Mohamed Khider University of Biskra
Faculty of Exact Sciences
Department of Computer Science

MASTER THESIS

Speciality : Master - Computer Science Applied to Business Management (Professional) - Ref : P1499/2026

TITLE

AI-Based Video Highlights Generator for Recorded Sessions

Prepared and defended by

Hadil - asma GOUNI

Nour el houda SEGHIROU

Lina farah LAMRI SAHRAOUI

Defense date : 07/06/2026

DEFENSE COMMITTEE

Full Name	Grade	Role
Amina BOUGUETITICHE	MCB	**President**
Belkacem ABDELLI	MCB	**Supervisor**
Ammar HAMIDA	MAA	**Examiner**
Nadjoua HABBA	Prof	**Incubator Representative**

Academic Year : 2025/2026

ACKNOWLEDGEMENTS

First and foremost, we express our deepest gratitude to Allah, the Most Merciful and the Most Gracious, for His endless blessings, guidance, and strength throughout the completion of this work.

We would like to extend our sincere appreciation to our supervisor, **Dr.Abdelli Belkacem**, for his invaluable guidance, continuous support, patience, and encouragement throughout this project. His expertise, advice, and constructive feedback greatly contributed to the success of this thesis and helped us overcome many challenges during this journey.

We are also deeply thankful to the **Department of Computer Science** and to all the teachers who accompanied us throughout our academic journey. Their dedication, teachings, and support played a fundamental role in developing our knowledge and skills.

Our heartfelt gratitude goes to our parents for their unconditional love, sacrifices, encouragement, and constant belief in our abilities. Their support has always been our greatest source of motivation and strength.

We would also like to thank our brothers and sisters for their moral support, encouragement, and continuous motivation throughout this journey.

Finally, we sincerely appreciate everyone who contributed directly or indirectly to the completion of this thesis. This work would not have been possible without the collective support and assistance of many people.

May Allah bless you all abundantly for your kindness, support, and belief in us.

Abstract

With the ever-growing flood of video content and recorded lectures, finding the information you actually need has become a real challenge one that often means sitting through entire videos just to catch a few important moments.

This project tackles that problem by building an AI-powered web application that automatically analyzes videos, extracts key segments with their exact timestamps, and delivers clean, structured summaries. The pipeline leverages OpenAI Whisper for multilingual speech recognition and the Groq API for chapter generation, summarization, and key point extraction, with an intelligent audio classification mechanism that adapts processing based on detected content type: speech, music, or silence.

Users can register via email or Google OAuth, upload video files or submit YouTube URLs, and organize analyses into topic-based collections. The interface supports English, French, and Arabic with full RTL support, and summaries can be downloaded for offline reference. A companion Android mobile application developed with React Native and Expo further extends the platform's accessibility to mobile users.

Keywords: Artificial Intelligence, Video Analysis, Automatic Speech Recognition, Groq API, Whisper, Multimodal Analysis, Web Application, Mobile Application, Full-Stack.

Résumé

Face à l'explosion du contenu vidéo et à la multiplication des cours enregistrés, trouver l'information dont on a vraiment besoin est devenu un véritable défi qui oblige souvent à regarder des vidéos entières juste pour en retenir quelques instants clés.

Ce projet développe une application web alimentée par l'intelligence artificielle, capable d'analyser automatiquement les vidéos, d'en extraire les segments essentiels avec leurs horodatages précis, et de générer des résumés clairs et structurés. Le pipeline utilise OpenAI Whisper pour la reconnaissance vocale multilingue et l'API Groq pour la génération de chapitres et de résumés, avec une classification audio intelligente adaptant le traitement selon le contenu détecté : parole, musique ou silence.

Les utilisateurs peuvent s'inscrire via email ou Google OAuth, téléverser des vidéos ou soumettre des liens YouTube, et organiser leurs analyses en collections thématiques. L'interface est disponible en anglais, français et arabe avec support RTL complet, et les résumés peuvent être téléchargés pour consultation hors ligne. Une application mobile Android développée avec React Native et Expo complète la plateforme en étendant son accessibilité aux utilisateurs mobiles.

Mots-clés : Intelligence artificielle, Analyse vidéo, Reconnaissance vocale automatique, API Groq, Whisper, Analyse multimodale, Application web, Application mobile, Full-Stack.

Contents

Acknowledgements

Abstract i

Résumé ii

General Introduction 1

1 State of the Art 4
- 1.1 Introduction . . . 4
- 1.2 Artificial Intelligence Fundamentals . . . 4
  - 1.2.1 Definition of Artificial Intelligence . . . 4
  - 1.2.2 Machine Learning . . . 5
  - 1.2.3 Deep Learning . . . 5
  - 1.2.4 Transformer Architecture . . . 6
  - 1.2.5 Pre-trained Language Models . . . 6
- 1.3 AI-Based Technologies for Video Processing . . . 8
  - 1.3.1 Multimedia Processing . . . 8
  - 1.3.2 Automatic Speech Recognition . . . 9
  - 1.3.3 YouTube Video Acquisition . . . 10
  - 1.3.4 Web API Architecture . . . 10
- 1.4 Video Content Analysis . . . 11
  - 1.4.1 Video Highlight Detection . . . 11
  - 1.4.2 Video Summarization . . . 12
- 1.5 Existing Solutions and Systems . . . 13
  - 1.5.1 Academic Research . . . 13
  - 1.5.2 Commercial Solutions . . . 14
  - 1.5.3 Open Source Tools . . . 16
- 1.6 Comparative Analysis . . . 16
  - 1.6.1 Evaluation Criteria . . . 16
  - 1.6.2 Comparative Table . . . 17
  - 1.6.3 Critical Analysis . . . 17
- 1.7 Positioning of Our Solution . . . 17
- 1.8 Conclusion . . . 18

2 Requirements Analysis and Specifications 19
- 2.1 Introduction . . . 19
- 2.2 Requirements Analysis . . . 19
  - 2.2.1 Identification of Actors . . . 19

2.2.2 Functional Requirements . . . 20
2.2.3 Non-Functional Requirements . . . 22
2.3 System Specifications . . . 23
2.3.1 Project Scope . . . 23
2.3.2 System Boundaries . . . 23
2.3.3 System Constraints . . . 23
2.4 Technology Stack . . . 24
2.4.1 Frontend . . . 24
2.4.2 Backend . . . 25
2.4.3 AI Microservice . . . 25
2.4.4 Authentication . . . 25
2.4.5 Database . . . 26
2.5 Development Methodology . . . 26
2.5.1 Agile Approach . . . 26
2.5.2 Project Iterations . . . 26
2.6 Technical Risks and Mitigation Strategies . . . 27
2.6.1 Technical Risks . . . 27
2.6.2 Mitigation Strategies . . . 27
2.7 Conclusion . . . 28

3 Conception **29**
3.1 Introduction . . . 29
3.2 System Architecture . . . 29
3.2.1 Architectural Pattern . . . 29
3.2.2 Global Architecture . . . 30
3.2.3 Detailed Architecture . . . 31
3.2.4 Communication Flow . . . 32
3.3 UML Diagrams . . . 33
3.3.1 Use Case Diagram . . . 33
3.3.2 Sequence Diagram . . . 35
3.3.3 Class Diagram . . . 36
3.3.4 Activity Diagrams . . . 39
3.4 Database Design . . . 42
3.4.1 Database Schema . . . 42
3.4.2 Collections Description . . . 43
3.5 Conclusion . . . 46

4 Implementation and Development **47**
4.1 Introduction . . . 47
4.2 Development Environment . . . 47
4.2.1 Development Tools . . . 47
4.2.2 Frontend Development . . . 48
4.2.3 Backend Development . . . 48
4.2.4 AI Microservice Development . . . 49
4.2.5 AI and Media Processing Tools . . . 49
4.2.6 Database . . . 50
4.2.7 API Testing . . . 50
4.2.8 Version Control and Collaboration . . . 50
4.2.9 Mobile Application . . . 51

4.3 System Components 51
4.3.1 Frontend Component 51
4.3.2 Backend Component 52
4.3.3 AI Microservice Component 53
4.3.4 Database Component 53
4.4 Module Implementation 53
4.3.1 Web Application Interfaces 53
4.3.2 Mobile Application Interfaces 68
4.5 Conclusion 77

General Conclusion **78**
Perspectives 79

Bibliography **80**

List of Tables

1.1 Common Video Formats Supported by FFmpeg 9
1.2 Whisper Model Variants 10
1.3 Comparison of Existing Video Analysis Solutions 17

2.1 Technology Stack 24

List of Figures

1.1 AI over time 6

3.1 System Architectural Patterns 30
3.2 Global System Architecture 30
3.3 Use Case Diagram 34
3.4 Sequence Diagram — Video Processing Flow 35
3.5 Class Diagram 38
3.6 Activity Diagram – Authentication and Verification Workflow 39
3.7 Activity Diagram – Video Upload and AI Processing Workflow 40
3.8 Activity Diagram – User Management and Settings Workflow 41
3.9 Activity Diagram – AI Processing Pipeline 42
3.10 Database Schema 43

4.1 VS Code Logo 47
4.2 React.js Logo 48
4.3 Tailwind CSS Logo 48
4.4 Node.js Logo 48
4.5 Express.js Logo 48
4.6 Python Logo 49
4.7 FastAPI Logo 49
4.8 FFmpeg Logo 49
4.9 Whisper Logo 49
4.10 Groq API Logo 50
4.11 MongoDB Logo 50
4.12 Postman Logo 50
4.13 GitHub Logo 50
4.14 Expo Logo 51
4.15 Render Logo 51
4.16 Railway Logo 51
4.17 Landing Page 54
4.18 Authentication Modal 54
4.19 Login Interface 55
4.20 Registration Interface 56
4.21 Email Verification Interface 57
4.22 Forgot Password Interface 57
4.23 Password Reset Interface 58
4.24 Google OAuth Authentication Flow 59
4.25 Video Upload Interface with Progress Tracking 59
4.26 User Dashboard 60

4.27 Analysis Results Page 60
4.28 Chapters Sidebar 61
4.29 Transcript Section 62
4.30 Summary Section 63
4.31 Chapter Information Panel 63
4.32 PDF Export Feature 64
4.33 Collections Management Interface 64
4.34 General Settings Interface 65
4.35 Security Settings Interface 65
4.36 Notification Settings Interface 66
4.37 History Sidebar 67
4.38 Profile Interface 68
4.39 Mobile Application – landing page 69
4.40 Mobile Application – Login and Registration Screens 70
4.41 Mobile Application – Home and Video Submission 71
4.42 Mobile Application –Dashboard 72
4.43 Mobile Application – Analysis Results and Chapters 73
4.44 Mobile Application – Visual-Only Analysis: Frame Extraction and Groq Vision Processing 74
4.45 Mobile Application – Collections 75
4.46 Mobile Application – Settings and edit profil 76

General Introduction

General Context

Digital transformation has significantly reshaped the way information is created, shared, and consumed in recent years. Among the various forms of digital media, video content has emerged as one of the most dominant forms of digital information, extensively utilized in education, professional training, and online learning platforms. Universities record lectures, companies archive meetings, and educational institutions provide entire courses in video format, making video a central medium for knowledge dissemination.

The rapid growth of online platforms and remote learning systems has led to an exponential increase in recorded video content. While this expansion has improved accessibility to information, it has also introduced a major challenge: efficiently navigating and extracting relevant information from long video recordings. Unlike textual content, which can be easily scanned and searched, video requires sequential viewing, forcing users to spend considerable time locating specific segments.

This challenge becomes particularly critical in educational and professional environments where time efficiency and quick access to information are essential. Students often need to revisit specific explanations within recorded lectures, while professionals may search for key moments in long meeting recordings. However, most existing platforms provide only basic playback functionalities without intelligent tools capable of automatically analyzing and extracting meaningful content.

In this context, Artificial Intelligence (AI) has emerged as a powerful enabler for automating video content analysis. Advancements in Natural Language Processing (NLP) and Automatic Speech Recognition (ASR) technologies have made it possible to convert spoken language into text, identify important moments, and generate concise summaries from long recordings. These recent technological advancements have opened new opportunities for intelligent multimedia systems capable of transforming raw video data into structured and searchable knowledge.

Such intelligent systems have the potential to significantly improve how users interact with video content by reducing manual navigation efforts and providing direct access to relevant information.

Problem Statement

Despite the wide availability of video-based content, most existing platforms offer only basic playback functionalities and lack intelligent tools for highlight detection or automatic summarization. Users encounter various difficulties, such as:

- Finding crucial moments in lengthy videos is time-consuming and challenging.

- Manual navigation and keyword searching across video content is highly inefficient.
- Few existing platforms provide reliable automated tools for generating structured summaries from recorded sessions.
- Video materials are often poorly organized, making archived content difficult to retrieve later.
- Accessing essential information from long educational or professional recordings requires considerable effort and time.

These limitations reduce learning efficiency and professional productivity, especially in environments where time management is critical.

Objectives

To address these challenges, we propose the development of an **AI-Based Video Highlights Generator** a full-stack web application designed to automatically analyze recorded videos and generate structured highlights and summaries.

The main objectives of this project are:

- To design and implement a full-stack web application for video upload and management, supporting both local file uploads and YouTube URL processing.
- To integrate AI-based speech recognition using OpenAI Whisper for multilingual transcription capabilities.
- To generate automatic chapters, summaries, and key points using the Groq API and Llama large language models.
- To implement multimodal analysis combining audio transcription and visual frame analysis through Groq Vision models.
- To provide a secure authentication system supporting both email/password authentication and Google OAuth integration.
- To build a multilingual interface supporting English, French, and Arabic with full RTL support.
- To develop a collection management system enabling users to organize and share their video analyses efficiently.
- To provide customizable user settings such as themes, notifications, timezone management, and video processing preferences.

The originality of this project lies in combining multimodal AI analysis, multilingual processing, and modern web technologies within a unified and user-friendly platform.

Organization of the Thesis

This thesis is organized as follows:

- **Chapter 1** presents the state of the art, including the theoretical background related to Artificial Intelligence, speech recognition, Natural Language Processing, large language models, and an analysis of existing solutions.
- **Chapter 2** focuses on system analysis, where user requirements are identified, functional and non-functional specifications are defined, and the selected technologies are introduced.
- **Chapter 3** describes the system design phase, including the overall architecture, UML diagrams, database modeling, and module organization.
- **Chapter 4** presents the implementation and development phase, detailing the development environment, system modules, AI integration process, and obtained results illustrated through interface screenshots.
- Finally, the **General Conclusion** summarizes the work accomplished, discusses the main limitations encountered during development, and presents future perspectives and possible improvements for the proposed system.

Chapter 1

State of the Art

1.1 Introduction

The creation and consumption of digital material have grown at an unprecedented rate in recent years, with video becoming the most popular medium for imparting knowledge, information, and academic and professional training. The extensive distribution of video content across professional, educational, and entertainment platforms has significantly altered how people consume information. Nevertheless, despite these benefits, this rapid growth has created a major challenge: processing enormous volumes of content and retrieving crucial information within a short timeframe.

The emergence of artificial intelligence (AI) technologies has qualitatively transformed multimedia processing, offering a strategic solution to this problem. To convert unprocessed data into useful knowledge, fields such as video analysis, automatic speech recognition (ASR), natural language processing (NLP), and text summarization have become critical. Among the most prominent applications in this domain are automatic key-moment extraction systems, which aim to reduce viewing time and improve access to high-value information.

The goal of this chapter is to present a thorough examination of the theoretical underpinnings and current research status concerning intelligent video analysis and highlight production. We first establish the basic principles and review significant related works and current solutions. Following this, we conduct a critical comparative study to identify their respective advantages and disadvantages. By employing this methodology, we aim to provide a robust scientific basis that supports the technological decisions made in our proposed system and demonstrates its effectiveness compared to existing options.

1.2 Artificial Intelligence Fundamentals

1.2.1 Definition of Artificial Intelligence

Artificial Intelligence is defined as the study and design of intelligent agents that perceive their environment and take actions that maximize their chances of achieving goals [1]. The field encompasses both the theoretical foundations and practical applications of systems capable of performing tasks traditionally requiring human intelligence.

Russell and Norvig classify AI definitions along two main dimensions [1]:

- Systems that think like humans versus systems that act like humans

- Systems that think rationally versus systems that act rationally

Among these perspectives, the rational agent approach is considered the most general and widely accepted. A rational agent selects actions that maximize expected performance based on the available percepts and built-in knowledge. Our proposed system embodies this paradigm by automatically perceiving video content through speech recognition and visual frame analysis, then acting upon it to generate structured highlights, summaries, and chapters maximizing the user's ability to access relevant information efficiently.

1.2.2 Machine Learning

Machine learning is not just a technical buzzword it is the idea that fundamentally changed the way we think about programming. Rather than creating explicit rules for each problem, we give the computer data and let it solve problems on its own. Mitchell [2] captured this idea in an elegant formal definition: a computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance on T as measured by P improves with experience E. In plain terms: the more data the program sees, the smarter it gets.

As Goodfellow et al. [3] describe, learning approaches are generally divided into three paradigms. **Supervised learning** trains a model on labeled input-output pairs, enabling it to generalize and make accurate predictions on unseen data. **Unsupervised learning** lets the model discover hidden patterns and structures in data without any predefined labels or guidance. Finally, **reinforcement learning** trains an agent through trial and error, rewarding correct decisions and penalizing poor ones much like how humans learn from experience and feedback.

These three paradigms collectively form the theoretical backbone of modern AI systems, and understanding them is essential before examining the more specialized models and architectures discussed in the sections that follow. The AI models integrated in our pipeline are direct products of these learning paradigms: OpenAI Whisper was trained using **supervised learning** on 680,000 hours of labeled audio-text pairs [20], while the Llama models accessed via the Groq API were developed using a combination of supervised fine-tuning and **reinforcement learning from human feedback** (RLHF) [12], enabling them to generate high-quality summaries and structured chapter titles from video transcripts.

1.2.3 Deep Learning

Deep learning is a subfield of machine learning that uses neural networks with multiple layers to learn hierarchical representations of data. These models are capable of automatically extracting complex features from raw data through successive layers of nonlinear transformations. According to [3], deep learning has achieved significant success in many domains such as speech recognition, computer vision, and natural language processing. Both core AI components of our system Whisper and the Llama models are deep learning systems built on the Transformer architecture [6]. Whisper processes raw audio spectrograms through stacked encoder layers to produce timestamped transcripts, while the Llama large language models leverage dozens of Transformer decoder layers to generate coherent, structured summaries and chapter titles from the transcribed video content.

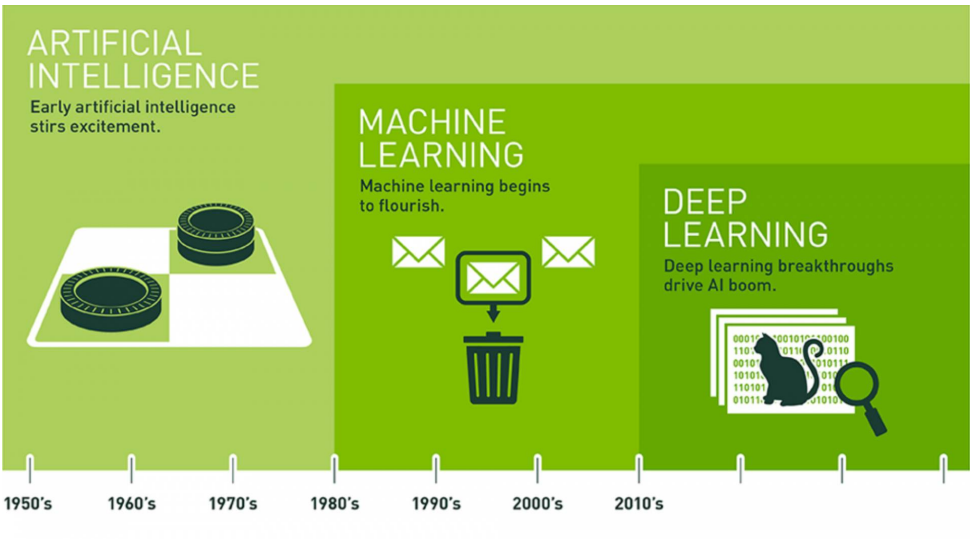

Figure 1.1: AI over time

1.2.4 Transformer Architecture

The Transformer architecture has become one of the most important developments in the field of Natural Language Processing (NLP). It relies on the self-attention mechanism, which enables the model to capture relationships between words in a sequence regardless of the distance between them. Unlike traditional sequential models, transformers process tokens in parallel, allowing more efficient training and better modeling of long-range dependencies.

A transformer model is composed of a stack of identical layers. Each layer mainly consists of two components: a Multi-Head Attention mechanism and a Feed-Forward Neural Network. The attention mechanism computes the importance between tokens using three matrices known as Query (Q), Key (K), and Value (V). Multi-head attention allows the model to learn different types of relationships within the text by running multiple attention operations in parallel.

Before entering the transformer layers, input words are converted into numerical vector representations called embeddings. These embeddings consist of two parts: token embeddings, which represent the semantic meaning of words, and positional embeddings, which encode the position of each token in the sequence. The combination of these embeddings forms the input representation of the model.

Finally, the output of the last transformer layer is passed to a language modeling head that transforms the representation into a probability distribution over the entire vocabulary using a softmax function. This enables the model to predict the most likely next token in a sequence.

The transformer architecture serves as the foundation of many modern language models due to its strong ability to capture contextual information and efficiently process large amounts of textual data .[4]

1.2.5 Pre-trained Language Models

Large pre-trained language models (PLMs) have fundamentally transformed the field of Natural Language Processing. Rather than training a model from scratch for each specific task, the pre-training paradigm involves first learning a generic language representation

from large unlabeled text corpora, then fine-tuning the model for a particular downstream task [7].

As Min et al. [7] describe, this paradigm shift was made possible by the advent of the Transformer architecture, whose multi-head self-attention mechanism allows every word to attend to every other word in a sequence, efficiently capturing long-range dependencies without the expensive recurrent computation of earlier models. Nearly all modern PLMs are built on this foundation and have achieved state-of-the-art performance across a wide range of NLP tasks [7].

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model based on the Transformer architecture that excels in a wide range of Natural Language Processing tasks [8]. Unlike earlier unidirectional models, BERT employs bidirectional training and a masked language modeling objective, enabling it to capture contextual information from both directions of a sequence simultaneously [8]. This bidirectional approach allows the model to understand the full context of each word based on all surrounding words, rather than relying solely on preceding or following tokens.

BERT is pre-trained on large unlabeled text corpora using two self-supervised tasks: Masked Language Modeling (MLM), where random tokens are masked and the model learns to predict them, and Next Sentence Prediction (NSP), where the model learns to understand relationships between sentence pairs. The resulting pre-trained model can then be fine-tuned for specific downstream tasks — including text classification, summarization, and named entity recognition — with minimal architectural modifications [8].

mT5

mT5 (Multilingual Text-to-Text Transfer Transformer) is a massively multilingual variant of the T5 model, introduced by Xue et al. [14]. While T5 was pre-trained exclusively on English-language text, mT5 extends this framework to 101 languages by pre-training on mC4, a multilingual variant of the C4 dataset constructed from the public Common Crawl web scrape [14].

mT5 inherits T5's unified text-to-text format, in which every NLP task — whether classification, question answering, or abstractive summarization — is cast as a sequence-to-sequence problem: the model receives a text input and generates a text output. This general-purpose format allows a single set of hyperparameters and the same training objective to be used across all downstream tasks, significantly simplifying the fine-tuning process [14].

The model is available in five sizes: Small, Base, Large, XL, and XXL, ranging from approximately 300 million to 13 billion parameters. mT5 uses an encoder-decoder Transformer architecture with a vocabulary of 250,000 wordpieces to accommodate the diverse character sets of 101 languages. Evaluated on the XTREME multilingual benchmark, the largest variant mT5-XXL achieves state-of-the-art performance on classification, question answering, and named entity recognition tasks across multiple languages [14].

Large Language Models and the Groq API

While BERT and mT5 represent foundational contributions to the field of pre-trained language models, recent advances in large language models (LLMs) have further expanded

the capabilities of NLP systems. Modern LLMs such as the Llama family [12] are trained on significantly larger corpora and offer stronger generalization across diverse tasks, including abstractive summarization, question answering, and multimodal understanding.

To leverage these capabilities efficiently, our system integrates the **Groq API** [11], a cloud-based inference service that provides extremely fast LLM inference through dedicated Language Processing Unit (LPU) hardware. Unlike traditional GPU-based inference, Groq's LPU architecture is specifically optimized for sequential token generation, delivering significantly lower latency. The effectiveness of combining transformer-based models with the Groq API for NLP classification tasks has been demonstrated by Choudhary et al. [13], who achieved 82.76% accuracy in text classification, validating Groq as a reliable inference backend for NLP pipelines.

