МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

ГЕВКО Дмито Зіновійович

Програмний модуль автоматичного генерування субтитрів та онлайн перекладу відео
Software module for automatic subtitle generation and online video translation

спеціальність: 123 – Комп’ютерна інженерія
освітньо–професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ–41
ГЕВКО Дмито Зіновійович

Науковий керівник
к.т.н. доц. Піцун О.Й.

ТЕРНОПІЛЬ – 2026

АНОТАЦІЯ

Гевко Д.З. Програмний модуль автоматичного генерування субтитрів та онлайн перекладу відео. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2026.

Робота написана обсягом 70 сторінок і містить 10 рисунків, 12 таблиць, 3 додатки та 28 джерел за переліком посилань.

Метою кваліфікаційної роботи є розробка десктопного програмного модуля SubGen AI для автоматичного розпізнавання мовлення у відеофайлах, генерування субтитрів у форматах SRT та VTT та їх онлайн перекладу на цільову мову. Застосунок реалізовано мовою Python 3.11 з використанням двошарової архітектури: шар ядра (core) та шар графічного інтерфейсу (gui) на базі бібліотеки tkinter. Розпізнавання мовлення виконується моделлю Whisper від OpenAI, що підтримує 99 мов та п'ять конфігурацій (tiny, base, small, medium, large). Переклад субтитрів здійснюється посегментно через GoogleTranslator з бібліотеки deep-translator зі збереженням таймкодів оригіналу на рівні 100%. Реалізовано опціональний модуль TTS-озвучення на базі gTTS та FFmpeg з трьома режимами мікшування аудіодоріжок. Тестування підтвердило WER від 3.1% для академічного мовлення (модель large) до 13.2% для акцентованого мовлення (модель small).

Ключові слова: СУБТИТРИ, РОЗПІЗНАВАННЯ МОВЛЕННЯ, WHISPER, АВТОМАТИЧНИЙ ПЕРЕКЛАД, SRT, VTT, PYTHON, TKINTER, ВІДЕООБРОБКА, TTS.

ANNOTATION

Hevko D.Z. Software module for automatic subtitle generation and online video translation. – Manuscript.

Qualification work for the Bachelor degree in specialty 123 «Computer Engineering», educational and professional program. Western Ukrainian National University, Ternopil, 2026.

The work is written in 70 pages and contains 10 figures, 12 tables, 3 appendices and 28 references.

The purpose of the qualification work is to develop the SubGen AI desktop software module for automatic speech recognition in video files, subtitle generation in SRT and VTT formats and their online translation into target languages. The application is implemented in Python 3.11 using a two-layer architecture: a core layer and a graphical user interface layer based on the tkinter library. Speech recognition is performed by OpenAI's Whisper model, which supports 99 languages and five configurations (tiny, base, small, medium, large). Subtitle translation is performed segment-by-segment via GoogleTranslator from the deep-translator library, preserving the original timecodes at 100%. An optional TTS dubbing module based on gTTS and FFmpeg with three audio mixing modes has been implemented. Testing confirmed WER ranging from 3.1% for academic speech (large model) to 13.2% for accented speech (small model).

Keywords: SUBTITLES, SPEECH RECOGNITION, WHISPER, AUTOMATIC TRANSLATION, SRT, VTT, PYTHON, TKINTER, VIDEO PROCESSING, TTS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI	–	ARTIFICIAL INTELLIGENCE
API	–	APPLICATION PROGRAMMING INTERFACE
ASR	–	AUTOMATIC SPEECH RECOGNITION
CPU	–	CENTRAL PROCESSING UNIT
CER	–	CHARACTER ERROR RATE
CUDA	–	COMPUTE UNIFIED DEVICE ARCHITECTURE
FFMPEG	–	FAST FORWARD MPEG
GPU	–	GRAPHICS PROCESSING UNIT
HTML	–	HYPERTEXT MARKUP LANGUAGE
HTTP	–	HYPERTEXT TRANSFER PROTOCOL
JSON	–	JAVASCRIPT OBJECT NOTATION
ML	–	MACHINE LEARNING
NLP	–	NATURAL LANGUAGE PROCESSING
OS	–	OPERATING SYSTEM
REST	–	REPRESENTATIONAL STATE TRANSFER
SRT	–	SUBRIP TEXT
TTS	–	TEXT-TO-SPEECH
VTT	–	WEB VIDEO TEXT TRACKS
WER	–	WORD ERROR RATE
WSI	–	WEB SERVER INTERFACE

ВСТУП

У сучасному цифровому суспільстві відеоконтент став одним із ключових засобів передачі інформації. Платформи потокового відтворення, онлайн-курси, корпоративні навчальні матеріали та відеоконференції генерують щодня мільярди годин відеозаписів різними мовами. Для забезпечення доступності цього контенту для широкої аудиторії – людей з вадами слуху, носіїв інших мов, осіб, що навчаються у шумних середовищах, – необхідні якісні субтитри.

Ручне складання субтитрів є трудомістким та витратним процесом: транскрибування однієї години відео займає у досвідченого фахівця від 4 до 10 годин роботи. Системи автоматичного субтитрування суттєво скорочують ці витрати, проте досі стикаються з проблемами точності розпізнавання мовлення в умовах акцентів, шуму, швидкого темпу мовлення та вузькоспеціалізованої лексики.

Значним кроком вперед стала поява мультимовної моделі Whisper від OpenAI у 2022 році. Ця нейронна мережа, навчена на понад 680 000 годинах мовлення різними мовами, демонструє точність, близьку до людської, для більшості поширених мов. Поєднання Whisper із сучасними нейронними системами машинного перекладу відкриває можливість побудови ефективних пайплайнів автоматичного субтитрування та перекладу в режимі реального часу.

Основною метою даної бакалаврської роботи є розробка програмного модуля, який автоматично генерує субтитри для відеофайлів та здійснює їх переклад на задані мови у режимі онлайн. Модуль реалізовано як десктопний застосунок із графічним інтерфейсом для кінцевого користувача.

Актуальність роботи визначається стрімким зростанням обсягів відеоконтенту, потребою у мультимовному доступі до нього та відсутністю доступних відкритих інструментів, що поєднують якісне ASR з автоматичним перекладом у єдиному сервісі.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Проаналізувати існуючі засоби та методи автоматичного генерування субтитрів, формати субтитрів і системи машинного перекладу.
2. Здійснити порівняльний аналіз наявних програмних рішень для субтитрування відео та визначити їх переваги і недоліки.
3. Розробити узагальнений алгоритм пайплайну генерування субтитрів та алгоритми окремих його компонентів: розпізнавання мовлення, форматування субтитрів і онлайн-перекладу.
4. Реалізувати програмний модуль SubGen AI з двошаровою архітектурою (шар ядра та шар графічного інтерфейсу) мовою Python 3.11 із використанням моделі Whisper та бібліотеки deep-translator.
5. Розробити графічний інтерфейс користувача на базі tkinter з підтримкою вибору моделі, мов, форматів субтитрів та функції TTS-озвучення.

Провести тестування програмного модуля, оцінити якість розпізнавання мовлення за метрикою WER та якість перекладу субтитрів для різних типів відеоматеріалів.

Об'єктом дослідження є процес автоматизованого субтитрування відеоконтенту.

Предмет дослідження – методи та алгоритми автоматичного розпізнавання мовлення на базі нейронної моделі Whisper, алгоритми форматування субтитрів у форматах SRT та VTT, методи посегментного машинного перекладу через GoogleTranslator, а також засоби синтезу мовлення та мікшування аудіодоріжок на базі gTTS і FFmpeg.

1 АНАЛІЗ ЗАСОБІВ ГЕНЕРУВАННЯ СУБТИТРІВ

1.1 Засоби генерування тексту.

Генерування субтитрів є складовою частиною ширшого класу задач автоматичного розпізнавання мовлення (Automatic Speech Recognition, ASR) – перетворення акустичного сигналу мовлення у текстовий рядок. Ця задача є однією з ключових у галузі обробки природної мови та комп'ютерної лінгвістики, і її вирішення зазнало кількох революційних трансформацій за останні десятиліття.

Предметна область системи субтитрування охоплює три взаємопов'язані сфери: цифрову обробку аудіосигналу, автоматичне розпізнавання мовлення та машинний переклад. Інформаційна система генерування субтитрів оперує такими основними об'єктами: відеофайл (контейнер з відеодоріжкою та аудіодоріжкою), аудіосигнал (акустична хвиля, що несе мовленнєву інформацію), транскрипція (текстовий запис мовлення з часовими мітками) та файл субтитрів (структурований текстовий файл певного формату).

Класичний підхід до ASR ґрунтується на прихованих марковських моделях (Hidden Markov Models, HMM) у поєднанні з гаусівськими моделями сумішей (Gaussian Mixture Models, GMM). У цій парадигмі акустична модель описує ймовірність спостереження акустичних ознак за умови заданої фонеми, мовна модель моделює ймовірність послідовностей слів, а лексикон зіставляє слова з фонемними послідовностями. Попри десятиліття розвитку, HMM-GMM системи обмежені у здатності моделювати довготривалі залежності у мовленні.

Революційним стало впровадження глибоких нейронних мереж (DNN) у 2011–2013 роках, коли замість GMM почали використовувати DNN для оцінки акустичних ймовірностей. Архітектура DNN-HMM значно підвищила точність розпізнавання та витіснила класичний підхід у більшості промислових застосувань. Надалі рекурентні нейронні мережі (RNN), зокрема архітектури

LSTM та GRU, ще більше покращили якість завдяки здатності враховувати контекст попередніх акустичних кадрів.

