МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ГУМЕНЧУК ПАВЛО СЕРГІЙОВИЧ

Допускається до захисту:
В.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ Оксана ЗЕЛІНСЬКА
«_____» _____ 2025р.

РОЗРОБКА ВЕБ-ІНТЕРФЕЙСУ СИСТЕМИ ДЛЯ РОБОТИ З ГЕНЕРАТИВНИМ ШТУЧНИМ ІНТЕЛЕКТОМ

Спеціальність 122 Комп’ютерні науки
Кваліфікаційна (бакалаврська) робота

Керівник:
Павло РИМАР, старший викладач
кафедри інформаційних технологій
_____

Оцінка: _____/ _____/ _____
(бали за шкалою СКТС/за національною шкалою)
Голова ЕК: _____

Вінниця – 2025

АНОТАЦІЯ

Гуменчук П.С. Розробка веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2025.

У кваліфікаційній роботі досліджено розробку інтегрованого веб-інтерфейсу та API для системи генерації відеоконтенту на основі ГШІ. Представлено архітектуру системи, що включає автентифікацію, налаштування параметрів генерації, ініціацію процесу, перегляд результатів та управління контентом. Описано створення інтерфейсу на Angular та RESTful API на FastAPI для взаємодії клієнтської і серверної частин.

Ключові слова: генеративний штучний інтелект, генерація відео, веб-інтерфейс, взаємодія людина-комп’ютер, FastAPI, API.

57 сторінок, 24 рис., 8 табл., 37 джерел.

ABSTRACT

Humenchuk P.S. Development of a web interface for a system working with generative artificial intelligence. Speciality 122 «Computer Science», educational program «Computer Science». Vasyl’ Stus Donetsk National University, Vinnytsia, 2025.

This qualification thesis investigates the development of an integrated web interface and API for a GenAI-based video content generation system. The system's architecture and implementation are presented, including user authentication, generation parameter configuration, process initiation, result preview, and content management. The development of a user interface using the Angular framework and a RESTful API based on the FastAPI framework for client-server interaction is described.

Keywords: generative artificial intelligence, video generation, web interface, human-computer interaction, FastAPI, API. 57 pages, 24 figures, 8 tables, 37 sources.

ЗМІСТ

ВСТУП

Актуальність дослідження. Сучасний етап розвитку інформаційних технологій характеризується стрімким удосконаленням методів генеративного штучного інтелекту (ГШІ), зокрема для створення високоякісного відеоконтенту.

Одночасно дедалі більше моделей ГШІ доступні як локально встановлювані програми, що значно розширює можливості досвідчених у цій галузі користувачів без постійного підключення до хмарних сервісів та знижує витрати на обчислювальні ресурси. Проте більшість існуючих інструментів часто вимагають глибоких технічних знань, що обмежує їх впровадження в науковій, освітній та творчій сферах.

Водночас неповна інтеграція модулів автентифікації, управління параметрами генерації та попереднього перегляду результатів створює розрив між можливостями моделей ГШІ та зручністю їх практичного застосування.

З наукової точки зору, проблема полягає у відсутності єдиного інтегрованого середовища, яке б забезпечило не лише потужний функціонал API для побудови запитів до моделей, але й зрозумілий користувацький інтерфейс, що відповідає базовим принципам взаємодії людина-комп'ютер (HCI). Вирішення цієї проблеми сприятиме подальшому розвитку досліджень у сфері адаптивних інтерфейсів для мультимедійних застосунків, а також дозволить провести глибший аналіз ефективності різних підходів до візуалізації процесу генерації контенту.

Практична значущість дослідження полягає у створенні прототипу системи, яка охоплює повний цикл роботи з відеогенерацією: від реєстрації та авторизації користувача до налаштування параметрів, ініціації обчислювальних задач і подальшого управління результатами у профілі. Використання RESTful API на базі FastAPI та сучасні фреймворки для реалізації користувацьких інтерфейсів такі як зокрема Angular дозволяє гарантувати масштабованість та високу продуктивність сервісу, а модульність архітектури – легко інтегрувати

нові модулі обробки мультимедіа та алгоритми ГШІ. Реалізація такого рішення відповідає актуальним запитам індустрії креативних технологій та відкриває нові можливості для автоматизації виробництва відеоконтенту.

Мета дослідження полягає в розробці архітектури та ключових компонентів веб-інтерфейсу системи для роботи з генеративним штучним інтелектом, орієнтованої на створення відеоконтенту, що забезпечує інтуїтивну взаємодію, ефективне управління користувацькими даними та процесом генерації. Кінцевою ціллю є подолання існуючого розриву між розширеними можливостями сучасних моделей ГШІ та зручністю їх практичного застосування широким колом користувачів.

Досягнення поставленої мети передбачає вирішення таких **задач дослідження**:

1. Проаналізувати сучасний стан розвитку технологій генеративного штучного інтелекту для створення відеоконтенту та існуючі підходи до розробки користувацьких інтерфейсів для відповідних систем.
2. Обґрунтувати вибір архітектурних рішень та технологічного стеку для розробки клієнтської та серверної частин веб-інтерфейсу, зокрема використання фреймворку Angular для фронтенду та FastAPI для реалізації RESTful API.
3. Спроектувати структуру бази даних, необхідну для зберігання інформації про користувачів системи, їхні активні сесії, історію запитів на генерацію відео, а також метадані згенерованого контенту.
4. Розробити ключові функціональні модулі веб-інтерфейсу, що забезпечують:
    - автентифікацію та авторизацію користувачів;
    - конфігурування широкого спектру параметрів генерації відео;
    - ініціацію процесу генерації;
    - оперативний перегляд проміжних та фінальних результатів;

- управління згенерованим контентом у персональному профілі користувача.

5. Реалізувати зручний RESTful Application Programming Interface (API) для забезпечення гнучкої, ефективної та масштабованої взаємодії між клієнтською частиною (веб-інтерфейсом) та серверною логікою системи.

Об'єктом дослідження є процес розробки веб-інтерфейсу для системи взаємодії з генеративним штучним інтелектом, що спеціалізується на контрольованій генерації відеоконтенту.

Предметом дослідження виступають моделі, методи, технології та інструментальні засоби проектування та реалізації користувацьких веб-інтерфейсів і відповідних програмних інтерфейсів додатків (API) для систем генеративного штучного інтелекту. До предмету також належать принципи взаємодії «людина-комп'ютер» (Human-Computer Interaction, HCI) у контексті ГШІ-систем, архітектурні рішення, спрямовані на забезпечення масштабованості, розширюваності функціоналу та зручності використання таких систем.

Предмет дослідження має дещо міждисциплінарний характер, позаяк знаходиться на перетині кількох ключових галузей інформаційних технологій: веб-розробки (проектування та реалізація клієнтської частини), штучного інтелекту (розуміння базових особливостей роботи генеративних моделей та їхніх параметрів) та взаємодії людини з комп'ютером (забезпечення інтуїтивності, ефективності та зручності використання для користувача). Успішне дослідження в цій комплексній області може сприяти не лише вирішенню конкретної задачі розробки, але й формуванню найкращих практик та рекомендацій для проектування інтерфейсів для широкого класу систем на основі штучного інтелекту, не обмежуючись лише сферою генерації відео. Принципи ефективної взаємодії, виявлені та реалізовані в рамках даної роботи, потенційно можуть бути адаптовані для інших видів систем.

Апробація результатів дослідження. Основні результати дослідження опубліковано у науковій фаховій статті (фаховий журнал категорії Б):

Римар П.В., Гуменчук П.С. Розробка веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. *Наука і техніка сьогодні (Серія «Педагогіка», Серія «Право», Серія «Економіка», Серія «Фізико-математичні науки», Серія «Техніка»)*. 2025.

Результати роботи обговорювалися на VI Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих вчених «Прикладні інформаційні технології 2025» з публікацією тез доповідей:

Гуменчук П.С., Римар П.В. Розробка веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. *Прикладні інформаційні технології 2025*: Матеріали всеукр. науково-практ. конф. здобувачів вищ. освіти та молодих вчен., м. Вінниця, 22 трав. 2025р. 2025.

Структура роботи. Кваліфікаційна (бакалаврська) робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку. Робота містить 24 рисунки, 9 таблиць та 37 літературних джерел.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Постановка задачі

Сучасний етап розвитку інформаційних технологій характеризується стрімким прогресом у галузі генеративного штучного інтелекту (ГШІ), зокрема його застосування для створення відеоконтенту різної якості. Моделі ГШІ демонструють все більш вражаючі можливості, що розкривають для людства нові горизонти для автоматизації як творчих, так і більш технічних і складних процесів, персоналізації контенту та створення інноваційних мультимедійних продуктів. Водночас, зростаюча складність та потужність цих моделей породжує нові виклики, пов'язані із забезпеченням ефективної та доступної взаємодії з ними для широкого кола користувачів.

Аналіз існуючих інструментів для роботи з ГШІ, орієнтованих на генерацію відео, виявляє суттєву проблему: більшість з них вимагають від користувача глибоких технічних знань, специфічних навичок програмування або роботи з командним рядком. Це значно обмежує їх впровадження та використання у науковій, освітній та креативній сферах, де потенційні користувачі не завжди володіють відповідною технічною експертизою. Як наслідок, значна частина потенціалу ГШІ залишається в тіні через високий поріг входження.

Додатковою проблемою є неповна інтеграція ключових функціональних модулів у рамках єдиного програмного середовища. Часто процеси автентифікації користувачів, управління розширеними параметрами генерації, оперативного перегляду результатів та подальшого менеджменту згенерованого контенту є розрізненими або реалізовані не в повному обсязі. Це створює відчутний розрив між теоретичними можливостями сучасних моделей ГШІ та практичною зручністю їх застосування, змушуючи користувачів вдаватися до

використання декількох окремих інструментів, що знижує ефективність та ускладнює робочий процес.

З наукової точки зору, проблема полягає у відсутності єдиного інтегрованого програмного середовища, яке б, з одного боку, надавало потужний та гнучкий програмний інтерфейс додатків (API) для взаємодії з моделями ГШІ, а з іншого – пропонувало інтуїтивно зрозумілий та функціональний користувацький веб-інтерфейс, розроблений з урахуванням базових принципів взаємодії «людина-комп'ютер» (HCI). Таким чином, ключова проблема, на вирішення якої спрямована дана розробка, полягає у подоланні цього розриву між зростаючими обчислювальними та генеративними можливостями моделей ГШІ у сфері створення відео та забезпеченням зручності їх практичного застосування для пересічного користувача.[1,2]

Постановка задачі передбачає розробку не просто графічної «оболонки» для існуючих моделей ГШІ, а створення цілісної екосистеми взаємодії. Таке середовище має забезпечувати безперервний та логічно пов'язаний робочий процес користувача: від моменту першого знайомства з системою та реєстрації, через етапи налаштування параметрів генерації, запуску процесу створення відео, моніторингу його виконання, до отримання, перегляду, збереження та подальшого використання згенерованих результатів. Відсутність такого інтегрованого середовища безпосередньо призводить до зниження продуктивності користувачів, збільшення часу, необхідного на освоєння різноманітних інструментів, та підвищення ймовірності виникнення помилок через необхідність маніпулювати різними програмними компонентами окремо. Успішне вирішення поставленої задачі може не тільки задовольнити потреби конкретних користувачів, але й встановити певний стандарт для інтерфейсів ГШІ-систем майбутнього, де акцент зміщується з простої демонстрації технічних можливостей самої моделі штучного інтелекту на забезпечення продуктивної, комфортної та ефективної роботи кінцевого користувача.

1.2 Огляд сучасних технологій генеративного штучного інтелекту для створення відео

Генеративний штучний інтелект для створення відео – є галуззю, що динамічно розвивається, спираючись на досягнення в глибинному навчанні та нейронних мережах. Серед ключових технологічних підходів, що використовуються для генерації відео, можна виділити декілька основних:

Генеративно-змагальні мережі (GANs). Цей підхід, запропонований Яном Гудфеллоу та його колегами , полягає у взаємодії двох нейронних мереж – генератора та дискримінатора. Генератор намагається створити реалістичні відеокадри, а дискримінатор – відрізнити їх від справжніх. З часом обидві мережі вдосконалюються, що призводить до генерації все більш якісного відео. Модифікації GAN, такі як StyleGAN та його наступники, продемонстрували значний прогрес у генерації високоякісних зображень, і ці принципи активно адаптуються для відео.[3,4]

Дифузійні моделі (Diffusion Models), що стали надзвичайно популярними останнім часом завдяки своїй здатності генерувати високоякісні та різноманітні зображення, а тепер і відео. Процес генерації в дифузійних моделях полягає у поступовому знешумленні випадкового шуму до отримання осмисленого зображення або послідовності кадрів, керуючись текстовим описом або іншими вхідними даними.[5]

Моделі на основі трансформерів (Transformer-based Models). Дана архітектура, що на початку здійснила революцію в обробці природної мови, зараз успішно застосовується і для генерації відео. Трансформери здатні вловлювати довготривалі залежності в послідовностях даних, що є критично важливим для створення когерентних та логічно пов'язаних відеороликів.