In our system, two Groq-hosted models are employed for distinct tasks:

- **meta-llama/llama-4-scout-17b-16e-instruct:** A vision-capable model used to analyze extracted video frames and generate natural language descriptions of visual content in Arabic, French, and English.
- **llama-3.3-70b-versatile:** A large language model used to generate structured chapter titles, abstractive summaries, and key points from the transcribed video content.

When the Groq API is unavailable, the system automatically falls back to extractive summarization techniques, ensuring robustness and continuity of service [11].

1.3 AI-Based Technologies for Video Processing

This section presents the enabling technologies that support the system pipeline.

1.3.1 Multimedia Processing

Video files, in their raw form, are not directly consumable by AI models. Before any speech recognition or content analysis can take place, the media must first be decoded, processed, and converted into a format that downstream models can work with. This preprocessing stage is as critical as the AI itself without it, the pipeline cannot begin [22].

FFmpeg

Among all available media processing frameworks, FFmpeg stands out as the de facto standard in both research and industry. FFmpeg [35] is a free and open-source multimedia framework capable of decoding, encoding, transcoding, and filtering virtually any media format ever created. Originally developed by Fabrice Bellard in 2000 and maintained by a worldwide community of contributors, it has become an indispensable component in multimedia pipelines across academia and industry alike.

The most critical operation FFmpeg performs in the context of video analysis systems is **audio extraction** the process of separating the audio track from a video container file. Given a video input in any common format (MP4, AVI, MKV, MOV), FFmpeg isolates the audio stream and exports it as a standalone file compatible with downstream speech recognition models. Beyond simple extraction, FFmpeg handles audio normalization adjusting sampling rates, bit depths, and channel configurations to meet the exact input

specifications required by models such as Whisper. Table 1.1 summarizes the main input formats supported by FFmpeg that are relevant to video analysis systems.

Table 1.1: Common Video Formats Supported by FFmpeg

Format	Type	Common Use
MP4	Video container	Online streaming, uploads
AVI	Video container	Legacy recordings
MKV	Video container	High-definition content
MOV	Video container	Apple devices
WAV	Audio output	ASR model input

1.3.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the technology that enables machines to convert spoken language into written text. Modern general-purpose speech recognition systems are generally based on Mel Frequency Cepstral Coefficients (MFCCs) for audio signal representation and Hidden Markov Models (HMMs) to model the speech stochastic process [15].

Traditional ASR systems rely on three core components working together. The **acoustic model** contains a statistical representation of the distinct sound units that make up words, where each distinct sound unit corresponds to a phoneme [15]. The **language model** contains a list of words and their probability of occurrence in a given sequence, providing word-level language structure through either graph-driven models or N-Gram models [15]. Finally, the **decoder** searches for the most likely phoneme sequence by treating speech recognition as a graph search problem [15].

Whisper

Among the most significant recent advances in ASR is Whisper, an open-source model developed by Radford et al. [20] at OpenAI. Trained on 680,000 hours of multilingual, multi-task supervised audio data collected from the web, Whisper achieves remarkable robustness to accents, background noise, technical terminology, and diverse recording conditions.

The model is available in five sizes *tiny*, *base*, *small*, *medium*, and *large* each offering a different trade-off between processing speed and transcription accuracy. It performs automatic language detection across 99 languages without additional configuration, and produces structured, timestamped output that provides the precise start and end time of each spoken segment a feature that is fundamental to any system requiring temporal alignment between transcribed text and video content [21]. Table 1.2 summarizes the five available Whisper model variants and their respective trade-offs between size, speed, and accuracy.

In our system, the `base` model was selected as the default configuration, offering a practical balance between transcription accuracy and processing speed on standard hardware without GPU acceleration. The model size can be configured via an environment

Table 1.2: Whisper Model Variants

Model	Parameters	Speed	Accuracy
Tiny	39M	Very Fast	Low
Base	74M	Fast	Moderate
Small	244M	Moderate	Good
Medium	769M	Slow	High
Large	1550M	Very Slow	Highest

variable, allowing users to switch to a more accurate variant when computational resources permit.

1.3.3 YouTube Video Acquisition

A significant proportion of educational and professional video content is hosted on YouTube, the world's largest video sharing platform. To support direct processing of online content without requiring users to manually download videos, our system integrates **yt-dlp** [36], an open-source command-line tool for downloading videos from YouTube and hundreds of other platforms.

yt-dlp is a feature-rich fork of the widely used youtube-dl project, offering improved download speeds, broader format support, and active maintenance. In our pipeline, yt-dlp is invoked by the AI microservice to retrieve the video stream corresponding to a submitted YouTube URL, storing it temporarily in the server's file system before passing it to the FFmpeg audio extraction stage. The tool also provides access to video metadata such as the title, duration, and description, which are used to populate the video record in the database.

This integration eliminates a major friction point for users who work with online content, enabling them to submit a URL directly through the web interface without any additional steps.

1.3.4 Web API Architecture

Modern web applications are no longer built as single monolithic systems they are composed of multiple specialized components that must communicate in a structured and reliable way. As Biehl [16] describes, API architecture encompasses the complete picture of how APIs connect clients, services, and infrastructure within a software system.

API

An Application Programming Interface (API) is an interface that allows software applications to communicate with each other. It defines a set of rules and endpoints through which external programs can access specific functionalities of a system without exposing its internal implementation. In the context of web services, APIs enable applications to interact over the internet in a structured and predictable manner [17].

RESTful API

A RESTful API is an application programming interface that follows the principles of Representational State Transfer (REST), a widely used architectural style for designing web services. RESTful APIs rely on standard HTTP methods such as GET, POST, PUT, and DELETE to perform operations on resources. Each resource is identified by a unique URL and data is typically exchanged using lightweight formats such as JSON.

Data Exchange Format

JavaScript Object Notation (JSON) is the standard data interchange format used in RESTful APIs. Its lightweight, human-readable structure makes it ideal for transmitting structured data between client and server. JSON supports nested objects and arrays, enabling the representation of complex data structures such as video metadata, transcript segments, and chapter objects within a single response payload [16].

According to Lubanovic [18], REST-based services allow different applications to communicate over the web in a simple, scalable, and platform-independent manner. This approach makes it easier to build modular systems where the client and server can evolve independently.

1.4 Video Content Analysis

Video content analysis refers to the automated process of extracting meaningful information from video recordings. Rather than treating video as a passive media file, intelligent analysis systems examine the content spoken words, structure, and temporal flow to identify what matters most and present it in an organized, accessible form. This section examines two core tasks that are central to our system: highlight detection and video summarization.

1.4.1 Video Highlight Detection

Video highlight detection is the task of automatically identifying the most informative and relevant segments within a video recording. Rather than forcing users to watch an entire video, the goal is to surface only what truly matters with precise timestamps [24].

Visual-Based Approach

Early approaches to highlight detection relied on visual analysis of video frames. Karpathy et al. [24] explored CNN architectures for large-scale video classification, examining how spatial features extracted from individual frames could be combined across time to understand video-level content. In parallel, Hochreiter and Schmidhuber [23] introduced Long Short-Term Memory (LSTM) networks, which addressed the vanishing gradient problem in recurrent neural networks and enabled models to capture long-range temporal dependencies a critical requirement when reasoning about events that unfold over minutes or hours of footage.

Despite their effectiveness, visual-based approaches present significant limitations. They require substantial computational resources, depend heavily on visual distinctiveness, and fail to capture the semantic meaning of spoken content which is often the most valuable source of information in educational and professional recordings.

Audio-Based Approach

A more practical and semantically richer alternative is audio-based highlight detection, which analyzes the spoken content of a video rather than its visual frames. By combining automatic speech recognition with natural language processing, it becomes possible to identify the most semantically important segments of a recording and return them to the user with their exact timestamps regardless of the visual content of the video.

Radford et al. [20] demonstrated that large-scale speech recognition models such as Whisper can produce highly accurate, timestamped transcripts across 99 languages and diverse recording conditions. Combined with NLP models for semantic scoring, this approach enables highlight detection that is both computationally efficient and domain-independent making it particularly well suited for processing recorded lectures, meetings, and educational content.

1.4.2 Video Summarization

Video summarization aims to create a concise representation of a long recording that preserves its essential content. Rather than selecting random frames or segments, an effective summarization system must understand the structure and importance of different parts of the video and present them in a coherent and accessible form.

Visual-Based Summarization

Zhang et al. [26] proposed one of the most influential approaches to this problem, using Long Short-Term Memory networks to model the sequential structure of video content and automatically select the most representative keyframes. Their work demonstrated that temporal modeling understanding not just what happens in a video but when and in what order is essential for generating meaningful summaries.

Gygli et al. [25] approached the problem differently, creating summaries from user-generated videos by selecting diverse and visually representative shots. Their work highlighted the importance of combining relevance and variety when constructing video summaries.

Text-Based Summarization

While visual approaches analyze frames and motion, text-based summarization focuses on the spoken content of the video. Once a transcript is generated via automatic speech recognition, NLP models can identify the most important sentences, group them by topic, and produce structured summaries that capture the key ideas of the recording [9].

Raffel et al. [10] introduced T5, a unified text-to-text framework that treats summarization as a mapping from input text to output text. Extended to multilingual settings by Xue et al. [14] as mT5, this approach enables summarization across multiple languages including Arabic, French, and English making it particularly suited for diverse educational and professional content.

1.5 Existing Solutions and Systems

1.5.1 Academic Research

Academic research in the field of automated video analysis has explored a wide range of approaches, from visual-based summarization to speech-driven content extraction. This section reviews the most relevant contributions that have directly informed the design of our system.

Speech-Based Video Summarization

A closely related system was proposed by researchers at IJRASET [29], who developed an automated video summarization pipeline specifically targeting online educational content. Their approach follows a two-stage process: first, OpenAI Whisper is applied to perform speech-to-text conversion on the video's audio track, producing a full textual transcript. Second, the transcript is processed using TF-IDF scoring and Gensim-based extractive summarization algorithms to identify and extract the most informative sentences. The resulting output provides users with a concise, topic-focused summary of the original video content without requiring manual intervention.

This work is directly relevant to our system, as it validates the core pipeline we adopt: audio extraction, speech recognition via Whisper, and NLP-based summarization. The primary distinction is that our system extends this approach by adding timestamp-based highlight detection, multilingual support via mT5, and a full user-facing web interface with collection management and history tracking.

Deep Reinforcement Learning for Video Summarization

A relevant study by Zhou et al. [30] proposes a deep reinforcement learning approach for unsupervised video summarization. The method models video summarization as a sequential decision-making process, where a deep summarization network (DSN) selects important video frames based on learned probabilities.

To guide the learning process, the authors introduce a reward function that combines two key criteria: diversity and representativeness. The diversity reward ensures that the selected frames are varied and not redundant, while the representativeness reward guarantees that the summary accurately reflects the overall content of the video.

This approach is particularly effective as it does not require labeled data and learns automatically through interaction, making it fully unsupervised. Experimental results demonstrate that the proposed method achieves performance comparable to, or even better than, several supervised approaches.

Unsupervised Transcript-assisted Video Summarization and Highlight Detection

Barbakos et al. [31] propose an unsupervised multimodal framework for video summarization and highlight detection that jointly leverages visual frames and their corresponding transcripts. Unlike previous approaches that rely either on unimodal visual features or supervised learning with annotated datasets, the authors formulate the problem as a reinforcement learning task operating on large-scale unannotated data. Their model employs a multimodal Transformer architecture to fuse frame-level features extracted using S3D with textual embeddings obtained from RoBERTa, enabling cross-modal interactions through an

Align & Attend mechanism. A policy gradient optimization strategy is then used to learn a selection policy that maximizes a reward function combining diversity, representativeness, and transcript saliency. Experimental results demonstrate that incorporating transcript information significantly improves the quality of generated summaries and the accuracy of highlight detection compared to visual-only baselines.

Video Summarization Using Fully Convolutional Sequence Networks

Rochan et al. [27] propose an advanced approach for video summarization based on Fully Convolutional Sequence Networks (FCSN), where the problem is formulated as a sequence labeling task that selects keyframes representing the most important content of a video. Unlike traditional recurrent models such as LSTM, the authors adopt convolutional architectures that enable parallel processing of video frames, thus improving computational efficiency and scalability. The model leverages temporal convolutions to capture long-range dependencies between frames, and introduces both supervised and unsupervised variants to address the limitations of annotated datasets. Experimental results demonstrate that this approach achieves competitive performance compared to state-of-the-art methods, particularly in large-scale video analysis scenarios.

Despite its effectiveness, this approach remains fundamentally based on visual information extracted from video frames. In contrast, our system adopts a speech-driven approach to video understanding, focusing primarily on the analysis of audio content rather than visual frames. By integrating automatic speech recognition using Whisper and natural language processing techniques for semantic analysis, our method is able to identify meaningful segments, generate summaries, and detect highlights based on the actual spoken content.

Text Summarization with Pretrained Encoder Models

Liu and Lapata [28] present a comprehensive framework for leveraging BERT in the context of automatic text summarization. The authors demonstrate how a single pretrained encoder can be effectively adapted to support both extractive and abstractive summarization within a unified architecture.

For extractive summarization, the proposed model encodes the full document using BERT and applies additional Transformer layers on top to model inter-sentence relationships and select the most informative sentences. For abstractive summarization, the authors introduce a novel fine-tuning strategy that employs separate optimizers for the pretrained encoder and the randomly initialized decoder. Experiments conducted on three benchmark datasets demonstrate that the proposed approach achieves state-of-the-art performance in both summarization paradigms.

1.5.2 Commercial Solutions

Fireflies.ai

Fireflies.ai [32] is an AI-powered meeting assistant that provides automatic transcription and summarization for virtual meetings. The platform integrates with popular video conferencing tools and offers features such as speaker identification, action item extraction, and searchable transcripts.

Strengths:

- High accuracy transcription
- Real-time processing capabilities
- Integration with Zoom, Teams, and Google Meet
- Automated action item detection

Limitations:

- Limited to live meeting recordings
- Does not support pre-recorded educational content
- Requires paid subscription for advanced features ($10-19/month)
- Cloud-based processing raises privacy concerns

Otter.ai

Otter.ai [33] provides real-time transcription and meeting summarization with enterprise-grade accuracy. The platform employs advanced NLP techniques to generate summaries and identify key discussion points.

Strengths:

- Exceptional transcription quality
- Collaborative note-taking features
- Mobile application support
- Speaker identification and labeling

Limitations:

- Focused primarily on professional meetings
- Limited support for diverse video formats
- Expensive pricing model for regular use
- No support for YouTube URLs or local file uploads

Descript

Descript [34] is a video editing platform that combines transcription with text-based editing capabilities.

Strengths:

- Innovative text-based video editing approach
- High-quality automatic transcription
- Advanced audio enhancement features
- Multi-track editing capabilities

Limitations:

- Primarily a video editor, not an analysis tool
- No automatic highlight detection
- Requires manual intervention for content summarization
- High cost ($24-40/month for professional features)

1.5.3 Open Source Tools

FFmpeg

FFmpeg [35] is a comprehensive multimedia framework widely used for video processing, format conversion, and audio extraction. While not an AI tool, it provides essential infrastructure for video manipulation.

Whisper

Whisper [20] is OpenAI's open-source multilingual speech recognition model, demonstrating robust performance across 99 languages. The model handles diverse audio conditions including accents, background noise, and technical terminology.

1.6 Comparative Analysis

1.6.1 Evaluation Criteria

In order to objectively assess the strengths and limitations of existing solutions, we define **nine** evaluation criteria that reflect the core requirements of an effective video analysis system:

- **Highlight Detection:** The ability to automatically identify and extract the most informative moments within a video, returning them with precise timestamps.
- **Summarization:** The capacity to generate structured, readable summaries from the video content.
- **Pre-recorded Video Support:** The ability to process uploaded video files rather than being restricted to live meeting recordings.
- **YouTube Input:** Direct processing of YouTube URLs without requiring the user to manually download the video beforehand.
- **Multilingual Support:** The capacity to process and analyze video content in multiple languages without requiring manual configuration.
- **Timestamp-Based Navigation:** The ability to associate extracted highlights and summary segments with their precise temporal positions within the original video.
- **Content Organization:** The availability of tools for managing and organizing analyzed content, such as collections, history tracking, and search capabilities.

- **Cost:** The pricing model and overall accessibility of the solution.
- **Privacy:** The degree to which user data remains protected.

1.6.2 Comparative Table

Table 1.3 presents a comparative overview of the most relevant existing solutions for video transcription and analysis, evaluated against the nine criteria defined above.

Table 1.3: Comparison of Existing Video Analysis Solutions

Solution	Highlights	Summarization	Pre-recorded	YouTube	Multilingual	Timestamps	Organization	Cost	Privacy
Fireflies.ai	✓	✓	✗	✗	✗	✓	✗	$10–19/mo	Low
Otter.ai	✗	✓	✓	✗	✗	✗	✗	$17/mo	Low
Descript	✗	✗	✓	✗	✗	✗	✗	$24–40/mo	Low
Our System	✓	✓	✓	✓	✓	✓	✓	**Free**	**High**

✓ Supported ✗ Not Supported

1.6.3 Critical Analysis

Strengths of Existing Solutions

- High transcription accuracy in commercial platforms
- Polished user interfaces and user experience
- Robust cloud infrastructure for scalability
- Professional customer support

Identified Limitations

- **High Costs:** Commercial platforms require expensive subscriptions
- **Domain Specificity:** Most systems target professional meetings only
- **Privacy Concerns:** Cloud-based processing compromises data privacy
- **Limited Customization:** Proprietary systems lack flexibility
- **No Highlight Detection:** Most solutions focus solely on transcription
- **Restricted Input:** Limited support for diverse video sources

1.7 Positioning of Our Solution

Our proposed system addresses the identified limitations through:

- **Multi-Domain Applicability:** Supports education, business, sports, and personal content

- **Comprehensive Analysis:** Combines transcription with intelligent highlight detection
- **Open Architecture:** Built using open-source technologies (MERN stack, Whisper, FFmpeg)
- **Multilingual Support:** Native support for Arabic, French, and English with full RTL rendering, addressing a gap that all reviewed commercial solutions fail to fill for Algerian and Arab users.
- **Privacy-First Design:** Enables local processing and user data control
- **Free Access:** No subscription fees for core functionality
- **Flexible Input:** Accepts file uploads and YouTube URLs
- **User-Centric Interface:** Intuitive design requiring no technical expertise

1.8 Conclusion

The basic ideas of artificial intelligence, video analysis, speech recognition, and text summarization were covered in this chapter. These ideas serve as the theoretical cornerstone of the suggested system. The analysis of current solutions also showed that while many commercial platforms offer strong services, they are frequently expensive and don't offer enough flexibility.

Our suggested system offers an integrated framework built on tried-and-true open-source technologies in a versatile, multi-domain setting to address these issues. The system attempts to provide a useful and effective way to generate video highlights while protecting user privacy and enabling more customization by fusing contemporary Automatic Speech Recognition (ASR), Natural Language Processing (NLP), and video processing techniques.

Chapter 2

Requirements Analysis and Specifications

2.1 Introduction

Building an effective AI-powered system requires a thorough understanding of what the system must do, who will use it, and under what constraints it must operate. Before any design or implementation can take place, it is essential to clearly define the functional and non-functional requirements that will guide every subsequent decision.

This chapter presents a comprehensive analysis of the requirements for our AI-Based Video Highlights Generator. We begin by identifying the actors involved in the system, then detail the functional requirements that describe what the system must accomplish, followed by the non-functional requirements that define how well it must perform. We then present the system specifications and conclude with a presentation of the technology stack selected to meet these requirements.

2.2 Requirements Analysis

2.2.1 Identification of Actors

The system is designed around a single primary actor type, reflecting its focus as a self-service platform:

- **Authenticated User:** Any registered individual who interacts with the system to process video content. This actor can perform the following actions:
  - Register a new account and authenticate securely
  - Upload video files or submit YouTube URLs
  - Request AI-powered analysis and view generated results
  - Organize videos into collections
  - Access processing history and statistics
  - Download summaries for offline use
- **Unauthenticated Visitor (Secondary):** Individuals who access the landing page but cannot use core features until they register.

No administrative role has been defined in this version, as the system operates on a self-service model where each user independently manages their own content.

2.2.2 Functional Requirements

Functional requirements describe the specific behaviors and features that the system must provide. The following functional requirements have been identified for our system:

User Authentication

- The system must allow new users to create an account by providing their name, email address, and password.
- The system must allow registered users to log in using their credentials and log out securely.
- The system must protect all user data through token-based authentication using JSON Web Tokens (JWT).

Video Upload and Input

- The system must allow users to upload video files from their local device in common formats (MP4, AVI, MKV, MOV).
- The system must allow users to submit a YouTube URL as an alternative input, enabling direct processing of online video content without manual downloading.
- The system must validate uploaded files and URLs before processing begins.

AI-Powered Video Analysis

- The system must automatically extract the audio track from the uploaded video using FFmpeg.
- The system must transcribe the extracted audio into text using OpenAI Whisper, producing a timestamped transcript.
- The system must analyze the transcript and video frames using the Groq API (llama-3.3-70b) to generate structured chapters, summaries, and key points.
- The system must classify the audio content of each video as speech, music, or silence using Whisper-based audio analysis, and adapt the processing pipeline accordingly.
- The system must analyze extracted video frames using Groq Vision (llama-4-scout) when visual content analysis is required.
- The system must support multilingual content, including Arabic, French, and English.

Highlights and Timestamps

- The system must return a list of key video segments identified as highlights, each associated with its exact start and end timestamp.
- The system must present highlights in a structured and readable format on the user interface.

Summary Generation and Download

- The system must generate a structured, readable summary of the video content based on the analyzed transcript.
- The system must allow users to download the generated summary directly to their device for offline reference.

Collections Management

- The system must allow users to organize their analyzed videos into topic-based collections.
- The system must allow users to create, rename, and delete collections according to their needs.

History Tracking

- The system must maintain a history log of all videos previously analyzed by the user.
- The system must allow users to access and review past analyses at any time.

Statistics Dashboard

- The system must provide users with a statistics dashboard displaying key metrics such as the total number of videos, completed analyses, and videos currently being processed.

Email Verification

- The system must send a confirmation email to the user upon successful registration containing a verification link.
- The system must restrict access to core features until the user verifies their email address.

Multilingual Interface

- The system must support three interface languages: English, French, and Arabic.
- The system must apply right-to-left (RTL) text direction when Arabic is selected.
- All interface labels, messages, and notifications must be fully translated across the three supported languages.

Google OAuth Authentication

- The system must allow users to register and sign in using their Google account via OAuth 2.0, in addition to the standard email/password method.
- Google-authenticated users must be clearly identified in the profile interface.

User Settings

- The system must allow users to customize their interface language, visual theme (light/dark/system), and date format.
- The system must allow users to update their password and permanently delete their account.

User Profile

- The system must allow users to update their display name, biography, university, and profile photo.
- Profile changes must be persisted in the database and reflected immediately across the interface.

2.2.3 Non-Functional Requirements

Non-functional requirements define the quality attributes and constraints under which the system must operate.

Performance

- The system must process uploaded videos within a reasonable time frame relative to their duration.
- The AI microservice must handle requests asynchronously to avoid blocking the user interface during long-running analysis tasks.

Security

- All API endpoints must be protected and accessible only to authenticated users via JWT.
- User passwords must be stored in hashed form and never transmitted in plain text.

Privacy

- All video processing must be performed locally on the server without transmitting user content to any third-party service.
- Each user must have access only to their own videos, summaries, and history.

Usability

- The user interface must be intuitive and accessible to non-technical users without requiring any prior training.
- The system must provide clear feedback during video processing, including progress indicators and error messages.

2.3 System Specifications

2.3.1 Project Scope

The scope of this project is to develop a full-stack web application that enables users to automatically analyze recorded video content using artificial intelligence. The system targets students, educators, and professionals who regularly work with long video recordings and need to extract key information efficiently.

The system accepts two types of input: locally uploaded video files and YouTube URLs. It processes each input through an AI pipeline consisting of audio extraction, speech recognition, and NLP-based summarization, then returns structured highlights and summaries to the user through a web interface.

2.3.2 System Boundaries

The system encompasses the following components:

- A **frontend** web interface built with React.js, accessible via any modern web browser.
- A **backend** REST API built with Node.js and Express, responsible for user management, data storage, and communication between components.
- An **AI microservice** built with Python and FastAPI, responsible for all video processing and AI analysis tasks.
- A **MongoDB database** for storing user accounts, video metadata, transcripts, summaries, collections, and history.