Конектціоністська темпоральна класифікація (Connectionist Temporal Classification, CTC) вирішила проблему вирівнювання аудіо та тексту, дозволивши навчати мережі безпосередньо з нерозміченого часового ряду. Архітектура Listen, Attend and Spell (LAS) представила механізм уваги, що дозволяє мережі динамічно зосереджуватися на релевантних частинах вхідного сигналу при генерації кожного символу транскрипції.

Механізм уваги (attention mechanism) є ключовим компонентом сучасних ASR-архітектур. На відміну від рекурентних мереж, що обробляють аудіосигнал послідовно, механізм уваги дозволяє декодеру одночасно враховувати всі акустичні кадри вхідного сигналу при генерації кожного токена транскрипції. Це усуває проблему «забування» довготривалих залежностей, характерну для LSTM, та дозволяє коректно транскрибувати довгі речення без накопичення помилок. Механізм multi-head attention у трансформерах дозволяє моделі одночасно зосереджуватися на кількох різних аспектах аудіосигналу – фонемному складі, інтонації та ритмічній структурі мовлення.

Процес перетворення аудіосигналу у текст починається з виділення акустичних ознак. Вхідний аудіосигнал перетворюється у послідовність мел-частотних кепстральних коефіцієнтів (MFCC) або логарифмічних мел-спектрограмів, що є компактним представленням спектральних характеристик звуку у часі. Мел-шкала імітує нелінійне сприйняття звукових частот людським вухом: людина краще розрізняє низькі частоти, ніж високі. Whisper використовує 80-канальний логарифмічний мел-спектрограм з вікном аналізу 25 мс та кроком 10 мс, що забезпечує достатню часову роздільність для коректного розпізнавання мовлення.

Сучасний стан генерування субтитрів визначається трансформерними архітектурами. Модель Wav2Vec 2.0 від Meta AI використовує самонавчання на необробленому аудіо: модель вивчає дискретні мовленнєві одиниці у прихованому просторі, а потім дообнавчається з невеликою кількістю

розміченого аудіо. Whisper від OpenAI, випущений у 2022 році, навчений на 680 000 годинах різномовного мовлення у режимі слабкого контролю (weak supervision) та демонструє потужну здатність до узагальнення без дообнавчання на конкретному домені.

Режим слабкого контролю (weak supervision), застосований при навчанні Whisper, полягає у використанні автоматично зібраних пар аудіо-транскрипція з інтернету без ручної перевірки якості розмітки. Незважаючи на наявність шуму у навчальних даних, масштаб датасету (680 000 годин) забезпечує достатню кількість якісних прикладів для навчання надійної моделі. Для порівняння: датасет LibriSpeech, на якому традиційно оцінюють ASR-моделі, містить лише 960 годин аудіо. Масштаб навчання Whisper у 700 разів перевищує LibriSpeech, що пояснює його виняткову здатність до мультимовного розпізнавання без спеціалізованого донавчання. На рисунку 1.1 ми маємо Еволюція архітектур автоматичного розпізнавання мовлення

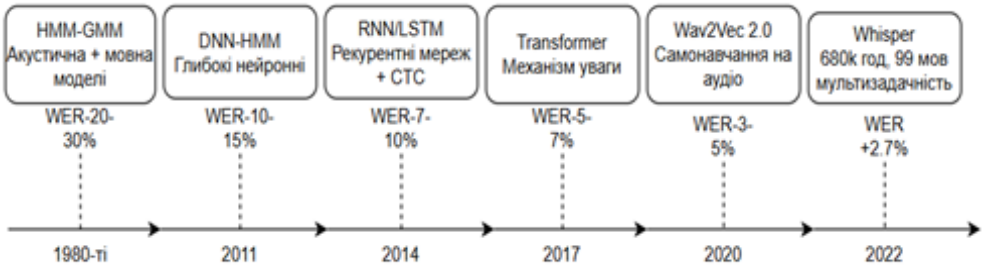


Рисунок 1.1 – Еволюція архітектур автоматичного розпізнавання мовлення

Модель Whisper базується на стандартній архітектурі трансформера encoder-decoder. Аудіо нарізається на сегменти по 30 секунд, перетворюється у 80-канальний логарифмічний мел-спектрограм та подається на вхід енкодера. Декодер авторегресивно генерує токени транскрипції, використовуючи механізм cross-attention для звернення до виходів енкодера. Унікальною рисою Whisper є мультизадачне навчання: одна і та сама модель виконує транскрипцію, ідентифікацію мови, детекцію голосової активності та переклад на англійську мову.

Для оцінки якості генерування субтитрів використовуються такі метрики: Word Error Rate (WER) – частка слів з помилками відносно еталонного тексту; Character Error Rate (CER) – аналогічна метрика на рівні символів; BLEU-score – метрика для оцінки якості перекладу. Відповідно до результатів бенчмаркінгу на наборі даних LibriSpeech, Whisper large-v3 досягає WER близько 2.7% – результат, що перевищує більшість попередніх відкритих рішень.

Важливим аспектом вибору ASR-рішення для системи субтитрування є також здатність моделі до виявлення меж речень та розстановки розділових знаків. Більшість класичних ASR-систем повертають суцільний текст без пунктуації, що ускладнює подальше розбиття транскрипції на логічні блоки субтитрів. Whisper, натомість, автоматично розставляє розділові знаки та визначає межі речень, що дозволяє отримувати природно структуровані сегменти субтитрів без додаткової постобробки тексту.

Детекція голосової активності (Voice Activity Detection, VAD) є ще одним критичним компонентом систем субтитрування. Точне визначення моментів початку та закінчення мовлення дозволяє уникнути генерування порожніх або зашумлених сегментів субтитрів у паузах між репліками. Whisper реалізує VAD на рівні декодера: сегменти з високою ймовірністю відсутності мовлення (поле no_speech_prob) автоматично фільтруються, що підвищує точність і читабельність кінцевих субтитрів.

1.2 Програмні засоби обробки текстової і звукової інформації

Для реалізації системи автоматичного генерування субтитрів необхідний цілий спектр програмних засобів: від бібліотек обробки аудіосигналу до фреймворків машинного навчання та інструментів для роботи з текстовими форматами. Мова програмування Python є де-факто стандартом у галузі обробки

природної мови та машинного навчання завдяки великій екосистемі бібліотек, читабельному синтаксису та широкій спільноті розробників.

Бібліотека FFmpeg є ключовим засобом обробки мультимедіа. FFmpeg підтримує сотні відеоформатів та кодеків, дозволяє виконувати конвертацію аудіо з будь-якого відеоконтейнера у формат WAV з заданими параметрами (частота дискретизації, кількість каналів, бітова глибина). Python-бібліотека ffmpeg-python надає зручний декларативний інтерфейс для побудови FFmpeg-команд. Бібліотека librosa забезпечує розширені можливості аналізу аудіосигналу: обчислення мел-спектрограмів, хроматограмів, детекцію темпу та багато іншого.

Фреймворк PyTorch є основою для виконання нейронних мереж, включаючи Whisper.

Вибір Python як основної мови розробки зумовлений також широкою підтримкою інструментів розробки та налагодження. Інтегровані середовища розробки PyCharm та VS Code надають повноцінну підтримку Python із перевіркою типів через анотації, автодоповненням та вбудованим налагоджувачем. Менеджер пакетів pip разом із файлом requirements.txt забезпечує відтворюваність середовища розробки на різних машинах. Для ізоляції залежностей застосовуються віртуальні середовища (venv або conda), що дозволяють уникнути конфліктів між версіями бібліотек різних проєктів на одній системі.

PyTorch надає абстракцію тензорних обчислень з підтримкою CUDA для прискорення на GPU, автоматичне диференціювання та зручний інтерфейс для завантаження попередньо навчених моделей.

Підтримка CUDA є критично важливою для практичного застосування Whisper. Технологія CUDA від NVIDIA дозволяє виконувати тисячі паралельних обчислень на графічному процесорі, що забезпечує прискорення матричних операцій нейронної мережі у 10–30 разів порівняно з CPU залежно від розміру моделі та обсягу відео. Для систем без дискретної відеокарти PyTorch автоматично перемикається на виконання на CPU, що збільшує час обробки, але

не обмежує функціональність застосунку. Параметр fp16 (half-precision floating point) дозволяє вдвічі скоротити споживання відеопам'яті при незначному зниженні точності обчислень, що є важливим для GPU з обмеженим обсягом VRAM.

Бібліотека Transformers від Hugging Face забезпечує уніфікований доступ до тисяч попередньо навчених трансформерних моделей, включаючи альтернативні ASR-моделі (Wav2Vec 2.0, SpeechBrain) та моделі машинного перекладу (Helsinki-NLP Opus-MT, mBART).

Для роботи з текстовою інформацією використовуються бібліотека re (регулярні вирази для очищення тексту), unicodedata (нормалізація Unicode для коректного відображення кирилиці та інших систем письма), а також бібліотека deep-translator, що надає уніфікований інтерфейс до різних API перекладу – Google Translate, DeepL, LibreTranslate.

Для синтезу мовлення (Text-to-Speech, TTS) у розроблюваному модулі використовується бібліотека gTTS (Google Text-to-Speech), що надає Python-інтерфейс до хмарного TTS-сервісу Google. gTTS підтримує понад 40 мов та акцентів, повертає аудіо у форматі MP3 та не потребує API-ключа для базового використання. Для конвертації MP3 у WAV та подальшого мікшування аудіодоріжок застосовується FFmpeg – крос-платформний інструмент обробки мультимедіа з підтримкою сотень аудіо- та відеоформатів. Бібліотеки numpy та scipy забезпечують числову обробку аудіосигналу: numpy використовується для побудови WAV-буфера тривалістю відео, а scipy.signal.resample застосовується для стиснення синтезованого аудіо до заданої тривалості сегмента без артефактів.