Авторегресійні моделі які генерують відео кадр за кадром (або піксель за пікселем), де кожен наступний елемент залежить від попередніх. Хоча вони можуть створювати деталізований контент, їх обчислювальна складність часто є високою.[6]

Сучасні моделі ГШІ для відео демонструють значні досягнення щодо якості генерованого контенту, роздільної здатності, тривалості відеороликів та можливостей керування процесом генерації (наприклад, за допомогою текстових підказок, вхідних зображень або скриптів). Важливим аспектом в популяризації є зростаюча доступність цих моделей. Окрім хмарних сервісів, що надають доступ до потужних обчислювальних ресурсів, все більше моделей стають доступними у вигляді локально встановлюваних програм, що розширює можливості для досвідчених користувачів та дослідників, дозволяючи працювати без постійного підключення до Інтернету та потенційно знижуючи ресурси на обчислення.

Однак, незважаючи на стрімкий прогрес, технології ГШІ для відео залишаються складними як у теоретичному розумінні, так і в практичному застосуванні. Їх навчання вимагає величезних масивів даних та значних обчислювальних ресурсів. Налаштування параметрів генерації для отримання бажаного результату часто є нетривіальним завданням, що вимагає експериментування та глибокого розуміння роботи моделі. Саме ця складність та високі вимоги до ресурсів посилюють потребу в розробці ефективних та інтуїтивно зрозумілих користувацьких інтерфейсів. Чим потужнішими та гнучкішими стають моделі ГШІ (з більшою кількістю параметрів, складнішими опціями налаштування), тим складнішим стає завдання для розробників користувацьких інтерфейсів: надати користувачеві повний контроль над цими можливостями, не перевантажуючи його надмірною кількістю опцій та технічних деталей, і забезпечити при цьому інтуїтивно зрозумілу та вичерпну взаємодію.

1.3 Порівняння аналогів

Для більш глибокого розуміння контексту розробки та обґрунтування унікальності власного рішення, було проведено аналіз низки популярних

інструментів та платформ, що використовуються для генерації відеоконтенту на основі технологій штучного інтелекту.

Pictory AI.

Pictory AI є веб-платформою, що позиціонується як інструмент для автоматичного створення коротких відео з текстових джерел (статті, скрипти) або наявних відеоматеріалів. Інтерфейс зазвичай пропонує покроковий процес: завантаження або введення тексту, вибір візуальних стилів, автоматичний підбір стокових відео/зображень, додавання озвучення (генерованого ШІ або власного) та брендування. Користувачі взаємодіють з візуальними редакторами для коригування сцен, тексту та таймінгів.[7]

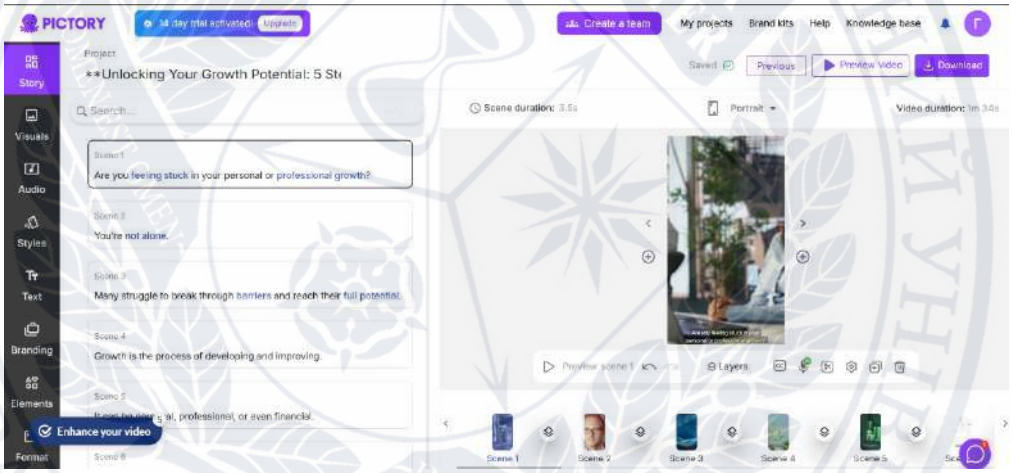

Рисунок 1.1 – Інтерфейс Pictory AI

Переваги:

- Швидкість створення відео з існуючого контенту.
- Автоматизація підбору візуальних матеріалів зі стокових бібліотек.
- Вбудовані інструменти для генерації голосу та додавання музики.
- Відносно низький поріг входження для користувачів без досвіду відеомонтажу.

Недоліки:

- Обмежена гнучкість у кастомізації візуального ряду порівняно з більш просунутими генеративними моделями.
- Якість автоматично підібраних стокових матеріалів не завжди може відповідати специфіці тексту.
- Залежність від хмарної платформи та її обчислювальних ресурсів, що може впливати на вартість та швидкість обробки для великих обсягів.
- Менший контроль над глибинними параметрами генерації, якщо порівнювати з інструментами, що працюють безпосередньо з моделями ГШІ.
- Дуже багато візуального шуму, хоч і багато можливостей, але легко заплутатись в інтерфейсі.

Synthesia.

Synthesia – це веб-платформа, що спеціалізується на створенні відео з фотореалістичними або стилізованими ШІ-аватарами. Користувацький інтерфейс дозволяє обирати аватар з бібліотеки, вводити текст, який аватар буде озвучувати, обирати мову та голос. Платформа надає шаблони та можливість базового редагування фону, додавання тексту та медіаелементів на екран.[8]

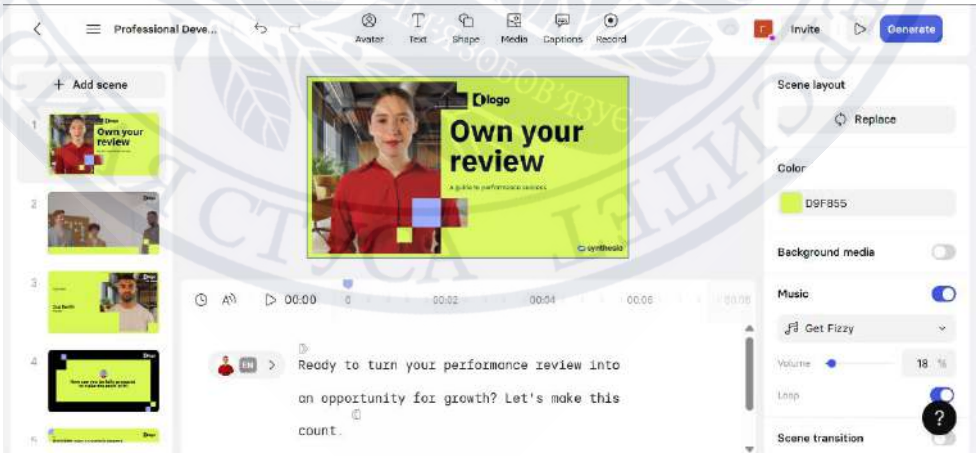

Рисунок 1.2 – Інтерфейс Synthesia

Переваги:

- Можливість створення відео з «ведучими» без необхідності зйомки реальних людей.
- Велика бібліотека доступних аватарів та голосів багатьма мовами.
- Простий у використанні інтерфейс, орієнтований на швидке створення презентаційних та навчальних відео.
- Можливість оновлення відео шляхом простої зміни тексту.

Недоліки:

- Висока вартість при регулярному використанні, особливо для розширеного функціоналу.
- Обмежена кастомізація самих аватарів (рухи, емоції можуть бути виключно шаблонними).
- Якість синхронізації губ та природність мовлення аватарів може варіюватися.
- Повна залежність від хмарної платформи та її моделей.
- Відсутність інструментів для роботи з більш абстрактною або художньою генерацією відео, не пов'язаною з аватарами.

UI для Stable Diffusion (Automatic1111)

Це переважно локально встановлювані веб-інтерфейси (або десктопні програми з веб-технологіями), які надають доступ до широкого спектру налаштувань моделі Stable Diffusion для генерації зображень та, за допомогою розширень (як Deforum), відео. Інтерфейси зазвичай містять численні вкладки, текстові поля для промптів (позитивних та негативних), слайдери, випадаючі списки для керування параметрами (steps, CFG scale, sampler, seed, роздільна здатність, моделі, LoRA тощо).[9]

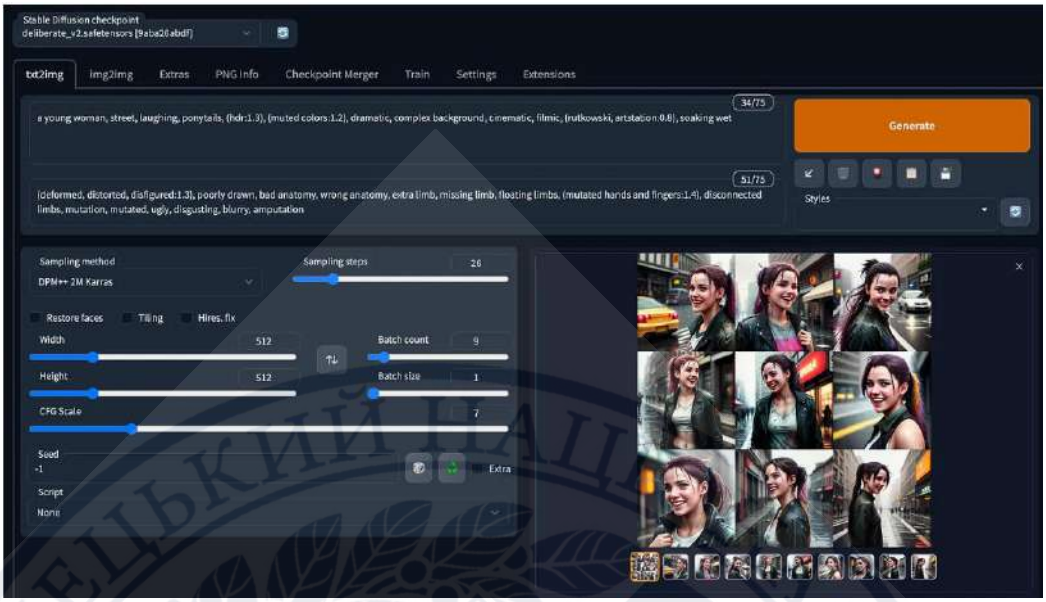

Рисунок 1.3 – Інтерфейс UI для Stable Diffusion

Переваги:

- Надзвичайно високий рівень контролю над процесом генерації завдяки доступу до багатьох параметрів моделі.
- Можливість використання власних моделей, чекпойнтів, LoRA та інших кастомних розширень.
- Активна спільнота, що постійно розробляє нові функції та розширення.
- Безкоштовне використання (окрім витрат на власні обчислювальні ресурси).
- Локальне виконання, що забезпечує приватність даних.

Недоліки:

- Дуже високий поріг входження для користувачів без технічних знань та розуміння принципів роботи Stable Diffusion.
- В інтерфейсі дуже багато візуального шуму різними опціями, що може дезорієнтувати пересічних користувачів.
- Відсутність централізованого управління користувачами, їхніми роботами, сесіями (кожен запуск часто є ізольованим).

- Масштабованість та API для інтеграції в інші системи зазвичай не передбачені «з коробки».
- Якість та стабільність роботи залежать від конфігурації локального середовища.

