МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КОЛІБАБЧУК ДМИТРО ІГОРОВИЧ

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

РОЗРОБКА ПРОГРАМИ ДЛЯ СТВОРЕННЯ ВІДЕО З ВИКОРИСТАННЯМ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

Спеціальність 122 Комп’ютерні науки
Кваліфікаційна (бакалаврська) робота

Керівник:
Павло РИМАР, старший викладач
кафедри інформаційних технологій
_____

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

Вінниця – 2025

АНОТАЦІЯ

Колібабчук Д. І. Розробка програми для створення відео з використанням генеративного штучного інтелекту. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2025.

У кваліфікаційній роботі досліджено підходи до автоматизації створення відео із застосуванням генеративного ШІ. Розроблено консольну програму на Python для генерації відеоконтенту на основі тексту з використанням Pexels і Pixabay API, OpenAI Whisper, Piper TTS, Coqui XTTS, MoviePy, FFmpeg, Claude та Meta Llama. Програма автоматично створює відео з озвученням і візуальним супроводом відповідно до тексту.

Ключові слова: відеогенерація, штучний інтелект, Python, Whisper, Coqui XTTS, MoviePy, генеративні моделі.

78 стор., 29 рис., 8 табл., 37 джерел.

ABSTRACT

Kolibabchuk D. I. Development of a Video Creation Program Using Generative Artificial Intelligence. Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2025.

The qualification (bachelor's) thesis explores approaches to automating video creation using generative artificial intelligence. A command-line Python program has been developed for generating video content based on input text, utilizing Pexels and Pixabay APIs, OpenAI Whisper, Piper TTS, Coqui XTTS, MoviePy, FFmpeg, Claude, and Meta Llama. The program automatically creates video clips with AI-generated voiceovers and visual materials matching the text context.

Keywords: video generation, artificial intelligence, Python, Whisper, Coqui XTTS, MoviePy, generative models.

78 pages, 29 figures, 8 tables, 37 sources.

ЗМІСТ

ВСТУП

Актуальність дослідження. Сучасний ринок цифрового контенту стрімко розвивається і автоматизація створення відео набуває дедалі більшого значення. Популяризація платформ, таких як YouTube, TikTok та Instagram, створює високий попит на швидке та якісне виробництво відеоконтенту. Однак традиційний процес створення відео є довготривалим і вимагає значних людських ресурсів, особливо якщо потрібно працювати з озвученням, монтажем та підбором візуального матеріалу.

Генеративний штучний інтелект відкриває нові можливості для автоматизації цього процесу. В останні роки активно розвиваються технології генерації тексту, синтезу мовлення та автоматичного монтажу відео, проте їх інтеграція у єдину систему залишається недостатньо дослідженою. Більшість існуючих рішень або є надто складними для користувачів без технічної підготовки, або не дозволяють створювати повністю готове відео в автоматичному режимі.

Запропонована в роботі консольна програма, написана на Python надає можливість автоматизувати процес створення відео, об’єднуючи різні AI-моделі та API-сервіси в єдиний конвеєр. Використання таких технологій, як OpenAI Whisper для транскрипції, Piper TTS і Coqui XTTS для генерації голосу, а також MoviePy і FFmpeg для відеомонтажу, дозволяє отримати повністю готове відео з мінімальним втручанням користувача.

Актуальність дослідження полягає у вирішенні кількох важливих проблем. По-перше, існує обмежена кількість простих у використанні інструментів для автоматизації створення відео з використанням генеративного штучного інтелекту. По-друге, процес пошуку відповідного стокового контенту та його адаптації до тексту займає значний час, що знижує ефективність виробництва контенту. По-третє, існуючі рішення для генерації голосового озвучення часто мають обмежену функціональність або низьку якість синтезованого мовлення.

Крім того, наразі відсутні єдині інтегровані системи, які б поєднували всі необхідні компоненти у зручний автоматизований процес.

Таким чином, розробка запропонованої системи сприяє спрощенню та прискоренню створення відеоконтенту, відкриває нові можливості для контент-мейкерів, а також підсилює роль штучного інтелекту у креативних індустріях.

Мета. Мета даної роботи полягає у розробці програмного засобу для автоматизації процесу створення відео з використанням генеративного штучного інтелекту. Програма повинна забезпечувати повний цикл обробки тексту, підбору відповідного стокового відео та музики, генерації голосового озвучення та фінального монтажу, що дозволить мінімізувати необхідність ручного втручання у процес виробництва відеоконтенту.

Досягнення цієї мети передбачає інтеграцію сучасних технологій машинного навчання та мультимедійної обробки, зокрема використання OpenAI Whisper для транскрипції, Piper TTS і Coqui XTTS для генерації голосу, а також API Pexels та Pixabay для автоматичного підбору відео та музики. Реалізація проєкту у форматі консольної програми на Python сприятиме її гнучкості, масштабованості та можливості інтеграції у різні робочі процеси.

Результатом роботи має стати ефективне рішення, яке дозволить користувачам швидко та з мінімальними зусиллями створювати повноцінні відеоролики, що можуть використовуватися у сфері контент-мейкінгу, навчання та автоматизованого створення інформаційного відеоматеріалу.

Завдання дослідження. Спочатку потрібно виявити основні проблеми, пов’язані з автоматизацією створення відеоконтенту, та проаналізувати існуючі рішення, що використовують генеративний штучний інтелект для генерації тексту, озвучення та відеомонтажу.

Дослідити можливості застосування сучасних AI-моделей для автоматичного підбору стокового відео та музики відповідно до змісту тексту, а також оцінити їхню ефективність у контексті автоматизованого створення відео.

Розробити програму у вигляді консольного додатка на Python, яка інтегрує API Pexels та Pixabay для отримання мультимедійного контенту, OpenAI Whisper

для транскрипції, Piper TTS і Coqui XTTS для генерації голосу, а також MoviePy та FFmpeg для автоматичного відеомонтажу.

Здійснити тестування та аналіз продуктивності розробленої системи, визначити її переваги та обмеження, а також можливі напрями подальшого вдосконалення.

Оцінити практичну значущість створеного програмного рішення та його потенційне застосування у сфері контент-мейкінгу, навчальних матеріалів та автоматизованого відеопродакшину.

Об’єктом дослідження є процес автоматизації створення відеоконтенту за допомогою генеративного штучного інтелекту. Це включає використання сучасних технологій машинного навчання для генерації тексту, синтезу мовлення, автоматичного підбору мультимедійного контенту та відеомонтажу.

Предметом дослідження є методи та технології автоматизації процесу створення відеоконтенту шляхом інтеграції генеративного штучного інтелекту та мультимедійних API. Це включає алгоритми аналізу тексту, автоматичний підбір стокового відео та музики, синтез мовлення, а також методи монтажу відео на основі заданого сценарію.

Особлива увага приділяється взаємозв’язкам між різними компонентами автоматизованої системи, зокрема інтеграції OpenAI Whisper для транскрипції, Piper TTS та Coqui XTTS для генерації голосу, API Pexels і Pixabay для пошуку мультимедійного контенту, а також MoviePy та FFmpeg для відеомонтажу. Дослідження цих взаємозв’язків дозволяє оцінити ефективність запропонованої методики та її вплив на якість та швидкість створення відео.

Теоретичне значення одержаних результатів полягає у дослідженні методів автоматизації процесу створення відеоконтенту із застосуванням генеративного штучного інтелекту. Робота сприяє поглибленню знань про сучасні технології мультимедійної обробки, аналізу тексту, синтезу мовлення та алгоритми автоматичного підбору відео- і аудіоматеріалів. Отримані результати можуть бути використані у подальших дослідженнях, пов’язаних із розвитком

автоматизованих систем відеопродакшну, а також у сфері штучного інтелекту для генерації та обробки мультимедійного контенту.

Практичне значення роботи полягає у створенні програмного засобу, який дозволяє автоматизувати процес виробництва відео, значно скорочуючи витрати часу та ресурсів на його створення. Розроблена консольна програма може бути застосована в різних сферах, таких як контент-мейкінг, освіта, автоматизоване створення рекламних і навчальних відеоматеріалів. Крім того, результати роботи можуть бути корисними для розробників, які працюють у сфері автоматизації мультимедійного контенту, а також для компаній, що займаються генерацією відео за допомогою штучного інтелекту.

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

Римар П.В., Колібабчук Д.І. Програма для автоматизації створення відео з використанням генеративного штучного інтелекту. *Наука і техніка сьогодні (Серія «Педагогіка», Серія «Право», Серія «Економіка», Серія «Фізико-математичні науки», Серія «Техніка»)*. 2025. № 4(45). С. 1498–1510. https://doi.org/10.52058/2786-6025-2025-4(45)-1498-1509

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

Колібабчук Д. І., Римар П. В. Порівняльний аналіз систем для автоматизованого озвучення відео. *Прикладні інформаційні технології 2025*: Матеріали всеукр. науково-практ. конф. здобувачів вищ. освіти та молодих вчен., м. Вінниця, 22 трав. 2025р. 2025.

Структура кваліфікаційної (бакалаврської) роботи. Робота містить 3 розділи, 8 підрозділів, 8 таблиць, 29 рисунків, 37 літературних джерел. Загальний об’єм роботи складає 78 сторінок.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Постановка задачі

У межах даної кваліфікаційної роботи ставиться задача розробки програмного забезпечення для автоматизації процесу створення відеоконтенту на основі текстових даних. Кінцевим продуктом розробки має стати консольна програма, яка інтегрує сучасні технології генеративного штучного інтелекту та обробки медіа.

По перше, потрібно розробити програмний модуль для обробки вхідного тексту. Система повинна приймати готовий текст для відео або, за необхідності, генерувати сценарій на основі наданих ключових слів чи теми, використовуючи моделі обробки природної мови. Необхідно реалізувати механізм семантичного аналізу тексту для виділення ключових сутностей та визначення контексту для подальшого підбору візуального ряду.

Потрібно реалізувати інтеграцію з API сервісів стокових медіа. Програма повинна взаємодіяти з програмними інтерфейсами для автоматизованого пошуку та завантаження релевантних відеофрагментів та фонової музики. Алгоритм пошуку має базуватися на результатах семантичного аналізу тексту, забезпечуючи відповідність візуального контенту змісту сценарію.

Важливо імплементувати функціонал синтезу мовлення. Необхідно інтегрувати інструменти перетворення тексту на мовлення (Text-to-Speech) для генерації голосового супроводу на основі підготовленого тексту. Система має дозволяти певне налаштування параметрів синтезу (наприклад, вибір голосу) через конфігураційні файли. Також передбачається використання моделей штучного інтелекту для можливих завдань транскрипції (хоча основний потік орієнтований на текст).

Також потрібно розробити модуль компіляції та рендерингу відео. Використовуючи бібліотеки обробки відео, необхідно створити механізм для збірки фінального відеоролика. Цей модуль повинен об’єднувати підібрані

відеофрагменти, синхронізувати їх із згенерованою аудіодоріжкою та фоновою музикою.

Параметри роботи програми (ключі API, налаштування моделей ШІ, параметри вихідного відео тощо) мають зберігатися у зовнішніх конфігураційних файлах, що дозволить адаптувати систему без модифікації програмного коду.

Програма повинна мати зрозумілий та функціональний консольний інтерфейс для взаємодії з користувачем, приймання вхідних даних (текст або шлях до файлу) та параметрів генерації.

Таким чином, постановка задачі полягає у створенні програмного комплексу, який реалізує повний цикл автоматизованого виробництва відео: від обробки або генерації текстового сценарію до видачі готового аудіовізуального продукту, що поєднує відповідний відеоряд, музичний супровід та синтезований голосовий супровід. Результатом має стати інструмент, що дозволяє генерувати відеоконтент на основі тексту з мінімальним втручанням користувача у сам процес виробництва.

У підсумку, розроблений програмний комплекс має стати потужним інструментом для автоматизації створення відеоконтенту, що значно скорочує час та ресурси, необхідні для виробництва професійних відеоматеріалів. Це відкриває нові можливості для креативних професіоналів, маркетологів, освітян та інших користувачів, які потребують швидкого та якісного відеоконтенту для своїх цілей.

1.2 Аналіз категорій і понять

У сучасному цифровому ландшафті спостерігається значне зростання інтересу до автоматизації процесу створення відео, що зумовлено поширенням цифрових платформ та зростаючим попитом на відеоконтент у різних галузях. У цьому контексті генеративний штучний інтелект (ШІ) виступає як трансформаційна технологія, що відкриває нові можливості для створення відеоконтенту з мінімальною участю людини. Необхідно провести аналіз ключових категорій та понять для розробки програми, спрямованої на

автоматизацію створення відео з використанням генеративного ШІ. У роботі буде розглянуто основні дефініції, співвідношення між автоматизацією та креативністю, еволюцію технологій генерації відеоконтенту, а також ключові терміни генеративного ШІ, що використовуються у відеосинтезі.

Автоматизація створення відео визначається як використання технологій, зокрема штучного інтелекту, для спрощення та оптимізації різних етапів виробництва відео, починаючи від його створення та редагування до кінцевого розповсюдження. Це спрямовано на підвищення ефективності, зниження витрат та забезпечення масштабованого виробництва відеоконтенту для різноманітних платформ і застосувань[1].

З академічної точки зору, редагування відео визначають як процес нарізання та з'єднання фрагментів з одного або кількох джерел для створення відредагованого фільму, а інструменти для редагування відео – як комп'ютерні програми, що дозволяють виконувати ці завдання. Вони також зазначають, що ШІ використовується для автоматизації або розширення можливостей редакторів відео. Це раннє визначення підкреслює початковий фокус ШІ на допомозі людині в конкретних завданнях редагування[2].

Проте сучасне розуміння автоматизації створення відео є значно ширшим. Автоматизоване відео, або ШІ-відео, передбачає використання технологій для автоматизації як процесу виробництва, так і редагування відео. Програмне забезпечення використовує ШІ для створення відео з шаблонів, додавання субтитрів, застосування переходів і навіть включення вступу та кінцівки. У контексті YouTube автоматизація включає автоматизацію створення, редагування та завантаження відео, допомагаючи підтримувати стабільний потік контенту та оптимізувати робочий процес.

З індустріальної точки зору, автоматизація створення відео розглядається як спосіб заощадити час, підвищити послідовність, знизити витрати та забезпечити персоналізацію контенту. Інструменти автоматизації можуть генерувати різні варіанти одного відео, створювати гіперперсоналізовані відео на основі даних користувачів, автоматично змінювати розмір відео для різних

платформ і навіть усувати необхідність запису аудіо за допомогою технології перетворення тексту на мовлення[3].

Отже, автоматизація створення відео являє собою використання технологій, особливо штучного інтелекту, для оптимізації та спрощення всього циклу виробництва відео, від генерації контенту до його розповсюдження, з метою підвищення ефективності, зниження витрат та забезпечення масштабованості.

Генеративний штучний інтелект – це підмножина ШІ, яка використовує моделі машинного навчання, навчені на великих обсягах даних, для створення нового контенту, такого як текст, зображення, аудіо або відео, у відповідь на запит. На відміну від пошукових систем, генеративний ШІ не знаходить існуючі джерела, а створює новий контент, передбачаючи наступний елемент у послідовності даних[4].

Генеративний ШІ навчається, імітуючи великі обсяги даних для створення контенту, включаючи відео, на основі вхідних даних або підказок. Він використовує обчислювальні методи, здатні генерувати зовні новий, змістовний контент із навчальних даних. Ключовою відмінністю є здатність генерувати оригінальний контент шляхом прогнозування наступного слова, звуку чи пікселя в шаблоні, вивченому з масивних наборів даних[5].

Принцип роботи генеративного ШІ полягає у використанні моделей машинного навчання, зокрема глибокого навчання, для аналізу та вивчення закономірностей, стилів і структур у великих наборах даних. Після навчання модель здатна генерувати нові екземпляри даних, які мають подібні характеристики до тих, на яких вона навчалася. Цей процес включає етапи навчання, де модель налаштовує свої внутрішні параметри для мінімізації розбіжностей між згенерованим контентом і реальними даними, та етап генерації, де модель використовує вивчені закономірності для створення нового контенту у відповідь на запит[6].

У медіаіндустрії генеративний ШІ знаходить широке застосування, включаючи створення реалістичних зображень, написання текстів, композицію

музики та синтез відео. Він використовується для автоматичного створення відео з текстових описів, генерування сценаріїв, створення закадрового голосу та навіть для редагування відео. Генеративний ШІ також використовується для персоналізації контенту, адаптації відео до індивідуальних уподобань глядачів та для оптимізації рекламних кампаній[7][8].

Таким чином, генеративний ШІ є потужним інструментом, який використовує машинне навчання для створення нового та змістовного контенту, включаючи відео, відкриваючи широкі можливості для автоматизації та інновацій у медіавиробництві.

Роль нейронних мереж у генеративному відеосинтезі. Нейронні мережі, особливо глибокі нейронні мережі, є основною архітектурою багатьох генеративних моделей ШІ, що використовуються для синтезу відео. Натхненні структурою та функціями людського мозку, нейронні мережі складаються з тисяч або мільйонів простих обробних вузлів, щільно з'єднаних у шари. Дані проходять через ці шари, де кожен вузол виконує прості обчислення, а зв'язки між вузлами мають ваги, які регулюються в процесі навчання[9].

Глибокі нейронні мережі, які мають багато прихованих шарів, здатні вивчати складні закономірності та ієрархічні представлення даних, що є критично важливим для розуміння та відтворення складної природи відеоконтенту. У контексті генеративного відеосинтезу нейронні мережі використовуються для вивчення зв'язків між візуальними елементами, рухом та звуком у відеоданих.

Процес навчання нейронної мережі для генерації відео зазвичай включає подання великого набору реальних відеоданих. Мережа намагається відтворити ці дані та її внутрішні параметри (ваги зв'язків між вузлами) поступово налаштовуються на основі різниці між згенерованим виходом і реальними даними. Цей процес, відомий як зворотне поширення, дозволяє мережі вчитися генерувати нові відео, які мають статистичні характеристики, подібні до навчальних даних.

Існують різні типи нейронних мереж, які використовуються для генерації відео, включаючи згорткові нейронні мережі (CNN), які добре підходять для обробки візуальних даних, та рекурентні нейронні мережі (RNN) і трансформерні мережі, які ефективні для моделювання послідовних даних, таких як відеокадри. Комбінації цих архітектур часто використовуються для створення складних генеративних моделей, здатних створювати високоякісний та реалістичний відеоконтент.

Отже, нейронні мережі, особливо глибокі та спеціалізовані архітектури, відіграють центральну роль у генеративному відеосинтезі, забезпечуючи моделям ШІ можливість вивчати складні закономірності відеоданих та генерувати новий, правдоподібний контент.