2.3.3 System Constraints

- **API Dependency:** The summarization and visual analysis features depend on the availability of the Groq API. In case of unavailability, the system falls back to extractive summarization techniques.
- **File Size:** Uploaded video files are subject to server-side size limitations managed by Multer.
- **Processing Time:** Analysis duration depends on video length and the selected Whisper model size.
- **Internet Connectivity:** YouTube URL processing requires an active internet connection for video acquisition via yt-dlp.

2.4 Technology Stack

The selection of technologies was guided by three primary criteria: open-source availability, suitability for AI integration, and compatibility with modern full-stack web development practices. Table 2.1 summarizes the complete technology stack adopted for this project.

Table 2.1: Technology Stack

Layer	Technology	Role
Frontend	React.js	User interface
Backend	Node.js / Express	REST API and business logic
AI Microservice	Python / FastAPI	Video processing and AI analysis
Database	MongoDB	Data storage
Authentication	JWT	Secure user sessions
File Handling	Multer	Video upload management
Media Processing	FFmpeg	Audio extraction
Speech Recognition	Whisper	Speech-to-text conversion
Summarization	Groq API (llama-3.3-70b)	NLP analysis and summarization
Vision Analysis	Groq Vision (llama-4-scout)	Frame description and visual analysis
Audio Classification	Whisper + FFprobe	Speech/music/silence detection
YouTube Input	yt-dlp	YouTube video acquisition
Multilingual Support	React Context API	EN/FR/AR with RTL
OAuth Authentication	Google OAuth 2.0	Social login
Mobile Application	React Native / Expo	Android mobile client

2.4.1 Frontend

React.js

React.js is a JavaScript library for building interactive user interfaces. Its component-based architecture enables the development of a modular, maintainable frontend where each feature of the platform is encapsulated as an independent component. **React.js was selected for this project due to its declarative rendering model, efficient virtual DOM, and strong ecosystem, making it particularly well suited for building dynamic, real-time interfaces such as the video processing dashboard and results display.**

2.4.2 Backend

Node.js and Express

Node.js provides a non-blocking, event-driven runtime environment for the backend API. Combined with the Express framework, it enables the rapid development of RESTful endpoints for user authentication, video management, and communication with the AI microservice. **Node.js was chosen for its asynchronous architecture, which is essential for handling concurrent video upload requests and polling operations without blocking the main thread.**

2.4.3 AI Microservice

Python and FastAPI

The AI processing pipeline is implemented as a separate microservice using Python and FastAPI. This separation allows computationally intensive AI tasks to run independently without affecting the responsiveness of the main backend. **Python was selected as the primary language for the AI microservice due to its dominant ecosystem of AI and machine learning libraries. FastAPI was chosen for its native asynchronous support and automatic API documentation generation.**

FFmpeg

FFmpeg is integrated into the AI microservice to extract audio tracks from uploaded videos and convert them into formats compatible with the speech recognition pipeline. **FFmpeg was selected as the media processing framework due to its format-agnostic capabilities, active maintenance, and widespread adoption in both research and industry multimedia pipelines.**

Whisper

OpenAI Whisper performs speech-to-text conversion from the extracted audio tracks, generating accurate timestamped transcripts of the spoken content across 99 supported languages. **Whisper was selected for its state-of-the-art multilingual performance, open-source availability, and ability to produce timestamped output — a critical requirement for temporal highlight alignment.**

2.4.4 Authentication

JWT

JSON Web Tokens provide a stateless and secure authentication mechanism between the interface, backend, and AI services. **JWT was selected for its stateless nature, which eliminates the need for server-side session storage and enables seamless authentication across the three independent system components.**

Multer

Multer handles multipart form data and manages the upload of video files from the user's device to the server before AI processing begins. **Multer was selected for its native**

integration with Express and its built-in support for file type validation and size limitation.

2.4.5 Database

MongoDB

MongoDB is a document-oriented NoSQL database that stores data as flexible JSON-like documents. It is well suited for this project because the stored data , including video metadata, transcripts, summaries, and generated highlights varies in structure and does not require a rigid relational schema. **MongoDB was selected over relational databases due to its natural compatibility with the variable-length, nested structures produced by the AI pipeline, such as transcript segment arrays and chapter objects, which cannot be efficiently stored in fixed-schema tables.**

2.5 Development Methodology

2.5.1 Agile Approach

In this project, an Agile development approach was adopted in order to ensure flexibility and continuous improvement throughout the development process. Agile methodology is based on iterative development, where the system is built incrementally through small, manageable cycles.

2.5.2 Project Iterations

The development process was structured into five distinct iterations, each with specific objectives and deliverables:

- **Iteration 1: Foundation and Planning**
  - Conduct comprehensive requirements analysis
  - Define system scope and actors
  - Select appropriate technology stack
  - Set up development environment
- **Iteration 2: Backend Infrastructure**
  - Implement user authentication with JWT
  - Develop RESTful API endpoints
  - Integrate MongoDB database
  - Set up file upload handling with Multer
- **Iteration 3: AI Microservice Development**
  - Develop Python FastAPI service
  - Integrate Whisper for speech-to-text
  - Integrate Groq API for summarization and chapter generation

  - Integrate Groq Vision for frame-based visual analysis
  - Implement audio classification pipeline
  - Set up FFmpeg for audio extraction

- **Iteration 4: Frontend and Integration**
  - Build React.js user interface
  - Implement video upload interface with progress tracking
  - Create dashboard for history and collections
  - Integrate frontend with backend and AI service

- **Iteration 5: Testing and Optimization**
  - Conduct comprehensive system testing
  - Optimize AI model performance
  - Fix bugs and improve user experience
  - Prepare documentation and defense materials

2.6 Technical Risks and Mitigation Strategies

2.6.1 Technical Risks

Several technical risks were identified during the development of this system:

- **High processing time:** Whisper transcription on CPU hardware may require significant time for long videos.
- **Groq API rate limiting:** The Groq API enforces request rate limits that may cause delays during high-volume processing.
- **Large file handling:** Uploading and processing large video files may cause memory or storage issues.
- **Model accuracy:** Speech recognition and summarization models may produce inaccurate or incomplete results.
- **Integration complexity:** Ensuring smooth communication between frontend, backend, and AI microservice can be challenging.
- **Multilingual challenges:** Handling multiple languages may affect the consistency of results.

2.6.2 Mitigation Strategies

To address the identified risks, the following mitigation strategies were implemented:

- **Asynchronous Processing:** The AI microservice executes all video analysis tasks asynchronously using background tasks, ensuring that the interface remains responsive during long-running operations and preventing processing time from affecting user experience.

- **Exponential Backoff:** Automatic retry mechanisms with exponential backoff are implemented to handle external API rate limit errors gracefully, reducing the impact of temporary service interruptions.
- **Modular Architecture:** The system is decomposed into three independent components interface, backend, and AI microservice each deployable and scalable independently, significantly reducing integration complexity and fault propagation.
- **Multilingual Testing:** The system was tested with video content in multiple languages including Arabic, French, English, and Spanish to validate the robustness of the speech recognition and content generation pipeline across diverse linguistic contexts.
- **Speech Recognition Optimization:** The speech recognition model configuration is exposed as an environment variable, allowing administrators to balance processing speed and accuracy based on available hardware resources.

2.7 Conclusion

This chapter presented a comprehensive analysis of the requirements and specifications that govern the development of the AI-Based Video Highlights Generator system.

We identified the primary actor (Authenticated User) and detailed the functional requirements covering eight key areas: user authentication, video input handling, AI-powered analysis, highlight detection, summary generation, collections management, history tracking, and statistics visualization. The non-functional requirements established quality benchmarks for performance, security, privacy, and usability.

The requirements and specifications established in this chapter serve as the foundation for the system design phase. The following chapter will translate these requirements into a detailed architectural design.

Chapter 3

Conception

3.1 Introduction

The design phase represents a critical step in the development of any software system, as it translates the requirements identified during the analysis phase into a structured and implementable solution. In this chapter, we present the overall design of the proposed system, detailing its architecture, components, and internal organization.

The objective of this chapter is to provide a clear and comprehensive description of how the system is structured and how its different parts interact to achieve the desired functionalities. This includes defining the system architecture, illustrating user interactions through UML diagrams, modeling the database structure, and identifying the main modules that compose the application.

The chapter is organized as follows: we first present the global system architecture, then we illustrate key system interactions using selected UML diagrams. After that, we describe the database design and structure, and finally, we define the main functional modules of the system and their respective roles.

3.2 System Architecture

3.2.1 Architectural Pattern

The system is built upon a combination of three complementary architectural patterns, each addressing a distinct aspect of the platform's design requirements.

The first pattern adopted is the **Client-Server** architecture, which establishes a clear separation between the user-facing interface and the server-side logic. The interface is responsible solely for rendering the user interface and handling user interactions, while all business logic, data management, and processing are delegated to the server side.

The second pattern is a **Lightweight Microservice-based** architecture, which decomposes the server side into two independent, specialized services: a backend service responsible for user management, authentication, and data persistence, and an AI microservice dedicated exclusively to video processing and intelligent content analysis.

The third pattern is **Representational State Transfer (REST)**, which governs all inter-component communication within the system. Every interaction between the interface and the backend, as well as between the backend and the AI microservice, is conducted through stateless HTTP requests using standard methods (GET, POST, PUT, and DELETE).

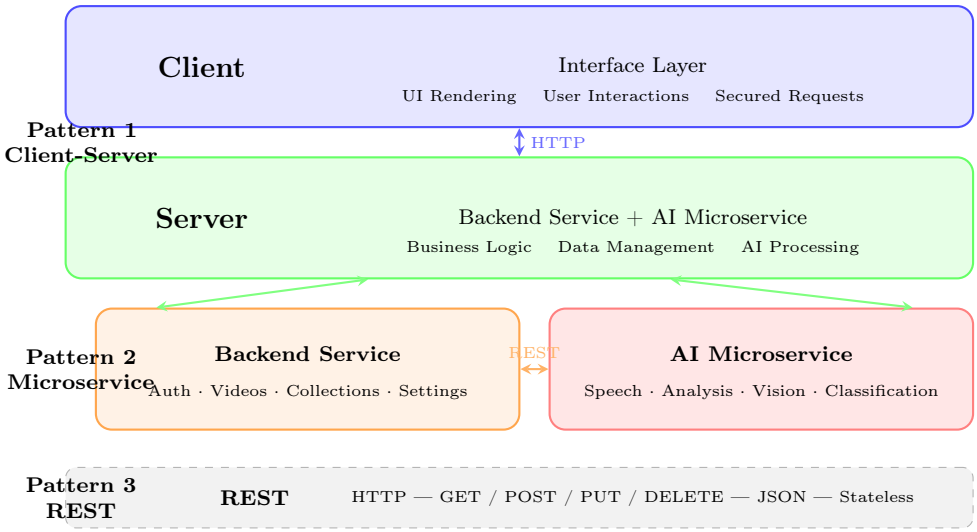

Figure 3.1: System Architectural Patterns

3.2.2 Global Architecture

Figure 3.2 provides a high-level overview of the system's global architecture, showing the four main modules and their primary interactions.

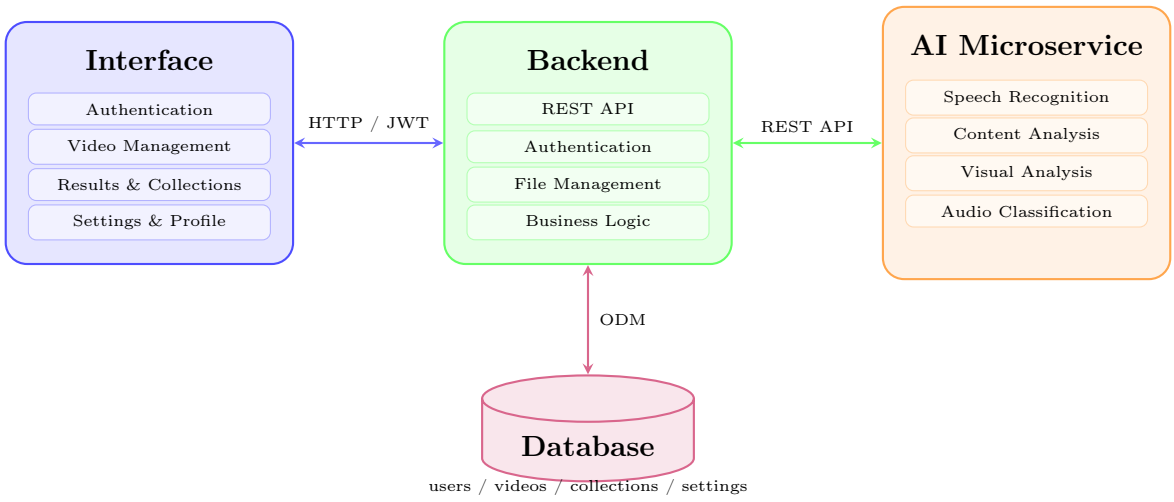

Figure 3.2: Global System Architecture

The global architecture is organized around four principal modules:

- **Interface:** The user-facing layer responsible for rendering all visual elements of the platform, managing user interactions, and communicating with the backend via authenticated HTTP requests.
- **Backend:** The central orchestrator of the system, handling authentication, video management, business logic, and communication between the interface and the AI microservice.
- **AI Microservice:** The intelligence core of the platform, executing the complete video analysis pipeline including speech recognition, content analysis, visual analysis, and audio classification asynchronously.

- **Database:** A cloud-hosted NoSQL database storing all persistent data including user accounts, video metadata, transcripts, summaries, chapters, and collection records.

3.2.3 Detailed Architecture

This section provides a detailed description of each system component and its internal organization, explaining the responsibilities, interactions, and technical design of every layer within the platform.

Interface Layer

The interface layer is implemented as a single-page application (SPA) responsible for rendering all visual elements of the platform. It handles all user interactions including authentication forms, video submission, results visualization, collections management, and profile settings.

Communication with the backend is handled through asynchronous HTTP requests, with authentication tokens attached to every request header to ensure secure, authenticated access. The interface is fully decoupled from the server side, enabling independent deployment and updates.

The interface supports three languages — English, French, and Arabic — with full right-to-left layout rendering when Arabic is selected.

Backend Layer

The backend layer provides a RESTful API that serves as the central orchestrator of the system. It is responsible for the following functions:

- **User Authentication:** Managing user registration, login, and session validation through token-based authentication and secure password hashing.
- **Video Management:** Handling file uploads, storing video metadata in the database, and managing the lifecycle of each video record.
- **AI Service Communication:** Forwarding video files and online video URLs to the AI microservice and receiving analysis results for storage.
- **Collections and History:** Providing endpoints for organizing videos into collections and maintaining a per-user processing history.

The backend exposes its API on a dedicated port and interacts with the database for all persistent data operations.

AI Microservice Layer

The AI microservice constitutes the core intelligence of the system. It operates as an independent service and is responsible for the entire video analysis pipeline, which proceeds as follows:

1. **Audio Classification:** The system detects the presence of an audio track and analyzes a short sample to classify the content as speech, music, or silence, determining which processing branch is activated.

2. **Audio Extraction:** The audio track is isolated from the uploaded video and converted into a format compatible with the speech recognition module.

3. **Frame Extraction:** When visual analysis is required, representative frames are extracted from the video at regular intervals for processing by the visual analysis module.

4. **Speech Recognition:** The speech recognition module transcribes the extracted audio into a timestamped text transcript, supporting automatic language detection across 99 languages.

5. **Visual Analysis:** The visual analysis module processes extracted frames and generates natural language descriptions of the visual content in multiple languages.

6. **Content Analysis:** The language model processes the transcript to generate structured chapter titles, abstractive summaries, and key points.

7. **Thumbnail Extraction:** A representative frame is extracted and encoded for storage in the database as a video preview.

8. **Result Transmission:** The complete analysis results — transcript segments, chapters, summaries, key points, and thumbnails — are transmitted back to the backend.

All processing tasks are executed asynchronously, enabling non-blocking request handling throughout the pipeline. When the language model service is unavailable, the system automatically falls back to extractive summarization to ensure service continuity.

Database Layer

The data layer is implemented using a document-oriented NoSQL database that stores data as flexible JSON-like documents. This database model was selected due to its natural compatibility with the variable-length, nested data structures produced by the AI analysis pipeline including transcript segments, chapter objects, and key point arrays which do not conform to a rigid relational schema.

The database is organized into four main collections: `users`, `videos`, `collections`, and `usersettings`. The `videos` collection embeds all analysis results directly within each video document, eliminating the need for additional joins and ensuring efficient data retrieval.

3.2.4 Communication Flow

The interaction between system components follows a structured pipeline that ensures reliable data exchange and processing:

1. The user accesses the platform and initiates registration by providing their credentials. Upon successful registration, a confirmation email is sent containing a verification link to activate the account.

2. Once authenticated, the user submits a video file or an online video URL through the interface via secured HTTP requests.

3. The interface forwards the request to the backend, which validates the submission and stores the initial video metadata in the database.

4. The backend forwards the video content to the AI microservice asynchronously, allowing the interface to remain responsive during processing.

   The AI microservice executes the full processing pipeline — audio classification, audio extraction, speech transcription, visual analysis, and content summarization — and transmits the complete results back to the backend.

5. The backend updates the database record with the analysis results and sets the video status to `completed`.

6. The interface periodically polls the backend and, upon detecting the completed status, retrieves and displays the generated transcript, chapters, summary, and highlights to the user.

3.3 UML Diagrams

3.3.1 Use Case Diagram

The use case diagram illustrates the relationships between the different functionalities of the platform through the use of `«include»` and `«extend»` associations. The `«include»` relationship represents mandatory dependencies between use cases, meaning that some functionalities cannot be executed without others. The `«extend»` relationship is used to represent optional or alternative behaviors, such as Google OAuth authentication as an extension of the standard login process.

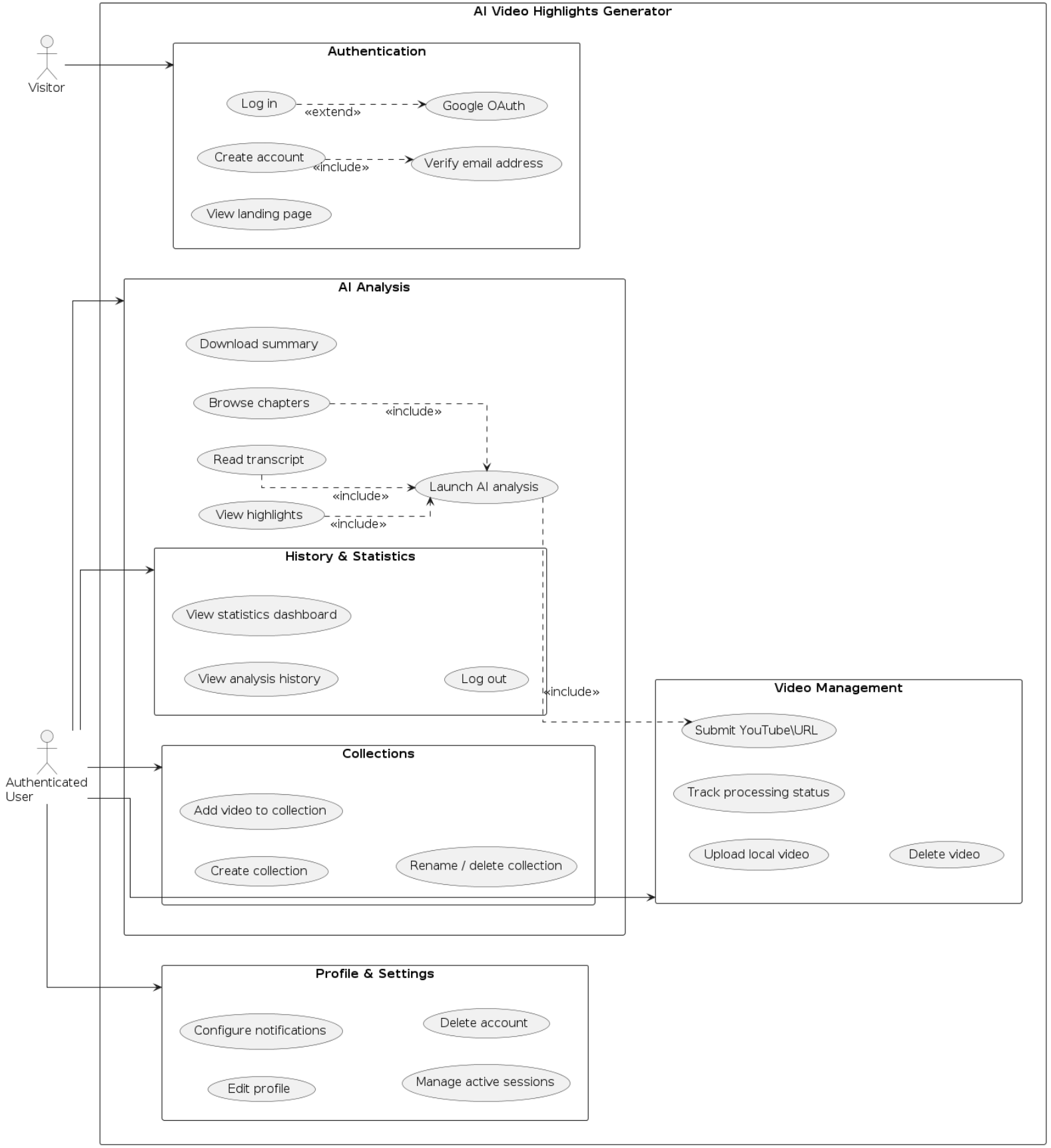

Figure 3.3: Use Case Diagram

3.3.2 Sequence Diagram

Video Processing Flow

The sequence diagram in Figure 3.4 illustrates the complete interaction flow triggered when a user submits a video for analysis, capturing the chronological exchange of messages between the React.js frontend, Node.js backend, Python AI microservice, MongoDB database, Whisper, and Groq API.

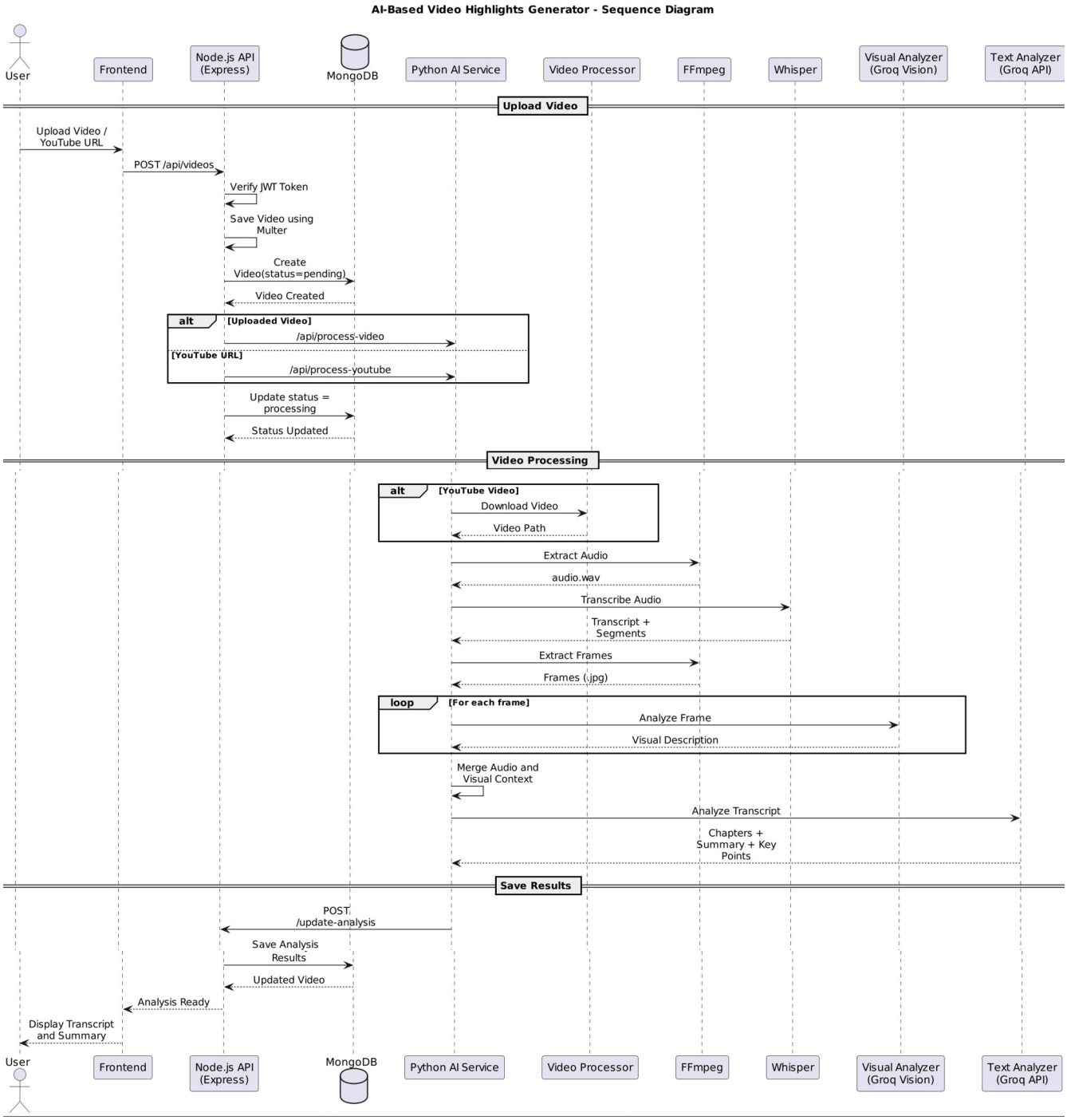

Figure 3.4: Sequence Diagram — Video Processing Flow

3.3.3 Class Diagram

The class diagram in Figure 3.5 represents the main data entities of the system and their relationships. The following paragraphs briefly describe the role of each class within the AI-Based Video Highlights Generator architecture.

