Snap-Seek: Turning Questions Into Timelines

TEHREEM BABAR
01-136221-030
ASMA BABAR
01-136221-005

Supervisor: DR. MUHAMMAD ASFAND-E-YAR

Bachelor of Science in Artifical Intelligence

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "Snap-Seek: Turning Questions into Timelines", written by Tehreem Babar AND Asma Babar as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: DR. MUHAMMAD ASFAND-E-YAR

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Sohail Akhtar

_____

Head of the Department: Dr. Khurram Ehsan

_____

Abstract

Snap-Seek is an AI-powered video assistant designed to help users navigate and extract information from video content efficiently. Our project integrates state-of-the-art artificial intelligence models for speech recognition, natural language processing, and information retrieval to enable features such as automatic transcription of video audio, abstractive summarization of transcripts, and an interactive question-answering system for video content. The assistant processes a given video (from a file or YouTube link) by first transcribing its audio using a robust speech recognition model (OpenAI's Whisper), then generating a concise summary of the video using a pre-trained abstractive summarization model (Facebook's BART). It further allows users to ask questions about the video; the system finds relevant segments in the transcript and provides answers using a transformer-based QA model, supplemented by a semantic search mechanism and summary for context. Snap-Seek supports multilingual queries by translating questions and answers when the video or query language is non-English, leveraging cross-lingual transformer models and translation APIs.

Our thesis presents the design, implementation, and evaluation of Snap-Seek. We discuss the underlying algorithms and models, system architecture, and experimental results demonstrating the system's ability to summarize videos and answer detailed questions about their content accurately. The outcome of our project is an interactive AI assistant that enhances video accessibility and knowledge extraction, showcasing the application of modern AI techniques in multimedia data analysis.

Acknowledgments

To begin with, we thank Allah Almighty for the numerous blessings, guidance and mercy He has bestowed upon us in order for us to be able to fulfil this project successfully. This project could not have been completed without His blessings and support. We would also like to thank our supervisor, Dr. Muhammad Asfand-e-Yar, for all the invaluable assistance in the form of his guidance, encouragement and support throughout the duration of this project. Through his vast expertise and insight he has played an important part in both shaping our project and assisting with its execution. A special thanks goes to the Department of Computer Science at Bahria University Islamabad for the resources and support that made it possible for us to carry out this research project. Furthermore, we would like to extend our deepest appreciation to our families and friends for their continued support, patience and inspiration during the more difficult moments of our project. Finally, we are extremely grateful to all of the researchers and practitioners in the areas of Artificial Intelligence, Natural Language Processing and Video Analysis whose research work has contributed to the development of our project through their innovative methods, published papers and open-source materials. They have provided us with the necessary tools to gain a thorough understanding of and apply the concepts of AI-based video analysis.

TEHREEM BABAR AND ASMA BABAR
Bahria University
E-8, Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Abbreviation	Description
AI	Artificial Intelligence
ASR	Automatic Speech Recognition
NLP	Natural Language Processing
QA	Question Answering
QG	Question Generation
GPU	Graphics Processing Unit
CLI	Command-Line Interface
FFmpeg	FFmpeg (Fast-Forward MPEG), an open-source audio/video processing tool
yt-dlp	yt-dlp, a YouTube video downloading library (fork of youtube-dl)
Whisper	Whisper, OpenAI's speech recognition model
BART	BART, Bidirectional and Auto-Regressive Transformer (text generation model by Facebook)
XLM-R	XLM-RoBERTa, a cross-lingual language model based on RoBERTa
T5	T5, Text-to-Text Transfer Transformer by Google