Машинне навчання в медіа виробництві: забезпечення автоматизованих процесів. Машинне навчання (МН) є підмножиною штучного інтелекту, яка дозволяє комп'ютерам навчатися на даних без явного програмування. У контексті медіа виробництва машинне навчання використовується для автоматизації різних процесів, включаючи аналіз контенту, редагування відео, персоналізацію та створення контенту[10].

Основний принцип машинного навчання полягає в тому, що алгоритми навчаються на великих наборах даних, виявляють закономірності та використовують ці знання для прийняття рішень або прогнозування майбутніх результатів. Процес машинного навчання зазвичай включає етапи збору та підготовки даних, вибору відповідної моделі, навчання моделі на даних, оцінки її продуктивності та подальшого налаштування для покращення точності.

У медіа виробництві машинне навчання застосовується для широкого спектру завдань. Наприклад, у відеоредагуванні алгоритми машинного навчання можуть автоматично виконувати такі завдання, як виявлення змін сцени, розпізнавання об'єктів, додавання субтитрів та навіть пропонування варіантів редагування на основі аналізу контенту. У сфері створення контенту машинне навчання є основою для генеративного ШІ, дозволяючи створювати нові відео, зображення та текст на основі текстових описів або інших вхідних даних[11].

1.3 Аналіз комерційних платформ для генерації відео з використанням ШІ

Fliki є AI-платформою, яка дозволяє користувачам конвертувати текст у відео, використовуючи при цьому штучні голоси та бібліотеку медіа-активів. Платформа позиціонується як інструмент з інтуїтивно зрозумілим інтерфейсом, розроблений для контент-креаторів, які не мають спеціальних навичок у відеомонтажі. Fliki використовує штучний інтелект для перетворення тексту на відео, пропонуючи широкий вибір (понад 2500) AI-голосів більш ніж 80 мовами, а також доступ до бібліотеки стокових зображень та відео[12].

Процес створення відео у Fliki складається з чотирьох основних кроків: введення тексту, персоналізація AI-голосу, генерація AI-візуалів та перегляд/експорт. До переваг платформи належать простота використання, великий вибір AI-голосів та мов, наявність готових шаблонів, а також позитивні відгуки користувачів щодо зручності та підвищення продуктивності[12].

Серед недоліків можна відзначити обмеження на обсяг введеного тексту (до 15 000 символів), обмежені можливості кастомізації відео порівняно з більш професійними інструментами, залежність від стокових матеріалів та потенційно «роботизоване» звучання голосів. Деякі користувачі також відзначають обмеження в якості відео та рівні контролю[12].

Аналіз відгуків та порівнянь показує, що Fliki є зручним та швидким інструментом, особливо для користувачів без досвіду у відеомонтажі. Проте, відсутність глибоких налаштувань може бути обмеженням для більш вимогливих проєктів. Велика кількість користувачів (понад 8 мільйонів) та високі оцінки задоволеності (4,8/5) свідчать про значний попит на прості у використанні рішення для генерації відео з тексту. Орієнтація Fliki на широкий спектр голосів та мов підкреслює важливість доступності та локалізації у сфері відеогенерації[12].

Рисунок 1.1 – Зразок відео яке створене Fliki

VEED.IO є онлайн-платформою для редагування відео, яка також пропонує AI-інструменти для перетворення тексту на відео. Платформа орієнтована на створення привабливих відео для соціальних мереж, освіти та маркетингу. VEED.IO використовує AI-генератор відео, AI-генератор сценаріїв, функціональність перетворення тексту на мовлення з різними профілями голосів та мовами, а також інтеграцію з повноцінним онлайн-редактором відео. Платформа також застосовує AI для таких функцій, як автоматичні субтитри та видалення фону[13][14][15][16][17].

Процес створення відео включає введення запиту, налаштування відео (формат, стиль, сценарій) та подальше редагування. До переваг VEED.IO належать зручний інтерфейс, безшовна інтеграція AI-інструментів з функціями редагування, широкий набір інструментів редагування, а також наявність автоматичних субтитрів та перекладів[14][15][16][17][18].

Серед недоліків можна відзначити наявність водяних знаків у безкоштовній версії, потенційні складності при роботі з завантаженими користувачем медіафайлами, а також те, що платформа може не підходити для професійного редагування відео високого рівня . Деякі відгуки також вказують на проблеми з експортом відео[18][19].

Аналіз відгуків та порівнянь показує, що VEED.IO забезпечує баланс між автоматизацією на основі AI та контролем користувача завдяки інтегрованому редактору відео. Значний трафік вебсайту (23,76 млн візитів у лютому 2025 року) та велика кількість щомісячних активних користувачів (3 млн) свідчать про сильну ринкову позицію та популярність VEED.IO. Орієнтація платформи на створення контенту для соціальних мереж підкреслює ключову сферу застосування інструментів автоматизованої генерації відео.

Рисунок 1.2 – Зразок відео яке створене VEED.IO

Steve AI позиціонується як AI-генератор відео, розроблений для швидкого та легкого створення відео, особливо для користувачів без значного досвіду у відеомонтажі. Платформа робить акцент на різноманітних типах відео, включаючи анімаційні та live-action. Steve AI використовує генеративний AI для перетворення тексту на відео, AI-перетворення голосу на відео, пропонує бібліотеку AI-аватарів (понад 300) та велику медіа-бібліотеку. Платформа також підтримує конвертацію блогів та аудіо у відео.

Процес створення відео часто включає написання або генерацію сценарію, вибір шаблону та автоматичну генерацію відео за допомогою AI. До переваг Steve AI належать простота використання, широкий вибір шаблонів та стилів

відео, наявність AI-озвучення та аватарів, а також придатність для різних сценаріїв використання, таких як маркетинг та освіта[21].

Серед потенційних недоліків можна відзначити нижчу якість відео порівняно з більш професійними інструментами, обмежені можливості розширеного редагування та залежність від шаблонів, що може обмежувати створення унікальних відео[22].

Аналіз відгуків та порівнянь показує, що Steve AI орієнтований на користувачів, яким потрібне швидке та легке створення відео для конкретних цілей, таких як соціальні мережі або презентації. Заявлена велика кількість користувачів (понад 25 мільйонів), хоча попередні дані вказували на меншу кількість (1 мільйон у червні 2023 року), свідчить про значний ринок для шаблонних AI-інструментів генерації відео. Орієнтація Steve AI на конвертацію різних типів вхідних даних (текст, аудіо, блоги) у відео підкреслює універсальність, яку шукають користувачі в автоматизованій генерації відео[22][23].

Рисунок 1.3 – Зразок відео яке створене Steve AI

Elai.io є AI-генератором відео, який дозволяє користувачам створювати відео з AI-аватарами на основі лише тексту. Платформа робить акцент на створенні професійного та ефективного відеоконтенту для корпоративного навчання та маркетингу. Elai.io використовує AI-аватари (понад 80 готових, можливість створення власних), функцію перетворення тексту на мовлення

більш ніж 75 мовами, конвертацію статей та презентацій PowerPoint у відео, а також такі функції, як клонування голосу та автоматичні переклади. Платформа також пропонує інтерактивні функції відео.

Процес створення відео часто включає вибір шаблону, введення тексту, вибір аватара та генерацію відео. До переваг Elai.io належать висока якість вихідного відео, широкий вибір аватарів та голосів, простота використання, потужна підтримка багатьох мов та унікальні можливості інтерактивного відео[24].

Серед потенційних недоліків можна відзначити обмежені можливості кастомізації за межами шаблонів та аватарів, іноді «роботизований» вигляд аватарів, потенційну ненадійність служби підтримки та випадкову нестабільність програмного забезпечення. [26]

Аналіз відгуків та порівнянь показує, що Elai.io орієнтований на створення професійних відео з AI-презентаторами, націлених на бізнес та освітніх користувачів. Позитивні відгуки користувачів та акцент на корпоративному навчанні свідчать про значний ринок для AI-генерації відео в сфері корпоративного навчання та комунікацій. Інтерактивні функції Elai.io вказують на тенденцію до більш залучаючого та персоналізованого відеоконтенту. [24]

Рисунок 1.4 – Зразок відео яке створене Elai.io

Крім вищезазначених, існують й інші комерційні платформи, що пропонують схожу функціональність. До них належать Synthesia, яка

спеціалізується на створенні відео з реалістичними AI-аватарами студійної якості; Runway , що пропонує інструменти генеративного AI для відео з акцентом на високу якість та креативність; InVideo AI , яка орієнтована на створення відео для соціальних мереж; Peech для команд, що займаються контент-маркетингом; revid.ai з AI-шаблонами; LTX Studio для користувачів, які потребують високого рівня творчого контролю; Animaker, що спеціалізується на анімаційних відео; Pictory для створення коротких відео для соціальних мереж з довгого контенту; Deepbrain AI з реалістичними AI-аватарами; HeyGen для створення персоналізованих відео з говорючими аватарами; та інші платформи, згадані в . Кожна з цих платформ має свої особливості, переваги та недоліки, орієнтуючись на різні потреби користувачів та сценарії використання.

Таблиця 1.1 – Порівняння ключових функцій комерційних платформ

Платформа	Основна функціональність	Інтеграція стокових медіа	Рівень кастомізації	Модель ціноутворення
Fliki	Текст у відео, AI-голоси	Так	Обмежений	Підписка
VEED.IO	Редагування відео, текст у відео	Так	Помірний	Підписка, безкоштовний план
Steve AI	Текст у відео, AI-аватари	Так	Обмежений	Підписка, безкоштовний план
Elai.io	Текст у відео, AI-аватари	Ні	Помірний	Підписка, безкоштовний план
Synthesia	AI-аватари, текст у відео	Ні	Високий	Підписка
Runway	Генеративне AI для відео	Ні	Високий	Кредитна система
InVideo AI	Текст у відео, AI-інструменти	Так	Помірний	Підписка, безкоштовний план

1.4 Аналіз відкритих моделей для генерації відео з використанням ШІ

Поряд з комерційними рішеннями, існують також моделі з відкритим вихідним кодом, які демонструють значний прогрес у сфері генерації відео на основі тексту.

HunyuanVideo є відкритою фундаментною моделлю для генерації відео, розробленою Tencent. За результатами оцінок професійних експертів, її продуктивність у генерації відео є порівнянною або навіть перевершує провідні комерційні моделі. Модель має понад 13 мільярдів параметрів, що робить її найбільшою серед усіх відкритих моделей у цій галузі.

Архітектура HunyuanVideo включає використання Causal 3D VAE для просторово-часового стиснення та трансформерної архітектури з механізмом Full Attention для уніфікованої генерації зображень та відео. Модель використовує гібридний дизайн «Dual-stream to Single-stream» для генерації відео[28][29].

До ключових особливостей HunyuanVideo належать текстовий енкодер MLLM, 3D VAE для ефективного стиснення латентного простору та механізм переписування промптів[28][29].

За результатами бенчмарків, HunyuanVideo демонструє високі показники, зокрема, лідируючи в VBench. У порівнянні з такими моделями, як Runway Gen-3 та Luma 1.6, HunyuanVideo показала кращі результати, особливо в якості руху. Розробники також застосували нові методи масштабування, які дозволили знизити обчислювальні витрати до 80%[28][29].

Перевагами HunyuanVideo є її відкритий вихідний код, висока якість генерації, відмінна якість руху та доступність коду для висновування та попередньо навчених моделей. Недоліком є значні обчислювальні ресурси, необхідні для навчання та висновування, що може обмежити її використання для окремих користувачів зі стандартним обладнанням.

Рисунок 1.5 – Зразки генерації з HunyuanVideo, що демонструють реалістичні концептуальні узагальнення та автоматичні функції розбиття сцени

Open-Sora є ініціативою з відкритим вихідним кодом, спрямованою на ефективне створення високоякісного відео та демократизацію доступу до передових технологій генерації відео. Основною метою проєкту є досягнення рівня продуктивності комерційних моделей при контрольованих витратах на навчання.

Архітектура Open-Sora включає Spatial-Temporal Diffusion Transformer (STDiT) та ефективний фреймворк 3D автоенкодера. Модель також використовує попередньо навчені моделі, такі як T5-XXL та CLIP-Large[31][32][33].

Ключові особливості Open-Sora включають підтримку генерації відео з тексту та зображень, змінну роздільну здатність та тривалість, підтримку будь-якого співвідношення сторін, а також розширені елементи керування, такі як оцінка руху та покращення промптів. Важливою є також економічно ефективна стратегія навчання[31][32][33].

За результатами бенчмарків VBench, Open-Sora 2.0 демонструє результати, порівнянні з провідними моделями, такими як HunyuanVideo та Runway Gen-3 Alpha. Оцінки експертів також свідчать про високу якість зображення, відповідність промптам та якість руху. Значним є також зменшення розриву в продуктивності з OpenAI Sora[30].

Перевагами Open-Sora є відкритий вихідний код, висока продуктивність, економічно ефективне навчання та доступність коду для висновування та попередньо навчених моделей. Незважаючи на прагнення до ефективності, запуск цих моделей все ще вимагає значних обчислювальних ресурсів.

Рисунок 1.6 – Високоякісне відео яке згенероване Open-Sora 2.0 [31]

Крім HunyuanVideo та Open-Sora, існують й інші відкриті моделі, що заслуговують на увагу. Mochi, розроблена Genmo, позиціонується як високоякісна модель для генерації відео з тексту. Wan2.1 від Alibaba пропонує моделі з різною кількістю параметрів (14 мільярдів та 1,3 мільярда) та інтеграцію з ComfyUI. AnimateDiff-Lightning від Bytedance є швидшою версією популярної моделі AnimateDiff, яка використовується як відеоадаптер для існуючих моделей генерації зображень з тексту, таких як Stable Diffusion. Step-Video-T2V є моделлю з 30 мільярдами параметрів від маловідомого стартапу Stepfun. Ці моделі демонструють різний рівень продуктивності та функціональності, надаючи дослідникам та розробникам різноманітні інструменти для експериментів у галузі генерації відео з тексту. Дані для наступних таблиць були взяти з [30]

Таблиця 1.2 – Порівняння ключових функцій та продуктивності відкритих моделей

Модель	Ключові архітектурні особливості	Приблизна кількість параметрів	Звітна продуктивність (VBench)	Простота використання / розгортання
HunyuanVideo	Causal 3D VAE, Transformer з Full Attention, Dual-stream to Single-stream	> 13 мільярдів	Висока	Потребує значних технічних знань
Open-Sora	STDiT, 3D автоенкодер	11 мільярдів (v2.0)	Порівнянна з HunyuanVideo	Потребує значних технічних знань
Mochi	Невідомо	10 мільярдів	Порівнянна з HunyuanVideo	Відносно просте розгортання на Modal
Wan2.1	Трансформер	14 мільярдів, 1.3 мільярда	Висока	Потребує знань ComfyUI
AnimateDiff-L	Відеоадаптер для моделей генерації зображень з тексту (Stable Diffusion)	Залежить від базової моделі	Залежить від базової моделі	Потребує знань ComfyUI

Таблиця 1.3 – Бенчмарк моделей на VBench

Модель	Доступність	Дата	Загальний бал	Бал якості	Семантичний бал
Open-Sore 2.0	Відкритий код	2025-03-14	81.71%	82.10%	80.14%
Wan2.1-T2V-1.3B	Відкритий код	2025-03-13	84.26%	85.30%	80.09%
Sora	API	2025-01-14	84.28%	85.51%	79.35%
HunyuanVideo	Відкритий код	2024-12-16	83.24%	85.09%	75.82%
Mochi-1	Відкритий код	2024-11-07	80.13%	82.64%	70.08%
AnimateDiff-V1	Відкритий код	2024-08-02	77.46%	80.24%	66.32%

Висновок до розділу 1

У цьому розділі було сформульовано завдання розробки програмного комплексу для автоматизації створення відеоконтенту на основі текстових даних. Визначено основні модулі майбутньої системи, що включають обробку тексту, інтеграцію з API стокових медіа, синтез мовлення та компіляцію фінального відеоролика. Далі проведено аналіз ключових категорій та понять, таких як автоматизація створення відео, генеративний штучний інтелект, а також роль нейронних мереж та машинного навчання у цій сфері. Було розглянуто комерційні платформи для генерації відео, наприклад Fliki та VEED.IO, проаналізовано їхні функції та особливості. На завершення, здійснено аналіз відкритих моделей штучного інтелекту, зокрема HunyuanVideo та Open-Sora, призначених для генерації відео, та їхніх архітектурних рішень.

РОЗДІЛ 2
КОНЦЕПТУАЛЬНА МОДЕЛЬ ТА ОПИС ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Концептуальна модель системи автоматизації створення відео

Система автоматизації створення відео являє собою програмний комплекс, що реалізує послідовний конвеєр обробки даних та генерації контенту. В основі концепції лежить модульний підхід, де кожен етап обробки виконується спеціалізованим компонентом системи. Взаємодія між компонентами здійснюється через чітко визначені інтерфейси та структури даних. Загальна архітектура системи передбачає наступні ключові модулі:

1. **Модуль обробки вхідних даних:** Відповідає за отримання та первинну обробку тексту, що слугуватиме основою для відео.
2. **Модуль аналізу та структурування тексту:** Виконує семантичний аналіз тексту, виділення ключових сутностей та поділ на логічні сегменти (сцени).
3. **Модуль генерації мовлення (Text-to-Speech, TTS):** Синтезує аудіодоріжку на основі обробленого тексту.
4. **Модуль пошуку та підбору медіаконтенту:** Здійснює пошук релевантних стокових відео та фонової музики через зовнішні API на основі результатів аналізу тексту.
5. **Модуль синхронізації та монтажу:** Відповідає за часове узгодження аудіо та відео доріжок, накладання ефектів (за потреби) та формування єдиної відео послідовності.
6. **Модуль рендерингу:** Здійснює фінальну обробку та експорт готового відеофайлу у заданому форматі.
7. **Модуль конфігурації:** Забезпечує керування параметрами системи (ключі API, налаштування моделей ШІ, параметри вихідного відео тощо).

Взаємодія модулів відбувається послідовно, хоча деякі процеси, як-от генерація мовлення та пошук медіа, можуть виконуватись паралельно після етапу аналізу тексту для оптимізації загального часу роботи.

