НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
Комп'ютерних наук
(назва кафедри)
_____ Голуб Б. Л.
(підпис) (ПІБ)
"2" червня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

"Програмне забезпечення системи автоматизації дублювання англомовних відеофайлів"

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доцент _____ Вайганг Г. О.
(наукова ступінь та вчене звання) (підпис) (ПІБ)

Керівник Бакалаврської кваліфікаційної роботи

к.т.н., доцент _____ Бородкіна І. Л.
(наукова ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Мельник Олександр Сергійович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
(назва кафедри)
_____ Голуб Б. Л.
(підпис) (ПІБ)
"16" грудня 2024 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Мельнику Олександру Сергійовичу
(прізвище, ім'я, по батькові)
Спеціальність 121 – «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи Програмне забезпечення системи автоматизації дублювання англомовних відеофайлів

Затверджена наказом ректора НУБіП України від "16" грудня 2024 р. № 2249 "С"

Термін подання завершеної роботи на кафедру 2025.06.02
(рік, місяць, число)
Вихідні дані до бакалаврської кваліфікаційної роботи
Опис предмету дослідження, опис програмного забезпечення
Перелік питань, які потрібно розробити:
Системний аналіз предметної області
Аналіз предметної області
Розробка програмного забезпечення
Тестування програмного забезпечення

Дата видачі завдання "16" грудня 2024 р.

Керівник Бакалаврської кваліфікаційної роботи
к.т.н. доцент _____ Бородкіна І.Л.
(наукова ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання
_____ Мельник Олександр Сергійович
(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Timestamps - часова розмітка, коли в відео було сказано фразу(в контексті додатку)(з якого по який час було сказано певну фразу).

Creator - той хто створив **YouTube - канал**, чи завантажив відео на нього.

.srt - розширення текстового файлу, в якому вказується час для виводу субтитрів.

AI(artificial intelligence) - штучний інтелект, або ж інтелектуальна машина для вирішення проблем поставленими людьми перед ними.

Web - сервіс - простими словами сайт.

https - адрес(URL) - посилання до ресурсу де зберігається інформація(в нашому випадку відео).

Extensions - розширення в браузері (приклад: **Picture-in-Picture Extension (by Google)** яке дозволяє аби відео з **YouTube** було видно навіть тоді, коли користувач в іншій вкладці).

API - простими словами, це ключ для доступу до взаємодії з сайтами чи додатками.

ВСТУП

На сьогоднішній день всі **YouTube** канали(не враховуючи гігантів мільйонників), роблять все можливе, аби набрати більшу кількість глядачів, аби збільшити аудиторію яка буде переглядати їх відео. Але є кілька проблем з якими вони змушені зустрітися, одна з яких, це нестача зацікавлених глядачів, які розуміють мову **YouTube creator -а**. Хоча він(вона) можуть робити цікавий контент для окремого зрізу глядачів та гарно оформлювати відео, цікаво розповідати, відстежують коли краще завантажувати відео. Та їм не щастить “піднятись на хвилю слави”, через те що немає інтересу до створюваного контенту, для більшості **YouTube** аудиторії в її(його) мовному просторі. Та це спричиняє вигорання, аби полегшити для них просування було розроблено додаток який буде поширювати їх відео на нові простори до яких вони не могли підступитись через мову. Це сильно упростить процес поширення каналу по різним країнам та зробити відео доступними для великого сектору глядачів з інших країн та для людей з вадами слуху, що при перекладі на англійську мову зможе збільшити їх шанси(стосовно власників каналів) на розповсюдження їх за кордоном та надає приріст глядачів в значному об’ємі. Якщо взяти за приклад англомовних каналів то для них приріст буде як мінімум 34%, та так як англійська найрозповсюдженіша мова, то при перекладі для каналів з інших країн цей відсоток може перевищувати 200%, а то й 1000% в залежності від регіону в якому проживають та мови якою вони публікують контент. Це буде гарним інструментом для просування, та значним для формування людини як публічної особи і мережі інтернет, та й в користувачів збільшиться обсяг для перегляду у відповідному співвідношенні.

Розроблений програмний продукт, буде слугувати допоміжним інструментом для допомоги власникам каналів, аби більше користувачів побачити їх відео. Даний додаток, буде робити переклад відео, створення субтитрів, та додавання їх до відео. Ці операції допоможуть просунути відео краще, ніж аби завантажували відео просто однією мовою, це значно збільшить

поширеність відео й буде масовий приріст переглядів кожному хто буде користуватись даним сервісом, та й завдяки тому що люди будуть використовувати його, то й буде великий приплив зацікавлених в цьому додатку користувачів, що відповідно збільшить кількість замовлень та спричинить масштабування для компаній які будуть оперувати даними сервісами.

Даний додаток автоматизує процес дублювання англомовних відеофайлів. Продукт буде використовувати інструменти штучного інтелекту (ШІ) для створення визначення тексту з відео(з timestamps), також буде додавати субтитри та перекладати визначений текст(текст визначається враховуючи аудіо), забезпечуючи високу якість та швидкість обробки контенту, потребуючи мінімального втручання співробітника для виправлення помилок системи в процесі обробки завдання. Всі ці вище вказані елементи необхідні для виконання повного циклу програмного продукту, бо без попереднього кроку не буде можливості перейти до наступного, хіба що на етапі опрацювання помилок.

Методи дослідження

На початку аналізу “програмного забезпечення система автоматизації дублювання англомовних відеофайлів” був аналіз додатків аналогів. З аналогів було знайдено лише веб сервіси які можуть виконувати поставлені задачі коректно та швидко, але в них всіх був недолік. Вони могли виконувати лише одну функцію, що було великою проблемою, та так - як це все Web - додатки, то мені не зовсім підходили ці платформи для зразку створення обраної системи. Дані сайти це :savefrom.net - призначений для завантаження відео з YouTube; clideo та Clipchamp призначені для додавання субтитрів до відео але їм потребується ручний ввід бажаних субтитрів, або ж .srt файли, тож це не повноцінні Web - сервіси, якщо оцінювати за функціоналом, та якщо їх порівнювати з додатком який є темою дипломного програмного продукту.

Отож довелося з нуля шукати які використовувати механізми для завантаження відео(вони завантажуються в додатку з використанням youtube - API), як визначати сказаний текст(вся процедура відбувається за допомогою

assemblyai яка визначає текст та перетворює відразу на .srt файл), як перекладати .srt файли без порушення структури файлу та як додавати субтитри до відео(як правильно їх вирівнювати по сітці робити обведення, обирати колів обведення та тексту), також було проведено аналіз, чому перекладений текст не хотів виводитися як субтитри, а вияснилось, що при перекладі файлу, його структура змінювалась. Тобто файл при перекладі втрачав певні елементи, без яких файл некоректно зчитувало. Наприклад: “-->” перетворювалось на “->”, та час з умовного “3.286” перетворювався на “3286” чи на “3 286”, що стало фатальною помилкою, яку було вирішено перекладаючи лише ті частини де вміст містились буквені символи.

Теоретична значимість

Теоретична значимість цього дослідження полягає в тому щоб продемонструвати як взаємодіють та як використовуються в симбіозі різні елементи коду, такі як механізм розпізнавання мовлення, машинний переклад та створення субтитрів, та ці всі функції мають відповідати(бути з однаковим інтелектуальним змістом) один одному без викривлення переданої інформації при дублюванні її з попереднього, до наступного файлу.

Практична значимість

Практична значимість розробленого додатку полягає в створенні додатку(програмної системи), котра зможе автоматично витягувати аудіо - мовлення з завантажуваного файлу, для подальшого її перекладу на потрібну мову, та водночас створення субтитрів з витягнутим текстом.

Продукт може бути застосований для покращення доступності відео(для людей з вадами слуху чи іноземних глядачів), для навчання(якщо не існує відповідних матеріалів з доступним поясненням для розуміння теми чи проблематики в відкритому доступі), для локалізації відео(розповсюдження

реклами певного каналу) чи для текстового запису відео - лекцій(при умові що відео було завантажено на YouTube).

Актуальність

В сучасному світі відео контент є важливою складовою в таких сферах як розваги, саморозвиток та навчання. Зі збільшенням популярності мультимедійних платформ та долученням до них всіх людей які мають бажання створювати контент з’являється проблема в власників каналів(акаунтів) в наборі нових глядачів(підписників), а в глядачів(підписників) з’являється відчуття що контент різних каналів(зазвичай непопулярних чи нових) повторює те що вже було продемонстровано більшими(популярнішими) каналами, та при їх спробах створити новий унікальний контент, на ці канали не звертають увагу.

Також є проблема що не всі люди з вадами слуху та звичайні користувачі можуть знати іноземні мови, що їм відрізає можливість до перегляду контенту в якому вони зацікавлені, але не мають можливості в розумінні вмісту через мовний бар’єр, це обмеження скорочує обсяг контенту який вони можуть переглядати що суттєво скорочує кількість можливих відео які вони мають змогу переглянути. В англомовному YouTube просторі ситуація найкраща, так-як 66% на платформі англомовних, це дає зрозуміти те що при відсутності знань англійської мови, ви не можете переглянути більшість вмісту якого було опубліковано, та це ще означає що інші 34% не мають шансу вийти на міжнародну арену й набути світової популярності.

Система автоматизації дублювання англомовних відеофайлів має можливість виправити всі ці проблеми, будучи перекладачем аудіо вмісту, та показуючи переклад текстом на екрані, та з використанням технологій штучного інтелекту(ШІ або AI) визначаючи слова з відео полегшуючи створення субтитрів, що в рази скорочує час та сили які могли б бути витрачені на виконання поставлених задач людьми.

Зв'язок роботи з науковими програмами, планами, темами

Робота виконувалась відповідно плану науково-дослідних робіт кафедри комп'ютерних наук факультету інформаційних технологій Національного університету біоресурсів і природокористування України.

Мета

Дана розроблювана система, створювалась для спрощення поширення відео контенту між користувачами з різних мовних сфер, збільшення попиту на відео з малою мовною охопленістю на платформі, збільшення контенту для глядачів та для людей з вадами слуху які обмежені лише своєю мовою від перегляду більшої кількості контенту, прискорення надання перекладу та субтитрів до відео на різних мовах та різними мовами.

Задачі дослідження

1. Провести дослідження способів витягування відео з YouTube.
2. Провести дослідження способів витягування тексту з відео.
3. Провести дослідження стосовно створення перекладу для вмісту файлу.
4. Розробити додаток для автоматичного визначення тексту, створення перекладу, створення субтитрів мовою оригіналу та перекладеною мовою.
5. Створити робочий простір(інтерфейс додатку) для легкої роботи з зрозумілим інтерфейсом.
6. Налаштувати структуру збереження даних для організованого збереження створеного вмісту.
7. Інтегрувати ШІ для визначення слів з аудіо яке у відео.
8. Оптимізувати додаток для максимальної швидкодії, та мінімальних затрат ресурсів девайсу на якому відбувається обробка.

9. Провести тестування та виправити неполадки які виникали при підключені модулів, невідповідність бібліотек до внесених змін в відео - трансляційний платформі, чи з інших технічних причин.

10. Створити можливість для подальшого масштабування додатку, для спрощення підключення більшої кількості мов.

Об’єкт дослідження

Проводилось дослідження над Web - сервісами, такими як: **savefrom.net** - призначений для завантаження відео з YouTube; **clideo** та **Clipchamp** призначені для додавання субтитрів до відео; Вони обмежені в функціоналі, але поставлені перед ними задачі виконують неймовірно добре та швидко, мають широкий спектр функціоналу та багато функцій для варіативності операцій з ними.

Предмет дослідження

Визначалось як саме має завантажуватись відео з медіа - платформи(YouTube), як має визначатися текст з аудіо, після чого його(визначеного тексту) коректний переклад та додавання до відео, тільки вже у форматі субтитрів, яким чином має вказуватись час відображення субтитрів на відео, та як не змінити їх при перекладі.

Також дослідження було зосереджено на оцінці того, чи правильно додаток виконує всі ці процеси в послідовному алгоритмі, та чи не виникає збоїв при обробці даних, при виникненні проблем потрібно було шукати пов’язані елементи та проводити тестування, та проводити аналіз створених файлів на правильність запису чи відображення інформації.

Практичне значення одержаних результатів

В визначеному програмному забезпечені не відбудеться наступного кроку до того як створиться файл з попереднього, тому що вони взаємно пов’язані, та

не може відбутись процедура до того як будуть створені файли які потрібні для даного процесу.

Завдяки додатку люди зможуть охопити більшу сферу зацікавленості інших людей у вигляді контенту з зрозумілою для них мовою, це приємно коли відомі люди роблять локалізацію своїх відео на рідну для тебе мову що ще зацікавлює глядаціч підписатись на канали, бо коли люди бачать щось близьке для себе в місцях де цього не очікував це приємно дивує та активує якісь внутрішні механізми людей що активує відчуття радості всередині них, змушуючи їх зацікавитися каналом та її власником.

Апробація результатів роботи

За даною темою здавалися тези і апробація на конференціях в НУБІП та КНУКіМ в яких розповідалось про функціонал додатку, відображалися алгоритми структури та виконання задач, уточнювалось хто має бути потенційним користувачем з алгоритмами виконання процедур які повинен провести користувач для повної робочої схеми, аби відобразити функціонал та вимоги до користувача та вимоги до апаратного забезпечення з мінімальними вимогами, при тому щоб додаток показував добрі результати та не показував ніяких збоїв в процесі експлуатації.

Публікації

Роботи по даному застосунку публікувались на навчальній платформі **elearn.nubip.edu.ua/** як частина навчального плану, де було описано основний функціонал додатку. Загалом публікацій на цій платформі стосовно цього проекту було в межах від 25 - до 35 робіт так як робота виконувалась цілий навчальний рік.