Chapter 1

Introduction

1.1 Project Background

In recent years, digital platforms such as YouTube, Coursera, Udemy, and other streaming services have experienced exponential growth in the amount of uploaded video content. Educational lectures, tutorials, interviews, documentaries, podcasts, and meeting recordings are now widely available online. Although this abundance of information is beneficial, it also presents a significant challenge: manually consuming long-form video content requires a substantial amount of time and attention. Due to limitations in their ability to watch hours of video just to find key content, clarify a specific topic, or review valuable content, the majority of users have difficulty using long-form, unedited video as a primary means of obtaining information. Advances in Artificial Intelligence (AI) in the fields of speech recognition and Natural Language Processing (NLP) have created new ways for users to engage and interact with video content in previously unimaginable ways. The use of modern ASR models like Whisper [1] makes it possible to accurately transcribe audio files (video and/or audio files) in multiple languages and in high-noise environments (e.g., conferences) using automated means. Similarly, transformer-based models, such as BART [2] and T5 [3], have allowed high-quality summaries and automated question answering to be produced with minimal human intervention. Together, these technologies have made it feasible to automate the process of extracting meaningful content from videos through machines (not humans).

Our project, **Snap-Seek**, combines the technologies used in ASR and transformer-based summarizers to create an intelligent virtual assistant that automatically transcribes video/audio content, auto-generates summaries, and provides an interactive mechanism for users to ask questions. The goal of Snap-Seek is to create an efficient, interactive experience for users who wish to retrieve knowledge from videos rather than consume the passive nature of watching a video.

1.2 Problem Statement

Video content is continuing to grow rapidly online; therefore, the number of users relying upon manual methods to obtain data has increased dramatically. The act of watching a 1-hour video just to retrieve key content, clarify a topic, or review important content is inefficient and time-consuming. Existing tools (auto-generated subtitles, keyword search, and chapter markers) offer little assistance and most often fail to locate the specific piece of information that users are looking for.

To date, there is no online platform providing users with a method of asking natural-language questions about video content and receiving answers that are both immediately relevant and contextually accurate. For instance, users currently have no way to ask **"What are the main findings of this video?"** or **"What does the speaker say about these concepts?"** and receive accurate, direct answers based on the material in the video. As such, an absence of an automated transcript-based summarization system that enables interactive questions and answers regarding video content leaves this gap.

1.3 Problem Description

To illustrate the challenge of information retrieval from long videos, consider a student revising a lecture on YouTube. The student may only need clarification on a specific concept discussed midway through the lecture; however, without intelligent assistance, they are forced to scrub through the timeline or rewatch large portions of the video. This process is time-consuming, inefficient, and often frustrating, particularly when deadlines are approaching or when the required information is brief but buried within lengthy content.

Similarly, professionals frequently rely on recorded meetings, webinars, or training sessions for reference at a later stage. Locating a specific decision, explanation, or key discussion point from such recordings becomes impractical through manual navigation alone. The lack of structured indexing or searchable content significantly reduces the usefulness of these recordings over time.

An effective solution to these challenges requires the automatic transformation of unstructured video data into meaningful, searchable information. Such a system should transcribe spoken content accurately, support fast text-based searching, generate concise summaries of long videos, and enable users to ask questions in natural language. Additionally, the ability to jump directly to relevant video segments based on user queries would greatly enhance efficiency. Together, these capabilities form the foundation of an interactive, AI-driven system that bridges the gap between raw video content and accessible, actionable knowledge.

As illustrated in Figure 1.1, the capabilities required to effectively utilize these technologies are clear. The ideal system will: Automatically create written transcripts from

speech. Enable users to search an entire transcript in real time. Provide users with a brief summary of the transcript for a quick overview that summarizes contents. Have the ability for users to ask natural-language questions about the contents of the transcript. Allow users to jump to the exact location of a video where relevant content appears. The technologies necessary to meet these requirements create a need for an interactive AI-driven solution to connect unstructured video data with the acquisition of knowledge and information.