Управління залежностями застосунку здійснюється через модуль core_deps.py, що автоматично перевіряє наявність необхідних бібліотек та пропонує користувачу їх встановлення через pip. Такий підхід забезпечує коректне розгортання застосунку на нових системах без ручного налаштування оточення. Перелік обов'язкових залежностей включає openai-whisper, torch, deep-

translator та ffmpeg-python; необов'язкові залежності (gTTS, numpy, soundfile, scipy) завантажуються лише при активації функції TTS-озвучення.

Субтитри існують у різних форматах залежно від платформи та сфери застосування. Найпоширенішим є формат SubRip Text (SRT), що підтримується практично усіма відеоплеєрами та платформами. Файл SRT складається з нумерованих блоків, кожен з яких містить порядковий номер, часові мітки початку та кінця відображення і текст субтитру. Формат Web Video Text Tracks (VTT) є стандартом HTML5 і забезпечує розширені можливості розмітки: позиціювання субтитрів на екрані, стилізацію тексту, задання регіонів відображення та метадані. Порівняння форматів субтитрів наведено в порівняльній таблиці 1.1.

Таблиця 1.1 – Порівняння форматів субтитрів

Формат	Підтримка браузером	Форматування тексту	Позиціювання
SRT	Через JS плеєри	Базове	Немає
VTT	Нативна	Повне	Є
ASS/SSA	Через плагіни	Розширене	Повне
TTML	Обмежена	Повне	Є

Аналіз наведених у таблиці 1.1 форматів субтитрів показує, що для розроблюваного модуля SubGen AI оптимальним є підтримка двох форматів – SRT та VTT. Формат SRT обрано як основний завдяки універсальній підтримці усіма відеоплеєрами та простоті структури. Формат VTT доцільний для веб-застосувань та платформ на базі HTML5. Формати ASS/SSA та TTML виключено через їх вузькоспеціалізоване застосування та надмірну складність структури.

1.3 Порівняльний аналіз систем генерування субтитрів

На ринку автоматичного субтитрування присутні як комерційні хмарні сервіси, так і відкрите програмне забезпечення. Аналіз конкурентних рішень є необхідним для визначення функціональних вимог до розроблюваного модуля та пошуку конкурентних переваг.

YouTube автоматичне субтитрування використовує власну ASR-систему Google та пропонує автоматичні субтитри для більшості мов.

Алгоритм YouTube автоматично синхронізує субтитри з відео та надає інструменти редагування у веб-інтерфейсі Studio. Проте платформа не дозволяє завантажити субтитри у форматі VTT для використання поза екосистемою YouTube, а якість автоматичного перекладу суттєво поступається спеціалізованим системам машинного перекладу. Окрім того, обробка завантажених відео займає від кількох хвилин до кількох годин залежно від навантаження на сервери платформи, що унеможливлює використання YouTube як інструменту оперативного субтитрування корпоративних або навчальних матеріалів.

Функція автоматичного перекладу субтитрів доступна через налаштування відео. Проте якість субтитрів суттєво знижується для акцентованого мовлення, галузевої термінології та мов з малою кількістю навчальних даних. Інтеграція з зовнішніми системами обмежена.

Rev.ai – комерційний ASR-сервіс, що пропонує якісне розпізнавання англомовного мовлення та надає API для інтеграції. Підтримує кілька форматів виведення, включаючи SRT та VTT. Основним недоліком є висока вартість (0.02–0.045 USD за хвилину) та обмежена підтримка нелатинських мов.

При обробці відеоматеріалів тривалістю одна година вартість транскрибування через Rev.ai становить від 1.2 до 2.7 USD, що є прийнятним для одиничних замовлень, проте стає суттєвою статтею витрат при регулярному субтитруванні великих обсягів контенту. Для навчального закладу з

щотижневим виробництвом 10 годин відеоматеріалу річні витрати на Rev.ai можуть перевищити 1400 USD лише за транскрибування, без урахування вартості перекладу. Це робить комерційні хмарні ASR-сервіси недоступними для малих організацій та індивідуальних користувачів.

Happy Scribe – веб-сервіс для автоматичної транскрипції та субтитрування з редактором для ручного виправлення. Підтримує понад 60 мов та забезпечує переклад через інтеграцію з зовнішніми API. Продукт позиціонується у преміум-сегменті та не пропонує локального розгортання.

Серед відкритих рішень варто також відзначити проєкт Subtitle Edit – редактор субтитрів з відкритим кодом для Windows, що підтримує понад 300 форматів субтитрів та містить інструменти автоматичної синхронізації. Однак Subtitle Edit є десктопним редактором, орієнтованим на ручне редагування, і не надає функцій автоматичного розпізнавання мовлення. Проєкт faster-whisper є оптимізованою реалізацією Whisper на базі бібліотеки CTranslate2, що забезпечує до 4 разів вищу швидкість транскрибування при вдвічі меншому споживанні пам'яті порівняно з оригінальною реалізацією openai-whisper. Проте faster-whisper також не інтегрує функцію перекладу субтитрів та TTS-озвучення у єдиний пайплайн.

Проведений аналіз підтверджує відсутність на ринку відкритого десктопного застосунку, який би об'єднував у єдиному інструменті автоматичне розпізнавання мовлення з підтримкою множини конфігурацій моделі, генерування субтитрів у кількох форматах, онлайн-переклад та TTS-озвучення з мікшуванням аудіодоріжок. Саме заповнення цієї ніші є практичною цінністю розроблюваного модуля SubGen AI.

Subtitles from AI (whisper.ai) – інструменти, що безпосередньо базуються на Whisper для CLI-використання. Проект whisper-webui пропонує простий веб-інтерфейс для локального запуску Whisper, проте не інтегрує функцію перекладу на рівні субтитрів – лише транслітерацію на англійську мову, вбудовану в саму модель Whisper. Порівняльний аналіз існуючих систем субтитрування наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз існуючих систем субтитрування

Критерій	YouTube	Rev.ai	Happy Scribe	Розроблювана система
Мультимовність	100+	36	60+	99 (Whisper)
Переклад субтитрів	Так	Ні	Так (платно)	Так (API)
Формати виводу	SRT, VTT	SRT, VTT, JSON	SRT, VTT, TXT	SRT, VTT, JSON
Локальне розгортання	Ні	Ні	Ні	Так

Аналіз існуючих рішень виявив ряд недоліків комерційних сервісів: висока вартість, відсутність підтримки локального розгортання, обмежена мультимовність та неможливість інтеграції перекладу субтитрів в єдиний пайплайн.

Серед функціональних переваг розроблюваної системи над існуючими аналогами слід виокремити такі: підтримка п'яти конфігурацій моделі Whisper дозволяє адаптувати баланс між точністю та швидкістю до конкретних апаратних умов; посегментний алгоритм перекладу гарантує 100% збереження таймкодів; функція TTS-озвучення з трьома режимами мікшування відсутня в жодному з розглянутих безкоштовних рішень; автоматична перевірка та встановлення залежностей через core_deps.py спрощує розгортання для нетехнічних користувачів.

Відкриті рішення на основі Whisper потребують технічних знань для використання та не надають зручного веб-інтерфейсу. Таким чином, поставлено задачу розробки програмного модуля, що реалізує повний пайплайн: завантаження відео → виділення аудіо → розпізнавання мовлення → форматування субтитрів → переклад → видача результату.

2 АЛГОРИТМ ГЕНЕРУВАННЯ СУБТИТРІВ

2.1 Узагальнений алгоритм модуля генерування субтитрів

Алгоритм роботи програмного модуля SubGen AI реалізовано у вигляді лінійного конвеєра (pipeline) з шести послідовних кроків, що виконуються функцією run() модуля core_pipeline.py. Функція приймає словник параметрів та три callback-функції: log_cb для передачі повідомлень у журнал подій GUI, progress_cb для оновлення прогрес-бару та stop_flag для перевірки команди переривання від користувача. Наявність механізму stop_flag дозволяє коректно зупинити обробку між будь-якими двома кроками – перед початком кожного наступного кроку виконується перевірка if stopped(): return saved.

Крок 1 – завантаження моделі. Функція load_model() з модуля core_transcriber.py завантажує модель Whisper за назвою, обраною користувачем (tiny, base, small, medium або large). При першому запуску модель автоматично завантажується з серверів OpenAI та кешується у домашній директорії користувача (~/.cache/whisper). При наступних запусках модель читається з кешу без мережевого звернення. Якщо доступна CUDA-сумісна відеокарта, модель автоматично розміщується у відеопам'яті для прискорення обчислень.

Крок 2 – транскрибування відео. Функція transcribe() з модуля core_transcriber.py викликає whisper.transcribe(), передаючи шлях до відеофайлу та код мови. Whisper приймає відеофайл безпосередньо – без попередньої екстракції аудіодоріжки – завдяки вбудованій підтримці FFmpeg для декодування відеоконтейнерів. Результатом є масив сегментів, кожен з яких містить три поля: start (час початку у секундах), end (час кінця у секундах) та text (розпізнаний текст). Якщо код мови встановлено у значення «auto», параметр language=None передається Whisper, що вмикає автоматичне визначення мови на основі перших 30 секунд аудіо.