Розділ 1. Системний аналіз предметної області

В цьому розділі буде проведено великий аналіз розроблюваної системи як одного об’єкту, буде розібрано з чого складається програмний продукт, розглянемо зв’язок складових додатку між собою, буде проведено аналіз вимог до системи від користувачів, та в зворотному порядку, тобто які має мати характеристики користувач аби мати змогу користуватися додатком без проблем, та щоб мати базове розуміння що потрібно робити.

Проведений аналіз мінімальних характеристик для користувача та для системи буде зображено нижче та буде описано обов’язки чи вимоги необхідні від акторів чи системи, буде описано кроки взаємодії актора з системою в яких буде детальний аналіз процесів які необхідні для виконання замовлення.

1.1 Опис предметної області

Сфера в якій застосовується розроблюваний продукт це поширення відеороликів серед іноземних користувачів, аби збільшити попит на контент який створює певний автор **YouTube-каналу**, та хотів би приріст глядачів, для збільшення популярності його та каналу, щоб в майбутньому принесло йому світову славу, та зробило публічною людиною.

Додаток розрахований на те щоб автоматично витягнути текст який говорився у відео, визначати мову якою було сказано текст, переклад та додавання вже перекладеного тексту як субтитри, також є можливість передати субтитри мовою оригіналу.

Цільовою мовою (тобто мова на яку буде перекладатись визначений текст) було обрано українську, для наглядної демонстрації коректного виконання поставлених задач для функціоналу застосунку(це було зроблено, бо не всі можуть знати іноземні мови).

Аби виконати повний цикл програмного продукту прийдеться користувачем виконати наступні дії які вказані в списку по порядку(для більшого розуміння, ще буде зображено схему):

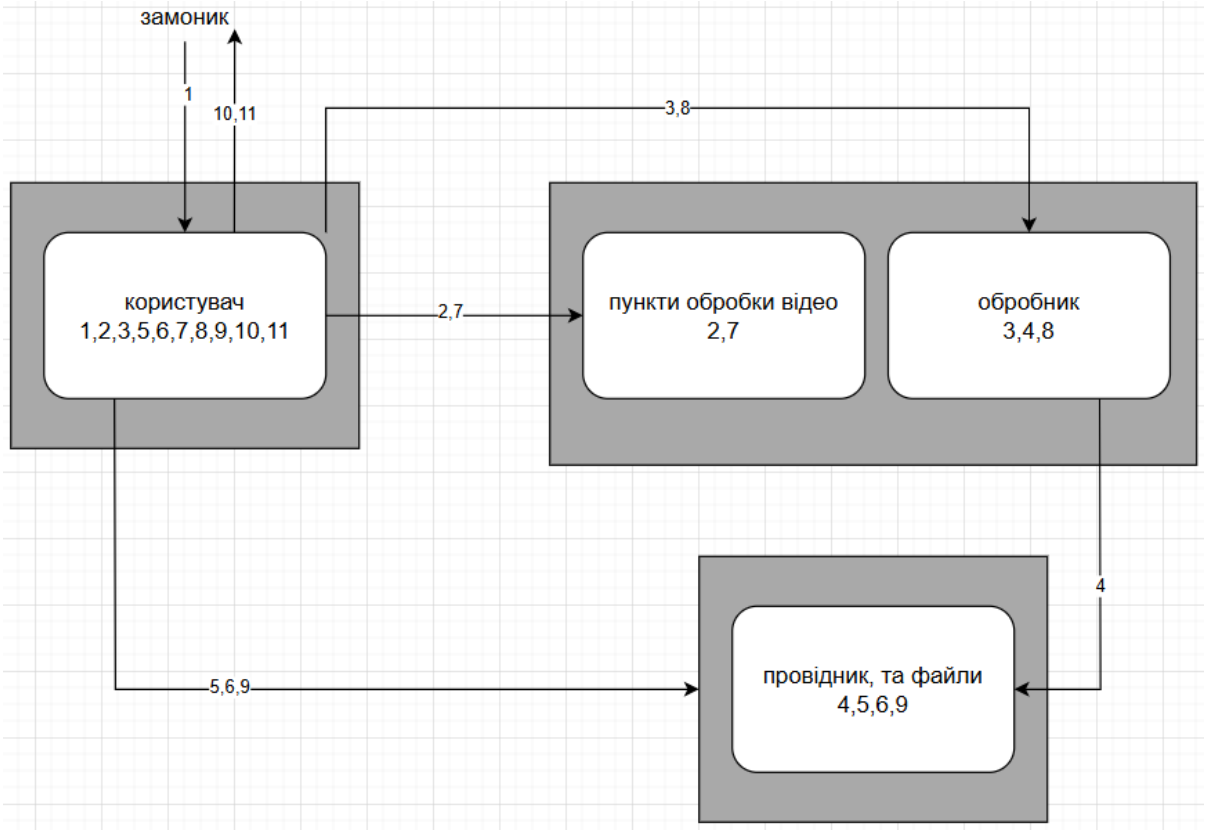

Рис.1 Діаграма послідовності у блоках

- Переглянути вимоги до оброблення замовлення(1)
- Ввести необхідні пункти до системи для обробки(2)
- Запустити обробник(3)
- Обробник створює папку з файлами відповідно до замовлення(4)
- Відкрити провідник з обробленими файлами(5)
- Ввести коректні виправлення до створених файлів(6)
- Виставити в додатку галочки на наступних етапах від тих де були помилки(якщо помилка була в субтитрах)(7)
- Повторно запустити обробник(8)
- За потреби повторно перевірити файли(9)
- Формування рахунку(10)
- Відправити замовлення до клієнта(11)

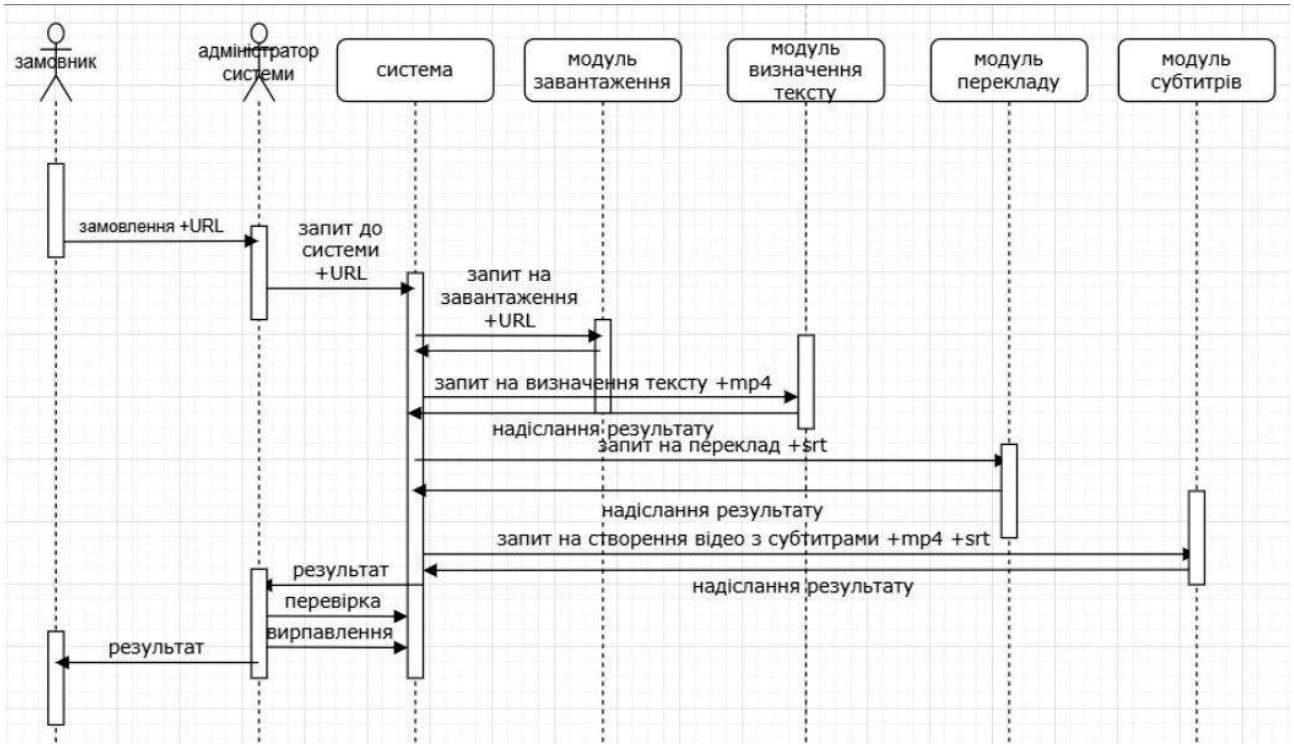

Рис.2 Діаграма послідовності

1.2 Аналіз вимог до програмної системи

Функціональні вимоги:

- Перехід за посиланням.
- Перевірка на існування материнської папки.
- Перевірка на існування папок в яких має міститись відео, та похідні файли.
- Завантаження відео.
- Створення файлу для зберігання визначених слів(.srt).
- Визначення мови відео.
- Витягування сказаних слів з відео та додавання до попередньо. створеного файлу.
- Створення файлу для перекладеного тексту(.srt).
- Додавання перекладеного тексту.
- Створення копії відео №1.
- Додавання до копії відео №1 субтитри оригінальною мовою.

- Створення копії відео №2.
- Додавання до копії відео №2 субтитри перекладеною мовою.
- Очікування.

Нефункціональні вимоги:

- Інтуїтивно зрозумілий інтерфейс
- Приємні для перегляду кольори
- Швидкодія
- Можливість продовжити процес обробки з середини, а не з початку алгоритму
- Високий рівень точності перекладу, та виставлення timestamps-ів

1.3 Моделювання предметної області

Загалом системою буде користуватись тільки “Регулювальник робочого процесу додатку”, але для захоплення всього робочого процесу, включно з комунікацією з клієнтами, потрібні люди які будуть працювати за межами додатку.

Отож структура з акторами та описом їх обов’язків буде виглядати наступним чином.

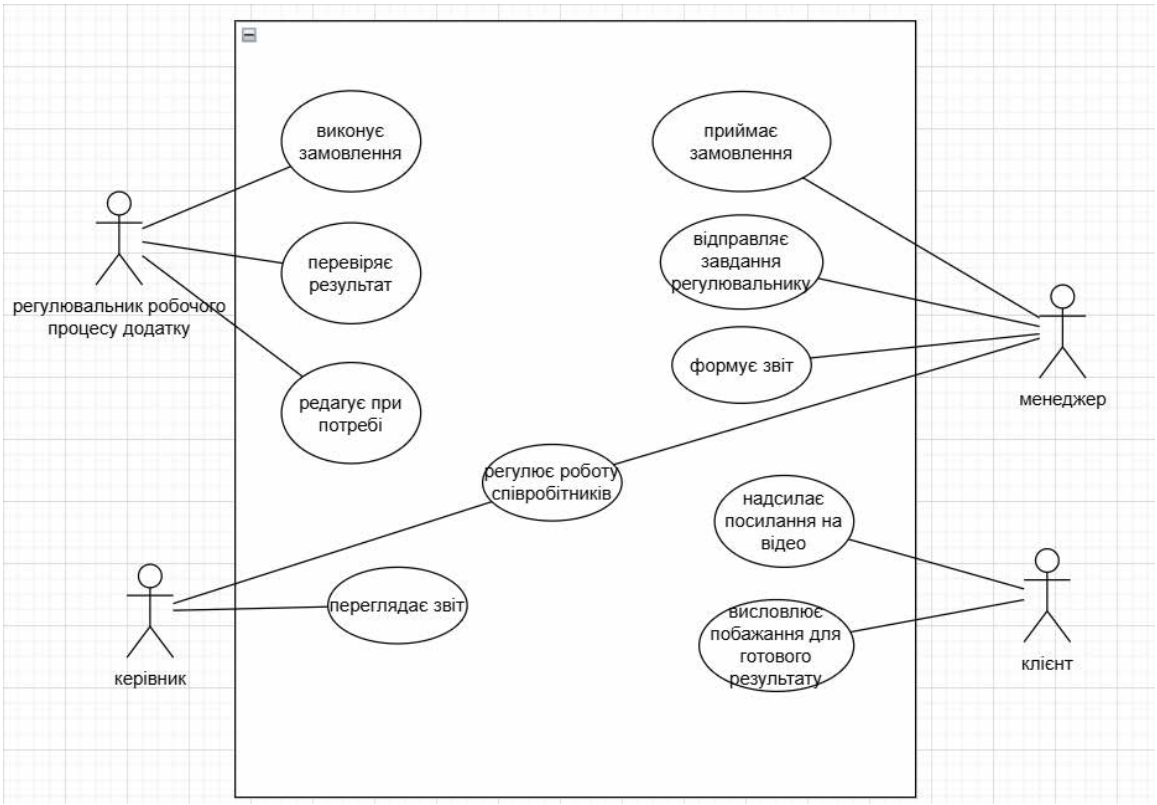

Рис.3 Діаграма прецедентів

Регулювальник робочого процесу додатку

- Виконання замовлення:
  - Додавання посилання до строки вводу;
  - Вибір необхідних пунктів обробки(які були обрані клієнтом завчасно);
  - Запуск обробника;
- Перевірка результату:
  - Перевірка чи завантажилось відео без помилок;
  - Перевірка чи внесені дані в .srt файл коректні:
    - Чи коректно вказаний час;
    - Чи коректно введений визначений текст;
- Редагування при виявлені неточностей в перекладі:
  - Виконуються всі ті самі пункти що й в виконанні замовлення, але без натискання клавіші для створення .srt файлів, чи їх перекладі.