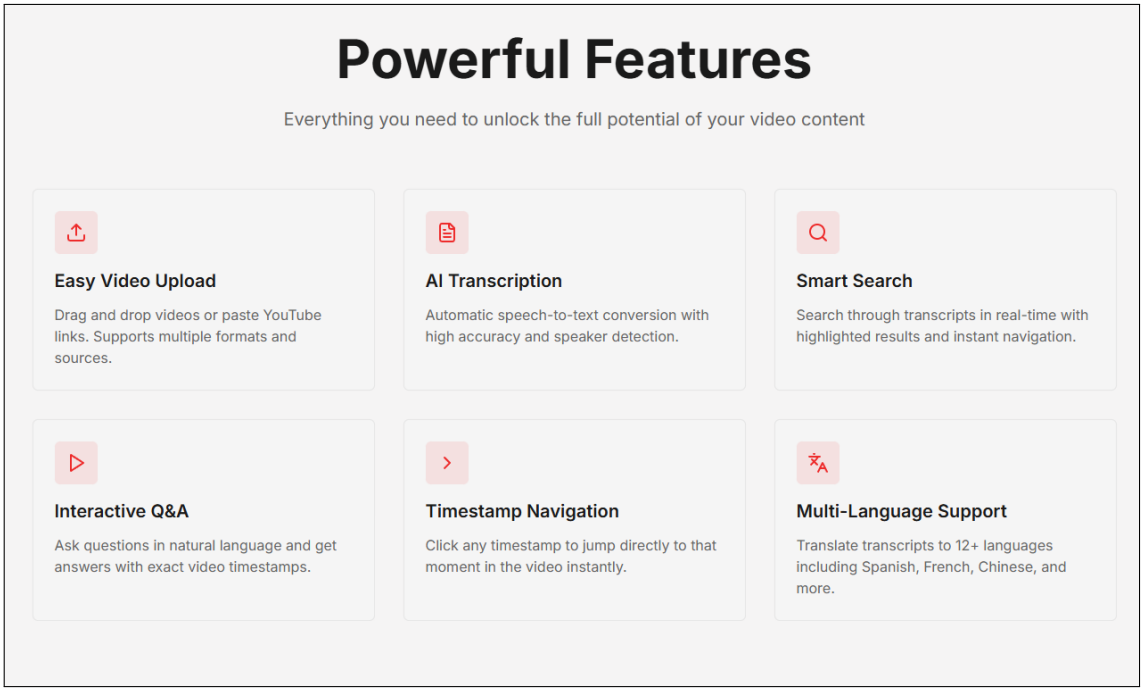

Figure 1.1: Snap-Seek's powerful features.

In addition to this necessity, people who attend online meetings may also need to quickly reference important decisions, discussions, or points made during the support. However, it is impractical for the user to try to locate specific points of interest in hours of recorded meetings using only their eyes. An ideal interactive AI-driven solution would allow:

1. The accurate transcription of spoken content to written content.
2. The summarization of the transcription into a concise, easy-to-read format.
3. The ability to respond to questions written in natural language using the transcript.
4. The ability to input and output in multiple languages.
5. The ability to operate offline to maintain users' privacy and reliability.

All these user-oriented requirements create a need for Snap-Seek, an interactive AI solution that will bridge the gap between unstructured video data and accessible knowledge.

1.4 Project Objectives

The objectives of Snap-Seek are to bridge the gap between unstructured video data and accessible knowledge by employing advanced AI technology to convert passive video content into an interactive knowledge experience. It will help the user with a quick grasp of the central ideas without having to see the whole content. Moreover, Snap-Seek makes learning faster, wiser, and more interactive. Instead of getting lost in what's said in long clips, users can search and browse and understand with ease. Snap-Seek supports learners at school or while researching and learners in daily activities by offering up insight from just any video. Save time and increase understanding among all, and reach further with information that will make it easier to grasp. Eventually, Snap-Seek will turn videos into yet another commodity we just see as a medium that has something to learn from. It encourages curiosity because users can dive deep into what matters to them. Complex information would suddenly present itself clearer and easier to understand with Snap-Seek. It sees a change in the interaction with the video over the long term, as depicted in Figure 1.2. To achieve this goal, Snap-Seek combines functions of offline speech recognition, summarization, question generation, and question answering to allow the user to easily query video content. Concretely, the project aims to:

1. **Automatic Transcription:** Utilizing advanced ASR technologies (e.g., Whisper) [1] to create highly accurate text-based transcripts from audio recordings.

2. **Content Summarization:** Generating short abstracts from long-form video content utilizing transformer-based summarizers (e.g., BART [2]).

