Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Розробка веб-додатку з використанням генеративного штучного інтелекту для створення анімованих зображень»

Виконав: студент групи К23-2м
Спеціальність 122 «Комп'ютерні науки»
Корюк Р. В.
(прізвище та ініціали)

Керівник: к. ф.-м. н., доц. Лебідь О. Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент: ТОВ "МАСТ ПЕЙ"
(місто роботи)
начальник відділу автоматизації процесів
(посада)
Гарбуз О. О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Корюк Р. В. Розробка веб-додатку з використанням генеративного штучного інтелекту для створення анімованих зображень.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2024.

Кваліфікаційна робота магістра присвячена розробці додатку для створення генеративного анімованого контенту через штучний інтелект. Метою розробки проекту є створення інструменту, який дозволяє генерувати унікальний контент під запити користувача.

У роботі проведено аналіз сучасних технологій розробки та інструментів для створення генеративного штучного інтелекту, та веб-додатка, а також вивчення вимог та особливостей процесу генерації анімованих зображень. На основі отриманих знань, реалізовано сервер з генеративним штучним інтелектом та веб-додатку, взаємопов'язаних між собою для створення зручного інтерфейсу користувача.

У ході дослідження вирішені такі завдання: аналіз сценаріїв використання, проектування архітектури системи, розробка та тестування функціональності генеративного штучного інтелекту та веб-додатку.

Ключові слова: генеративний штучний інтелект, Python FastAPI, генерація анімованих зображень, веб-додаток, Angular framework.

Список публікацій здобувача:

1. Корюк Р. В. Зарубіжний досвід публічного регулювання зовнішньоекономічної діяльності. Світовий досвід публічного регулювання зовнішньоекономічної діяльності: митна безпека та протидія корупції : матеріали міжнародної науково-практичної конференції (31 травня 2024р., м. Дніпро). Дніпро: Університет митної справи та фінансів, 2024, с. 70-72.

ABSTRACT

Koriuk, R.V. Development of a web application using generative artificial intelligence for creating animated images.

Qualification work for obtaining a master’s degree in the specialty 122 Computer Science. – University of Customs and Finance, Dnipro, 2024.

The master's qualification work is devoted to the development of an application for creating generative animated content through artificial intelligence. The goal of the project development is to create a tool that allows you to generate unique content based on user requests.

The paper analyzes modern development technologies and tools for creating generative artificial intelligence and a web application, as well as studying the requirements and features of the process of generating animated images. Based on the knowledge gained, a server with generative artificial intelligence and a web application interconnected to create a convenient user interface were implemented.

In the course of the research, the following tasks were solved: analysis of usage scenarios, design of the system architecture, development and testing of the functionality of generative artificial intelligence and a web application.

Keywords: generative artificial intelligence, Python FastAPI, generation of animated images, web application, Angular framework.

List of publications:

1. Koryuk R. V. Foreign experience of public regulation of foreign economic activity. World experience of public regulation of foreign economic activity: customs security and anti-corruption: materials of the international scientific and practical conference (May 31, 2024, Dnipro). Dnipro: University of Customs and Finance, 2024, p. 70-72.

ЗМІСТ

ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДЛЯ ВИКОРИСТАННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ 10
1.1 Аналіз використання генеративного штучного інтелекту та загальні проблеми використання 10
1.2 Визначення технічних, безпекових та юридичних вимог до інтеграції генеративного штучного інтелекту 21
1.3 Постанова завдання 25
1.4 Висновки до першого розділу 26
РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ З ГЕНЕРАТИВНИМ ШТУЧНИМ ІНТЕЛЕКТОМ, АНАЛІЗ ТА ВИБІР МЕТОДІВ ВИРІШЕННЯ 28
2.1. Дослідження існуючих інструментів для створення штучного інтелекту з інтегрованою моделлю 28
2.2 Аналіз та вибір методів рішення 34
2.3 Дослідження проектування системи з генеративним штучним інтелектом із використанням досліджених технологій 39
2.4 Висновки до другого розділу 40
РОЗДІЛ 3. РОЗРОБКА ВЕБ-ДОДАТКУ З ВИКОРИСТАННЯМ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ СТВОРЕННЯ АНІМОВАНИХ ЗОБРАЖЕНЬ 43
3.1 Створення системи з генеративним штучним інтелектом. 43
3.2 Реалізація серверу використовуючи інструмент Python Fast API 49
3.3 Розробка дизайну та реалізація додатку 51
3.4 Висновки до третього розділу 55
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61
ДОДАТОК А 64

ВСТУП

На сучасному етапі розвитку суспільства інформаційні технології займають ключову роль у всіх аспектах людського життя. Завдяки стрімкому прогресу цифрових технологій спостерігається значне зростання їхнього впливу на різні сфери діяльності – від побуту до бізнесу. Автоматизація процесів стала невід'ємним елементом успішного функціонування організацій, адже вона дозволяє підвищити продуктивність праці, мінімізувати помилки та знизити витрати часу.

Одним із сучасних напрямів автоматизації є застосування штучного інтелекту, який відкриває нові горизонти у виконанні рутинних завдань і допомагає вирішувати складніші завдання, що потребують аналітичного підходу. Генеративний штучний інтелект, зокрема, викликає все більший інтерес у науковій спільноті та серед підприємців завдяки його здатності створювати унікальний контент, аналізувати дані та пропонувати інноваційні рішення.

У сучасних умовах, коли швидкість, якість і ефективність є ключовими вимогами до будь-якого процесу, автоматизація стає обов'язковим компонентом будь-якої успішної діяльності. Проте традиційні методи автоматизації не завжди дозволяють досягти бажаного результату через обмеження в технологіях або недостатню інтеграцію інноваційних підходів. У зв'язку з цим дедалі більшої популярності набуває використання генеративних моделей штучного інтелекту, які демонструють унікальну здатність адаптуватися до потреб користувачів та створювати високоякісний контент із мінімальними зусиллями.

У рамках цього дослідження особлива увага приділяється розробці веб-додатків, які не лише автоматизують певні аспекти роботи, а й інтегрують новітні можливості штучного інтелекту для вирішення прикладних завдань. Веб-додатки стають зручним інструментом для багатьох користувачів, адже вони забезпечують доступність, універсальність і простоту у використанні, дозволяючи залучати широке коло спеціалістів і непрофесіоналів до роботи з

інноваційними технологіями.

Об'єктом дослідження є процеси автоматизації створення контенту за допомогою генеративних моделей штучного інтелекту, які включають інтеграцію новітніх технологій штучного інтелекту в робочі процеси. Особливу увагу приділено аналізу сучасних підходів до автоматизації, які дозволяють значно скоротити час та ресурси на створення якісного графічного контенту. Застосування генеративних моделей відкриває нові можливості для розробників, дизайнерів і маркетологів, які працюють у сфері цифрових технологій.

Предметом дослідження є методи розробки та реалізації веб-додатків з інтегрованим генеративним штучним інтелектом для створення анімованих зображень. У межах дослідження вивчаються підходи до побудови ефективної архітектури веб-додатків, вибору оптимальних інструментів розробки, а також способи забезпечення високої якості кінцевого продукту. Предмет дослідження охоплює аспекти взаємодії користувачів із системою, питання безпеки та адаптації технологій до потреб сучасного ринку.

Основною метою роботи є розробка веб-додатку, який інтегрує технології генеративного штучного інтелекту для автоматизації створення анімованих графічних матеріалів. Для досягнення цієї мети необхідно виконати такі завдання:

- провести аналіз сучасних технологій у сфері ШІ та створення анімаційних зображень;
- визначити вимоги до функціональності веб-додатку та створити технічне завдання;
- розробити оптимальну архітектуру системи, яка забезпечить ефективну роботу з користувачами та обробку графічних даних;
- реалізувати функціонал генерації анімованих зображень з використанням алгоритмів ШІ;
- забезпечити інтерактивний інтерфейс, що дозволить користувачам легко взаємодіяти із системою;
- провести тестування системи для перевірки відповідності вимогам та

усунення можливих недоліків;

- підготувати рекомендації для впровадження системи у реальне середовище.

Результати проєкту можуть знайти застосування в різних сферах, зокрема у дизайні, маркетингу та творчих індустріях, де потреба у швидкому створенні якісного контенту є надзвичайно актуальною. Особливо це стосується сучасних медіа-компаній, які постійно працюють з великими обсягами графічних матеріалів для соціальних мереж, рекламних кампаній та мультимедійних проектів. Використання генеративного штучного інтелекту дозволяє значно скоротити час на створення анімованих зображень та підвищити їхню візуальну привабливість.

У сфері дизайну цей інструмент може стати незамінним для створення концептів, прототипів та готових продуктів, адаптованих до індивідуальних потреб клієнтів. У маркетингових кампаніях можливості швидкої генерації унікальних матеріалів дозволяють оперативно реагувати на нові тренди та запити ринку. Творчі індустрії, такі як кінематограф, геймдизайн та віртуальна реальність, також можуть використовувати результати роботи генеративного ШІ для створення унікальних персонажів, сцен або інтерактивного контенту, що знижує витрати і пришвидшує процес розробки.

Крім того, автоматизація створення контенту відкриває нові можливості для малого та середнього бізнесу, які не завжди мають ресурси для найму великих команд дизайнерів чи аніматорів. Завдяки генеративному ШІ такі підприємства отримують доступ до високоякісного контенту за значно нижчою вартістю. Це робить інструмент універсальним і затребуваним у найрізноманітніших сферах.

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел, що складається з 31 джерела, 1 додатку. Обсяг роботи 61 сторінка, 14 рисунків.

РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ ТА ВИМОГ ДЛЯ ВИКОРИСТАННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Аналіз використання генеративного штучного інтелекту та загальні проблеми використання

На сьогоднішній день використання генеративного штучного інтелекту (ГШІ) набуває все більшої актуальності в різних сферах діяльності, включаючи дизайн, мистецтво, медіа та маркетинг. Принцип роботи ГШІ полягає у здатності алгоритму створювати новий, унікальний контент – зображення, тексти, музичні треки чи навіть анімовані відео – на основі наданих даних та навчених моделей. Такий підхід дозволяє автоматизувати процеси творчості та знизити залежність від людського фактору, що особливо важливо у сучасних умовах динамічного розвитку технологій та збільшення обсягів інформації, яку необхідно обробляти.

Актуальність генеративного штучного інтелекту полягає в його здатності створювати нові форми медіаконтенту, які можуть бути використані для різноманітних цілей: від створення унікальних рекламних матеріалів та ілюстрацій до генерації анімованих персонажів для відеоігор чи віртуальної реальності. Оскільки світовий ринок диджитал-контенту стрімко зростає, технології генерації зображень та анімацій набувають все більшого значення. Генеративні моделі дозволяють отримувати візуально привабливий матеріал із мінімальним втручанням людини, що знижує витрати часу та ресурсів.

У контексті веб-додатків, інтеграція ГШІ відкриває нові можливості:

- автоматизація творчих процесів;
- індивідуалізація контенту;
- швидкість оновлення медіа контенту.

Розробники можуть надати користувачам інструменти для швидкого створення анімованих зображень без глибоких знань у сфері графічного дизайну чи анімації. Застосування генеративних моделей спрощує процес, оскільки

користувачеві достатньо вибрати базові параметри чи надати референс, а система самостійно згенерує кінцевий результат.

Завдяки генеративному штучному інтелекту можна створювати унікальні матеріали для кожного користувача, враховуючи його смаки, потреби чи брендові вимоги. Це особливо актуально для маркетингових кампаній, коли необхідно швидко генерувати велику кількість унікальних банерів, ілюстрацій чи роликів.

Генеративні моделі дозволяють оперативно оновлювати чи адаптувати контент залежно від змін на ринку, нових трендів або персональних переваг користувачів. Це підвищує конкурентоспроможність та забезпечує швидке реагування на потреби аудиторії.

Попри значний потенціал застосування генеративного штучного інтелекту, існує низка загальних проблем, пов’язаних з його використанням.

Сучасні генеративні моделі можуть створювати високоякісні візуальні матеріали, все ж є ризик отримання штучних чи нереалістичних результатів. Деякі згенеровані зображення можуть містити артефакти, спотворення чи непередбачувані елементи, що знижує їх цінність для користувача. Для підвищення якості необхідно вдосконалювати моделі, збільшувати обсяги та різноманітність навчальних даних, а також впроваджувати додаткові фільтри та метрики оцінювання якості.

Створення анімованих зображень за допомогою генеративного штучного інтелекту вимагає значних обчислювальних потужностей, особливо при генерації високоякісного контенту у великих обсягах. Це може призвести до підвищення витрат на інфраструктуру, необхідність використання хмарних платформ та оптимізації архітектури систем. Неврахування цих аспектів може негативно позначитися на швидкості роботи веб-додатку та досвіді користувачів.

Генерація контенту без чіткого контролю може призвести до появи некоректних, образливих або навіть незаконних матеріалів. Особливо актуальними є питання авторського права, оскільки згенеровані зображення можуть мати схожість із роботами інших авторів. Необхідно враховувати

юридичні обмеження та впроваджувати механізми фільтрації, модерації та ліцензування створеного контенту.

Генеративні моделі потребують великих обсягів даних для навчання. Якщо ці дані містять конфіденційну або чутливу інформацію, існує ризик її ненавмисного розголошення. Забезпечення безпеки даних користувачів та захист інтелектуальної власності стає критично важливим завданням.

Занадто реалістична генерація зображень (зокрема облич) може призвести до створення глибинних фейків (deepfake) чи маніпулятивних матеріалів, які вводять користувачів в оману. Це ставить питання про відповідальність розробників та регуляторів, необхідність створення чітких етичних норм та правил користування технологіями ГШІ.

Незважаючи на ці проблеми, застосування генеративного штучного інтелекту у створенні анімованих зображень у веб-додатках має значний потенціал. Завдяки можливості швидкої та індивідуалізованої генерації контенту, інтеграція ГШІ може оптимізувати робочі процеси, розширити функціональність продуктів та підвищити привабливість послуг. Проте розробники та бізнес-організації мають ретельно аналізувати ризики, впроваджувати належні заходи безпеки, дотримуватися етичних принципів та юридичних вимог, а також постійно вдосконалювати моделі та процеси їх впровадження.

Для створення простої системи з генерацією анімованих зображень у дипломному проекту буде використовуватись локальний сервер і його функціонал. Актуальність розробки подібної системи полягає у потребі швидкому та зручному способі генерації унікального анімованого контенту. У сучасному світі бізнес-процеси стають все більш швидкими та складними, а ефективне та швидке створення контенту є ключовим фактором для успіху. Традиційні методи створення контенту, такі як створення векторних зображень завдяки продукції інструментів Adobe можуть бути дуже ресурс затратними. Також актуальністю теми є широке поширення використання таких ШІ як, ChatGPT, Midjourney, Stable Diffusion, серед користувачів як на особистому, так

і на комерційному рівні, ШІ надає зручні рішення певних задач, що робить його привабливим для впровадження в бізнес-середовищі. Використання ГШІ для генерації анімацій може спростити процеси створення контенту. Крім того, загальні тенденції у сфері автоматизації та цифровізації бізнесу підкреслюють важливість впровадження таких інструментів для оптимізації робочих процесів та підвищення продуктивності. Автоматизація стає необхідним елементом сучасного підприємництва, що дозволяє зменшити рутинні операції, знизити ризики та покращити якість та ефективність роботи.

ШІ сам по собі не передбачає використання функціоналу для автоматизації певних процесів, так як він очікує звернень до моделі для вирішення певних процесів, але це дуже легко вирішується завдяки написання власних скріптів, які через певний час запускають методи для виконання робочих процесів.

Створення штучного інтелекту на сьогоднішній день значною мірою спирається на використання вже навчених моделей, які раніше були сформовані на величезних масивах даних. Ці моделі не просто здатні виконувати окремі, чітко визначені завдання, а й продемонструвати надзвичайну гнучкість у застосуванні. Завдяки цьому вони стали потужним інструментом, що дозволяє масштабно впроваджувати автоматизацію, оптимізувати робочі процеси та генерувати новий, унікальний контент. Такий підхід вирізняється від класичних методів розробки програмного забезпечення, де функціонал систем закладався переважно вручну, а кожне нове завдання вимагало довготривалої підготовки. Натомість використання наперед навчених моделей дає змогу суттєво знизити вартість та скоротити час впровадження інтелектуальних рішень у різних сферах.