- Відправлення замовнику його замовлення;

Менеджер:

- Приймає замовлення;
- Відправляє замовлення “регулювальнику робочого процесу додатку”;
- Регулює роботу співробітників;
- Відправляє виконане замовлення клієнтові;
- Формує звіт;

Керівник:

- Регулює роботу співробітників;
- Переглядає звіти;

Клієнт:

- Відправляє замовлення;
    - Надсилає посилання до відео на яке хоче зробити переклад та субтитри;
    - Висловлює побажання того яким бажає бачити результат;

1.4 Огляд інформаційних джерел та існуючих рішень

Аналоги системи:

1)**savefrom.ne**

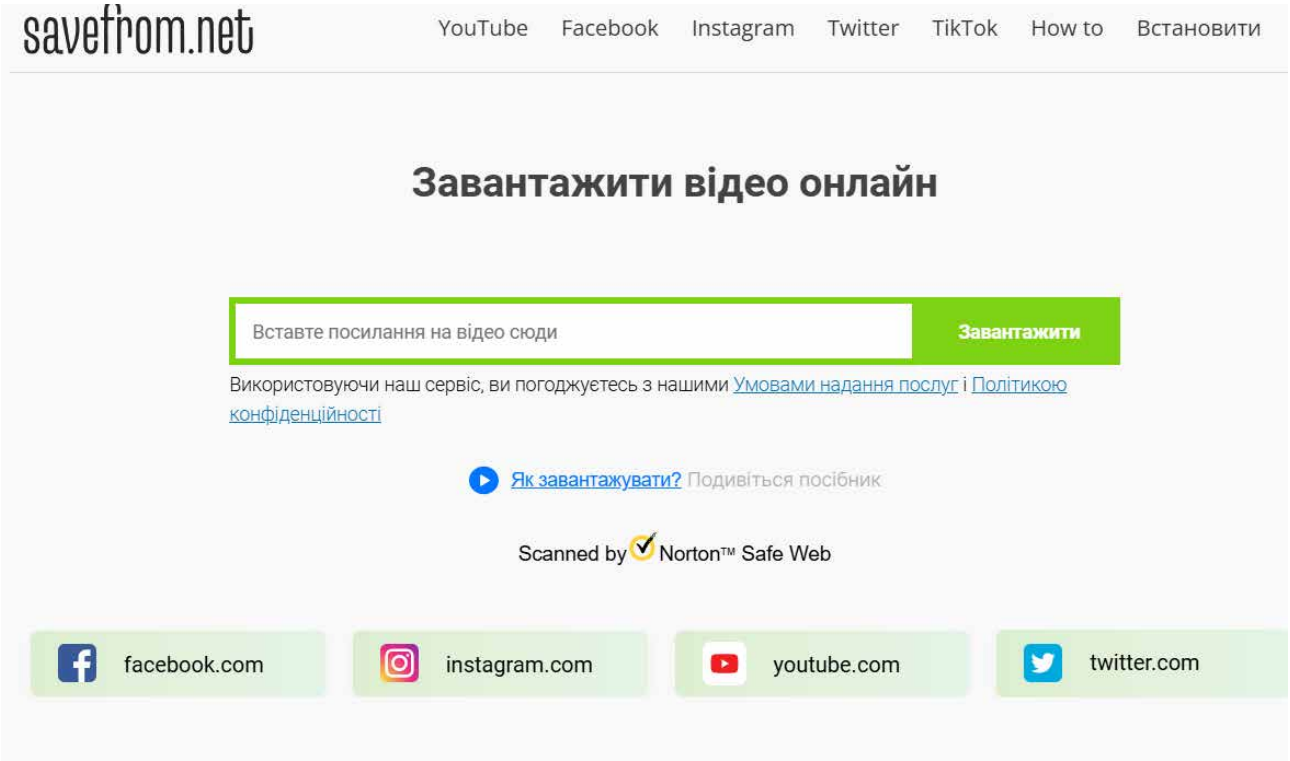

Рис.4 Savefrom.net

На зображені вище, веб-сайт який призначений для завантаження відео з Youtube безпосередньо на девайс користувача, використовуючи наданий (користувачем) https - адрес. Надає для власників інших каналів можливість для завантаження, а в подальшому користувачі самі вирізають та редагують певний елемент з відео та редагують його.

Звісно через цей додаток можна завантажити не лише з Youtube, але ще з Instagram, Twitter(поточний X), TikTok та з FaceBook. Також має можливість для вибору бажаного формату завантаження відео таких як: 4K, HD, .mp3 та .mp4.

Сервіс може бути використаний в таких браузерах як: Google Chrome, Mozilla Firefox, Safari, Opera та в браузерах які створені на основі Chromium.

2) clideo

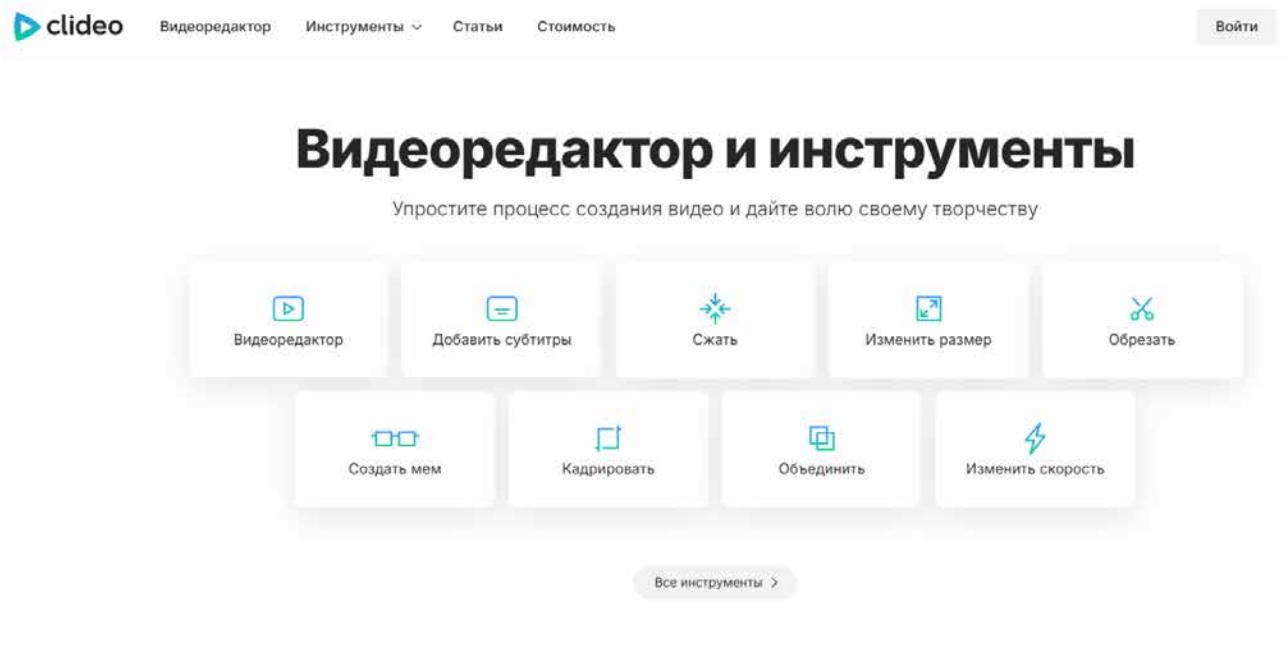

Рис.5 Clideo

На зображені веб-сайт який може створювати субтитри для завантаженого відео В ньому є функції на вибір: автоматичний ввід субтитрів, ручний ввід субтитрів, завантаження .srt файлів та переклад.

Сайт містить в собі такі пункти як: “відео-редактор” в якому можна монтувати відео, “додати субтитри” - він додає текст під відео, “стискання” - зменшує розмір виділення пам’яті під зберігання для відео, “змінити розмір” - змінює розширення відео(висоту в пікселях та ширину), “обрізати” - може скоротити тривалість відео чи вирізати непотрібну(виділену) частину, “створити мем” - інструмент для створення картинок, GIF чи коротких відео з текстом на ньому, “кадрування” - зміна співвідношення сторін, “об’єднання” - може поєднати різні відео в одне(або додати статичну картинку, чи звук на відео), “змінити швидкість” - змінює швидкість відтворення відео.

Також має додаткові функції які згорнуті. В даній секції можна робити все що було попередньо сказане, та додати “слайд шоу”, “створити GIF”, “повернути відео”(змінити його позицію з горизонтального на вертикальне), “віддзеркалити відео”, “реверс відео”, “видалити звук”, додати “відео фільтри”, запис аудіо та інше.

3) Clipchamp

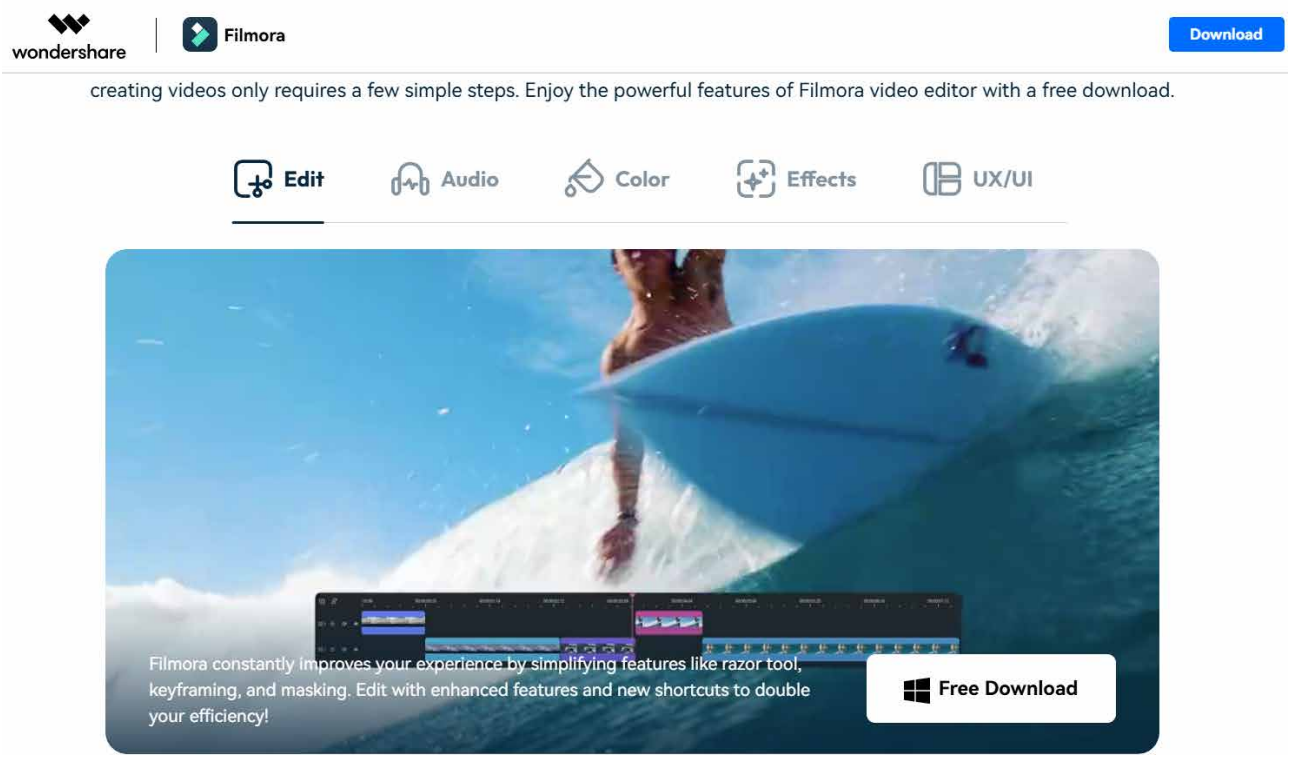

Рис.6 Clipchamp

На зображені веб-сайт сервісу від Microsoft який може створювати субтитри для відео В ньому є наступні функції: субтитри різними мовами світу, мовні фільтри й інструменти редагування та творчі стилі й ефекти.

Даний сервіс можна завантажити на робочий стіл як додаток та користуватись не заходячи до браузеру.

Має такі функції як:

- **Редагувати відео** в якому можна:
  - Додати відео;
  - Обрізати відео;
  - Перекинути відео на іншу доріжку;
- **Аудіо:**
  - Додати звук;
  - Змінити гучність;
  - Змінити “глибину звуку”;
  - Обрізати звук;

- Колір:
  - Зміна фільтру;
  - Налаштування кольору;
  - Налаштування яскравості;
  - Налаштування глибини кольору;
- Ефекти:
  - Додає маски на обличчя;
  - Додає обведення для відео;
  - Додає стікери на відео;

Має зручний інтерфейс та гарно підібрані кольори, що можна довго працювати без напруження для очей, це також впливає позитивно на робочий процес та на ефективність з якістю виконання проектів.

Таб. 1

Сервіс	Переваги	Недоліки
savefrom.net	Підтримка завантаження відео в різних браузерах, є розширення додатку в браузері(Extensions).	Обмеження якості завантажуваного відео, реклама, часто блокується, не працює в певних країнах, працює з перебоями.
clideo	Простий в використанні, багатофункціональний, має хмарне збереження, мультиформатність (стосовно форматів відео).	Водяні знаки, обмеження розмірів файлів, повільна швидкість обрбки
Clipchamp	Інтегрований з Microsoft, приємний інтерфейс, онлайн та офлайн версії, є вибірка шаблонів	Обмежена безкоштовна версія, проблеми з обробкою великих файлів, важко виставити

Сервіс	Переваги	Недоліки
		додавання нового відео чи звуку, багато функцій потребує з'єднання з інтернетом(навіть в офлайн версії для багатьох функцій потребується інтернет).