User Class

The `User` class manages authentication, profile information, and account operations such as login, registration, and password reset.

Session Class

The `Session` class stores active sessions, connected devices, and authentication tokens for security management.

UserSettings Class

The `UserSettings` class stores user preferences including language, theme, timezone, and notification settings.

Video Class

The `Video` class represents uploaded or imported videos and stores metadata such as title, language, summary, transcript, and processing status.

TranscriptSegment Class

The `TranscriptSegment` class stores small parts of transcribed speech with timestamps.

Chapter Class

The `Chapter` class represents AI-generated video chapters with titles and summaries.

Highlight Class

The `Highlight` class stores important moments detected within videos along with timestamps.

Collection Class

The `Collection` class allows users to organize videos into thematic groups.

AIPipeline Class

The `AIPipeline` class coordinates the overall AI video analysis workflow.

AudioClassifier Class

The `AudioClassifier` class identifies the audio type such as speech, music, or silence.

AudioExtractor Class

The `AudioExtractor` class extracts audio tracks from uploaded videos.

FrameExtractor Class

The `FrameExtractor` class extracts frames used for visual analysis.

Transcriber Class

The `Transcriber` class converts speech into text using the Whisper model.

VisualAnalyzer Class

The `VisualAnalyzer` class analyzes video frames using AI vision models.

ContentAnalyzer Class

The `ContentAnalyzer` class generates summaries, chapters, and highlights from analyzed content.

ThumbnailExtractor Class

The `ThumbnailExtractor` class generates preview thumbnails for videos.

VideoProcessor Class

The `VideoProcessor` class handles preprocessing operations such as YouTube downloading.

WhisperModel Class

The `WhisperModel` class provides multilingual speech recognition capabilities.

GroqAPI Class

The `GroqAPI` class provides AI-based summary, chapter, and highlight generation services.

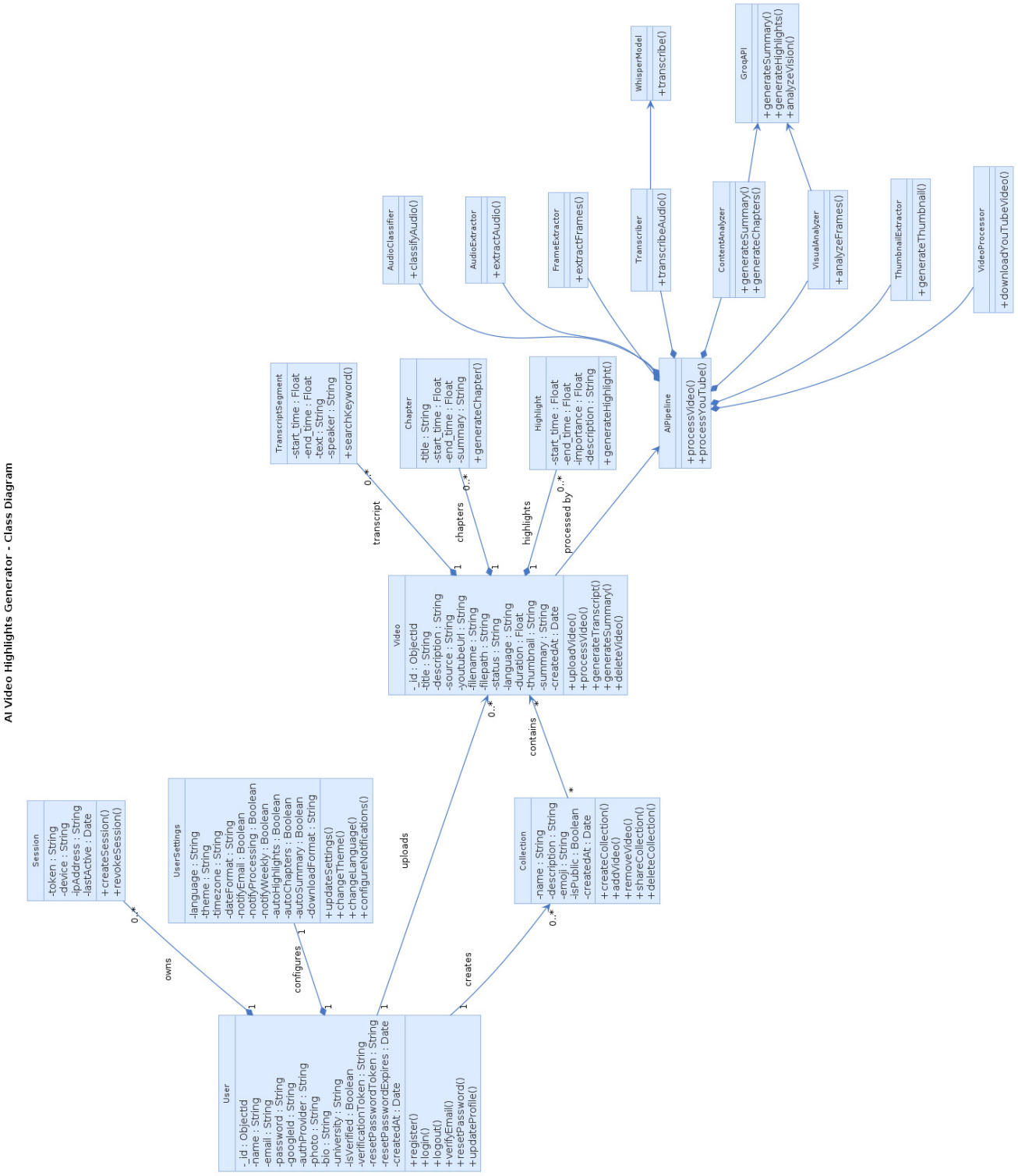

Figure 3.5: Class Diagram

3.3.4 Activity Diagrams

The following activity diagrams illustrate the main workflows of the AI-Based Video Highlights Generator system. Each diagram focuses on a specific process of the application, including authentication, video upload and AI processing, collection management, and user settings management.

Authentication and Verification Workflow

Figure 3.6 illustrates the user authentication workflow, including account registration, email verification, login, JWT generation, and session creation.

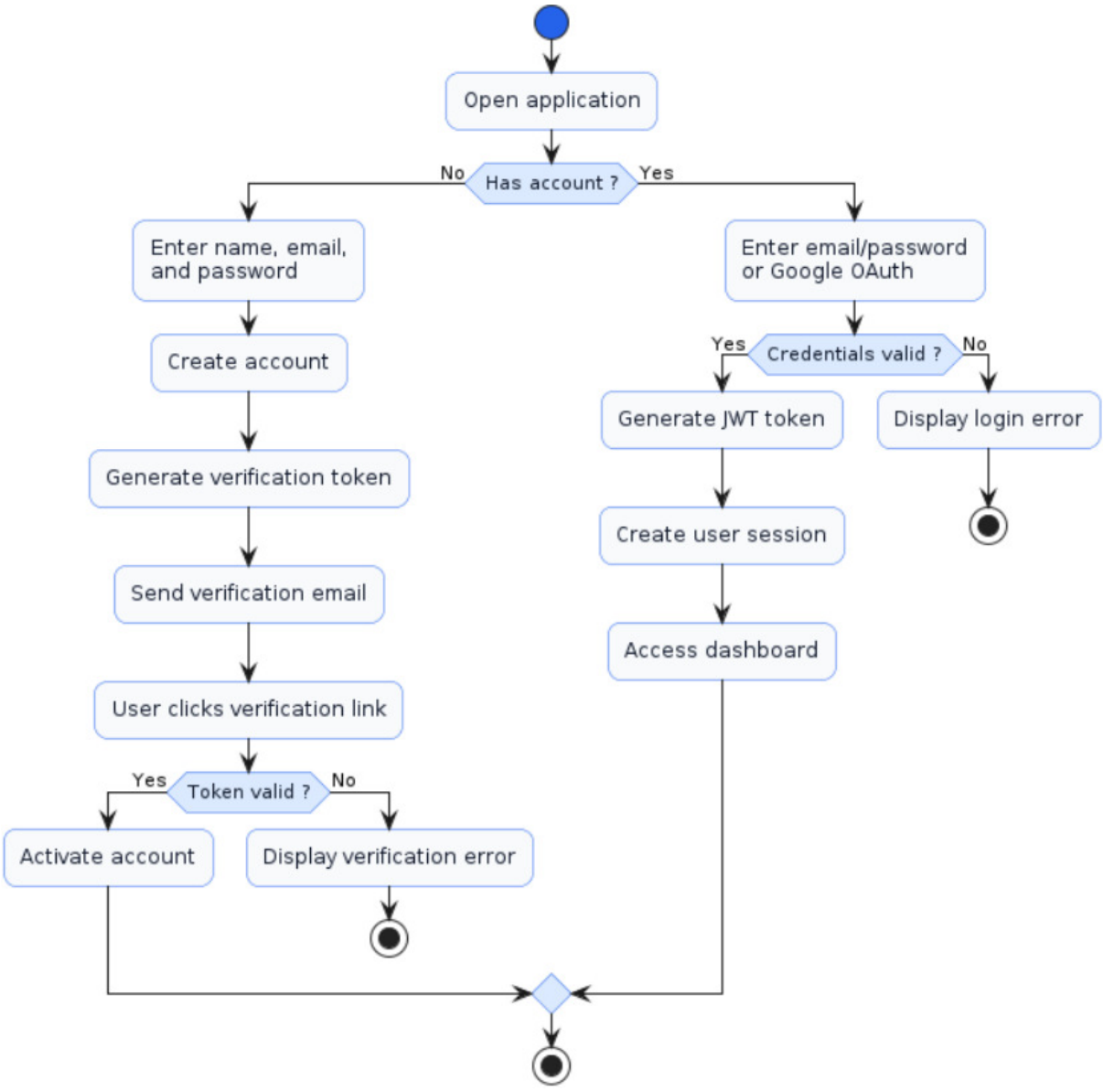

Figure 3.6: Activity Diagram – Authentication and Verification Workflow

Video Upload and AI Processing Workflow

Figure 3.7 presents the complete video upload and AI processing pipeline. The workflow includes video submission, audio extraction, speech recognition using Whisper, visual

analysis using Groq Vision, summary generation, and result storage.

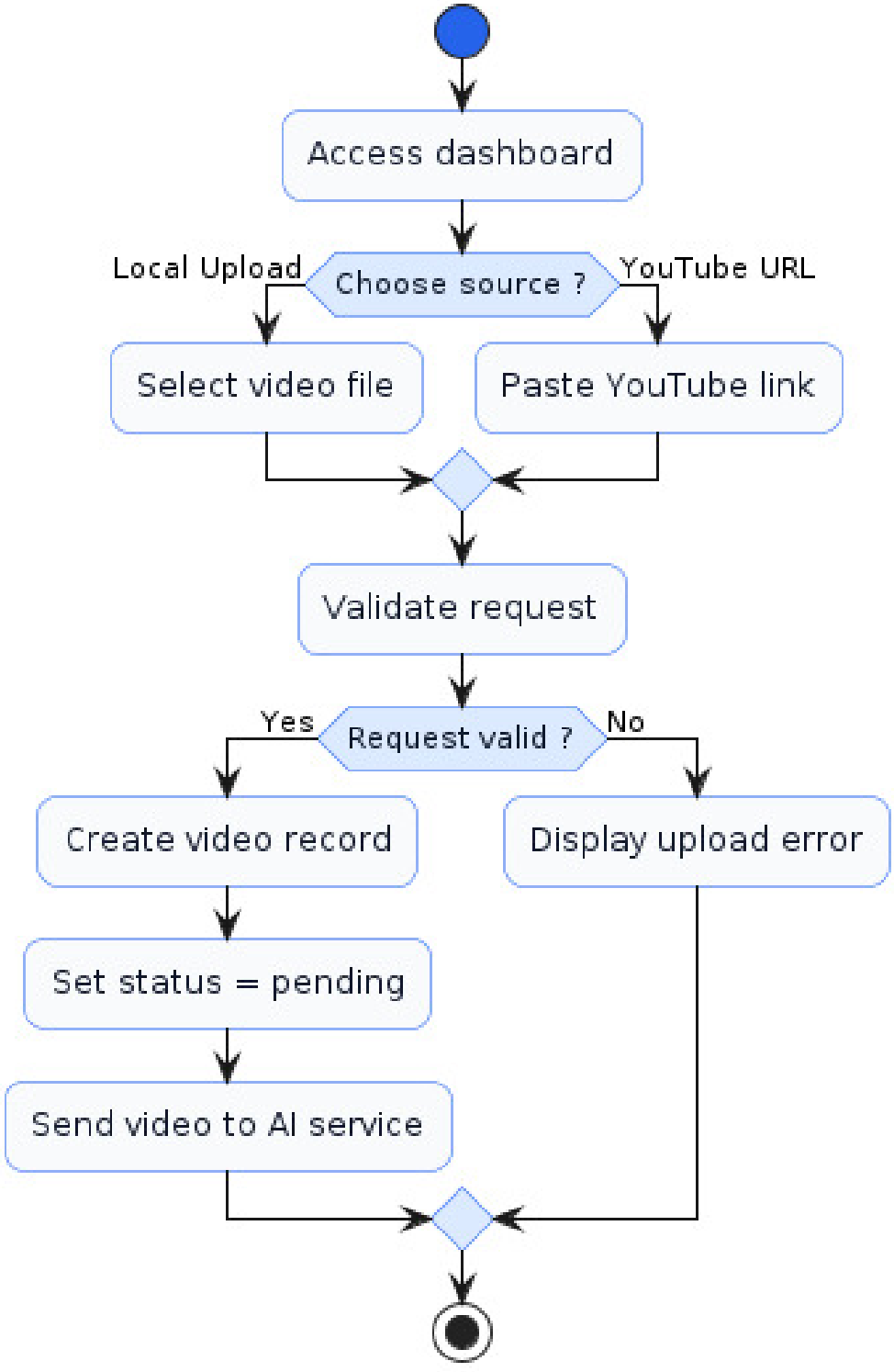

Figure 3.7: Activity Diagram – Video Upload and AI Processing Workflow

User Management and Settings Workflow

Figure 3.8 presents the workflow related to user personalization and content management features. The diagram includes collection creation and organization, notification preferences,

profile customization, and application settings management.

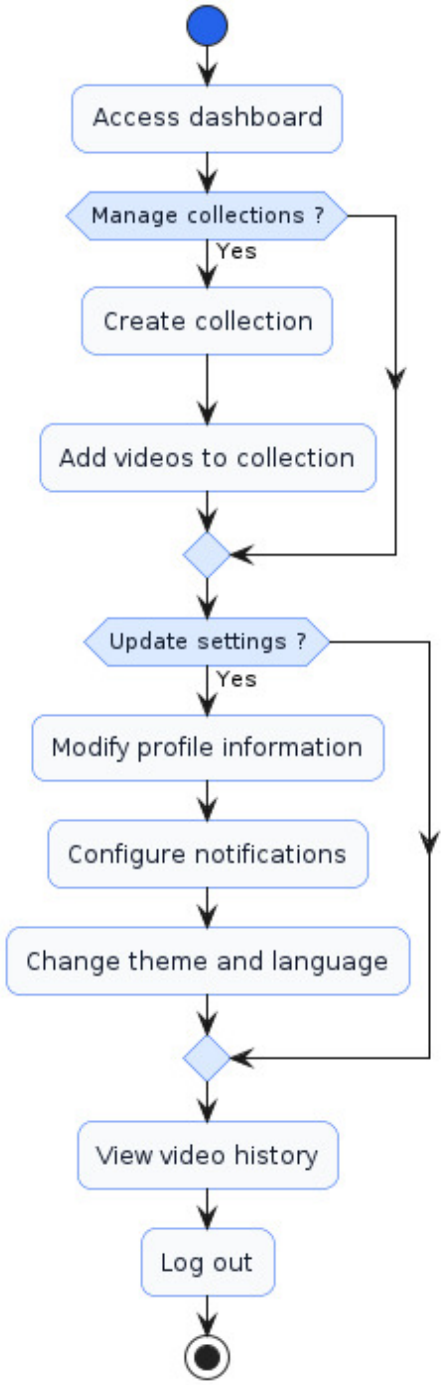

Figure 3.8: Activity Diagram – User Management and Settings Workflow

AI Processing Pipeline Workflow

Figure 3.9 illustrates the complete AI video analysis pipeline, including video submission, audio extraction, speech transcription, visual analysis, summary generation, and storage of the final results.

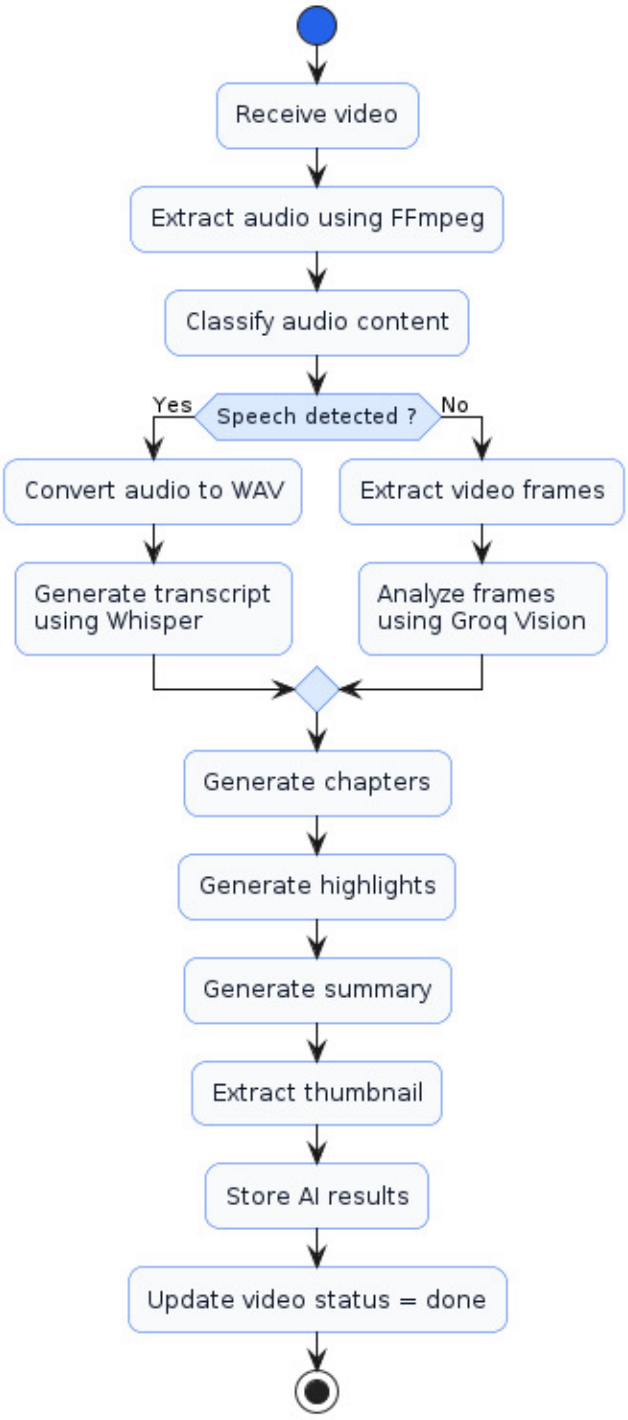

Figure 3.9: Activity Diagram – AI Processing Pipeline

3.4 Database Design

3.4.1 Database Schema

The system uses MongoDB as the primary NoSQL database for storing users, videos, collections, and user settings. MongoDB was selected due to its flexibility in handling semi-structured AI-generated data such as transcripts, chapters, summaries, and user preferences.

The database is organized into four main collections:

- `users`
- `videos`
- `collections`
- `usersettings`

Figure 3.10 illustrates the database schema and the relationships between the collections.

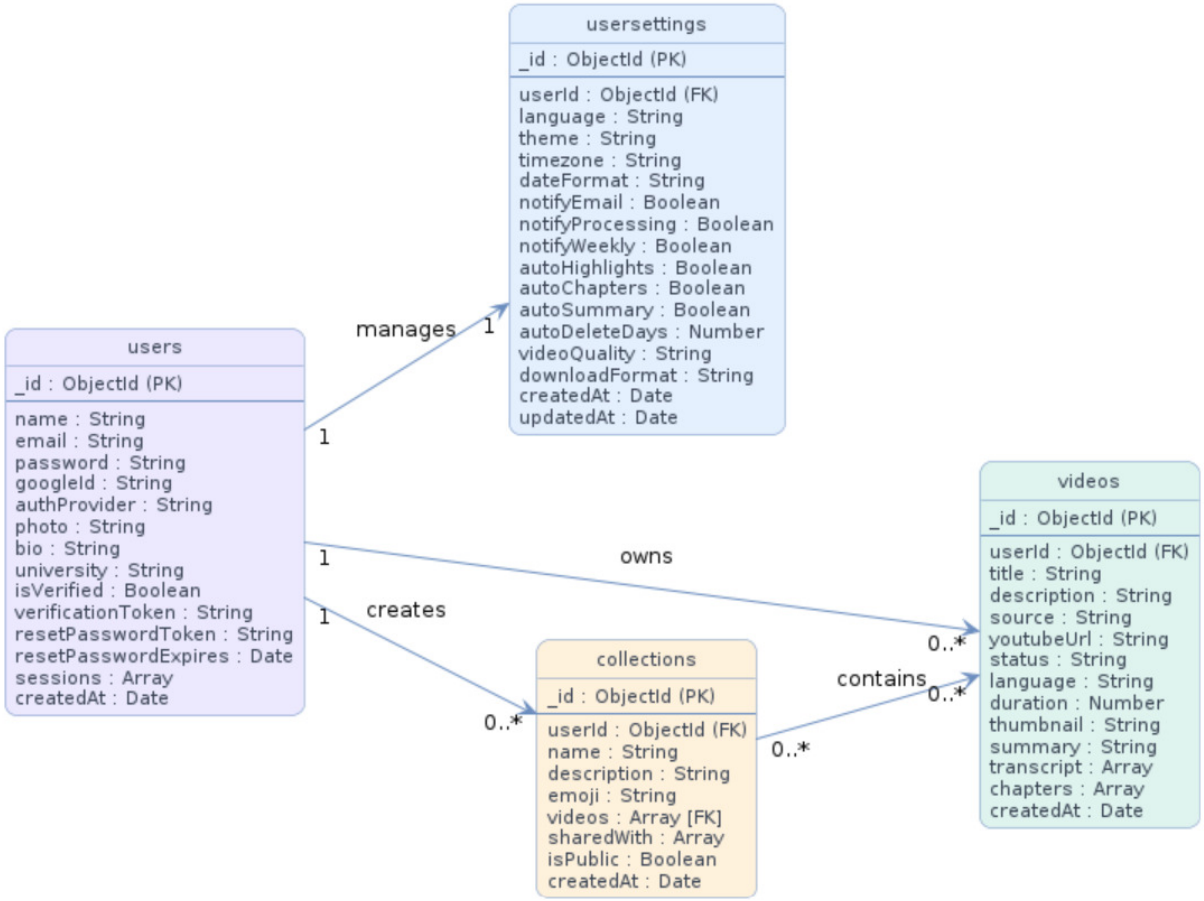

Figure 3.10: Database Schema

3.4.2 Collections Description

Users Collection

The `users` collection stores all registered user accounts and authentication information. The platform supports both local authentication using email/password and Google OAuth authentication.