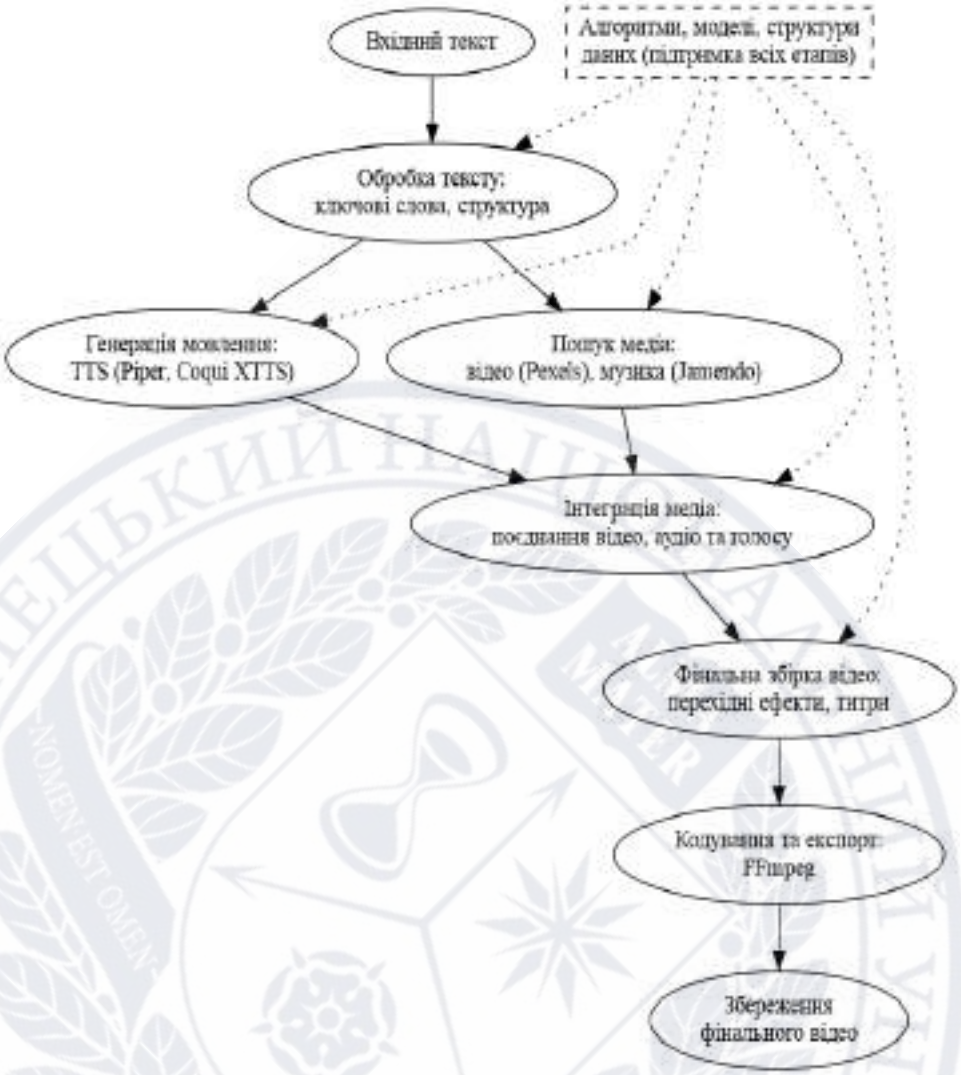

Рисунок 2.1 – Загальна блок-схема архітектури системи автоматизації створення відео

Першим етапом є ініціалізація та обробка вхідного тексту. На цьому етапі система отримує вхідні дані та готує їх для подальшої обробки. Відбувається завантаження конфігурації, тобто зчитування параметрів з файлу конфігурації (наприклад, у форматі JSON або YAML), що містить ключі до API, шляхи до моделей, налаштування якості відео, параметри голосу тощо. Далі здійснюється отримання тексту, яке може відбуватися двома способами: або користувач надає готовий текст як вхідний параметр командного рядка або через файл, або він надає тему або ключові слова, і система використовує велику мовну модель (LLM), наприклад, подібну до GPT (Generative Pre-trained Transformer), для генерації тексту сценарію за заданими параметрами.

Після цього виконується первинна обробка тексту, що включає видалення зайвих символів, нормалізацію тексту, а також можливу перевірку орфографії та граматики. Завершальним кроком цього етапу є сегментація тексту, тобто розбиття його на логічні одиниці, що відповідатимуть окремим сценам або фрагментам відео. Це може бути здійснено на рівні речень або абзаців, використовуючи як прості алгоритми на основі розділових знаків, так і більш складні моделі обробки природної мови (NLP) для аналізу структури тексту.

Другим етапом є аналіз тексту та виділення ключових понять. Метою цього етапу є вилучення з тексту інформації, необхідної для підбору релевантного візуального ряду та визначення загального настрою відео. Спершу виконується виділення ключових слів або фраз: для кожного сегмента тексту, тобто речення чи абзацу, визначаються слова чи вирази, які найкраще описують його зміст. Для цього можуть застосовуватись як статистичні методи, наприклад TF-IDF, так і моделі обробки природної мови на основі нейронних мереж, зокрема BERT-подібні моделі, що використовуються для виділення іменованих сутностей та ключових понять. Далі може здійснюватися семантичний аналіз – опціональний етап, що передбачає визначення тональності тексту, тобто його емоційного забарвлення (позитивна, негативна або нейтральна оцінка), як для окремих сегментів, так і для всього тексту загалом. Така інформація може бути використана для вибору відповідної музики та візуального стилю відео. Для аналізу тональності можуть використовуватися як великі мовні моделі, так і спеціалізовані моделі аналізу сентименту. Після цього формується структура сценарію – внутрішнє представлення, яке описує послідовність сцен, їх текстовий зміст, виділені ключові слова, а також попередню тривалість кожної сцени, що буде уточнена після генерації аудіо.

Таблиця 2.1 – Приклад структури даних для опису сцени

Поле	Опис	Приклад значення
scene_id	Унікальний ідентифікатор сцени	1
text_segment	Текстовий фрагмент, що відповідає сцені	«Генеративний ШІ швидко розвивається».
keywords	Список ключових слів/фраз для пошуку відео	[«штучний інтелект», «розвиток», «технології»]
sentiment	Тональність сегменту (опціонально)	«neutral»
audio_duration	Тривалість згенерованого аудіо для цього сегменту (у секундах)	3.5
video_clip_path	Шлях до підібраного відеофайлу для цієї сцени	«/path/to/selected_video_1.mp4»
start_time	Час початку сцени у фінальному відео (у секундах)	0.0
end_time	Час кінця сцени у фінальному відео (у секундах)	3.5

Наступним етапом є генерація аудіоряду. На цьому етапі створюється голосовий супровід для відео. Він включає перетворення текстових сегментів у звукову форму, вимірювання їх тривалості та можливе об’єднання в єдиний аудіопотік. Кожен текстовий сегмент зі структури сценарію передається до системи синтезу мовлення (TTS). Для цього можна використовувати такі моделі, як Tacotron або FastSpeech, або ж хмарні сервіси, наприклад Google Cloud TTS чи Azure Cognitive Services. Модель обробляє текст і генерує відповідний аудіофайл для кожного фрагмента.

Після створення аудіофрагментів фіксується їхня точна тривалість. Ці дані записуються у структуру сценарію в поле audio_duration, що дозволяє точно синхронізувати звук із візуальним рядом. За потреби окремі аудіофайли можна об’єднати в єдину доріжку. При цьому зберігається інформація про часові мітки початку та кінця кожного сегмента, що важливо для подальшого монтажу та корегування відео.

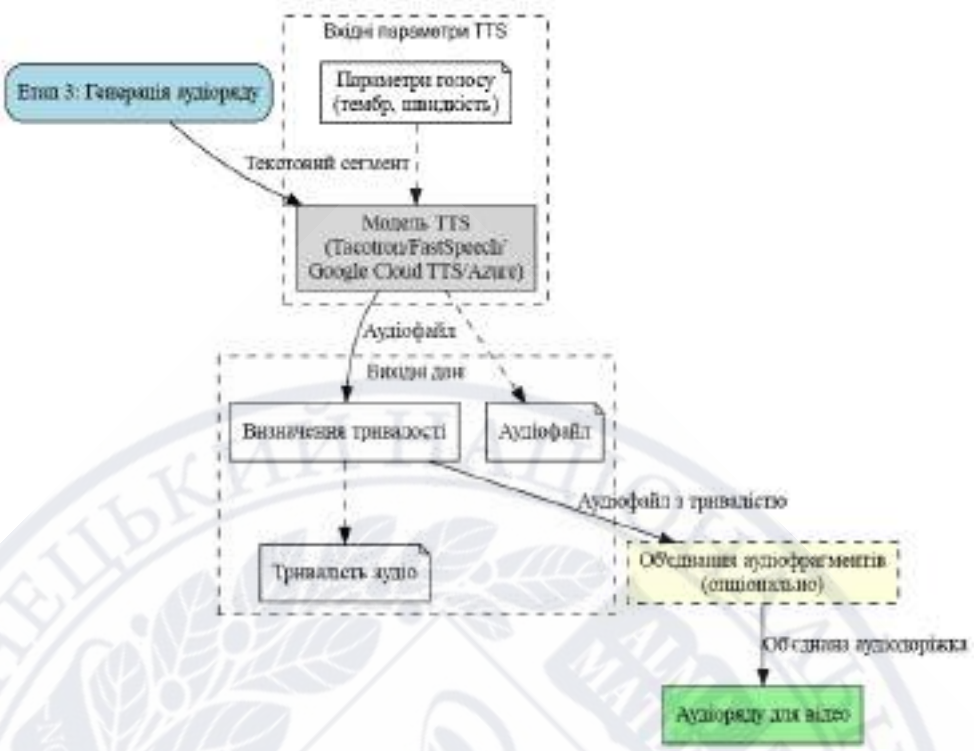

Рисунок 2.2 – Процес генерації мовлення

Наступний етап це пошук та підготовка медіаконтенту. Цей етап передбачає пошук та підготовку візуальних матеріалів і фонової музики, які будуть використовуватися у відео. Процес може відбуватися паралельно з генерацією аудіо або послідовно.

Система здійснює пошук відеоматеріалів, використовуючи ключові слова, отримані на другому етапі. Для цього надсилаються запити до API популярних стокових сервісів, наприклад таких як Pexels, Pixabay або Shutterstock. Це дозволяє отримати широкий вибір відео, що відповідають тематиці проекту.

Отримані результати фільтруються за низкою параметрів, включаючи орієнтацію (горизонтальну або вертикальну), роздільну здатність та тривалість. Алгоритм відбору враховує релевантність відео ключовим словам, доступність за ліцензією та відповідність тривалості аудіофрагментів (поле audio_duration). Важливо, щоб тривалість обраного відео була не меншою за тривалість відповідного сегменту звукового супроводу.

Після вибору відповідних відеоматеріалів вони завантажуються локально для подальшої обробки. Шляхи до завантажених файлів фіксуються у структурі сценарію, що дозволяє легко інтегрувати їх у відеоредактор.

Аналогічним чином здійснюється пошук фонової музики. Система може використовувати API музичних бібліотек або локальні колекції, орієнтуючись на такі критерії, як жанр, настрій (визначений на другому етапі) або темп. Це забезпечує гармонійне поєднання музики із загальною атмосферою відео.

Обраний музичний трек завантажується для подальшого використання. Як і у випадку з відео, інформація про файл зберігається у структурі сценарію, що спрощує процес монтажу.

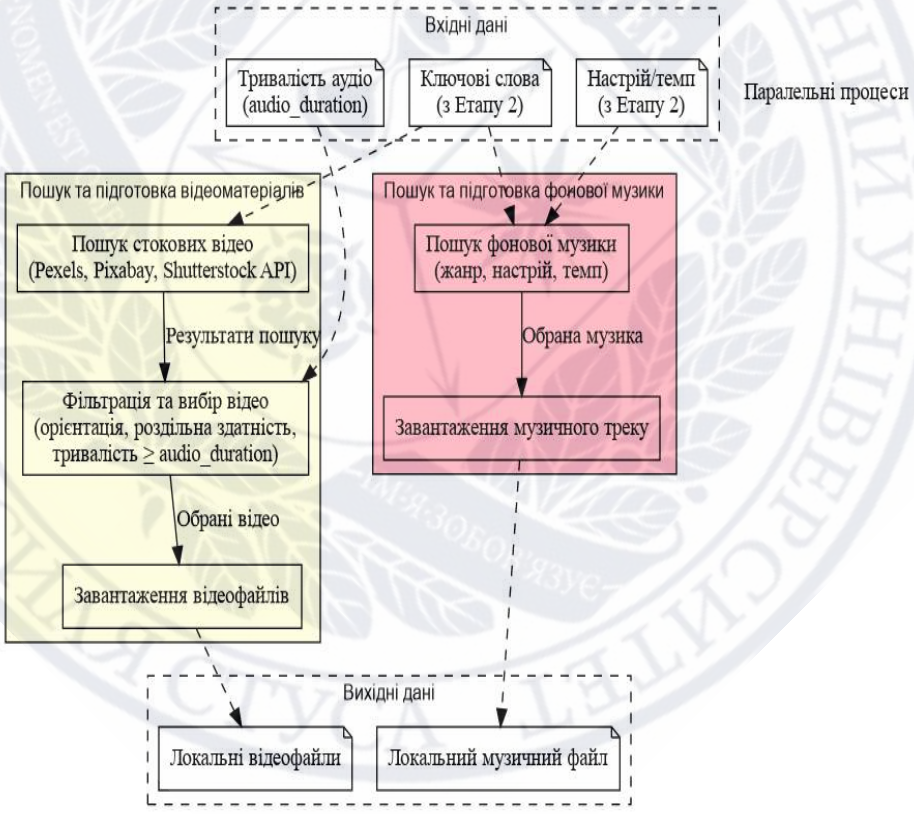

Рисунок 2.3 – Алгоритм пошуку та підготовка медіаконтенту

Останнім етапом є синхронізація, монтаж та рендеринг. Це заключний етап створення відеоролика, де всі підготовлені компоненти об’єднуються в єдине ціле. Процес включає кілька ключових операцій.

Кожен завантажений відеоматеріал адаптується під тривалість відповідної сцени. Відео обрізається або змінює швидкість відтворення (з урахуванням збереження якості), щоб точно відповідати тривалості аудіофрагменту (audio_duration). Якщо відео значно довше за необхідний сегмент, вибирається найбільш релевантний фрагмент.

Підготовлені відеокліпи з’єднуються у послідовність відповідно до структури сценарію. Між окремими сценами можуть додаватися плавні переходи, такі як згасання або інші візуальні ефекти, що покращують сприйняття.

Дикторський голос точно синхронізується з відеорядом за таймкодами. Фонова музика додається окремим шаром із автоматичним зниженням гучності під час мовлення (ефект ducking), що забезпечує чіткість дикторського тексту.

На завершальному етапі всі компоненти – відеоряд, дикторська доріжка та музичний супровід – об’єднуються в єдиний файл. Параметри експорту включають:

- Відеокодек (наприклад, H.264).
- Роздільну здатність (1920x1080).
- Частоту кадрів (30 fps).
- Бітрейт відео та аудіо.

Результатом є готовий до публікації відеоролик, який відповідає всім технічним вимогам та творчому задуму.

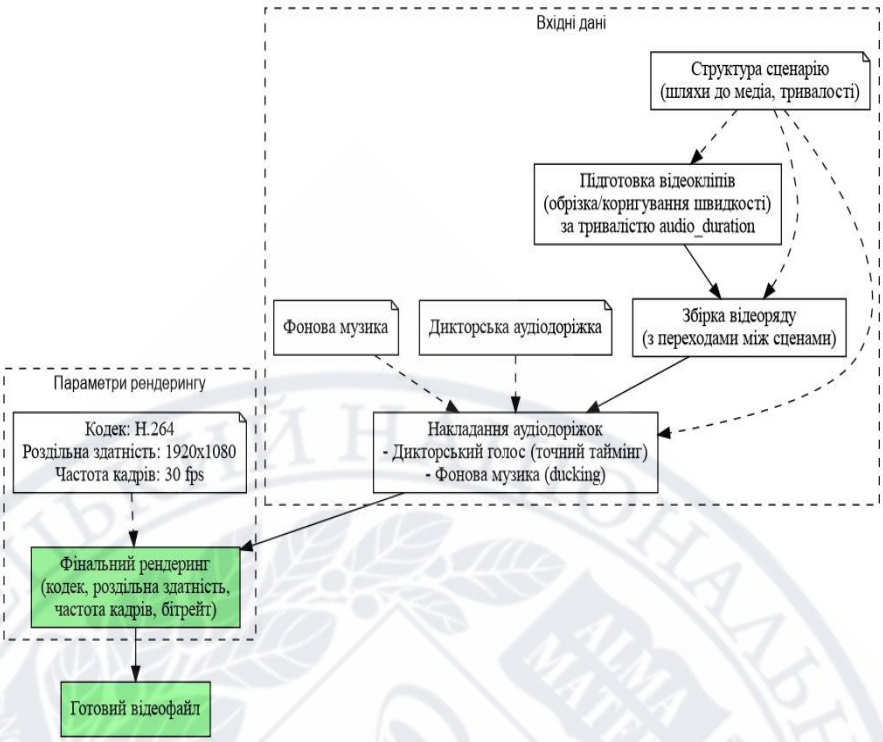

Рисунок 2.4 – Алгоритм синхронізації та збірки відео

2.2 Опис обраних моделей штучного інтелекту та сервісів для програми

Для генерації та обробки тексту було обрано моделі Gemini 2.0 flash та Gemma 3 1b та 4b.

Gemini 2.0 Flash є частиною сімейства моделей Gemini 2.0 від Google, розробленого з акцентом на швидкість та ефективність виконання різноманітних завдань. Ця модель позиціонується як потужний інструмент для щоденних завдань, що потребують швидкої обробки даних.

Однією з важливих особливостей Gemini 2.0 Flash є його здатність обробляти мультимодальні вхідні дані, включаючи текст, зображення, аудіо та відео. Хоча основним завданням є генерація тексту та вибір ключових слів на основі тексту, ця мультимодальність може стати в нагоді для майбутніх розширень функціоналу програми або для аналізу відеозаписів. Модель підтримує різні формати вхідних даних, такі як текст, код, зображення, аудіо та відео, і може надавати текстові результати, а також експериментально підтримує виведення зображень та аудіо.

Таблиця 2.2 – Основні відомості Gemini 2.0 Flash

Параметр	Значення
Підтримувані вхідні типи даних	Текст, рисунки, відео, аудіо
Підтримувані вихідні типи даних	Текст
Підтримувана кількість токенів для вхідних даних	1 мільйон
Підтримувана кількість токенів для вихідних даних	8 тисяч
Період актуальності знань	Червень 2024

Здатність безпосередньо обробляти відеоматеріали відкриває потенційну можливість для вибору ключових слів безпосередньо з відеоконтенту в майбутньому, що може бути ефективнішим, ніж обробка лише текстових транскриптів. Якщо відеоконтент сам містить релевантні ключові слова (наприклад, усне мовлення, текст на екрані), їхня пряма обробка може бути ефективнішою, ніж використання потенційно неповних або неточних транскриптів. Наприклад, Gemini 2.0 Flash може вилучати структуровані дані з різноманітних фінансових медіа, включаючи зображення та відео , що свідчить про його здатність аналізувати нетекстові елементи для виявлення ключових слів.