Однією з ключових особливостей уже навчених моделей є їхня здатність аналізувати дані будь-якої складності та масштабу. Зокрема, йдеться про великі обсяги структурованої та неструктурованої інформації – тексти, зображення, відео, аудіозаписи, дані сенсорних пристроїв. Застосування методів машинного навчання дозволяє моделі виявляти закономірності та тренди, які раніше залишалися поза межами людського сприйняття. Як наслідок, компанії та організації отримують змогу точніше прогнозувати динаміку ринку, визначати

конкурентні переваги, аналізувати поведінку споживачів та максимально ефективно використовувати свої ресурси. Це особливо важливо в час, коли швидкі зміни в економіці, технологіях та соціальній сфері змушують підприємства постійно адаптуватися та переглядати стратегії.

Окрім аналітичних можливостей, вже навчені моделі відіграють значну роль у побудові та оптимізації бізнес-процесів. Маючи здатність аналізувати навіть найдрібніші елементи виробничих ланцюгів, логістичних систем, маркетингових кампаній чи управлінських процесів, вони допомагають організаціям удосконалювати внутрішню структуру. Наприклад, використання штучного інтелекту може сприяти виявленню «вузьких місць» у виробництві, покращенню якості продукції, зниженню собівартості та раціоналізації розподілу ресурсів. Моделі також дозволяють прогнозувати вплив зовнішніх чинників – змін на ринку сировини, коливань попиту, законодавчих нововведень – на стабільність і ефективність бізнесу. Завдяки цьому менеджери можуть ухвалювати більш зважені та далекоглядні рішення, базуючись не лише на власному досвіді чи інтуїції, а й на чітких числових показниках.

Ще одним важливим аспектом залучення вже навчених моделей є менеджмент процесів. Управління проектами, командами, запасами та цілими організаційними системами стає набагато ефективнішим, коли у розпорядженні управлінців є інтелектуальні інструменти, здатні оцінювати продуктивність, розподіляти завдання, контролювати терміни й визначати найбільш пріоритетні ділянки роботи. Замість того, щоб покладатися на статичні плани та людський фактор, можна використовувати системи, які динамічно адаптуються до змін. Наприклад, якщо в якийсь момент відбувається різке збільшення попиту на певну продукцію, інтелектуальна модель може швидко запропонувати перерозподіл ресурсів, залучення додаткового персоналу чи навіть перехід на інший ланцюг постачання, щоб виконати замовлення вчасно й без втрати якості.

Крім аналізу та оптимізації внутрішніх процесів, штучний інтелект дедалі частіше застосовується для генерації контенту. Уже навчені моделі здатні створювати тексти, зображення, музичні твори, відеоролики, тривимірні моделі

чи навіть інтерактивні середовища. Цей прорив відкриває нові горизонти у сфері дизайну, реклами, медіа та мистецтва. Замість ручної, кропіткої роботи графічних дизайнерів або копірайтерів моделі можуть миттєво генерувати тисячі варіантів логотипів, рекламних слоганів, сюжетів, анімаційних кліпів чи інфографічних матеріалів. Зрозуміло, це не означає, що творчі професії перестануть існувати – навпаки, завдяки інструментам штучного інтелекту креативні фахівці можуть звільнитися від рутинних завдань, зосередившись на концептуальних аспектах та перевірці якості. Таке поєднання людської майстерності з машинною продуктивністю дозволяє створювати нові, унікальні продукти, які краще відповідають запитам і потребам аудиторії.

Варто підкреслити, що застосування вже навчених моделей не зводиться до вузького технічного аспекту. Воно має вагомі соціально-економічні наслідки. Штучний інтелект впливає на ринки праці, структуру кваліфікацій, освітні системи. Підприємства, які впроваджують інтелектуальні моделі, зазвичай отримують конкурентні переваги, можуть швидше реагувати на зовнішні виклики, пропонувати клієнтам більш якісні та персоналізовані продукти та послуги. З іншого боку, компанії, що ігнорують такі можливості, ризикують втратити свої позиції, залишившись позаду технологічно підкованих суперників. До того ж, упровадження ШІ потребує обміркованого підходу: які саме дані слід використати для навчання моделей, як захистити конфіденційну інформацію, які етичні норми застосувати для запобігання дискримінації та неправильного використання технологій.

Усвідомлюючи комплексність цього питання, розробники та дослідники штучного інтелекту дедалі частіше зосереджуються на прозорості та зрозумілості моделей. Зокрема, вирішується завдання пояснюваності алгоритмів, коли користувач може зрозуміти логіку прийнятих рішень. Це є критичним у випадках, коли моделі використовуються у сфері охорони здоров’я, фінансів або державного управління. Прозорі та інтерпретовані моделі дозволяють підвищити довіру до систем на основі ШІ, адже користувачі та контролюючі органи можуть відстежувати, чому модель дійшла саме до таких висновків. Це допомагає вчасно

виявити помилки, упередження чи небажані наслідки, запобігти зловживанням.

Крім цього, розвиток технологій, які стоять за створенням таких моделей, невпинно триває. Удосконалюються архітектури штучних нейронних мереж, реалізуються нові методи регуляризації, створюються більш ефективні алгоритми навчання, що дозволяють збільшувати продуктивність системи і знижувати потребу в колосальних обчислювальних ресурсах. З'являються техніки дистиляції знань, що дозволяють передавати здобуті навички від великої та ресурсоємної моделі до меншого та більш оперативного варіанту. Таким чином, бізнес, наука та урядові організації отримують змогу гнучкіше керувати своїми інтелектуальними інструментами, оперативніше адаптувати їх до нових реалій.

Паралельно з цим розширюється доступність ШІ-технологій. Раніше розробка власних моделей вимагала солідного інвестиційного капіталу, команди досвідчених фахівців та дорогої обчислювальної інфраструктури. Сьогодні ж великі ІТ-компанії та open-source спільноти пропонують готові, уже навчені моделі та платформи, які можна інтегрувати у наявні бізнес-процеси чи програмні продукти. Це робить штучний інтелект ближчим до малого та середнього бізнесу, стартапів, освітніх закладів і громадських організацій. Кожен, хто має бачення ідеї, може скористатися результатами багаторічної праці тисяч дослідників, перетворюючи власні проекти на інноваційні рішення, які здатні змінити цілу галузь.

Зрештою, актуальність і корисність уже навчених моделей ШІ полягає в тому, що вони є своєрідним інтелектуальним двигуном, здатним перетворювати сирий потік інформації та складних завдань на чіткі, зрозумілі та готові до використання результати. Ці моделі стають фундаментом сучасних інформаційних систем, що опікуються як аналітикою даних, так і управлінням ресурсами, а також створенням нового контенту. Вони допомагають підвищити точність прогнозів, знайти оптимальний шлях досягнення поставлених цілей, вирішити складні інженерні, управлінські чи маркетингові питання. Врешті-решт, саме вони дають змогу нам упевнено дивитися в майбутнє, де штучний

інтелект ставатиме дедалі важливішим партнером у розвитку технологій та суспільства.

При впровадженні генеративного штучного інтелекту у бізнес-процеси підприємства відкривається ціла низка можливостей для оптимізації витрат, що безпосередньо впливають на економічну ефективність компанії. Одним із головних аспектів є потенційне зниження операційних витрат, до яких належать витрати на оплату праці, придбання матеріалів та споживання енергії.

Передусім, варто звернути увагу на оплату праці. В умовах традиційного підприємства більшість рутинних та механічних завдань виконується працівниками, які витрачають чимало часу й ресурсів на обробку інформації, введення даних, формування звітності або підготовку документації. Застосування ГШІ дозволяє автоматизувати більшу частину таких завдань. Наприклад, системи на основі штучного інтелекту можуть самостійно збирати, аналізувати та структурувати великі обсяги інформації, готувати звіти або формувати персоналізовані рекомендації, що раніше вимагало залучення людини. Завдяки цьому працівники можуть спрямувати свої зусилля на більш креативні, аналітичні чи стратегічні аспекти роботи. У результаті скорочення частки ручної праці дає можливість зменшити кількість персоналу, необхідного для виконання рутинних операцій, або перевести частину команди на більш продуктивні завдання, тим самим оптимізуючи витрати на заробітну плату.

Слід підкреслити, що досягнення економії завдяки впровадженню ГШІ не є миттєвим процесом. Спершу необхідно вкласти час і ресурси у налаштування систем, навчання моделей, адаптацію бізнес-процесів та навчання персоналу новим навичкам. Проте з часом ці інвестиції окупаються, оскільки оптимізація витрат стає сталим результатом. До того ж, автоматизуючи рутинні завдання та мінімізуючи ймовірність людських помилок, підприємство може знизити ризик неефективного використання ресурсів та фінансові втрати, пов’язані з браком чи дефектами продукції.

Розглядаючи економічні переваги впровадження генеративного штучного інтелекту, не можна забувати про стратегічні вигоди. Зниження витрат дозволяє

підприємству вивільнити додаткові ресурси, які можна спрямувати на інноваційні проекти, дослідження та розвиток, маркетинг, формування нового асортименту продукції, вихід на нові ринки чи вдосконалення обслуговування клієнтів. Усе це посилює конкурентну позицію компанії та сприяє її довгостроковому зростанню.

Отже, впровадження генеративного штучного інтелекту у підприємство відкриває широкі можливості для оптимізації витрат: зниження затрат на оплату праці за рахунок автоматизації рутинних операцій, економія матеріальних ресурсів завдяки точному прогнозуванню потреб та ефективнішому управлінню ланцюгами постачання, а також зниження енергоспоживання через розумне керування виробничими та інженерними процесами. У сукупності ці фактори створюють потужний інструмент для підвищення економічної стійкості підприємства, його продуктивності та здатності успішно конкурувати на глобальному ринку.

Зрозуміння цих промислових потреб та вимог дозволяє розробити ефективну систему, що відповідає конкретним вимогам підприємств. У контексті розробки системи для генерації анімованих зображень, важливо враховувати ці потреби та вимоги, щоб забезпечити оптимальне використання генеративниї системи та досягнення поставлених цілей.

У магістерській дипломній роботі розглядається розробка комплексної системи, яка поєднує у собі сучасні технології та генеративний штучний інтелект. Завдяки динамічному розвитку програмних фреймворків, бібліотек та протоколів безпеки, інтегрувати генеративний штучний інтелект в існуючі бізнес-процеси або інформаційні системи стає простіше та гнучкіше. Такий підхід дозволяє не лише ефективно поєднувати інноваційні методи обробки даних, але й гармонійно інтегрувати нові технології у вже створені інфраструктури.

Сучасний IT-ринок пропонує велику кількість інструментів для розробки серверних та веб-додатків. Ці інструменти постійно еволюціонують, розширюють свій функціонал та пропонують зручні методи інтеграції між

різними компонентами системи. Зокрема, у даній роботі будуть розглянуті такі ключові технології:

- Angular Framework [1]. Цей фреймворк надає можливість створювати динамічні веб-додатки з підтримкою server-side rendering (SSR). Завдяки SSR можна оптимізувати швидкість завантаження сторінок, покращити індексацію пошуковими системами та забезпечити кращий користувацький досвід. Angular, як фреймворк із чіткою архітектурою, типовою структурою проектів та великою спільнотою розробників, значно спрощує процес розробки, тестування та розгортання веб-інтерфейсів.

- Python [2]. Ця мова програмування відома своєю простотою, читабельністю та величезною кількістю бібліотек. У контексті цієї роботи Python буде використаний для реалізації сервісної логіки, зокрема для генерації контенту та інтеграції з API. Широкий вибір бібліотек для роботи з даними, мережею, безпекою, а також інструментів машинного навчання та ШІ робить Python ідеальним вибором для створення комплексних систем. Підтримка асинхронних операцій, простота вбудовування сторонніх модулів та швидкий цикл розробки дозволяють оперативно адаптувати систему до вимог проєкту.

- MySQL [3]. Це реляційна система управління базами даних, що вирізняється стабільністю, високою продуктивністю та широкою підтримкою у спільноті розробників. MySQL ідеально підходить для зберігання структурованих даних та забезпечення швидкого доступу до них. Завдяки розвинутій інфраструктурі реплікації та кластеризації MySQL може обробляти значні навантаження та забезпечувати високу доступність даних. У межах даного проєкту вона слугуватиме сховищем робочих даних системи та буде тісно пов’язана як із серверною частиною, так і з ГШІ-моделлю, що використовуватиметься.

Використання фреймворків, таких як Angular, та мов програмування із багатою екосистемою бібліотек, як Python, дозволяє спростити й прискорити процес розробки. Фреймворки беруть на себе реалізацію типових функціональних блоків: роботу з базою даних, автентифікацію, управління

сесіями користувачів, обробку помилок, маршрутизацію запитів. Це зменшує обсяг «ручного» коду, який повинен писати розробник, підвищує надійність додатку, а також забезпечує перевірені часом архітектурні рішення. Таким чином, використання фреймворків робить розробку структурованішою, передбачуванішою та менш схильною до появи критичних помилок.

Не менш важливим елементом у сучасній розробці є бібліотеки. Вони являють собою колекції кодових модулів, які призначені для розв'язання певних завдань, і можуть бути оперативно інтегровані у проєкт. Завдяки швидкому темпу розвитку індустрії та активній спільноті розробників, бібліотеки дозволяють зекономити час на розв'язанні нетривіальних задач, пропонуючи готові рішення. У даній роботі бібліотеки Python можуть бути використані для генерації даних, що надходять до ГШІ.

Впровадження генеративного штучного інтелекту у систему вимагає чіткого визначення вимог до взаємодії між штучним інтелектом і сервером та веб-додатком. ГШІ може виступати у ролі «розумного помічника», який отримує від користувача певні команди (промпти) та, аналізуючи їх, приймає рішення про виконання тих чи інших операцій. Наприклад, користувач може через інтерфейс веб-додатку запитати згенерувати картинку «Машини», а ШІ-модель, отримавши запит, передасть необхідні дані на сервер для пошуку у базі даних, а потім поверне користувачеві сформований результат. Таким чином, забезпечується тісний зв'язок між ШІ-моделлю, сервером та веб-додатком. Дані, сформовані під час роботи системи, будуть зберігатися у базі даних, у різних тематичних колекціях чи таблицях, залежно від типу інформації та логіки бізнес-процесів.

Для підвищення безпеки та захисту від несанкціонованого доступу система буде використовувати технологію OAuth2.0 [4]. Цей протокол авторизації дозволяє контролювати доступ користувачів та сервісів до захищених ресурсів, не надаючи безпосередньо логіни та паролі. Завдяки цьому знижується ризик витоку облікових даних та забезпечується безпечний обмін інформацією між різними компонентами системи. Безпека та надійність – критично важливі

аспекти для будь-яких корпоративних рішень, особливо якщо вони працюють із конфіденційною інформацією або персональними даними користувачів.

Втім, при інтеграції ГШІ існують і певні недоліки. Зокрема, користувачі не завжди мають можливість безпосередньо редагувати результат, який надає модель штучного інтелекту. Щоб отримувати дійсно корисну інформацію від ШІ-моделі, необхідно вміти складати чіткі та релевантні промпти. Це вимагає базових знань у сфері так званого «промпт-інжинірингу» – мистецтва правильного формулювання запитів до ШІ, аби він давав максимально корисні та точні відповіді. Без цього користувач може швидко розчаруватися у функціоналі системи. Тому впроваджуючи ГШІ, необхідно не лише технічно налаштувати модель, але й забезпечити інструкції, поради чи підказки для користувачів, як правильно ставити запитання.

Аби мінімізувати недоліки та обмеження, розробникам слід ретельно готувати модель ШІ. Наприклад, можна виконувати попереднє навчання або донавчання моделі на корпоративних даних компанії, вдосконалювати алгоритми обробки природної мови, формувати спеціалізовані набори правил та контекстів, що відображатимуть галузеву специфіку чи внутрішні стандарти підприємства. Таким чином, результат стане більш чітким, передбачуваним та корисним для кінцевих користувачів. Правильна підготовка моделі – це інвестиція в якість та надійність системи, яка в перспективі дозволить суттєво розширити сфери застосування та підвищити рівень задоволеності користувачів.

Отже, у даній магістерській роботі буде розглянуто підхід до створення комплексної системи з використанням генеративного штучного інтелекту, фреймворків та бібліотек. Аналізуватимуться можливості, які надають сучасні інструменти розробки, а також вимоги до взаємодії між ШІ та серверною частиною. Буде також розглянуто методи забезпечення безпеки, шифрування та авторизації. Такий комплексний підхід дозволить закласти надійний фундамент для побудови ефективних автоматизованих рішень, що відповідають актуальним вимогам ринку, галузевій специфіці та очікуванням користувачів.