Each document contains the following fields:

- **_id:** Unique MongoDB identifier
- **name:** User full name

- **email:** Unique email address
- **password:** Bcrypt hashed password for locally authenticated users
- **googleId:** Google account identifier for OAuth users
- **authProvider:** Authentication provider (`local` or `google`)
- **bio:** User biography
- **university:** User university or institution
- **photo:** Profile picture URL or base64 image
- **isVerified:** Email verification status
- **verificationToken:** Verification token used during email confirmation
- **resetPasswordToken:** Password reset token
- **resetPasswordExpires:** Expiration date for password reset
- **sessions:** Active login sessions
- **createdAt:** Account creation timestamp

Videos Collection

The `videos` collection stores uploaded or YouTube video records together with all AI-generated analysis results.

Each document contains:

- **_id:** Unique MongoDB identifier
- **userId:** Reference to the owner user
- **title:** Video title or YouTube URL
- **description:** Video description
- **source:** Video source (`upload` or `youtube`)
- **youtubeUrl:** Original YouTube link
- **status:** Processing state (`pending`, `processing`, `done`, `error`)
- **language:** Detected video language
- **duration:** Video duration in seconds
- **summary:** AI-generated abstractive summary
- **transcript:** Array of timestamped transcription segments
- **chapters:** Array of generated chapters with timestamps and summaries
- **createdAt:** Upload date and time

Collections Collection

The `collections` collection allows users to organize videos into custom folders for easier access and sharing.

Each document contains:

- **_id:** Unique MongoDB identifier
- **userId:** Reference to the owner user
- **name:** Collection name
- **description:** Collection description
- **emoji:** Emoji icon representing the collection
- **videos:** Array of referenced video IDs
- **isPublic:** Visibility status
- **createdAt:** Collection creation timestamp

UserSettings Collection

The `usersettings` collection stores personalized platform preferences and AI processing configurations for each user.

Each document contains:

- **_id:** Unique MongoDB identifier
- **userId:** Reference to the associated user
- **language:** Preferred interface language
- **theme:** Interface theme (`light` or `dark`)
- **timezone:** User timezone configuration
- **dateFormat:** Preferred date format
- **displayName:** Optional custom display name
- **avatarUrl:** User avatar URL
- **notifyEmail:** Email notification preference
- **notifyProcessing:** Processing completion notifications
- **notifyWeekly:** Weekly digest notifications
- **videoQuality:** Preferred video quality
- **autoHighlights:** Automatic highlights generation
- **autoChapters:** Automatic chapter generation

- **autoSummary:** Automatic summary generation
- **autoDeleteDays:** Automatic deletion delay
- **downloadFormat:** Preferred download format
- **createdAt:** Settings creation date
- **updatedAt:** Last update timestamp

3.5 Conclusion

This chapter presented the complete design of the AI-Based Video Highlights Generator system. The three-tier microservice architecture ensures clear separation of concerns between the user interface, business logic, and AI processing. The UML diagrams illustrated the key interactions and structural organization of the system, while the database design demonstrated how MongoDB's flexible document model accommodates the complex, variable-length data produced by the AI pipeline. The module definitions established a clear decomposition of responsibilities that guided the implementation phase presented in the following chapter.

Chapter 4

Implementation and Development

4.1 Introduction

This chapter presents the implementation and development process of the AI-Based Video Highlights Generator system. Based on the analysis and design presented in the previous chapters, we describe the technologies, frameworks, and development tools used to build the application. We also explain the implementation of the main system modules and illustrate the obtained results through screenshots and practical demonstrations of the platform functionalities.

The system was developed using a modular architecture composed of three interconnected layers: a React.js frontend, a Node.js backend, and a Python-based AI microservice. Each component was implemented and tested separately before being integrated into a complete and fully operational web application.

Special attention was given to system scalability, maintainability, and performance during the development process. The integration of artificial intelligence models such as Whisper and Llama required efficient communication between services and optimized handling of multimedia data.

4.2 Development Environment

This section outlines the tools, technologies, and platforms used throughout the development of the AI-Based Video Highlights Generator. Choosing the right development environment was essential to ensure efficient collaboration, scalability, and compatibility across the three system components.

4.2.1 Development Tools

Visual Studio Code

Visual Studio Code is a lightweight, open-source code editor developed by Microsoft. It offers built-in support for debugging, Git integration, intelligent code completion, and a rich ecosystem of extensions. Its versatility across multiple programming languages and frameworks made it the primary development environment used throughout this project [37].



Figure 4.1: VS Code Logo

4.2.2 Frontend Development

React.js

React.js is an open-source JavaScript library developed by Meta for building dynamic and component-based user interfaces. Its declarative architecture and virtual DOM allow efficient rendering and responsive user experiences. React's component reusability and state management capabilities made it well suited for building the interactive dashboard, video upload interface, and results display of this platform [39].

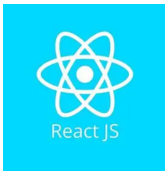

Figure 4.2: React.js Logo

Tailwind CSS

Tailwind CSS is a utility-first CSS framework used to design responsive and modern user interfaces. It accelerates frontend development by providing reusable utility classes directly within HTML and JSX components. Its flexible design system enabled the team to rapidly build a consistent and visually appealing interface across all pages of the platform, including light and dark theme support [40].



Figure 4.3: Tailwind CSS Logo

4.2.3 Backend Development

Node.js

Node.js is an open-source, cross-platform runtime environment that allows developers to execute JavaScript code on the server side. Built on the V8 JavaScript engine, it is designed for building scalable network applications and is known for its event-driven, non-blocking I/O model [41].



Figure 4.4: Node.js Logo

Express.js

Express.js is a minimalist and flexible backend framework for Node.js that simplifies API development and request handling. It was used to define all backend routes, middleware, authentication logic, and communication with both the database and the AI microservice. Its lightweight structure allowed rapid development of a clean and well-organized RESTful API [42].



Figure 4.5: Express.js Logo

4.2.4 AI Microservice Development

Python

Python is a high-level, general-purpose programming language widely used in artificial intelligence and machine learning applications. It was selected as the main language for the AI microservice due to its extensive ecosystem of AI and data processing libraries, including Whisper, FFmpeg bindings, and HTTP clients for communicating with the Groq API [44].



Figure 4.6: Python Logo

FastAPI

FastAPI is a modern and high-performance Python framework for building APIs. Its asynchronous architecture allows efficient handling of long-running AI processing tasks while maintaining responsiveness. FastAPI's background task mechanism was used to run the entire video analysis pipeline asynchronously, ensuring that the main API remained responsive while Whisper and Groq processed the uploaded content [45].



Figure 4.7: FastAPI Logo

4.2.5 AI and Media Processing Tools

FFmpeg

FFmpeg is an open-source multimedia framework used for audio extraction, video preprocessing, and thumbnail generation. It converts uploaded videos into formats compatible with the AI pipeline, handling decoding, audio normalization, and frame extraction efficiently. In our system, FFmpeg isolates the audio track, extracts representative frames for visual analysis, and generates the video thumbnail stored in the database [35].



Figure 4.8: FFmpeg Logo

OpenAI Whisper

Whisper is an automatic speech recognition model developed by OpenAI. It generates accurate timestamped transcripts across multiple languages and serves as the speech-to-text engine of the platform. The model supports 99 languages and produces structured output with precise start and end timestamps for each spoken segment [20].



Figure 4.9: Whisper Logo

Groq API

The Groq API provides fast inference for large language models using dedicated LPU (Language Processing Unit) hardware. It was used to generate summaries, chapter titles, key points, and visual analysis results. Unlike traditional GPU-based inference, Groq's LPU architecture is specifically optimized for sequential token generation, delivering significantly lower latency for all NLP tasks in the pipeline [11].



Figure 4.10: Groq API Logo

4.2.6 Database

MongoDB Atlas

MongoDB Atlas is a cloud-based NoSQL database service used for storing users, videos, transcripts, summaries, and generated highlights. Its flexible JSON-like document model is well suited for AI-generated multimedia data, accommodating variable-length nested structures such as transcript segments, chapter arrays, and key point lists without requiring a rigid relational schema [46].



Figure 4.11: MongoDB Logo

4.2.7 API Testing

Postman

Postman is an API development and testing platform used to validate backend endpoints and inspect HTTP requests and responses. It played an important role during debugging and integration testing, allowing the team to simulate authenticated requests, inspect JSON payloads, and verify the correct behavior of all RESTful routes before frontend integration [47].



Figure 4.12: Postman Logo

4.2.8 Version Control and Collaboration

GitHub

GitHub is a cloud-based platform for version control and collaborative software development built on top of Git. Throughout this project, GitHub was used to manage the source code of all three system components, track changes through structured commits, and coordinate development across team members. It also served as a centralized backup and documentation hub for the entire project repository [38].



Figure 4.13: GitHub Logo

4.2.9 Mobile Application

React Native & Expo

React Native is a cross-platform framework developed by Meta for building native mobile applications from a single JavaScript codebase. Combined with Expo, a development platform that simplifies build tooling and provides access to native device APIs, the mobile client of VeoIA was developed for Android devices. The application shares core logic with the web interface while delivering a native mobile experience, including user authentication, video URL submission, results display with chapters and summaries, collections management, notification system, and user settings [43].



Figure 4.14: Expo Logo

Render

Render is a cloud hosting platform used to deploy and host the backend services of VeoIA. It provides automatic deployments, secure hosting, monitoring tools, and continuous availability of the API used by both the web and mobile applications. Its simplicity and integration with Git repositories facilitated the deployment and maintenance of the platform throughout the development process [48].



Figure 4.15: Render Logo

Railway

Railway is a cloud deployment platform that simplifies the hosting and management of backend applications and databases. In the VeoIA project, Railway was used to deploy backend services and manage the development environment through an easy-to-use interface. The platform provides automatic deployments, environment variable management, database integration, and monitoring capabilities, allowing the team to focus on development rather than infrastructure configuration [49].



Figure 4.16: Railway Logo

4.3 System Components

This section presents the main components that constitute the implemented system. Each component corresponds to an independent layer of the application architecture, contributing specific functionalities to the overall platform.

4.3.1 Frontend Component

React.js Application

The frontend component is a fully responsive single-page application (SPA) built with React.js and styled using Tailwind CSS. It provides users with an intuitive interface to interact with all platform features. The component is organized into the following modules:

- **Authentication Module:** Handles user registration, login, email verification, password reset, and Google OAuth integration. All authentication flows are managed via JWT tokens stored securely on the client side.
- **Video Management Module:** Provides drag-and-drop file upload for local videos and a URL submission form for YouTube links. A real-time progress tracker displays the current processing stage.
- **Results Display Module:** Renders the complete analysis output including AI-generated chapters with timestamps, the full searchable transcript, key points, and the abstractive summary.
- **Collections Module:** Allows users to create topic-based collections, assign analyzed videos to collections, and share collections with other users via email.
- **History Sidebar:** Displays a chronological list of all previously analyzed videos, enabling instant access to past results without navigating away from the current view.
- **Settings and Profile Module:** Provides customizable preferences including interface language (English, French, Arabic), visual theme (light/dark), timezone, notification settings, and profile information.
- **Multilingual Module:** Implements full internationalization via React Context API, dynamically updating all UI elements when the language is changed. Arabic mode enables right-to-left (RTL) layout rendering.

4.3.2 Backend Component

Node.js REST API

The backend component is implemented with Node.js and Express, exposing a RESTful API organized into four route groups. It serves as the central coordinator between the frontend, the AI microservice, and the MongoDB database.

- **Authentication Routes (/api/auth):** Handle user registration with email verification, JWT-based login and logout, Google OAuth callback, password reset, session management, and profile updates. Bcrypt hashing protects all stored passwords.
- **Video Routes (/api/videos):** Manage video creation, retrieval, status polling, analysis result storage, thumbnail management, and deletion. Multer middleware handles multipart file uploads before passing them to the AI service.
- **Collection Routes (/api/collections):** Provide CRUD operations for collections, video assignment, and sharing functionality.
- **Settings Routes (/api/settings):** Persist and retrieve user preferences including language, theme, timezone, and AI processing configurations.

4.3.3 AI Microservice Component

Python FastAPI Service

The AI microservice is the core intelligence of the platform. It receives requests from the Node.js backend and executes the complete video analysis pipeline asynchronously. The pipeline consists of eight sequential stages:

1. **Audio Classification:** A 30-second audio sample is analyzed by Whisper to detect whether the video contains speech, music, or silence, adapting the processing strategy accordingly.
2. **Audio Extraction:** FFmpeg isolates and normalizes the audio track to a 16kHz mono WAV file for Whisper compatibility.
3. **Frame Extraction:** For videos requiring visual analysis, FFmpeg captures frames at regular intervals for Groq Vision processing.
4. **Speech Recognition:** Whisper `base` model transcribes the audio, producing timestamped segments with automatic language detection across 99 languages.
5. **Visual Analysis:** The `llama-4-scout-17b-16e-instruct` model analyzes extracted frames and generates descriptive text about visual content.
6. **Content Summarization:** The `llama-3.3-70b-versatile` model processes the transcript and generates chapter titles, key points, and an abstractive summary.
7. **Thumbnail Generation:** FFmpeg extracts a frame at the third second, encoded as base64 for database storage.
8. **Result Delivery:** The complete analysis payload is sent back to the Node.js backend via HTTP POST.

4.3.4 Database Component

MongoDB

The MongoDB database stores all persistent data generated by the platform, organized into four collections. The flexible document model allows each video record to embed its full analysis output transcript segments, chapter objects, summary text, and thumbnail within a single document, minimizing query complexity and ensuring fast retrieval.

4.4 Module Implementation

4.3.1 Web Application Interfaces

Landing Page

Figure 4.17 presents the landing page of the application, which serves as the public entry point introducing the platform's core features and directing visitors to register or sign in.

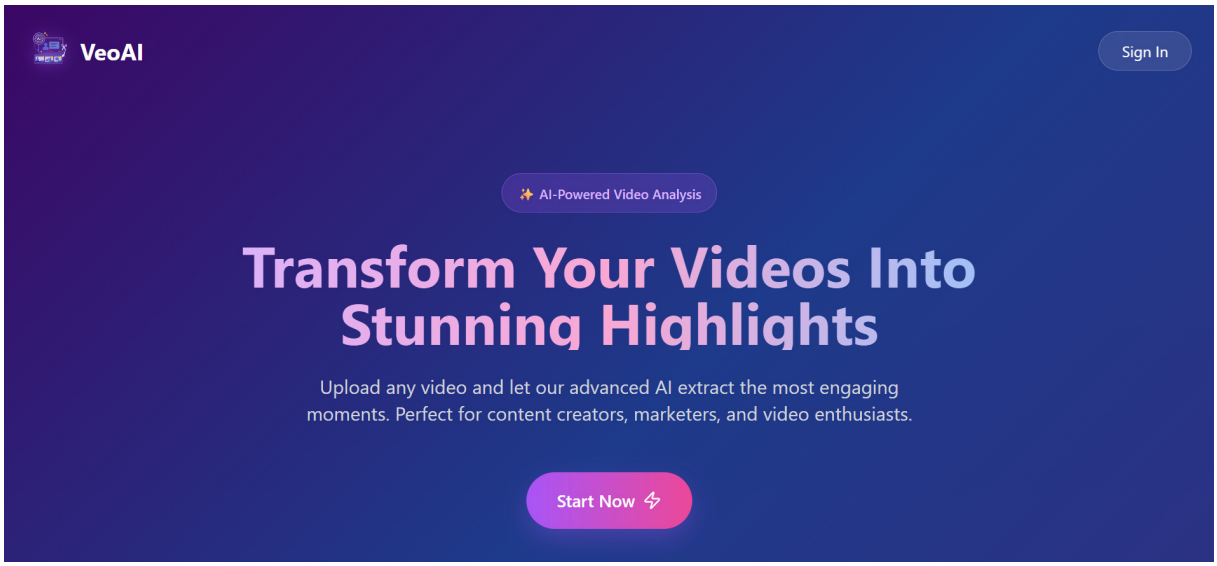

Figure 4.17: Landing Page

Authentication Modal

Figure 4.18 presents the authentication modal displayed when users access the platform. It serves as the entry point to the authentication process, allowing users to either sign in to an existing account, create a new account, or continue directly using Google OAuth authentication for a faster and more convenient experience.

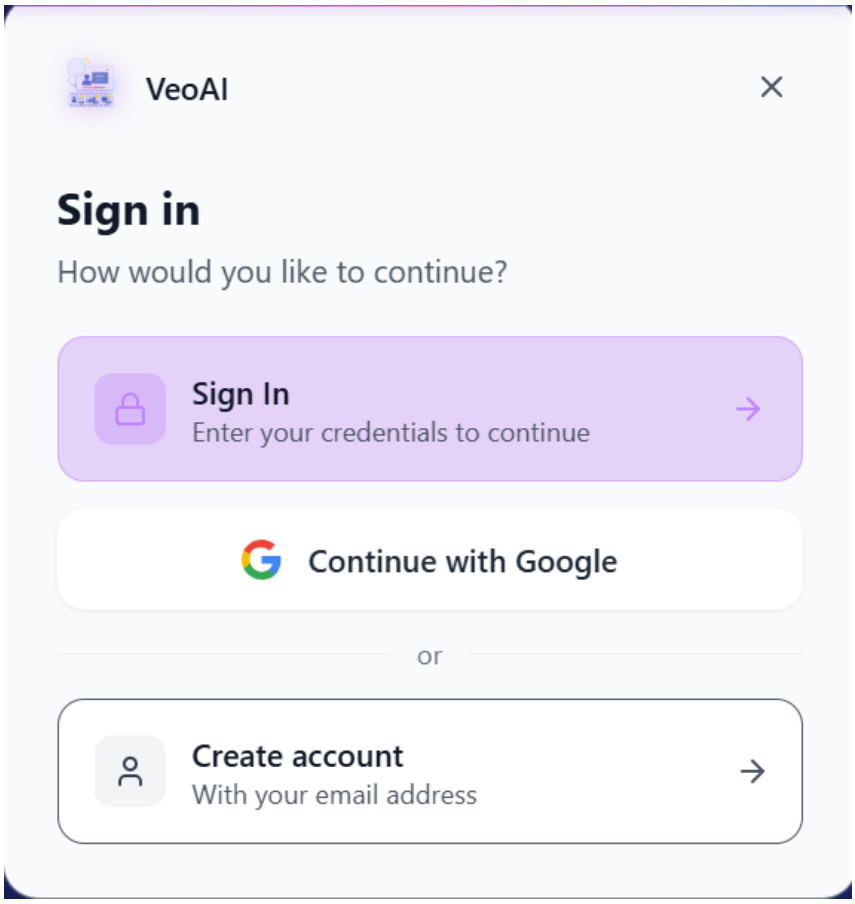

Figure 4.18: Authentication Modal

Login Interface

Figure 4.19 illustrates the login interface allowing users to authenticate using their email and password credentials or through Google OAuth authentication for faster access.

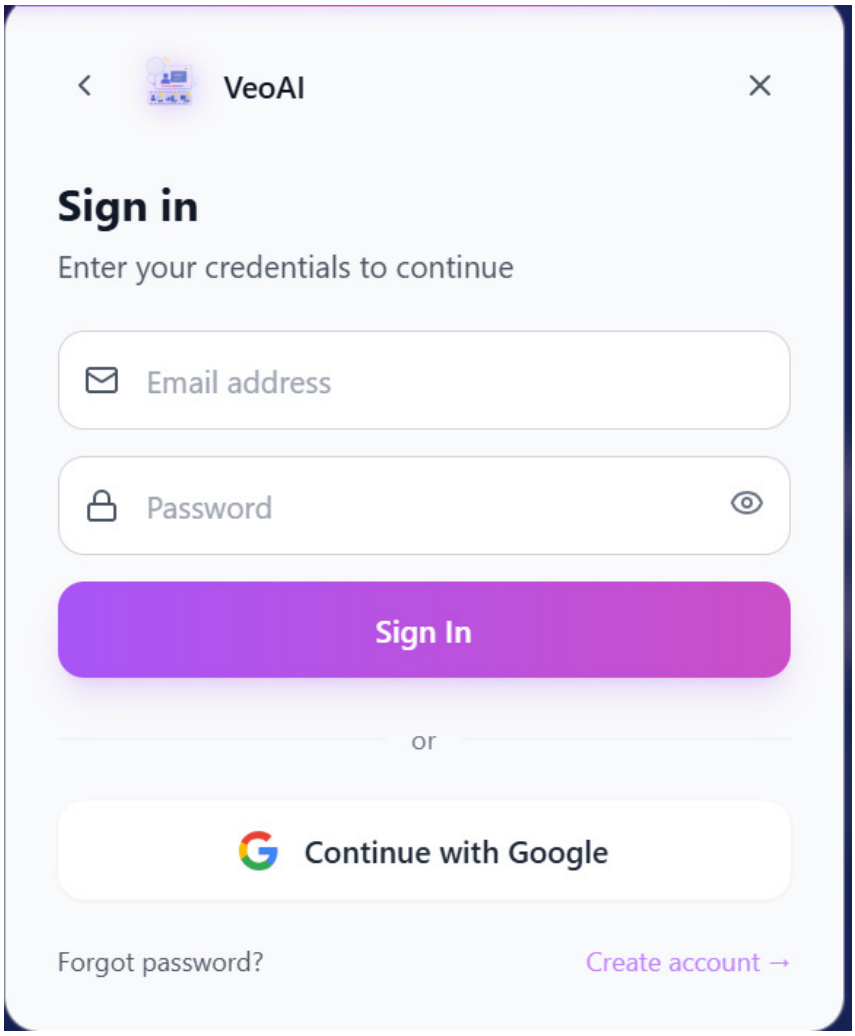

Figure 4.19: Login Interface

Registration Interface

The registration interface allows new users to create an account by providing their name, email address, and password, or by continuing with Google OAuth integration for a faster onboarding experience.

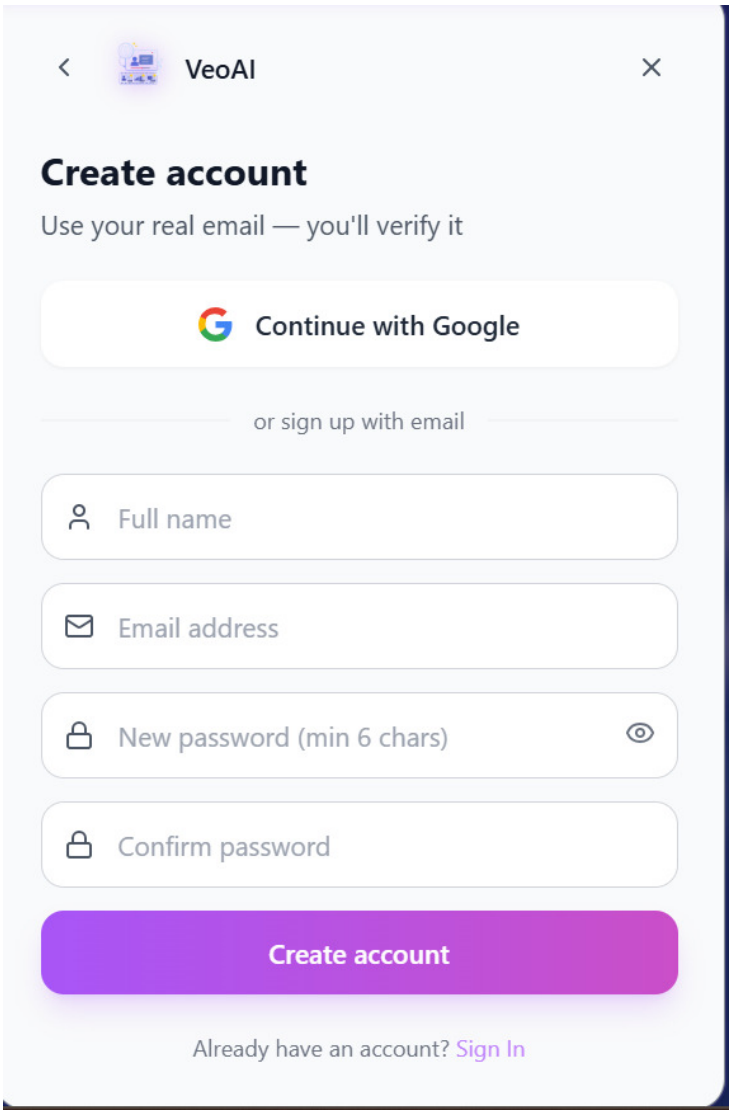

Figure 4.20: Registration Interface

Email Verification Interface

After registration, the system automatically sends a verification link to the user's email address. The user must click this link to activate their account before accessing the platform features.