Значною перевагою Gemini 2.0 Flash є велике вікно контексту, що становить 1 мільйон токенів. Це дозволяє моделі обробляти довгі відеозаписи або одночасно аналізувати велику кількість документів. Велике вікно контексту може допомогти моделі зрозуміти загальну тему довгого відео або серії пов’язаних відео, що призведе до більш релевантного вибору ключових слів.

Модель характеризується високою швидкістю та низькою затримкою, що робить її придатною для застосувань у реальному часі або робочих процесів, де потрібна швидка обробка даних. Вона працює вдвічі швидше за Gemini 1.5 Pro.

Результати бенчмарків демонструють високу продуктивність Gemini 2.0 Flash у завданнях обробки природної мови. Модель показала результат 76.4% на бенчмарку MMLU-Pro, що свідчить про її потужні можливості розуміння мови та є значним покращенням у порівнянні з попередніми версіями. Високий

результат MMLU-Pro вказує на те, що модель має широке розуміння різних тем, що є корисним для виявлення релевантних ключових слів у відео різної тематики. Цей бенчмарк демонструє здатність моделі обробляти різноманітну лексику та концепції, які можуть зустрічатися у відеозаписах.

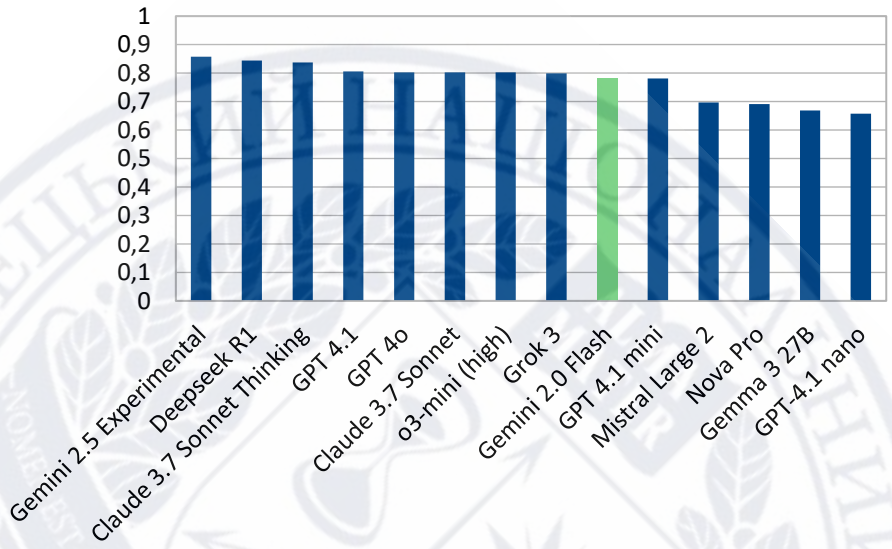

Рисунок 2.5 – Результати бенчмарку MMLU

Продуктивність на Natural2Code склала 92,9%, що вказує на високу ефективність обробки тексту, пов’язаного з кодом, що може бути важливим, якщо відеоконтент включає навчальні матеріали або обговорення програмування. Хоча цей результат безпосередньо не пов’язаний із загальним вибором ключових слів, він свідчить про високий рівень розуміння технічної мови, що може бути корисним для певних категорій відеоконтенту.

API Gemini пропонує щедрий безкоштовний план, який є вигідним для початкової розробки та експериментів. Безкоштовний план для Gemini 2.0 Flash включає обмеження на кількість запитів: 15 запитів на хвилину (RPM), 1 мільйон токенів на хвилину (TPM) та 1500 запитів на день (RPD). Цих лімітів має бути достатньо для розробки та тестування, але вони можуть стати обмеженням для великомасштабного виробничого використання. Для обробки великих обсягів може знадобитися платний план зі значно вищими лімітами (наприклад, 30 000

RPM). Слід зазначити, що використання безкоштовного плану може бути використано для покращення продуктів Google, що може викликати занепокоєння щодо конфіденційних даних. Для виробничих середовищ з конфіденційним відеоконтентом рекомендується перейти на платний план, де дані не використовуються для покращення моделі.

Таблиця 2.3 – Ліміти API безкоштовного плану моделей Gemini

Модель	RPM	TPM	RPD
Gemini 2.5 Pro Experimental	5	1,000,000	25
Gemini 2.0 Flash	15	1,000,000	1,500
Gemini 2.0 Flash-Lite	30	1,000,000	1,500
Gemini 2.0 Flash Thinking Experimental 01-21	10	4,000,000	1,500
Gemini 1.5 Flash	15	1,000,000	1,500

Окрім моделей Gemini також було обрано як альтернативний варіант локальну модель Gemma 1b та 3b від компанії Google. Сімейство моделей Gemma представлено як легкі, відкриті моделі, розроблені на основі досліджень Gemini. Gemma 3 є останньою ітерацією, побудованою на тій самій технології, що й Gemini 2.0. Вона доступна в різних розмірах, включаючи 1b та 4b. Gemma 3 1b є текстовою моделлю з вікном контексту 32k токенів, тоді як Gemma 3 4b є мультимодальною моделлю, здатною обробляти текст та зображення з вікном контексту 128k токенів.

Моделі мають відносно невеликий розмір (1 мільярд та 4 мільярди параметрів) у порівнянні з більшими хмарними моделями, що дозволяє локально розгортати їх на споживчому обладнанні. Локальне розгортання пропонує такі переваги, як доступність в автономному режимі, потенційно нижча затримка (залежно від обладнання та затримки хмари) та конфіденційність даних.

Моделі Gemma 3 демонструють сильні сторони в таких областях, як математика, міркування та можливості чат-ботів. Моделі Gemma 3 спеціально навчені для покращення математичних здібностей, міркування та чату. Модель з 27 мільярдами параметрів навіть конкурує з більшими моделями за продуктивністю.

Результати бенчмарків демонструють продуктивність моделей Gemma 3 1b та 4b. Моделі показали результати MMLU близько 38.8% для 1b та 59.6% для 4b. Модель 4b демонструє значне покращення в цій категорії. Модель 4b демонструє значно краще розуміння ширшого кола предметів порівняно з моделлю 1b, що може призвести до кращої точності вибору ключових слів. Результат HumanEval для генерації коду склав близько 36% для моделі 4b. Результат моделі 1b у цих фрагментах явно не згадується.

Мінімальні та рекомендовані системні вимоги для локального запуску моделей Gemma 3 1b та 4b включають достатній обсяг оперативної пам’яті (RAM). Для моделі 1b рекомендується щонайменше 4 ГБ ОЗП , тоді як для моделі 4b знадобиться більший обсяг ОЗП, хоча точні цифри у наданих фрагментах не вказані. Обидві моделі призначені для роботи на стандартних ноутбуках та настільних комп’ютерах.

Для оптимальної продуктивності моделі 4b рекомендується використовувати графічні процесори NVIDIA. Модель 1b може працювати ефективно і без дискретної відеокарти, використовуючи ресурси центрального процесора (CPU), хоча використання GPU може покращити швидкість обробки. Моделі 1b та 4b можуть бути розгорнуті навіть на платах NVIDIA Jetson для периферійних застосувань.

Розмір моделі 1b становить близько 529 МБ, а розмір моделі 4b – близько 3.3 ГБ, що слід враховувати при визначенні обсягу необхідної пам'яті. Системні вимоги для Gemma 3 1b та 4b наведено в таблиці нижче.

Таблиця 2.4 – Системні вимоги для Gemma 3 1b та 4b

Вимога	Gemma 3 1b	Gemma 3 4b
RAM	4 ГБ+	8 ГБ+
CPU	Стандартний ноутбук/ПК	Стандартний ноутбук/ПК
GPU	Не обов'язково	NVIDIA GPU для кращої продуктивності

Для генерації голосу було обрано модель piper tts та coqui xtts. Piper TTS являє собою швидку, локальну нейронну систему синтезу мовлення. Piper TTS є проєктом з відкритим вихідним кодом, доступним на GitHub. Модель оптимізована для роботи на пристроях з обмеженими ресурсами, таких як Raspberry Pi 4, що вказує на її енергоефективність. Орієнтація Piper на локальну роботу та ефективність надає потенційну перевагу користувачам, які цінують конфіденційність і бажають запускати TTS на власному обладнанні без використання хмарних сервісів. Локальна обробка означає, що дані не передаються на зовнішні сервери, що вирішує питання конфіденційності. Ефективність дозволяє використовувати її на менш потужному обладнанні, знижуючи витрати та складність для деяких користувачів.

Модель характеризується як «блискавично швидка» та оптимізована для роботи в режимі реального часу. Piper TTS працює локально на машині користувача, гарантуючи конфіденційність та контроль над даними. Модель може працювати на невеликих пристроях, таких як Raspberry Pi, що свідчить про низькі вимоги до обчислювальних ресурсів. Це робить Piper доступним для ширшого кола користувачів з різними можливостями обладнання. Нижчі вимоги до ресурсів означають, що її можна інтегрувати у вбудовані системи або використовувати на менш потужних комп'ютерах без проблем з продуктивністю.

Piper TTS підтримує понад 40 мов та понад 100 голосових моделей Широка підтримка мов та голосів робить Piper універсальним для різноманітних потреб у створенні відеоконтенту. Користувачі можуть обирати голоси, які відповідають мові та тону їхнього відео, покращуючи залучення аудиторії. Модель дозволяє

налаштовувати голос, мову та якість синтезованого мовлення. TTS пропонує моделі з різними рівнями якості (x_low, low, medium, high) для балансування між якістю та використанням ресурсів. Це забезпечує гнучкість для користувачів з різними потребами в продуктивності та обмеженнями обладнання.

Хоча якість голосу Piper TTS є хорошою, вона може не завжди відповідати високоякісним та експресивним голосам, які пропонують деякі комерційні хмарні сервіси, такі як ElevenLabs або преміальні голоси від Google та Azure. Piper TTS іноді відзначається браком сильної емоційної виразності порівняно з моделями, розробленими спеціально для цієї мети. Деякі користувачі повідомляли про випадкові проблеми з вимовою. Для відеосценаріїв з незвичайними словами або специфічною вимовою може знадобитися ретельна перевірка та потенційна ручна корекція. Piper TTS розроблена переважно для Linux, хоча може працювати на Windows та macOS. 32-бітні системи не підтримуються. Користувачі старіших або менш поширених операційних систем можуть зіткнутися з проблемами сумісності.

Coqui XTTS v2 – це передова багатомовна модель, що дозволяє клонувати голоси з мінімальними вхідними даними (6 секунд аудіо) та генерувати природне мовлення 17 мовами.

Серед ключових особливостей Coqui XTTS v2 варто відзначити підтримку 17 мов, включаючи українську, англійську, німецьку, французьку та інші. Модель здатна не просто відтворювати текст, а й передавати емоції та стиль мовлення, зберігаючи інтонацію й темп оригінального запису. Окремою перевагою є можливість клонування голосу між мовами – це означає, що голос, записаний українською, може бути використаний для генерації тексту англійською чи будь-якою іншою підтримуваною мовою. Технічні параметри моделі, такі як частота дискретизації 24 кГц і низька затримка менше 200 мілісекунд, роблять її придатною для професійного використання.

Порівняно з іншими популярними моделями синтезу мовлення, Coqui XTTS v2 має ряд переваг. На відміну від Tacotron 2, вона пропонує готове рішення для клонування голосів без необхідності додаткового навчання. У порівнянні з

комерційними хмарними сервісами, такими як Google Cloud TTS чи Amazon Polly, XTTS v2 є безкоштовною для некомерційного використання та підтримує українську мову, що є рідкістю серед аналогічних інструментів. Однак варто враховувати, що Microsoft Azure TTS може пропонувати більш стабільну роботу для корпоративних рішень.

Головними перевагами моделі для створення відеоконтенту є висока якість звуку, швидкість генерації та гнучкість у роботі з різними мовами. Вона дозволяє швидко створювати унікальні голоси для персонажів чи закадрового озвучення без необхідності записувати професійних акторів. Відкритий код робить її доступною для широкого кола користувачів, особливо для некомерційних проєктів.

Проте існують певні обмеження, які варто враховувати. Модель поширюється під некомерційною ліцензією, що ускладнює її використання в комерційних відеопроєктах. Для комфортної роботи потрібен потужний графічний процесор, оскільки генерація на CPU може бути доволі повільною. Також якість синтезу може відрізнятися для різних мов, тому для україномовного контенту рекомендується попереднє тестування.

Для пошуку стокових матеріалів було обрано сервіс Pexels. Pexels пропонує велику колекцію безкоштовних високоякісних фото та відео для комерційного використання. Його API надає програмний доступ до цієї медіатеки, що підтверджується використанням у відомих платформах, таких як Canva, Typeform. Для роботи з API потрібен безкоштовний ключ, який легко отримати.

Використання API Pexels надає низку значних переваг для автоматизованого створення відео. До них належить доступ до великої та різноманітної бібліотеки якісних медіафайлів, що забезпечує візуальну складову для широкого спектра тем. Важливим фактором є економічна ефективність, оскільки безкоштовний доступ до контенту суттєво знижує операційні витрати. Крім того, API пропонує потужні можливості пошуку з детальною фільтрацією за параметрами (орієнтація, розмір, колір, тривалість відео), доступ до

підібраного та популярного контенту, а також наявність клієнтських бібліотек для різних мов програмування, що спрощує інтеграцію.

API Pexels демонструє високу придатність для інтеграції в програму автоматизованого створення відео. Його функціональні можливості, зокрема пошук та фільтрація медіафайлів за релевантними параметрами, безпосередньо відповідають потребам автоматизованого процесу. Стратегічне використання параметрів API дозволяє здійснювати цілеспрямований відбір візуальних матеріалів. Проте, для стабільної роботи необхідно впровадити механізми керування використанням API в межах встановлених лімітів та забезпечити коректну атрибуцію.

Таблиця 2.5 – Обрані технології та сервісі для програми

Назва технології	Мета використання	Обґрунтування вибору
Gemini 2.0 Flash	Генерація та обробка тексту, вибір ключових слів.	Швидкість та ефективність; Мультимодальність (текст, зображення, аудіо, відео) з потенціалом для майбутніх розширень (напр., аналіз відео); Велике вікно контексту (1 млн токенів) для обробки довгих текстів/відео; Висока швидкість та низька затримка; Висока продуктивність (бенчмарки MMLU-Pro); Наявність щедрого безкоштовного плану API для розробки та тестування.
Gemma 3 (1b, 4b)	Генерація та обробка тексту (як локальна альтернатива).	Локальне розгортання (конфіденційність даних, офлайн-доступ, потенційно нижча затримка); Відкриті моделі; Відносно невеликий розмір, що дозволяє запуск на споживчому обладнанні; Модель 4b є мультимодальною (текст, зображення) та має краще розуміння (вищий MMLU); Модель 1b має дуже низькі вимоги до RAM (4 ГБ+).
Piper TTS	Генерація голосу (синтез мовлення).	Швидка, локальна система TTS; Відкритий вихідний код; Оптимізована для пристроїв з обмеженими ресурсами; Локальна робота забезпечує конфіденційність даних; Низькі вимоги до обчислювальних ресурсів; Підтримка понад 40 мов та 100+ голосів; Можливість налаштування голосу, мови, якості.

Coqui XTTS v2	Генерація голосу (синтез мовлення), клонування голосу.	Передова багатомовна модель (17 мов, вкл. українську); Високоякісне клонування голосу з мінімального аудіо (6 сек); Здатність передавати емоції та стиль мовлення; Можливість крос-мовного клонування голосу; Професійні параметри (24 кГц, <200 мс затримка); Безкоштовна для некомерційного використання; Відкритий код.
Pexels API	Пошук та отримання стокових фото та відео.	Доступ до великої бібліотеки безкоштовних, високоякісних медіафайлів; Дозвіл на комерційне використання; Економічна ефективність (безкоштовний доступ); Програмний доступ через API зі зручними можливостями пошуку та фільтрації; Наявність клієнтських бібліотек для спрощення інтеграції.

Висновок до розділу 2

У цьому розділі було розроблено концептуальну модель системи автоматизації створення відео, яка базується на модульному підході та послідовному конвеєрі обробки даних. Детально описано ключові етапи роботи системи: від обробки вхідного тексту та його аналізу до генерації аудіоряду, пошуку медіаконтенту та фінального монтажу й рендерингу відео. Також було представлено обґрунтований вибір конкретних технологій, моделей штучного інтелекту та сервісів для реалізації кожного модуля. Зокрема, для обробки тексту обрано моделі Gemini 2.0 Flash та Gemma 3, для синтезу мовлення – Piper TTS та Coqui XTTS, а для пошуку медіаматеріалів – Pexels API. Обґрунтування вибору включало аналіз їх функціональних можливостей, переваг та обмежень у контексті завдань системи.

РОЗДІЛ 3
ОПИС РЕАЛІЗОВАНОЇ ПРОГРАМИ

3.1 Реалізація програми

Програму було реалізовано на мові Python. Першим етапом система приймає вхідні дані через аргументи командного рядка, обробляє їх та генерує кінцевий відеофайл з озвученням, фоновим відеорядом, а також опціонально з субтитрами та музичним супроводом.

Спочатку, за допомогою модуля argparse, відбувається обробка параметрів запуску. Система може отримувати текст для відео або безпосередньо через аргумент --text (як рядок або шлях до файлу), або генерувати його за допомогою великої мовної моделі (LLM) на основі запиту, переданого через --text-prompt. Обов'язковим є аргумент --config, що вказує шлях до файлу конфігурації (у форматі TOML), який містить налаштування, такі як параметри синтезу мовлення, шляхи до вихідних файлів, бажану якість та орієнтацію відео. Також передбачені опції для додавання субтитрів (-s), музики (-m), вибору вертикальної орієнтації відео (-v) та використання специфічних джерел фонового відео (--footage-dir, --download-clips-by-query).

Порівнюючи реалізований код із запропонованим підходом до «Модуля обробки вхідних даних», можна відзначити наступне: завантаження конфігурації та отримання тексту (як безпосередньо, так і через LLM) відповідають описаній концепції.

Спочатку виконується парсинг аргументів командного рядка. Далі визначається джерело тексту: якщо надано --text-prompt, викликається функція llm.submit_prompt для генерації тексту; в іншому випадку текст береться з аргументу --text, причому якщо цей аргумент вказує на існуючий файл, текст зчитується з нього. Після отримання тексту та завантаження конфігурації з файлу викликається функція generate_speech, яка обирає відповідний провайдер Text-to-Speech (XTTS або Piper TTS згідно конфігурації) і генерує аудіофайл з озвученням тексту. Потім функція background_video.make_background_video

створює відеоряд, використовуючи або локальні файли, або завантажуючи стокові відео (наприклад, з Pexels, що імпліментовано у відповідному модулі pexels_stock, який використовується background_video), тривалість якого підлаштовується під тривалість згенерованого аудіо. Фонові відеокліпи обрізаються або комбінуються таким чином, щоб їх загальна тривалість точно відповідала тривалості аудіодоріжки.