1.2 Визначення технічних, безпекових та юридичних вимог до інтеграції генеративного штучного інтелекту

Із розвитком технологій штучного інтелекту та широким впровадженням генеративних моделей постає все більше питань, пов’язаних із формуванням чітких та виважених вимог до їх використання в реальних виробничих і бізнес-процесах. Визначення технічних, безпекових і юридичних аспектів стає ключовою умовою для успішної інтеграції генеративного штучного інтелекту у систему, адже саме ці аспекти задають певні «правила гри», відповідно до яких працюватиме вся інфраструктура. Зрештою, чим ретельніше буде опрацьовано ці питання, тим стабільнішою, безпечнішою та прозорішою буде експлуатація інноваційних ШІ-рішень.

З технічної точки зору, інтеграція генеративного ШІ вимагає ґрунтовної оцінки апаратного та програмного середовища, у якому функціонуватиме модель. По-перше, необхідно чітко визначити потреби в обчислювальних ресурсах. Генеративні моделі здатні виконувати складні обчислення над великими наборами даних, а для цього часто потрібні потужні сервери з графічними процесорами (GPU), специфічне апаратне прискорення (наприклад, за допомогою тензорних процесорів – TPU) або хмарні рішення, що динамічно масштабуються. Відсутність необхідної інфраструктури може призвести до того, що модель працюватиме занадто повільно, надаватиме неповні або застарілі відповіді, або й узагалі виявиться нездатною обробити потрібні обсяги інформації. Крім того, потрібно забезпечити безперебійну роботу мережевих з’єднань, оперативне масштабування ресурсів та резервне копіювання даних, аби навіть у пікові періоди навантаження робота системи залишалася стабільною.

Не менш вагомим фактором є вимоги до інтеграції ГШІ у вже наявне програмне оточення. Наприклад, якщо підприємство використовує мікросервісну архітектуру, то нова модель має органічно вписатися у цей підхід. Знадобиться розробка або адаптація REST чи GraphQL API, впровадження механізмів кешування відповідей моделі, оптимізація передачі даних між

сервісами, а також налаштування інструментів моніторингу та логування. Такий технічний фундамент має забезпечити безпроблемну взаємодію між ГШІ та іншими компонентами системи: базами даних, інтерфейсами користувачів, засобами автентифікації тощо. Лише за умов дотримання чіткого набору технічних вимог можна гарантувати, що ГШІ виконуватиме свою роль належним чином, а системні архітектори та розробники зможуть ефективно управляти життєвим циклом моделі – від розгортання та навчання до оновлення, масштабування й виведення з експлуатації.

Безпека є одним із центральних питань, коли йдеться про роботу з сучасними ШІ-технологіями. ГШІ зазвичай взаємодіє з цінними масивами даних, які можуть містити комерційну таємницю, персональну інформацію користувачів, результати досліджень чи інші конфіденційні відомості. Відповідно, впровадження ШІ-моделі потребує продуманих рішень у сфері інформаційної безпеки. Застосування протоколів шифрування (наприклад, TLS/SSL для передачі даних по мережі, шифрування баз даних або локальних сховищ), використання VPN для безпечного доступу до внутрішніх ресурсів, впровадження багатофакторної автентифікації та регулярне оновлення системного програмного забезпечення – усе це стає стандартом безпечної інфраструктури. Не менш важливою є наявність інструментів для виявлення та запобігання вторгненням, систем сигналізації про аномальну активність, а також регулярні аудити безпеки.

Окрім технічного захисту, потрібно врахувати можливість появи «соціальних атак» – наприклад, спроб обманути модель через підбір відповідних промптів або використання згенерованого контенту з метою введення в оману інших користувачів. З огляду на це, до функціоналу ГШІ варто інтегрувати механізми фільтрації результатів, виявлення токсичного чи неприпустимого контенту, обмеження доступу для користувачів із сумнівними намірами. Контроль версій та розмежування прав доступу для розробників і адміністраторів системи допоможуть уникнути внутрішніх загроз, а чітка політика ротації ключів та сертифікатів забезпечить стабільну безпеку у

довгостроковій перспективі.

Юридичні та етичні вимоги додають ще один рівень складності. Сьогодні правові норми, що регламентують використання штучного інтелекту, все ще формуються, але вже зараз існують закони про захист персональних даних (наприклад, GDPR у ЄС або CCPA у США), які встановлюють вимоги до збору, зберігання та обробки інформації. Інтегруючи ГШІ у бізнес-процеси, організація зобов'язана дотримуватися цих норм: не передавати конфіденційні дані без відповідних дозволів, інформувати користувачів про спосіб, у який використовується їхня інформація, надавати можливість видаляти чи коригувати персональні дані. Недотримання юридичних вимог може призвести до штрафів, судових позовів, а головне – втрати довіри клієнтів та ділових партнерів.

Окрему увагу варто приділити авторським правам та інтелектуальній власності. Генеративні моделі здатні створювати оригінальний контент, такі як тексти, зображення, музику, але питання про право власності контенту, наявність елементів, що захищені авторським правом третіх сторін, усе частіше постають перед розробниками та компаніями. Необхідно забезпечити чіткі умови використання ГШІ, прописати правовий статус згенерованих матеріалів і механізми їх ідентифікації. Юридичні експерти та фахівці з інтелектуальної власності мають бути залучені ще на етапі планування, щоб зменшити ризики конфліктів та непорозумінь у майбутньому.

Етична складова також набуває дедалі більшого значення. ГШІ може бути використаний як інструмент впливу на суспільну думку, поширення фейкових новин, пропаганди або дезінформації. У деяких випадках моделі можуть мимовільно відтворювати дискримінаційні патерни, закладені в даних навчання, що призведе до упередженості у прийнятті рішень. Щоб уникнути такого сценарію, потрібно впроваджувати процедури перевірки результатів моделі, розробляти спеціальні фільтри та коригувальні механізми, а також періодично проводити оцінку впливу ШІ на користувачів. Команди з етики ШІ, до складу яких входять юристи, соціологи, психологи, можуть допомогти визначити та мінімізувати негативний соціальний ефект технології.

Зрештою, технічні, безпекові та юридичні вимоги до інтеграції ГШІ варто розглядати як взаємопов’язані аспекти однієї цілісної системи. Технічна інфраструктура впливає на безпеку – наприклад, недостатньо захищене серверне середовище може стати вразливим для кібератак. Врахування цих вимог на стадії планування та проектування системи дозволяє уникнути масштабних проблем у майбутньому, забезпечуючи надійне, законне та етично відповідальне використання ШІ-технологій. Інтеграція ГШІ у бізнес не може обмежуватися винятково технічними рішеннями – це комплексний процес, що вимагає міждисциплінарного підходу та співпраці інженерів, юристів, фахівців з безпеки, етики та топ-менеджменту. Лише за такого підходу організація здатна отримати реальні переваги від впровадження ШІ, зберігаючи репутацію, довіру користувачів та дотримуючись найвищих стандартів у своїй діяльності.

1.3 Постановка завдання

Для успішної розробки веб-додатку, що використовує генеративний штучний інтелект для створення анімованих зображень, необхідно визначити ключові аспекти, які впливають на його функціональність, зручність використання та технологічну реалізацію.

Основною метою проекту є розробка комплексного рішення, яке дозволяє інтегрувати сучасні технології для автоматизованого створення анімаційних зображень, забезпечуючи зручність використання та високу якість кінцевого продукту.

Результатом виконання завдань стане створення веб-додатку, який поєднує в собі передові технології штучного інтелекту та сучасні інструменти веб-розробки, відповідаючи високим стандартам якості та вимогам ринку, які дозволять користувачам:

- створювати унікальні анімовані зображення, з використанням інтегрованого ГШІ на основі вхідних даних;

- зберігати та управляти результатами, за допомогою імплементації PostgreSQL [5] у серверну частину проекту, користувач матиме змогу отримувати зображення та завантажувати їх на свій пристрій;

- завдяки зручному інтерфейсу на основі Angular Framework, отримувати доступ до додатку з різних пристроїв;

- гарантувати стабільну роботу серверної частини, використовуючи Python FastAPI для реалізації серверної частини та інтегрувати алгоритм штучного інтелекту;

- дотримуватись безпекових стандартів, з забезпечуючи надійну авторизацію та автентифікацію для доступу до функціоналу додатку.

Таким чином, завдання кваліфікаційної роботи включає в себе розробку всіх необхідних компонентів, від аналізу потреб користувачів до впровадження сучасних технологій для реалізації функціоналу. Результатом роботи має стати веб-додаток, який відповідає сучасним вимогам ринку, має високу якість та забезпечує зручність використання.

1.4 Висновки до першого розділу

У першому розділі було проведено аналіз використання генеративного штучного інтелекту та загальних проблем, що виникають під час його застосування. Аналіз показав, що ШІ відіграє все більш значущу роль у сучасному світі, зокрема й в Україні. Підприємства постійно шукають шляхи оптимізації робочих процесів, прагнучи знизити витрати та підвищити ефективність. У цьому контексті штучний інтелект стає ключовим інструментом для досягнення поставлених цілей.

Потреби та вимоги промисловості щодо використання ШІ варіюються залежно від конкретної галузі та специфіки діяльності. Проте можна виділити ряд загальних вимог, таких як забезпечення ефективної, швидкої та точної обробки даних, мінімізація ризиків та помилок, покращення комунікації та

співпраці між різними підрозділами, а також забезпечення зручного доступу до необхідної інформації.

В процесі аналізу було ідентифіковано низку загальних проблем, пов'язаних із застосуванням ШІ, особливо в контексті розробки додатків для генерації анімованого контенту. Серед цих проблем варто відзначити обмежені можливості інтерактивної взаємодії з користувачами. Це може проявлятися у складнощах з налаштуванням параметрів генерації, відсутністю зворотного зв'язку або обмеженими можливостями редагування згенерованого контенту. Крім того, існують виклики, пов'язані з обчислювальними ресурсами, необхідними для ефективної роботи ШІ, та питаннями етичного використання технології.

Аналіз використання генеративного штучного інтелекту підкреслює необхідність розробки ефективних та зручних інструментів для створення анімованого контенту. Існуючі проблеми в цій області вказують на потенціал для вдосконалення та створення інноваційних рішень. Розробка веб-додатку, що використовує ГШІ для генерації анімованих зображень, спрямована на вирішення цих проблем та задоволення зростаючих потреб у створенні якісного анімованого контенту.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ З ГЕНЕРАТИВНИМ ШТУЧНИМ ІНТЕЛЕКТОМ, АНАЛІЗ ТА ВИБІР МЕТОДІВ ВИРІШЕННЯ

2.1 Дослідження існуючих інструментів для створення штучного інтелекту з інтегрованою моделлю

На даний момент, існує дуже багато інструментів завдяки яким є можливість створювати додатки з ШІ. Для цього використовують будь-яку мову програмування, яка дозволяє створювати ПЗ. В основному використовують, найпопулярніші мови програмування, такі як: C++, Python, C#, JavaScript, PHP, Java.

Мова програмування C++ [6] дуже популярна, та має широкий спектр використання, також має дуже великий інструментарій і велику кількість можливостей для контролю ресурсів комп’ютера, таких як пам’ять і процесор. ЇЇ використовують для системного програмування, розробки прикладного програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java. C++ має декілька переваг:

- компілюється безпосередньо в машинний код, що дозволяє досягати високої продуктивності та швидкості програм;
- забезпечує прямий доступ до пам’яті та ефективне керування системними ресурсами;
- підтримує об’єктно-орієнтований парадигму програмування, що сприяє впорядкованій організації коду і полегшує його розширюваність та підтримку.

Проте C++ також має низку недоліків:

- висока складність мови та розмаїття її можливостей ускладнюють процес навчання та підвищують ризик створення заплутаного коду ;
- низькорівневий доступ і значна свобода дій можуть призвести до

поширених помилок, наприклад до витоків пам’яті чи некоректної роботи з покажчиками;

- C++ не забезпечує автоматичної перевірки та звільнення невикористовуваної пам’яті, що ускладнює запобігання типових помилок при роботі з ресурсами.

Мова C# (C-Sharp) [7] є мовою програмування, яку розробила компанія Microsoft. Вона використовується для розробки різних додатків, включаючи додатки для Windows, веб-додатки.

Має декілька плюсів та мінусів:

- володіє простим і лаконічним синтаксисом, що сприяє швидкому створенню програм та робить її доступною для новачків у програмуванні;
- підтримує об’єктне-орієнтоване програмування, що дозволяє розробникам ефективно організовувати код;
- функціонує на основі платформи .NET, яка полегшує управління пам’яттю та допомагає уникнути проблем з витоками її.

З мінусів даної мови можна виділити:

- у порівнянні C++, C# може бути трохи повільнішою, оскільки вона працює на вищерівневому віртуальному середовищі;
- залежність від платформи Windows. C# та .NET дуже тісно пов’язані з платформою Windows, що може обмежити кросплатформенну розробку додатків.

Отже C++ та C# можуть підійти для створення генеративної системи, але у дипломному проекті планується використовувати прості та швидкі рішення, тому далі буде розглянуто більш прості інструменти, які дозволять створити ГШІ без нагрузки на пам’ять серверу.

Java [8] є високорівневою мовою програмування, здобула широку популярність завдяки своїй розширеності та кросплатформінгу.

У цієї мови програмування багато особливостей та плюсів:

- однією з ключових переваг Java є її кросплатформенність. Програми, написані на Java, можуть використовуватися на будь-якій платформі без

необхідності модифікації коді, що дозволяє розробляти додатки на одній системі та використовувати їх на інших;

- підтримка ООП. Також вона підтримує класи, об’єкти, успадкування, поліморфізм так інші концепції ООП;

- є можливість автоматичного керування пам’яттю, може автоматично вивільняти пам’ять. Це спрощує роботу та запобігає багатьом типам помилок, таких як витоки пам’яті або некоректне використання її;

- надає вбудовану підтримку багатопотоковості, що дозволяє створювати і керувати багатьма потоками виконання;

- має вбудовану систему безпеки, що дозволяє запускати програми в обмеженому середовищі. Це дозволяє запобігти виконанню зловмисного коду та захищати систему від можливих загроз;

- велика екосистема, має велику кількість бібліотек та фреймворків, які спрощують розробку різних типів додатків;

- широке використання даної мови програмування, дозволяє використовувати її у різних сферах.

Незважаючи на чисельні переваги, Java має декілька недоліків, а саме:

- завдяки вбудованому контролі пам’яті, програми можуть споживати більше ніж інші мови програмування;

- низька швидкодія причиною якої є використання віртуальної середи розробки.

Незважаючи на деякі недоліки, Java продовжує бути однією з найбільш використовуваних мов програмування, особливо в рамках великих корпоративних проектів та веб-розробки. Завдяки своїй широкій функціональності, кросплатформенності та багатій екосистемі, Java є ефективним інструментом для різноманітних сценаріїв розробки, включаючи інтеграцію штучного інтелекту в проекти.

Java володіє розширеними можливостями та багатою екосистемою, що робить її ідеальним вибором для розробки систем на базі генеративного штучного інтелекту. Завдяки своїй гнучкості, Java дозволяє розробникам

адаптувати і використовувати різноманітні бібліотеки, фреймворки та інструменти, що задовольняють конкретні технічні вимоги проекту, забезпечуючи тим самим ефективне досягнення поставлених цілей.

Python є високорівневою інтерпретованою мовою програмування, відзначається простотою синтаксису та високою читабельністю коду, що робить її однією з найпопулярніших мов програмування.

Ця мова програмування має декілька плюсів:

- Python має простий активне та зрозумілий синтаксис, що полегшує розробку програм;

- має велику підтримку розробників, що сприяє наявності великої кількості бібліотек, фреймворків та розширень. Це робить цю мову програмування потужним інструментом для швидкого розробки різних додатків;

- кросплатформінг, Python підтримується на багатьох платформах. Це дозволяє розробляти програми, які можуть працювати на різних платформах без необхідності переписування коду;

- Python має велику кількість стандартних бібліотек, які охоплюють різні сфери, включаючи роботу з мережами, обробку тексту, наукові обчислення, веб-розробку, бази даних та багато іншого. Це дозволяє швидко розробляти програми та використовувати готові рішення.

Python є популярним вибором для створення ШІ систем через свою простоту використання та багатий набір бібліотек. Використовуючи бібліотеки, такі як pyLLM [9], PySQL[10], Requests і Fast API, можна автоматизувати веб-скрапінг, взаємодію з API, роботу з файлами та багато іншого. Python також підтримує різні фреймворки, такі як Robot Framework [11], які дозволяють створювати розширені автоматизовані тестові сценарії.