1.5 Постановка завдання

Система автоматизації дублювання англомовних відеофайлів розроблювалась з метою дати власникам каналів можливість поширити їх публічність на платформі, для глядачів які не чують можливість читати субтитри їх мовою, а для звичайних користувачів дає розуміння вмісту відео.

Зараз YouTube має функцію субтитрів, але вони загалом англійською та не підлаштовуються під регіон глядача базуючись на відео які він переглядав на рідній мові, що створює проблеми для певного прорізу користувачів платформи.

Додаток має вирішити дані проблеми аби всі вказані користувачі могли позбутись своїх обмежень, та відчувати себе вільніше користуючись платформою без спроб зрозуміти іноземні слова які могли б бути не правильно чи незрозуміло сказані.

Розроблюваний застосунок має прибрати мовний бар'єр використовуючи автоматизований переклад, він має завантажувати відео для обробки, за допомогою AI визначити текст який був сказаний актором на відео, перекласти створений файл та додати готові субтитри до відео(за потреби ще можна створити відео з субтитрами мовою оригіналу).

Після запуску додатку користувач має вписати в поле вводу URL-посилання з відео, після чого він має обрати які дії хоче клієнт щоб додаток виконав(переклад додати субтитри і так далі), далі натискає клавішу на екрані, яка знаходиться внизу вікна(там одна клавіша не враховуючи ті які вгорі праворуч), як додаток обробить відео та виконає всі вказані дії користувач має

перевірити вміст файлів на коректність(чи правильно перекладено, та чи в правильний час виведено субтитри) якщо виявлено помилку то виправляє, в кінці це все відправляється до клієнта а саме всі створені файли такі як файли субтитрів, відео - файл з доданими субтитрами та перекладені зразки(оригінальне відео не надсилається).

Для реалізації буде використано YouTube-API для доступу до завантаження, AI для розпізнавання мовлення у відео, машинний переклад для зміни мови створюваних субтитрів.

Обробка одного замовлення немає сталих часових рамок, так як відео може мати неймовірно довгу тривалість(наприклад завантажений стрім який може тривати від 1 години - до 5+годин).

Висновки до Розділу 1

Під час системного аналізу тематичної області було виявлено, що розробка автоматизованої системи перекладу відео з YouTube має велике практичне значення, особливо в умовах глобалізації контенту та зростаючого попиту на багатомовне споживання інформації. Така система має на меті вирішити нагальну проблему, таку як мовний бар'єр в сприйняті контенту та його поширені, який обмежує доступ іноземних користувачів до всього наявного відео контенту який розміщений на платформі, та створене авторами з певних регіонів з не надто популярною мовою в межах платформи **YouTube**.

Завдяки функціоналу автоматичного завантаження відео, розпізнавання мовлення, перекладу тексту та додавання субтитрів до відео, додаток забезпечує повний цикл обробки замовлення з мінімальною участю користувача. Це дозволяє не тільки автоматизувати повторювані процеси, але й забезпечити високу продуктивність під час роботи з відеоматеріалами великого обсягу.

Аналіз подібних сервісів, таких як **Savefrom.net, Clideo та Clipchamp**, дозволив визначити основні переваги та недоліки існуючих рішень та врахувати ці аспекти під час розробки додатку, було обрано що краще використати з визначеного. Створений програмний продукт було розроблено з урахуванням

головних помилках подібних систем, та позичаючи переваги, доповнюючи їх автоматизованими процесами, гнучкою структурою взаємодії з користувачем, та підтримкою перекладу.

Отже, завдання є актуальним, технічно можливим в реалізації та відповідає вимогам сучасного ринку цифрових послуг в сфері мультимедіа. Розроблений застосунок має перспективу подальшого масштабування та інтеграції з іншими сервісами з метою розширення аудиторії користувачів та покращення доступності відео контенту для користувачів з різних країн.

Розділ 2. Проектування інформаційного та програмного забезпечення

У даному розділі буде представлено логічну структуру та архітектуру організованої автоматизованої системи перекладу відео з платформи поширення відео YouTube, яка відповідає за процеси, що забезпечують завантаження відео, генерацію субтитрів у окремих файлах з розширенням(.srt), переклад цих новостворених файлів(.srt), додавання перекладених та оригінальних субтитрів до відео файлу, а також структуру управління збереженими відео матеріалами. Основою для розробки даного застосунку стала потреба в швидкому та коректному способі опрацювання відео для додавання субтитрів до нього, аби в подальшому користувачі з різних країн, та з вадами слуху, могли дивитись іноземні відео, без потреби в розумінні їх мови.

Спроектована системи охоплюватиме логічну модель даних у вигляді ER-діаграми, яка буде описувати основні сутності проекту(материнські, дочірні папки, відеофайли, файли субтитрів) та їх зв’язки між собою. Діаграма класів показуватиме взаємодію між системними об’єктами, відображатиме структуру даних і передавання інформації між внутрішніми елементами. Діаграма пакетів відображатиме розділення системи на функціональні блоки: графічний інтерфейс, обробку та сховище, що полегшуватиме розуміння архітектури додатку. Діаграма компонентів даватиме змогу зрозуміти як окремі модулі програмного забезпечення взаємодіятимуть між собою для досягнення цілі, яка полягає в формуванні перекладеного та перенесення аудіо в текстовий формат відео контенту.

Продемонстровані моделі дозволятимуть детально описати логіку роботи системи, визначити структуру та залежності між її елементами, а також забезпечуватимуть передумови для подальшої реалізації функціоналу.

2.1 Логічна модель даних у вигляді ER-діаграми

mother_folder:

Атрибути:

- **Path_M(FK) : text**(повний шлях до материнської папки)
- **name: str**(назва головної папки).
- **Location: text**(розташування материнської папки відносно **.exe** додатку).
- **Folder_voluem: size_t**(розмір вмісту папки).
- **Content_name: text**(назви файлів чи папок які розміщені в материнській папці).

Асоціація:

Містить **child_folder**, материнська папка містить багато дочірніх папок.

child_folder:

Атрибути:

- **Path_ch(FK) : text**(повний шлях до дочірньої папки).
- **Folder_voluem: size_t**(розмір вмісту папки).
- **name: str**(назва дочірньої папки).
- **Short_location: text**(локація папки в системних файлах).

Content_name: text(вміст папки).

Асоціація:

Містить **video**, тобто в одній дочірній папці може зберігатись багато відео.

Video:

Атрибути:

- **Path(FK): text**(повний шлях до відеофайлу).
- **Voluem: size_t**(розмір відеофайлу).
- **name: str(**назва відео).
- **Short_location: text**(локація відео в системних файлах).

- **Expancion: text**(розширення відеофайлу).
- **Language: text**(мова вмісту файлу).

Асоціація:

Має зв’язок з **language**, тобто кожне відео може мати певну окрему мову.

Language:

Атрибути:

- **code: str** - код мови (наприклад, "en", "uk").
- **name: str** - назва мови (наприклад, "English", "Українська").

Асоціація:

Має зв’язок з **video** бо кожне відео має мати певну мову.

Загальні зв’язки:

- mother_folder → містить кілька → child_folder
- child_folder → містить кілька → video
- video → пов’язане з → language

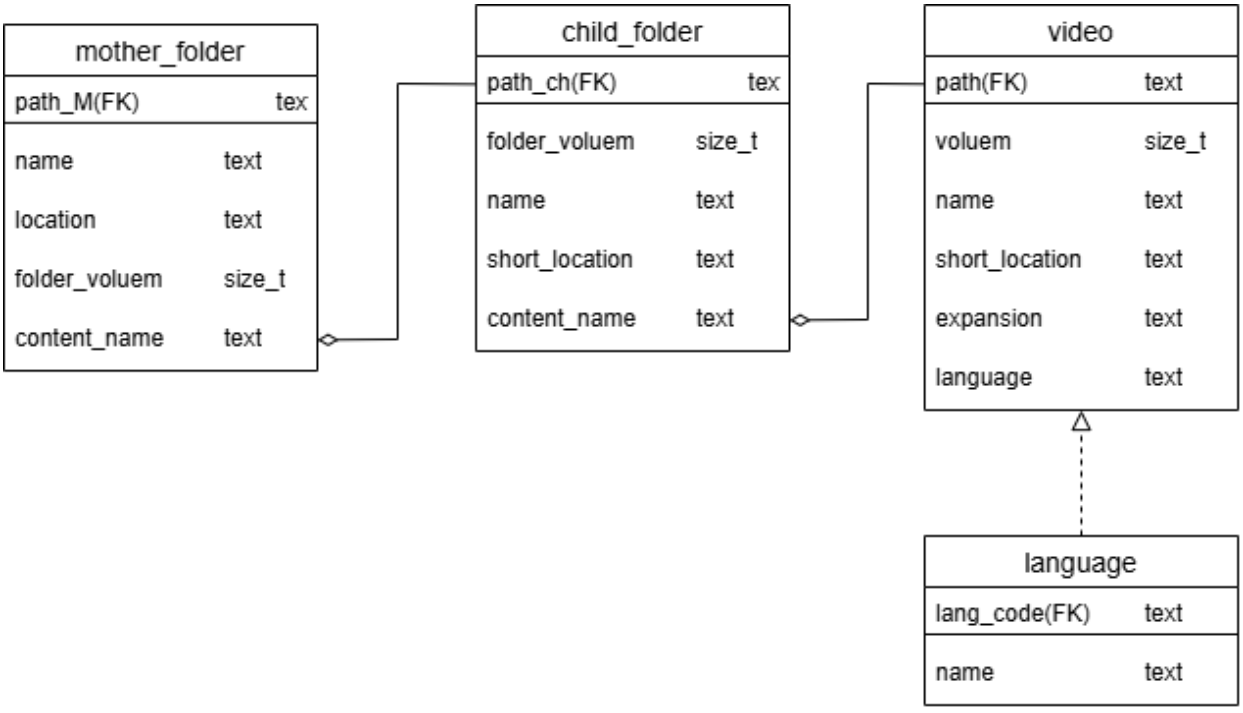

Рис.7 ER - Модель даних обраного програмного забезпечення

2.2 Діаграма класів та кооперацій

Метою було зображення кооперацій які присутні в додатку. Загалом тут зображується як основні створювані файли передають свої елементи до інших класів. Таким чином можна побачити що міститься в елементах, що можна передати, що зберігається для передачі на наступні кроки, та що саме передається до наступних елементів.

Відео:

Local:

- -Назва();
- -Розширення();
- -Аудіо();
- Мова();

Public:

- **+saveVideo();**

Процеси:

- Завантажує відео;
- Зберігає локально;

Переклад:

Local:

- -Назва();
- -Мова();
- -Код_мови();
- -Розширення();

Public:

- **+saveText();**

Процеси:

- Відкриває .txt;
- Перекладає текст ;
- Створює новий файл для запису ...ua.srt;

- Записує переклад в створений файл;

Субтитри:

Local:

- -Вміст();
- -Назва();
- -Мова();
- -Розширення();
- -Код_мови();

Public:

- **+unite(.txt+.mp4);**

Процеси:

- Відкриває .srt;
- Об’єднує з відео;

Проста кооперація

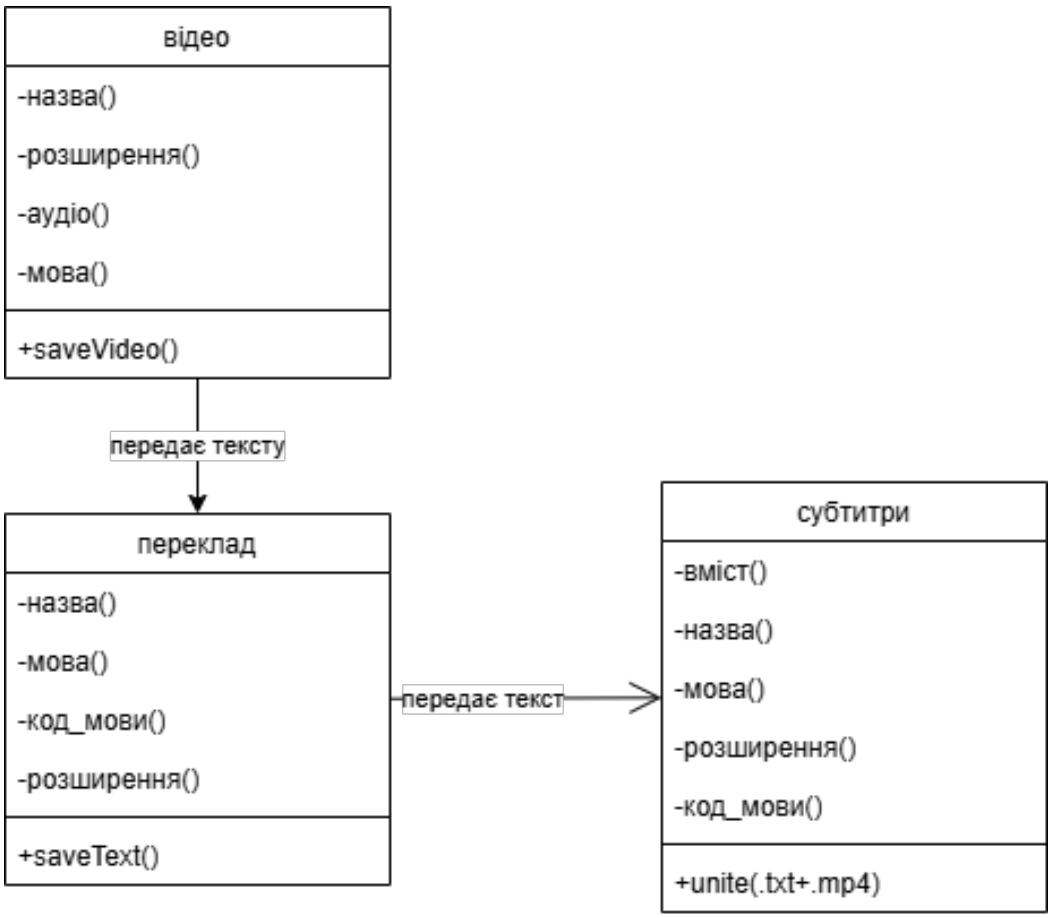

Рис.8 Діаграма класів з кооперціями

2.3 Діаграма пакетів

1) Графічний інтерфейс:

Аби звичайні користувачі могли взаємодіяти з системою не думаючи чи система може виконати певні дії розробляються графічні інтерфейси користувача. Цей пакет відповідає за взаємодію користувача із системою через зручний для користувача спосіб. Містить підсистеми:

- **Вибір дій над відео:** дозволяє користувачеві обрати операцію, яку необхідно виконати над відео.
- **Обробник:** є посередником між інтерфейсом і логікою обробки
- **Виведення процедури:** показує користувачу перебіг виконання обраної дії.
- **Керування сховищем:** забезпечує доступ до функцій керування сховищем.