Після генерації базового відео (відеоряд + озвучення), система перевіряє наявність аргументів -s та -m. Якщо встановлено аргумент -s, викликається функція subtitle для генерації та накладання субтитрів на відео. Якщо встановлено аргумент -m, викликається функція music.add_musit_to_video_from_jamendo, яка додає фонову музику до відео, регулюючи її гучність згідно з конфігурацією. Кожен з цих кроків (додавання субтитрів, додавання музики) створює новий відеофайл, оновлюючи ім’я вихідного файлу. Наприкінці роботи програма виводить інформацію про авторів використаних матеріалів, шлях до створеного відеофайлу та загальний час, витрачений на генерацію. Реалізовано також обробку непередбачуваних винятків за допомогою custom_excepthook для більш інформативного виведення помилок.

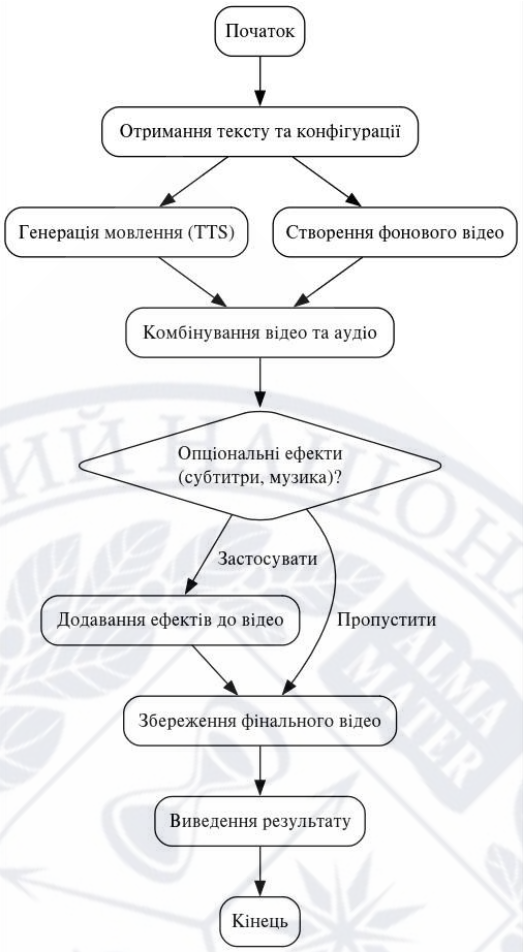

Рисунок 3.1 – Процес автоматичного створення відео (загальна схема)

Конфігураційний файл використовує формат TOML (Tom’s Obvious, Minimal Language), який є простим для читання завдяки зрозумілій семантиці. Файл складається з секцій, що позначаються квадратними дужками [section_name], та пар ключ-значення всередині кожної секції.

Секція [vidnet] є основною і містить загальні налаштування для генерації відео. Параметр output_video_dir визначає директорію, куди буде збережено фінальний відеофайл. language вказує мову для модуля синтезу мовлення (TTS), що є важливим для правильної вимови та вибору відповідної моделі голосу. Speaker задає шлях до конкретного файлу моделі голосу, яка буде використовуватися для озвучення тексту; формат цього шляху залежить від обраного TTS-провайдера (XTTS або Piper). Параметр quality контролює роздільну здатність вихідного відео (наприклад, «480p», «720p»). orientation

встановлює орієнтацію відео – горизонтальну («landscape») або вертикальну («portrait»), що впливає на компонування фонових кліпів.

Секція [pexels_stock] містить налаштування, пов'язані із завантаженням стокових відео з сервісу Pexels. Параметр output_dir визначає місце на диску для збереження завантажених кліпів, якщо програма налаштована на використання Pexels як джерела фонового відеоряду.

Секція [music] відповідає за параметри додавання фонової музики. Параметр volume дозволяє налаштувати гучність музичного супроводу відносно основного звуку (озвучення). Значення встановлюється в діапазоні від 0,0 до 1,0, де 0,1 означає 10% від максимальної гучності. Цей параметр використовується, якщо додавання музики активовано відповідним прапором командного рядка (-m).

Така структура конфігураційного файлу дозволяє гнучко керувати різними аспектами процесу генерації відео, відокремлюючи налаштування від основного коду програми.

```
[vidnet]
output_video_dir = "./out"
language = "en"
speaker = "./voices/piper_tts/en/en_US/lessac/medium/en_US-lessac-medium.onnx"
quality = "480p"
orientation = "landscape"

[pexels_stock]
output_dir = "./stock/download"

[music]
volume = 0.1
```

Рисунок 3.2 – Приклад конфігураційного файлу

Функція generate_speech аналізує конфігураційний файл, щоб визначити, який TTS-провайдер використовувати (XTTS або Piper TTS) та які параметри застосувати.

У випадку вибору XTTS, процес починається з використання бібліотеки TTS.api. Вхідний текст спочатку проходить попередню обробку функцією sanitize_text, яка видаляє зайві пробіли та символи нового рядка. Далі система

визначає доступний обчислювальний пристрій – CUDA GPU, якщо доступний, або CPU в іншому випадку. На цей пристрій завантажується та ініціалізується модель XTTS (конкретно tts_models/multilingual/multi-dataset/xtts_v2). Потім викликається метод tts.tts_to_file. Цьому методу передаються підготовлений текст, шлях до файлу-зразка голосу (speaker_wav) та код мови (language), отримані з конфігураційного файлу. XTTS обробляє весь текст за один раз і генерує єдиний аудіофайл ./speech/final.wav. На завершення, функція generate_speech повертає шлях до цього згенерованого файлу.

Якщо ж у конфігурації вказано Piper TTS, застосовується інший підхід. Оскільки Piper TTS є легкою та швидкою моделлю, вона має певні обмеження: самостійно не обробляє пунктуацію для створення природних пауз між реченнями та має обмеження на довжину тексту при прямому використанні через командний рядок. Тому було реалізовано обхідний шлях. Спочатку текст форматується функцією format_text: видаляються зайві пробіли, символи нового рядка та лапки, а знаки питання та оклику замінюються на крапки для стандартизації роздільників речень. Після форматування текст розбивається на окремі речення за крапками, а будь-які порожні рядки, що могли утворитися, видаляються за допомогою filter_sentences.

Далі система послідовно обробляє кожне речення. Для кожного речення викликається функція make_sentence_audio. Ця функція запускає зовнішній виконуваний файл Piper TTS (через subprocess), передаючи йому одне речення як вхідні дані та шлях до відповідної моделі Piper з конфігурації. В результаті для кожного речення створюється окремий WAV-файл (наприклад, ./speech/1.wav, ./speech/3.wav). Одразу після генерації аудіо для речення викликається функція make_silent_audio, яка за допомогою бібліотеки pydub створює короткий WAV-файл з тишею і ці файли тиші (наприклад, ./speech/2.wav, ./speech/4.wav) слугуватимуть паузами між реченнями.

Після того, як для всіх речень та пауз згенеровані окремі WAV-файли, в роботу вступає бібліотека moviepy. Всі ці аудіофайли завантажуються як об’єкти

AudioFileClip. Потім функція concatenate_audioclips з'єднує ці кліпи (фрагменти мовлення та паузи) у правильній послідовності в єдиний аудіотрек (AudioClip). Цей фінальний аудіотрек зберігається у файл ./speech/final.wav за допомогою методу write_audiofile. Як і у випадку з XTTS, функція generate_speech повертає шлях до цього остаточного аудіофайлу.

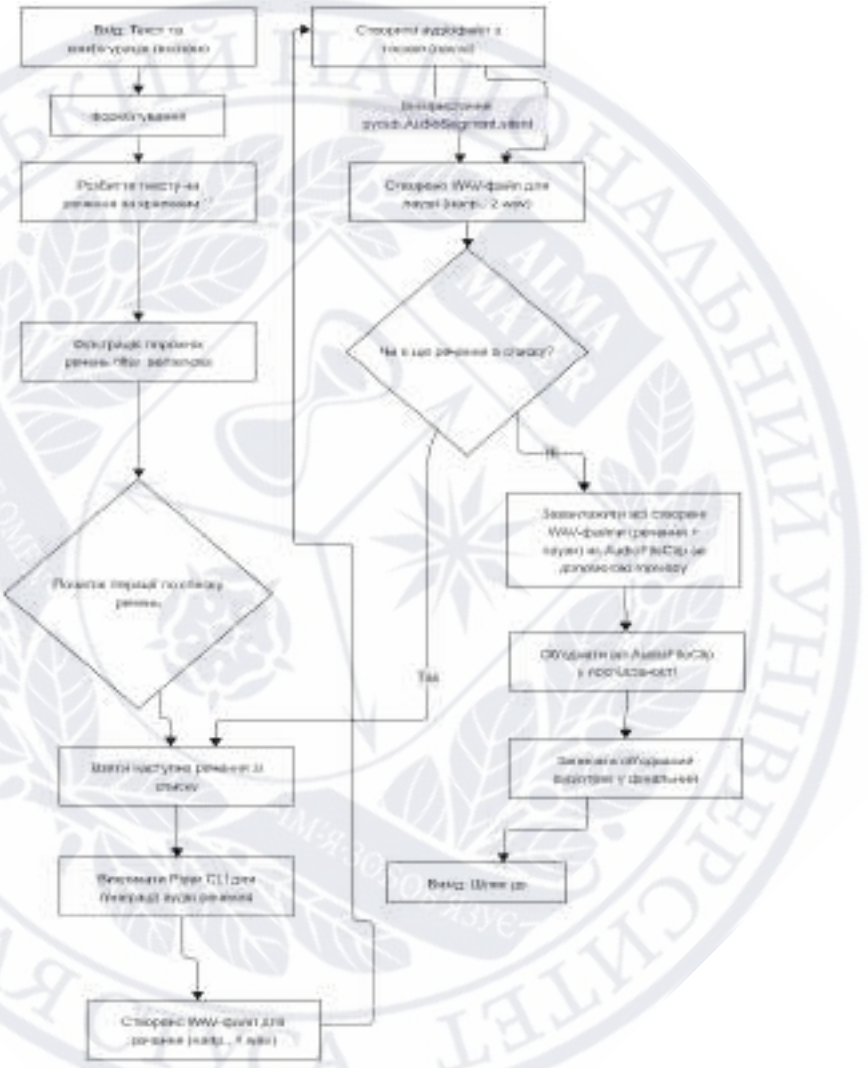

Рисунок 3.3 – Алгоритм генерації голосу з допомогою Piper TTS

Після успішного створення аудіофайлу ./speech/final.wav одним із описаних методів, його тривалість стає ключовим параметром для наступного етапу. Функція background_video.make_background_video бере цю тривалість і створює фоновий відеоряд відповідної довжини.

Ця функція реалізує кілька стратегій для створення відеоряду, вибір між якими залежить від наданих аргументів командного рядка або налаштувань за замовчуванням. Якщо користувачем вказано аргумент --footage-dir, система використовує відеофайли виключно з зазначеної локальної директорії, випадковим чином обираючи та комбінуючи їх до досягнення необхідної тривалості. У разі використання аргументу --download-clips-by-query, програма звертається до модуля pexels_stock для пошуку та завантаження стокових відео з сервісу Pexels за єдиним, наданим користувачем запитом. Завантажені кліпи потім аналогічно комбінуються.

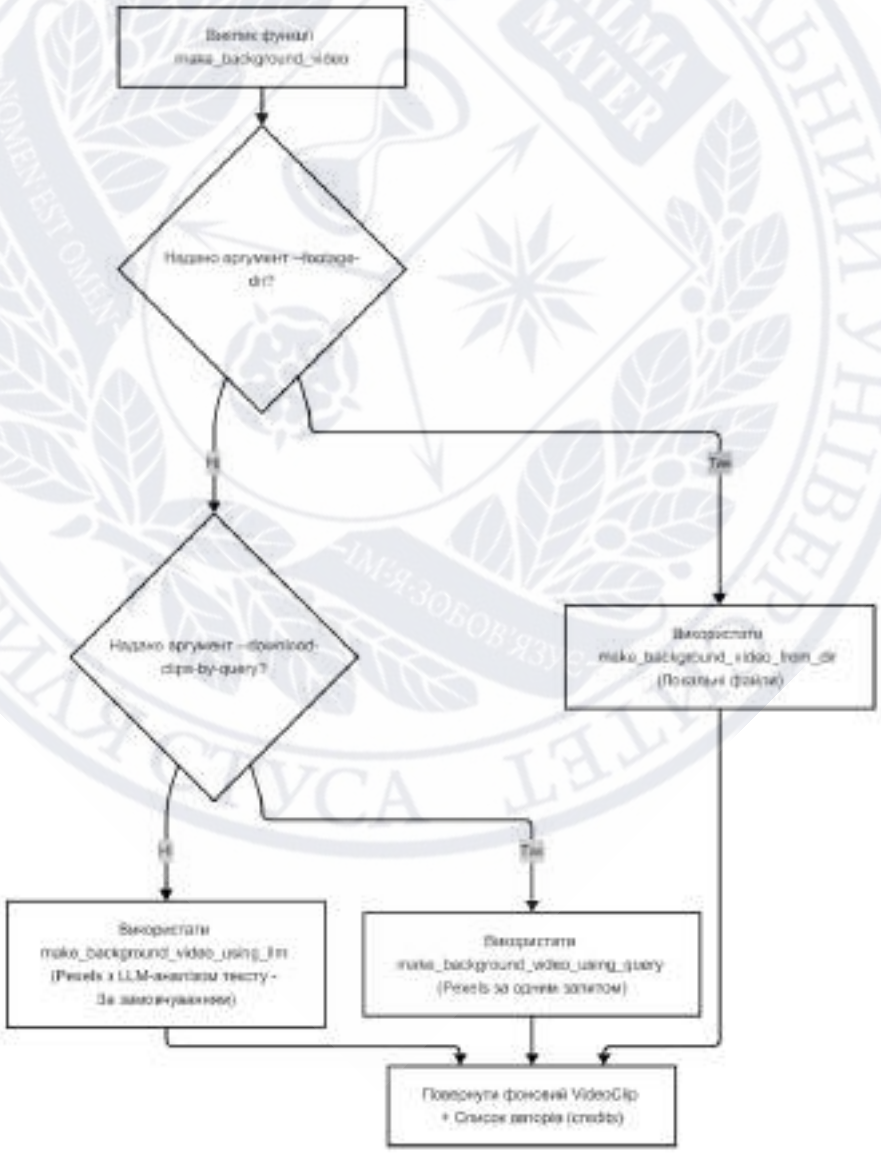

Рисунок 3.4 – Схема вибору джерела фонового відео

За відсутності цих специфічних вказівок активується найбільш розширений режим роботи, що передбачає інтелектуальний підбір відео на основі змісту згенерованого мовлення. Цей режим реалізовано у функції background_video.make_background_video_using_llm. Процес починається з транскрипції аудіофайлу ./speech/final.wav за допомогою функції subtitle.transcribe. Ця функція використовує модель розпізнавання мовлення Whisper (через бібліотеку faster-whisper) для отримання не лише тексту, але й точних часових міток для кожного слова та сегмента мовлення.

Отримані сегменти мовлення потім обробляються попарно. Для кожної пари сегментів їх текстовий вміст об’єднується. Цей об’єднаний текст слугує основою для генерації релевантного пошукового запиту для стокових відео. Система викликає велику мовну модель (в поточній реалізації використовується модель gemma3:4b, доступна через локальний сервер ollama) з інструкцією сформулювати короткий, лаконічний пошуковий запит (кілька ключових слів англійською мовою), що найкраще відображає зміст даного фрагмента тексту.

Згенерований LLM запит негайно передається функції pexels_stock.search_and_download_videos. Ця функція виконує пошук на платформі Pexels, враховуючи не лише ключові слова, але й параметри орієнтації відео (горизонтальна «landscape» чи вертикальна «portrait») та бажаної якості (наприклад, «1080p»), взяті з конфігураційного файлу. Важливим критерієм пошуку є також мінімальна тривалість відеокліпу: система шукає кліпи, тривалість яких не менша за часовий проміжок, що покривається поточною парою аудіосегментів, плюс невеликий запас (1 секунда). Це забезпечує достатню кількість матеріалу для подальшої обробки. З результатів пошуку завантажується один відеокліп.

Після завантаження відеокліп проходить етап підготовки. По-перше, за допомогою функції resize_clip (з бібліотеки moviepy) його розміри приводяться у відповідність до цільової роздільної здатності та орієнтації відео, заданих у конфігурації. По-друге, розраховується точна необхідна тривалість цього відеофрагмента – вона дорівнює різниці між часом початку першого сегмента в

парі та часом початку першого сегмента наступної пари (або кінцем усього аудіофайлу для останньої пари). Відеокліп обрізається методом subclip до цієї розрахованої тривалості. На цьому ж етапі зберігається ім’я автора завантаженого відеокліпу для подальшого формування списку авторів стокових відео (credits).

Цей процес (транскрипція -> групування сегментів -> генерація запиту LLM -> пошук і завантаження відео -> ресайз та обрізка) повторюється для кожної пари аудіосегментів. В результаті створюється послідовність відеофрагментів, кожен з яких тематично відповідає певному уривку мовлення та має точно підігнану тривалість.