Fliki AI

Fliki AI є веб-платформою, яка дозволяє перетворювати текст (статті, блоги, скрипти) на відео з використанням стокових медіа та ШІ-генерованих голосів. Інтерфейс схожий на Pictory AI: користувач вводить текст, система розбиває його на сцени, підбирає візуальні матеріали (відео, зображення) та пропонує варіанти озвучення. Є можливість редагувати сцени, замінювати медіа, налаштовувати голос та музику.[10]

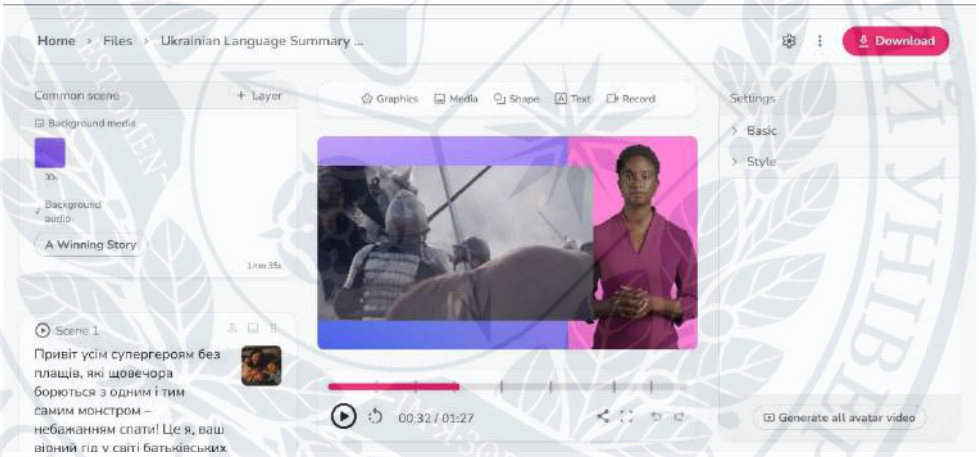

Рисунок 1.4 – Інтерфейс Fliki AI

Переваги:

- Простий та швидкий спосіб створення відео з текстового контенту.
- Велика бібліотека ШІ-голосів та підтримка багатьох мов.
- Інтеграція зі стоковими бібліотеками медіа.
- Інтуїтивно зрозумілий інтерфейс для користувачів без навичок відеомонтажу.

Недоліки:

- Обмежені можливості для створення унікального візуального стилю, оскільки переважно використовуються стокові матеріали.
- Менший контроль над параметрами генерації голосу порівняно зі спеціалізованими TTS-системами.
- Залежність від хмарної платформи та її тарифних планів.
- Як і інші подібні платформи, може не задовольняти потреби користувачів, які прагнуть працювати з власними моделями ГШІ або мати глибокий контроль над процесом генерації.

Переваги розробленого проекту над аналогами

Розроблюваний веб-інтерфейс системи для роботи з генеративним штучним інтелектом має на меті запропонувати рішення, яке враховує недоліки існуючих аналогів та надає користувачам низку суттєвих переваг таких як:

1. Баланс між простотою використання та гнучкістю налаштувань. Інтерфейс проектується інтуїтивно зрозумілим для користувачів без глибоких технічних знань, водночас надаючи доступ до широкого спектру параметрів генерації (обширний вибір мов та голосів на відміну від аналогів, орієнтація, якість або навіть автоматичний пошук новин і генерація відео на їх основі), що дозволяє більш точно контролювати кінцевий результат порівняно з деякими надто спрощеними SaaS-рішеннями.

2. Власна серверна логіка та API. Наявність RESTful API, розробленого на FastAPI, забезпечує гнучку взаємодію між клієнтською та серверною частинами. Це відкриває можливості для масштабування системи, інтеграції з різними моделями ГШІ (включаючи потенційно локальні) та розробки альтернативних клієнтських застосунків, чого позбавлені багато закритих платформ або прості локальні UI.

3. Комплексне управління користувацькими даними. Система передбачає реєстрацію, автентифікацію, управління профілем, історією запитів та згенерованим контентом. Це вирішує проблему відсутності таких можливостей у багатьох локальних UI для Stable Diffusion та інших подібних інструментів.

4. Потенціал для кастомізації та розширення. Модульна архітектура як клієнтської (Angular), так і серверної частини (FastAPI) спрощує подальший розвиток системи, додавання підтримки нових моделей ГШІ, розширення набору параметрів та інтеграцію з зовнішніми сервісами (наприклад, новинними порталами).

Таким чином, розроблюваний проект спрямований на створення більш універсального, контрольованого та зручного інструменту для роботи з генеративним штучним інтелектом у сфері відеогенерації, заповнюючи прогалини, наявні в існуючих рішеннях.

Висновок до розділу 1

У даному розділі було здійснено аналіз предметної області, пов’язаної з розробкою веб-інтерфейсу для системи яка працює із генеративним штучним інтелектом, орієнтованого на створення відеоконтенту. Визначено ключову проблему, що полягає у розриві між зростаючими можливостями ГШІ та зручністю їх практичного застосування, особливо для користувачів без глибоких технічних знань. Проведено огляд сучасних технологій ГШІ для генерації відео, включаючи GAN, дифузійні моделі та моделі на основі трансформерів, а також проаналізовано низку існуючих аналогічних платформ та інструментів, таких як Pictory AI, Synthesia, UI для Stable Diffusion та Fliki AI. Це дозволило виявити їхні переваги та недоліки, зокрема, обмежену гнучкість, високий поріг входження або відсутність комплексного управління користувацькими даними та історією запитів. На основі проведеного аналізу було сформульовано та обґрунтовано постановку задачі даної кваліфікаційної роботи, спрямованої на розробку інтегрованого та рішення орієнтованого на пересічного.

У наступному розділі буде детально розглянуто обрані методи, алгоритми, технології та інструменти, що були застосовані для проектування та розробки системи.

РОЗДІЛ 2
ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ТА РІШЕНЬ

2.1. Опис використаних методів дослідження та розробки

При виконанні даної кваліфікаційної роботи було застосовано комплекс методів, що дозволили вирішити поставлені задачі та досягти мети дослідження. Серед основних методів можна виділити наступні:

Аналіз науково-технічної літератури та існуючих рішень. Цей метод був застосований на початковому етапі для визначення актуальності проблеми, аналізу стану предметної області, виявлення переваг та недоліків існуючих аналогів веб-інтерфейсів для систем генеративного штучного інтелекту. Результати аналізу лягли в основу постановки задачі та формування вимог до розроблюваної системи.

Системний аналіз та проектування. Використовувався для декомпозиції загальної задачі на окремі підзадачі, визначення архітектури системи, її основних компонентів та зв'язків між ними. На цьому етапі було спроектовано логічну структуру бази даних та визначено основні програмні інтерфейси (API).

Об’єктно-орієнтоване проектування (ООП). Позаяк фреймворк Angular та мова TypeScript, що є його технологічною основою, за своєю архітектурою та функціональними можливостями значною мірою спираються на принципи об'єктно-орієнтованого програмування (ООП), ці ж принципи були застосовані при розробці програмних компонентів клієнтської частини. Зокрема, це знайшло відображення у проектуванні та реалізації компонентів, сервісів та моделей даних засобами TypeScript в середовищі Angular.[11]

Метод прототипування. При розробці системи було застосовано метод ітераційного прототипування. Цей підхід полягав у послідовному створенні та аналізі функціональних прототипів окремих програмних модулів та елементів користувацького інтерфейсу. Кожна ітерація включала етапи визначення вимог до поточної версії прототипу, його розробку, тестування ключових аспектів

функціональності або дизайну, та подальший рефакторинг або внесення необхідних коректив на основі отриманих результатів.

Такий ітераційний процес дозволив на ранніх стадіях розробки здійснювати валідацію архітектурних та функціональних рішень, оперативно ідентифікувати потенційні проблеми та вносити зміни, що сприяло зниженню загальних ризиків проекту та забезпечувало поступове, контрольоване нарощування функціоналу системи. Застосування прототипування забезпечило гнучкість процесу розробки та сприяло більш точному формуванню вимог до кінцевого продукту.

Методи розробки програмного забезпечення. Процес розробки системи базувався на сучасних інженерних підходах та методологіях, спрямованих на забезпечення якості, гнучкості та підтримуваності програмного продукту. Включаючи компонентний підхід (Angular), використання систем контролю версій (Git), а також принципи REST для проектування API.

2.2. Огляд використаних алгоритмів та технологій

Для реалізації поставлених завдань було обрано набір сучасних технологій та програмних інструментів, що забезпечують ефективність розробки, надійність та масштабованість системи.

Технології клієнтської частини

Angular (MVVM). Ключовим аспектом при розробці клієнтської частини на основі фреймворку Angular стало застосування компонентно-орієнтованого підходу. Цей підхід, що реалізує архітектурний шаблон MVVM (Model-View-ViewModel), дозволив декомпонувати користувацький інтерфейс на незалежні, повторно використовувані компоненти. Одна із найважливіших особливостей в Angular архітектурі, що дає перевагу над іншими технологіями це те, що кожен компонент інкапсулює власну логіку відображення (HTML, SCSS, Bootstrap) та поведінки, що реалізується засобами TypeScript. Такий підхід значно спростив розробку складних інтерфейсів, їх тестування та подальшу модифікацію.

Використання RxJS забезпечило ефективне управління асинхронними потоками даних та подіями в клієнтському застосунку.[12]

Переваги. Компонентна архітектура, двостороннє зв'язування даних, потужна система маршрутизації, інструменти для роботи з реактивними формами та саме реактивне програмування, вбудована підтримка TypeScript, активна спільнота.

TypeScript. Мова програмування, що є надмножиною JavaScript та додає статичну типізацію. Використання TypeScript дозволяє підвищити надійність коду, виявляти помилки на етапі компіляції та покращити підтримуваність та розширення великих проектів.[13]

RxJS (Reactive Extensions for JavaScript). Бібліотека для реактивного програмування з використанням спостережуваних послідовностей (Observables). В Angular RxJS активно використовується для обробки асинхронних операцій, таких як HTTP-запити, управління подіями та станом застосунку.[14]

HTML (HyperText Markup Language) та SCSS (Sassy CSS). HTML використовується для структурування контенту веб-сторінок. SCSS, як препроцесор CSS, застосовується для написання більш гнучкого, структурованого та підтримуваного коду стилів, надаючи можливості використання змінних, вкладеності, міксинів, циклів тощо.

Bootstrap. Для оптимізації процесу розробки користувацького інтерфейсу та забезпечення його адаптивності було інтегровано CSS-фреймворк Bootstrap. Застосування даного інструментарію дозволило ефективно використовувати попередньо розроблені та стилізовані компоненти, такі як навігаційні панелі, кнопки, форми введення та модальні вікна, що суттєво прискорило верстку сторінок. Одною із ключових переваг використання Bootstrap стала його вбудована система сіток (grid system), яка забезпечила створення гнучких та адаптивних макетів, що коректно відображаються на пристроях з різною роздільною здатністю екрана – від настільних комп'ютерів до мобільних

телефонів. Це сприяло формуванню візуально привабливого, консистентного та інтерфейсу системи орієнтованого на пересічного користувача.[15]

Технології проміжної частини між сервером та клієнтом (API)

FastAPI. Сучасний веб-фреймворк для створення API мовою Python. Обраний через його високу продуктивність (завдяки асинхронності на базі ASGI та Starlette), автоматичну генерацію документації API (Swagger UI), валідацію даних за допомогою Pydantic-моделей та зручну систему ін'єкції залежностей.[16]

Переваги. Асинхронність, швидкість, простота використання, автоматична валідація та документація, підтримка типів даних для Python.

JWT (JSON Web Tokens). Технологія для створення токенів доступу, що використовуються для реалізації механізму автентифікації та авторизації користувачів. На сервері для генерації та валідації токенів використовувалась бібліотека python-jose.[17]

Алгоритми та підходи

- **RESTful API Design.** Серверний API спроектовано відповідно до принципів REST, що передбачає використання стандартних методів HTTP протоколу такі як: GET, POST, PUT, DELETE для операцій над ресурсами, ідентифікованими за URL.[18,19]
- **Автентифікація на основі токенів (JWT).** Реалізовано стандартний потік автентифікації: користувач надає облікові дані, у відповідь отримує JWT, який потім використовується для доступу до захищених ресурсів шляхом передачі у заголовку Authorization кожного запиту.
- **Хешування паролів з використанням солі (bcrypt).** Для безпечного зберігання паролів користувачів використовується алгоритм bcrypt, який автоматично генерує сіль для кожного пароля перед хешуванням.
- **Асинхронна обробка запитів.** FastAPI дозволяє визначати асинхронні обробники запитів (async def), що є важливим для операцій, які можуть

бути тривалими (наприклад, взаємодія з зовнішніми сервісами або моделями ГШІ).

- **Фонові задачі (BackgroundTasks):** Для виконання довготривалих операцій, таких як безпосередньо процес генерації відео, що не потребують негайної відповіді клієнту, використовується механізм фонових задач FastAPI. Це дозволяє уникнути блокування основного потоку обробки запитів та покращити відгук системи.

2.3. Огляд інструментів розробки

У процесі розробки системи було використано наступні інструменти:

Інтегроване середовище розробки (IDE). Visual Studio Code – багатоплатформний редактор коду з потужною підтримкою TypeScript, Python, вбудованим терміналом, інструментами для відлагодження та системою розширень.