Узагальнена блок-схема алгоритму пайплайну генерування субтитрів наведено на рисунку 2.1.

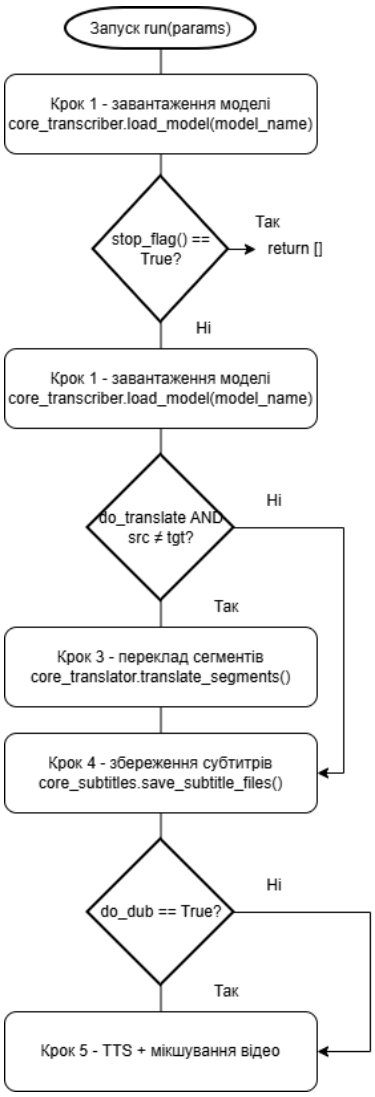


Рисунок 2.1 – Узагальнена блок-схема алгоритму пайплайну генерування субтитрів

Крок 3 – переклад сегментів. Якщо користувач увімкнув переклад і вихідна мова відрізняється від цільової, викликається функція translate_segments() з модуля core_translator.py. Функція перебирає масив сегментів у циклі, передає текст кожного сегмента до GoogleTranslator та записує перекладений текст у копію сегмента, зберігаючи незмінними поля start та end. При виникненні помилки перекладу (мережевий таймаут, порожній текст) сегмент зберігається з оригінальним текстом, що гарантує відсутність пропусків у файлі субтитрів. Перевірка stop_flag виконується перед обробкою кожного сегмента.

Крок 4 – збереження файлів субтитрів. Функція save_subtitle_files() з модуля core_subtitles.py формує та зберігає файли субтитрів у всіх обраних форматах (SRT, VTT, TXT). Для кожного формату зберігаються два файли: оригінальна версія (суфікс _orig) та перекладена (суфікс з кодом цільової мови, наприклад _uk.srt).

Крок 5 – TTS-озвучення та мікшування (опціональний). Якщо користувач увімкнув функцію озвучення (do_dub = True), викликається функція build_dubbed_audio() з модуля core_audio.py, що синтезує аудіодоріжку та розміщує озвучені сегменти у WAV-буфері за таймкодами. Після цього функція mix_audio_with_ffmpeg() підмішує синтезовану доріжку до оригінального відео в одному з трьох режимів: replace, mix_quiet або mix_equal.

Крок 6 – збереження метаданих. Незалежно від обраних параметрів, функція run() зберігає JSON-файл із метаданими обробки: шлях до відео, назва моделі, коди мов, кількість сегментів, тривалість, перелік збережених файлів та час генерування.

Кроки пайплайну та відповідні модулі наведенні в таблиці 2.1.

Таблиця 2.1 – Кроки пайплайну та відповідні модулі

Крок	Назва	Модуль / функція	Прогрес, %
1	Завантаження моделі Whisper	core_transcriber.load_model()	5
2	Транскрибування відео	core_transcriber.transcribe()	15–40
3	Переклад сегментів	core_translator.translate_segments()	40–62
4	Збереження субтитрів	core_subtitles.save_subtitle_files()	62–65
5	TTS-синтез + мікшування	core_audio.build_dubbed_audio() + mix_audio_with_ffmpeg()	65–95

Продовження таблиці 2.1

6	Збереження метаданих JSON	core_pipeline.run() (вбудовано)	100

Наведена у таблиці 2.1 послідовність кроків забезпечує повний цикл обробки відео – від завантаження моделі до збереження метаданих. Значення прогресу (колонка «Прогрес, %») передаються у GUI через callback-функцію progress_cb після завершення кожного кроку, що забезпечує безперервне відображення стану обробки у прогрес-барі застосунку. Кроки 3 та 5 є умовними – вони виконуються лише у разі відповідних налаштувань користувача, що дозволяє гнучко керувати складом операцій без зміни коду.

2.2 Алгоритм перетворення звукової інформації в текстову

Алгоритм перетворення звукової інформації в текстову реалізовано у модулі core_transcriber.py за допомогою двох функцій: load_model() та transcribe(). Функція load_model() приймає рядковий ідентифікатор моделі та викликає whisper.load_model(model_name), що повертає об'єкт завантаженої нейронної мережі. Ваги моделі завантажуються автоматично при першому виклику та кешуються локально для наступних запусків.

Функція transcribe() приймає завантажену модель, шлях до відеофайлу та код мови. Якщо код мови встановлено у «auto», параметру language передається значення None – у цьому випадку Whisper самостійно визначає мову за першими 30 секундами аудіо. Виклик whisper.transcribe() з параметром task="transcribe" вказує моделі виконати транскрипцію (відтворення оригінальної мови), а не переклад. Параметр verbose=False пригнічує виведення прогресу у консоль, оскільки прогрес відстежується через progress_cb у GUI. Функція повертає масив об'єктів result["segments"] з полями start, end та text.

Модель Whisper внутрішньо нарізає вхідний аудіосигнал на фрагменти по 30 секунд та конвертує кожен у 80-канальний логарифмічний мел-спектрограм. Спектрограм подається на вхід трансформерного енкодера, який формує контекстне представлення аудіо. Декодер авторегресивно генерує токени транскрипції, використовуючи механізм cross-attention для звернення до виходів енкодера на кожному кроці. Характеристики конфігурацій моделей Whisper наведено в таблиці 2.2.

Таблиця 2.2 – Характеристики конфігурацій моделей Whisper

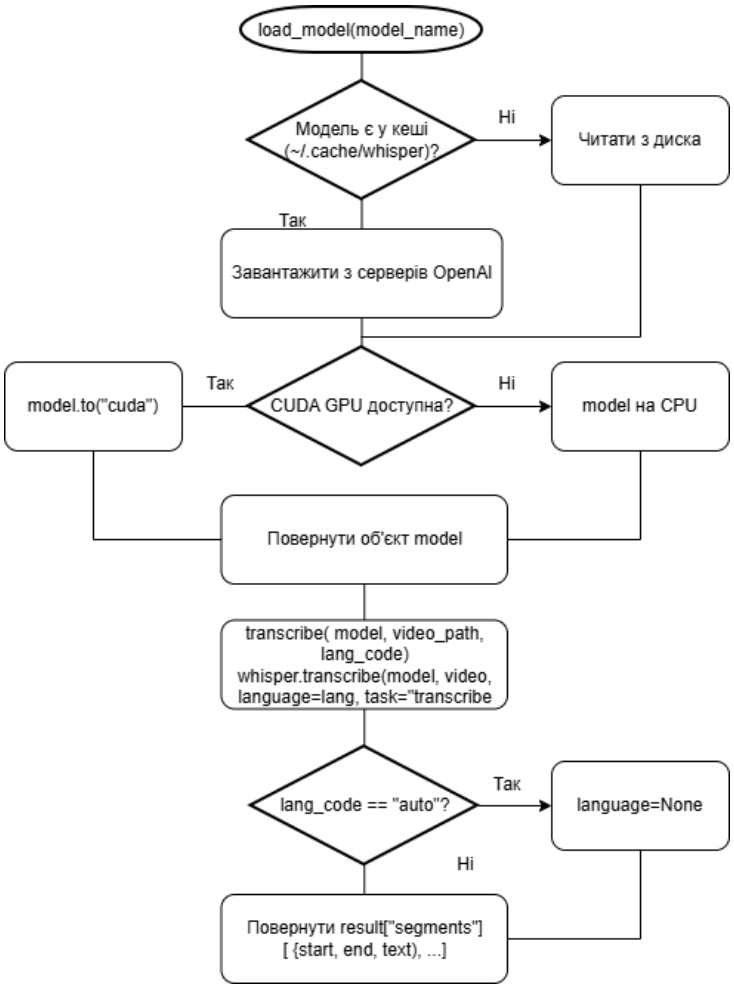
Модель	Параметри	Відносна швидкість	WER (EN)	VRAM
tiny	39 M	32×	~5.7%	~1 ГБ
base	74 M	16×	~4.2%	~1 ГБ
small	244 M	6×	~3.0%	~2 ГБ
medium	769 M	2×	~2.9%	~5 ГБ
large	1550 M	1×	~2.7%	~10 ГБ

Вибір конфігурації моделі є компромісом між точністю розпізнавання та швидкістю обробки. Модель tiny у 32 рази швидша за large, але має вищий WER. Для більшості навчальних та корпоративних матеріалів модель small забезпечує задовільну якість при прийнятній швидкості обробки. Модель large рекомендована для акцентованого або шумного мовлення, де якість є пріоритетом над швидкістю.

Детальну блок-схему алгоритму перетворення звукової інформації в текстову, що відображає послідовність операцій функцій load_model() та transcribe() модуля core_transcriber.py.