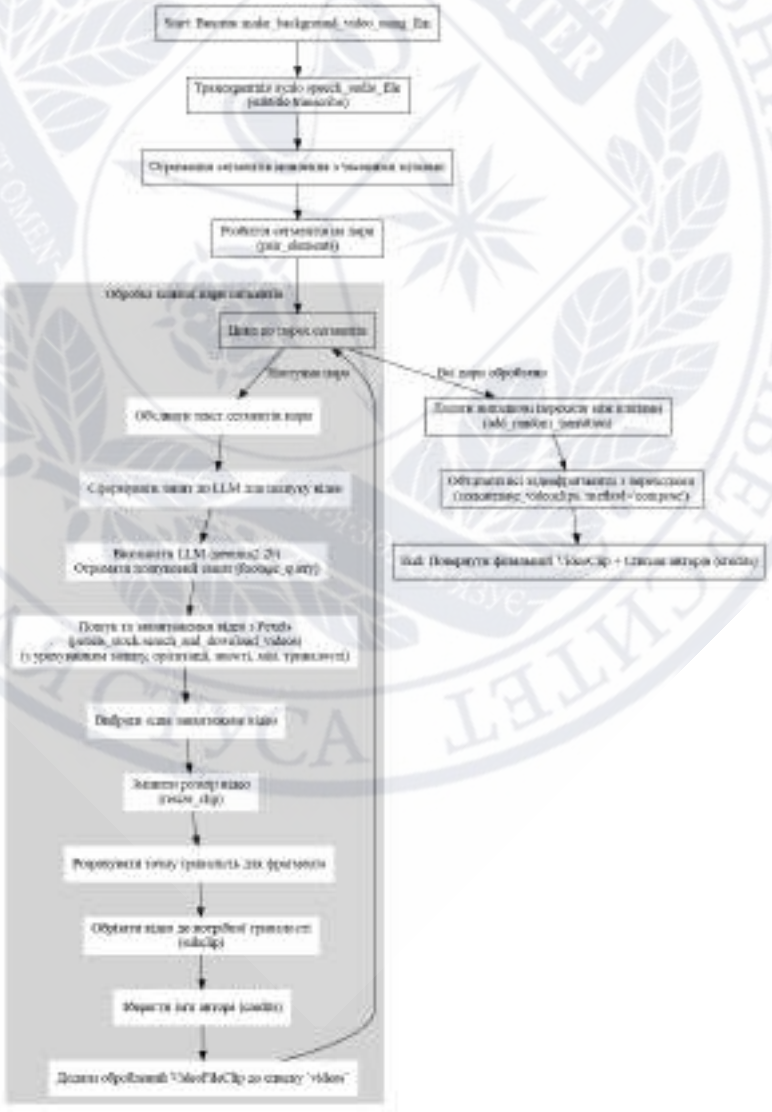

Рисунок 3.5 – Схема генерації відеоряду за допомогою LLM

Наступним кроком є об’єднання цих окремих відеофрагментів в єдиний фоновий відеоряд. Щоб переходи між різними стоковими кліпами виглядали більш природно та динамічно, між сусідніми фрагментами додаються візуальні переходи. Функція add_random_transition випадковим чином обирає один з доступних ефектів переходу (наприклад, наплив (crossfade), згасання (fade), ковзання (slide)) та застосовує його. Тривалість переходу зазвичай встановлюється невеликою (наприклад, 0,5 секунди).

Остаточне з’єднання всіх відеофрагментів, включно з переходами між ними, виконується за допомогою функції concatenate_videoclips з бібліотеки moviepy, використовуючи метод композиції (method="compose"). Результатом роботи функції background_video.make_background_video є єдиний об’єкт VideoClip, що представляє собою фоновий відеоряд, тривалість якого точно відповідає тривалості згенерованого аудіофайлу ./speech/final.wav. Разом з відеокліпом функція повертає список credits, що містить імена авторів усіх використаних стокових відео. Цей відеоряд та аудіофайл потім використовуються для створення основного відеофайлу, до якого, за потреби, додаються субтитри та музика.

Наступним потенційним кроком є додавання субтитрів до відео. Ця операція є опціональною і активується лише тоді, коли користувач вказав прапорець -s або --subtitle при запуску програми. Якщо цей прапорець встановлено, основна функція make_video викликає спеціалізовану функцію subtitle.subtitle, передаючи їй шлях до щойно створеного відеофайлу (наприклад, ./output/відео_назва.mp4) та, опціонально, бажаний розмір моделі Whisper з конфігураційного файлу (config["vidnet"].get("subtitles_model_size", "tiny")).

Процес генерації та додавання субтитрів, інкапсульований у функції subtitle.subtitle, складається з кількох послідовних етапів. Перш за все, необхідно отримати чисту аудіодоріжку з відеофайлу, оскільки моделі розпізнавання мовлення працюють саме з аудіоданими. Цю задачу виконує функція extract_aduio. Вона використовує бібліотеку ffmpeg-python для взаємодії з

інструментом FFmpeg. Створюється команда FFmpeg, яка приймає на вхід відеофайл, ігнорує відеопотік та вилучає аудіопотік, зберігаючи його в окремий файл у форматі WAV (наприклад, ./subtitle/extracted_audio.wav). Цей файл перезаписується при кожному запуску, щоб уникнути накопичення тимчасових даних. Функція повертає шлях до вилученого аудіофайлу.

Вилучений аудіофайл передається функції transcribe. Ця функція є серцем процесу генерації субтитрів. Вона задіює модель розпізнавання мовлення Whisper, що реалізована через високоефективну бібліотеку faster-whisper. Спочатку ініціалізується модель Whisper (WhisperModel) із зазначеним розміром (від tiny до large, що впливає на точність та ресурсоємність) та налаштуванням обчислювального пристрою (WHISPER_DEVICE, зазвичай "cpu" або "cuda"). Далі викликається метод model.transcribe, якому передається шлях до аудіофайлу та важливий параметр word_timestamps=True. Цей параметр вказує моделі на необхідність визначення часових міток не лише для цілих сегментів фраз, але й для кожного окремого слова. Хоча поточна реалізація використовує лише мітки сегментів для SRT, наявність міток слів відкриває можливості для майбутніх розширень (наприклад, створення субтитрів у інших форматах). Модель автоматично визначає мову аудіо (language) та повертає список об’єктів-сегментів (segments). Кожен сегмент містить розпізнаний текст (segment.text), час початку (segment.start) та час кінця (segment.end) у секундах, а також список слів з їхніми власними часовими мітками (завдяки word_timestamps=True).

На основі отриманих сегментів мови та їх часових міток викликається функція generate_subtitle_file_rst. Вона форматує дані у стандартний файл субтитрів формату SubRip Text (.srt). Для кожного сегмента створюється запис у файлі .srt, який містить:

- Порядковий номер субтитру.
- Часові мітки початку та кінця відображення субтитру у форматі HH:MM:SS,ms --> HH:MM:SS,ms. Функція format_time1 використовується

для перетворення секунд (з плаваючою комою), отриманих від Whisper, у цей стандартний формат.

- Текст самого субтитру (вміст segment.text).
- Порожній рядок, що відокремлює поточний запис від наступного. Згенерований текстовий контент записується у файл з іменем, що включає код мови (наприклад, ./subtitle/sub.uk.srt). Функція повертає шлях до цього .srt файлу.

Формат SubRip Text (.srt) є одним з найпростіших і найпоширеніших форматів текстових субтитрів. Його структура чітко визначена і легко читається як людиною, так і програмою. Розглянемо приклад з наданого файлу sub.uk.srt.txt:

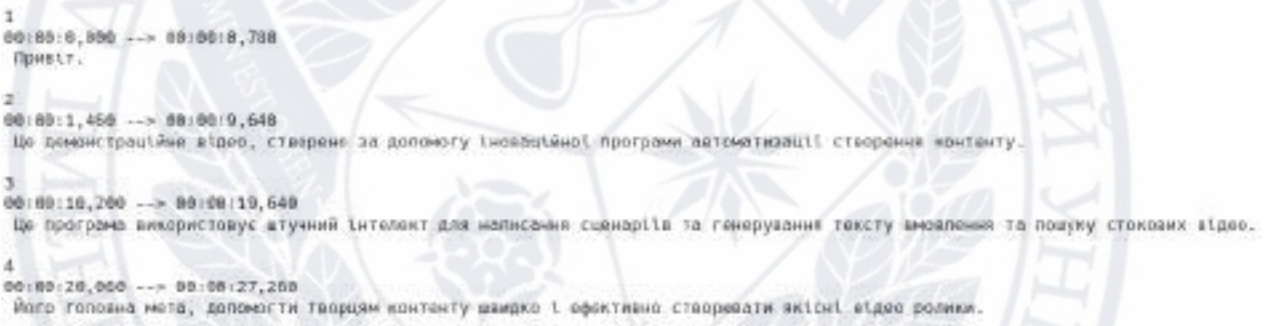

Рисунок 3.6 – Приклад .rst файлу

Цифра 2 це порядковий номер (індекс) блоку субтитрів, починаючи з 1. 00:00:1,460 --> 00:00:9,640: Це часовий код, який вказує, коли субтитр повинен з'явитися на екрані і коли зникнути. Формат: години:хвилини:секунди,мілісекунди --> години:хвилини:секунди,мілісекунди. Це демонстраційне відео...: Це сам текст субтитру, який буде відображено. Текст може займати один або кілька рядків. Порожній рядок: Слугує роздільником між блоками субтитрів.

Перевагами формату .srt є простота, він дуже легкий для розуміння, редагування вручну за допомогою будь-якого текстового редактора. Він також підтримується переважною більшістю медіаплеєрів, відеоредакторів та онлайн-

платформ. RST файли мають невеликий розмір, оскільки містять лише текст та часові коди.

Однак цей формат має кілька недоліків: По перше це обмежене форматування, цей формат не підтримує розширені стилі тексту, такі як вибір шрифту, розміру, кольору, складне позиціонування або анімацію безпосередньо у форматі. Стилізація залежить від можливостей плеєра або вимагає «впалювання» з певними стилями на етапі кодування відео.

Останнім кроком є інтеграція згенерованого .srt файлу безпосередньо у відеопотік. Функція add_subtitle_to_video знову використовує ffmpeg-python. Вона приймає шлях до оригінального відео та шлях до .srt файлу. Формується команда FFmpeg, яка використовує відеофільтр (vf) subtitles=. Цей фільтр читає вказаний .srt файл і накладає (рендерить) текст субтитрів на відповідні кадри відео згідно з часовими мітками. Параметр force_style='Alignment=2,MarginV=50' використовується для базового налаштування зовнішнього вигляду субтитрів під час рендерингу засобами FFmpeg (в даному випадку, вирівнювання по центру внизу з вертикальним відступом). В результаті створюється *новий* відеофайл, ім’я якого зазвичай містить суфікс "-subtitled" (наприклад, ./output/відео_назва-subtitled.mp4). Оригінальний відеофайл без субтитрів залишається незмінним. Шлях до відео з субтитрами використовується для подальших кроків (наприклад, додавання музики).

Нижче представлена діаграма яка описує алгоритм створення і додавання субтитрів.

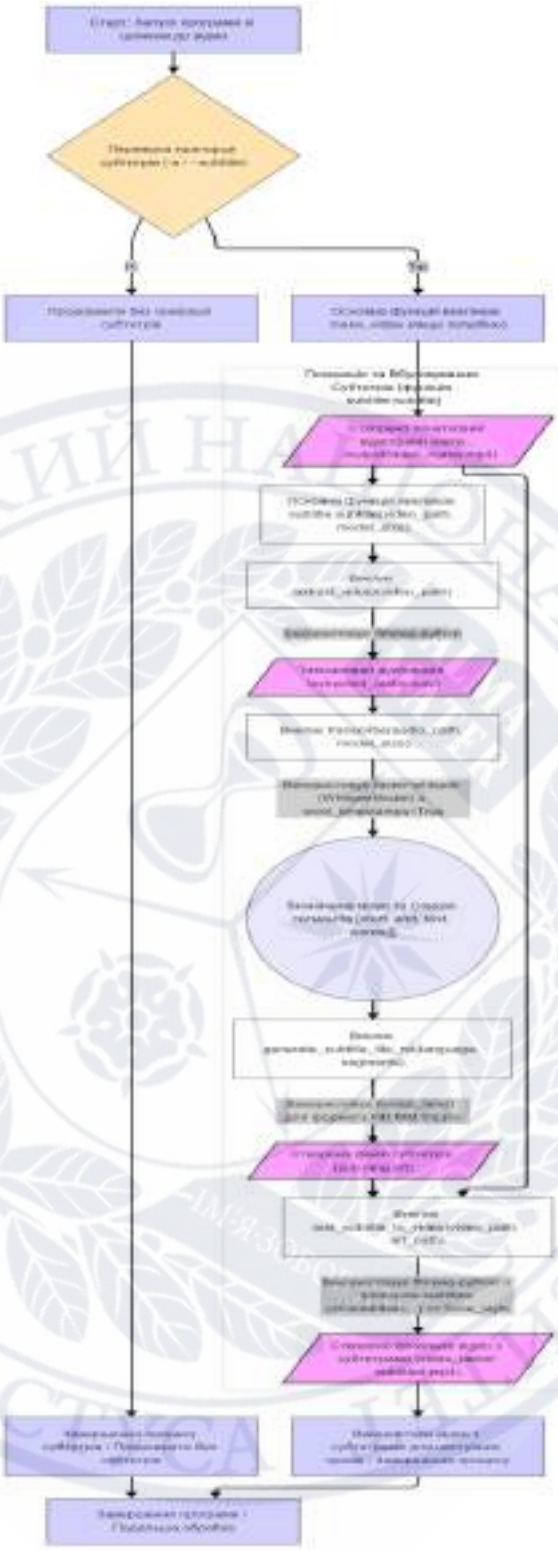

Рисунок 3.7 – Алгоритм генерації та додавання субтитрів

Реалізований алгоритм субтитрування має кілька суттєвих переваг. По-перше, ключовою перевагою є автоматизація: весь ланцюжок операцій, починаючи від вилучення аудіодоріжки і закінчуючи вбудовуванням готових

субтитрів у відео, виконується програмою без ручного втручання. Це значно економить час та зусилля користувача.

По-друге, використання сучасної моделі розпізнавання мови Whisper забезпечує високу точність як у розпізнаванні самого мовлення, так і в синхронізації згенерованого тексту з відповідними моментами в аудіодоріжці. Додаткову гнучкість надає можливість вибору розміру моделі Whisper, що дозволяє користувачеві знайти оптимальний баланс між швидкістю обробки та точністю результату залежно від його потреб та обчислювальних ресурсів. Важливою є і багатомовність моделі Whisper, яка здатна ефективно працювати з аудіо багатьма мовами, автоматично визначаючи мову без необхідності її попереднього зазначення.

По-третє, застосування поширеного формату файлів .srt для генерації субтитрів забезпечує стандартизацію та максимальну сумісність з більшістю медіаплеєрів та платформ. Метод інтеграції, при якому субтитри «впалюються» безпосередньо у відеопотік, гарантує їхнє відображення на будь-якому пристрої або плеєрі, незалежно від того, чи підтримує він зовнішні файли субтитрів.

Таким чином, якщо користувач активує опцію субтитрів, програма автоматично генерує точні, синхронізовані субтитри у стандартному форматі та інтегрує їх у фінальний відеофайл, підвищуючи доступність та зручність перегляду контенту.

Наступним, також опціональним, етапом процесу генерації відео є додавання фонової музики, яке активується за допомогою прапорця -m або --music під час запуску програми. Ця функціональність дозволяє збагатити відеоряд атмосферним музичним супроводом, підвищуючи його емоційну привабливість та професійний вигляд. Якщо ця опція увімкнена, основна функція make_video, що міститься у файлі vidnet.py, після потенційного етапу додавання субтитрів, ініціює процес додавання музичного супроводу. Для цього вона викликає спеціалізовану функцію, основною з яких є music.add_musit_to_video_from_jamendo, передаючи їй поточний об’єкт

відеокліпу (отриманий після генерації мовлення та можливого додавання субтитрів) та параметри, визначені в конфігурації.

Функція music.add_musit_to_video_from_jamendo, визначена у модулі music.py, є центральним елементом інтеграції музики з зовнішнього джерела – платформи Jamendo. Вона приймає як основні аргументи:

- Об’єкт VideoFileClip бібліотеки MoviePy, що представляє відео, до якого необхідно додати музику.
- Рядок query, що містить ключові слова для пошуку музики на Jamendo (наприклад, "lofi", "cinematic", "upbeat"). За замовчуванням встановлено значення 'lofi', але користувач може вказати інший запит через конфігурацію або, потенційно, через параметри запуску.
- Значення music_volume, що визначає відносну гучність фонової музики порівняно з оригінальною аудіодоріжкою (голосом диктора). Це значення, зазвичай десятковий дріб (наприклад, 0.1 для 10% гучності), отримується з конфігураційного файлу (config["music"]["volume"]).

Ключовим першим кроком у функції add_musit_to_video_from_jamendo є отримання відповідного аудіотреку з Jamendo. Цю задачу виконує функція jamendo.get_track(query), яка інкапсулює всю логіку взаємодії з Jamendo API. Процес, реалізований у jamendo.py, складається з таких етапів:

Першим кроком іде сам пошук треків: викликається функція search_tracks(query, limit=10). Вона формує HTTP GET-запит до кінцевої точки Jamendo API (https://api.jamendo.com/v3.0/tracks/). У параметри запиту включаються:

- client_id: Ідентифікатор додатка для доступу до API
- format: Бажаний формат відповіді (json).
- limit: Максимальна кількість треків у результатах пошуку (за замовчуванням 10).
- search: Ключові слова для пошуку, отримані з параметра query.

- audiodownload_allowed: Важливий фільтр, встановлений у true, який гарантує, що у результатах будуть лише ті треки, для яких автори дозволили безкоштовне завантаження аудіофайлу.

Якщо запит успішний (статус код 200), функція повертає список словників, де кожен словник містить метадані одного знайденого треку (назва, ім'я виконавця, URL для завантаження тощо). В іншому випадку виводиться повідомлення про помилку.

Рисунок 3.8 – Алгоритму пошуку треків

Наступним етапом іде вибір треку, де функція get_track перевіряє, чи були знайдені треки. Якщо список результатів порожній, виводиться повідомлення і функція завершується безрезультатно (повертає None). Якщо треки знайдені, для

наочності (хоча в автоматичному режимі це може бути лише логуванням) виводиться нумерований список знайдених композицій. Далі, з отриманого списку треків випадковим чином обирається один трек за допомогою random.choice(tracks). Цей елемент випадковості дозволяє урізноманітнити музичний супровід для різних відео, навіть якщо використовується той самий пошуковий запит.

Обраний трек передається функції download_track(track). Ця функція виконує безпосереднє завантаження аудіофайлу. Спочатку вона вилучає необхідні метадані: назву треку (track["name"]), ім’я виконавця (track["artist_name"]) та пряме посилання на завантаження аудіо (track["audiodownload"]).

Потім вона формує безпечне ім’я файлу для збереження на локальному диску. Використовується шаблон ./music/{artist_name} - {track_name}.mp3. Перед формуванням імені назва треку проходить через функцію sanitize_filename(name), яка замінює символи, неприпустимі в іменах файлів у більшості операційних систем ([\\/*?:"<>|]), на символ підкреслення (_), щоб уникнути помилок під час збереження. Також перевіряється існування директорії ./music/ і, якщо вона відсутня, створюється за допомогою os.mkdir().

Далі здійснюється HTTP GET-запит за отриманим download_url за допомогою бібліотеки requests, використовуючи потокову передачу (stream=True) для ефективної роботи з потенційно великими аудіофайлами. Якщо завантаження успішне (статус код 200), вміст відповіді записується у створений файл частинами (chunk_size=8192), що запобігає надмірному використанню пам’яті.

Потім функція повертає повний шлях до завантаженого файлу (наприклад, ./music/Artist Name - Track_Title.mp3). У разі невдачі завантаження виводиться повідомлення про помилку і функція повертає None.