Система контролю версій. Git – розподілена система контролю версій для відстеження змін у коді, спільної роботи та управління версіями проекту не залежно від операційної системи. Для хостингу репозиторію, що в свою чергу дозволило розділити розробку на серверну і клієнтську частини для роботи в парі, для цього використовувалась платформа GitHub.[20]

Менеджери пакетів:

- **npm (Node Package Manager).** Для управління залежностями та скриптами у клієнтському Angular-проекті.
- **pip (Python Package Installer).** Для управління залежностями у серверному FastAPI-проекті.

Інструменти командного рядка:

- **Angular CLI.** Інтерфейс командного рядка для створення, розробки, тестування та збірки Angular-застосунків. Забезпечує швидку і зручну генерацію компонентів, сервісів, модулів, пайпів, інтерфейсів тощо.

- **Uvicorn.** ASGI-сервер для запуску FastAPI-застосунків та самого сервера безпосередньо під час розробки.

Інструменти для тестування API. Вбудована інтерактивна документація FastAPI (Swagger UI) для ручного тестування ендпоінтів. Також використовувався інструмент Postman, що розширює можливості для тестування API.

Висновок до розділу 2

У другому розділі роботи було детально описано методологічну та технологічну базу дослідження і розробки. Обґрунтовано вибір комплексу методів, що включають аналіз науково-технічної літератури, системний аналіз, об’єктно-орієнтоване проектування та метод ітераційного прототипування, який дозволив на ранніх етапах валідувати ключові рішення. Проведено огляд використаних технологій для клієнтської частини, зокрема фреймворку Angular з архітектурним шаблоном MVVM, мови TypeScript, бібліотеки RxJS, а також HTML, SCSS та CSS-фреймворку Bootstrap для забезпечення адаптивного, привабливого та зручного інтерфейсу орієнтованого на пересічного користувача. Для передачі даних між серверною та клієнтською частинами (API) обґрунтовано вибір фреймворку FastAPI, технології JWT для автентифікації та алгоритму bcrypt для хешування паролів. Також представлено короткий огляд основних інструментів розробки, таких як IDE Visual Studio Code, система контролю версій Git, менеджери пакетів npm та pip, ASGI-сервер Uvicorn та інструменти для тестування API. Обраний стек технологій та інструментів створює надійне підґрунтя для реалізації поставлених завдань.

Наступний розділ буде присвячено детальному опису повної програмної реалізації веб-інтерфейсу системи генерації відеоконтенту, включаючи опис функціональних можливостей, архітектури застосунку, проектування бази даних та реалізації API.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ІНТЕРФЕЙСУ СИСТЕМИ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ

3.1 Опис функціональних можливостей веб-інтерфейсу

Розроблений веб-інтерфейс надає користувачеві набір функцій для повного циклу роботи з генерацією відеоконтенту, починаючи від реєстрації та закінчуючи управлінням згенерованими матеріалами. При проєктуванні інтерфейсу особлива увага приділялася принципам взаємодії «людина-комп’ютер» (HCI) та користувацького досвіду (UX), щоб забезпечити інтуїтивність, контроль та ефективність, особливо важливі при роботі зі складними системами на базі ГШІ. Система прагне мінімізувати когнітивне навантаження на користувача, надаючи чіткий зворотний зв'язок та передбачувані результати дій.

Модулі реєстрації та автентифікації користувачів

Доступ до основного функціоналу системи вимагає реєстрації та автентифікації користувача, що забезпечує персоналізацію досвіду та безпеку даних.

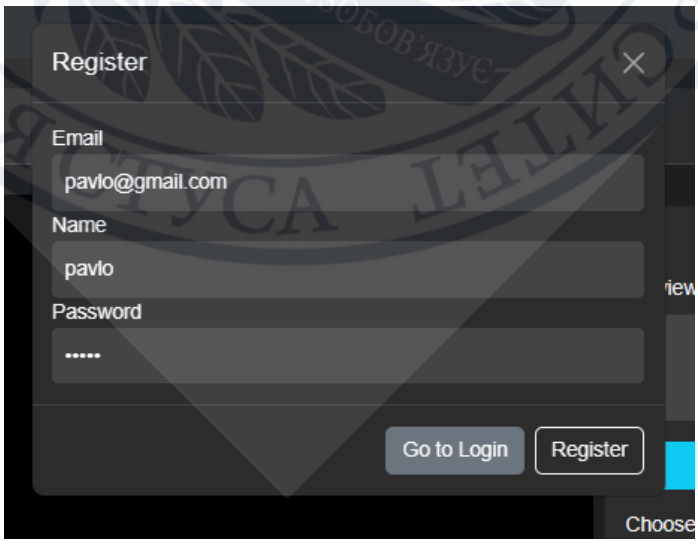

Рисунок 3.1 – Модуль реєстрації користувача

Інтерфейсні елементи модулю. Форма з полями для введення логіна (name), адреси електронної пошти (email), пароля (password). Кнопка «Register» та кнопка для переходу або повернення до модулю логіну «Go to Login» для вже зареєстрованих користувачів.

Взаємодія. Користувач заповнює поля. Здійснюється клієнтська валідація даних (наприклад, коректність формату email, мінімальна довжина пароля, співпадіння паролів). При відправці форми дані надсилаються на сервер. Сервер виконує серверну валідацію (наприклад, унікальність логіна та email) та у разі успіху створює новий обліковий запис, хешуючи пароль з використанням bcrypt та солі. Користувач отримує повідомлення про успішну реєстрацію або про помилки.

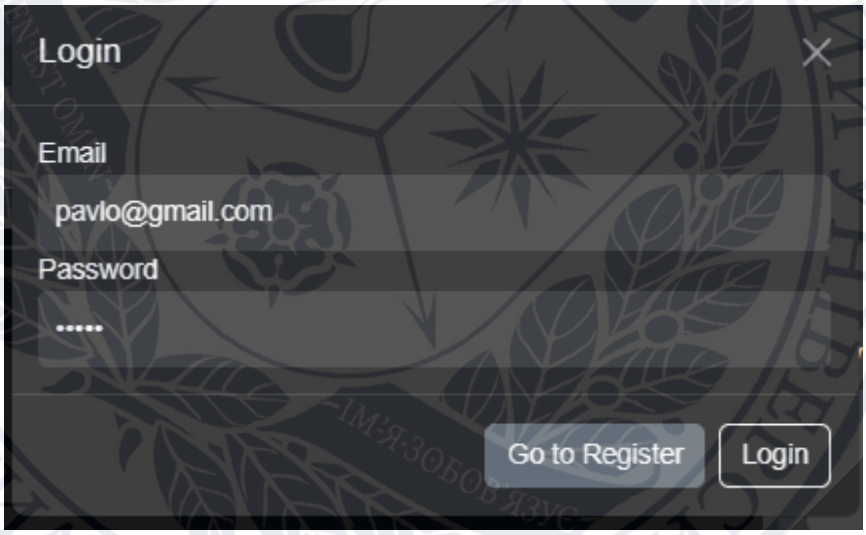

Рисунок 3.2 – Модуль авторизації користувача

Інтерфейсні елементи модулю. Форма з полями для введення логіна (email) та пароля (password). Кнопка «Login». Посилання на сторінку реєстрації для нових користувачів «Go to Register».

Взаємодія. Користувач вводить свої облікові дані. Дані надсилаються на сервер. Сервер перевіряє відповідність наданих даних збереженим у базі (порівнюючи хеш введеного пароля зі збереженим). У разі успішної автентифікації генерується JWT (JSON Web Token), який повертається на

клієнтську частину де зберігається токен у localStorage та базу даних для управління сесіями і використовує його для подальших запитів до захищених ресурсів API, передаючи у заголовку Authorization. Користувач перенаправляється на головну сторінку системи. У разі невдачі – виводиться повідомлення про помилку.

Головна сторінка та перегляд згенерованого відео

Після успішної автентифікації користувач потрапляє на головну сторінку системи. Ця сторінка слугує центральним робочим простором для взаємодії з основним функціоналом – генерацією та переглядом відео.

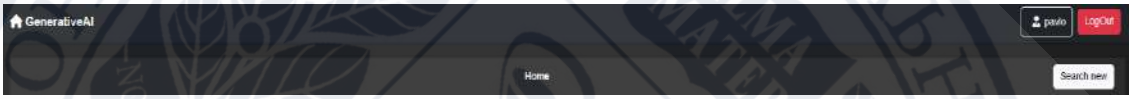

Рисунок 3.3 – Основна та другорядна навігаційні панелі

Навігаційна основна панель (Header). Розташована у верхній частині сторінки. Містить логотип системи, посилання на сторінку профілю користувача («My Videos») та кнопку для виходу з системи (Logout).

Другорядна навігаційна панель. Виконує роль навігаційної панелі для кожної сторінки окремо з індивідуальними функціями. Містить кнопку, що викликає панель для пошуку новин та тексту із вижимкою основного матеріалу див. рис. 3.4.

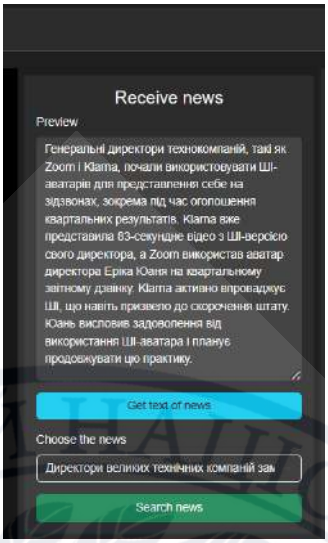

Рисунок 3.4 – Прихована панель для пошуку актуальних новин.

Інтерфейсні елементи прихованої панелі для пошуку новин

Поле для виводу вижимки із основного тексту новини. Сама кнопка отримання тексту. Селектор для вибору потрібної новини. Кнопка пошуку новин.

Взаємодія. При натисканні кнопки «Search new» на другорядній навігаційній панелі, з’являється панель (рис. 3.4). Натискаючи кнопку зеленого кольору «Search news» серверу надсилається запит на пошук релевантних та актуальних новин. Після чого, над кнопкою в випадаючому списку можна вибрати одну із новин. Натискаючи на кнопку «Get text of news» сервер робить аналіз новини, вибирає основну і важливу інформацію і повертає текст який ми можемо побачити в полі для виводу як показано на рис. 3.4.

Основна робоча область. Центральну частину сторінки займає відеоплеєр. Якщо користувач щойно згенерував відео, воно автоматично і миттєво відображається тут.

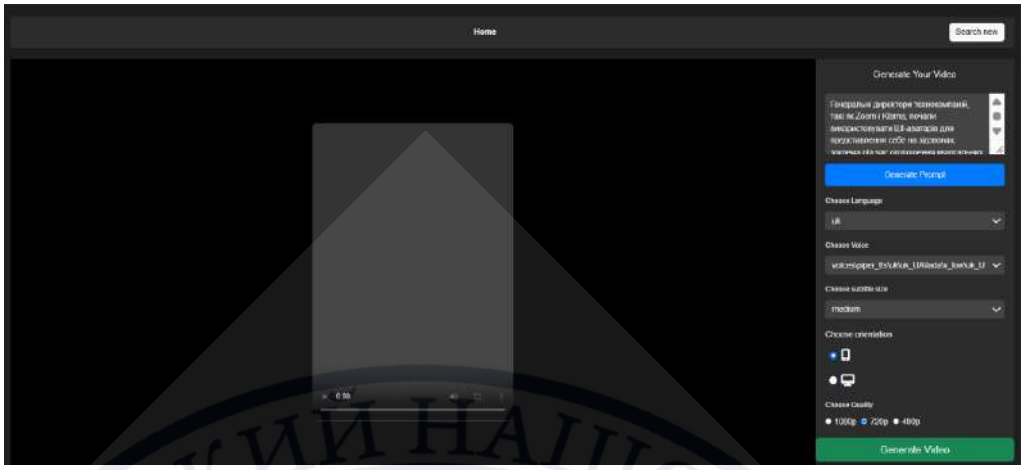

Рисунок 3.5 – Повна сторінка із робочою областю та панеллю налаштувань.

Відеоплеєр. Надає стандартні елементи керування відтворенням: кнопка «Відтворити/Пауза», таймлайн для навігації по відео, регулятор гучності, кнопка для переходу в повноекранний режим. Згенероване відео відображається разом із субтитрами (якщо їх було увімкнено при генерації) та відеорядом, підібраним або згенерованим ГШІ. Також сам відеоплеєр може змінювати форму (див. рис. 3.6 та рис. 3.7) в залежності яку орієнтацію вибере користувач портретну (для мобільних приладів) та ландшафтну(для телевізорів, комп’ютерів тощо).