2) Обробка:

В даному контексті це логіка опрацювання відео, яка виконує всі внутрішні процеси додатку, та не відображається для звичайного користувача, На ній зосереджено основну масу навантаження, й там виконується загальна маса внутрішніх процесів, та в даному контексті вона складається з наступних елементів(процесів):

- **Завантаження:** скачування відео з YouTube;
- **Переклад:** зміна мови субтитрів на обрану;
- **Субтитри:** створення чи обробка субтитрів;
- **Об'єднання:** поєднання перекладеного контенту з оригінальним відео (накладання субтитрів);

3) Сховище:

Даний елемент потрібний для організованого зберігання даних, для легкого пошуку створених вторинних, чи первинних файлів. В даній системі збереження даних організоване таким чином, що всі файли які з'явилися під час

виконання одного замовлення, вони мають однакову назву та зберігаються в одній папці. Цей пакет забезпечує зберігання і управління контентом:

- **Показ вмісту:** перегляд доступного контенту;
- **Управління вмістом:** додавання, видалення або редагування даних у сховищі.

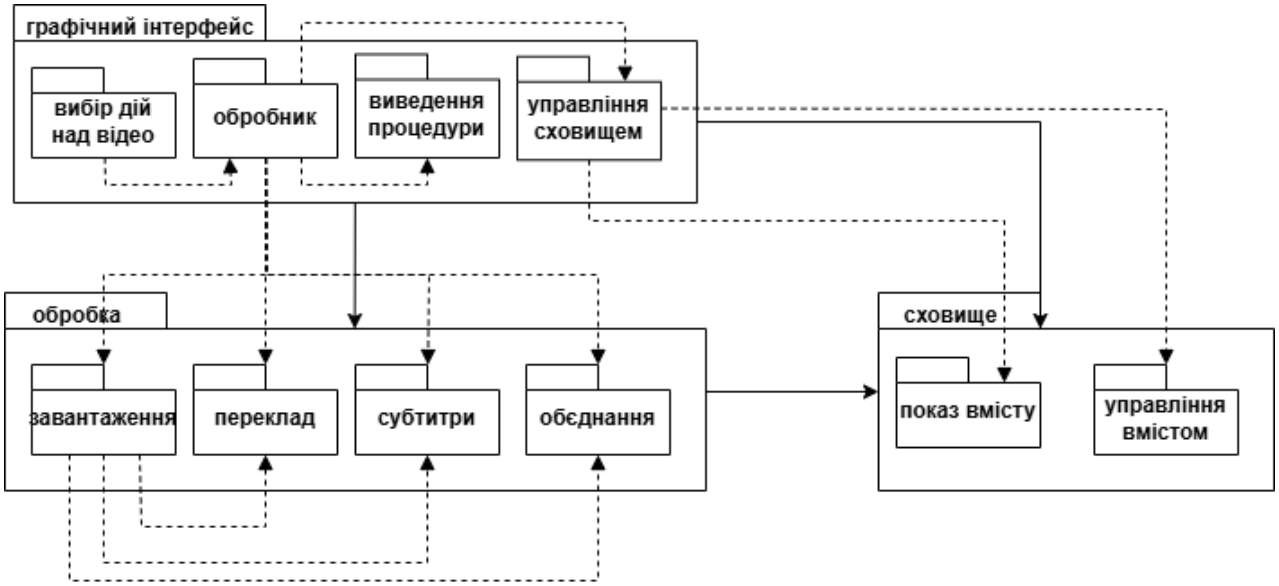

Рис.9 Діаграма пакетів

Ця діаграма пакетів, показує архітектуру програмного забезпечення для системи автоматизованого перекладу відео. На ній зображені три головні пакунки та їх складові з зв’язками, таким чином:

1) Після того як користувач вибрав певні дії над відео(після того як передав посилання на нього) в графічному інтерфейсі в пункті **“вибір дій над відео”**, дана інформація передається до внутрішнього компоненту **“обробник”**.

2) Компонент **“обробник”** з групи графічний інтерфейс, передає команду для всіх компонентів з групи обробка(а саме **“завантаження”**, **“переклад”**, **“субтитри”**, **“об’єднання”**), та до внутрішніх процесів його групи для виведення процедури того яке зараз відео обробляється, та для того щоб відправити запит на керувати вмістом(а саме додавання нових файлів до сховища).

3) потім в **“обробка”** відбувається послідовність процесів для обробки замовлення, та після виконання одного пункту створений файл передається до **“сховище”.**

Послідовність в групі **“обробка”** відбувається наступним чином.

Завантаження →переклад → субтитри → об’єднання.

2.4 Діаграма компонентів

1) Вікно обробника:

Загалом цей структурний елемент повинен містити в собі всі функції котрі виконуються в процесі виконання обробки замовлення разом з інтерфейсом, що робить його універсальним для всіх поставлених перед ним проблем. Саме там будуть виконуватися всі дії системи, такі як переклад, завантаження, створення субтитрів, виклик сховища та відображення інтерфейсу, та виконується функціонал який вказаний далі:

- **Інтерфейс:** зображає вікно в якому користувач може вставити посилання на відео, вибрати вимоги поставлені замовником натискаючи на відповідні **CheckBox-си**, після чого натиснувши на кнопку яка там відображається знизу посередині запускає обробку файлів.
- **Обробник файлів:** даний елемент відповідає за порядок виконання дій функціями, щоб не було збоїв в алгоритмах виконання процесів, він забезпечує послідовне виконання процесів, що забезпечує коректне виконання запитів до системи.
- **Субтитри:** відповідає за створення субтитрів. Субтитри створюються в процесі визначення аудіо-слів(слова що було озвучено у відео), та записує ці слова(словосполучення), порядковий номер фрази,та час коли це було сказано розділяючи різні фрази(разом з їх орієнтирами - маркерами) міжрядковим пропуском(як при натисканні клавіші **Enter** двічі, або як в мові програмування **C++** є відповідником до **“/n/n”**). Дані субтитри записуються до файлу формату **.srt**. Весь процес визначення тексту здійснюється за допомогою **AI**.

- **Переклад:** створений файл субтитрів пускається через переклад для зміни мови. Створюється копія **.srt** файлу, але ще додається модифікатор перед розширенням, який позначається як символи відповідники до мови на який було перекладено файл(**приклад:** назва файлу зберігається + ua.srt).Перекладач перекладає лише ті рядки файлу де є літери, щоб не пошкодити інші частину файлу.
- **Завантаження:** завантаження здійснюється з використанням **YouTube – API** яке дає змогу саме завантажити файл який був за посиланням на платформі.
- **Виклик сховища:** внутрішня функція додатку яка допомагає у створені, чи перезаписі файлів які вже було створено. Також відновлює материнську чи дочірні папки якщо їх було видалено(дочірні папки відновлює при обробці того відео яке вже раніше було оброблене фактично створюючи нову папку в материнській).

2) Сховище:

Середовище яке слугує для збереження завантажених файлів, та простором для створення нових(дочірніх) файлів з якими можна буде в подальшому працювати чи відправити замовникові після виконання потребовуваних вимог.

- **Вміст:** всі файли які зберігаються в материнські папці.
- **Управління вмістом:** додавання, видалення певної частки вмісту(при потребі перепис вмісту файлів з субтитрами).
- **Субтитри:** .srt файл який буде переноситись до відеофайлу для створення відео з субтитрами(оригінальною мовою).
- **Перекладені субтитри:** ua.srt файл який буде переноситись до відеофайлу для створення відео з перекладеними субтитрами.
- **Відео з оригінальними субтитрами:** відеофайл з доданими субтитрами мовою оригіналу.
- **Відео з перекладеними субтитрами:** відеофайл з доданими субтитрами цільовою(обраною) мовою.

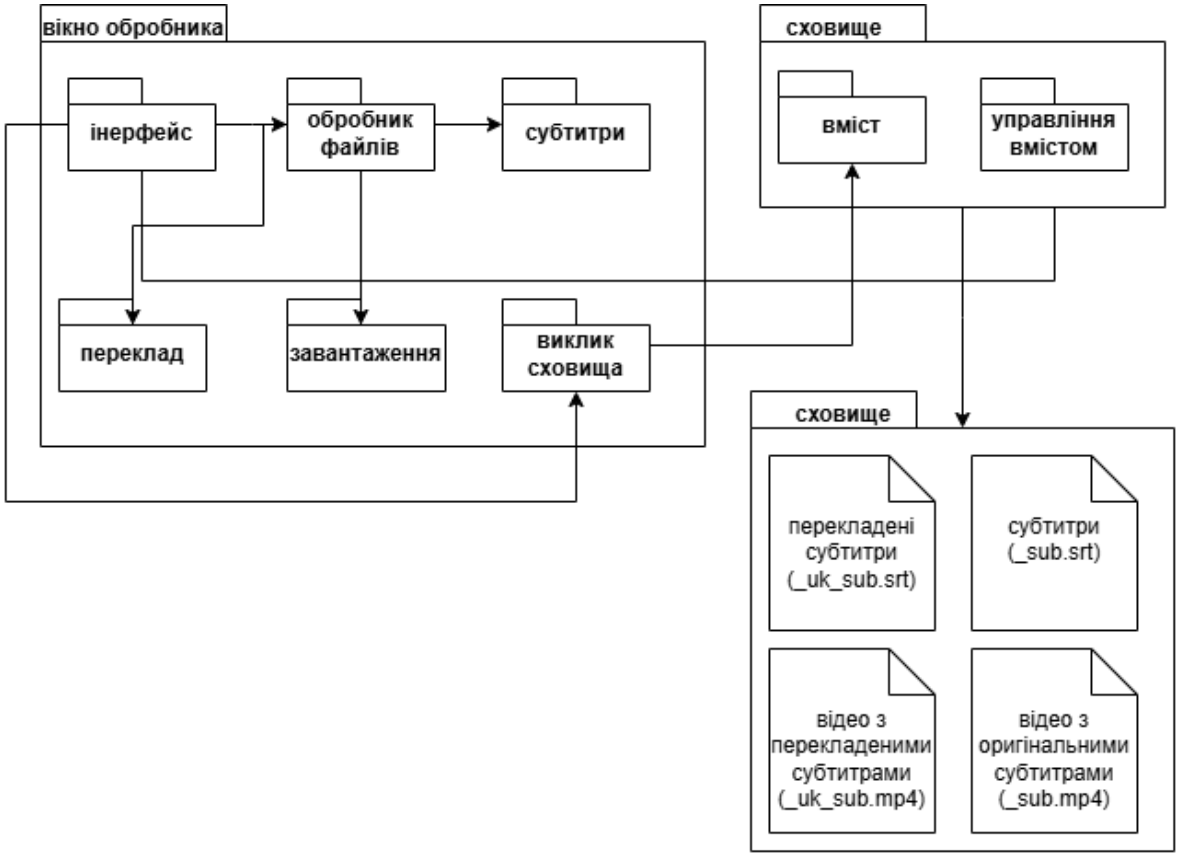

Рис.10 Діаграма компонентів

На цій діаграм зображено які компоненти до яких звертаються для виконання певних функцій їх зв’язки .

1) **Інтерфейс** звертається до обробника файлів з внесеними користувачем вимогами для обробки (інтерфейс → обробник файлів);
2) **Обробник файлів** звертається до елементів **переклад, субтитри, завантаження** відповідно до алгоритму, аби виконати поставлене завдання(Інтерфейс →завантаження, Інтерфейс → переклад, Інтерфейс → субтитри).
3) Після виконання однієї дії обробник зберігає результат в сховищі, та повторює цю дію поки не виконуються всі завдання(вікно обробника → сховище).
4) В процесі обробки створюються файли які вказані в сховищі що нижче.

Висновки до Розділу 2

У даному розділі було представлено логічну структуру та архітектуру організованої автоматизованої системи перекладу відео з платформи поширення відео YouTube, яка відповідає за процеси, що забезпечують завантаження відео, генерацію субтитрів у окремих файлах з розширенням(.srt), переклад цих новостворених файлів(.srt), додавання перекладених та оригінальних субтитрів до відео файлу, а також структуру управління збереженими відео матеріалами. Основою для розробки даного застосунку стала потреба в швидкому та коректному способі опрацювання відео для додавання субтитрів до нього, аби в подальшому користувачі з різних країн, та з вадами слуху, могли дивитись іноземні відео, без потреби в розумінні їх мови.

