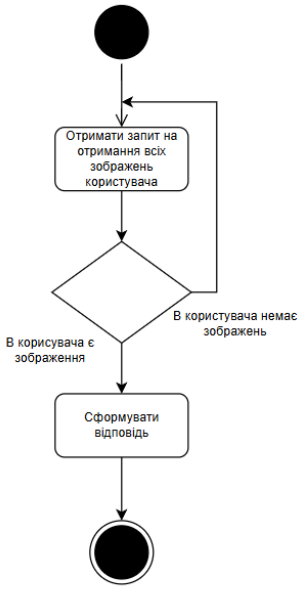

Рисунок В.5 – Діаграма діяльності серверу під час процесу отримання зображень користувача для галереї

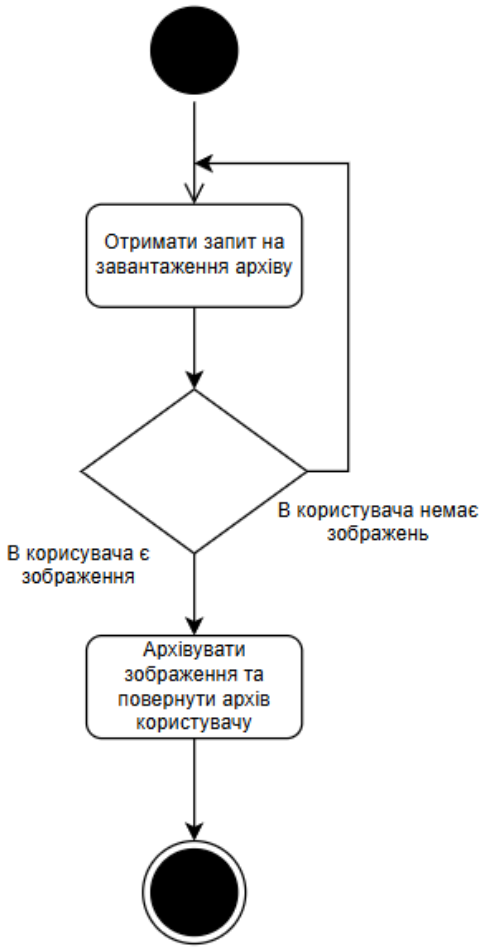

Рисунок В.6 – Діаграма діяльності серверу під час процесу скачування користувачем архіву