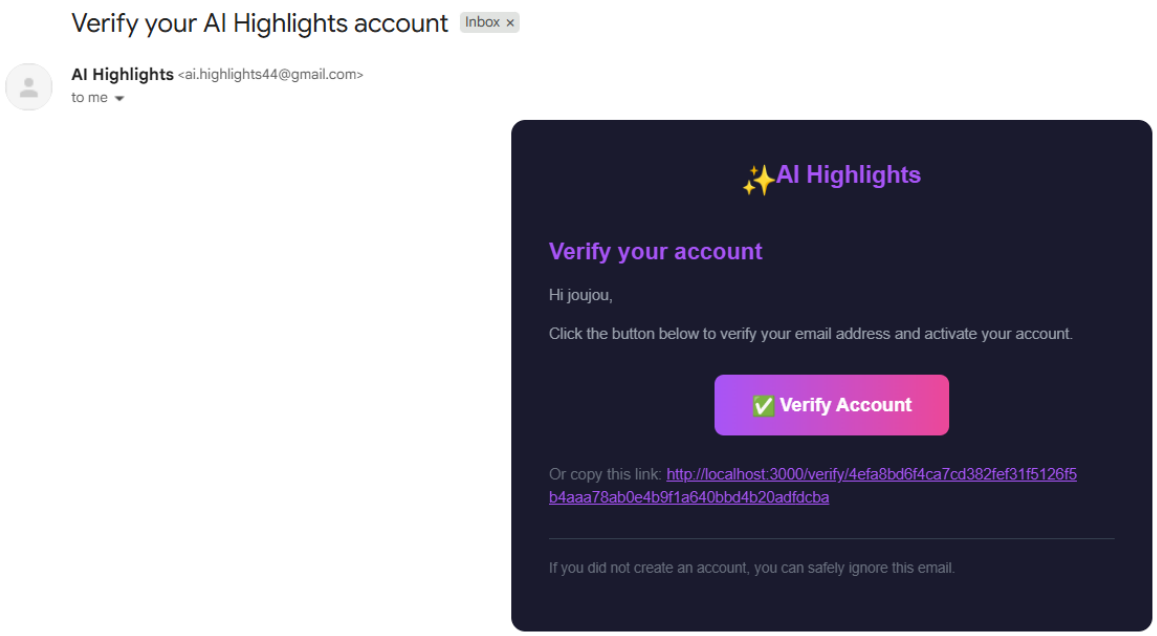

Figure 4.21: Email Verification Interface

Forgot Password Interface

The password recovery interface allows users to request a secure password reset link sent directly to their registered email address, enabling them to restore access without contacting support.

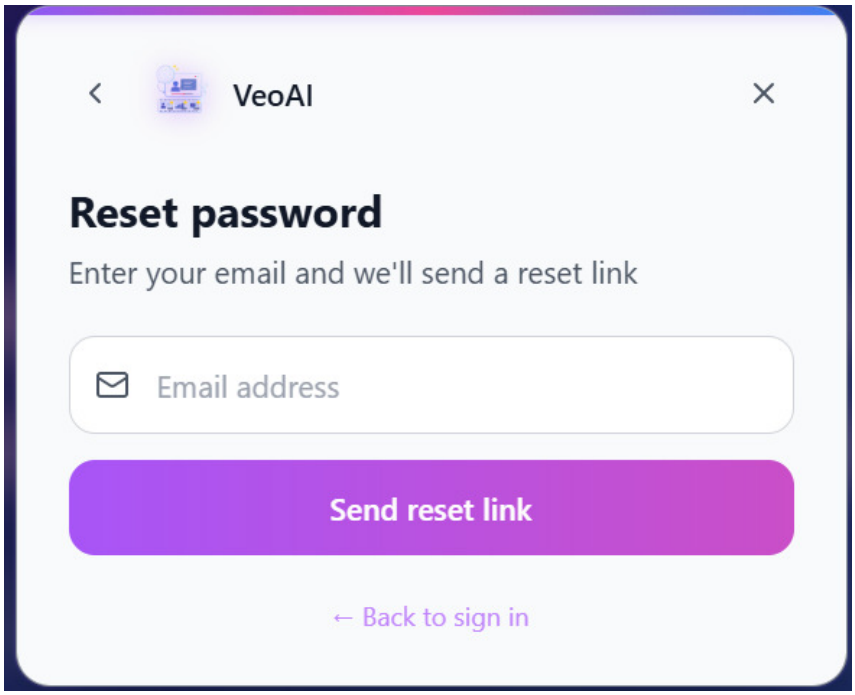

Figure 4.22: Forgot Password Interface

Password Reset Interface

The password reset page allows users to securely define a new password using the tokenized reset link received by email. The token is validated server-side before any changes are applied.

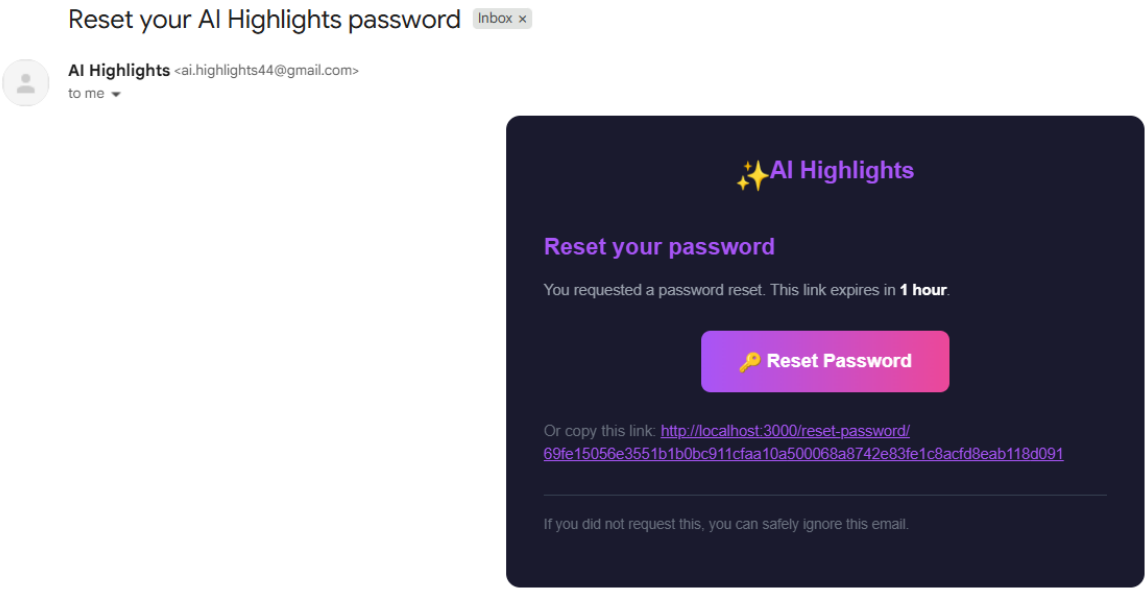

Figure 4.23: Password Reset Interface

Google OAuth Authentication

The Google OAuth authentication flow enables users to access the platform quickly without creating a traditional account manually. Users are redirected to Google's authentication page and returned to the platform upon successful verification.

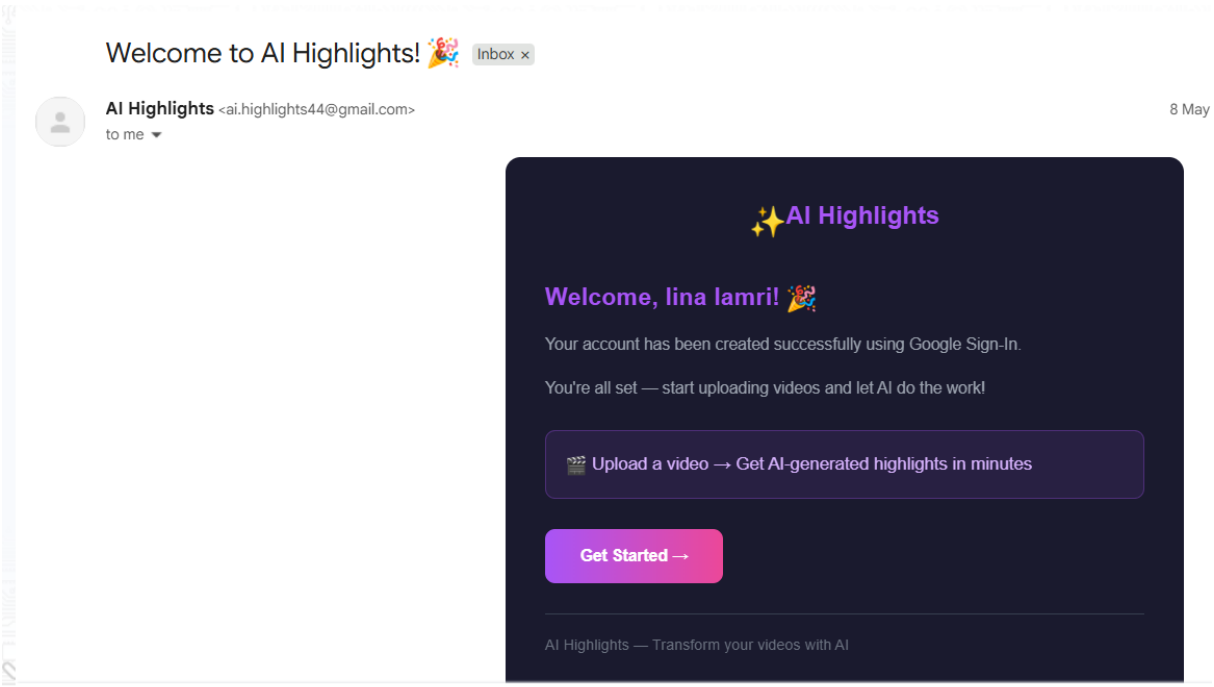

Figure 4.24: Google OAuth Authentication Flow

Video Upload Interface

Figure 4.25 presents the upload interface allowing users to submit local video files or YouTube URLs, with real-time progress tracking for each stage of the processing pipeline.

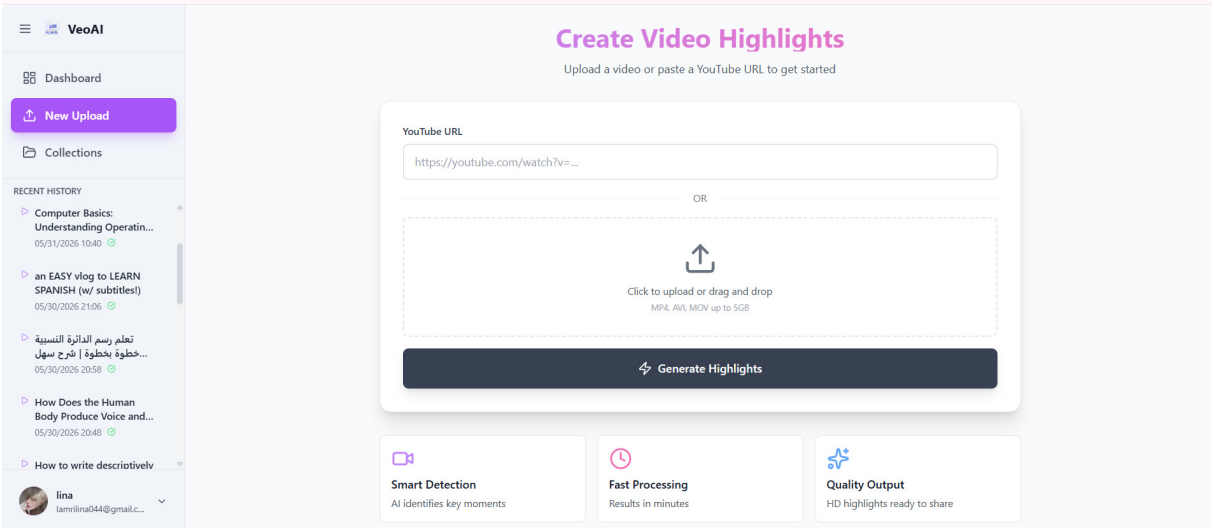

Figure 4.25: Video Upload Interface with Progress Tracking

User Dashboard

The main dashboard displays all analyzed videos with thumbnails, processing status badges, chapter counts, and filtering and search capabilities.

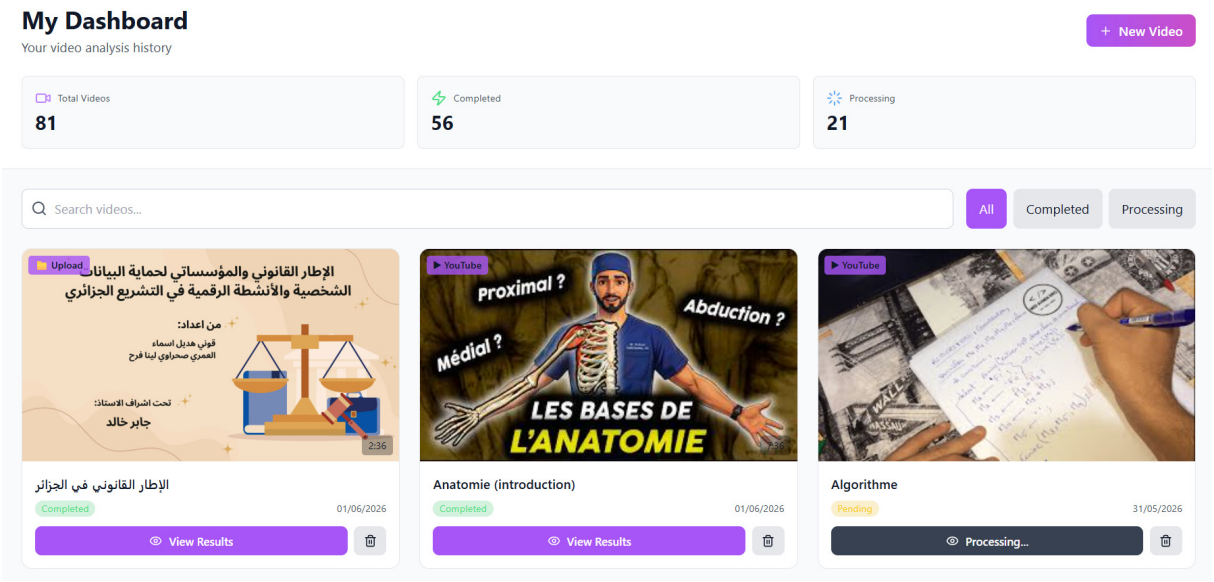

Figure 4.26: User Dashboard

Analysis Results Page

Figure 4.27 presents the main analysis results page. It serves as a centralized workspace where users can access generated chapters, transcripts, summaries, chapter details, and export functionalities after video processing.

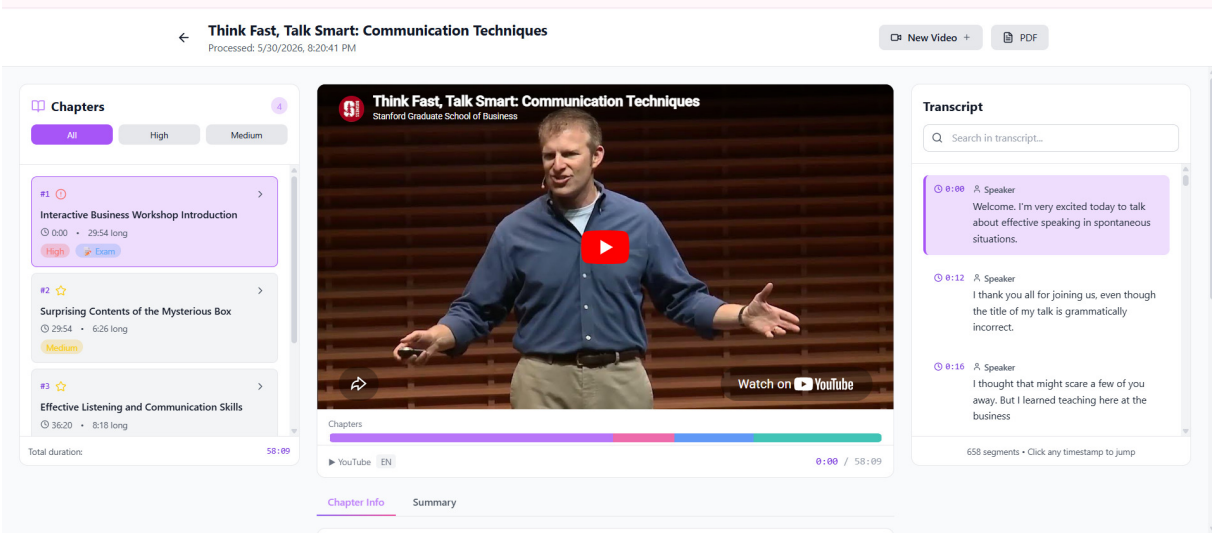

Figure 4.27: Analysis Results Page

Chapters Sidebar

The chapters sidebar provides a structured list of AI-generated chapters with their corresponding timestamps. Users can quickly navigate to specific sections of the video by selecting a chapter from the sidebar.

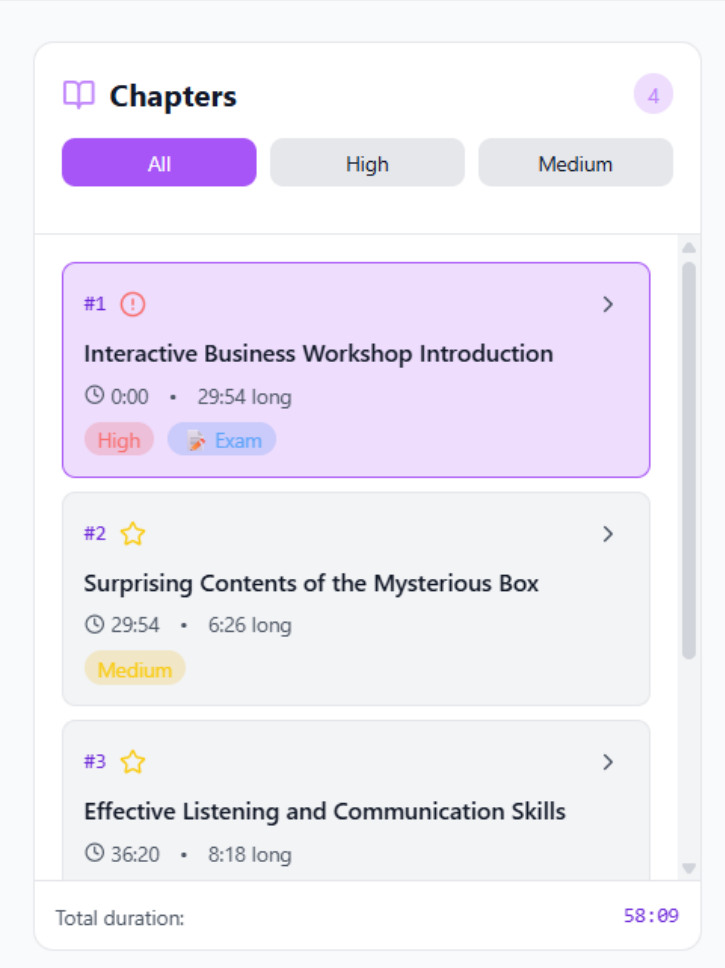

Figure 4.28: Chapters Sidebar

Transcript Section

The transcript section displays the complete video transcription generated by the speech-to-text system. It supports keyword search and highlighting to help users locate relevant information efficiently.

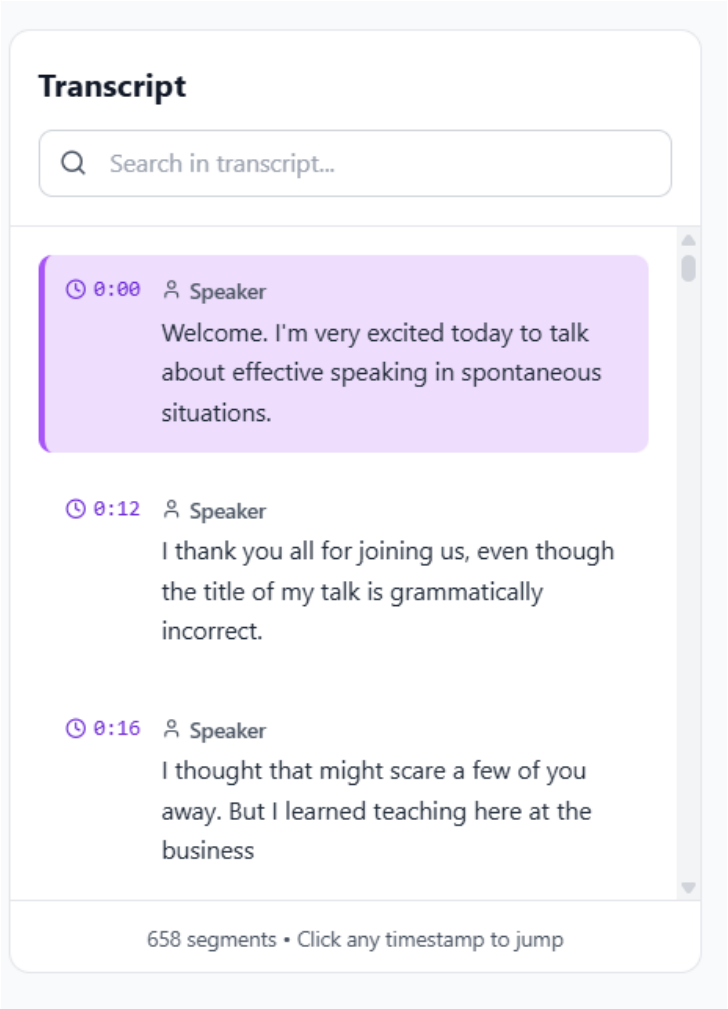

Figure 4.29: Transcript Section

Summary Section

The summary section presents an abstractive summary generated using the Groq API, providing users with a concise overview of the video's key points.

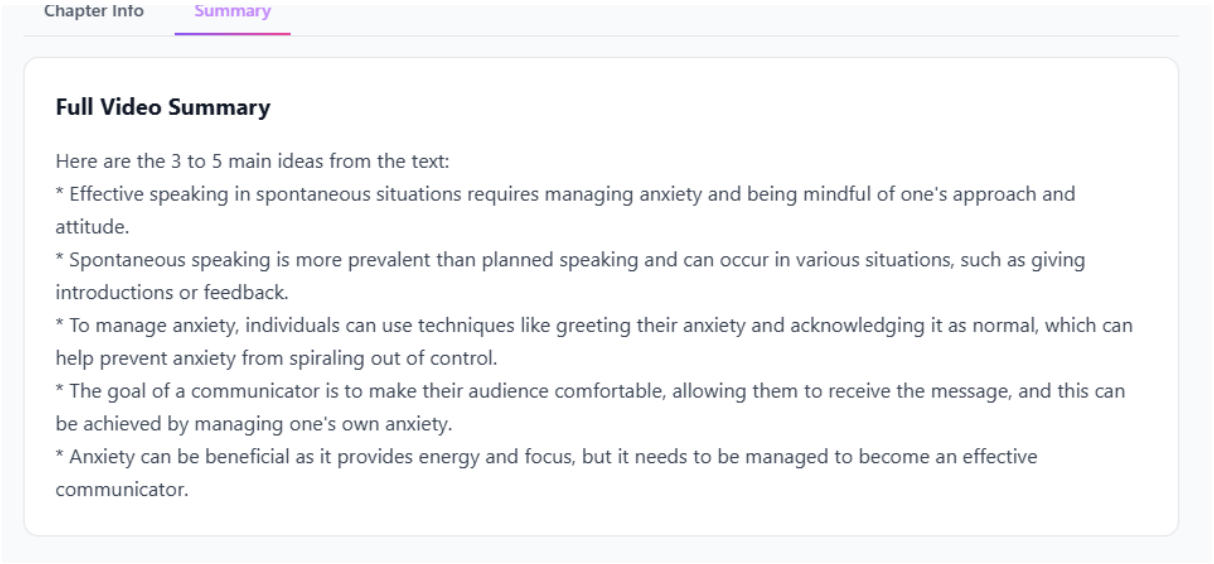

Figure 4.30: Summary Section

Chapter Information Panel

The chapter information panel displays detailed information about the selected chapter, including its title, timestamp range, and generated content.