JavaScript [12] є мовою програмування, що широко використовується для створення динамічних веб-сторінок і застосунків. Вона підтримується усіма сучасними браузерами, дозволяючи виконувати код прямо на боці клієнта. JavaScript є ключовою для клієнтської веб-розробки, забезпечуючи інтерактивність сторінок та адаптивність веб-додатків через доступ до DOM

(Document Object Model) та браузерних API. Окрім веб-розробки, JavaScript також застосовується для створення серверних додатків за допомогою Node.js, мобільних застосунків із React Native та Ionic [13], та навіть настільних програм з Electron. Ця мова підтримує асинхронну модель програмування, що забезпечує ефективну обробку запитів і взаємодію з компонентами без затримок у виконанні коду. Велика кількість доступних бібліотек і фреймворків, таких як React, Angular, та Vue.js, роблять JavaScript незамінним інструментом для сучасних розробників.

У даної мови програмування є декілька плюсів:

- JavaScript є найпопулярнішою мовою для веб-розробки, тому вона має велику кількість ресурсів, документацію та підтримку співтовариства;
- надає зручні методи для зміни та маніпулювання вмістом веб- сторінок через DOM, що дозволяє створювати багатофункціональні та динамічні веб-додатки;
- асинхронна модель програмування в JavaScript дозволяє ефективно обробляти багато завдань без блокування виконання коду, що робить його особливо корисним для веб-додатків, які вимагають взаємодії з сервером та виконання запитів.

У цієї мови програмування є декілька недоліків:

- залежність від браузера, JavaScript виконується в середовищі браузера, тому його функціональність може бути обмеженою або по-різному інтерпретованою в різних браузерах;
- недостатня підтримка безпеки: JavaScript виконується на стороні клієнта, що робить його вразливим до атак, таких як впровадження коду (XSS) або маніпулювання клієнтським кодом;
- в порівнянні з іншими мовами програмування, такими як C++ або Java, JavaScript може бути менш продуктивним у виконанні обчислювально важких завдань.

Серед мов програмування для створення веб-додатків, також існує PHP. PHP [14] є мовою програмування, спеціально розробленою для веб-розробки.

Вона використовується для створення динамічних веб-сторінок та розробки веб-додатків з серверною частиною. PHP спеціально розроблена для веб-розробки, тому вона має велику кількість функцій та фреймворків, що спрощують розробку веб-додатків. PHP має простий синтаксис, що робить його легким для вивчення та розуміння, особливо для розробників, які вже мають досвід з HTML. PHP має широку підтримку для різних баз даних, таких як MySQL, PostgreSQL, Oracle [15] та інші, що робить його популярним вибором для розробки додатків, які потребують збереження даних.

Основні плюси мови PHP:

- PHP має велику та активну спільноту розробників, що забезпечує багато ресурсів, документацію та підтримку;
- PHP є однією з найпопулярніших мов для веб-розробки, що означає, що багато хостинг-провайдерів підтримують PHP та мають легке налаштування сервера;
- PHP є досить швидкою мовою виконання, особливо при використанні оптимізованих фреймворків та кешуванні.

До мінусів цієї мови програмування можна віднести:

- PHP не має однорідної структури, що може призводити до нечіткого коду та складнощів у розумінні деяких конструкцій;
- PHP виконується на сервері, тому для розробки та тестування додатків потрібен локальний сервер або веб-хостинг;
- PHP має репутацію мови з низьким рівнем безпеки через можливість вразливостей, таких як SQL-ін'єкція та зловмисний код.

Отже, для розробки систем генеративного штучного інтелекту з інтегрованою моделлю доступний широкий спектр інструментів, що включає високорівневі мови програмування та спеціалізовані фреймворки та бібліотеки. Ці інструменти дозволяють не тільки забезпечити масштабування і гнучкість, але й спростити реалізацію конкретних задач, таких як обробка мови, розпізнавання образів, та автоматичне генерування контенту. Python особливо підходить для створення таких систем завдяки своєму багатому арсеналу готових рішень, що

включає численні бібліотеки та фреймворки, призначені для специфічних завдань у сфері штучного інтелекту.

2.2 Аналіз та вибір методів рішення

У рамках магістерської дипломної роботи розглядається створення системи для генерації анімованого контенту з використанням генеративного штучного інтелекту. Основні вимоги до системи – це кросплатформеність та швидкість реакції на дії користувача. В якості ключових інструментів для реалізації проекту було обрано Python та TypeScript [16]. Python вибрано через його широкі можливості, велику підтримку спільноти та численні бібліотеки, що забезпечують надійність та високу продуктивність серверної частини, здатної обробляти дані та інтегрувати ШІ з веб-додатком. Важливою перевагою Python є його кросплатформеність, що дозволяє користувачам використовувати систему на різноманітних пристроях, від мобільних до десктопних.

Оскільки JavaScript є однією з найпопулярніших мов програмування з великою підтримкою розробників, існує TypeScript. TypeScript – це надбудова над JavaScript, яка вносить статичну типізацію та розширює можливості мови, забезпечуючи більш строгу структуру коду та підвищуючи його надійність у складних проектах розробки.

Завдяки цьому можна створювати більш точніші додатки, завдяки своїм перевагам:

- TypeScript дозволяє визначати типи змінних, параметрів функцій та інших елементів програми, що поліпшує надійність та допомагає виявляти помилки на ранніх етапах розробки;
- включає в себе функціональні можливості, такі як класи, модулі, інтерфейси та декоратори, що полегшують розробку складних систем та поліпшують перевикористання коду;
- TypeScript є суперсетом JavaScript, тому весь наявний JavaScript-код може

бути безпосередньо використаний у проекті TypeScript. Це означає, що ви можете поступово переносити ваш проект на TypeScript, додавши поступово типи та використовуючи нові функціональні можливості.

Приклад роботи мови програмування TypeScript на рисунку 2.1.

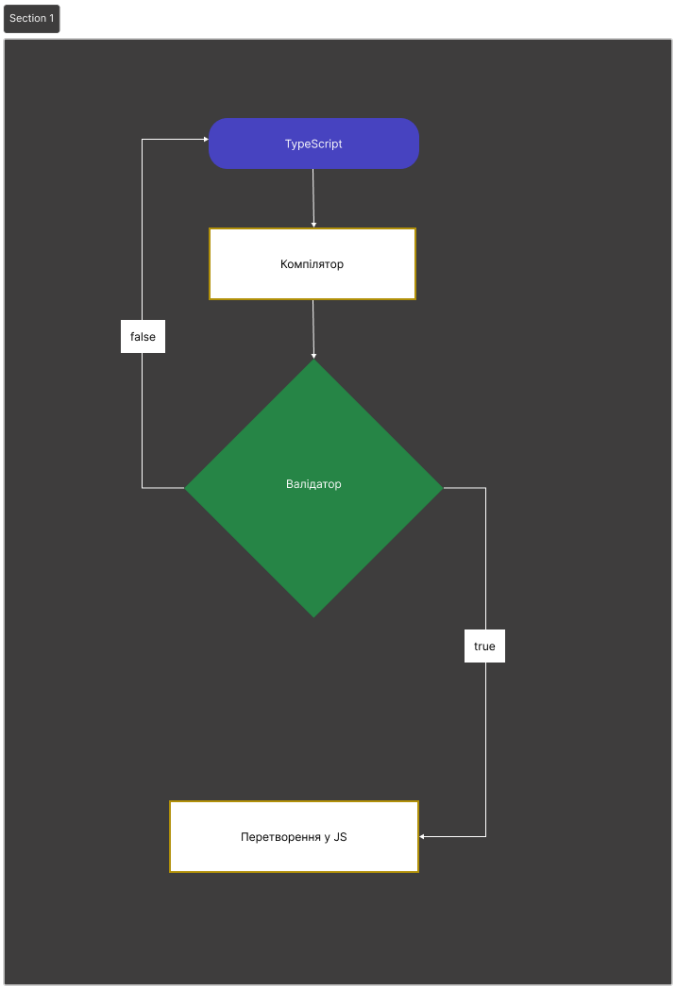

Рисунок 2.1 – Схема роботи мови TypeScript

Тобто, при компіляції додатку, TypeScript компілюється у JavaScript файл, який потім читає браузер.

Завдяки підтримці розробниками мови програмування TypeScript є багато фреймворків і бібліотек, які використовують її. Такі як React, Angular, Vue.js. Ці інструменти служать для створення веб-додатків.

React [17] – це відкрита бібліотека для створеня веб-додатків, дозволяє створювати прості та ефективні SPA, має декілька переваг, а саме:

- React використовує віртуальний DOM (Virtual DOM), що дозволяє ефективно оновлювати тільки необхідні елементи сторінки, що призводить до покращеної продуктивності і швидшої реакції веб-додатку;

- побудований на основі компонентної моделі, що дозволяє розбивати інтерфейс на незалежні компоненти, які можна повторно використовувати. Це спрощує розробку, тестування та підтримку коду.

Angular framework, розроблений компанією Google, представляє собою відкритий фреймворк, написаний на мові програмування TypeScript. Це потужний інструмент для створення односторінкових застосунків (SPA), який забезпечує розробників комплексним набором функцій для ефективної роботи над великими та складними проектами. Angular надає вбудовані рішення для різних аспектів розробки, таких як маршрутизація, управління станами, валідація форм, і тестування, спрощуючи тим самим процес розробки та підтримки веб-додатків.

Основна перевага Angular полягає в його здатності використовувати TypeScript, який розширює JavaScript, надаючи можливості статичної типізації та інші об'єктно-орієнтовані характеристики, такі як класи та інтерфейси. Це не тільки підвищує продуктивність і надійність розроблюваних додатків, але й сприяє кращому управлінню кодом та його перевикористанню, особливо великими командами розробників.

Крім того, Angular має велику кількість готових рішень та велику екосистему, що включає різноманітні бібліотеки та плагіни, забезпечуючи підтримку мобільних платформ, рендеринг на стороні сервера (Server-Side Rendering, SSR [18]) для підвищення швидкодії та кращої SEO-оптимізації, а також розробку прогресивних веб-додатків (Progressive Web Apps, PWA).

Vue.js [19] – це JavaScript фреймворк, який створений для розробки веб-додатків на основі моделей даних, через реактивне зв’язування даних. Має простий та легкий у вивченні синтаксис, що дозволяє швидко почати розробку веб-додатків. Він також добре підходить для поступового впровадження в існуючих проектах. Має оптимізоване внутрішнє ядро, яке забезпечує високу

продуктивність та ефективне оновлення відображення.

Отже, для створення системи генерації контенту, доцільно використовувати потужні інструменти багатими на підтримку розробників, такі як Python, Angular framework. Також для коректної та стабільної роботи автоматизованої системи необхідно використовувати монорепо структуру додатків. Тобто, сервер з веб-додатком заходяться в одному репозиторії і працюють паралельно. Для створення серверних додатків, доцільно використовувати мову програмування Python та її бібліотеку Fast Api, яка розширює можливості цієї мови.

FastAPI [20] представляє собою сучасний веб-фреймворк для Python, що заслужив визнання завдяки своїй швидкості та ефективності. Розроблений на основі стандартів Python 3.6+, цей фреймворк інтегрує можливості асинхронного програмування, що дозволяє йому виконувати операції з високою швидкістю, порівнянною з такими мовами, як NodeJS [21] і Go. Основа FastAPI - це бібліотеки Starlette для мережевих операцій та Pydantic для валідації даних, які разом формують потужну платформу для розробки веб-додатків.

Однією з ключових переваг FastAPI є його здатність до швидкої розробки надійних API завдяки системі автоматичної валідації вхідних даних та серіалізації вихідних даних. Ця функціональність значно знижує кількість помилок у роботі та спрощує процес інтеграції та тестування додатків. Крім того, FastAPI автоматично генерує документацію для API за допомогою Swagger і ReDoc, що робить API доступним для тестування в реальному часі та забезпечує високий рівень документування проекту без додаткових зусиль з боку розробників.

FastAPI також надзвичайно гнучкий у плані сумісності та масштабування. Завдяки підтримці асинхронного програмування та сумісності з ASGI, фреймворк може легко впроваджуватись у великі системи та інтегруватися з іншими асинхронними фреймворками. Це відкриває можливості для створення масштабованих рішень, які здатні обслуговувати велику кількість користувачів та додатків.

Окрім технічних переваг, важливим аспектом FastAPI є його внесок у забезпечення безпеки розроблюваних додатків. Фреймворк включає багато вбудованих функцій безпеки, які допомагають управляти автентифікацією, авторизацією та іншими аспектами безпеки через стандарти OAuth2 та інші механізми захисту. Це робить FastAPI відмінним вибором для проектів, де критично важлива надійність та безпека даних.

Для створення серверної частини та імплементації ШІ доцільно використовувати мову програмування Python. Python має велику кількість бібліотек, які мають великий функціонал для створення додатків. Існує багато різноманітних бібліотек з документаціями, які дозволяють інтегрувати ШІ з їх моделями. Для роботи з ШІ використовується бібліотека PyTorch [22].

PyTorch, розроблений Facebook, є одним з лідерів серед фреймворків машинного навчання завдяки своїй гнучкості та високій продуктивності. Його особливість полягає в динамічному побудові обчислювальних графів, що дозволяє модифікувати архітектуру нейронних мереж в режимі реального часу та спрощує процес експериментування та дебагінгу. Така особливість робить PyTorch особливо зручним для наукових досліджень та розробки прототипів.

Використання PyTorch з підтримкою GPU через CUDA [23] дозволяє значно прискорити процеси обчислення, що є надзвичайно важливим при тренуванні складних моделей глибокого навчання. Висока обчислювальна потужність, яку забезпечує GPU, ефективно вирішує задачі, що потребують інтенсивних обчислень, такі як тренування нейронних мереж. Оптимізовані алгоритми обробки даних в PyTorch подальше підвищують продуктивність, роблячи цей фреймворк ідеальним рішенням для проектів, де важлива висока обчислювальна продуктивність.

Крім технічних переваг, PyTorch славиться своєю великою та активною спільнотою. Розробники та дослідники постійно додають нові бібліотеки та інструменти, що розширюють можливості фреймворку, включно з покращенням роботи з даними, новими алгоритмами навчання та поліпшеннями у візуалізації моделей. Таке багатство ресурсів сприяє інноваціям та забезпечує широкі

можливості для розробки складних систем штучного інтелекту.

PyTorch також має інтуїтивно зрозумілий API, що спрощує роботу з моделями, даними та оптимізаціями. Вбудована підтримка для глибокого навчання включає численні готові модулі та функції, які дозволяють швидко розгортати та масштабувати нейронні мережі. Автоматична документація, заснована на анотаціях коду, допомагає утримувати проекти організованими та зрозумілими для великих команд розробників.

PyTorch створює міцну платформу, яка ефективно використовується як в академічних дослідженнях, так і в комерційному секторі, відповідаючи на зростаючі вимоги до швидкості розробки та адаптивності технологічних рішень. Цей фреймворк особливо цінується за свою гнучкість у моделюванні та здатність до масштабування, що робить його відмінним вибором для проектів, що потребують розробки складних систем штучного інтелекту.

2.3 Дослідження проектування системи з генеративним штучним інтелектом із використанням досліджених технологій

У даному проекті передбачається використання монорепо структури [24] додатку. Монорепо це стратегія управління кодом, при якій весь код проекту, включно з бібліотеками, залежностями, сервісами та іншими компонентами, зберігається у єдиному репозиторії. Цей підхід забезпечує централізоване управління джерелами і може спростити процеси розробки, тестування та впровадження.

У монорепозиторії всі компоненти проекту розташовані разом, що усуває необхідність управління версіями зовнішніх залежностей для різних проектів. Всі частини системи можуть бути оновлені одночасно, що мінімізує конфлікти залежностей і спрощує масштабування. Дозволяє легко використовувати загальні бібліотеки і компоненти без необхідності створення окремих пакетів чи модулів. Це сприяє повторному використанню коду та знижує ймовірність

дублювання зусиль у різних проектах. Єдина кодова база в монорепо може спростити налаштування робочих процесів, таких як збірка, тестування та розгортання. Це може забезпечити більш єдиний підхід до CI/CD [25] (Continuous Integration/Continuous Deployment) і полегшити автоматизацію цих процесів.

Але у цієї структури є ряд недоліків, таких як, у великих системах, де десятки або сотні розробників працюють над різними частинами проекту, монорепо може створювати проблеми з продуктивністю та керуванням. Збільшення обсягу коду може призвести до збільшення часу збірки та тестування. Централізація всієї кодової бази в одному репозиторії може створювати потенційні ризики безпеки, якщо доступ не належно контролюється, особливо в багатокористувацьких або відкритих середовищах.