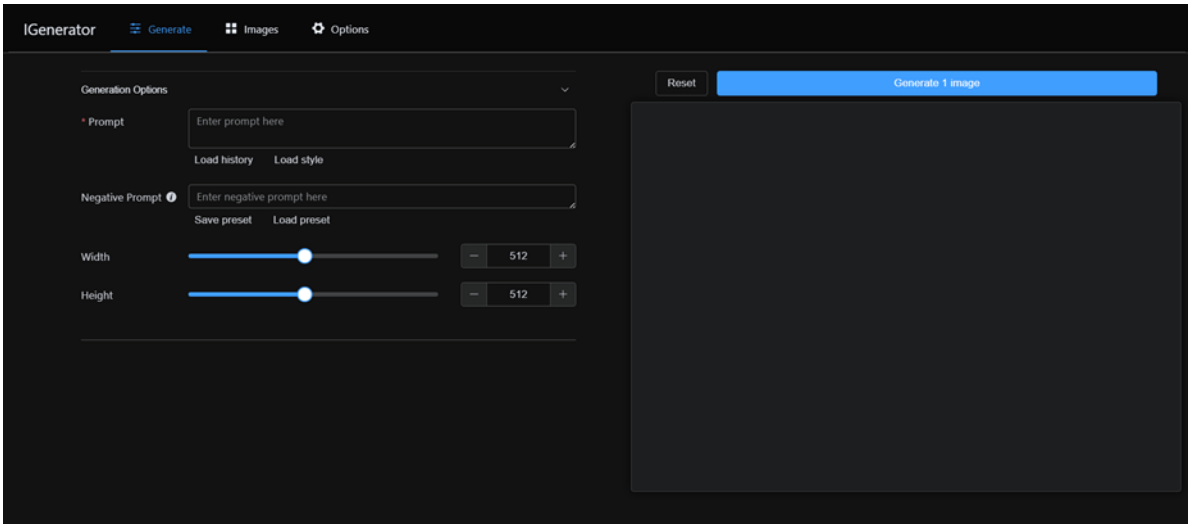

Рисунок В.7 – Загальний вигляд інтерфейсного вікна сторінки генерації

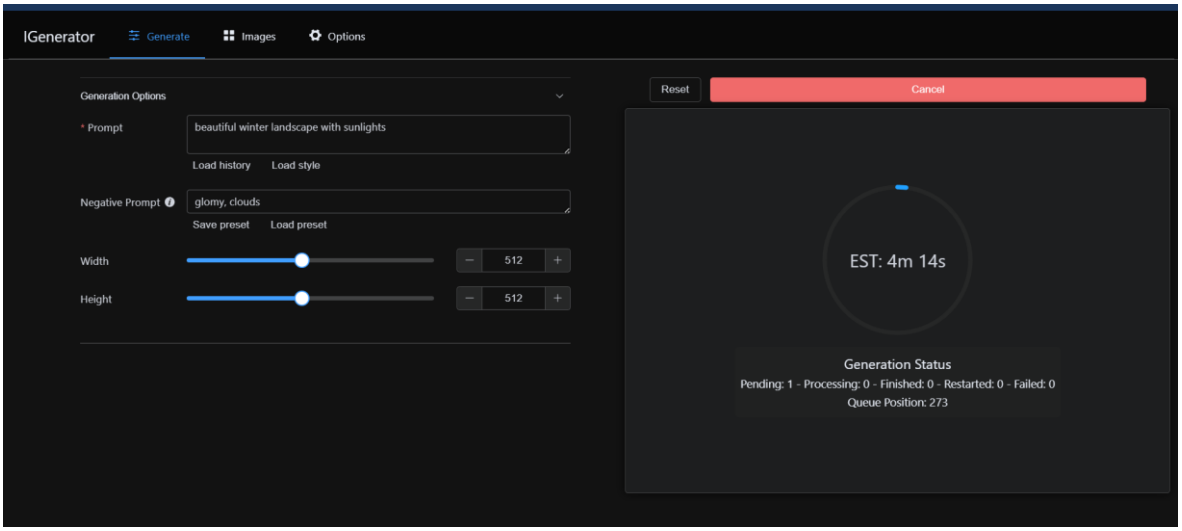

Рисунок В.8 – Загальний вигляд інтерфейсного вікна сторінки генерації під час генерації

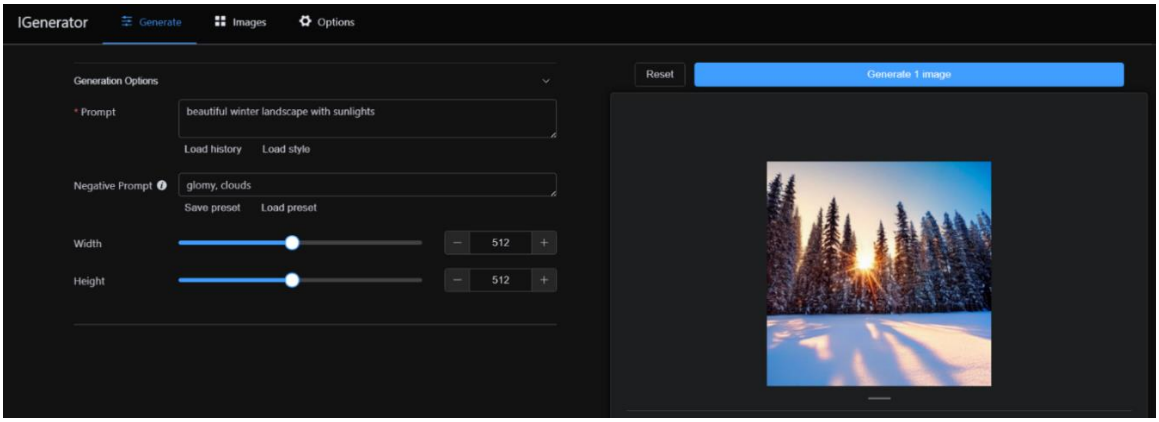

Рисунок В.9 – Загальний вигляд інтерфейсного вікна сторінки генерації з готовим зображенням

Додаток Г (довідниковий)

Інструкція користувача

1. Після запуску WEB-ресурсу відкривається головна сторінка (рис. Г.1).