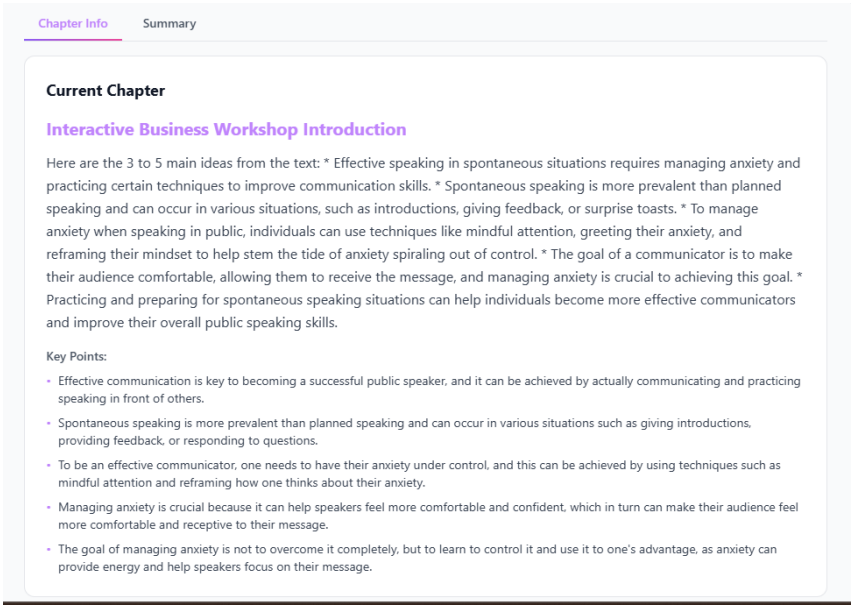

Figure 4.31: Chapter Information Panel

PDF Export Feature

The PDF export feature enables users to download the generated analysis results, including summaries, transcripts, and chapter information, in a portable document format for offline use and sharing.

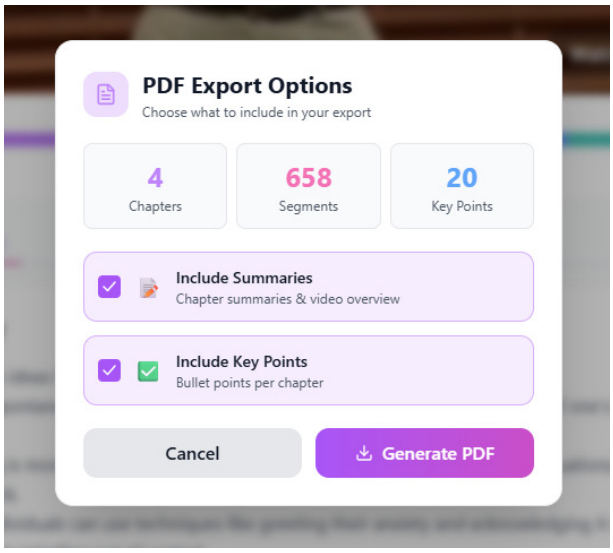

Figure 4.32: PDF Export Feature

Collections Management

The collections interface allows users to create topic-based groups of videos and share them with colleagues by email. Collections can be made public or kept private.

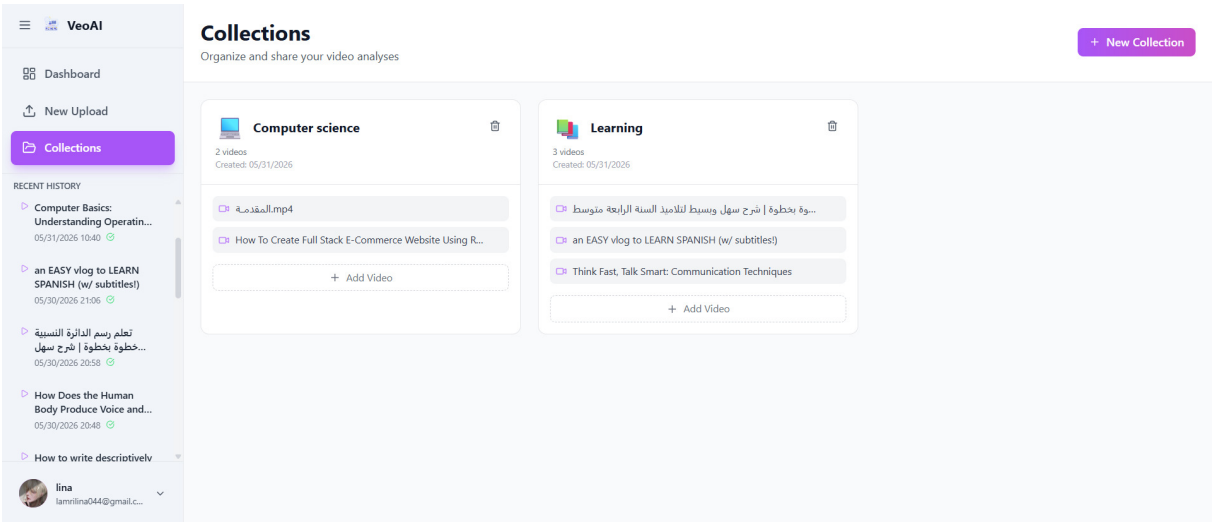

Figure 4.33: Collections Management Interface

Settings Interface

General Settings Interface

Figure 4.34 illustrates the General Settings tab, where users can customize application preferences such as language selection (English, French, or Arabic), visual theme (light, dark, or system mode), timezone, and date format settings.

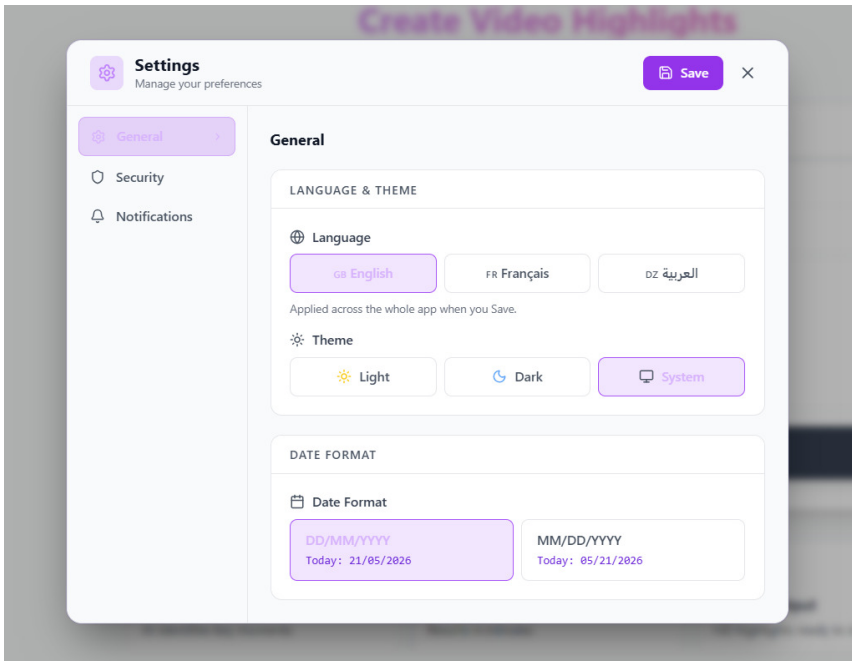

Figure 4.34: General Settings Interface

Security Settings Interface

Figure 4.35 presents the Security Settings tab, which allows users to change their password, manage active sessions across devices, revoke unauthorized sessions, and permanently delete their account if desired.

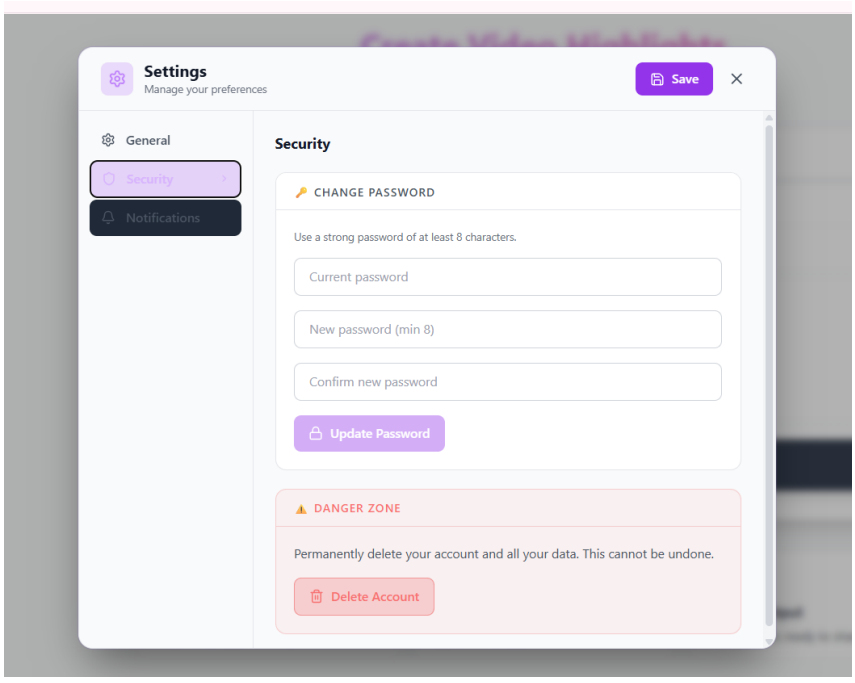

Figure 4.35: Security Settings Interface

Notification Settings Interface

Figure 4.36 shows the Notification Settings tab, where users can configure email notifications, processing completion alerts, and weekly digest preferences according to their needs.

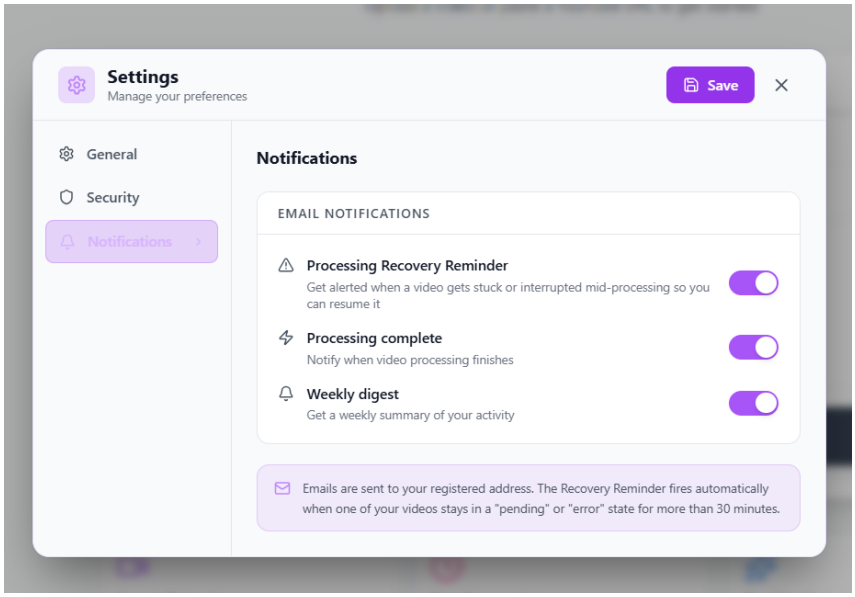

Figure 4.36: Notification Settings Interface

History Sidebar

The history sidebar provides chronological access to all previously analyzed videos. Users can click any item to instantly reload its results.

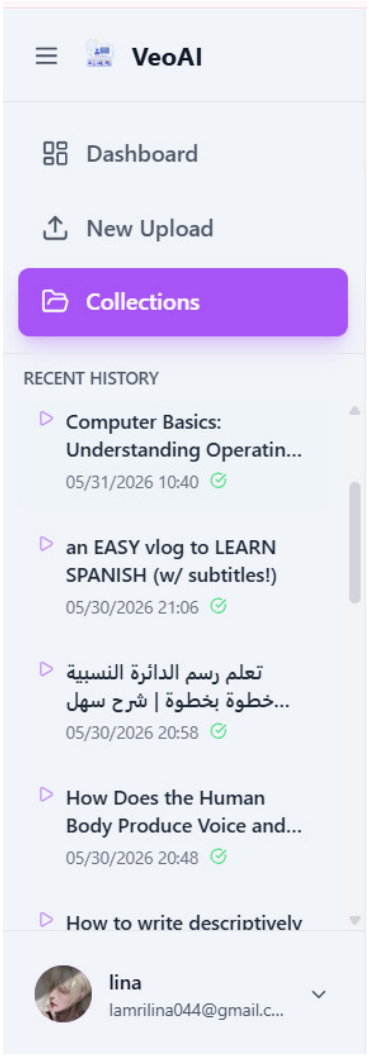

Figure 4.37: History Sidebar

Profile Interface

The profile interface allows users to view and manage their personal account information. Users can update their profile details, including their name, email address, and profile picture. This section provides a centralized location for maintaining account information and personal preferences.

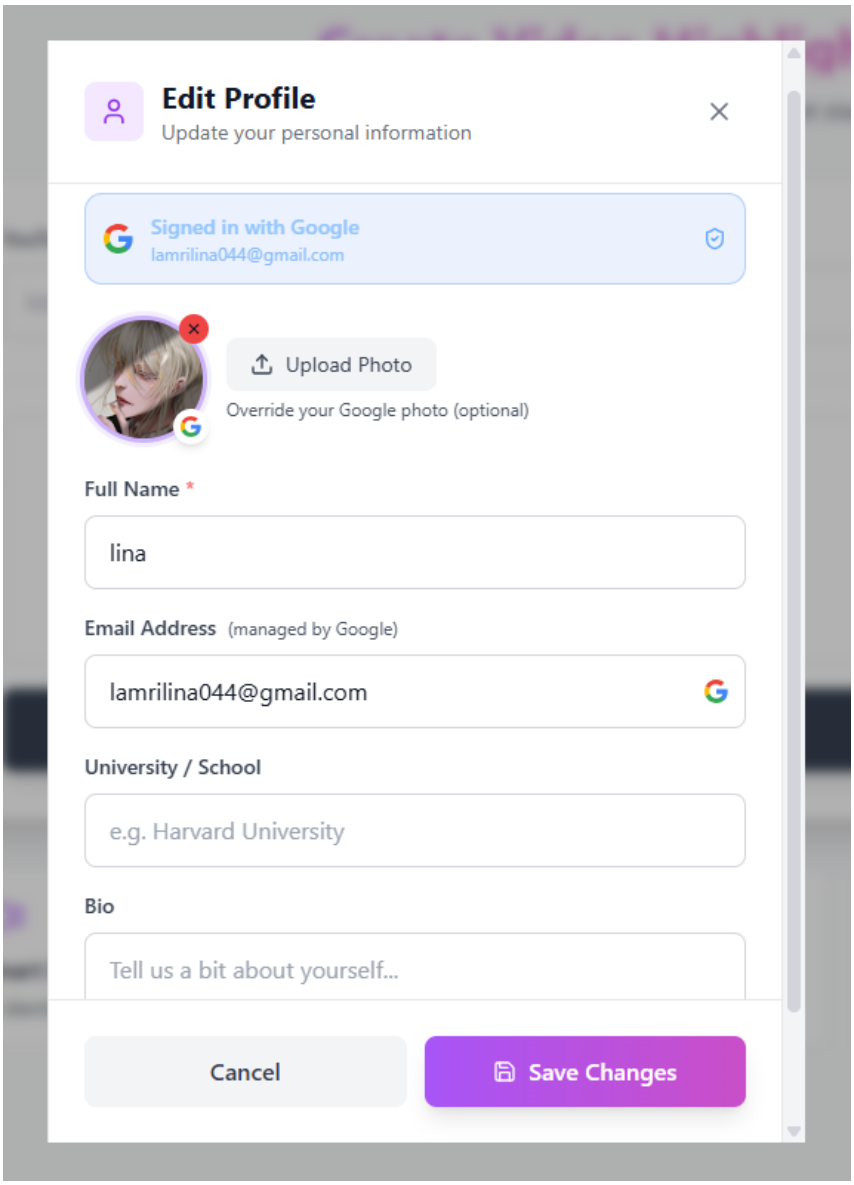

Figure 4.38: Profile Interface

4.3.2 Mobile Application Interfaces

The following figures present the main screens of the Android mobile application developed using React Native and Expo. The application provides core platform functionalities through a native mobile interface optimized for smaller screens.

Landing Page

Figure 4.39 presents the landing page of the application, which serves as the public entry point introducing the core features and directing visitors to register or sign in

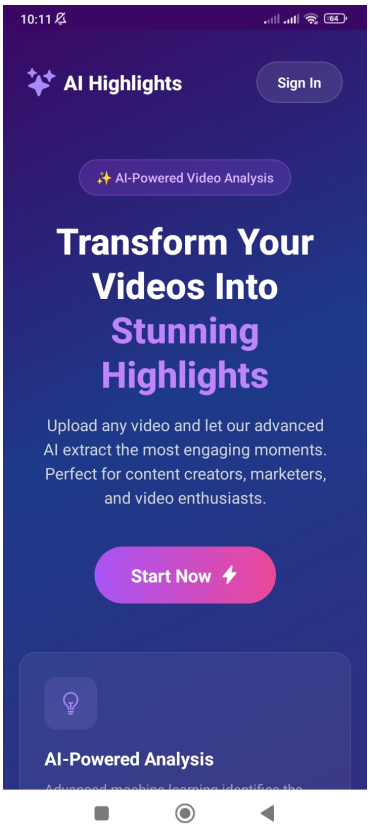

Figure 4.39: Mobile Application – landing page

Mobile Authentication

Figure 4.40 presents the mobile login and registration screens, allowing users to authenticate using their email credentials or Google OAuth directly from their Android device.

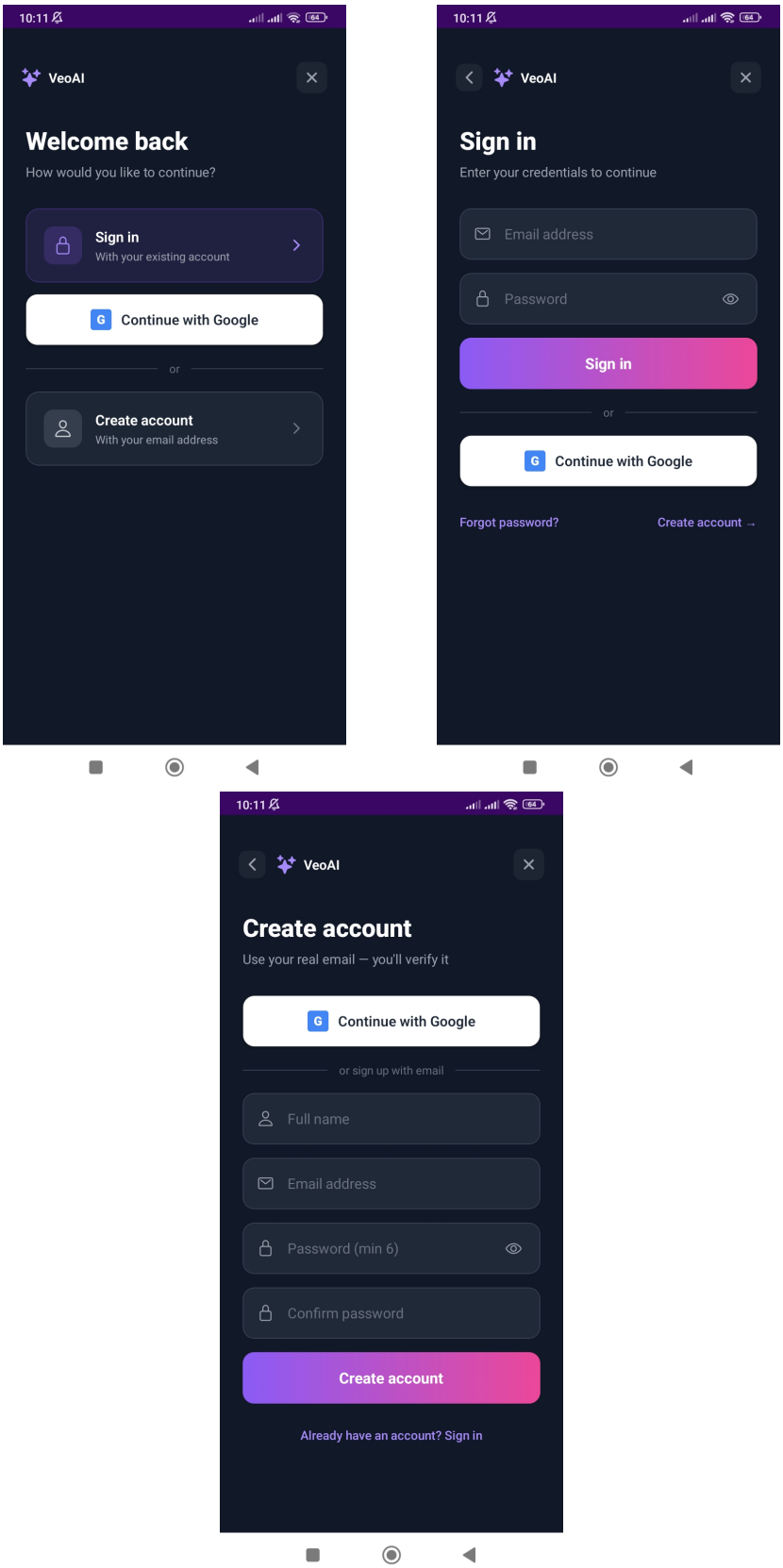

Figure 4.40: Mobile Application – Login and Registration Screens

Mobile Home and Video Submission

Figure 4.41 illustrates the mobile home screen and the video URL submission interface, enabling users to submit YouTube links for AI analysis directly from their smartphone.

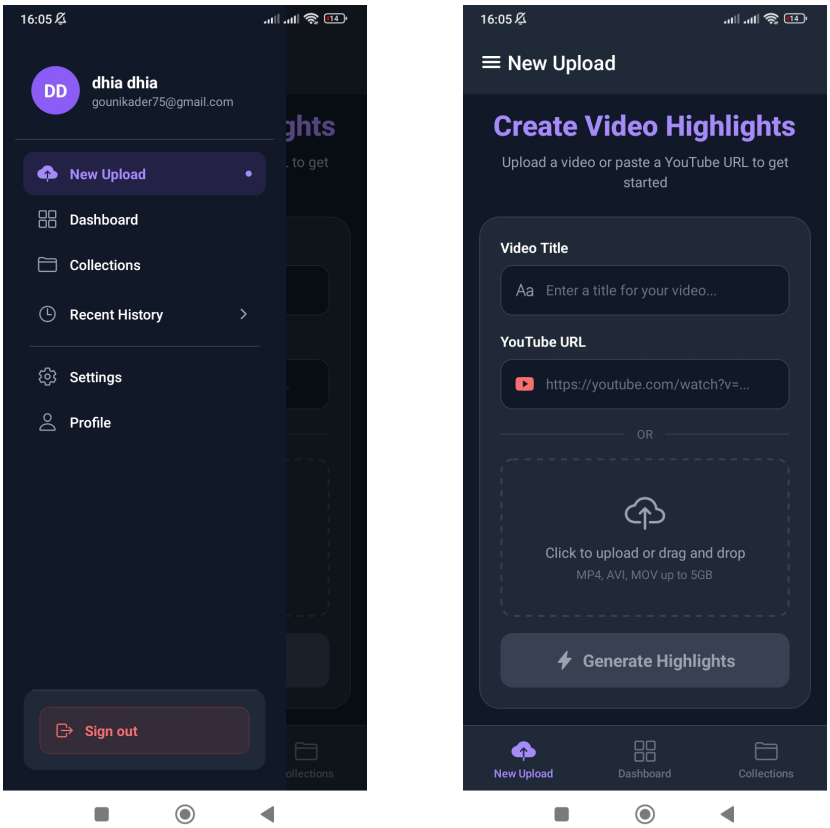

Figure 4.41: Mobile Application – Home and Video Submission

User Dashboard

Figure 4.42 presents The main dashboard displays all analyzed videos with thumbnails, processing status badges, chapter counts, and filtering and search capabilities.

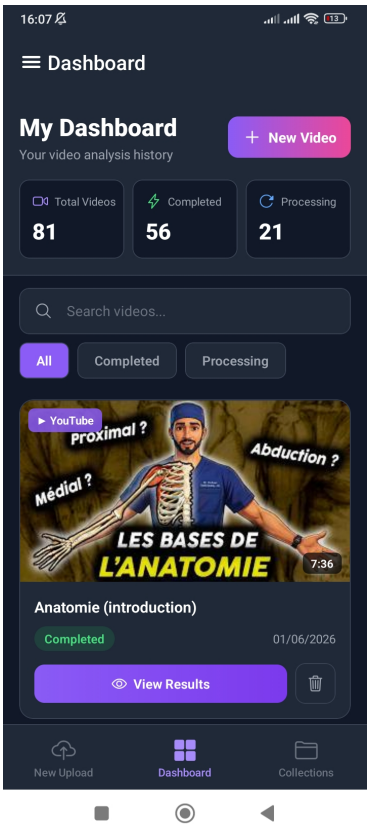

Figure 4.42: Mobile Application –Dashboard

Mobile Analysis Results

Figure 4.43 illustrates the analysis results for a standard YouTube video containing speech content. The system successfully transcribes the audio, generates structured chapters with timestamps, extracts key points, and produces an abstractive summary from the spoken content.