Отже у контексті дипломного проекту, зосередженого на створенні системи для генерації анімованого контенту, монорепозиторій може надати кілька переваг. Централізація усіх компонентів проекту, включаючи бекенд, фронтенд та інструменти генерації зображень, може спростити розробку та тестування. Єдина кодова база може полегшити інтеграцію різних модулів та компонентів системи, гарантувати консистентність даних і налаштувань по всьому проекту.

Саме важливе в цій системі це база-даних в якій будуть зберігатись облікові записи користувачів, генеровані анімації, які можливо отримати у веб-додатку або через електрону пошту. Для збереження файлів у базі-даних, ШІ кодує їх у бінарний формат, та передає на сервер, який в свою чергу зберігає новий запис у базі. Якщо, необхідно отримати генерований файл, сервер може відправити бінарний формат до веб-додатку після чого він декодує файл, та покаже його користувачу, чи через електронну пошту, тоді сервер декодує файл і відправляє його на пошту.

2.3 Висновки до другого розділу

У даному розділі детально проведено огляд існуючих інструментів, які дозволяють розробити систему для генерації анімованих зображень за допомогою ШІ. Розглядаються різноманітні мови програмування, фреймворки та бібліотеки, кожна з яких має свої сильні та слабкі сторони в контексті даної задачі. Зокрема, аналізуються такі мови як C++, C#, Java, Python та JavaScript, з акцентом на їхню продуктивність, зручність розробки, наявність необхідних бібліотек для машинного навчання (наприклад, TensorFlow, PyTorch, scikit-learn для Python) та можливості інтеграції з веб-технологіями. Окрім мов програмування, розглядаються також веб-фреймворки, такі як Angular та React, для створення інтерактивного інтерфейсу користувача. Аналіз охоплює порівняння цих інструментів за критеріями швидкості виконання, масштабованості, підтримки різних платформ та наявності розвиненої екосистеми. Також розглядаються можливості використання TypeScript як надбудови над JavaScript для покращення структури коду та забезпечення статичної типізації.

Проведено ретельний аналіз інструментів з метою визначення найбільш оптимальних для реалізації автоматизованої системи генерації анімованих зображень. Цей аналіз враховує специфіку проєкту, включаючи вимоги до швидкості обробки даних, необхідність інтеграції з існуючими системами, доступність необхідних бібліотек та фреймворків, а також рівень складності розробки та підтримки. На основі цього аналізу обґрунтовано вибір конкретних інструментів, таких як Python для бекенду (завдяки потужним бібліотекам для машинного навчання, таким як PyTorch та TensorFlow) та Angular для фронтенду (для створення інтерактивного та динамічного інтерфейсу). Також обґрунтовано використання FastAPI як веб-фреймворку для створення швидкого та ефективного API для взаємодії між фронтендом та бекендом. Окрему увагу приділено вибору стратегії управління кодом, а саме монорепозиторію, який дозволяє централізовано зберігати та керувати всіма частинами проєкту (бекенд,

фронтенд, інструменти генерації) та спрощує процеси розробки, тестування та розгортання.

Окрім того, в розділі розглянуто питання зберігання даних, зокрема використання бази даних для облікових записів користувачів та згенерованих анімацій. Описано процес кодування файлів у бінарний формат для зберігання в базі даних та процес їх декодування для відображення користувачеві або відправлення на електронну пошту. Також обґрунтовано вибір монорепозиторію як стратегії управління кодом, з урахуванням його переваг та недоліків для даного проєкту, таких як спрощення процесів розробки та тестування, забезпечення консистентності даних та потенційні проблеми з масштабуванням. Розділ завершується висновками щодо вибору інструментів та їхнього обґрунтування для досягнення поставлених цілей проєкту.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ДОДАТКУ З ВИКОРИСТАННЯМ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ СТВОРЕННЯ АНІМОВАНИХ ЗОБРАЖЕНЬ

3.1 Створення системи з генеративним штучним інтелектом.

Для реалізації ГШІ на мові програмування на Python через бібліотеку PyTorch, необхідно встановити велику кількість залежностей та бібліотек, які забезпечують необхідні функції для розробки, тренування та розгортання моделей ГШІ. Залежності, такі як NumPy [26] та Pillow [27], відіграють важливу роль в обробці та маніпуляції даними, особливо зображеннями, що є критичним для будь-яких завдань, пов'язаних із візуальним контентом. Бібліотека FastAPI, що також включена до списку, сприяє легкому створенню веб-додатків та API, що є необхідним для інтеграції моделей ГШІ в мережеві сервіси. Це дозволяє легко ділитися результатами моделі з кінцевими користувачами через інтернет. Використання Uvicorn [28] як ASGI сервера сприяє асинхронній роботі веб-додатків, що може значно підвищити продуктивність та відповідь системи під час обробки великої кількості запитів або великих об'ємів даних.

Transformers [29] від Hugging Face [30] надає доступ до великої кількості передтренованих моделей, які можуть бути використані для розуміння природної мови, генерації тексту, або навіть створення анімованого контенту. Це важливо для розширення можливостей вашого проекту без необхідності тренування власних моделей з нуля, що може бути ресурсомістким. Крім програмного забезпечення, налаштування обладнання для ефективного тренування нейронних мереж є критичним. Використання GPU, зокрема тих, що підтримують CUDA, може значно прискорити процес тренування, знижуючи час від ітерації до впровадження рішень. Це також впливає на вибір хостингу та розгортання моделі, де вимоги до продуктивності обладнання мають бути узгоджені з технічними характеристиками вибраних технологій.

Після вибору інструментів для створення ГШІ, дуже важливо створити блок-схему, на якій відображається принцип роботи ШІ та його сценарії. Блок-схема наведена на рисунку 3.1.

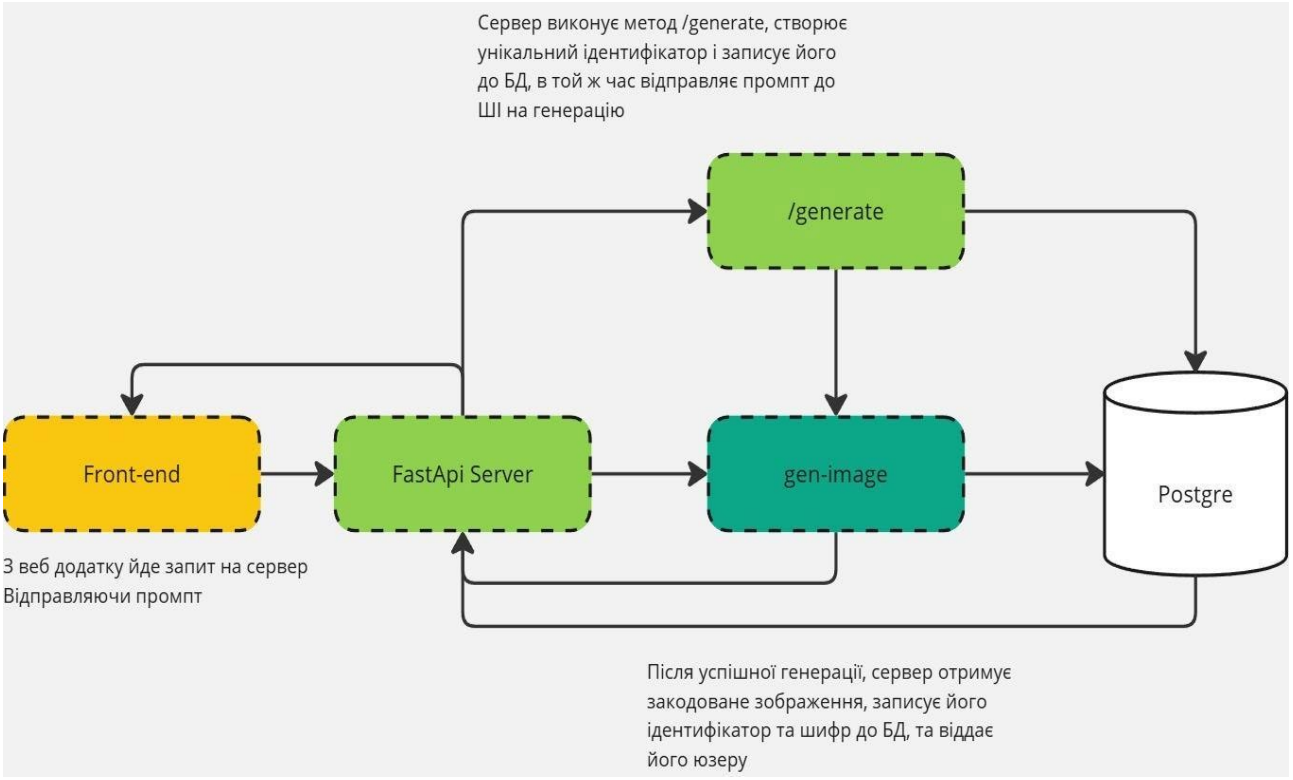

Рисунок 3.1 – Блок-схема принципу роботи системи

Початок роботи системи генеративного штучного інтелекту тісно пов'язаний з процесом авторизації клієнта, оскільки забезпечення безпеки є критичним аспектом при використанні таких сервісів. На першому етапі користувач мусить авторизуватися у системі, щоб підтвердити свою ідентичність і отримати доступ до функціоналу. Під час авторизації система перевіряє, чи є інформація про користувача у базі даних. Якщо користувач зареєстрований, то він отримує ключ доступу, який використовується для подальших запитів до системи. Цей ключ є своєрідним "перепусткою", яка надає можливість користувачеві взаємодіяти з системою, отримувати або модифікувати дані. У разі невідповідності даних, система поверне помилку, повідомляючи про неможливість ідентифікації користувача.

Використання ключа доступу є стандартною практикою у багатьох сучасних веб-сервісах, оскільки воно дозволяє зберігати взаємодію між клієнтом і сервером захищеною і приватною. Ключ дозволяє серверу відстежувати сесії користувачів та надавати дозволи на виконання певних дій відповідно до рівня доступу користувача. Наприклад, користувач може змінювати свої особисті дані, такі як пароль, електронну адресу чи ім'я, а також може запитувати ресурси, які є відкритими лише для авторизованих користувачів.

Процес генерації контенту також вбудований в систему контролю доступу на основі ключів. Коли користувач відправляє запит на створення контенту, система генерує унікальний ідентифікатор для кожної операції, який реєструється в базі даних. Цей ідентифікатор допомагає серверу ідентифікувати та розрізняти завдання від різних користувачів, що забезпечує організацію і безпеку в процесі обробки даних. Користувачі отримують можливість переглядати чи завантажувати результати своїх запитів, виключно їхній власний генерований контент.

Всі дії в системі зберігаються та керуються через базу даних, що гарантує централізоване управління інформацією та високий рівень безпеки. База даних відіграє ключову роль у зберіганні не тільки персональних даних користувачів, але й історії їхніх операцій, збереженні генерованих ресурсів та управлінні доступом до різних частин системи. Така архітектура дозволяє впроваджувати складні функціональні можливості при забезпеченні високого рівня надійності та зручності користувачів. На рис. 3.2 зображено колекцію з бази користувача.

Коли процес підготовки запускається у системі генеративного штучного інтелекту, відбувається перевірка на локальну наявність спеціалізованих моделей для анімації, зокрема на базі технологій Diffusers та Stable Diffusion [31]. Якщо система не виявляє необхідні моделі у локальному сховищі, вона автоматично ініціює процес їх завантаження з віддалених репозиторіїв. Цей механізм забезпечує велику гнучкість у виборі між використанням вже наявних локальних ресурсів та доступом до загальнодоступних моделей з глобальних бібліотек.

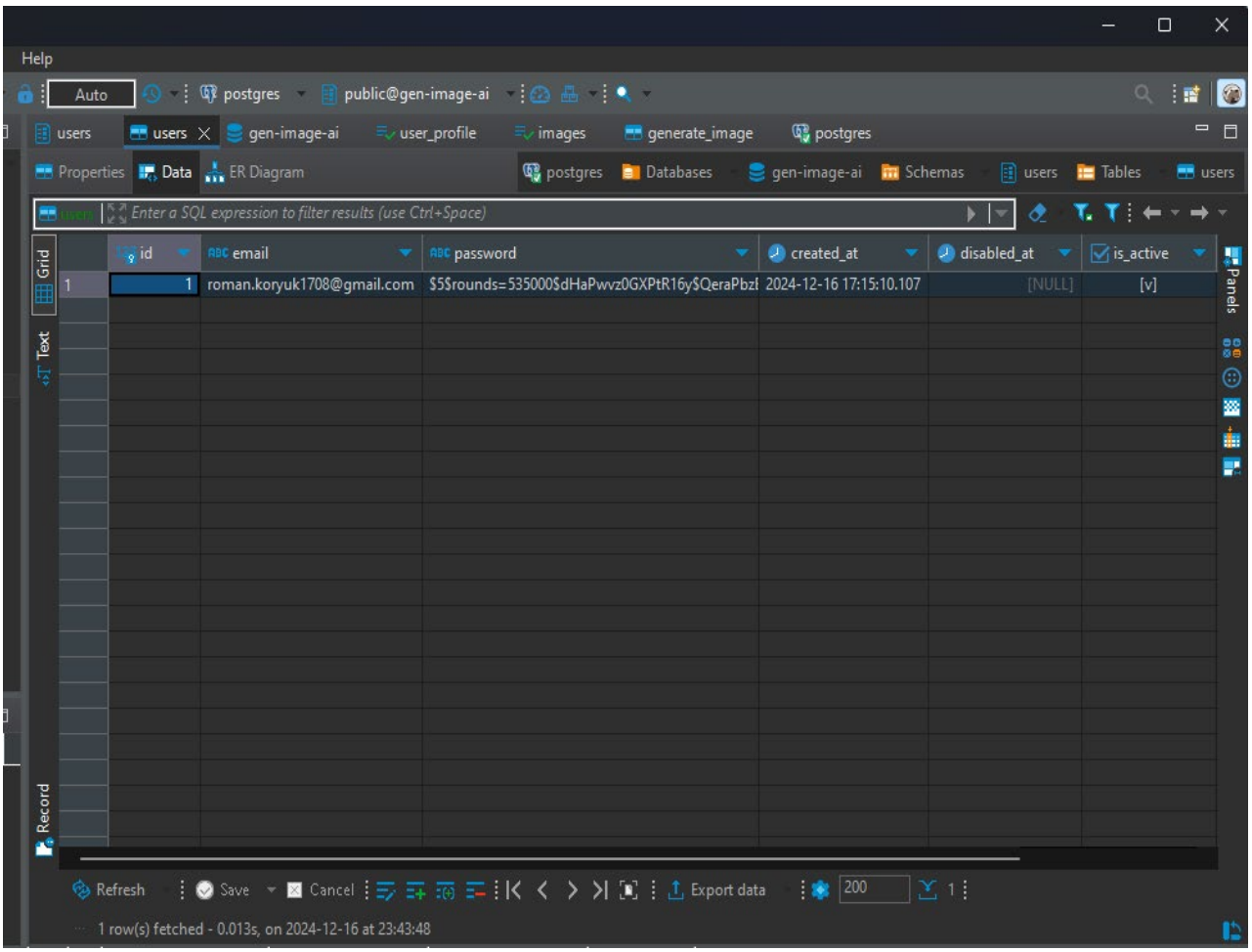

Рисунок 3.2 – Колекція бази даних

Після того, як моделі завантажені та готові до використання, система переходить до налаштування анімаційного пайплайну. В цьому контексті кожен кадр анімації створюється на основі текстових підказок та відповідних параметрів, заданих користувачем. Цей процес дозволяє не тільки детально налаштовувати кожну анімацію, але й гарантує, що кожен кадр буде відповідати вимогам та очікуванням користувача. Цей механізм забезпечує велику гнучкість у виборі між використанням вже наявних локальних ресурсів та доступом до загальнодоступних моделей з глобальних бібліотек. При детальному огляду моделей для ШІ, було обрано дві моделі, які дуже добре працюють один з одним, а саме «toonyou» та «animatediff-motion-adapter», завдяки цим моделям є можливість генерувати картинки, та анімувати їх. Підготовка ШІ для генерації зображено на рисунку 3.3.