Спроектована системи охоплюватиме логічну модель даних у вигляді ER-діаграми, яка буде описувати основні сутності проекту(материнські, дочірні папки, відеофайли, файли субтитрів) та їх зв’язки між собою. Діаграма класів показуватиме взаємодію між системними об’єктами, відображатиме структуру даних і передавання інформації між внутрішніми елементами. Діаграма пакетів відображатиме розділення системи на функціональні блоки: графічний інтерфейс, обробку та сховище, що полегшуватиме розуміння архітектури додатку. Діаграма компонентів даватиме змогу зрозуміти як окремі модулі програмного забезпечення взаємодіятимуть між собою для досягнення цілі, яка полягає в формуванні перекладеного та перенесення аудіо в текстовий формат відео контенту.

Продемонстровані моделі дозволятимуть детально описати логіку роботи системи, визначити структуру та залежності між її елементами, а також забезпечуватимуть передумови для подальшої реалізації функціоналу.

Розділ 3. Розробка інформаційного та програмного забезпечення

В даному розділі буде проведено демонстрацію процесів розробки інформаційного та програмного забезпечення для системи автоматизації дублювання англомовних відеофайлів. В проекті буде присутнє локальне збереження файлів, для оптимізації використання внутрішніх процесів на передачі даних до віддаленого серверу, та якомога більше уникаючи витоку інформації на зовні. Також буде продемонстровано створення структури зберігання даних, в яких назва відео яке буде завантажуватись, буде таких назвою папки в якій буде міститись оригінальне відео, та всі створені від цього відео дочірні файли(матимуть свої примітки для організації даних). буде продемонстровано зразки назв файлів, з умовною початковою назвою та продемонстровано код який виконує самі функції. Також буде описано які механізми використовуватись, бібліотеки та їх компоненти, аби написати дипломний проект.

3.1 Система управління інформаційною базою

В проекті **“програмне забезпечення система автоматизації дублювання англомовних відеофайлів”** всі дані зберігаються локально, аби зекономити час на виконання операцій, та локальне зберігання забезпечує меншу ймовірність витоку всіх даних, хоча локальне збереження не є саме по собі безпечним, але краще зберігати на різних девайсах інформацію, ніж зберігати на одному.

Якщо буде запущено додаток, а материнської папки (мається на увазі саме папка в якій будуть зберігатися теки з відеофайлами, та пов’язаними з ними) не буде існувати, то система це помітить, та створить таку саму папку, в якій будуть зберігатися всі створенні дані стосовно замовлень, та дані які буде створено від одного відео, будуть мати однакову назву з різними закінченнями та різними розширеннями(папка в якій будуть зберігатися дані з одного відео, буде мати

назву відео також). Назва буде визначатися на етапі завантаження відеофайлу, та визначатиме його за допомогою **YouTube – API.**

Отож: назва відео визначається з платформи з якої було завантажено, ця назва передається до папки в якій будуть зберігатися дані стосовно його, та файли матимуть ту саму назву що і папка(а папка має назву оригінального відео), але матимуть різні розширення)(**.mp4** та **.srt**), та матимуть різні закінчення відповідно до модифікацій.

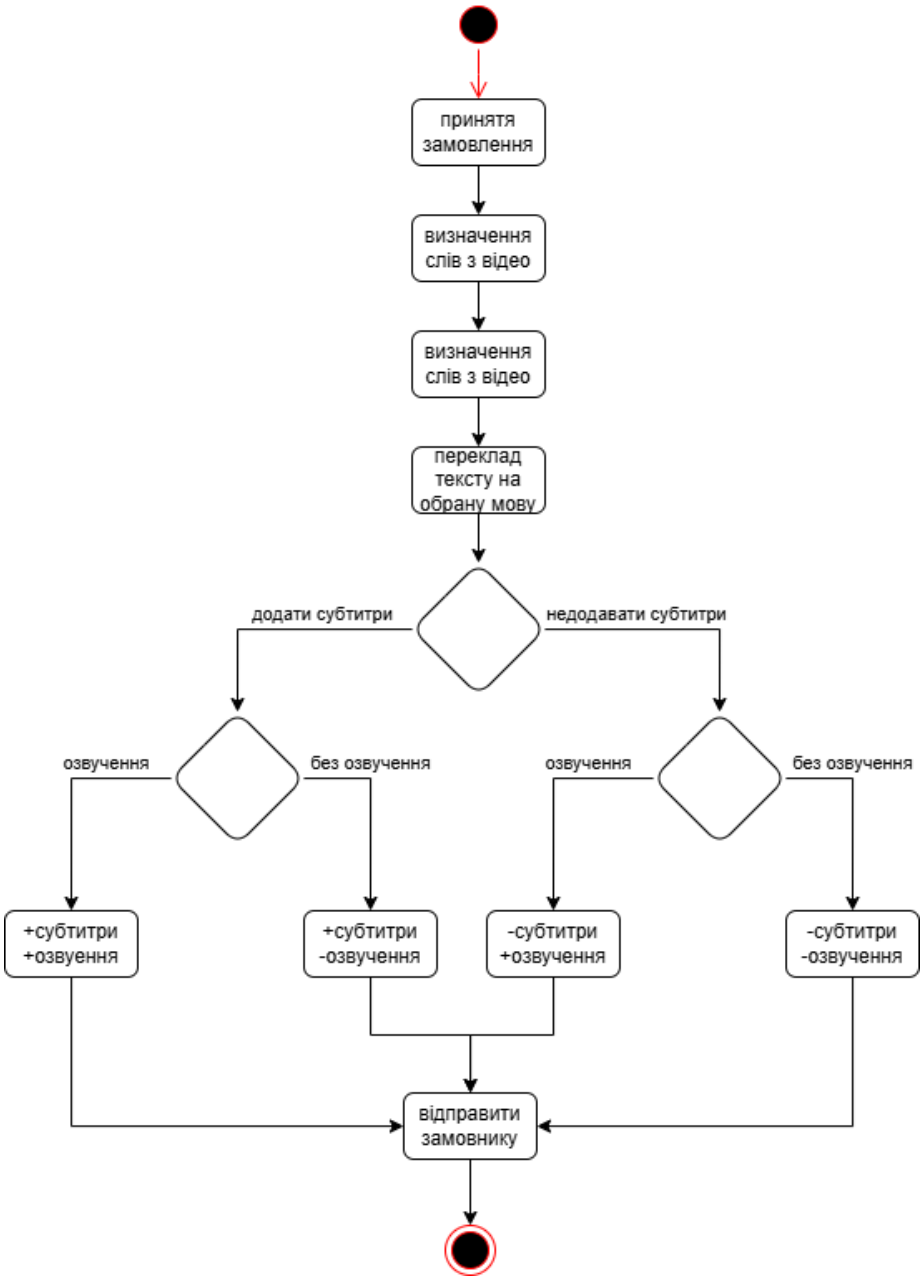

Рис.11 Діаграма активності

При умовній назві оригіналу “A” створювані файли будуть мати наступні назви:

Таб.2

	Субтитри	Відео з субтитрами
Без перекладу	A.srt	A_sub.mp4
З перекладом	A_ua.srt	A_sub_ua.mp4

Виходить що субтитри записуються в **.srt** файли. В ньому розміщені порядковий номер фрази (1,2,3...) на наступній стрічці часові межі в яких було сказано певну фразу(00:00:00,320 --> 00:00:03,720), де перші 00 позначають години, другі 00 хвилини, 03 це вже секунди, а числа після коми це мілісекунди(720), й таким чином фраза починається о 00 годині 00 хвилині 00 секунді та 320 мілісекунді, а закінчується о 00 годині 00 хвилині 03 секунді та 720 мілісекунді, таким чином визначається коли має виводитися записане в файл словосполучення чи слово. Потім саме словосполучення (умовне: сьогодні сонячно.). після цього всього йде пропуск та все знову записується для наступних слів(фраз), та цей процес триває до закінчення файлу.

Шлях до материнської папки в середині папки проекту

```
PATH_TO_VIDEO = './videos'
```

Перевіряє чи існує материнська папка

```
def create_dirs(filename_notype) -> None:
    if not os.path.exists(f'{PATH_TO_VIDEO}/'):
        os.mkdir(f'{PATH_TO_VIDEO}/')

    if not os.path.exists(f'{PATH_TO_VIDEO}/{filename_notype}'):
        os.mkdir(f'{PATH_TO_VIDEO}/{filename_notype}')
```

Визначення та запис назви відео

```
# Отримати назву відео та його розширення
def get_file_name(video_data) -> Tuple[str, str]:
    filename = f'{video_data['title']}.{video_data['stream'].subtype}'
    filename = filename.replace('?', '') # Видалити ? із назви відео
    # Назва відео без розширення файлу
    filename_notype = '.'.join(filename.split('.')[:-1])
    return filename, filename_notype
```

Та **filename_notype** використовується для маркування відео однієї групи, та для того щоб знати куди відправляти всі файли з однією назвою.

3.2 Розробка інформаційної бази

Інформаційною базою слугують папки в яких зберігаються дані, аби дані були структуровані відповідно до кожної обробки відео створюються відповідні папки з назвами відео як власними що допомагає в пошуку потрібних файлів.

Всі створені дочірні папки зберігаються в материнській папці що допомагає з пошуком відповідних файлів. Тобто структура зберігання буде нагадувати дерево в якому материнська папка корінь, дочірні папки гілки, а створювані файли листя (материнська папка → дочірня папка → файли)

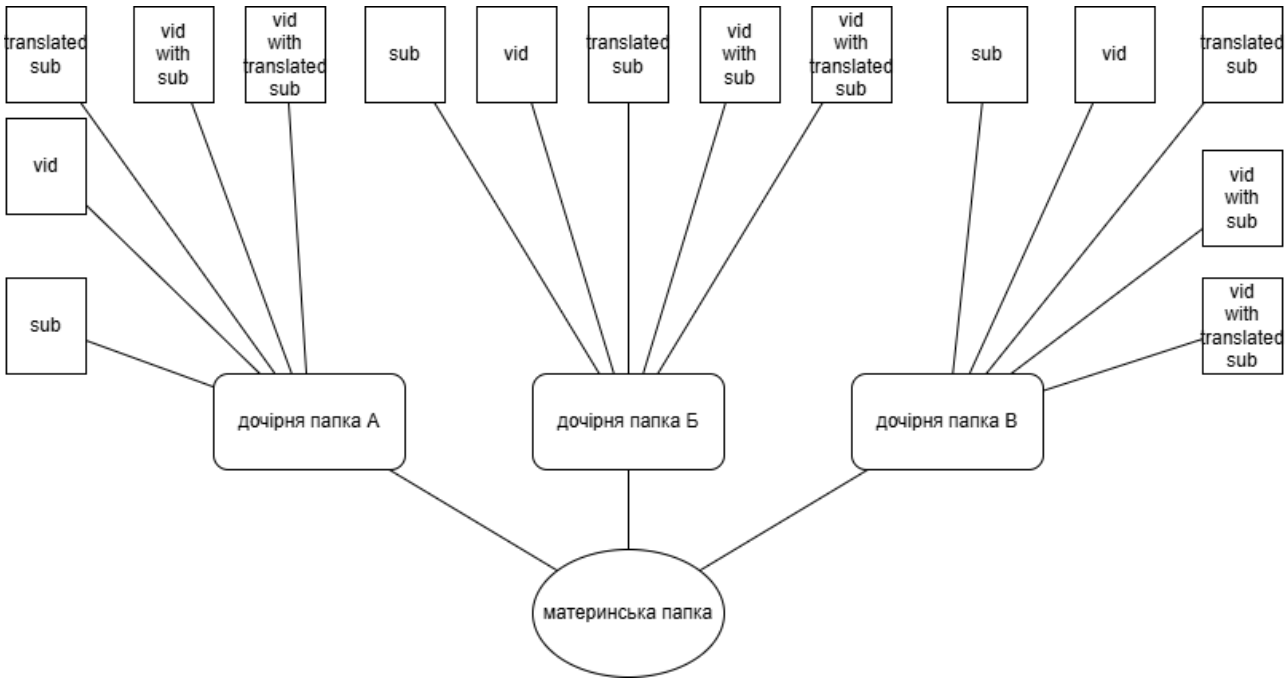

Рис.12 Зображення структури компонентів сховища та його структури

Зберігання файлів здійснюється паралельно процесу їх створення тому буде зображено код який відповідає за створення файлів в додатку.

Визначення та запис назви відео

```
# Отримати назву відео та його розширення
def get_file_name(video_data) -> Tuple[str, str]:
    filename = f'{video_data['title']}.{video_data['stream'].subtype}'
    filename = filename.replace('?', '') # Видалити ? із назви відео
    # Назва відео без розширення файлу
    filename_notype = '.'.join(filename.split('.')[:-1])
    return filename, filename_notype
```

Визначення та запис субтитрів(сказані слова перетворені на текст) відео

```
def get_transcription(filename: str) -> dict:
    try: # Транскрибування файлу
        transcript = aai.Transcriber().transcribe(filename)
        transcript.wait_for_completion()
```

```
            # Дістати субтитри
            subtitles = transcript.export_subtitles_srt()
            print("done!")

            return {
                'lang': transcript.json_response['language_code'],
                'subtitles': subtitles
            }
        except httpx.ConnectTimeout as e:
            print(f'{e}\n')
            print(f"СПРОБА ТРАНСКРИПЦІЇ НЕВДАЛА. ПОПРОБУЙТЕ
            ЩЕ РАЗ...")
```

Запис перекладених субтитрів відео в створений файл

```
#Перевірка, чи рядок містить літери (фактичний текст)
if re.search(r'[a-zA-Za-яA-ЯіІїЇєЄ]', line_stripped):
    # Це рядок із текстом, перекладаємо його
    try:
        translated_text = translator.translate(text=line_stripped)
        # Додаємо перекладений текст з тим самим форматуванням
        translated_lines.append(translated_text + '\n')
except Exception as e:
    print(f'Помилка перекладу: {e}')
    translated_lines.append(line)
else:
    # Це не текстовий рядок, залишаємо без змін
    translated_lines.append(line)
```

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для даного проекту було знайдено застосунки за подібною тематикою, та проведено аналіз на їх основі, що дало змогу обрати власний тип застосунку. Так як додаток планувався для окремої команди для обробки відео, то було обрано десктопний науково інженерний тип застосунку з нахилом на опрацювання медіа сегменту, він буде опрацьовувати, та перед цим завантажувати медіафайли. Після завантаження буде опрацьовувати завантажуваний файл, та створюватиме варіації даних та їх комбінації з витягнутим файлом включно створюючи файли що були вказані в таблиці №2.