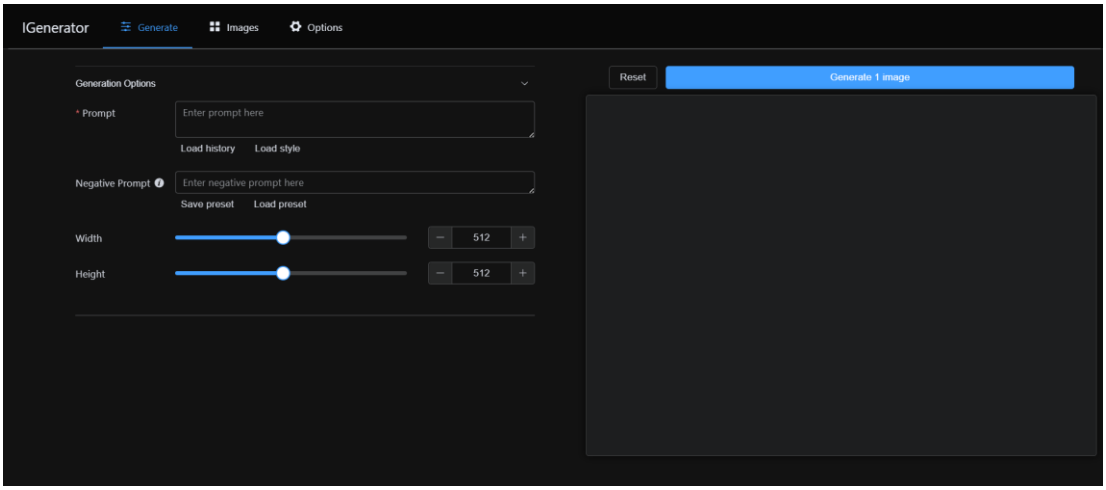

Рисунок Г.1 – Загальний вигляд інтерфейсного вікна WEB-ресурсу після його запуску

2. Спочатку потрібно описати яким має бути зображення, за бажанням встановити його розміри та описати яким зображення НЕ має бути. Потім натиснути на кнопку «Generate image», після чого обране зображення відобразиться в правій частині WEB-ресурсу (рис. Г.2).

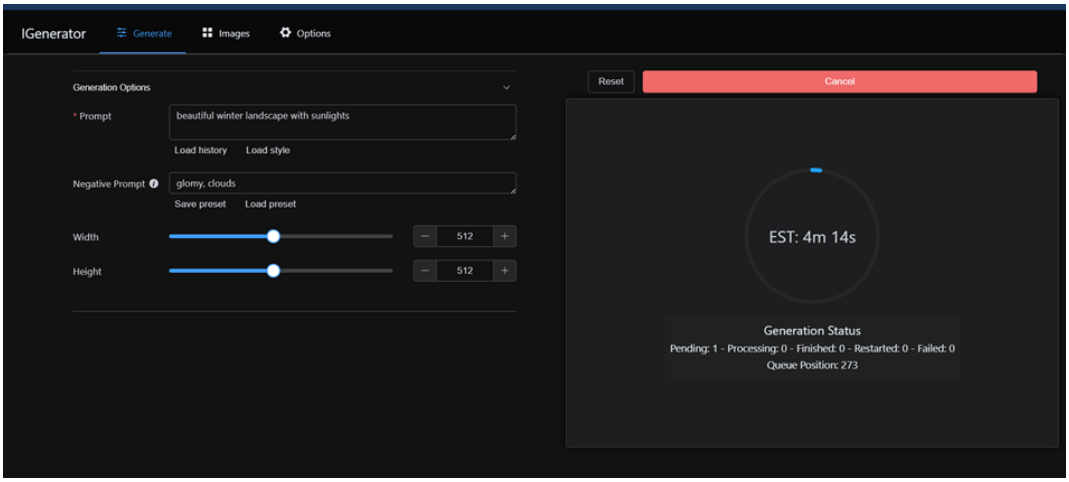

Рисунок Г.2 – Загальний вигляд інтерфейсного вікна WEB-ресурсу після запуску генерації

3. Дочекатися результату (рис. Г.3).