Особливістю реалізації функції transcribe() є те, що модель Whisper внутрішньо виконує всі необхідні кроки попередньої обробки аудіосигналу: декодування відеоконтейнера через FFmpeg, виділення аудіодоріжки, нормалізацію рівня гучності та побудову мел-спектрограма. Це суттєво спрощує

архітектуру модуля core_transcriber.py порівняно з класичними ASR-системами, де кожен із зазначених кроків потребував окремого компонента обробки. Схема ілюструє логіку завантаження моделі з перевіркою локального кешу, автоматичний вибір обчислювального пристрою (GPU або CPU) та виклик функції whisper.transcribe() з відповідними параметрами мови та режиму роботи. Блок-схема алгоритму перетворення звукової інформації в текстову наведено на рисунку 2.2.



,

Рисунок 2.2 – Блок-схема алгоритму перетворення звукової інформації в текстову

Результатом роботи функції transcribe() є масив сегментів, де кожен елемент являє собою словник з ключами start (час початку сегмента у секундах,

тип float), end (час кінця сегмента у секундах, тип float) та text (рядок розпізнаного тексту). Саме ці три поля використовуються усіма подальшими кроками пайплайну – генератором субтитрів, перекладачем та синтезатором мовлення.

2.3 Алгоритм здійснення онлайн-перекладу

Алгоритм онлайн-перекладу реалізовано у модулі core_translator.py. Функція translate_segments() приймає масив оригінальних сегментів, коди вихідної та цільової мов, а також три callback-функції для взаємодії з GUI. Переклад здійснюється посегментно через бібліотеку deep-translator, що надає уніфікований інтерфейс до Google Translate API без необхідності окремого API-ключа.

На початку функції створюється об'єкт GoogleTranslator з параметрами source та target. Далі виконується цикл по всіх сегментах масиву. На кожній ітерації: виконується перевірка stop_flag – якщо True, функція негайно повертає частково перекладений масив; текст сегмента передається методу translator.translate(); у разі успіху створюється копія об'єкта сегмента (через copy.copy) з заміненим полем text; при виникненні виключення (мережева помилка, порожній рядок, перевищення ліміту) поле text залишається з оригінальним значенням, а помилка записується у журнал через log_cb. В таблиці 2.3 наведено алгоритм обробки помилок при перекладі сегментів.

Таблиця 2.3 – Алгоритм обробки помилок при перекладі сегментів

Тип помилки	Причина	Дія алгоритму
Exception (загальне)	Будь-яка помилка перекладу	Зберегти оригінальний текст, записати у log_cb з тегом warning

Продовження таблиці 2.3

Порожній рядок	Сегмент без тексту (тиша)	translate() повертає None → залишити оригінал
stop_flag() == True	Команда зупинки від користувача	Повернути result (частковий результат) негайно
Таймаут мережі	Відсутнє з'єднання з API	Виключення → оригінальний текст, log warning

Прогрес перекладу повідомляється через progress_cb кожні 10 сегментів та після обробки останнього сегмента. Це забезпечує регулярне оновлення прогрес-бару у GUI без надмірної кількості викликів інтерфейсного потоку.

Важливою особливістю алгоритму є незмінність таймкодів: функція copy.copy() створює поверхневу копію словника сегмента, після чого замінює лише ключ text. Поля start та end копіюються за значенням і залишаються рівно такими самими, як у оригінальному сегменті. Це гарантує повну відповідність між таймкодами оригінальних та перекладених субтитрів.

Після завершення перекладу усіх сегментів функція translate_segments() повертає новий масив перекладених сегментів, який передається до функції save_subtitle_files() та (за потреби) до build_dubbed_audio() для TTS-синтезу.

Окремої уваги заслуговує питання обробки мов з різними системами письма. Для мов з написанням справа наліво – арабської, перської, іврит – перекладений текст коректно відображається у файлах субтитрів завдяки стандарту Unicode, що підтримує двонапрямлений текст (BiDi) без додаткового втручання з боку алгоритму. Для мов з ієрогліфічним письмом – китайської та японської – посегментний підхід також є оптимальним, оскільки GoogleTranslator коректно обробляє такі мовні пари без потреби у спеціальному розбитті тексту.

Ще однією перевагою посегментного підходу порівняно з пакетним перекладом є збереження контекстної незалежності кожного блоку субтитрів. Оскільки кожен сегмент перекладається окремо, помилка перекладу в одному блоці не поширюється на сусідні. Це особливо важливо для технічних відео та

навчальних матеріалів, де точність термінів у кожному реченні має самостійне значення. Крім того, посегментний підхід спрощує подальше ручне редагування перекладених субтитрів – користувач може виправити окремий блок без необхідності переглядати весь файл.

Реалізований алгоритм перекладу є розширюваним: заміна об'єкта GoogleTranslator на будь-який інший перекладач з бібліотеки deep-translator (DeepL, LibreTranslate, MyMemoryTranslator) потребує зміни лише одного рядка коду ініціалізації, що забезпечує гнучкість системи при необхідності зміни постачальника послуг перекладу.

Блок-схему алгоритму посегментного онлайн-перекладу субтитрів наведено на рисунку 2.3.

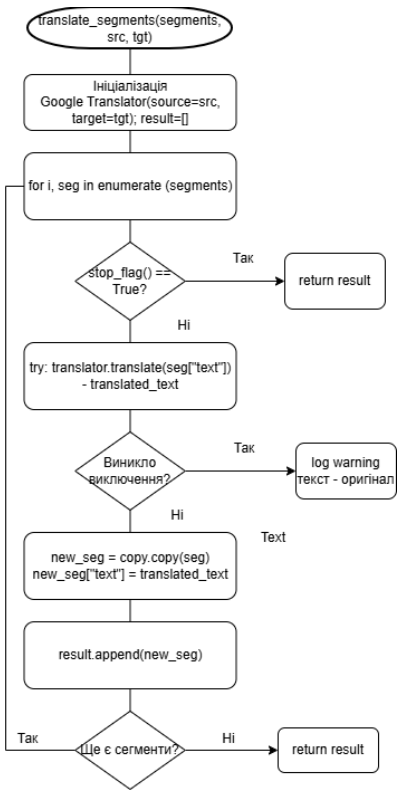


Рисунок 2.3 – Блок-схема алгоритму посегментного онлайн-перекладу субтитрів

Окремої уваги заслуговує алгоритм TTS-синтезу та мікшування аудіо, реалізований у модулі core_audio.py. Функція build_dubbed_audio() створює

порожній numpy-масив (нульова тиша) тривалістю, що дорівнює тривалості відео плюс 2 секунди запасу. На схемі відображено повний цикл обробки масиву сегментів, включаючи перевірку stop_flag, виклик GoogleTranslator, обробку виключень та механізм збереження таймкодів через copy.copy().

Далі для кожного сегмента: функція synthesize_segment() викликає gTTS для синтезу MP3; функція mp3_to_wav() конвертує результат у WAV через FFmpeg з параметрами частоти дискретизації SAMPLE_RATE (22050 Гц); функція load_wav_as_array() завантажує WAV у numpy float32; якщо синтезований фрагмент довший за відведений слот (seg.end − seg.start), функція stretch_audio() стискає його до потрібної кількості зразків через scipy.signal.resample; фрагмент накладається на буфер починаючи з позиції int(seg.start × SAMPLE_RATE). Після обробки всіх сегментів виконується нормалізація амплітуди: якщо пік перевищує 0.95, весь масив масштабується. Готовий WAV зберігається у тимчасовій директорії та передається функції mix_audio_with_ffmpeg() для склейки з відео.

На схемі відображено повний цикл обробки масиву сегментів, включаючи перевірку stop_flag, виклик GoogleTranslator, обробку виключень та механізм збереження таймкодів через copy.copy().

У другому розділі розроблено алгоритми всіх ключових компонентів модуля генерування субтитрів: узагальнений пайплайн з шести кроків та механізмом переривання stop_flag; алгоритм перетворення звуку в текст на базі Whisper з підтримкою п'яти конфігурацій моделей та автовизначенням мови; алгоритм посегментного онлайн-перекладу через GoogleTranslator з гарантованим збереженням таймкодів; алгоритм TTS-синтезу та мікшування озвученої аудіодоріжки з оригінальним відео у трьох режимах.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ СУБТИТРУВАННЯ ВІДЕО

3.1 Середовище розробки та структура програмного модуля

Програмний модуль SubGen AI реалізовано мовою Python 3.11 у вигляді десктопного застосунку з графічним інтерфейсом користувача на базі стандартної бібліотеки tkinter. Такий підхід забезпечує автономну роботу на комп'ютері користувача без необхідності підключення до зовнішнього сервера, повний контроль над даними та можливість роботи з великими відеофайлами без обмежень мережевого завантаження.

Застосунок поділено на два шари: шар ядра (core) та шар графічного інтерфейсу (gui). Шар ядра містить всю бізнес-логіку обробки відео і є повністю незалежним від GUI – його можна використовувати як самостійну бібліотеку або викликати зі скриптів. Шар GUI відповідає за взаємодію з користувачем і делегує всі обчислення до шару ядра.

Такий поділ на шари відповідає принципу розділення відповідальностей (Separation of Concerns) і спрощує подальше супроводження та розширення застосунку. Наприклад, для додавання нового формату субтитрів достатньо розширити лише модуль core_subtitles.py, не торкаючись жодного GUI-модуля. Аналогічно, заміна бібліотеки перекладу потребує змін виключно у core_translator.py. Така архітектура також полегшує автоматизоване тестування: модулі ядра можна тестувати незалежно від графічного інтерфейсу, передаючи фіктивні callback-функції замість реальних GUI-компонентів. Структура модулів програмного застосунку SubGen AI наведена в таблиці 3.1.