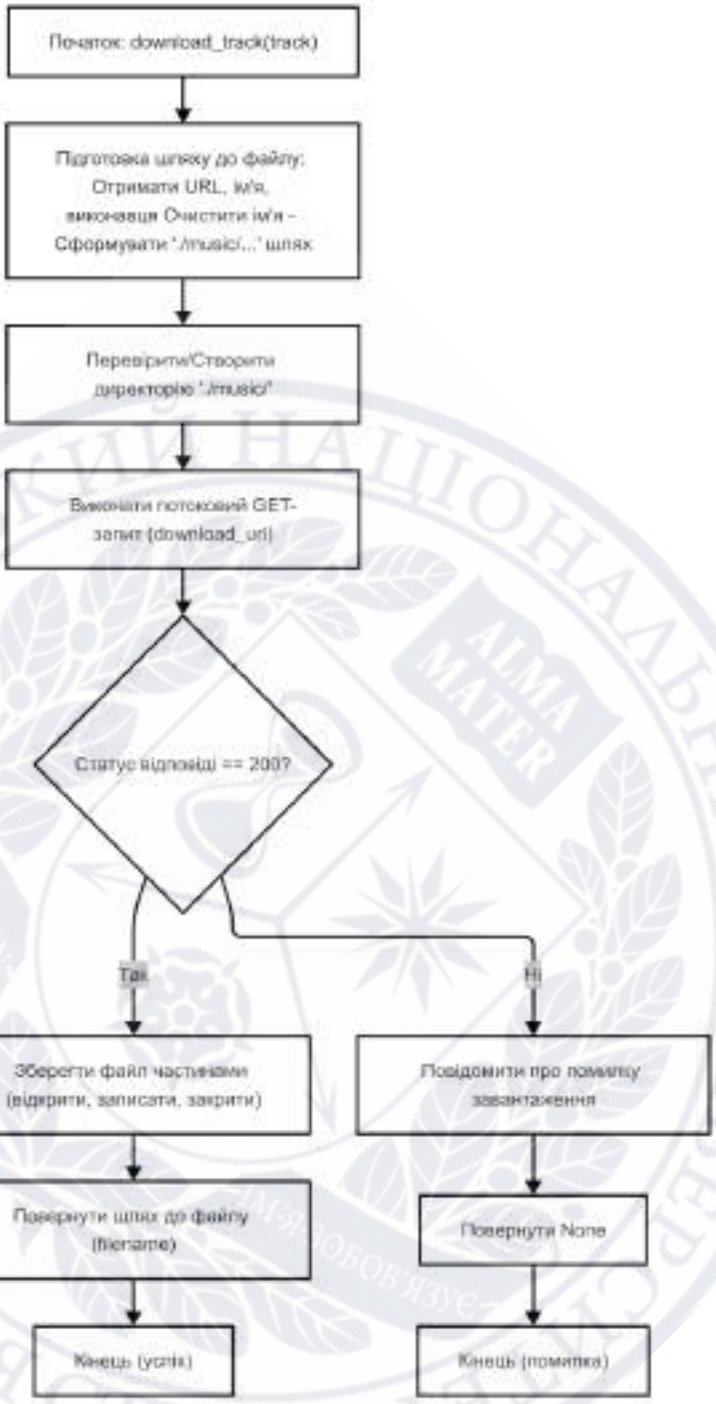

Рисунок 3.9 – Алгоритму завантаження треку

Останнім етапом іде обробка помилок завантаження та повернення результату де функція get_track містить простий механізм повторної спроби. Якщо перша спроба завантаження (download_track) завершилася невдало (повернула None), робиться ще одна спроба: знову випадково вибирається трек зі списку знайдених (track = random.choice(tracks)) і викликається download_track.

Це підвищує надійність процесу на випадок тимчасових проблем з доступністю конкретного треку. Успішна функція get_track повертає кортеж, що містить шлях до завантаженого локального аудіофайлу (file) та ім’я виконавця (track['artist_name']), яке використовується для формування списку авторів (credits).

Після успішного отримання шляху до локального музичного файлу (music_file_path) та інформації для зазначення авторства (credits), функція add_musit_to_video_from_jamendo продовжує обробку. Спочатку завантажений аудіофайл відкривається як об’єкт AudioFileClip бібліотеки MoviePy за допомогою music = AudioFileClip(music_file_path). Далі перевіряється тривалість музичного треку (music.duration) порівняно з тривалістю основного відеокліпу (video.duration). Якщо музичний трек виявляється довшим за відео, він обрізається до відповідної тривалості за допомогою методу subclip(0, video.duration), що гарантує завершення музичного супроводу одночасно з відеорядом.

Наступним кроком є налаштування гучності музичного треку відповідно до значення music_volume, отриманого з конфігурації; це здійснюється методом volumex(music_volume), який множить амплітуду аудіосигналу на вказаний коефіцієнт, дозволяючи зробити музику фоновою і не заглушати основну аудіодоріжку (голос диктора).

Потім створюється композитна аудіодоріжка шляхом змішування оригінальної аудіодоріжки відео (video.audio, що містить згенерований голос) та підготовленого музичного кліпу (music) за допомогою CompositeAudioClip([video.audio, music]). Ця композитна аудіодоріжка встановлюється як нова аудіодоріжка для відеокліпу методом video.set_audio(composite_audio). В результаті об’єкт video тепер містить як візуальний ряд, так і збалансовану комбінацію голосу та фонової музики. Насамкінець, функція повертає оновлений об'єкт VideoFileClip разом із рядком, що містить інформацію про автора музики (credits), отриману раніше від jamendo.get_track.

Повернувшись до функції make_video в vidnet.py, отриманий відеокліп з доданою музикою (music_video) використовується для фінального запису у файл. Інформація про автора музики (music_credits) додається до загального списку авторів (credits), який може містити також інформацію про джерела відеоматеріалів. Ім’я вихідного файлу часто модифікується додаванням суфікса "-music" (наприклад, ./output/відео_назва-subtitled-music.mp4), щоб позначити наявність музичного супроводу. Фінальний відеофайл записується на диск за допомогою music_video.write_videofile(...) з тими ж параметрами кодування, що й раніше.

Варто зазначити, що в модулі music.py також присутня альтернативна функція add_music_to_video_from_folder. Вона реалізує схожу логіку, але замість звернення до Jamendo API, вибирає випадковий музичний файл (.mp3 або .wav) з локальної директорії, вказаної користувачем (music_dir). Це надає користувачеві можливість використовувати власну бібліотеку музики замість пошуку в Інтернеті. Однак, основний потік виконання, продемонстрований у vidnet.py, використовує інтеграцію з Jamendo.

Отже, механізм додавання музики в програмі характеризується високим ступенем автоматизації, можливістю використання великої бази безкоштовної музики з Jamendo (з урахуванням ліцензійних умов платформи та необхідності зазначення авторства), гнучким налаштуванням гучності фонового супроводу та безшовною інтеграцією з основним відеорядом і голосовою доріжкою за допомогою можливостей бібліотеки MoviePy. Випадковий вибір треку з результатів пошуку додає елемент варіативності, а обов'язкове збирання інформації про авторів (credits) підкреслює важливість дотримання авторських прав навіть при використанні умовно-безкоштовних ресурсів. Весь процес відбувається автоматично, якщо користувач активував відповідну опцію при запуску програми.

Описаний алгоритм додавання музики має кілька значних переваг. Найважливішою є повна автоматизація процесу, що вимагає від користувача лише вказати відповідний аргумент, після чого програма самостійно виконує

пошук, завантаження, обробку та інтеграцію музичного супроводу, значно заощаджуючи час та зусилля. Інтеграція з Jamendo API відкриває доступ до великої бібліотеки переважно безкоштовної музики, причому використання фільтру audiodownload_allowed=true гарантує роботу з треками, завантаження яких дозволено авторами. Елемент варіативності вноситься завдяки випадковому вибору треку з результатів пошуку, що дозволяє урізноманітнити музичний супровід навіть при повторному використанні того самого запиту.

Важливою функцією є можливість базового налаштування гучності музики через конфігураційний файл, що допомагає збалансувати її з основним голосом диктора. Автоматичне обрізання тривалості музичного треку до тривалості відео забезпечує акуратне завершення аудіоряду. Надійність підвищується завдяки санітизації імен файлів для уникнення помилок збереження та простому механізму повторної спроби завантаження у разі невдачі. Також алгоритм намагається врахувати необхідність зазначення авторства, зберігаючи ім'я виконавця, а наявність альтернативної функції для роботи з локальними файлами надає додаткову гнучкість користувачам із власними музичними колекціями.

3.2 Демонстрація функціонування програми

Після успішного завершення всіх попередніх етапів – обробки або генерації тексту, синтезу мовлення, пошуку та завантаження релевантних стокових відеоматеріалів та фонової музики, а також їх автоматизованого монтажу – розроблена програма vidnet.py створює кінцевий продукт: готовий відеофайл.

Як приклад роботи програми, розглянемо відео, згенероване на тему фінансів англійською мовою, з використанням параметрів, визначених у конфігураційному файлі config2.toml.

Генерація тексту ініціюється за допомогою аргументу командного рядка --text-prompt. У цьому прикладі користувач вводить запит українською мовою, вказуючи бажану тему та мову фінального відео:

./vidnet.py --text-prompt "Відео про фінанси на англійській мові" --config ./config2.toml -v -s -m

Після нетривалого очікування, поки LLM обробляє запит, програма виводить у консоль згенерований текст сценарію. В даному випадку результат виглядає так:

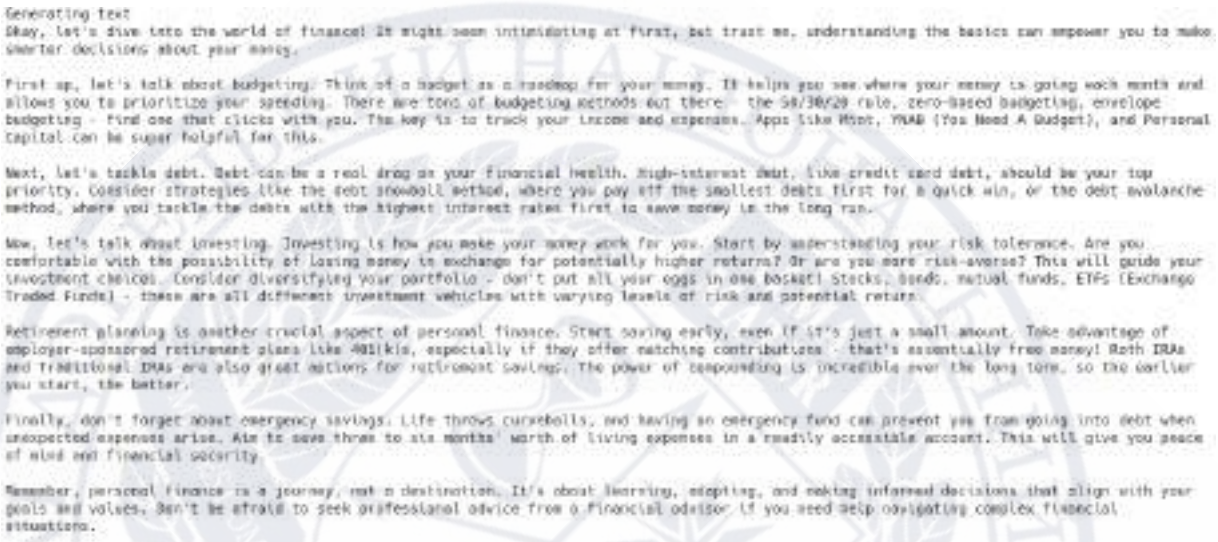

Рисунок 3.10 – Уривок згенерованого тексту на тему фінансів

Аналізуючи результат, можна відзначити, що текст повністю відповідає запиту користувача: він написаний англійською мовою та розкриває основи теми «фінанси». Структура тексту логічна: є вступ, основні пункти (бюджетування, борг, інвестування, пенсія, резервний фонд) та висновок. Стиль тексту – інформативний та доступний, підходить для формату відео. Обсяг тексту достатній для створення короткого відео (орієнтовно 1-2 хвилини, залежно від темпу мовлення). Таким чином, етап генерації тексту за допомогою LLM успішно виконано. Отриманий англомовний текст про фінанси тепер готовий для передачі на наступні етапи обробки: синтез мовлення та підбір відеоматеріалів.

Після цього іде генерація аудіо голосу. Нижче зображено форму звукової хвилі (waveform) та спектрограму запису голосу.

Форма хвилі демонструє зміни амплітуди сигналу з часом, тоді як спектрограма відображає розподіл частот у часі, дозволяючи візуалізувати енергетичну насиченість різних частот у голосі.

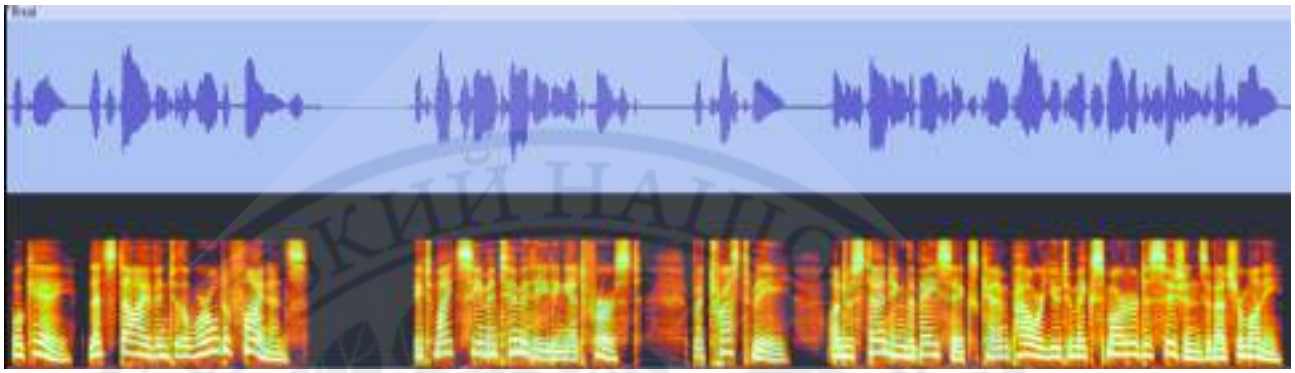

Рисунок 3.11 – Форма звукової хвилі та спектограма аудіо запису згенерованого голосу

Після генерації аудіодоріжки ключовим етапом для подальшого монтажу є точна синхронізація тексту сценарію з вимовленим аудіо. Програма vidnet.py виконує це завдання, розбиваючи оригінальний текст на смислові фрагменти та визначаючи точні часові мітки початку та кінця кожного фрагмента в згенерованому аудіофайлі.

Цей процес є критично важливим, оскільки саме на основі цих часових міток та змісту кожного текстового фрагмента програма буде підбирати відповідні стокові відеоматеріали та визначати їхню тривалість у фінальному відео. Для нашого прикладу з відео про фінанси, вивід має наступний вигляд:

Рисунок 3.12 – Розбиття тексту на фрагменти під які будуть вставлені стокові матеріали

Як видно з наведеного виводу, програма успішно розділила весь текст на окремі речення або логічно завершені фрази. Кожному такому фрагменту присвоєно часовий діапазон у секундах (з точністю до сотих), що вказує на момент початку та завершення його вимови у згенерованому аудіофайлі. Наприклад, вступна фраза «OK, let's dive into the world of finance.» звучить з 0.00 до 2.44 секунди, що дає тривалість 2.44 секунди для цього сегменту. Фрагмент про бюджетування «Think of a budget as a roadmap for your money.» вимовляється з 14.14s до 16.44s (тривалість 2.30 секунди).

Невеликі розриви у часі між послідовними фрагментами (наприклад, між 2.44s та 3.42s, або між 10.48s та 11.26s) відповідають природним паузам у синтезованому мовленні, що додає реалістичності.

Таким чином, етап розбиття тексту на синхронізовані з аудіо фрагменти створює структуровану основу для автоматичного та контекстуально

релевантного візуального оформлення відеоряду, що є однією з ключових функцій розробленої програми.

Маючи текст, розбитий на сегменти з точними часовими мітками, програма переходить до наступного важливого етапу – пошуку та завантаження релевантних відеоматеріалів для візуалізації кожного сегменту. Цей процес відбувається ітеративно для кожного фрагменту сценарію.

Для кожного сегменту програма формує пошуковий запит. LLM може виділяти ключові слова, генерувати короткий опис сцени або перефразовувати текст для більш ефективного пошуку на стокових платформах.

Сформований запит надсилається до API сервісів Pexels. Програма шукає відео, яке найкраще відповідає запиту та має достатню тривалість, щоб покрити часовий проміжок відповідного аудіосегменту.

Розглянемо вивід програми для перших кількох сегментів нашого прикладу відео про фінанси:

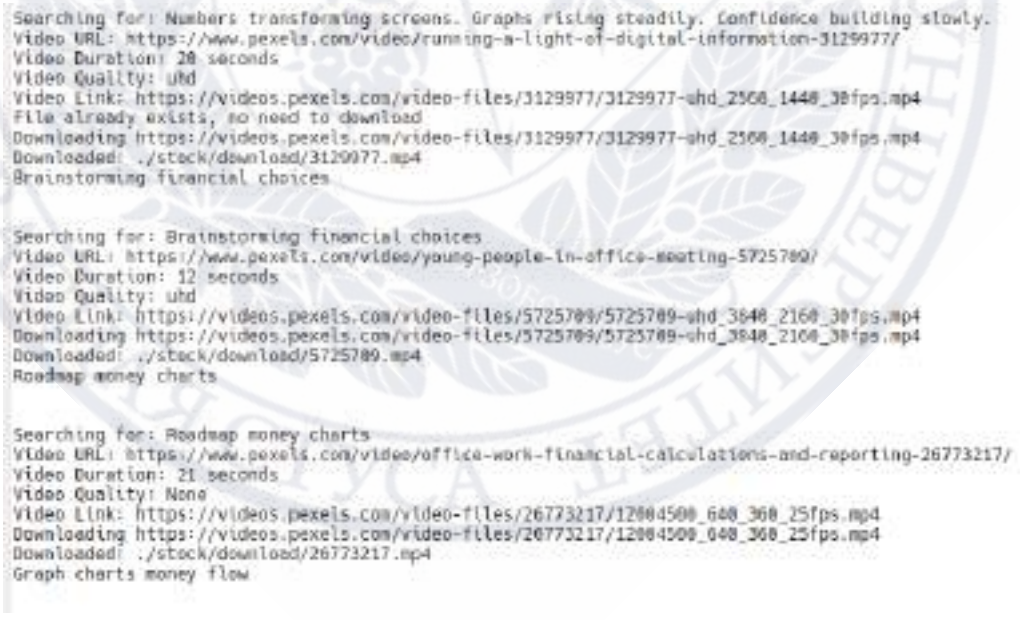

Рисунок 3.13 – Пошук стокового відео з Pexels для фрагменту відео

Для першого сегменту програма знайшла відео на Pexels, яке має тривалість 20 секунд (більше, ніж необхідні 2.44 секунди), доступне в UHD

якості та має пряме посилання для завантаження. У даному випадку файл вже існував локально (File already exists), тому повторне завантаження не відбулося.