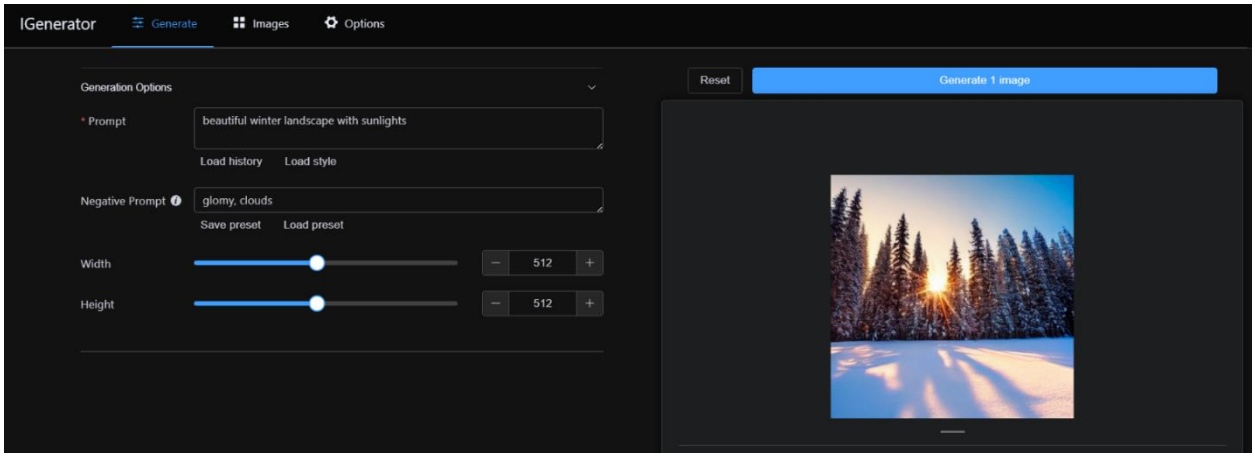

Рисунок Г.3 – Загальний вигляд інтерфейсного вікна WEB-ресурсу після успішної генерації

4. За бажанням відвідати сторінку Images щоб переглянути всі роботи (рис. Г.4).

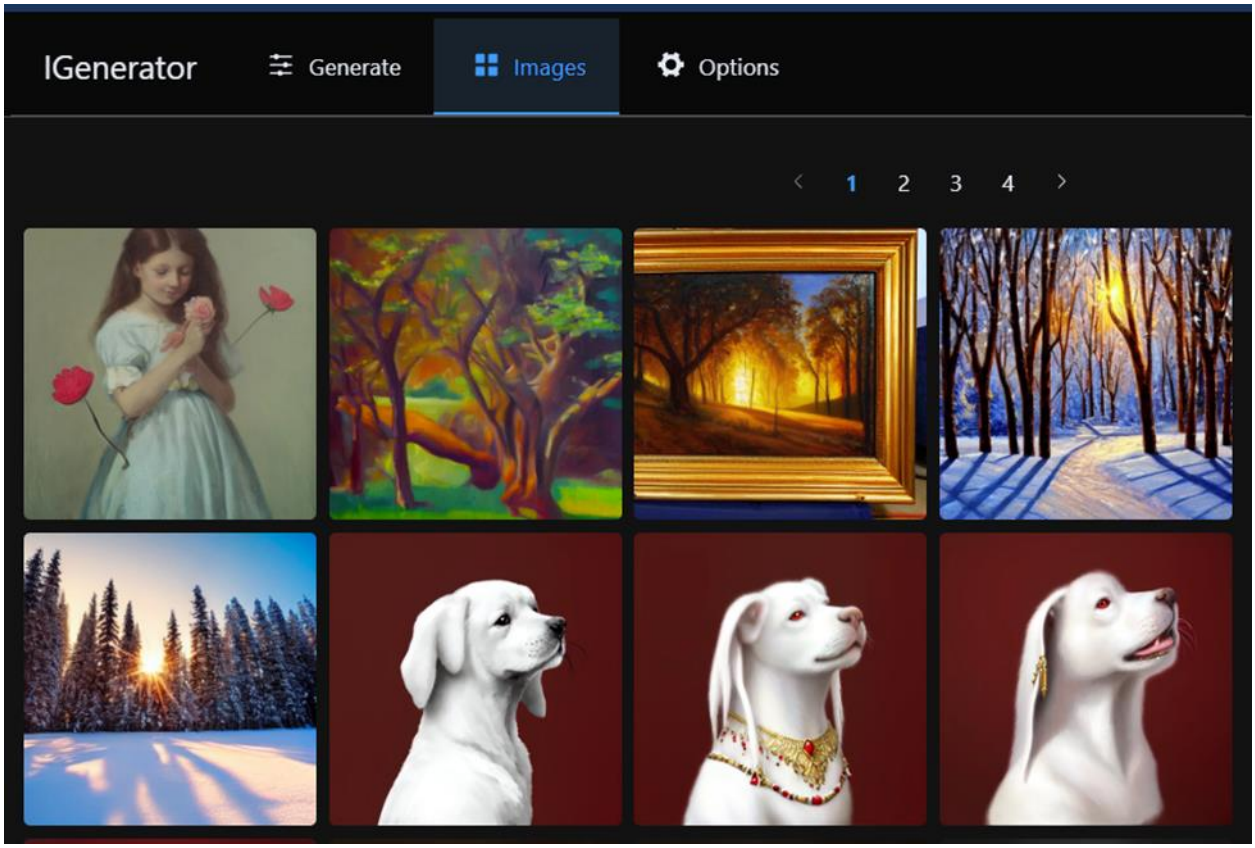

Рисунок Г.4 – Загальний вигляд інтерфейсного вікна сторінки Images

5. При бажанні завантажити всі картинки до себе на пристрій або завантажити свої картинки у WEB-ресурс (рис. Г.5).