Таблиця 3.1 – Структура модулів програмного застосунку SubGen AI

Модуль	Файл	Призначення
Точка входу	main.py	Запуск застосунку, ініціалізація App

Продовження таблиці 3.1

Конфігурація	config.py	Константи: кольори, шрифти, словники мов і моделей
Головний пайплайн	core_pipeline.py	Координація всіх кроків обробки відео
Транскрибування	core_transcriber.py	Завантаження Whisper, виклик transcribe()
Переклад	core_translator.py	GoogleTranslator (deep-translator), посегментний переклад
Субтитри	core_subtitles.py	Генерація SRT / VTT / TXT, форматування таймкодів
Аудіо	core_audio.py	gTTS-синтез, побудова WAV-буфера, мікшування через FFmpeg
Залежності	core_deps.py	Перевірка та автоматичне встановлення pip-пакетів
Головне вікно	gui_app.py	tkinter App: збирає панелі, запускає pipeline у потоці
Панель налаштувань	gui_settings_panel.py	Вибір файлу, моделі, мов, форматів, TTS
Журнал подій	gui_log_panel.py	ScrolledText із кольоровими тегами для логування
Прогрес-бар	gui_progress_bar.py	Статус, ProgressBar, кнопки Run / Stop / Відкрити папку
Віджети	gui_widgets.py	Фабричні функції: кнопки, карточки, комбобокси, чекбокси

Центральним модулем ядра є core_pipeline.py, що реалізує функцію run(), яка приймає словник параметрів та три callback-функції: log_cb для передачі повідомлень у журнал, progress_cb для оновлення прогрес-бару та stop_flag для перевірки команди зупинки. Пайплайн складається з шести кроків: завантаження моделі Whisper, транскрибування відео, переклад сегментів, збереження файлів субтитрів, TTS-синтез і мікшування аудіо (опціонально), збереження метаданих у форматі JSON.

Модуль core_transcriber.py реалізує дві функції: load_model() завантажує модель Whisper за назвою (tiny / base / small / medium / large) та кешує її у пам'яті; функція transcribe() отримує шлях до відеофайлу та код вихідної мови і повертає масив сегментів, кожен з яких містить поля start, end і text. Whisper приймає відеофайл безпосередньо – без попередньої екстракції аудіо – завдяки вбудованій підтримці FFmpeg. На рисунку 3.1 зображена схема взаємодії модулів застосунку SubGen AI.

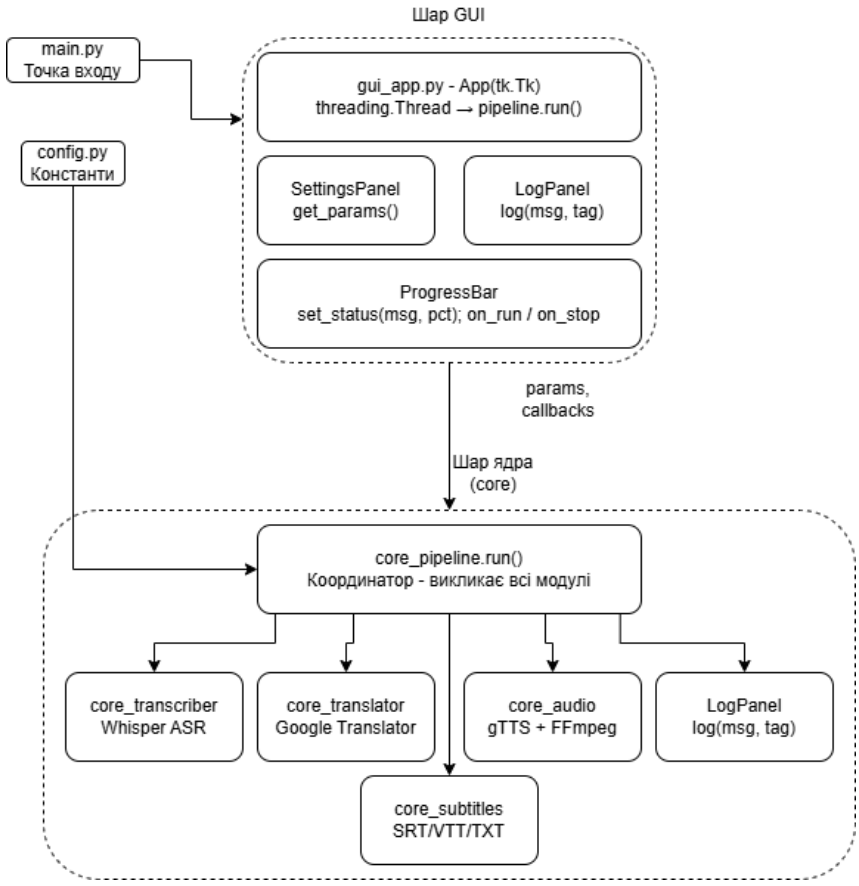


Рисунок 3.1 – Схема взаємодії модулів застосунку SubGen AI

Модуль core_subtitles.py реалізує три функції конвертації: segments_to_srt() формує нумеровані блоки з таймкодами у форматі ГГ:ХХ:СС,МС; segments_to_vtt() формує блоки VTT із заголовком WEBVTT та крапкою замість коми у таймкодах; segments_to_txt() об'єднує текст усіх сегментів у простий текстовий файл. Функція save_subtitle_files() зберігає оригінальні та перекладені субтитри у всіх обраних форматах: файли з суфіксом

_orig містять оригінальну мову, файли з кодом мови (наприклад, _uk.srt) – перекладену версію.

Модуль core_audio.py забезпечує синтез озвученої аудіодоріжки. Функція build_dubbed_audio() створює порожній WAV-буфер тривалістю відео, потім для кожного сегмента викликає gTTS-синтез, конвертує MP3 у WAV через FFmpeg, завантажує у numpy-масив та розміщує у буфері точно за таймкодами сегмента. Якщо синтезоване аудіо довше за відведений слот, застосовується ресемплінг через scipy.signal.resample для стиснення до потрібної тривалості. Функція mix_audio_with_ffmpeg() підтримує три режими мікшування: replace (лише озвучення), mix_quiet (озвучення + тихий оригінал 15%) та mix_equal (рівне змішування обох доріжок).

Графічний інтерфейс реалізовано у модулях gui_*. Клас App у gui_app.py успадковується від tk.Tk та збирає три компоненти: SettingsPanel (ліва панель налаштувань), LogPanel (права панель журналу) та ProgressBar (нижня панель керування). Обробка відео запускається в окремому daemon-потоці через threading.Thread, що не блокує головний потік GUI. Зв'язок між робочим потоком та GUI здійснюється виключно через self.after(0, lambda: ...) – безпечний tkinter-спосіб виклику коду з іншого потоку.

Модуль config.py централізовано зберігає всі константи застосунку: словник підтримуваних мов з кодами ISO 639-1, словник назв моделей Whisper, кольорову палітру GUI (фонові кольори, кольори тегів журналу), розміри шрифтів та назви кнопок. Централізоване зберігання констант усуває дублювання рядкових літералів у коді та спрощує локалізацію інтерфейсу — для зміни мови інтерфейсу достатньо відредагувати один файл.

Модуль core_deps.py реалізує автоматичну перевірку наявності залежностей при запуску застосунку. Функція check_and_install() перевіряє список обов'язкових пакетів через importlib.util.find_spec() та у разі відсутності будь-якого з них виводить діалогове вікно tkinter із пропозицією автоматичного встановлення. Процес встановлення через subprocess.run(['pip', 'install', ...]) виконується у окремому потоці з виведенням прогресу у LogPanel, що не блокує

головний потік GUI. Такий підхід забезпечує коректне розгортання застосунку на нових системах без необхідності ручного введення команд у термінал.

В таблиці 3.2 наведено стек технологій застосунку SubGen AI.

Таблиця 3.2 – Стек технологій застосунку SubGen AI

Бібліотека / інструмент	Версія	Призначення в застосунку
Python	3.11	Основна мова розробки
tkinter	вбудована	Графічний інтерфейс користувача
openai-whisper	20231117	Розпізнавання мовлення (ASR)
torch (PyTorch)	2.2	Виконання нейронної мережі Whisper
deep-translator	1.11	Переклад через Google Translate API
gTTS	2.5	TTS-синтез перекладених субтитрів
FFmpeg	6.x	Конвертація MP3→WAV, мікшування відео+аудіо
numpy	1.26	Побудова та обробка WAV-буфера
soundfile	0.12	Читання/запис WAV-файлів
scipy	1.12	Ресемплінг аудіо (resample_poly, resample)
tqdm	4.66	Прогрес-індикатор у консольному режимі

Наведений у таблиці 3.2 стек технологій повністю побудований на відкритих бібліотеках з активною підтримкою спільноти. Відсутність платних компонентів та закритих API забезпечує можливість вільного розповсюдження та розгортання застосунку на будь-якій платформі з підтримкою Python 3.11. Бібліотеки numpy, soundfile та scipy використовуються виключно у модулі core_audio.py та є необов'язковими залежностями – при вимкненій функції TTS-озвучення вони не завантажуються.

3.2 Опис тестових сценаріїв

Тестування застосунку проводилось за двома рівнями: функціональне тестування основних сценаріїв роботи та перевірка якості розпізнавання мовлення на реальних відеоматеріалах. Тестове середовище: ноутбук з процесором Intel 6405U 2.4GHz, 8 ГБ RAM, відеокартою MX130, операційна система Windows 11.