Рисунок 3.3 – Підготовка ШІ для генерації анімованного контенту

При готовності анімованного контенту, ГШІ відправляє результат на сервер, де йде збереження анімації в БД, готова генерованна анімація зображена на рисунку 3.4. Використання систем на базі генеративного штучного інтелекту, які залежать від складних обчислень для створення анімованих зображень, має кілька суттєвих недоліків. Одним із головних є вимога до високої обчислювальної потужності, що необхідна для ефективного функціонування таких систем. Це може вплинути на швидкість обробки та затримку у відповідях системи, особливо коли мова йде про генерацію складних анімацій або великі об'єми даних. Внаслідок цього, процес генерації контенту може займати значний час, що може стати бар'єром для надання послуг у режимі реального часу.

Вибір хостингового сервісу для таких систем вимагає особливої уваги до технічних характеристик серверів.

Необхідно звертати увагу на сервери, які забезпечують високу продуктивність та мають достатню кількість обчислювальних ресурсів, таких як CPU і GPU, які можуть ефективно справлятися з високим навантаженням. Це дозволить зменшити час обробки даних та підвищити загальну продуктивність системи, забезпечуючи користувачам швидкий доступ до генерованого контенту без значних затримок.

Рисунок 3.4 – Зразок згенерованого контенту

Після успішної генерації, анімація зберігається в базі даних. Зразок збереження анімації наведено на рисунку 3.5.

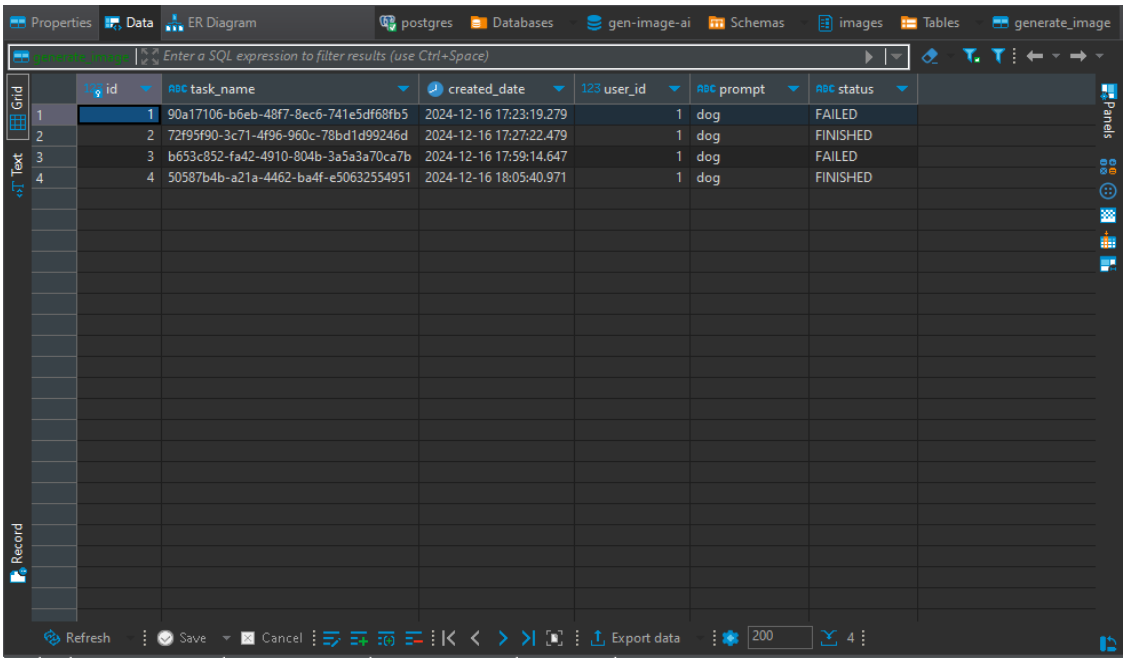

Рисунок 3.5 – Зразок збереження анімації в БД

Всі дані з якими працює ГШІ, одразу ж відправляються на сервер, який оновлює вигляд веб-додатку та актуалізує інформацію.

3.2 Реалізація серверу використовуючи інструмент Python Fast API

Для реалізації серверу на мові програмування Python з використанням Python Fast API бібліотеки, необхідно визначитись із структурою серверу, які методи до нього будуть входити та його функціонал за побудованою блок-схемою, яка наведена на рисунку 3.6. Для створення структури даних необхідно створити моделі, які представляють об’єкти, з якими працює сервер, які містять властивості та методи роботи з даними. Для того щоб забезпечити зв’язок між сервером та іншими додатками, необхідно реалізувати контролери які відповідають за обробку HTTP-запитів та взаємодію з моделями даних. Вони приймають запити від інших додатків, обробляють їх та повертають відповіді. Також необхідно підключити базу даних, яка буде зберігати дані користувачів та згенерованого контенту.

Блок схема яка побудована на рисунку 3.3 показує загальну роботу серверу, і більш підходить до взаємодії з веб-додатком, так як у самому веб-додатку буде реалізоване авторизація користувачів, отримання, редагування та створення даних. Але, при використані з ШІ, сервер дозволить використовувати свої контролери, так як сервер має токен доступу системи, та дозволяє отримувати та зберігати дані без авторизації.

Тобто, для використання даних потрібна авторизація користувача, без якої він не зможе використовувати автоматизовану систему, це вирішує питання пов’язане з безпекою всієї системи. Токен доступу, який генерує сервер є унікальним, тому при кожній авторизації він різний, це знизить ризики несанкціонованого доступу до автоматизованої системи. Така система авторизації через токен доступу зветься OAuth2.0.

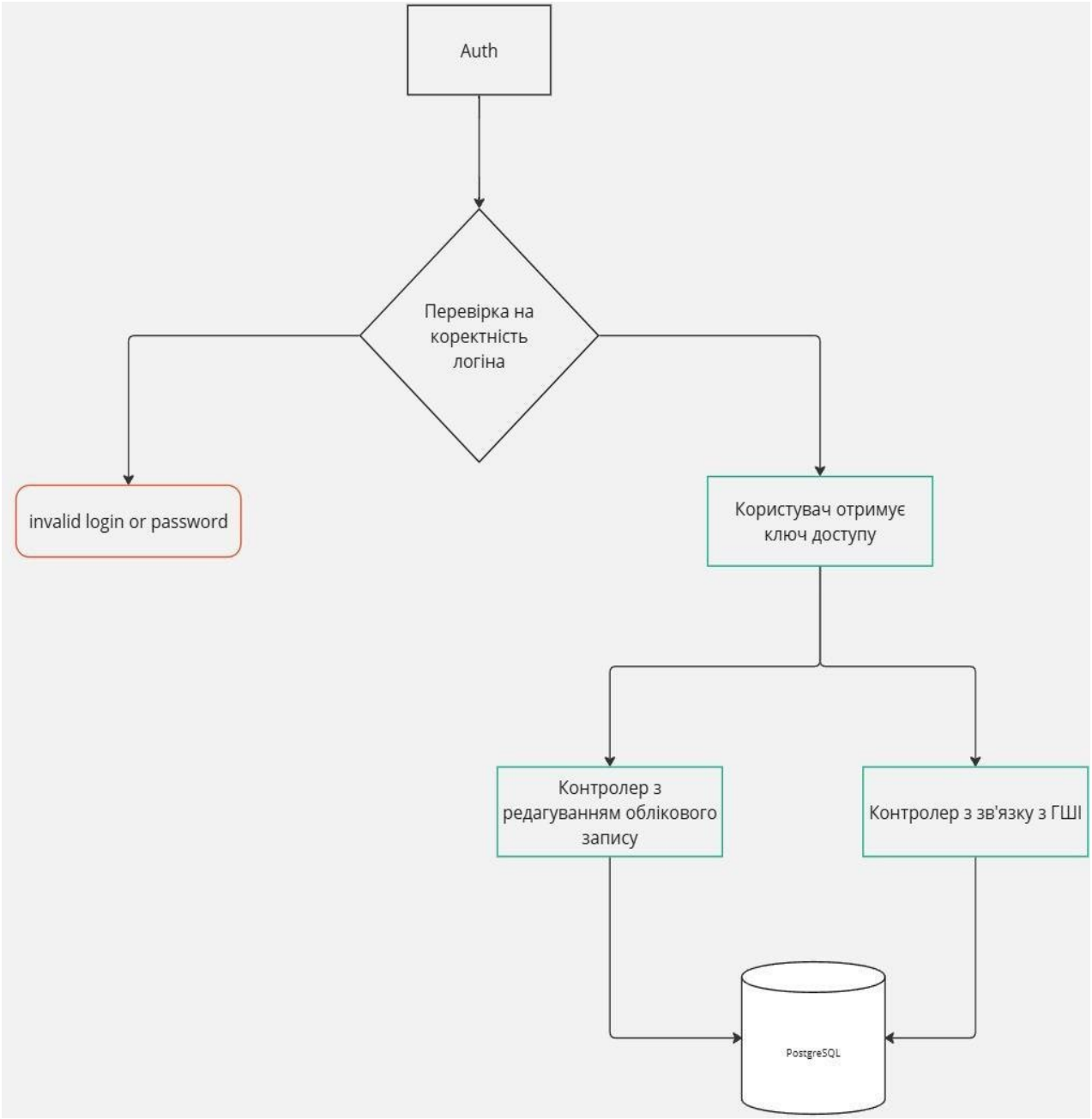

Рисунок 3.6 – Блок-схема функціоналу сервера

OAuth2.0 є протоколом авторизації, який використовується для забезпечення безпеки. Він став стандартом для авторизації у багатьох веб-додатках та API. Забезпечує безпечну авторизацію та передачу даних між клієнтом, авторизаційним сервером та сервером ресурсів.

Після того як користувач пройшов авторизацію та отримав токен доступу, він може отримувати, створювати дані, та зберігати їх. Усі ці дані сервер зберігає та отримує з бази даних.

3.3 Розробка дизайну та реалізація додатку

Розробка веб-додатку з використанням Angular починається з планування його дизайну та архітектури. Архітектурна структура проекту детально викладена у блок-схемі на рисунку 3.7. Завдяки своїм властивостям, Angular дозволяє розробляти високопродуктивні веб-додатки з високою швидкістю відгуку. Особливо важливою є здатність Angular створювати односторінкові додатки (SPA), що робить цей фреймворк ідеальним для розробки багатофункціональних веб-додатків, які потребують швидкої взаємодії з користувачем.

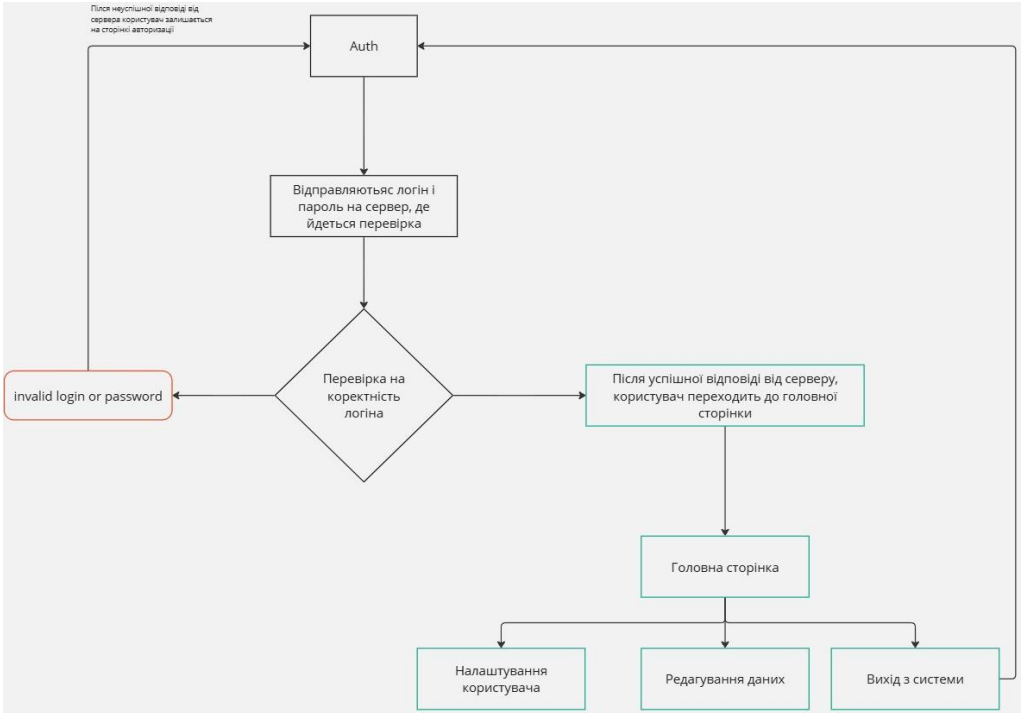

Рисунок 3.7 – Блок-схема роботи веб-додатку

На блок-схемі відображено процес роботи веб-додатку, розпочинаючи з моменту авторизації користувача. Спочатку користувач намагається увійти на сайт, вводячи свої логін і пароль. Якщо введені дані є неправильними, сервер відправляє повідомлення про помилку, і користувач залишається на сторінці авторизації для повторної спроби входу. Цей механізм забезпечує додатковий рівень безпеки, запобігаючи неавторизованому доступу до функцій додатку.

У випадку успішної авторизації, коли логін та пароль виявляються вірними, користувач перенаправляється на головну сторінку веб-додатку. На цій сторінці відкриваються різноманітні опції: від генерації анімованих зображень до завантаження цих зображень. Також користувач має доступ до налаштувань свого облікового запису, де може змінювати свої персональні дані або іншу важливу інформацію.

В процесі використання веб-додатку, користувач може переходити між різними сторінками, генерувати анімовані зображення та керувати своїм профілем. Після завершення роботи з додатком, користувач може вийти з системи через опцію «Вийти». Ця дія призводить до деактивації його токену доступу, що забезпечує додатковий рівень безпеки, оскільки після виходу з додатку користувач повністю втрачає доступ до функціоналу до наступної авторизації. Такий підхід допомагає забезпечити конфіденційність та захист даних користувачів, надаючи змогу контролювати доступ до важливої інформації та ресурсів веб-додатку.

Після створення блок-схеми та аналізу структури веб-додатку, необхідно створити дизайн додатка, саме завдяки дизайну, користувачу буде більш зрозуміло, як працює додаток. Дизайн додатків розробляється завдяки інструменту Figma, це новітній та потужний інструмент, який дозволяє створювати макети як веб-додатків, так і мобільних. Дизайн сторінки авторизації зображено на рисунку 3.8, дизайн головної сторінки на рисунку 3.9.

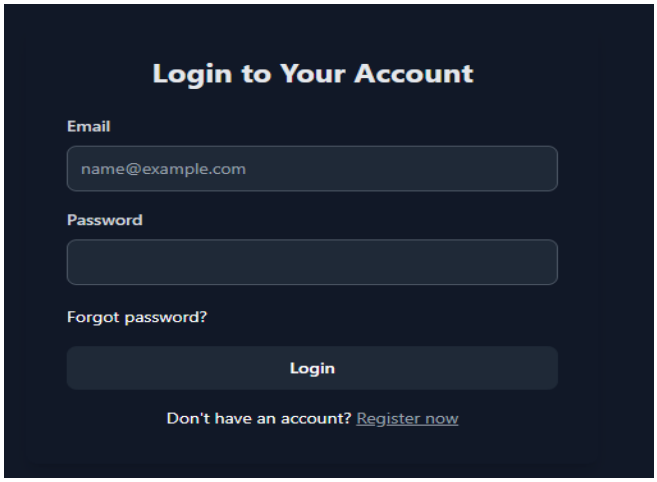

Рисунок 3.8 – Макет сторінки авторизації

На рисунку 3.8 зображено сторінку авторизації користувача, має картку з двома полями «Email» та «Password», і кнопку при натисканні якої йде запит на сервер, після якого йде перевірка на коректність авторизаційних даних. Але якщо користувач забув пароль, то він може скинути його через «Forgot password». На рисунку 3.9 зображено головну сторінку де є поле для вводу промпту, область де буде показуватись процес генерації контенту та кнопку «Generate».