Для другого сегменту (що охоплює [3.42s -> 6.60s] It might seem intimidating at first, but trust me, та [6.90s -> 9.74s] understanding the basics can empower you to make smarter decisions [9.74s -> 10.48s] about your money.) програма використала пошуковий запит "Brainstorming financial choices". Тут було знайдено 12-секундне відео, яке відповідає темі обговорення фінансових рішень. Оскільки цього файлу локально не було, програма завантажила його за вказаним посиланням (Video Link) та зберегла у директорії ./stock/download/ під іменем 5725709.mp4.

Аналогічно для третього сегменту, що стосується бюджетування ([11.26s -> 13.20s] First up, let's talk about budgeting. та [14.14s -> 16.44s] Think of a budget as a roadmap for your money.), було використано запит "Roadmap money charts". Тут програма знайшла 21-секундне відео, що візуалізує фінансові розрахунки. Воно було успішно завантажено.

Рисунок 3.14 – Знайдене стокове відео для другого сегменту

Після успішного підбору та завантаження всіх необхідних відеоматеріалів для кожного сегменту сценарію, програма vidnet.py переходить до наступного

важливого кроку – створення та підготовки субтитрів. Цей етап спрямований на підвищення доступності та зручності сприйняття відеоконтенту для ширшої аудиторії.

Програма використовує раніше визначені часові мітки початку та кінця кожного смислового фрагменту (як показано на Рисунку 3.12) та відповідний текст для автоматичної генерації файлу субтитрів у стандартному форматі .srt (SubRip Text). Цей формат є широко розповсюдженим та підтримується більшістю відеопрогравачів та платформ.

Для нашого прикладу з відео про фінанси згенерований .srt файл матиме такий вигляд:

```
1
00:00:0,000 --> 00:00:2,440
 OK, let's dive into the world of finance.

2
00:00:3,420 --> 00:00:6,600
 It might seem intimidating at first, but trust me,

3
00:00:6,900 --> 00:00:9,740
 understanding the basics can empower you to make smarter decisions

4
00:00:9,740 --> 00:00:10,480
 about your money.

5
00:00:11,260 --> 00:00:13,200
 First up, let's talk about budgeting.
```

Рисунок 3.15 – Уривок згенерованого SRT-файлу для відео про фінанси

Як видно з рис. 3.15, кожен блок субтитрів містить порядковий номер субтитру, часовий код початку та кінця відображення субтитру у форматі години:хвилини:секунди,мілісекунди. Ці часові мітки точно відповідають тим, що були визначені на етапі синхронізації аудіо та тексту. Текст самого субтитру, який буде показаний на екрані протягом зазначеного часового проміжку.

Згенерований .srt файл зберігається локально і буде використаний для накладання текстових субтитрів безпосередньо на відеоряд. Це реалізується за

допомогою ffmpeg, що дозволяє візуально відобразити вимовлений текст синхронно з аудіо та відео. Наявність субтитрів значно покращує сприйняття інформації, особливо для людей з вадами слуху, для перегляду відео без звуку, або для кращого розуміння контенту, вимовленого іноземною мовою.

Після успішної генерації файлу субтитрів, програма vidnet.py переходить до наступного важливого етапу – пошуку та завантаження фонової музики. Цей етап є невід'ємною частиною створення атмосфери та емоційного забарвлення кінцевого відеопродукту.

Програма використовує API сервісів Pexels та Pixabay для пошуку безкоштовних музичних треків.

У розглянутому прикладі, після завершення роботи з відеофрагментами, програма ініціювала пошук фонової музики. Пошук відбувався з урахуванням налаштувань, що передбачають спокійний, ненав'язливий супровід, як-от популярний для навчального та фінансового контенту жанр "lofi". Консольний вивід програми, що демонструє результати пошуку, виглядає наступним чином:

```
Found tracks:
1. LoFi Theory - lofi
2. Davide Bozzaro - Lofi Love - Ambient Lofi Vibe
3. Nightingale Lofi - Far Away | Non Copyright Lofi
4. Pumpupthemind - Lofi
5. The_Mountain - LoFi Chill
6. Lysergic Tempo - Master Beat inside Lofi
7. leberch - Dreamy LoFi
8. Nightingale Lofi - Follow Me | Non Copyright Lofi
9. TaigaSoundProd - LoFi HipHop Intro 69
10. An Ki - lofi hiphop chill
Downloading: ./music/Nightingale Lofi - Far Away _ Non Copyright Lofi.mp3
Saved to: ./music/Nightingale Lofi - Far Away _ Non Copyright Lofi.mp3
```

Рисунок 3.16 – Пошук музики для відео

Як видно з наведеного рисунку, програма успішно знайшла список з десяти музичних треків, що відповідають потенційним критеріям пошуку. З цього списку програма автоматично обирає один трек для використання у якості фонового супроводу для відео. У даному конкретному випадку було обрано трек

"Nightingale Lofi - Far Away | Non Copyright Lofi". Після вибору, програма переходить до його завантаження.

Обраний музичний файл Nightingale Lofi - Far Away _ Non Copyright Lofi.mp3 успішно завантажується та зберігається у локальній директорії ./music/ для подальшого використання під час фінального етапу – монтажу відео.

Таким чином, на цьому етапі програма vidnet.py завершує збір усіх необхідних медіа: згенерований голосовий супровід, підібрані та завантажені стокові відеофрагменти для кожного сегменту сценарію, а також обрана та завантажена фонова музика. Всі ці компоненти тепер готові до передачі на фінальний етап – автоматизованого зведення у єдиний відеофайл.

Після завершення всіх операцій монтажу програма зберігає готовий відеофайл у вказаній директорії (за замовчуванням, або згідно з конфігурацією). Кінцевий продукт – це повноцінне відео, озвучене синтезованим голосом, з візуальним рядом, що відповідає контексту тексту, субтитрами та фоновою музикою.

Рисунок 3.17 – Кадри зі згенерованого відео

Важливою частиною використання стокових матеріалів є належне зазначення авторів. Програма vidnet.py автоматично збирає інформацію про авторів використаних відео та музики під час їх завантаження з Pexels та Pixabay. По завершенню генерації відео, програма виводить у консоль список авторів, чиї

роботи були використані. Для нашого прикладу з відео про фінанси, цей вивід має наступний вигляд:

```
['Pressmaster', 'Artem Podrez', 'Jakub Zerdzicki', 'Jakub Zerdzicki', 'Tima Miroshnichenko',
'Kampus Production', 'Monstera Production', 'Zak Chapman', 'Mikhail Nilov', 'Jakub Zerdzicki',
'Tima Miroshnichenko', 'Photo By: Kaboompics.com', 'Monstera Production', 'Monstera Production',
'Tima Miroshnichenko', 'Tima Miroshnichenko', 'Jakub Zerdzicki', 'Mikhail Nilov', 'RDNE Stock
project', 'RDNE Stock project', 'Photo By: Kaboompics.com', 'Tima Miroshnichenko', 'Tima
Miroshnichenko', 'Tima Miroshnichenko', 'Kampus Production', 'Nightingale Lofi']
```

Рисунок 3.18 – Виведення авторів стокових відео та музики

Висновок до розділу 3

У цьому розділі було детально описано реалізацію програми на мові Python, призначеної для автоматичного створення відеоконтенту. Розглянуто ключові етапи роботи: обробку вхідних параметрів командного рядка та конфігураційного файлу у форматі TOML, генерацію тексту (включно з використанням LLM) та синтез мовлення за допомогою XTTS або Piper TTS. Було описано механізми створення фонового відеоряду, що передбачають як використання локальних чи стокових відео (наприклад, з Pexels), так і інтелектуальний підбір кліпів за допомогою LLM на основі транскрибованого тексту. Також представлено реалізацію опціональних функцій: автоматичне генерування та накладання субтитрів з використанням моделі Whisper і додавання фонової музики з сервісу Jamendo. Наприкінці розділу продемонстровано практичне застосування програми на конкретному прикладі, ілюструючи всі етапи від генерації сценарію до отримання фінального відео та зазначення авторів використаних матеріалів.

ВИСНОВКИ

У ході виконання кваліфікаційної (бакалаврської) роботи було проведено дослідження та розробку програмного засобу для автоматизації створення відео з використанням генеративного штучного інтелекту. Результати дозволяють сформулювати наступні висновки.

Було систематизовано проблеми традиційного відеовиробництва та проаналізовано наявні комерційні платформи й відкриті моделі генеративного ШІ. Це дозволило виявити їхні функціональні можливості, переваги та недоліки, що обґрунтувало вибір технологій для розробки власного рішення, орієнтованого на гнучкість та доступність.

Досліджено можливості застосування сучасних AI-моделей для автоматичного підбору стокового відео та музики. Встановлено, що інтеграція великих мовних моделей (Gemini, Gemma) для семантичного аналізу тексту та генерації пошукових запитів у поєднанні з API стокових сервісів (Pexels) дозволяє ефективно автоматизувати підбір мультимедійного супроводу, забезпечуючи його контекстуальну відповідність.

Практичним результатом стала розробка консольного додатка на Python, що реалізує повний цикл автоматизованого створення відео. В програму інтегровано API Pexels, модель OpenAI Whisper для транскрипції та генерації субтитрів, системи синтезу мовлення Piper TTS та Coqui XTTS (з підтримкою української мови), а також бібліотеки MoviePy та FFmpeg для відеомонтажу, накладання субтитрів та музики.

Для зручності користувача підготовлено гнучку систему конфігурації на основі TOML-файлів, що дозволяє легко налаштовувати ключові параметри генерації відео (мова, голос, якість, джерела контенту, ключі API) без модифікації програмного коду.

Здійснено тестування програми, яке підтвердило її здатність успішно генерувати відеоконтент на основі текстових запитів, включаючи автоматичну

генерацію сценарію, синтез мовлення, підбір відеофрагментів, створення субтитрів та додавання фонової музики.

Виявлено переваги розробленого рішення, такі як високий ступінь автоматизації, модульність, підтримка української мови та використання доступних технологій. Водночас визначено потенційні обмеження, пов'язані з якістю синтезу мовлення, залежністю від зовнішніх API та якістю автоматично підібраного контенту.

Обґрунтовано практичну значущість створеного програмного рішення, яке значно спрощує та прискорює створення відео, роблячи його доступнішим для широкого кола користувачів. Намічено напрями подальшого вдосконалення: розширення бази AI-моделей, покращення алгоритмів аналізу та підбору контенту, оптимізація продуктивності та можлива розробка графічного інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hight C. Automation within digital videography: From the Ken Burns Effect to 'meaning-making' engines. *Studies in Documentary Film.* 2024. Vol. 8, P. 235-250. DOI: 10.1080/17503280.2014.961632

2. Than H. S. AI video editing tools. What editors want and how far is AI from delivering?. 2021. DOI: 10.48550/arXiv.2109.07809

3. Video Automation: The Complete Guide. *https://wideo.co*. URL: https://wideo.co/blog/video-automation-the-complete-guide/

4. What is Generative AI? *University of Pittsburgh*. URL: https://teaching.pitt.edu/resources/what-is-generative-ai/#:~:text=Generative%20artificial%20intelligence%20(AI)%20tools,in%20response%20to%20a%20prompt. (дата звернення: 17.02.2025)

5. Feuerriegel S., Hartmann J., Janiesch C., Zschech P. 2023. Generative AI. *Business & Information Systems Engineering.* 2023. Vol. 66, P. 111–126. DOI: https://doi.org/10.48550/arXiv.2309.07930

6. Martineau K. What is generative AI?. *IBM.* URL: https://research.ibm.com/blog/what-is-generative-AI. (дата звернення: 08.03.2025)

7. Kumar S., Sharma M., Comprehensive Review of Generative AI - From its Origins to Today and Beyond. 2024. DOI: 10.13140/RG.2.2.19420.81281

8. Bhuvanesh Role of AI Video Generation Tool Across Industries. *Steve AI.* 2024. URL: https://www.steve.ai/blog/ai-video-generation-tool-across-industries/ (дата звернення: 25.01.2025)

9. Qamar R., Zardari B., Artificial Neural Networks: An Overview. *Mesopotamian Journal of Computer Science.* 2023. P. 130–139. DOI:10.58496/MJCSC/2023/015

10. What is machine learning?. *IBM.* URL: https://www.ibm.com/think/topics/machine-learning (дата звернення: 12.04.2025)

11. Hight C. Automation within digital videography: From the Ken Burns Effect to 'meaning-making' engines. *Studies in Documentary Film.* 2024. Vol. 8, P. 235-250. DOI: 10.1080/17503280.2014.961632

12. Офіційний сайт Fliki URL: https://fliki.ai/ (дата звернення: 06.02.2025)

13. Офіційний сайт Veed URL: https://www.veed.io/tools/ai-video/text-to-video (дата звернення: 29.03.2025)

14. Mircevska S. Veed.io Review. *Cybernews.* 2025. URL: https://cybernews.com/ai-tools/veed-io-review/ (дата звернення: 10.01.2025)

15. Francis A. The AI Video Research Powering a Higher Quality Future. *Bitmovin*. 2023. URL: https://bitmovin.com/blog/ai-video-research/ (дата звернення: 21.04.2025)

16. Heinrichs J. Veed.io Review: The Easiest AI Video Editor I've Ever Used. *Unite AI.* 2024. URL: https://www.unite.ai/veed-io-review/ (дата звернення: 03.03.2025)

17. Veed.io Review: The Best AI Video Editor?. *AutoGPT.* 2024. URL: https://autogpt.net/veed-io-review-the-best-ai-video-editor/ (дата звернення: 19.02.2025)

18. Статистика трафіку Veed. *Semrush.* 2025. URL: https://www.semrush.com/website/veed.io/overview/ (дата звернення: 27.01.2025)

19. Офіційний сайт Steve AI. URL: https://www.steve.ai/

20. Офіційний сайт Elai URL: https://elai.io. (дата звернення: 14.03.2025)

21. Horvath I. Interactive video statistics prove that you need this type of media in your business. *Elai*. URL: https://elai.io/interactive-video-statistics/ (дата звернення: 01.04.2025)

22. Hughes O. Elai Review: Is It Worth the Investment?. *EWeek.* 2024. URL: https://www.eweek.com/artificial-intelligence/elai-review/. (дата звернення: 19.01.2025)

23. Heinrichs J. Elai Review: An AI Video Generator Perfect for Corporations. *Unite AI*. 2024. URL: https://www.unite.ai/elai-review/. (дата звернення: 05.02.2025)

24. Сторінка Github HunyuanVideo: A Systematic Framework For Large Video Generation Model. URL: https://github.com/Tencent/HunyuanVideo. (дата звернення: 23.03.2025)

25. Kong W., Tian Q., Zhang Z., Min R., Dai Z., Zhou J., Xiong J., Li X., Wu B., Zhang J., Wu K., Lin Q., Yuan J., Long Y., Wang A., Wang A., Li C., Huang D., Yang F., Tan H., Wang H., Song J., Bai J., Wu J., Xue J., Wang J., Wang K., Liu M., Li P., Li S., Wang W., Yu W., Deng X., Li Y., Chen Y., Cui Y., Peng Y., Yu Z., He Z., Xu Z., Zhou Z., Xu Z., Tao Y., Lu Q., Liu S., Zhou D., Wang H., Yang Y., Wang D., Liu Y., Jiang J., Zhong C. HunyuanVideo: A Systematic Framework For Large Video Generative Models. 2025. DOI: https://doi.org/10.48550/arXiv.2412.03603

26. Huang Z., He Y., Yu J., Zhang F., Si C., Jiang Y., Zhang Y., Wu T., Jin Q., Chanpaisit N., Wang Y., Chen X., Wang L., Lin D., Qiao Y., Liu Z VBench: Comprehensive Benchmark Suite for Video Generative Models. 2023. DOI: https://doi.org/10.48550/arXiv.2311.17982

27. Peng X., Zheng Z., Shen C., Young T., Guo X., Wang B., Xu H., Liu H., Jiang M., Li W., Wang Y., Ye A., Ren G., Ma Q., Liang W., Lian X., Wu X., Zhong Y., Li Z., Gong C., Lei G., Cheng L., Zhang L., Li M., Zhang R., Hu S., Huang S., Wang X., Zhao Y., Wang Y., Wei Z., You Y. Open-Sora 2.0: Training a Commercial-Level Video Generation Model in $200k. 2025. DOI: https://doi.org/10.48550/arXiv.2503.09642

28. Офіційна сторінка Github Open Sora. URL: https://github.com/hpcaitech/Open-Sora (дата звернення: 30.04.2025)

29. Dickson. B How Open-Sora 2.0 cuts the costs of AI video generation without sacrificing quality. *TechTalks*. 2025. URL: https://bdtechtalks.com/2025/03/24/open-sora-2/. (дата звернення: 30.04.2025)

30. Офіційний сайт Gemini Google URL: https://deepmind.google/technologies/gemini/. (дата звернення: 30.04.2025)

31. Документація Gemini 2.0 Flash URL: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash. (дата звернення: 30.04.2025)

32. Офіційна сторінка Github Piper TTS URL: https://github.com/rhasspy/piper. (дата звернення: 16.04.2025)

33. Офіційна сторінка Github Coqui XTTS URL: https://github.com/coqui-ai/TTS. (дата звернення: 16.04.2025)

34. Casanova E., Davis K., Gölge E., Göknar G., Gulea I., Hart L., Aljafari A., Meyer J., Morais R., Olayemi S., Weber J. XTTS: a Massively Multilingual Zero-Shot Text-to-Speech Model. 2024. DOI: https://doi.org/10.48550/arXiv.2406.04904

35. Офіційний сайт Pexels URL: https://www.pexels.com/videos/. (дата звернення: 16.04.2025)

36. Римар П.В., Колібабчук Д.І. Програма для автоматизації створення відео з використанням генеративного штучного інтелекту. *Наука і техніка сьогодні (Серія «Педагогіка», Серія «Право», Серія «Економіка», Серія «Фізико-математичні науки», Серія «Техніка»)*. 2025. № 4(45). С. 1498–1510. https://doi.org/10.52058/2786-6025-2025-4(45)-1498-1509

37. Колібабчук Д. І., Римар П. В. Порівняльний аналіз систем для автоматизованого озвучення відео. *Прикладні інформаційні технології 2025*: Матеріали всеукр. науково-практ. конф. здобувачів вищ. освіти та молодих вчен., м. Вінниця, 22 трав. 2025р. 2025.