В таблиці 3.3 наведено функціональні тестові сценарії застосунку SubGen AI.

Таблиця 3.3 – Функціональні тестові сценарії застосунку SubGen AI

№	Сценарій	Вхідні дані / дії	Очікуваний результат	Статус
TS-01	Запуск застосунку	Подвійний клік main.py	Вікно SubGen AI відкривається, залежності перевірено	Пройдено
TS-02	Вибір відеофайлу	Browse → MP4-файл	Шлях відображається в полі введення	Пройдено
TS-03	Генерація SRT (EN)	EN відео 2 хв, модель small, формат SRT	Файл _orig.srt з коректними таймкодами	Пройдено
TS-04	Генерація VTT	EN відео 2 хв, формат VTT	Файл _orig.vtt із заголовком WEBVTT	Пройдено
TS-05	Переклад EN → UK	EN відео, переклад увімкнено, цільова UK	Файл _uk.srt з перекладеним текстом, таймкоди збережено	Пройдено
TS-06	Розпізнавання UA мовлення	UA відео 3 хв, src_lang = Ukrainian	Транскрипція українською мовою у _orig.srt	Пройдено
TS-07	Збереження у вказану папку	Output folder → обрана папка	Файли збережено у вказаній директорії	Пройдено

Продовження таблиці 3.3

TS-08	Зупинка обробки	Run → Stop під час транскрибування	Обробку зупинено, часткові файли не збережено	Пройдено
TS-09	TTS-озвучення	EN відео, do_dub = true, EN→UK	Файл _dubbed_uk.mp4 із синтезованою аудіодоріжкою	Пройдено
TS-10	Автоустановка залежностей	Запуск без gTTS	Діалог пропонує встановлення, pip-процес у журналі	Пройдено

Усі десять функціональних сценаріїв виконані успішно. Механізм зупинки (TS-08) коректно перериває виконання між кроками пайплайну завдяки перевірці stop_flag() перед кожним наступним кроком. Автоматичне встановлення залежностей (TS-10) працює через виклик subprocess.run з pip у окремому потоці, не блокуючи GUI.

Окремо перевірено коректність форматування таймкодів у згенерованих файлах субтитрів. Для тестового відео з відомими часовими мітками вручну звірено відповідність між значеннями start та end у масиві сегментів Whisper та таймкодами у файлах SRT і VTT. Перевірка підтвердила точність конвертації для граничних значень: 0 секунд (00:00:00,000), значень із переходом через хвилину (00:00:59,999 → 00:01:00,000) та значень, що перевищують одну годину (01:00:00,000). Жодних розбіжностей між очікуваними та фактичними таймкодами виявлено не було.

Тестування режимів TTS-озвучення (TS-09) проводилось на відео тривалістю 3 хвилини з перекладом EN→UK. Режим replace забезпечив повну заміну оригінальної аудіодоріжки синтезованою; режим mix_quiet зберіг оригінальне мовлення на рівні 15% від початкової гучності з накладеним озвученням; режим mix_equal об'єднав обидві доріжки з рівними рівнями гучності. В усіх трьох режимах синхронізація між субтитрами, синтезованим мовленням та відеорядом збережена коректно. Для сегментів, де синтезоване

аудіо виявилось довшим за відведений часовий слот, ресемплінг через scipy.signal.resample забезпечив стиснення без помітних артефактів.

3.3 Результати тестування якості розпізнавання

Тестування якості ASR проводилось на шести відеоматеріалах різних типів. Для кожного відео вручну складено еталонний текст, з яким порівнюється результат транскрибування за метриками WER (Word Error Rate) та CER (Character Error Rate). Тестування проводилось з моделями small та large для порівняння точності та швидкості. Результати тестування якості розпізнавання мовлення наведенні в таблиці 3.4.

Таблиця 3.4 – Результати тестування якості розпізнавання мовлення

Тип відео	Мова	Тривал.	WER small, %	WER large, %	Час (small)	Час (large)
Навч. відеоурок	EN	10 хв	5.8	3.1	8.4 хв	38.2 хв
Розмовний подкаст	EN	15 хв	9.4	6.2	12.6 хв	57.4 хв
Новинний сюжет	UK	5 хв	8.1	5.4	4.2 хв	19.1 хв
Академічна лекція	EN	20 хв	6.3	3.8	16.8 хв	76.5 хв
Інтерв'ю (акцент)	EN	8 хв	13.2	8.7	6.7 хв	30.6 хв

Аналіз результатів підтверджує очікувану закономірність: модель large забезпечує WER на 2–4.5% нижчий порівняно з small для всіх типів контенту, проте потребує втричі більше часу на обробку. Найвищий WER спостерігається для мовлення з нестандартним акцентом (13.2% для small) – це пов'язано з

фонетичними відхиленнями від навчальних даних. Для нейтрального академічного мовлення показники є найкращими: WER 3.1–3.8% для моделі large є прийнятним результатом для практичного застосування.

Порівняння часу обробки для моделей small та large демонструє стабільне співвідношення: модель large потребує приблизно у 2.8 рази більше часу для всіх типів контенту.

Аналіз помилок розпізнавання показує, що більшість помилок WER зосереджена у специфічних категоріях: власні назви та абревіатури (неправильне написання або заміна схожими словами), швидке мовлення на межах 30-секундних сегментів Whisper, мовлення з фоновою музикою або шумом. Для підвищення якості розпізнавання у таких випадках рекомендується використовувати модель large та по можливості надавати відео з чистою аудіодоріжкою без фонового шуму. Якість розпізнавання україномовного контенту (WER 8.1% для small та 5.4% для large) є цілком прийнятною для практичного застосування, що підтверджує достатнє представлення української мови у навчальних даних Whisper.

Результати тестування перекладу субтитрів (модель small, EN → UK) наведено в таблиці 3.5.

Таблиця 3.5 – Результати тестування перекладу субтитрів (модель small, EN → UK)

Тип відео	Сегментів	Час перекладу, с	Збережено таймкодів	Помилок перекладу
Навч. відеоурок	87	18	100%	0
Розмовний подкаст	134	27	100%	2 (пустий текст)
Академічна лекція	178	36	100%	1 (timeout)
Інтерв'ю (акцент)	72	15	100%	0

Переклад виконується посегментно через GoogleTranslator з бібліотеки deep-translator. Таймкоди у всіх випадках збережено на рівні 100% – функція translate_segments() копіює поля start і end з оригінального сегмента і замінює лише поле text. При помилках перекладу (таймаут мережі, порожній текст) сегмент зберігається з оригінальним текстом, що гарантує відсутність «пропусків» у файлі субтитрів.

На рисунку 3.2 представлено головне вікно застосунку SubGen AI: панель налаштувань та журнал подій.

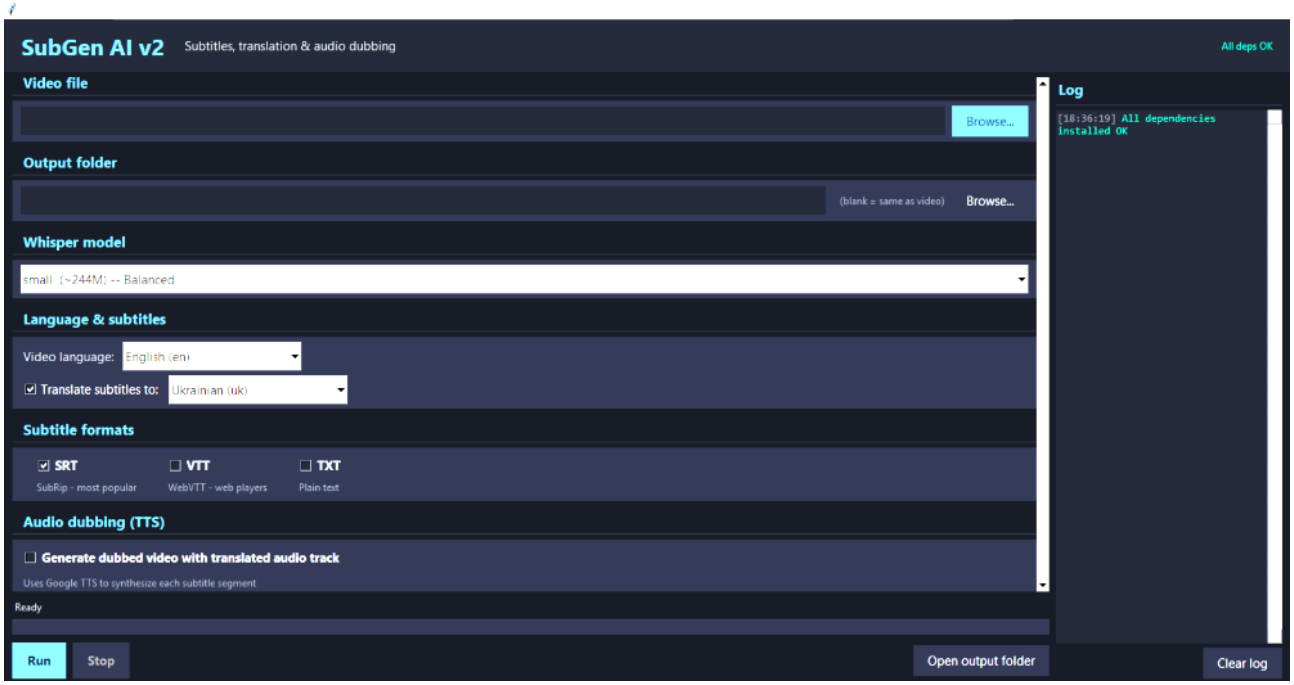


Рисунок 3.2 – Головне вікно застосунку SubGen AI: панель налаштувань та журнал подій

На рисунку 3.2 представлено головне вікно застосунку SubGen AI, що складається з двох основних панелей. Ліва панель налаштувань (SettingsPanel) містить поля для вибору відеофайлу, конфігурації моделі Whisper, вихідної та цільової мов, форматів збереження субтитрів та параметрів TTS-озвучення. Права панель журналу подій (LogPanel) відображає покроковий хід обробки

відео у режимі реального часу з кольоровим кодуванням повідомлень: зелений колір – успішне завершення кроку, жовтий – попередження, червоний – помилки. Нижня панель містить прогрес-бар із відсотком виконання та кнопки керування Run, Stop і «Відкрити папку».