Та перш ніж приступити до вибору певного інструментарію, було проведено аналіз у відповідності до потреб та було підібрано бібліотеки котрі можуть добре співпрацювати між собою, та мають широкий спектр функціоналу, що добре підходить для майбутнього масштабування (розширення функціоналу), та аби мав високу швидкість опрацювання. Додаток використовує відкриті бібліотеки для інтеграції з **API** котрі використовуються для визначення субтитрів. Час обробки замовлення варіюється від обсягу пам'яті яку займає відео, та збільшується прямо пропорційно до обсягу первинного (завантажуваного) файлу.

На основі вищезазначених критеріїв, для розробки програмного забезпечення було обрано такі інструменти як:

1) Мова програмування **Python** була обрана через великі кількість різносторонніх бібліотек, та їх варіативності, “чистий синтаксис” та легкість читання, що є значними перевагами при розробці складних систем, та мультиплатформеність.
2) Середовищем для написання коду було обрано **Visual Studio Code** який не споживає багато системних ресурсів, є безкоштовним, має підтримку від **Git,** має багато вбудованих бібліотек та вбудоване вікно розширень з ними.

Тепер як було описано про мову яка використовується та середовище в якому розроблявся додаток, то буде описано всі необхідні бібліотеки які застосовуються в коді,та за що вони відповідальні, а саме:

1) **os** - відповідає за створення та перевірку на те чи зараз існують такі папки як материнська та для отримання списку всіх каталогів для подальшої організації файлів.
2) **subprocess** - дає можливість паралельно запустити обробку наступного відео(дає дозвіл після команди користувача), бо за замовчуванням нічого не відбувається поки не виконується попередній процес, але ця бібліотека дозволяє умовно обійти ці обмеження.
3) **multiprocessing -** відповідає за створення паралельних процесів в системі, Тобто коли запущена обробка одного відео, то цей процес дозволяє додати обробку ше одного відео, але при цьому переходячи між процесами.
4) **pytubefix -** надає стійке підключення до **YouTube** для операцій з ним, аби можна було виконати перший та основний крок для програми, а саме завантаження відео.
5) **typing** - дозволяє точно вказати тип файлу, які повертаються, або завантажуються.
6) **assemblyai** - надає **API** для перетворення говоріння на текст (Speech-to-Text), а точніше створює з аудіо текст з точним часом коли він був сказаний впродовж відео, чи аудіо. Також має можливість розрізняти голоси за тембром, але це не використовується.
7) **httpx** - при відсутності з'єднання з **assemblyai** надішле повідомлення про це(в даному контексті).Загалом відповідає за виконання запитів да http адреси.
8) **TextClip** - компонент класу **moviepy**, який відповідає за відображення тексту на відео, його розташування, вигляд шрифт та те коли має з'явитися текст(навіль з часовою розміткою від **assemblyai,** потрібно конвертувати час).

9) **re** - займається пошуком елементів у файлі, а саме буквених символів завдяки компоненту **.search**.
10) deep_translator - займається перекладом визначеного тексту, має в собі багато типів перекладачі(було використано GoogleTranslator).
11) **tkinter** - створює вікно та користувацький інтерфейс в якому можуть бути розміщені кнопки, випадаючі списки чи інші функціональні компоненти.
12) **pillow** - бібліотека для зображення Gif анімацій на вікні при обробці, для зображення того, що певна функція в процесі.

3.4 Алгоритмізація та програмування програмних модулів

Перший компонент відповідальний за генерацію назви файлі базуючись на назві відео, тобто яка назва у відео, так і буде називатись файл.

```
def get_file_name(video_data) -> Tuple[str, str]:
    # Отримати назву відео та його розширення
    filename = f'{video_data['title']}.{video_data['stream'].subtype}'
    filename = filename.replace('?', '')  # Видалити `?` із назви відео

    # Назва відео без розширення файлу
    filename_notype = '.'.join(filename.split('.')[:-1])

    return filename, filename_notype
```

Спочатку створюється материнська папка, а потім дочірня(також матиме назву відео для організації даних, та в цій папці зберігатися файли з подібною назвою).

```
def create_dirs(filename_notype) -> None:
    if not os.path.exists(f'{PATH_TO_VIDEO}/'):
        os.mkdir(f'{PATH_TO_VIDEO}/')
```

```
        if not os.path.exists(f'{PATH_TO_VIDEO}/{filename_notype}'):
            os.mkdir(f'{PATH_TO_VIDEO}/{filename_notype}')
```

Записує визначений текст субтитрів до текстового файлу

```
def write_text_to_file(subtitles: str, path: str) -> None:
    try:
        with open(path, 'w+', encoding='UTF-8') as f:
            f.write(subtitles)
    except Exception as e:
        print(f"ПОМИЛКА ЗАПИСУ ТРАНСКРИПЦІЇ: {e}")
        raise e
```

Відповідає за переклад файлу субтитрів лише в тих рядках де є слова.

```
if re.search(r'[a-zA-Za-яA-ЯіЇїЇєЄ]', line_stripped):
    # Це рядок із текстом, перекладаємо його
    try:
        translated_text = translator.translate(text=line_stripped)
        # Додаємо перекладений текст з тим самим форматуванням
        translated_lines.append(translated_text + '\n')
    except Exception as e:
        print(f'Помилка перекладу: {e}')
        translated_lines.append(line)
else:
    # Це не текстовий рядок, залишаємо без змін
    translated_lines.append(line)
```

Відповідає за створення черги на виконання обробки компонентів замовлення.

```
# Функція обробки черги
def process_queue(queue):
```

```
        while True:
            # Отримує завдання з черги
            video_url, gui_check_boxes = queue.get()
            # Перевірка на сигнал про закінчення процесу
            if video_url is None:
                break

            # Виконати завдання
            run_main(video_url, gui_check_boxes)
```

Відправляє створений процес на виконання.

```
def start_processing_in_background(video_url, gui_check_boxes):
    # Використовує новий процес для обробки завдання
    processing_process = multiprocessing.Process(target=start_processing,
        args=(video_url, gui_check_boxes))
    processing_process.start()
```

Додавання субтитрів до відео(встановлені межі відобрадення).

```
def create_subtitle_clips(
    subtitles: pysrt.SubRipFile,
    offset: float,
    current_audio_dur: float
) -> List[TextClip]:
    text_box_x = 1000  # Ширина коробки субтитрів
    text_box_y = 1000  # Висота коробки субтитрів

    subtitle_clips = []  # Всі субтитри

    last_subtitle = len(subtitles) - 1  # Остання субтитра за списком
```

Визначення співвідношення сторін та зміна формату виведення субтитрів на відео.

```
def set_video_dementions(video_clip: VideoClip) -> VideoClip:
    # Отримання співвідношення сторін відео
    original_width, original_height = video_clip.size

    # Якщо відео 9:16(або близько до цього)
    current_ratio = original_height / original_width
    target_ratio = 16/9

    # Якщо відео не 9:16
    if not abs(current_ratio - target_ratio) < 0.01:
        # Визначення цільового розміру
        target_width = original_width
        target_height = int(target_width * (16/9))

    # Перевірте, чи потрібно обрізати ширину чи висоту
    if original_height / original_width > 16/9:  # Taller than 9:16
        # Обрізати висоту
        new_height = int(original_width * (16/9))
        y_center = original_height / 2
        video_clip = video_clip.cropped(y1=y_center - new_height/2,
        y2=y_center + new_height/2)
        # Змінити розмір до цільових розмірів
        video_clip = video_clip.resized((target_width, target_height))
    else:  # Wider than 9:16
        # Обрізати ширину
        new_width = original_height * (9/16)
        x_center = original_width / 2
        video_clip = video_clip.cropped(x1=x_center - new_width/2,
```

```
            x2=x_center + new_width/2)
            # Змінити розмір до цільових розмірів
            video_clip = video_clip.resized((target_width, target_height))

        return video_clip
```

Висновки до Розділу 3

У даному розділі було розглянуто створену систему управління даними, функції які оперують з цими даними, зображено архітектуру збереження даних та наведені зразки файлів які мають створювати при обробці загальних процесів.

Дана система забезпечує структуроване локальне збереження даних, що сприяє швидкодії проекту та потенційно має скоротити витік інформації назовні.

Був продемонстрований детальний опис структури **.srt** файлу з описом кожного елементу файлу, та його функції відповідно до потреб системи. Було відображено реалізацію зберігання даних через структуру коду, що відповідає за перевірку на наявність материнської папки, створення дочірньої папки з ім’ям таким, як назва відео яке завантажується до нього, також описано процеси які відповідають за визначення слів з відео, створення субтитрів з визначенням часу, визначення методів для створення перекладів, які є основними етапами виконання системи.

Було обґрунтовано вибір бібліотек, та описано для чого вони використовуються, обґрунтовано вибір середовища розробки та мови програмування, вказані їх переваги та яким чином вони використовуються в проекті.

та було відображено самі компоненти додатку які відповідальні за обмежений функціонал, але ключовий, щоб надати зображення готового додатку, та розуміння того що додаток готовой до експлуатації.

Загалом у цьому розділі було надано розуміння побудови системи її складових функціоналу з поданим описом складових, алгоритмів та інструментів використаних в застосунку, на яких базується додаток для дублювання

англомовних файлі.

Розділ 4. Рекомендації щодо впровадження та експлуатації системи

Додаток бажано завантажувати разом з python 3.8 та **visual studio code**, аби все було добре, потім провести інсталяцію бібліотек які розміщені в файлі **requirements.txt,** завдяки команді в терміналі **visual studio code,** яка виглядатиме наступним чином: **pip install -r requirements.txt**, можна буде завантажити всі необхідні бібліотеки, які було встановлено. Після попереньовказаних кроків, можна запускати додаток та використовувати його для створення відео з перекладеними субтитрами виконуючи замовлення.

4.1 Тестування системи

Додаток розрахований на одночасне одинацьке користування, бо додаток розроблявся як десктопний додаток, та має працювати лише з одним користувачем. Це не означає що в системи має бути лише один користувач, це означає що може бути один користувач за одним комп'ютером. Локальне збереження файлів забезпечує економію ресурсів ПК, на перекидання файлі, але натомість бере пам’ять з локального сховища під використання додатком, для використання цього виділеного простору для збереження в ньому відео файлів, та похідних файлів від процесів обробки материнського файлу.

Тестування проходило на кожному етапі впровадження нової дії чи функції що забезпечувало коректність коду, максимальне виключення помилок з системи, але через це надто багато часу йшло на розробку нових функцій, аби зразу передбачити всі винятки, потрібно багато часу та завзятість, аби помітити неполадку, та аби виправити помилку, чи додати раніше непомічене виключення до коду.

Було проведено тестування системи створення назв файлі та організації їх по файлах. Загалом спочатку визначається назва файлу який має завантажитись на обробку, потім його назва локально зберігається, та буде назва відео міститись в кожному пов'язаному файлі та дочірній папці, в якій всі файли по певному відео

будуть зберігатися. Та назва буде передаватися файлам які будуть утворюватися внаслідок обробки попередньо отриманих даних по даному відео(або ж замовленню). Модифікатори назв файлів будуть змінюватись у відповідності до того що містить файл, якщо там оригінальне відео, то й назва буде як у відео, якщо це субтитри, то назва буде як у відео, та додастся модифікатор **_sub** разом з форматом відео **.srt** що будуть відповідати за тип файлу через позначки у назві. Якщо файл перекладено, то незалежно від типу файлу, перед типом файлу, буде вказано його код мови якою було перекладено субтитри до відео.

4.2 Вимоги до апаратного та програмного забезпечення

Аби додаток запускався без проблем, та відповідав всім вимогам до системи, перед запуском застосунку необхідно відповідно підготуватись до його запуску, аби уникнути невідповідностей в версіях бібліотек чи середовища обробника. Отож має бути встановлено:

1) так як додаток розроблявся на операційній системі Windows, то для запуску даного застосунку рекомендовано також використовувати дану операційну систему, для отримання очікуваного результату кожного разу.
2) Має бути встановлений Python 3.8 або новіша версія для відповідності з використовуваними бібліотеками.
3) створити **.env** файл та додати до нього дані, для можливості визначення відео з YouTube, та визначення субтитрів(для цього необхідно додати **AAI_API,** який можна отримати на сайті **assemblyai**).
4) завантаження всіх необхідних бібліотек використовуючи файл який містить назви з усіма необхідними бібліотеками, а називається він **requirements.txt,** та щоб завантажити його, потрібно написати до терміналу **pip install -r requirements.txt**.
5) У комп’ютеру має бути процесор з частотою хоча б 2.0ГГц та з двома ядрами. операційна память 8 ГБ, бажано мати ssd - диск для швидкого завантаження відео, та обробки відео,обсяг вільної пам'яті, хоча б 50 ГБ(для створюваних файлів).

6) має бути стійке підключення до мережі інтернету, для обробки відео.
7) бажано мати графічний процесор(GPU).

4.3 Склад інсталяційного пакету

Інсталяційний пакет розробленого додатку буде містити всі необхідні елементи для встановлення додатку на комп’ютері, мінімізуючи дій від користувача.