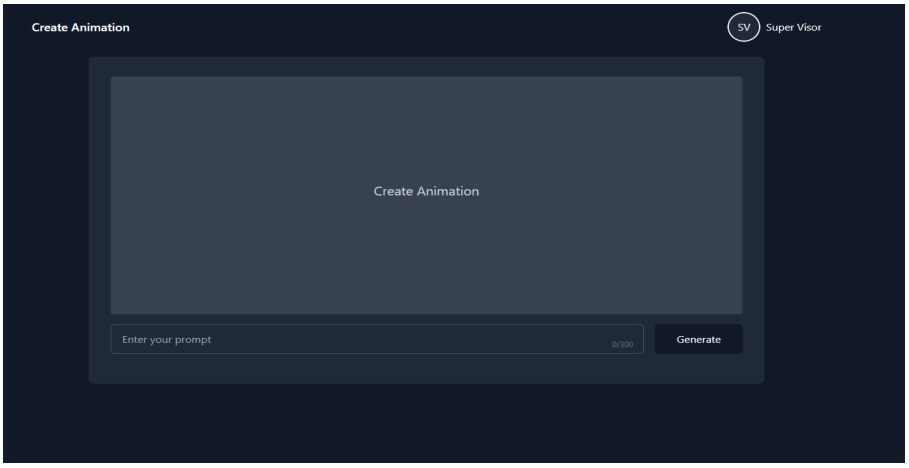

Рисунок 3.9 – Макет головної сторінки

При натисканні на кнопку «Generate» користувач побачить прогрес генерації контенту, після завершення якого, з’явиться анімація яку користувач хотів згенерувати. На рисунку 3.10 зображено процес виконання генерації, результат генерації зображено на рисунку 3.11.

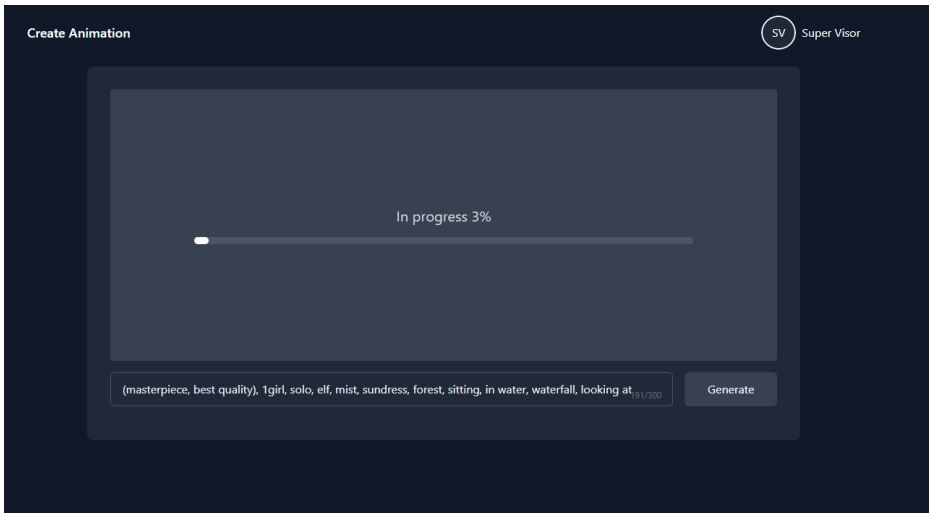

Рисунок 3.10 – Процес виконання генерації

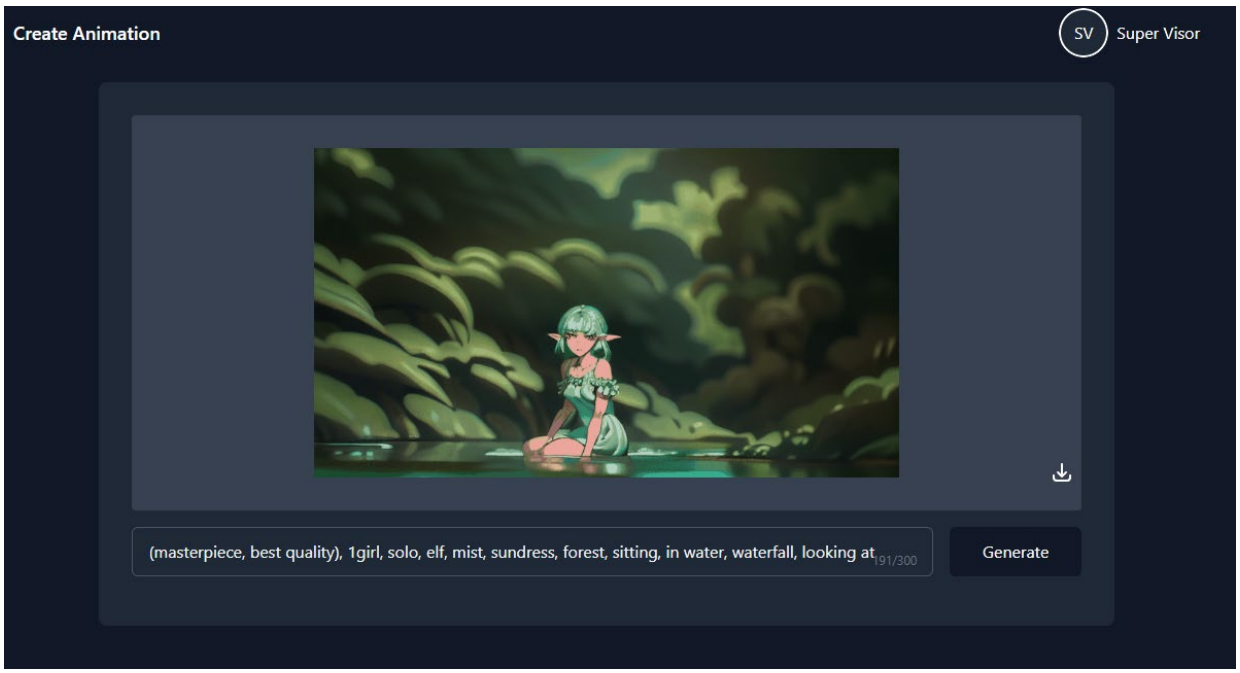

Рисунок 3.11 – Результат процесу генерації

Для перегляду даних користувача, створена окрема сторінка, де можна переглянути актуальну інформацію або відредагувати її. На рисунку 3.12 зображена сторінка налаштування профілю.

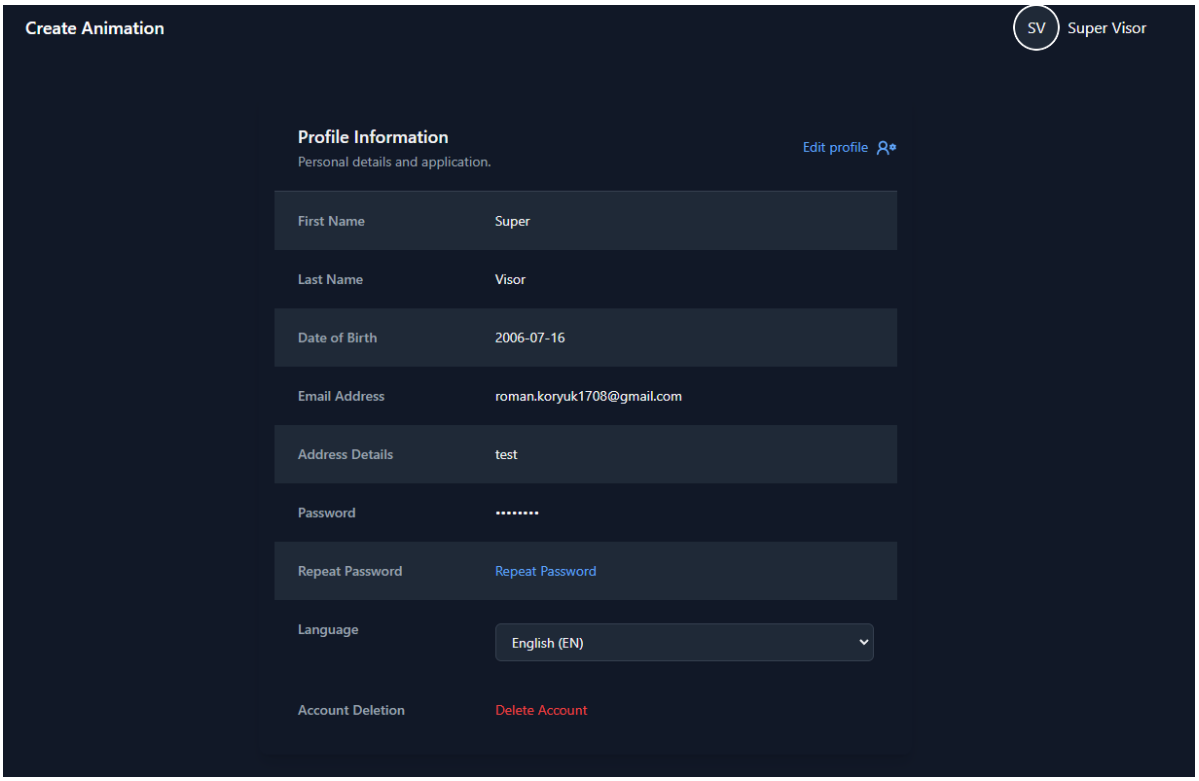

Рисунок 3.12 – Сторінка для налаштування профілю

На даній сторінці є можливість переглядати загальну інформацію, скидати пароль або видаляти акаунт. Також є можливість змінювати інформацію користувача якщо натиснути на кнопку «Edit profile». Сторінка редагування даних профілю зображена на рисунку 3.13.

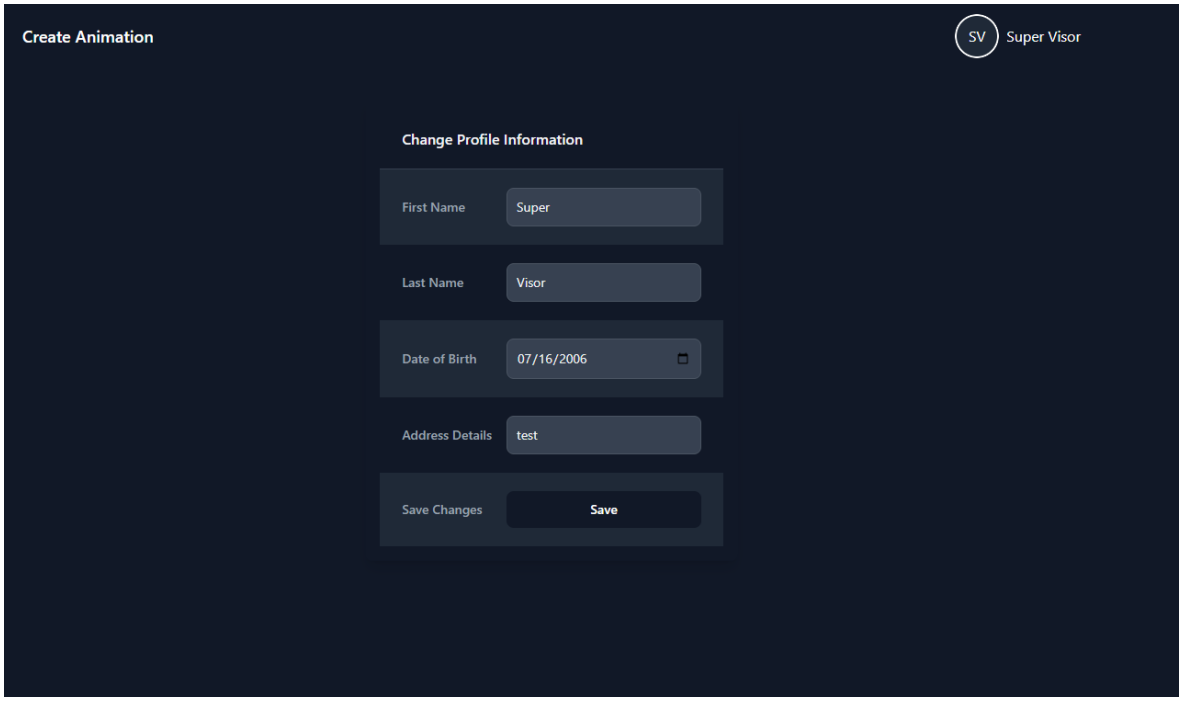

Рисунок 3.13 – Редагування профілю

Отже, при роботі з системою у кожного користувача існує власний ідентифікатор, завдяки якому, сервер розуміє, які дані який користувач хоче відправити або отримати. Без ідентифікатору та ключа доступу, система не дозволить користуватися генерацією.

3.4 Висновки до третього розділу

У розділі 3 було проведено детальний огляд можливостей досліджених інструментів, які дозволять створити ефективну систему для генерації анімованого контенту за допомогою генеративного штучного інтелекту (ГШІ).

Особливу увагу приділено інтеграції ШІ з різними моделями для генерації, такими як моделі дифузії (Diffusion models) та Stable Diffusion, які продемонстрували високу якість результатів у створенні зображень та відео. Розглянуто можливості використання попередньо навчених моделей з бібліотек Transformers та Hugging Face, що дозволяє значно скоротити час розробки та уникнути необхідності тренування моделей з нуля. Проаналізовано підходи до оптимізації процесу генерації, включаючи використання GPU для прискорення обчислень та методи адаптації моделей для конкретних завдань анімації. Також розглянуто використання методу diff-motion-adapter для ефективної анімації.

Детально досліджено структуру та схему роботи системи, починаючи від отримання запиту користувача до відображення згенерованого контенту. Розглянуто процес авторизації користувача, перевірки його прав доступу та генерації токена доступу. Описано етапи обробки запиту, включаючи підготовку моделей ШІ, генерацію анімації, обробку результатів та їх збереження в базі даних. Представлено блок-схеми, що наочно ілюструють взаємодію між різними компонентами системи, такими як веб-додаток, сервер та моделі ШІ. Окрему увагу приділено питанням масштабування системи та забезпечення її стабільної роботи при великій кількості запитів. Також розглянуто способи оптимізації використання обчислювальних ресурсів для мінімізації витрат та забезпечення швидкої відповіді системи.

Також досліджено створення серверу завдяки мові програмування Python з використанням бібліотеки FastAPI. FastAPI було обрано завдяки його високій продуктивності, простоті використання. Розглянуто структуру та функціонал серверу, включаючи обробку HTTP-запитів, валідацію даних, взаємодію з базою даних та моделями ШІ. Досліджено його взаємодію з ГШІ, включаючи передачу параметрів генерації, отримання згенерованих результатів та обробку помилок. Розглянуто питання кешування результатів для зменшення навантаження на сервер та прискорення віддачі контенту користувачам. Також було розглянуто використання ASGI сервера для забезпечення асинхронної роботи веб-додатків, що значно підвищує продуктивність та відповідь системи під час обробки

великої кількості запитів або великих об'ємів даних.

Розглянуто питання з приводу безпеки в системі, включаючи захист від несанкційованого доступу до додатку, використання протоколу OAuth2.0 для авторизації користувачів та генерації токенів доступу, що забезпечує безпечну передачу даних між клієнтом та сервером. Побудовано структуру та створено макет веб-додатку з використанням фреймворку Angular, який отримує та відправляє дані на сервер. Розглянуто структуру веб-додатку, включаючи компоненти для авторизації, генерації контенту, перегляду профілю та інші.

ВИСНОВКИ

Перш за все, розробка веб-додатків із застосуванням генеративного штучного інтелекту є актуальною та значущою темою у сучасному світі технологій, зокрема в Україні. Сфера IT постійно еволюціонує, збільшуючи обсяги даних та складність управління ними, що зумовлює потребу в автоматизації та оптимізації процесів. Застосування генеративного штучного інтелекту може істотно спростити розробку складних і функціонально багатих веб-додатків, підвищити їх продуктивність та забезпечити новітні методи взаємодії з користувачем. Це не тільки підсилює можливості розробників, але й відкриває широкі перспективи для інновацій і креативності в різних сферах, від розваг до освіти та бізнесу.

Отже, проведено огляд актуальності та потреби автоматизованих систем, та аналіз існуючих методів реалізацій їх. Також досліджено переваги та недоліки подібних систем. Розглянуто можливості покращення цих систем в таких аспектах, як покращення якості продукції, ефективності. Зроблено огляд способів інтеграції ШІ, а також детально досліджено інструменти, які допоможуть при створені системи.

Детально розглянуто інструменти, популярні інструменти для створення ГШІ систем. Досліджено їх переваги та недоліки. Обрано інструменти, завдяки яким є можливість створити ефективну систему та забезпечити захист додатку.

Реалізовані обрані інструменти які були обрані для створення ефективної системи. Удосконалена безпека для системи, через надання доступу завдяки наявності генерованого ключу доступу. Розроблено дизайн та структури додатків, які об’єднуються в єдину систему.

Дану систему можливо використовувати майже у будь-якій креативній сфері, так як завдяки ШІ користувач отримує той результат, який задовольнятимуть його потреби. Дозволяє отримати генерувати анімовані зображення для створення унікального контенту. Надає можливість переглядати

та редагувати інформацію користувача. У використанні подібної ГШІ системи є переваги та недоліки, такі як:

- зручність та доступність;
- легкий формат створення контенту;
- швидкість та ефективність.