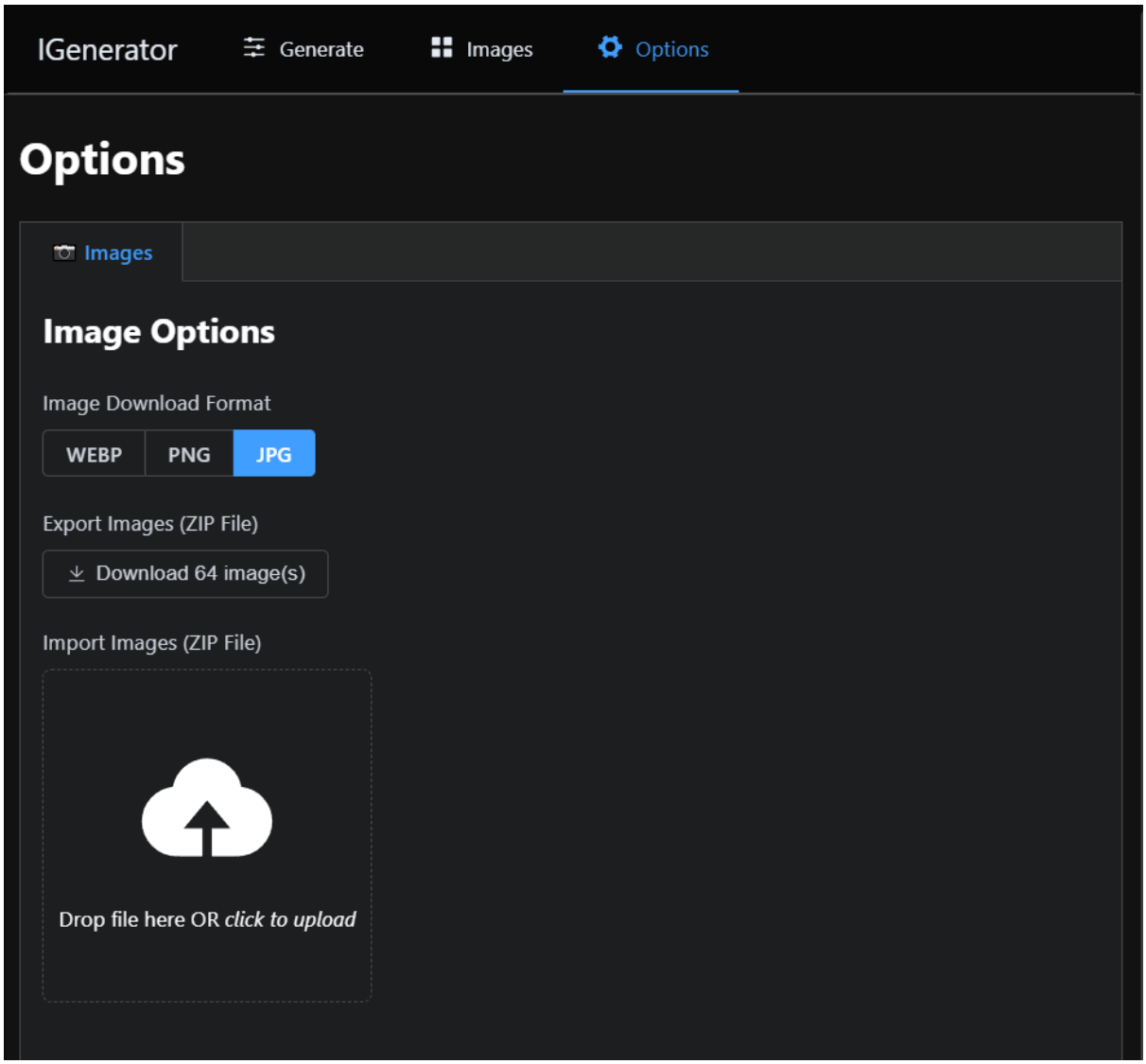

Рисунок Г.5 – Загальний вигляд інтерфейсного вікна сторінки Options