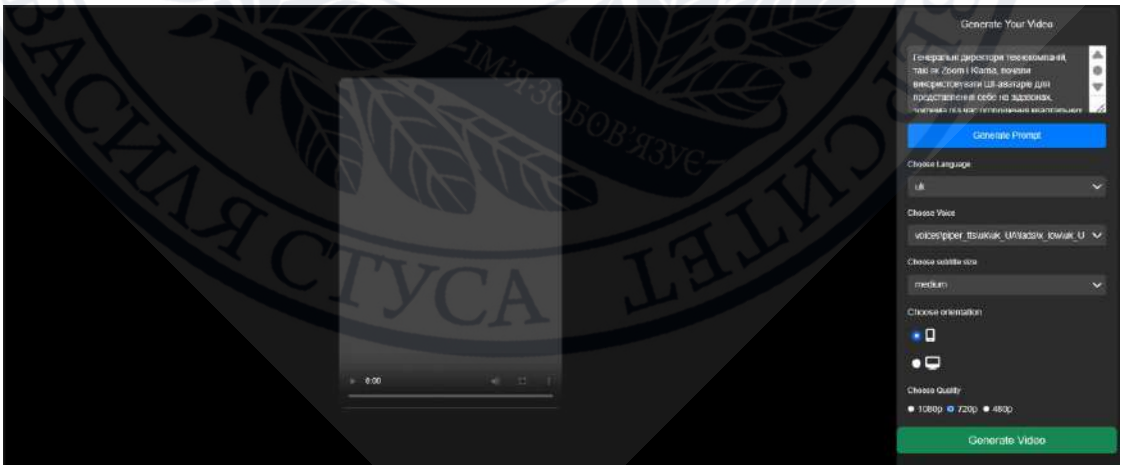

Рисунок 3.6 – Портретна орієнтація відеоплеєра

Рисунок 3.7 – Ландшафтна орієнтація відеоплеєра

Користувач може переглядати згенероване відео, використовуючи елементи керування плеєром. Головна сторінка забезпечує швидкий доступ до результатів генерації та до інструментів для створення нового контенту.

Дизайн головної сторінки спрямований на те, щоб користувач міг одразу зосередитися на ключових завданнях: перегляді результатів та налаштуванні нової генерації.

Панель налаштувань параметрів генерації відео

Права бічна панель є ключовим елементом для взаємодії користувача з генеративним функціоналом системи. Вона дозволяє гнучко налаштовувати всі аспекти майбутнього відео. Надання користувачеві такого широкого спектру контролю є важливим аспектом HCI, оскільки це дозволяє точніше керувати процесом та отримувати бажані результати від ГШІ.

Рисунок 3.8 – Панель налаштування генерації.

Введення сценарію відео. Багаторядкове текстове поле, де користувач може ввести власний детальний сценарій для відео. Це дає максимальний контроль над змістом.

Генерація сценарію за промптом. Текстове поле для введення короткого текстового запиту (промпту). Після введення промпту та натискання окремої кнопки «Згенерувати сценарій», система генерує розгорнутий сценарій, який потім можна редагувати у полі для сценарію. Це спрощує створення контенту для користувачів, які не мають готового сценарію або ідей для змісту відео.

Вибір мови озвучення. Випадаючий список з широким переліком доступних мов (наприклад, українська, англійська, німецька тощо). Вибір мови впливає на доступні голоси та генерацію аудіо доріжки.

Вибір голосу. Динамічний випадаючий список, вміст якого оновлюється залежно від обраної мови. Для кожної мови пропонується множина варіантів чоловічих та жіночих голосів, що дозволяє персоналізувати озвучення.

Розмір субтитрів. Випадаючий список для вибору розміру субтитрів («Малий», «Середній», «Великий»).

Вибір якості відео. Радіокнопки для вибору бажаної якості відео («SD 480p», «HD 720p», «Full HD 1080p»). Вибір якості може впливати на час генерації та розмір файлу.

Кнопка «Генерувати». Велика, помітна кнопка, яка ініціює процес генерації відео на сервері на основі всіх обраних налаштувань. Після натискання кнопки система надає зворотний зв’язок про початок процесу.

Така деталізована панель налаштувань, з одного боку, надає користувачеві значну гнучкість, а з іншого – вимагає продуманого дизайну, щоб не перевантажити його опціями. Групування елементів, чіткі підписи та візуальні підказки (як попередній перегляд орієнтації) сприяють кращому UX.

Сторінка профілю користувача та управління відеоматеріалами

Сторінка профілю користувача («My Videos») надає зареєстрованому користувачеві доступ до всіх раніше згенерованих ним відеоматеріалів та інструменти для їх перегляду та базового менеджменту. Це важлива функція для організації роботи та повторного використання контенту.

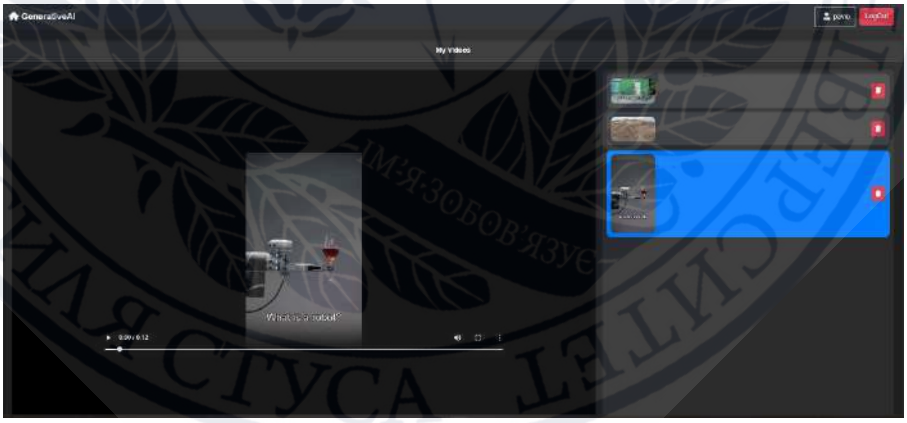

Рисунок 3.9 – Повна сторінка профілю користувача.

Карусель-галерея збережених відео (права частина). Відображає список мініатюр (прев’ю) всіх відео (рис. 3.9), згенерованих та збережених користувачем. Кожен елемент списку може містити: дату створення, тривалість.

Натискання на мініатюру відео активує його для детального перегляду у лівій частині.

Область перегляду збережених відео (ліва частина). При виборі відео з правої області, у лівій частині відображається більший плеєр для цього відео. Тут також присутні усі елементи керування плеєром: павза, хронометраж, керування гучністю, керування збільшенням відео та інші налаштування.

Функція видалення відео. Поруч з кожним відео у списку (права область рис. 3.9) присутня кнопка або також іконка («Кошик») для видалення відео. При натисканні система може запитувати підтвердження видалення, щоб уникнути випадкових дій. Після підтвердження відео видаляється із бази даних на сервері та зі списку в профілі користувача.

Надання користувачам можливості керувати власним контентом є стандартною практикою для багатьох веб-сервісів і значно підвищує зручність використання системи у довгостроковій перспективі.

3.2 Загальна архітектура застосунку

Для реалізації клієнтської частини веб-інтерфейсу було обрано фреймворк Angular з огляду на його потужні можливості для побудови складних односторінкових застосунків (SPA), компонентний підхід та розвинену екосистему. Клієнтська частина системи розроблена з дотриманням принципів сервісно-орієнтованої архітектури на рівні клієнтських сервісів, модульності для забезпечення гнучкості та реактивного програмування з використанням бібліотеки RxJS для ефективного управління асинхронними потоками даних, що є критичним для взаємодії з процесами генерації ШІ.[21,22]

Таблиця 3.1 – Загальна архітектура застосунку

Рівень	Опис	Основні файли
Презентаційний	Відповідає за відображення інтерфейсу користувача, взаємодію з користувачем	src/app/*.component.*, src/app/dashboard
Логічний (сервіси)	Містить бізнес-логіку, роботу з API, управління станом	src/app/_services/, src/app/_interceptors/
Моделі	Містить моделі даних, які використовуються в додатку для представлення структурованої інформації. Ці моделі слугують інтерфейсами та класами для взаємодії з API.	src/app/_models/
Навігація	Реалізує маршрутизацію між сторінками/компонентами	src/app/app.routes.ts, src/app/nav/
Захист	Реалізує захист маршрутів	src/app/_guards/
Конфігурація	Налаштування застосунку, середовища, стилі, залежності	angular.json, tsconfig*.json, environments/

Таблиця 3.2 – Переваги архітектури

Переваги	Причини
Модульність	Кожен функціональний блок ізольований у власній папці/компоненті, що спрощує підтримку
Масштабованість	Додаються нові компоненти/сервіси без зміни існуючої логіки
Повторне використання коду	Сервіси, моделі та інтерфейси можна використовувати у різних частинах застосунку
Чітке розділення відповідальностей	Презентаційний шар не містить бізнес-логіки, що підвищує якість коду
Тестованість	Окремі компоненти та сервіси легко покривати юніт-тестами
Підтримка сучасних стандартів	Використання Angular CLI, TypeScript, SCSS, Bootstrap

Таблиця 3.3 – Недоліки архітектури

Недоліки	**Причини**
Високий поріг входу	Необхідні знання Angular, TypeScript, RxJS, шаблонів
Надмірна структурованість для малих проєктів	Для невеликих застосунків структура може бути надмірно складною
Залежність від Angular	Міграція на інші фреймворки потребує значних зусиль
Складність налаштування	Велика кількість конфігураційних файлів та залежностей
Важкість дебагу асинхронних процесів	RxJS та інтерцептори можуть ускладнювати відстеження помилок

Важливо зазначити, що усвідомлення і врахування потенційних недоліків обраної архітектури клієнтської частини є невід'ємним етапом наукового обґрунтованого проєктування. Для кожного з виявлених недоліків (Таблиця 3) було передбачено стратегії їх врахування або пом'якшення, а також визначено причини, з яких певні компроміси є допустимими в контексті розробки даної системи.

Зокрема, такий недолік, як високий поріг входу, зумовлений необхідністю володіння технологіями Angular, TypeScript та концепціями реактивного програмування (RxJS), був врахований при реалістичному плануванні термінів виконання проєкту. Переваги, що надаються цими технологіями у створенні складних інтерактивних інтерфейсів, були визнані пріоритетними для досягнення поставлених цілей щодо якості та функціональності користувацького досвіду.

Щодо надмірної структурованості для малих проєктів, слід підкреслити, що розробка комплексного веб-інтерфейсу для системи взаємодії з генеративним штучним інтелектом апріорі не може являтись малим проєктом. Система передбачає значну кількість функціональних модулів, таких як автентифікація та авторизація користувачів, управління запитами на генерацію відео,

конфігурування параметрів моделей ГШІ, візуалізація проміжних та фінальних результатів, а також управління профілем користувача та історією генерацій. Більше того, архітектура закладає потенціал для подальшого розширення функціоналу, наприклад, інтеграції з новими моделями ГШІ або розширення можливостей постобробки контенту. За таких умов обрана структурованість є цілком виправданою, оскільки вона сприяє довгостроковій підтримці, модифікованості та масштабованості програмного продукту.

Проблема залежності від фреймворку Angular, що потенційно ускладнює міграцію на інші технології, є усвідомленим архітектурним компромісом. Вибір Angular був зроблений на основі його поточної відповідності вимогам проєкту, широкої підтримки спільнотою та наявності інструментів, що прискорюють розробку. Очікуваний життєвий цикл системи та швидкість еволюції фронтенд-технологій роблять цей ризик прийнятним.

Складність налаштування, пов’язана з великою кількістю конфігураційних файлів та залежностей, є характерною для більшості сучасних комплексних фреймворків. Цей аспект компенсується використанням Angular CLI, що автоматизує значну частину рутинних операцій з конфігурації та збірки проєкту, а також детальним документуванням процесу налаштування середовища розробки.

Нарешті, важкість відлагодження асинхронних процесів, особливо при використанні RxJS та HTTP інтерцепторів для взаємодії з API, була адресована через застосування спеціалізованих інструментів для профілювання та відлагодження (наприклад, Angular DevTools, засоби браузера для аналізу мережевих запитів), а також через ретельне проєктування потоків даних та обробки станів при взаємодії з асинхронними операціями генерації контенту на боці ГШІ. Чітке логування та обробка помилок на різних етапах асинхронної взаємодії також сприяли мінімізації цієї проблеми.

Таким чином, критичний аналіз потенційних недоліків та проактивне планування шляхів їх нівелювання дозволили сформувати збалансоване

архітектурне рішення для клієнтської частини, що відповідає вимогам проєкту та забезпечує необхідний рівень надійності та функціональності.