ГШІ система може бути використаний на різних пристроях, таких як смартфони або планшети, що робить його зручним для користувачів. Він не потребує встановлення окремих програм, оскільки має можливість працювати на потужному віддаленому сервері, завдяки чому не витрачає ресурси пристрою користувача. Може автоматизувати багато повторювальних процесів, таких як створення анімацій вручну, малювання персонажів або будь-якого контенту. Це допомагає збільшити ефективність робочих процесів зменшити ризик помилок.

Незважаючи на переваги, у подібної системи, яка використовує ГШІ, існують певні недоліки, а саме:

- потребує багато обчислювальних потужностей на сервері із-за чого падає продуктивність;
- залежність від мережі.

Оскільки ГШІ працюють у середовищі, якому існує певний ризик порушення конфіденційності даних. Необхідно додатково налаштовувати захист даних та системи в цілому. Можуть мати обмежені можливості в порівнянні з повноцінними платформами, які можуть надають змогу створювати контент вручну, що значно збільшує проміжок часу виконання роботи. При виборі хостингу для системи варто враховувати, що потрібно обирати варіанти з великими обчислювальними потужностями для ШІ, так як при генерації контенту ШІ використовую максимальну потужність серверу для виконання завдання. Та, найголовніше, це залежність ГШІ системи від інтернет мережі, для коректної роботи потрібний постійний та стабільний інтернет зв’язок.

Враховуючи всі недоліки та переваги, використання даної системи для створення анімованого контенту є перспективним напрямком. Проте, необхідно детально проаналізувати потреби індустрії для реалізації подібної системи та

розглянути детально інструменти для розробки ГШІ системи, щоб забезпечити високу продуктивність, та захищеність системи. Дослідити всі можливості забезпечення безпеки для додатку використовуючи інструмент OAuth 2.0, який дозволить створити надійний захист та швидкодію системи в цілому, так як цей інструмент не потребує великої кількості ресурсів.

У межах цієї кваліфікаційної роботи магістра було здійснено глибокий аналіз сфери створення контенту, а також розроблено систему генеративного штучного інтелекту, інтегровану з веб-додатком. Цей веб-додаток призначений для візуалізації роботи ГШІ і взаємодії з ним. Проект зосереджений на створенні зручного інструменту, який дозволяє вирішувати задачі зі створенням контенту високою точністю та ефективністю, використовуючи сучасні технології штучного інтелекту для оптимізації та креативних процесів. Це дозволяє не тільки підвищити продуктивність, але й значно покращити швидкість створення контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Angular Framework. URL: https://angular.dev/overview
2. Python. URL: https://www.python.org/doc/
3. MySQL. URL: https://dev.mysql.com/doc/
4. OAuth 2.0. URL: https://oauth.net/specs/
5. PostgreSQL. URL: https://www.postgresql.org/docs/
6. C++. URL: https://devdocs.io/cpp/
7. C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/
8. Java. URL: https://docs.oracle.com/en/java/
9. pyLLM. URL: https://github.com/HishamAlyahya/PyLLM
10. PySQL. URL: https://pymysql.readthedocs.io/en/latest/
11. Robot Framework. URL: https://robotframework.org/robotframework/
12. JavaSript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript
13. Ionic. URL: https://ionicframework.com/docs
14. PHP. URL: https://www.php.net/docs.php
15. Oracle. URL: https://docs.oracle.com/en/database/oracle/oracle-database/
16. TypeScript. URL: https://www.typescriptlang.org/docs/
17. React. URL: https://legacy.reactjs.org/docs/getting-started.html
18. SSR. URL: https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering
19. Vue.js. URL: https://vuejs.org/guide/introduction.html
20. FastApi. URL: https://fastapi.tiangolo.com/
21. NodeJS. URL: https://nodejs.org/docs/latest/api/
22. PyTorch. URL: https://pytorch.org/docs/stable/index.html
23. CUDA. URL: https://docs.nvidia.com/cuda/
24. Monorepo Structure. URL: https://monorepo.tools/
25. CI/CD. URL: https://docs.gitlab.com/ee/ci/

26. NumPy. URL: https://numpy.org/doc/
27. Pillow Library. URL: https://pillow.readthedocs.io/en/stable/
28. Unicorn. URL: https://www.unicorn-engine.org/docs/
29. Transforms. URL: https://huggingface.co/transformers/v3.0.2/index.html
30. Hugging face. URL: https://huggingface.co/docs
31. Stable Diffusion. URL: https://stablediffusionapi.com/docs/

ДОДАТОК А

КОДИ ОСНОВНИХ ПРОГРАМНИХ МОДУЛІВ

Код ГШІ generator.py:

```
import os
import torch
from diffusers import AnimateDiffPipeline, DDIMScheduler, MotionAdapter
from diffusers.utils import export_to_gif

store_dir = 'images'

LOCAL_ANIMATE_DIFF = "models/animatediff-motion-adapter-v1-5-2"
REMOTE_ANIMATE_DIFF = "guoyww/animatediff-motion-adapter-v1-5-2"
LOCAL_TOON_YOU = "models/toonyou_beta6"
REMOTE_TOON_YOU = "frankjoshua/toonyou_beta6"


def prepare_models() -> AnimateDiffPipeline:

    if os.path.isdir(LOCAL_ANIMATE_DIFF) and
os.path.isdir(LOCAL_TOON_YOU):
        adapter = MotionAdapter.from_pretrained(
            LOCAL_ANIMATE_DIFF, torch_dtype=torch.float16)
        model_id = LOCAL_TOON_YOU
    else:
        adapter = MotionAdapter.from_pretrained(
            REMOTE_ANIMATE_DIFF, torch_dtype=torch.float16)
        model_id = REMOTE_TOON_YOU
    pipe = AnimateDiffPipeline.from_pretrained(
        model_id, motion_adapter=adapter, torch_dtype=torch.float16)
    scheduler = DDIMScheduler.from_pretrained(
        model_id,
        subfolder="scheduler",
        clip_sample=False,
        timestep_spacing="linspace",
        beta_schedule="linear",
        steps_offset=1,
    )
    pipe.scheduler = scheduler

    pipe.enable_vae_slicing()
    pipe.enable_model_cpu_offload()
    return pipe


def gen_image(name: str, prompt: str):
    pipe = prepare_models()
    output = pipe(
        prompt=f"(masterpiece, best quality), {prompt}",
```

```
        negative_prompt="bad quality, worse quality",
        num_frames=32,
        guidance_scale=13,
        num_inference_steps=35,
        generator=torch.Generator("cuda").manual_seed(-1),
        sampling_method="Euler a",
        height= 432,
        width = 768
    )
    frames = output.frames[0]

    export_to_gif(frames, f"{store_dir}/{name}.gif")
    return True
```

Код зв’язку ГШІ з сервером:

```
import logging
import base64
from pydantic import BaseModel
from fastapi import FastAPI,  Depends, HTTPException, status, Security
from fastapi.responses import JSONResponse
from fastapi.security import OAuth2PasswordBearer, HTTPBasic,
HTTPBasicCredentials
from fastapi.middleware.cors import CORSMiddleware
from gen_image import prepare_models, gen_image
from settings_loader import load_settings

logger = logging.getLogger(__name__)
app = FastAPI()
security = HTTPBasic()
settings = load_settings()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE",
                   "OPTIONS"],
    allow_headers=["*"],
)

origins = [
    "http://localhost:8001",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


@app.on_event("startup")
async def startup_event():
```

```
    logging.info("Start prepering ML models")
    prepare_models()
    logging.info("ML models were prepered successfully")


class RequestData(BaseModel):
    prompt: str
    task_id: str


def __image_to_base64__(img_name: str):
    with open(f"images/{img_name}.gif", "rb") as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')


def authenticate_user(
    credentials: HTTPBasicCredentials = Depends(security)
):
    if (credentials.username != settings.USERNAME) or
(credentials.password != settings.PASSWORD):
        raise HTTPException(
            status_code=401,
            detail="Incorrect email or password",
            headers={"WWW-Authenticate": "Basic"},
        )
    return True


@app.post("/api/image-generator/generate")
async def process_data(
    request_data: RequestData,
    authorized: bool = Security(authenticate_user)
):
    prompt = request_data.prompt
    task_id = request_data.task_id

    result = gen_image(name=task_id, prompt=prompt)
    if result:
        return {"base64": __image_to_base64__(task_id)}
```

Код серверу:

```
import logging
from fastapi import FastAPI
from fastapi.responses import JSONResponse
from fastapi.middleware.cors import CORSMiddleware
from .routes import SignUpRouter, LoginRouter, UserProfileRouter,
ImagesRoute
from .models.sign_up import ValidationError
from .routes import (
    SignUpRouter, LoginRouter, UserProfileRouter,
    ImagesRoute, ResetPasswordRoute
)
from .routes.payment import EventListenerRouter, SubscriptionRouter
from .models.sign_up import ValidationError

from fastapi.middleware.cors import CORSMiddleware
from .settings_loader import load_settings

logger = logging.getLogger(__name__)
settings = load_settings()
app = FastAPI(
    openapi_url=settings.OPENAPI_URL,
    title="Gen-Image-Api",
    description="The project goal is the development of a website
allowing users to transform text prompts into customized 2D animations."
)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE",
                   "OPTIONS"],
    allow_headers=["*"],
)


app.include_router(SignUpRouter, tags=["Registration"])
app.include_router(LoginRouter, tags=["Authentication"])
app.include_router(UserProfileRouter, tags=["Profile"])
app.include_router(ImagesRoute, tags=["Images"])
app.include_router(EventListenerRouter, tags=["Payment"])
app.include_router(SubscriptionRouter, tags=["Payment"])
app.include_router(ResetPasswordRoute, tags=["ResetPassword"])

origins = [
    "http://localhost",
    "http://localhost:4200",
    "http://localhost:3000"
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
```

```
    allow_headers=["*"],
)


@app.exception_handler(ValidationError)
async def validation_exception_handler(request, exc):
    return JSONResponse(
        status_code=422,
        content={"detail": exc.name}
    )
```

Код роботи серверу з даними юзера для взаємодіїї з БД:

```
from datetime import datetime

from sqlalchemy import (
    Column, Integer, String, DateTime,
    Date, ForeignKey, Boolean
)
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import Session
from sqlalchemy.orm import relationship, Mapped, mapped_column

from ..models import UserDataScheme

Base = declarative_base()


class RefreshToken(Base):
    __tablename__ = 'refresh_token'
    __table_args__ = {"schema": "users"}

    id: Mapped[int] = mapped_column(primary_key=True)
    refresh_token = Column(String)
    exp_date = Column(DateTime, default=datetime.utcnow)


class User(Base):
    __tablename__ = 'users'
    __table_args__ = {"schema": "users"}

    id: Mapped[int] = mapped_column(primary_key=True)
    email = Column(String)
    password = Column(String)
    created_at = Column(DateTime, default=datetime.utcnow)
    disabled_at = Column(DateTime)
    is_active = Column(Boolean, default=True)
    is_agreed_with_terms = Column(Boolean, default=False)

    profile: Mapped["UserProfile"] = relationship(back_populates="user")


class UserProfile(Base):
    __tablename__ = 'user_profile'
    __table_args__ = {'schema': 'users'}
```

```
    id: Mapped[int] = mapped_column(primary_key=True)
    first_name = Column(String(100))
    second_name = Column(String(100))
    date_of_birthday = Column(Date)
    sub_choice = Column(String(25))
    lang = Column(String(2), default='en')

    # address_details: Mapped["AddressDetails"] =
relationship(back_populates="user_profile")
    address_details = Column(String)
    is_vip = Column(Boolean, default=False)

    user_id: Mapped[int] = mapped_column(ForeignKey("users.users.id"))
    user: Mapped["User"] = relationship(back_populates="profile")


class ResetPassword(Base):
    __tablename__ = 'reset_password'
    __table_args__ = {'schema': 'users'}

    id: Mapped[int] = mapped_column(primary_key=True)
    user_id = Column(Integer)
    created_at = Column(DateTime, default=datetime.utcnow)
    disabled_at = Column(DateTime, default=datetime.utcnow)
    token = Column(String)
    is_active = Column(Boolean)


async def save_new_user(session: Session, user, password: str):
    new_user = User(
        email=user.email,
        password=password,
        is_agreed_with_terms=user.is_agreed_with_terms,
        profile=UserProfile(
            first_name=user.profile.first_name,
            second_name=user.profile.second_name,
            date_of_birthday=user.profile.date_of_birthday,
            sub_choice=user.profile.sub_choice,
            lang=user.profile.lang,
            address_details=user.profile.address_details
        )
    )
    session.add(new_user)
    session.commit()
    session.close()


async def save_refresh_token(session: Session, refresh_token: str,
exp_date: datetime):
    session.add(RefreshToken(refresh_token=refresh_token,
exp_date=exp_date))
    session.commit()
    session.close()


async def remove_refresh_token(session: Session, refresh_token:
RefreshToken):
    session.delete(refresh_token)
    session.commit()
```

```
async def update_user_profile_entity(session: Session, profile:
UserDataScheme, user: User):
    if profile.is_active is not None and not profile.is_active:
        user.is_active = profile.is_active
        user.disabled_at = datetime.now()
    existing_profile = user.profile
    if existing_profile:
        # update params
        if profile.first_name:
            existing_profile.first_name = profile.first_name
        if profile.second_name:
            existing_profile.second_name = profile.second_name
        if profile.date_of_birthday:
            existing_profile.date_of_birthday = profile.date_of_birthday
        if profile.sub_choice:
            existing_profile.sub_choice = profile.sub_choice
        if profile.lang:
            existing_profile.lang = profile.lang
        if profile.address_details:
            existing_profile.address_details = profile.address_details
        # existing_profile.address_details.city =
profile.address_details.city
        # existing_profile.address_details.street =
profile.address_details.street
        # existing_profile.address_details.house =
profile.address_details.house
        # existing_profile.address_details.number =
profile.address_details.number
        # existing_profile.address_details.postal =
profile.address_details.postal
        # existing_profile.address_details.code =
profile.address_details.code
        session.add(existing_profile)
        session.commit()
        # session.close()
    else:
        raise ValueError(
            "User profile with user_id {} not found".format(user.id))


async def find_by_email(session: Session, email: str):
    return session.query(User).filter(User.email == email).first()


async def find_by_user_id(session: Session, user_id: str):
    return session.query(User).filter(User.id == user_id).first()


async def update_account_status(session: Session, user: User, vip: bool,
plan: str):
    user.profile.is_vip = vip
    if vip and plan:
        user.profile.sub_choice = plan
    elif not vip:
        user.profile.sub_choice = None
    session.add(user)
    session.commit()
```

```
    # session.close()


async def update_user_password(session: Session, user: User,
new_password: str):
    user.password = new_password
    session.add(user)
    session.commit()
    session.close()


async def remove_user_account(session: Session, user: User):
    user.disabled_at = datetime.now()
    user.is_active = False
    session.add(user)
    session.commit()
    session.close()


async def save_reset_password_token(session: Session, created_at:
datetime, disabled_at: datetime, user_id: int, token: str):
    reset_password = ResetPassword(
        user_id=user_id, created_at=created_at,
        disabled_at=disabled_at, token=token,
        is_active=True
    )
    session.add(reset_password)
    session.commit()
    # session.close()
```

Код роботи серверу з даними згенерованного конгтенту для взаємодіїї з БД:

```
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import MetaData, Column, Integer, String, DateTime,
Boolean
from datetime import datetime
from sqlalchemy.orm import Session
from . import create_session


metadata = MetaData()
Base = declarative_base(metadata=metadata)


class GenerateImage(Base):
    __tablename__ = 'generate_image'
    __table_args__ = {"schema": "images"}

    id = Column(Integer, primary_key=True)
    task_name = Column(String)
    created_date = Column(DateTime, default=datetime.utcnow)
    user_id = Column(Integer)
    prompt = Column(String)
    status = Column(String)
```

```
def save_gen_image(task_name: str, user_id: int, prompt: str, status:
str) -> GenerateImage:
    session = create_session()
    gen_image = GenerateImage(task_name=task_name,
                              user_id=user_id, prompt=prompt,
status=status)
    session.add(gen_image)
    session.commit()
    session.close()
    return gen_image


def update_gen_image_status(session: Session, gen_image_base:
GenerateImage, status: str):
    # session = create_session()
    gen_image_base.status = status
    session.add(gen_image_base)
    session.commit()
    # session.close()
```