Потрібно створити самодостатній файли, який може без додаткового встановлення python запустити застосунок, можна ще додати компоненти які відповідають за інтерфейс.

До інсталяційного пакету має входити .env файл, для того щоб користувач міг змінити параметри під себе(змінити ключ доступу, до AI що визначає слова використовуючи аудіо).

Інструкція з експлуатації в якій вказано порядок виконання дій для встановлення додатку, список мінімальних вимог до системи на якій має запускатись додаток, інструкція до налаштування системи при першому запускові,бажаний алгоритм виконання замовлення, можливості системи, та політика використання додатку, якій буде вказано, що не можна використовувати додаток для несанкціонованого завантаження відео без дозволу автора, на аби яку медіа платформу.

Висновки до Розділу 4

Було проведено процес підготовки, та тестування програмного забезпечення, провівши всі необхідні заходи для забезпечення того, аби додаток був працездатний. Розроблено однокористувацький десктопний додаток з локальним збереженням даних.

Тестування проводиться з кожним додаванням нових функцій, що забезпечило високий рівень виявлення помилок, та швидкі темпи їх усунення,

що спрощує майбутні додавання нового функціоналу з меншою кількістю непередбачуваних ситуацій, та кількості помилок.

Було визначено вимоги до апаратного, та програмного забезпечення, які забезпечать при запуску робочий додаток. Вказано які мінімальні вимоги до апаратного забезпечення, та які обов'язкові вимоги для програмного.

Додаток має бути достатньо зручним, з приємним інтерфейсом, та саме головне, щоб виконував свої функції, все з переліченого присутнє в даному продуктові.

ВИСНОВКИ

У ході виконання дипломного проекту було створено програмне забезпечення системи автоматизації дублювання англомовних відеофайлів, який автоматично може створити відео з субтитрами за вимогами поставленими користувачем. Метою побудованого проекту було створити додаток який мав би можливість сам визначати мову оригіналу, та перекладати визначений текст на обрану мову з мінімальним втручанням людини до циклу.

Поки проходила розробка додатку, то паралельно проходило тестування функціоналу, та тестування на відповідність виключень, та чи все враховано, аби не було потреби виправляти помилок на етапі здачі дипломної роботи.

Також проходило тестування на організацію ресурсів, щоб бути впевненими що при запускові додатку, та при багатьох запитам до нього, не було зупинок в обробці, та щоб інформація не накладалася одна на одну, та не змішувались файли, чи будь які дані не додались до “чужого” файлу.

Присутня функція яка відповідає за створення назв для файлів та каталогів для кращої організації по системі управління файлами, вони мають назви так щоб можна було зрозуміти який саме це тип файлу, та які операції були виконані дан ним.

В результаті було створено додаток який може за командою користувача виконати всі необхідні дії, при мінімальному редагуванні, чи навіть втручанні до процесу, що значно полегшує роботу з перекладом готових відео файлів, та додавання субтитрів до них, для поширення між людей з різних країн, та для людей з вадами слуху які не знають іноземних мов, надаючи їм змогу розширити горизонт знань, а для творців відео поширеність його відео на кілька мовних секторів, що буде стимулювати канал до популярності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Regex101 : веб-сайт. URL: https://regex101.com/ (дата звернення: 13.02.2025).
2. AssemblyAI Documentation: веб-сайт. URL: https://www.assemblyai.com/docs (дата звернення: 29.10.2024).
3. pytubefix : веб-сайт. URL: https://pytubefix.readthedocs.io/en/latest/ (дата звернення: 19.10.2024).
4. tkinter - Python interface to Tcl/Tk : веб-сайт. URL: https://docs.python.org/3/library/tkinter.html (дата звернення: 13.10.2024).
5. multiprocessing — Process-based parallelism : веб-сайт. URL: https://docs.python.org/3/library/multiprocessing.html (дата звернення: 05.10.2024).
6. Subprocess management - Python 3.13.3 documentation : веб-сайт. URL: https://docs.python.org/3/library/subprocess.html (дата звернення: 02.09.2024).
7. Python Docs : веб-сайт. URL: https://docs.python.org/3/ (дата звернення: 29.08.2024).
8. SaveFrom.net : веб-сайт. URL:https://uk.savefrom.net/ (дата звернення: 22.09.2024).
9. Безкоштовний онлайновий редактор відео | Clipchamp : веб-сайт. URL: https://clipchamp.com/uk/video-editor/ (дата звернення: 22.09.2024).
10. Онлайн-відеоредактор - Clideo : веб-сайт. URL: https://clideo.com/uk/video-editor (дата звернення: 22.09.2024).
11. Exploring the gaps in linguistic accessibility of media: The potential of automated subtitling as a solution : веб-сайт. URL: https://researchportal.helsinki.fi/en/publications/exploring-the-gaps-in-linguistic-accessibility-of-media-the-poten (дата звернення: 16.12.2024).

12. Graves, A., Mohamed, A., & Hinton, G. (2013). "Speech recognition with deep recurrent neural networks : веб-сайт. URL: https://ieeexplore.ieee.org/document/6638947 (дата звернення: 24.11.2024).
13. Improving Web Accessibility through Automatic Captioning : веб-сайт. URL: https://dl.acm.org/doi/abs/10.1145/2207016.2207053 (дата звернення: 03.12.2024).

ДОДАТКИ

Додаток А

main.py

```
from down_vid import download_video
from video_subtitle import get_transcription
from translator import get_translate
from typing import Dict, Any, Tuple
from gui import GUI
import os
import subprocess
from add_subtitle import create_subtitle_clips
from moviepy import CompositeVideoClip, TextClip, VideoClip, VideoFileClip



PATH_TO_VIDEO = './videos'

target_lang = 'uk'


def get_file_name(video_data) -> Tuple[str, str]:
    # Отримати назву відео та його розширення
    filename = f'{video_data['title']}.{video_data['stream'].subtype}'
    filename = filename.replace('?', '')  # Видалити `?` із назви відео

    # Назва відео без розширення файлу
    filename_notype = '.'.join(filename.split('.')[:-1])

    return filename, filename_notype
```

```
def create_dirs(filename_notype) -> None:
    if not os.path.exists(f'{PATH_TO_VIDEO}/'):
        os.mkdir(f'{PATH_TO_VIDEO}/')

    if not os.path.exists(f'{PATH_TO_VIDEO}/{filename_notype}'):
        os.mkdir(f'{PATH_TO_VIDEO}/{filename_notype}')


def write_text_to_file(subtitles: str, path: str) -> None:
    try:
        with open(path, 'w+', encoding='UTF-8') as f:
            f.write(subtitles)
    except Exception as e:
        print(f"ПОМИЛКА ЗАПИСУ ТРАНСКРИПЦІЇ: {e}")
        raise e


def open_folder(path: str) -> None:
    if os.name == 'nt':
        subprocess.run(['explorer', path])


def run_main(video_url: str, gui_check_boxes: Dict[str, bool]):
    # Скачати відео в папку videos з іменем та розширенням файлу як на YT
    video_data: Dict[str, Any] = download_video(video_url)

    filename, filename_notype = get_file_name(video_data)
```

```
    create_dirs(filename_notype)  # Забезпечити наявність необхідних папок

    ultimate_path = f'{PATH_TO_VIDEO}/{filename_notype}/{filename_notype}'

    print("ВІДЕО УСПІШНО ЗАВАНТАЖЕНО!")
    print(f"НАЗВА ВІДЕО: {filename_notype}")

    video_data['stream'].download(f'{PATH_TO_VIDEO}/{filename_notype}',
filename=filename)

    if gui_check_boxes.get('extract_var'):
        # Отримати транскрипцію
        transcript_data = get_transcription(f'{ultimate_path}.mp4')
        write_text_to_file(transcript_data['subtitles'], f'{ultimate_path}.srt')

        print(f"ТРАНСКРИПЦІЮ УСПІШНО ЗАПИСАНО ДО ТЕКСТОВОГО
ФАЙЛУ!")
        print('\n')

    if gui_check_boxes.get('translate_var'):
        # Перекладіть відео
        get_translate(
            source_lang=transcript_data['lang'],
            target_lang=target_lang,
            ultimate_path=ultimate_path
        )

    if gui_check_boxes.get('subtitle_var'):
        video_clip = VideoFileClip(ultimate_path + '.mp4', audio=True)
```

```
    subtitle_clips = create_subtitle_clips(
        path=ultimate_path + '.srt',
        video_size=video_clip.size
    )

    final_video = CompositeVideoClip([video_clip] + subtitle_clips)

    # Остаточне відео з субтитрами та аудіо
    final_video.write_videofile(
        ultimate_path + '_sub' + '.mp4',
        codec="libx264",
        preset='fast',
        threads=4,
        logger='bar'
    )

    video_clip.close()
    final_video.close()

    print("ЗРОБЛЕНО ВІДЕО З ОРИГІНАЛЬНИМИ СУБТИТРАМИ!")

if gui_check_boxes.get('translated_subtitle_var'):
    video_clip = VideoFileClip(ultimate_path + '.mp4', audio=True)

    subtitle_clips = create_subtitle_clips(
        path=ultimate_path + '_' + target_lang + '.srt',
        video_size=video_clip.size
    )

    final_video = CompositeVideoClip([video_clip] + subtitle_clips)
```

```
        # Експортуйте остаточне відео з субтитрами та аудіо
        final_video.write_videofile(
            ultimate_path + '_sub' + '_' + target_lang + '.mp4',
            codec="libx264",
            preset='fast',
            threads=4,
            logger='bar'
        )

        video_clip.close()
        final_video.close()

        print("ЗРОБЛЕНО ВІДЕО З ПЕРЕКЛАДЕНИМИ СУБТИТРАМИ!")

if __name__ == "__main__":
    # створює інтерфейс
    gui = GUI()

    # Запустіть цикл графічного користувацького інтерфейсу(GUI)
    gui.root.mainloop()
```

Додаток Б

Функція додавання субтитрів

```
def create_subtitle_clips(path: str, video_size: Tuple[int, int]) -> List[TextClip]:
    w, h = video_size

    subtitles = pysrt.open(path, encoding='utf-8')

    print('Відкриваємо шлях до субтитрів', path)
    print('Відображення субтитрів:')
    print(subtitles)

    text_box_x = int(w * 0.9)  # ширина сітки субтитрів
    text_box_y = int(h * 0.2)  # довжина сітки субтитрів

    font_size = int(text_box_x * 0.1)

    subtitle_clips = []  # масив субтитрів

    for subtitle in subtitles:
        # перетворення .srt часу в зрозумілий для компілятора формат
        start_time = subtitle.start.ordinal / 1000
        end_time = subtitle.end.ordinal / 1000

        duration = end_time - start_time  # Скільки часу буде відображатись текст(субтитри)

        # визначення параметрів для субтитрів
        text_clip = (TextClip(text=subtitle.text, font_size=font_size, color='blue', stroke_color='white',
```

```
                    stroke_width=int(font_size * 0.1),
font='C:\\Windows\\Fonts\\Arial.ttf', method='caption', text_align='center',
size=(text_box_x, text_box_y))
                .with_position(('center', 'bottom'))
                .with_duration(duration)
                .with_start(start_time))

        subtitle_clips.append(text_clip)

    return subtitle_clips
```

Додаток В

Функція перекладу субтитрів

```
def get_translate(
        source_lang: str,
        target_lang: str,
        ultimate_path
) -> None:
    source_lang = source_lang.split('_')[0]
    translator = GoogleTranslator(source=source_lang, target=target_lang)

    try:
        with open(f'{ultimate_path}.srt', 'r', encoding='UTF-8') as file:
            lines = file.readlines()

        translated_lines = []

        for line in lines:
            line_stripped = line.strip()

            # Перевірка, чи рядок містить літери (фактичний текст)
            if re.search(r'[a-zA-Za-яA-ЯіІїЇєЄ]', line_stripped):
                # Це рядок із текстом, перекладаємо його
                try:
                    translated_text = translator.translate(text=line_stripped)
                    # Додаємо перекладений текст з тим самим форматуванням
                    translated_lines.append(translated_text + '\n')
                except Exception as e:
                    print(f'Помилка перекладу: {e}')
                    translated_lines.append(line)
            else:
```

```
            # Це не текстовий рядок, залишаємо без змін
            translated_lines.append(line)

    output_path = f'{ultimate_path}_{target_lang}.srt'
    with open(output_path, 'w', encoding='UTF-8') as file:
        file.writelines(translated_lines)

    print(f"Субтитри перекладено та збережено у: {output_path}")

except Exception as e:
    print(f'Сталася помилка при обробці файлу: {e}')
    raise
```

Додаток Д

Функції мультизадачності

```
# Функція обробки черги
def process_queue(queue):
    from main import run_main

    while True:
        video_url, gui_check_boxes = queue.get()  # Отримує завдання з черги
        if video_url is None:  # Перевірка на сигнал про закінчення процесу
            break
        run_main(video_url, gui_check_boxes)  # Виконати завдання



def start_processing(video_url, gui_check_boxes):
    # Створює чергу
    task_queue = multiprocessing.Queue()

    # Список для зберігання процесів
    processes = []

    # Розпочати процес
    for _ in range(1):
        process = multiprocessing.Process(target=process_queue, args=(task_queue,))
        process.start()
        processes.append(process)

    # Додає URL-адресу відео до черги
    task_queue.put((video_url, gui_check_boxes))

    # Очікування на закінчення всіх процесів
```

```
    for process in processes:
        process.join()


def start_processing_in_background(video_url, gui_check_boxes, callback):
    # Використовує новий процес для обробки завдання
    processing_process = multiprocessing.Process(target=start_processing,
args=(video_url, gui_check_boxes))
    processing_process.start()
    callback(video_url)
```

Додаток Е

Інтерфейс додатку