3.3 Проєктування структури бази даних

Для забезпечення функціонування веб-інтерфейсу та збереження даних, пов'язаних з роботою користувачів та процесами генерації відеоконтенту, була розроблена схема бази даних. Обрано реляційну модель даних, що дозволяє структурувати інформацію та забезпечити зв'язки між ключовими сутностями системи. Схема включає таблиці для зберігання інформації про користувачів, їхні активні сесії, історію запитів на генерацію, а також дані про згенеровані відеофайли.[23]

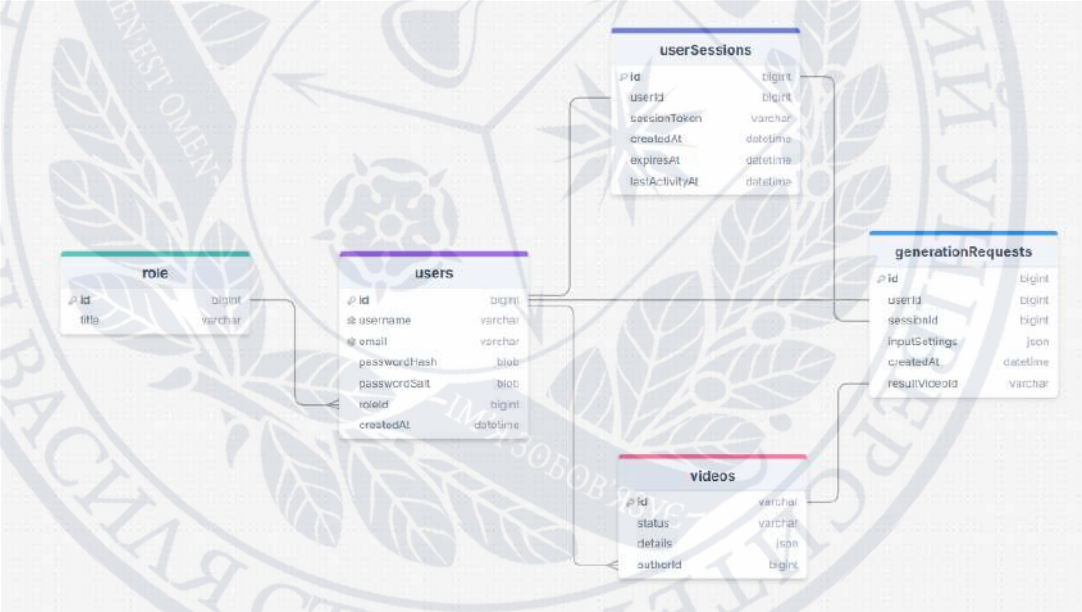

Рисунок 3.10 – Схема бази даних

Таблиця *users*. Призначена для зберігання реєстраційних даних користувачів системи.

Таблиця 3.4 – Опис таблиці users

id	Унікальний ідентифікатор користувача.
username	Логін користувача.
email	Адреса електронної пошти користувача.
passwordHash	Хеш пароля користувача для безпечного зберігання.
passwordSalt	Сіль, що використовується при хешуванні пароля.
roleId(Foreign Key).	Ідентифікатор ролі користувача в системі (має зв’язок із role.id).
createdAt	Час створення облікового запису.

Таблиця *userSessions*. Використовується для управління активними сесіями користувачів.

Таблиця 3.5 – Опис таблиці userSessions

id	Унікальний ідентифікатор сесії.
userId(Foreign Key).	Ідентифікатор користувача, якому належить сесія (має зв’язок із users.id).
sessionToken	Унікальний токен, що ідентифікує сесію.
createdAt	Час початку сесії
expiresAt	Час закінчення терміну дії сесії.
lastActivityAt	Час останньої активності користувача в рамках сесії.

Таблиця generationRequests. Слугує для фіксації історії запитів на генерацію відео.

Таблиця 3.6 – Опис таблиці generationRequests

id	Унікальний ідентифікатор запиту
userId (Foreign Key)	Ідентифікатор користувача, який створив запит (має зв’язок із users.id).

sessionId (Foreign Key):	Ідентифікатор сесії, з якої було зроблено запит (має зв’язок із userSessions.id).
inputSettings	Вхідні параметри та налаштування, вказані користувачем для генерації відео.
createdAt	Час створення запиту.
resultVideoId (Foreign Key):	Ідентифікатор згенерованого відеофайлу (посилається на videos.id), заповнюється після успішної генерації.

Таблиця videos. Зберігає інформацію про згенеровані відеофайли.

Таблиця 3.7 – Опис таблиці videos

id	Унікальний ідентифікатор відео.
status	Поточний статус відео ('обробка', 'готове', 'помилка').
details	Детальна інформація про відеофайл, включаючи шлях до сховища, метадані (розмір, тривалість тощо).
authorId	Ідентифікатор користувача-автора відео (посилається на users.id).

Таблиця *roles*. Використовується для управління ролями наданими користувачу.

Таблиця 3.8 – Опис таблиці roles

id	Унікальний ідентифікатор сесії.
title	Назва наданої ролі користувачу.

Розроблена в ході дослідження структура бази даних включає основні сутності, необхідні для функціонування веб-інтерфейсу системи роботи з генеративним штучним інтелектом. Таблиці users, userSessions, generationRequests, videos та roles дозволяють зберігати інформацію про

користувачів, управляти їхніми сесіями, фіксувати історію запитів на генерацію контенту, управляти ролями користувача(admin, user, moderator) та акумулювати метадані згенерованих відеофайлів.

Зокрема, таблиця users забезпечує зберігання облікових даних та інформації для автентифікації і авторизації. Таблиця userSessions призначена для відстеження активних сеансів роботи користувачів, що є важливим для безпеки та аналізу активності. Для ведення історії взаємодії користувача із системою та параметрів запитів на генерацію відео передбачена таблиця generationRequests. Результати генерації, тобто інформація про створені відео, зберігаються в таблиці videos.

Встановлені зв’язки між таблицями, реалізовані за допомогою зовнішніх ключів, забезпечують логічну узгодженість та цілісність даних. Використання типу даних JSON для окремих атрибутів, таких як inputSettings в generationRequests та details в videos, надає необхідну гнучкість для збереження структурованих, але потенційно змінних наборів параметрів та метаданих.

3.4 Архітектура та реалізація API

Для забезпечення взаємодії між розробленим веб-інтерфейсом та логікою системи генерації відео, а також для управління даними, було спроектовано та реалізовано серверну частину у вигляді Application Programming Interface (API). В якості основного фреймворку для побудови API було обрано FastAPI, що базується на мові програмування Python.[24,25]

Вибір FastAPI зумовлений його ключовими перевагами для розробки сучасних веб-сервісів:

- Висока продуктивність FastAPI зумовлена тим, що він побудований на основі Starlette (для веб-частини) та Pydantic (для валідації даних), що забезпечує асинхронну обробку запитів та швидкість, порівнянну з іншими технологіями.

- Використання Pydantic-моделей дозволяє автоматично валідувати вхідні та вихідні дані, зменшуючи кількість помилок та спрощуючи розробку.
- Автоматична генерація документації у форматах OpenAPI (раніше Swagger UI) та ReDoc, значно полегшує тестування та інтеграцію API.
- Підтримка async/await синтаксису що забезпечує асинхронність і дозволяє ефективно обробляти довготривалі операції, такі як генерація відео, без блокування основного потоку.
- Вбудована система ін'єкції залежностей (Dependency Injection) спрощує управління ресурсами, наприклад такими як сесії бази даних.

Основні компоненти архітектури API:

- Моделі даних (SQLAlchemy). Класи User та Video визначають структуру таблиць у базі даних та зв'язки між ними. І також сутності UserSession та GenerationRequest для управління сессіями користувачів і відслідковування повноцінного функціонування системи.[26]
- Схеми даних (Pydantic). Моделі Pydantic, такі як VideoGenerationSettings, використовуються для валідації даних, що надходять від клієнта, та для визначення структури відповіді API.[27]
- Маршрутизатори та обробники запитів (Endpoints). Кожен ендпоінт API (наприклад, /generate_video/{author_id}, /video_status/{video_id}) пов'язаний з асинхронною функцією-обробником, яка виконує відповідну бізнес-логіку на стороні серверної частини.
- Фонові задачі (Background Tasks). Для довготривалих операцій, таких як безпосередньо процес генерації відео, використовується механізм фонових задач FastAPI (BackgroundTasks). Це дозволяє API негайно повернути відповідь клієнту про прийняття запиту, в той час як сама генерація виконується асинхронно у фоні.
- Обробка CORS (Cross-Origin Resource Sharing). Для забезпечення можливості взаємодії з клієнтською частиною, що розміщена на іншому домені або порті, налаштовано CORSMiddleware.

3.5 UML-діаграма класів клієнтської частини та її опис

В додатку А представлено повну UML-діаграму класів, що ілюструє ключові компоненти клієнтської частини веб-інтерфейсу та основні взаємозв'язки між ними. Дана діаграма акцентує увагу на архітектурних рішеннях, прийнятих при розробці користувацького інтерфейсу на базі фреймворку Angular.[28,29]

Представлена UML-діаграма класів відображає високорівневу архітектуру клієнтської частини системи, розробленої на фреймворку Angular. Вона ілюструє декомпозицію користувацького інтерфейсу на ключові компоненти та сервіси, а також їх взаємодію через чітко визначені інтерфейси моделей даних.

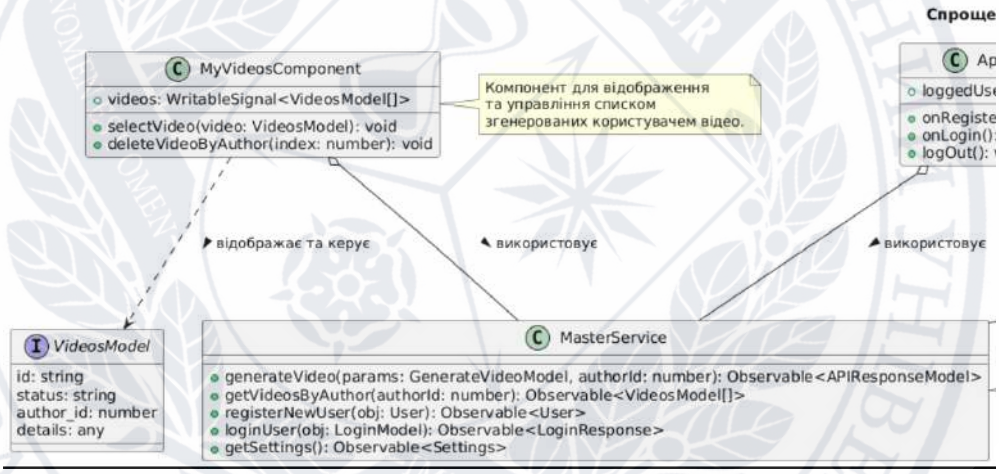

Рисунок 3.11 – Частина UML діаграми

Центральним елементом архітектури є сервіс MasterService. Цей клас виконує роль посередника між компонентами користувацького інтерфейсу та серверним API. MasterService інкапсулює логіку формування та надсилання HTTP-запитів, відповідаючи за отримання даних (наприклад, налаштувань системи через getSettings(), списку відео користувача через getVideosByAuthor()), ініціювання команд на генерацію відео (generateVideo()) та управління процесами автентифікації й авторизації користувачів (registerNewUser(), loginUser()). Застосування цього сервісного шару дозволяє відокремити логіку

взаємодії з бекендом від логіки відображення, що сприяє дотриманню принципу єдиної відповідальності (Single Responsibility Principle) та підвищує тестування та підтримку коду.

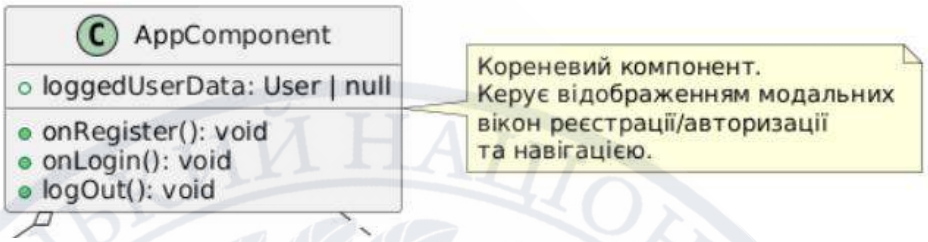

Рисунок 3.12 – Частина UML діаграми класу AppComponent

Клас AppComponent є кореневим компонентом Angular-застосунку. Він відповідає за загальну структуру та ініціалізацію веб-інтерфейсу. До його функцій належить управління станом автентифікації користувача, зокрема обробка логіки входу (onLogin()), реєстрації (onRegister()) та виходу з системи (logOut()), а також збереження даних поточного користувача (властивість loggedUserData). AppComponent також керує відображенням відповідних модальних вікон для цих операцій.