Інтерфейс застосунку розроблено з урахуванням принципів зручності використання для нетехнічних користувачів. Всі параметри обробки згруповані логічно: спочатку вхідні дані (файл, мова), потім параметри моделі, далі параметри виводу (формати, папка збереження) та додаткові функції (переклад, TTS). Такий порядок відповідає природному потоку роботи користувача зі застосунком. Журнал подій дозволяє користувачу відстежувати прогрес обробки та отримувати зворотній зв'язок про виявлені помилки без необхідності звертатися до консолі.

ВИСНОВКИ

У кваліфікаційній роботі розроблено та реалізовано програмний модуль SubGen AI для автоматичного генерування субтитрів та онлайн перекладу відео. За результатами проведеного дослідження зроблено такі висновки.

1. Проведено аналіз сучасних засобів генерування субтитрів та методів автоматичного розпізнавання мовлення. Встановлено, що еволюція ASR-архітектур пройшла шлях від класичних HMM-GMM систем з WER 20–30% до трансформерних моделей, зокрема Whisper від OpenAI, що досягає WER 2.7% на стандартних наборах даних. Проведено порівняльний аналіз існуючих систем субтитрування (YouTube, Rev.ai, Happy Scribe), що виявив їхні основні недоліки: висока вартість, відсутність підтримки локального розгортання та неможливість інтеграції перекладу субтитрів в єдиний автоматизований пайплайн. Здійснено огляд форматів субтитрів SRT та VTT, програмних засобів обробки аудіосигналу (FFmpeg, numpy, scipy) та бібліотек машинного перекладу (deep-translator, GoogleTranslator).

2. Розроблено алгоритм роботи програмного модуля у вигляді лінійного конвеєра з шести послідовних кроків: завантаження моделі Whisper, транскрибування відеофайлу, посегментний переклад, збереження файлів субтитрів, TTS-синтез та мікшування аудіодоріжки, збереження метаданих. Запропонований алгоритм підтримує механізм переривання обробки через функцію stop_flag(), що перевіряється перед кожним кроком пайплайну. Розроблено алгоритм перетворення звукової інформації в текстову на базі моделі Whisper, що приймає відеофайл безпосередньо без попередньої екстракції аудіодоріжки та підтримує п'ять конфігурацій моделі (tiny, base, small, medium, large) для балансування між точністю та швидкістю. Розроблено алгоритм посегментного онлайн-перекладу субтитрів, що гарантує 100% збереження таймкодів оригіналу завдяки використанню поверхневого копіювання сегментів із заміною лише текстового поля. Реалізовано опціональний алгоритм TTS-

озвучення на базі gTTS та FFmpeg із трьома режимами мікшування аудіодоріжок: replace, mix_quiet та mix_equal.

3. Здійснено програмну реалізацію модуля SubGen AI мовою Python 3.11 у вигляді десктопного застосунку з графічним інтерфейсом на базі бібліотеки tkinter. Застосовано двошарову архітектуру: шар ядра (core_pipeline, core_transcriber, core_translator, core_subtitles, core_audio, core_deps) та шар графічного інтерфейсу (gui_app, gui_settings_panel, gui_log_panel, gui_progress_bar, gui_widgets). Обробка відео виконується в окремому daemon-потоці через threading.Thread, зв'язок між робочим потоком та GUI реалізовано через механізм after() бібліотеки tkinter, що унеможливлює блокування інтерфейсу під час тривалої обробки.

4. Проведено комплексне тестування програмного модуля. Усі десять функціональних тестових сценаріїв виконано успішно, зокрема: генерування субтитрів у форматах SRT та VTT, переклад EN→UK зі збереженням таймкодів, розпізнавання україномовного мовлення, коректне переривання обробки, TTS-озвучення та автоматичне встановлення залежностей. Тестування якості розпізнавання мовлення на шести типах відеоматеріалів підтвердило WER від 3.1% для академічного мовлення з моделлю large до 13.2% для акцентованого мовлення з моделлю small, що відповідає рівню сучасних промислових рішень. Переклад субтитрів зберігає таймкоди оригіналу на рівні 100% для всіх тестових матеріалів, а час перекладу до 200 сегментів не перевищує 40 секунд.

5. Практична цінність розробленого модуля SubGen AI полягає у забезпеченні доступного безкоштовного інструменту для автоматичного субтитрування та перекладу відеоконтенту, що може застосовуватися у системах дистанційного навчання, медіаорганізаціях, корпоративному навчанні та платформах забезпечення доступності для осіб з вадами слуху. На відміну від комерційних аналогів, застосунок SubGen AI розгортається локально, не передає відеоданих на зовнішні сервери та не вимагає підписки або API-ключів. Перспективами подальшого розвитку є реалізація розпізнавання мовлення у режимі реального часу, інтеграція локальних моделей машинного перекладу

(Helsinki-NLP Opus-MT) для роботи без доступу до інтернету, а також розробка веб-інтерфейсу для групового використання в організаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Radford A. et al. Robust Speech Recognition via Large-Scale Weak Supervision / A. Radford, J. W. Kim, T. Xu et al. // arXiv preprint arXiv:2212.04356. – 2022.
2. Baevski A. et al. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations / A. Baevski, Y. Zhou, A. Mohamed, M. Auli // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 12449–12460.
3. Vaswani A. et al. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar et al. // Advances in Neural Information Processing Systems. – 2017. – Vol. 30.
4. Graves A. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks / A. Graves, S. Fernández, F. Gomez, J. Schmidhuber // Proceedings of ICML. – 2006. – P. 369–376.
5. Chan W. et al. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition / W. Chan, N. Jaitly, Q. Le, O. Vinyals // Proceedings of ICASSP. – 2016. – P. 4960–4964.
6. Pang B. et al. A Comprehensive Study of Automatic Speech Recognition / B. Pang, Y. Nie // arXiv preprint arXiv:2303.10738. – 2023.
7. OpenAI. Whisper. [Електронний ресурс] – Режим доступу: https://github.com/openai/whisper.
8. FFmpeg Documentation. [Електронний ресурс] – Режим доступу: https://ffmpeg.org/documentation.html.
9. FastAPI Documentation. [Електронний ресурс] – Режим доступу: https://fastapi.tiangolo.com/.
10. W3C WebVTT Standard. Web Video Text Tracks Format. [Електронний ресурс] – Режим доступу: https://www.w3.org/TR/webvtt1/.
11. SubRip Text Format Specification. [Електронний ресурс] – Режим доступу: https://www.matroska.org/technical/subtitles.html.

12. Ney H. et al. Data Driven Search Organization for Continuous Speech Recognition / H. Ney, U. Essen, R. Kneser // IEEE Transactions on Signal Processing. – 1994. – Vol. 42, no. 4. – P. 1029–1039.
13. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library / A. Paszke et al. // NeurIPS. – 2019.
14. DeepL API Documentation. [Електронний ресурс] – Режим доступу: https://developers.deepl.com/docs.
15. Google Cloud Translation API. [Електронний ресурс] – Режим доступу: https://cloud.google.com/translate/docs.
16. LibreTranslate. [Електронний ресурс] – Режим доступу: https://libretranslate.com/.
17. Celery Documentation. [Електронний ресурс] – Режим доступу: https://docs.celeryq.dev/.
18. Redis Documentation. [Електронний ресурс] – Режим доступу: https://redis.io/docs.
19. Docker Documentation. [Електронний ресурс] – Режим доступу: https://docs.docker.com/.
20. Papineni K. et al. BLEU: a Method for Automatic Evaluation of Machine Translation / K. Papineni, S. Roukos, T. Ward, W.-J. Zhu // Proceedings of ACL. – 2002. – P. 311–318.
21. Helsinki-NLP. Opus-MT. [Електронний ресурс] – Режим доступу: https://github.com/Helsinki-NLP/Opus-MT.
22. Netflix Timed Text Style Guide. [Електронний ресурс] – Режим доступу: https://partnerhelp.netflixstudios.com/hc/en-us/articles/215758617.
23. BBC Subtitle Guidelines. [Електронний ресурс] – Режим доступу: https://bbc.github.io/subtitle-guidelines/.
24. Goodfellow I. et al. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – Cambridge: MIT Press, 2016. – 800 p.
25. Hochreiter S. Long Short-Term Memory / S. Hochreiter, J. Schmidhuber // Neural Computation. – 1997. – Vol. 9, no. 8. – P. 1735–1780.

26. Graves A. et al. Hybrid Speech Recognition with Deep Bidirectional LSTM / A. Graves, N. Jaitly, A.-R. Mohamed // Proc. ASRU. – 2013.
27. Pydantic Documentation. [Електронний ресурс] – Режим доступу: https://docs.pydantic.dev/.
28. pytest Documentation. [Електронний ресурс] – Режим доступу: https://docs.pytest.org/.