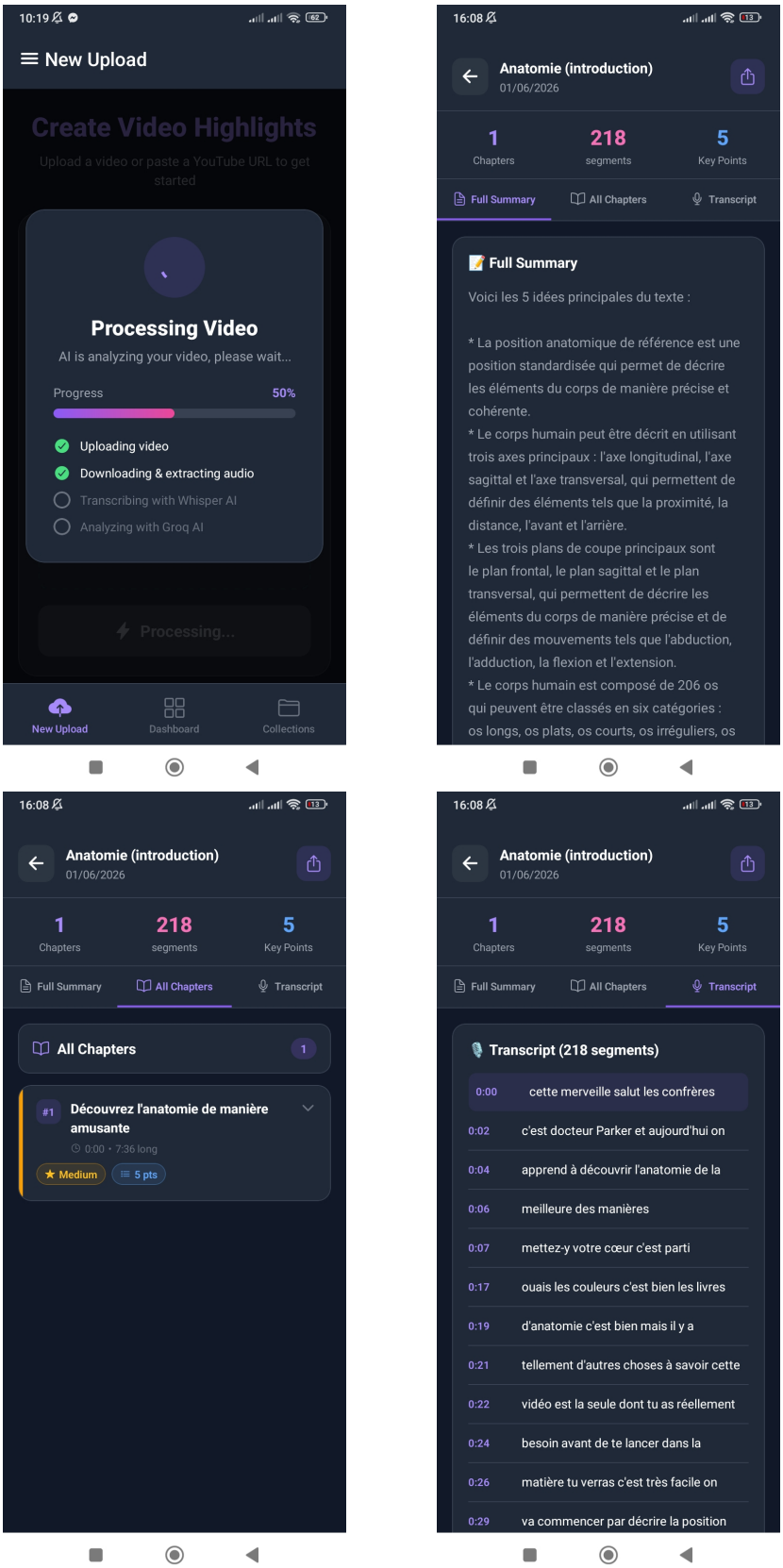

Figure 4.43: Mobile Application – Analysis Results and Chapters

Visual-Only Video Analysis

Figure 4.44 illustrates the system's behavior when processing videos without speech content. In such cases, the audio classification module automatically switches to the visual analysis pipeline, where Groq Vision analyzes the extracted frames and generates content based on the visual information available in the video.

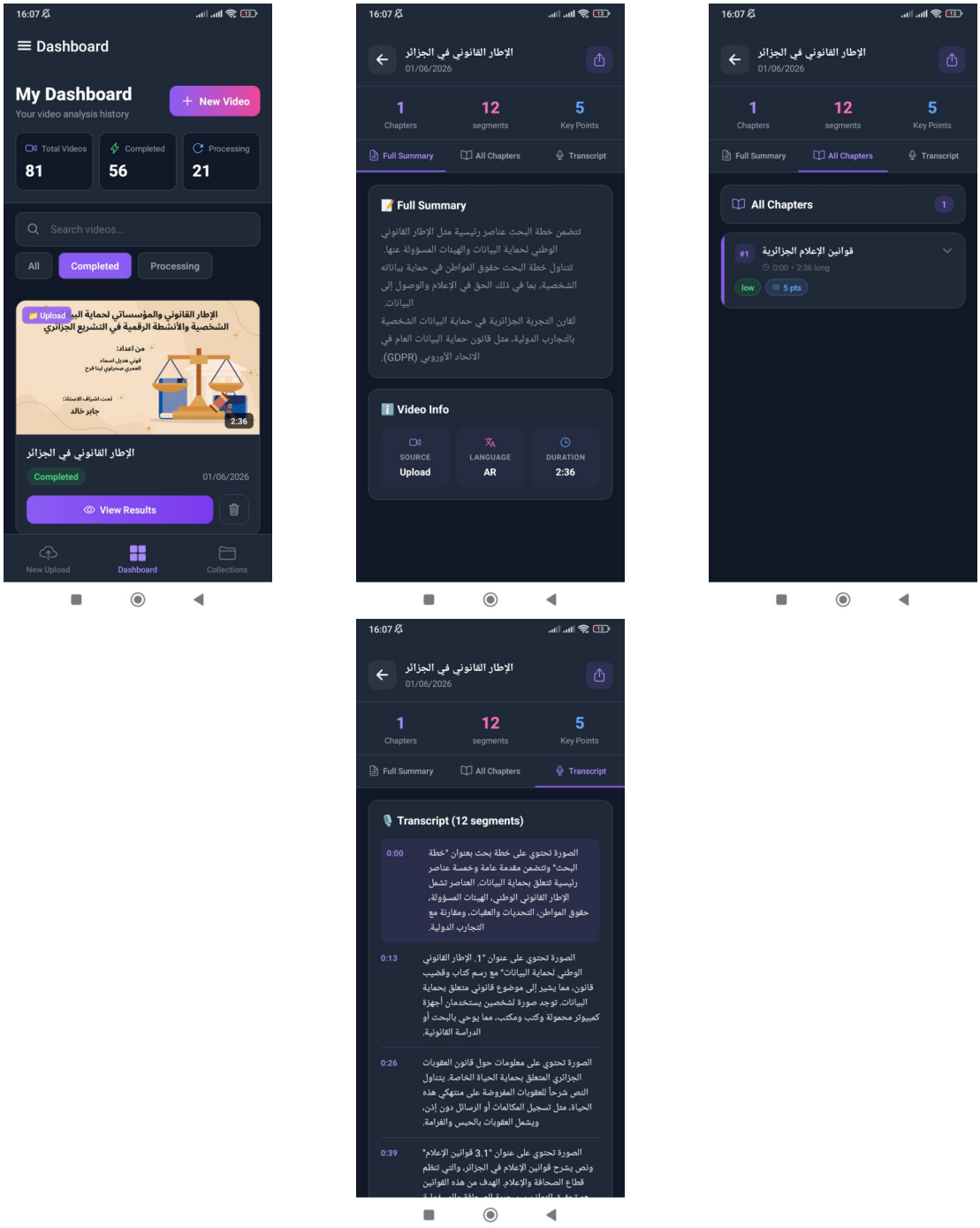

Figure 4.44: Mobile Application – Visual-Only Analysis: Frame Extraction and Groq Vision Processing

Collections, Settings and Profile Editing

Figure 4.45 llustrates the collection management interfaces, allowing users to create new collections and organize analyzed videos efficiently.
Figure 4.46presents the application settings screens, including general settings, security options, and profile editing functionalities that enable users to personalize and manage their accounts.

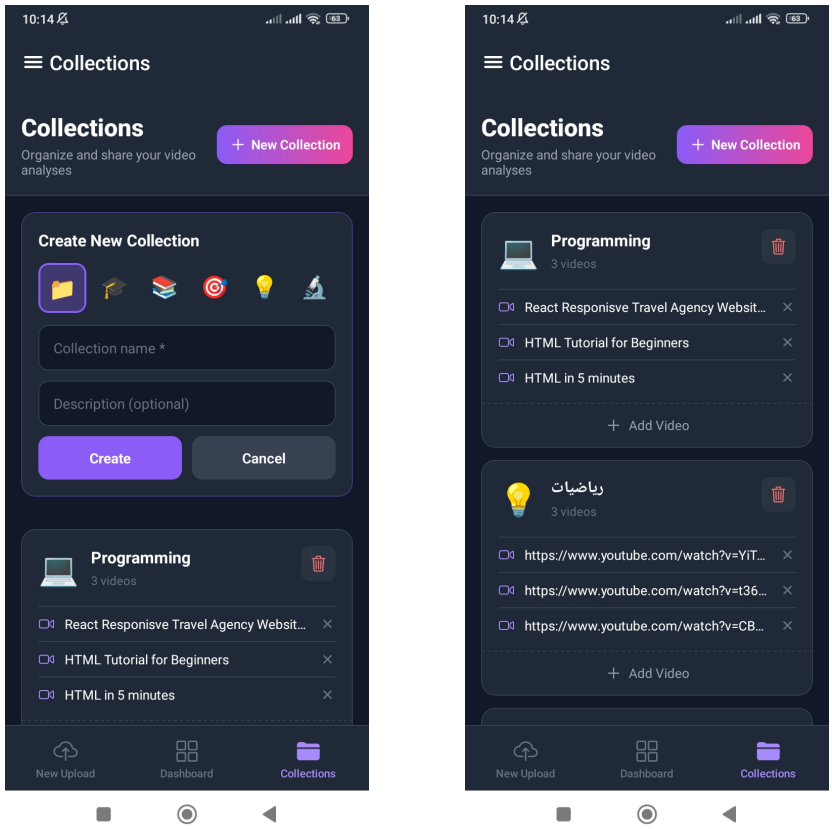

Figure 4.45: Mobile Application – Collections

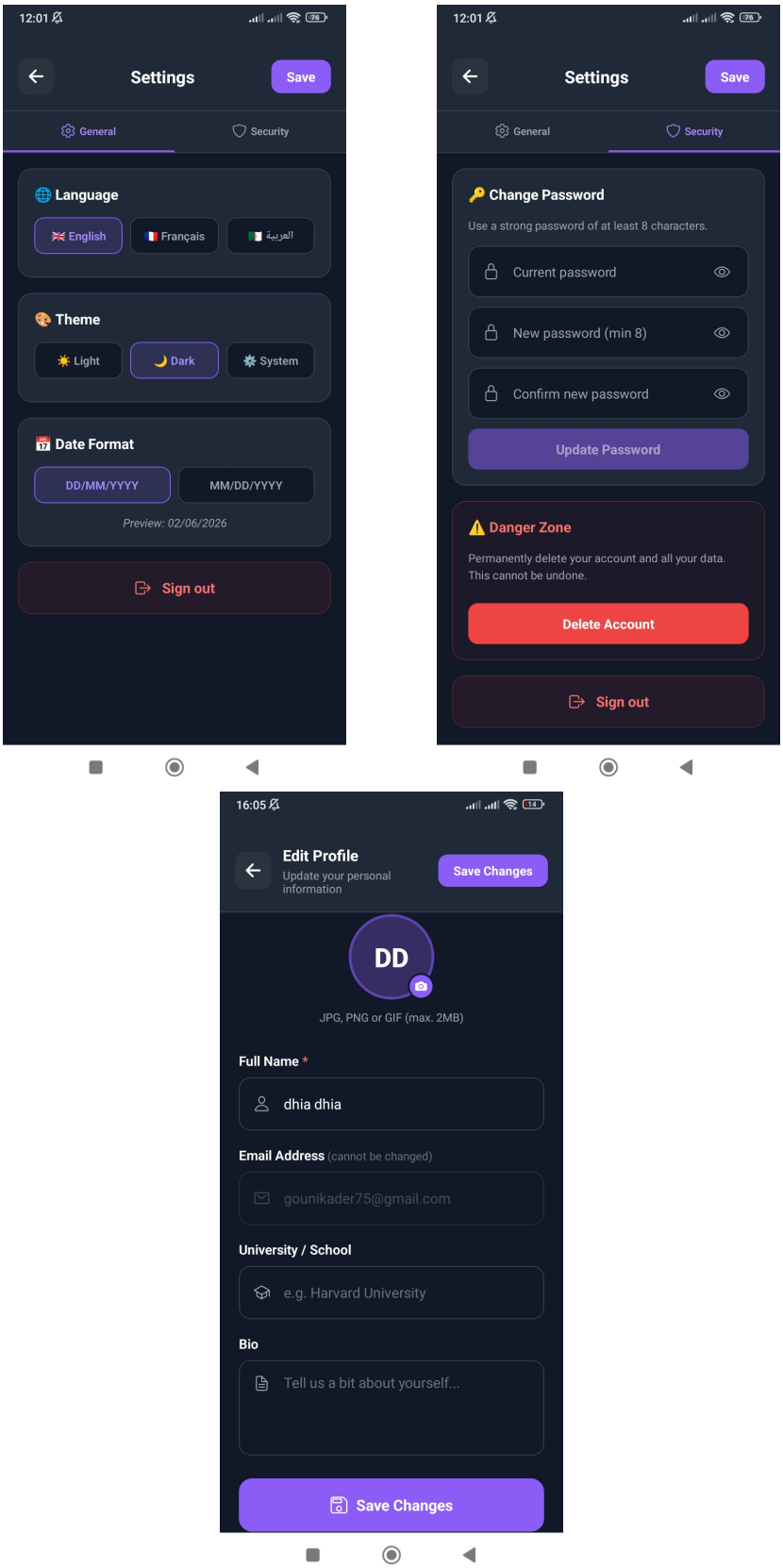

Figure 4.46: Mobile Application – Settings and edit profil

4.5 Conclusion

This chapter presented the complete implementation of the AI-Based Video Highlights Generator. The development environment was configured across three independent components using modern tools and frameworks. The system successfully delivers secure authentication with Google OAuth and email verification, flexible video input through file upload and YouTube URL processing, multimodal AI analysis combining Whisper and Groq, multilingual support across three languages with full RTL rendering, and a comprehensive user interface featuring dashboard, history sidebar, collections, profile, and settings management. The obtained results demonstrate the viability of the proposed approach and its readiness for final evaluation.

General Conclusion

This thesis presented the design and implementation of an AI-Based Video Highlights Generator for Recorded Sessions a full-stack web application that automatically analyzes video content and extracts key moments, structured summaries, and timestamped highlights using state-of-the-art artificial intelligence technologies.

The work was organized around four main chapters covering theoretical foundations, system requirements, architectural design, and practical implementation. The implemented system successfully achieves its core objectives: users can upload video files or submit YouTube URLs, receive AI-generated transcripts, chapters, and summaries, organize analyses into collections, and access processing history from any device. The platform supports English, French, and Arabic with full RTL support, offers Google OAuth authentication, and provides comprehensive user settings including theme, timezone, and session management.

The AI pipeline built around OpenAI Whisper and the Groq API adapts its processing strategy through an intelligent audio classification mechanism that handles speech, music, and silence content types. When speech is detected, Whisper produces a timestamped transcript passed to the Groq API for summarization and chapter generation. When visual content dominates, Groq Vision analyzes extracted frames. A fallback to extractive summarization ensures service continuity when the Groq API is unavailable.

Despite these achievements, processing speed on CPU hardware remains a bottleneck for long videos, and the system's dependence on the Groq API introduces an external availability constraint mitigated through exponential backoff mechanisms. These limitations point toward clear directions for future improvement.

Perspectives

Future improvements of the proposed system may include:

- **GPU acceleration:** Deploying the AI microservice on GPU-enabled hardware to significantly reduce Whisper transcription time for long videos and enable the use of larger, more accurate Whisper model variants such as `medium` or `large`.
- **Local LLM deployment:** Replacing the Groq API dependency with locally hosted open-source models such as Llama or Mistral, eliminating external availability constraints and further strengthening user data privacy by keeping all processing on-premise.
- **E-learning platform integration:** Connecting the system with platforms such as Moodle, Google Classroom, or Microsoft Teams to enable automatic highlight generation for recorded lectures within existing educational workflows.
- **Real-time processing:** Extending the pipeline to support live video streams and generate real-time subtitles, chapter markers, and key moment alerts during ongoing sessions.
- **Speaker diarization:** Integrating speaker identification capabilities to distinguish between multiple speakers in a recording, enabling per-speaker transcripts and targeted highlight extraction.
- **Offline mode:** Implementing a Progressive Web App (PWA) architecture to allow users to access previously analyzed videos and their summaries without an active internet connection.

Overall, this project demonstrates that combining modern web technologies with open-source AI models can produce effective, accessible, and privacy-respecting tools for intelligent video content analysis tools with real potential to improve how students, educators, and professionals interact with recorded video content.

Bibliography

[1] Russell, S., and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson Education.

[2] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

[3] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA. Available: http://www.deeplearningbook.org [Accessed: Mar. 5, 2026]

[4] Jurafsky, D., and Martin, J. H. (2023). *Speech and Language Processing*, 3rd ed. (draft). Available: https://web.stanford.edu/~jurafsky/slp3/ [Accessed: Mar. 5, 2026]

[5] Chowdhury, G. (2003). *Natural Language Processing*. Annual Review of Information Science and Technology, 37, 51–89.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*

[7] Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., and Roth, D. (2023). "Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey," *ACM Computing Surveys*, vol. 56, no. 2, Article 30, pp. 1–40.

[8] Gardazi, N. M., Daud, A., Malik, M. K., Bukhari, A., Alsahfi, T., and Alshemaimri, B. (2025). "BERT Applications in Natural Language Processing: A Review," *Artificial Intelligence Review*, vol. 58, Article 166. pp.2–49

[9] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proc. NAACL-HLT*, Available: https://arxiv.org/abs/1810.04805 [Accessed: Mar. 2026]

[10] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67.

[11] Groq Inc. (2024). *Groq API: Fast AI Inference*. Available: https://groq.com [Accessed: Apr. 2026]

[12] Meta AI (2024). *Llama: Open Foundation Models*. Available: https://llama.meta.com [Accessed: Apr. 2026]

[13] Choudhary, S., Kumar, S., Mallampatti, D., and Hegde, N. (2026). "NLP-Based Depression Severity Classification Using Fine-Tuned BERT and Groq API," *International Journal of Computing and Digital Systems*, University of Bahrain. Available:https://hdl.handle.net/20.500.14536/6787 [Accessed: Apr. 2026]

[14] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). "mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 483–498.

[15] Jia, P. (2010). *Automatic Speech Recognition*. Vision Open Working Group, First Edition. pp. 1–13

[16] Biehl, M. (2015). *API Architecture: The Big Picture for Building APIs*. API-University Press.

[17] Jin, B., Sahni, S., and Shevat, A. (2018). *Designing Web APIs: Building APIs That Developers Love*. O'Reilly Media.

[18] Lubanovic, B. (2023). *FastAPI: Modern Python Web Development*. O'Reilly Media.

[19] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97.

[20] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2023). "Robust Speech Recognition via Large-Scale Weak Supervision," Available: https://arxiv.org/abs/2212.04356 [Accessed: Mar. 2026]

[21] Amorese, T., Greco, C., Cuciniello, M., Milo, R., Sheveleva, O., and Glackin, N. (2023). "Automatic Speech Recognition (ASR) with Whisper: Testing Performances in Different Languages," *Proc. S3C Workshop, CHITALY 2023*, CEUR Workshop Proceedings, vol. 3574.

[22] Li, Z.-N., Drew, M. S., and Liu, J. (2021). *Fundamentals of Multimedia*, 3rd ed. Springer Nature Switzerland. ISBN: 978-3-030-62123-0.

[23] Hochreiter, S., and Schmidhuber, J. (1997). "Long Short-Term Memory," *Neural Computation*, Available: https://www.bioinf.jku.at/publications/older/2604.pdf [Accessed: Mar. 2026]

[24] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). "Large-Scale Video Classification with Convolutional Neural Networks," *Proc. IEEE CVPR*

[25] Gygli, M., Grabner, H., Riemenschneider, H., and Van Gool, L. (2014). "Creating Summaries from User Videos," *Proc. ECCV*, pp. 505–520.

[26] Zhang, K., Chao, W.-L., Sha, F., and Grauman, K. (2016). "Video Summarization with Long Short-term Memory," *Proc. ECCV*, pp. 766–782.

[27] Rochan, M., Ye, L., and Wang, Y. (2019). "Video Summarization Using Fully Convolutional Sequence Networks," *Proceedings of the European Conference on Computer Vision (ECCV)*

[28] Liu, Y., and Lapata, M. (2019). "Text Summarization with Pretrained Encoders," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*

[29] IJRASET (2023). "Summarization of Video using Audio," *International Journal for Research in Applied Science and Engineering Technology*, vol. 11.

[30] Zhou, K., Qiao, Y., and Xiang, T. (2018). "Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity-Representativeness Reward," *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[31] Barbakos, S., Antoniadis, C., Potamianos, G., and Setti, G. (2025). "Unsupervised Transcript-assisted Video Summarization and Highlight Detection," *arXiv preprint arXiv:2505.23268*.

[32] Fireflies.ai Inc. (2024). *AI Meeting Assistant Platform*. https://fireflies.ai [Accessed: Mar. 4, 2026]

[33] Otter.ai Inc. (2024). *AI Meeting Notes and Transcription*. https://otter.ai [Accessed: Mar. 4, 2026]

[34] Descript Inc. (2024). *Video Editing with Transcription*. https://descript.com [Accessed: Mar. 4, 2026]

[35] FFmpeg Team (2024). *FFmpeg Multimedia Framework*. https://ffmpeg.org [Accessed: Mar. 4, 2026]

[36] yt-dlp contributors (2024). *yt-dlp: A youtube-dl fork with additional features and fixes*. Available: https://github.com/yt-dlp/yt-dlp [Accessed: Apr. 2026]

[37] Microsoft (2024). *Visual Studio Code*. Available: https://code.visualstudio.com [Accessed: mai. 2026]

[38] Git SCM (2024). *Git: Distributed Version Control System*. Available: https://git-scm.com [Accessed: mai. 2026]

[39] Meta (2024). *React: A JavaScript Library for Building User Interfaces*. Available: https://react.dev [Accessed: mai. 2026]

[40] Tailwind Labs (2024). *Tailwind CSS: A Utility-First CSS Framework*. Available: https://tailwindcss.com [Accessed: mai. 2026]

[41] OpenJS Foundation (2024). *Node.js: JavaScript Runtime*. Available: https://nodejs.org [Accessed: Apr. 2026]

[42] OpenJS Foundation (2024). *Express.js: Fast, Unopinionated Web Framework*. Available: https://expressjs.com [Accessed: mai. 2026]

[43] Meta (2024). *React Native: Learn Once, Write Anywhere*. Available: https://reactnative.dev [Accessed: mai. 2026]

[44] Python Software Foundation (2024). *Python Programming Language*. Available: https://www.python.org [Accessed: mai. 2026]

[45] Tiangolo, S. (2024). *FastAPI: Modern Web Framework for Python*. Available: https://fastapi.tiangolo.com [Accessed: mai. 2026]

[46] MongoDB Inc. (2024). *MongoDB: The Developer Data Platform*. Available: https://www.mongodb.com [Accessed: mai. 2026]

[47] Postman Inc. (2024). *Postman: API Platform*. Available: https://www.postman.com [Accessed: mai. 2026]

[48] Render (2024). *Render Cloud Application Hosting Platform*. Available: https://render.com [Accessed: May 2026]

[49] Railway (2024). *Railway: Deployment Platform for Applications and Databases*. Available: https://railway.com [Accessed: May 2026]