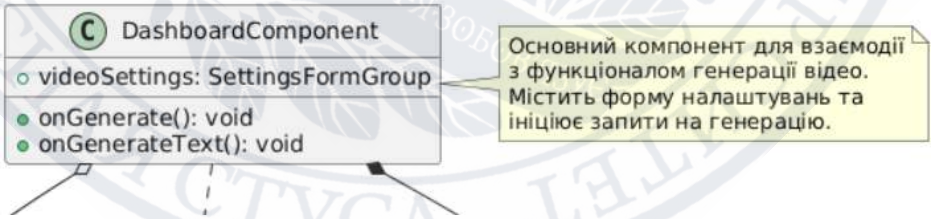

Рисунок 3.13 – Частина UML діаграми класу DashboardComponent

Компонент DashboardComponent реалізує основний функціонал користувацького інтерфейсу, пов'язаний із процесом генерації відео. Він агрегує екземпляр SettingsFormGroup (властивість videoSettings) для збору та валідації параметрів генерації, введених користувачем. Методи onGenerate() та

onGenerateText() цього компонента ініціюють запити до MasterService для запуску процесу створення відео або генерації текстового сценарію відповідно.

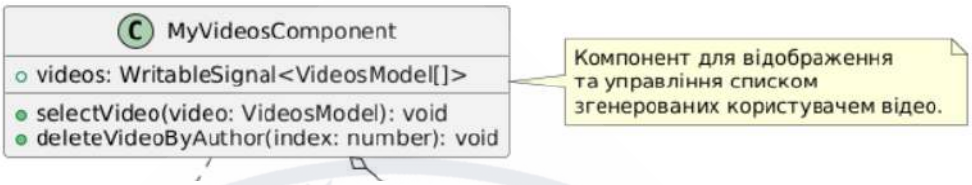

Рисунок 3.14 – Частина UML діаграми класу MyVideosComponent.

Клас MyVideosComponent призначений для відображення та управління списком відео, які були згенеровані поточним користувачем. Він отримує масив моделей VideosModel (властивість videos) через MasterService і надає користувачеві можливості для взаємодії з цими відео, такі як вибір для перегляду (selectVideo()) та видалення зі списку (deleteVideoByAuthor()).

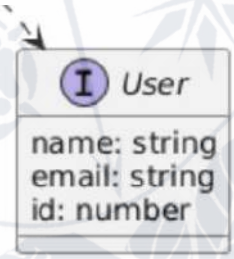

Рисунок 3.15 – Частина UML діаграми інтерфейсу User

Інтерфейс User визначає контракт даних для представлення користувача в системі. Він описує основні атрибути користувача, такі як ім’я (name), електронна пошта (email) та унікальний ідентифікатор (id), забезпечуючи стандартизоване представлення інформації про користувача в усьому клієнтському застосунку.

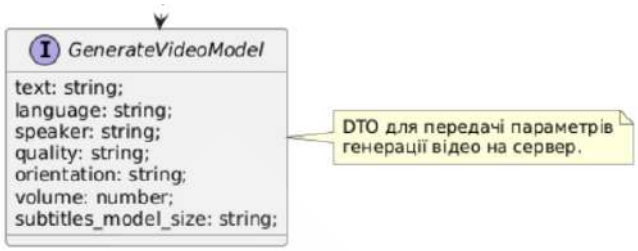

Рисунок 3.16 – Частина UML діаграми інтерфейсу GenerateVideoModel

Інтерфейс GenerateVideoModel слугує як об’єкт передачі даних (Data Transfer Object, DTO), що структурує параметри, необхідні для запиту на генерацію відео. Він включає поля для текстового сценарію або промпту (text), мови озвучування (language), обраного голосу (speaker) та інших специфічних налаштувань, що передаються до MasterService.[30,31]

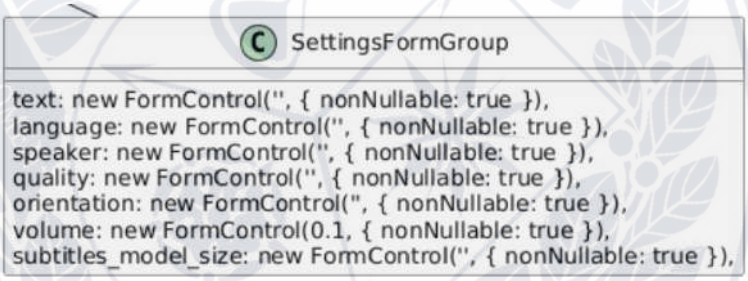

Рисунок 3.17 – Частина UML діаграми компоненту SettingsFormGroup

Компонент SettingsFormGroup представляє структуру реактивної форми Angular, що використовується в DashboardComponent для збору параметрів генерації відео. Ця форма інкапсулює набір полів введення (controls), валідаторів та логіки відстеження змін. Структура даних, яку агрегує ця форма, відповідає інтерфейсу GenerateVideoModel, забезпечуючи типізовану передачу параметрів.

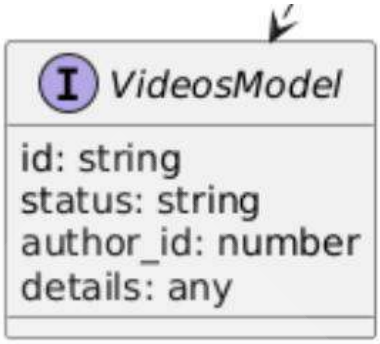

Рисунок 3.18 – Частина UML діаграми інтерфейсу VideosModel

Інтерфейс VideosModel описує структуру даних для окремого згенерованого відео. Він містить інформацію про унікальний ідентифікатор відео (id), його поточний статус генерації (status), ідентифікатор автора (author_id) та додаткові деталі (details), необхідні для відображення та управління відеозаписами.

3.6 UML-діаграма API сутностей та їх опис

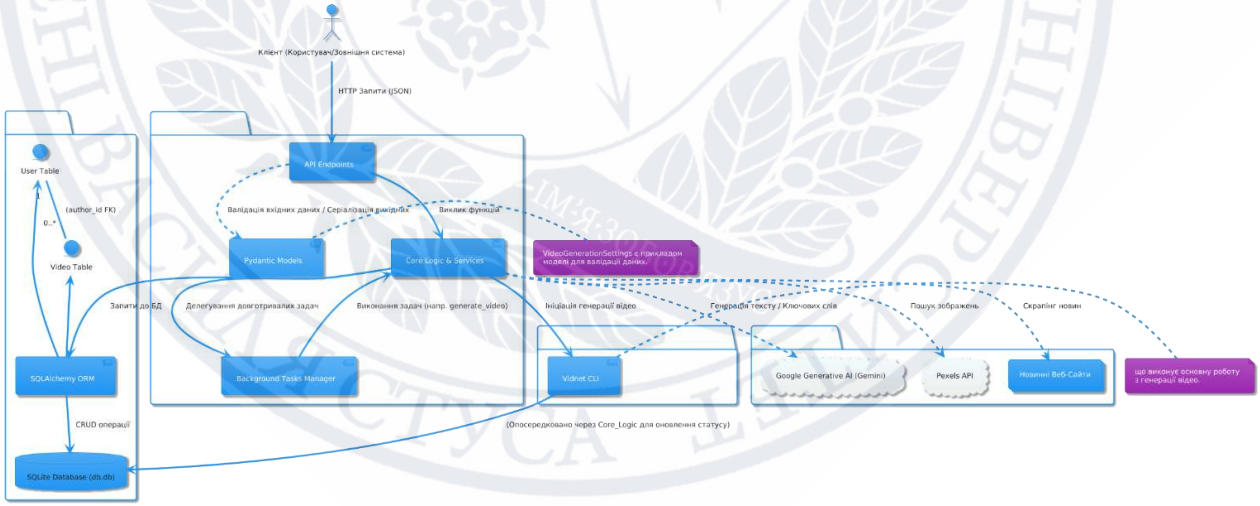

Рисунок 3.19 – Повне представлення UML діаграми API частини

Рисунок 3.19 показує, як влаштована розроблена система API для автоматичного створення відео. Ця діаграма дає змогу зрозуміти її основні сутності та їхню взаємодію. В основі архітектури лежить поділ на кілька ключових блоків, кожен з яких виконує свої функції. До них належать: клієнтська

частина, центральний серверний додаток, система зберігання даних та інтегровані зовнішні сервіси.

Серверний API, побудований на основі FastAPI, містить модулі для таких завдань: обробка HTTP запитів через, перевірка та підготовка даних за допомогою Pydantic моделей, виконання головних операцій системи (блок «Core Logic & Services») та запуск тривалих процесів у фоновому режимі, наприклад, генерації відео (через «Background Tasks Manager»). Дані зберігаються у базі даних SQLite, а робота з нею відбувається за допомогою SQLAlchemy ORM.

Для реалізації специфічних функцій, як-от створення сценарію для відео (за допомогою Google Generative AI), пошуку медіаконтенту (Pexels API), збору новин або безпосередньої збірки відео (модуль «Vidnet CLI»), система використовує підключення до відповідних зовнішніх платформ та бібліотек. [31] Розглянемо кожен блок діаграми та як вони взаємодіють між собою.

Client представляє собою зовнішню сутність, яка взаємодіє з системою через її API. Це може бути веб-інтерфейс користувача, мобільний застосунок або інша автоматизована система, що потребує функціоналу, наданого API, що забезпечує легке перенесення застосунку на будь-які платформи.

У розподілених архітектурах, зокрема мікросервісних або сервіс-орієнтованих, клієнт є ініціатором потоків даних та команд. Взаємодія зазвичай відбувається за стандартизованими протоколами, таким як HTTP/HTTPS, із використанням форматів обміну даними, наприклад JSON, що забезпечує інтероперабельність.

FastAPI Application. Цей блок є ядром системи та інкапсулює головну бізнес-логіку та інтерфейси взаємодії.

API Endpoints. Цей компонент, реалізований на основі фреймворку FastAPI, відповідає за прийом HTTP-запитів від клієнтської частини застосунку, їх маршрутизацію до відповідних обробників або ж по іншому контролерів які взаємодіють із основною бізнес логікою сервера, валідацію вхідних даних та

формування HTTP-відповідей. Він визначає контракт взаємодії системи із, наприклад, клієнтською частиною на основі Angular framework.

API endpoints є реалізацією інтерфейсу прикладного програмування. Використання FastAPI дозволяє автоматично генерувати інтерактивну документацію API таку як SwaggerUI що базується на типізованих даних (Pydantic моделі) та асинхронних операціях, оптимізуючи продуктивність та розробку. Асинхронні функції (async/await) є ключовими для обробки великої кількості одночасних запитів без блокування потоків виконання, що є критичним для I/O-bound операцій, таких як мережеві запити та взаємодія з файловою системою.

Pydantic Models використовуються для валідації, серіалізації та десеріалізації даних, що передаються через API. Вони забезпечують типізацію даних на рівні застосунку, підвищуючи надійність та полегшуючи дебагінг коду. VideoGenerationSettings є прикладом такої моделі.

Використанняи подібних моделей для валідації на межі системи є стандартною практикою щоб забезпечити цілісність даних та запобігання помилок обробки. Pydantic інтегрується з підказками типів Python, дозволяючи проводити валідацію на етапі виконання та автоматично генерувати JSON схеми для документації API.

Core Logic & Services містить основну бізнес-логіку застосунку. Він обробляє запити, отримані від API Endpoints, взаємодіє з базою даних через ORM, викликає зовнішні сервіси (Google GenAI, Pexels API, новинні сайти) та керує процесом генерації відео, зокрема через делегування завдань фоновим обробникам.

Даний метод дозволяє інкапсулювати бізнес-логіку в окремий шар, що сприяє підвищенню модульності, розширення та тестованості системи. Саме тут реалізуються алгоритми обробки даних, координація взаємодії між різними частинами системи та інтеграція з зовнішніми залежностями. Функції, такі як

generate_video, make_config, generate_text, search_news, get_news_text, generate_preview_image, є прикладами таких сервісів.

Background Tasks Manager від FastAPI, дозволяє виконувати довготривалі операції (наприклад, генерацію відео) асинхронно, не блокуючи основний потік обробки HTTP-запитів. Це покращує відгук системи для клієнта.

Асинхронне виконання завдань є критично важливим для масштабованих веб-застосунків. Передача ресурсозатратних операцій у фоновий режим дозволяє API негайно повернути відповідь клієнту, в той час як задача обробляється незалежно. Це запобігає затримки HTTP-з'єднань та покращує загальний досвід користувача.

SQLAlchemy ORM (Object-Relational Mapper) використовується для взаємодії з реляційною базою даних. Вона дозволяє працювати з даними на рівні об'єктів Python (User, Video, UserSessions, GenerationRequests, role), абстрагуючись від конкретних SQL-запитів, та забезпечує управління сесіями бази даних (get_db).

ORM системи спрощують розробку застосунків, що працюють з базами даних, надаючи об'єктно-орієнтований доступ до реляційних даних. Це підвищує продуктивність розробників та портативність коду між різними СУБД. SQLAlchemy є потужною бібліотекою, що підтримує як активний запис (Active Record), так і відображення даних (Data Mapper) патерни.

SQLite Database використовується як конкретна система управління базами даних (СУБД) для зберігання інформації про користувачів (users таблиця) та згенеровані відео (videos таблиця).

SQLite є простою, файловою СУБД, яка часто використовується для локальної та міжплатформової розробки, тестування та для застосунків з помірними вимогами до навантаження та конкурентного доступу.

External Services об'єднує зовнішні API та ресурси, які використовуються системою.

Google Generative AI (Gemini) використовується для генерації текстового контенту, а саме сценаріїв для відео на основі запитів (generate_text) та для вилучення ключових слів з тексту відео для пошуку зображень попереднього перегляду (generate_preview_image).

Інтеграція з великими мовними моделями (LLM) дозволяє збагатити функціонал системи можливостями обробки природної мови, генерації контенту та інтелектуального аналізу тексту.

Pexels API використовується для пошуку та завантаження стокових фотографій, які можуть бути використані як основа для генерації зображень попереднього перегляду відео.

Використання API для доступу до медіа-ресурсів дозволяє автоматизувати процес пошуку та інтеграції візуального контенту, розширюючи можливості системи без необхідності утримувати власну велику бібліотеку зображень.

News Websites. Система здійснює веб-скрапінг вказаних новинних веб-сайтів (search_news) для збору посилань на статті та вилучення текстового контенту з них (get_news_text), який потім може бути оброблений для генерації відео.

Веб-скрапінг є технікою автоматичного вилучення даних з веб-сторінок. Хоча він є потужним інструментом для збору інформації, його використання повинно враховувати етичні аспекти та умови користування відповідних ресурсів. Зібраний текст потім обробляється за допомогою NLP-технік через Google GenAI для резюмування або перефразування.

Vidnet CLI. Компонент vidnet є серверною частиною застосунку що є відповідальним за безпосередню генерацію відеофайлу на основі наданого тексту, налаштувань (мова, голос, якість, орієнтація), що приходять від клієнтської частини описаної в даній роботі та інших ресурсів (музика, стокові відео/фото).

Декомпозиція системи шляхом винесення спеціалізованих завдань, таких як комплексна обробка та генерація медіа, у зовнішні модулі або мікросервіси є

поширеним архітектурним підходом. Це дозволяє незалежно розробляти, масштабувати та оновлювати окремі компоненти. Vidnet інкапсулює складні процеси синтезу мови, накладання субтитрів, обробки зображень/відео та їхнього зведення в кінцевий відеофайл. Взаємодія з ним відбувається через чітко визначений інтерфейс.

Ці блоки разом формують комплексну систему для генерації відеоконтенту, керовану через API, з використанням сучасних технологій та підходів до розробки програмного забезпечення.

Висновок до розділу 3

У третьому розділі було детально представлено процес розробки та ключові аспекти програмної реалізації веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. Описано функціональні можливості розробленого інтерфейсу, що охоплюють повний цикл взаємодії користувача: від реєстрації та автентифікації до налаштування параметрів генерації відео на головній сторінці, перегляду результатів та управління згенерованим контентом у профілі користувача. Представлено загальну архітектуру клієнтського застосунку на базі Angular, проаналізовано її переваги та недоліки з обґрунтуванням шляхів їх усунення. Детально описано спроектовану структуру бази даних з описом усіх таблиць users, userSessions, generationRequests, videos та roles. Також присутні архітектурні рішення та основні компонентів серверного API на FastAPI, включаючи моделі SQLAlchemy, схеми Pydantic, роутінг, фонові задачі та обробку CORS. Наведено UML-діаграми класів клієнтської частини та сутностей API з їх детальним та покроковим описом.

ВИСНОВКИ

У ході виконання даної кваліфікаційної (бакалаврської) роботи було успішно вирішено актуальну науково-практичну задачу розробки архітектури та ключових компонентів веб-інтерфейсу системи для роботи з генеративним штучним інтелектом, орієнтованої на створення відеоконтенту. Початкова проблема полягала у значному розриві між стрімко зростаючими можливостями сучасних моделей ГШІ та зручністю їх практичного застосування широким колом користувачів, які не завжди володіють глибокими технічними знаннями. Метою дослідження було створення такого веб-інтерфейсу, який би забезпечував інтуїтивну взаємодію, ефективне управління користувацькими даними та процесом генерації, тим самим сприяючи легкого доступу до технологій ГШІ.

Для досягнення поставленої мети було вирішено низку ключових завдань. Проведено аналіз сучасного стану технологій генеративного ШІ для створення відео та існуючих підходів до розробки користувацьких інтерфейсів. На основі цього аналізу було обґрунтовано вибір архітектурних рішень та технологічного стеку: фреймворк Angular для розробки клієнтської частини та FastAPI для реалізації RESTful API для зв’язку із серверною частиною. Такий вибір забезпечив модульність, масштабованість та високу продуктивність системи.

Була спроєктована структура бази даних, що включає таблиці для зберігання інформації про користувачів (users), їх ролі (roles), активні сесії (userSessions), історію запитів на генерацію відео (generationRequests) та метадані згенерованого контенту (videos). Використання реляційної моделі даних із зовнішніми ключами забезпечило цілісність даних, а застосування типу JSON для зберігання налаштувань генерації та деталей відео – необхідну гнучкість для майбутніх розширень.

Розроблено ключові функціональні модулі веб-інтерфейсу, що охоплюють повний цикл взаємодії користувача з системою: реєстрацію та автентифікацію на основі JWT; конфігурування широкого спектру параметрів генерації відео

(включаючи введення сценарію, генерацію за промптом, пошук новин, вибір мови, голосу, орієнтації, якості); ініціацію процесу генерації; оперативний перегляд результатів на головній сторінці; та управління згенерованим контентом у персональному профілі користувача з можливістю видалення відео. Для забезпечення взаємодії між клієнтською та серверною частинами реалізовано RESTful API, що використовує асинхронні запити та фонові задачі для довготривалих операцій генерації.

Практична значущість розробленої системи полягає у створенні прототипу, який демонструє можливість ефективного поєднання потужних генеративних технологій ШІ зі зручним та інтуїтивно зрозумілим користувацьким інтерфейсом. Застосування принципів HCI, таких як надання користувачеві контролю, забезпечення чіткого зворотного зв’язку та можливості попереднього перегляду (наприклад, орієнтації відео), сприяє зниженню порогу входження та підвищенню ефективності роботи з системою. Порівняно з існуючими аналогами, які часто або є надто складними для пересічного користувача (як локальні UI для Stable Diffusion), або пропонують обмежену гнучкість та контроль (як деякі SaaS-платформи), розроблене рішення прагне знайти баланс, надаючи як простоту використання, так і широкі можливості кастомізації.

Результати даної роботи підтверджують досягнення поставленої мети та вирішення всіх визначених завдань. Створений прототип веб-інтерфейсу та спроєктовані компоненти системи є міцною основою для подальшого розвитку та вдосконалення інтегрованого середовища для роботи з генеративним ШІ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Amershi S., Weld D., Vorvoreanu M., et al. Guidelines for human-AI interaction. *Proceedings of the 2019 CHI conference on human factors in computing systems*. 2019. P. 1-13.
2. ResearchGate. (PDF) Human-AI Interaction Design Standards. URL: https://www.researchgate.net/publication/390115046_Human-AI_Interaction_Design_Standards
3. Goodfellow I. J., Pouget-Abadie J., Mirza M., et al. Generative adversarial nets. *Advances in neural information processing systems*. 2014. Vol. 27.
4. Karras T., Aittala M., Laine S., et al. Analyzing and improving the image quality of StyleGAN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. P. 8110-8119.
5. Stable Diffusion WebUI Goes Fully Serverless: A Xinference-Driven Multi-Tenant Solution. Medium. URL: https://medium.com/%40BrillAI/stable-diffusion-webui-goes-fully-serverless-a-xinference-driven-multi-tenant-solution-8a142a9dbffc
6. Azim, S. (n.d.). A Survey of Autoregressive Models for Image and Video Generation. URL: https://saqib1707.github.io/assets/pubs/autoregressive_generation_survey.pdf
7. Pictory AI. URL:https://pictory.ai/?el=2000b&htrafficsource=pictoryblog
8. Synthesia.URL: https://www.synthesia.io/
9. Automatic1111.URL: https://stable-diffusion-art.com/automatic1111/
10. Fliki AI. URL: https://fliki.ai/
11. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: elements of reusable object-oriented software. Addison-Wesley.
12. Офіційна документація Angular. URL: https://angular.io/docs
13. TypeScript Handbook. URL: https://www.typescriptlang.org/docs/handbook/intro.html

14. RxJS Overview. URL: https://rxjs.dev/guide/overview

15. Bootstrap Documentation. URL: https://getbootstrap.com/docs/

16. FastAPI Documentation. URL: https://fastapi.tiangolo.com/

17. JWT.IO. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction

18. Webandcrafts. FastAPI for Scalable Microservices: Best Practices & Optimisation. URL: https://webandcrafts.com/blog/fastapi-scalable-microservices

19. RESTful APIs: Principles and Best Practices. URL: https://api7.ai/learning-center/api-101/restful-api-best-practices

20. Official documentation https://git-scm.com/doc

21. Distributed Systems: Principles and Paradigms

22. Dev Centre House. Building Scalable Angular Applications: Best Practices for Large-Scale Projects. URL: https://www.devcentrehouse.eu/blogs/angular-applications-for-projects/

23. Основи розробки баз даних.Url: https://support.microsoft.com/uk-ua/topic/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8-%D0%B1%D0%B0%D0%B7-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5

24. DEV Community. Microservice in Python using FastAPI. URL: https://dev.to/paurakhsharma/microservice-in-python-using-fastapi-24cc

25. Webandcrafts. FastAPI for Scalable Microservices: Best Practices & Optimisation. URL: https://webandcrafts.com/blog/fastapi-scalable-microservices

26. SQLAlchemy Documentation. URL: https://www.sqlalchemy.org/

27. Pydantic Documentation. URL: https://pydantic-docs.helpmanual.io/

28. Visual Paradigm. What is Unified Modeling Language (UML)? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

29. Syncfusion Blogs. Easily Create UML Activity Diagrams in Angular. URL: https://www.syncfusion.com/blogs/post/create-angular-uml-activity-diagram/amp

30. What is a Data Transfer Object (DTO)? URL: https://www.baeldung.com/java-data-transfer-object

31. Data Transfer Objects: A Guide to Best Practices. URL: https://developer.okta.com/blog/2021/08/25/data-transfer-objects

32. Deng, J., & Sætra, H. S. (2024). Understanding the Ethics of Generative AI: Established and New Ethical Principles. AI and Ethics. URL: https://www.researchgate.net/publication/387854870_Understanding_the_Ethics_of_Generative_AI_Established_and_New_Ethical_Principles

33. Harvard Data Science Review. A Human-Centered Perspective on AI Transparency: Lessons Learned and Future Directions for LLMs. URL: https://hdsr.mitpress.mit.edu/pub/aelql9qy

34. Search Engine Land. AI-powered content management: How to make your workflows more efficient. URL: https://searchengineland.com/ai-powered-content-management-workflows-438271

35. EECS at UC Berkeley. Enhancing User Interface Design Tools with AI-Driven Evaluation. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/31646.html

36. Римар П.В., Гуменчук П.С. Розробка веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. *Наука і техніка сьогодні (Серія «Педагогіка», Серія «Право», Серія «Економіка», Серія «Фізико-математичні науки», Серія «Техніка»)*. 2025.

37. Гуменчук П.С., Римар П.В. Розробка веб-інтерфейсу системи для роботи з генеративним штучним інтелектом. *Прикладні інформаційні технології 2025*: Матеріали всеукр. науково-практ. конф. здобувачів вищ. освіти та молодих вчен., м. Вінниця, 22 трав. 2025р. 2025.

ДОДАТОК А
UML-ДІАГРАМА КЛАСІВ

На рисунку А.1 наведено повну UML-діаграму класів розробленої системи.

Рисунок А.1 – UML-діаграма класів