3. **Question Answering:** Allowing users to enter natural language text-based queries and obtain their corresponding answers using question-and-answer systems (e.g., using XLM-RoBERTa) [4].

4. **Question Generation (Optional):** Identifying key concepts and suggesting possible questions through the use of T5-like systems. [3].

5. **Multilingual Support:** Incorporating abilities for automatic language identification [5] as well as translation [6] to facilitate content delivery to international audiences through multilingual video streams.

6. **User-Friendly Interface:** Design a straightforward means for users to upload videos, view the related summaries, and submit questions to the video.

7. **Offline Functionality:** Support an environment in which users may utilize the software without needing to have access to the internet for local model loading and processing.

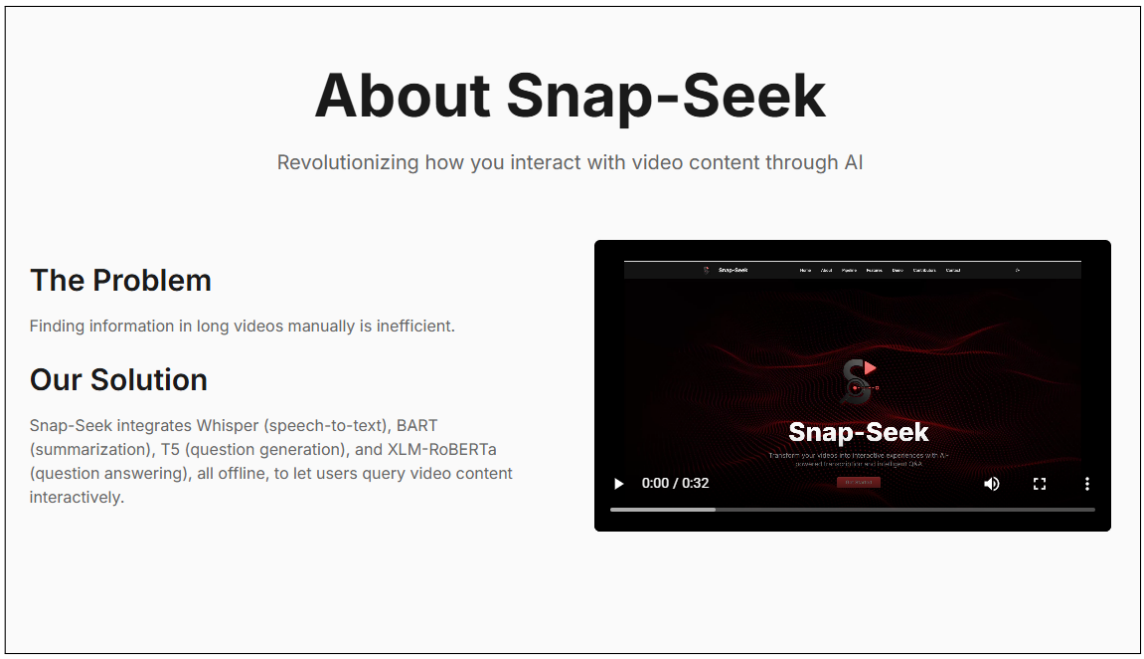

Figure 1.2: Overview of Snap-Seek.

1.5 Project Scope

Included in Scope

The goal of the project is to build a video-to-text information retrieval system that can derive useful content from videos that contain spoken material. The system's Main components will include:

1. **Processing of video/audio containing spoken content:** In addition to supporting user uploaded video files, videos hosted on YouTube will also be processed by pulling the audio from the video to facilitate further analysis.

2. **Offline ASR transcription using Whisper:** Offline ASR transcription functionality through the use of Whisper will allow users to perform transcriptions within an isolated environment, thereby avoiding reliance on third-party cloud services while ensuring accurate transcriptions in multiple languages.

3. **Transcript-based summarization and semantic retrieval:** The system will enable users to summarize and search transcripts produced through the use of custom-developed methods for text cleaning and summarization.

4. **Question answering using multilingual transformer models:** Queries are answered by finding relevant transcript fragments, and then obtaining the correct answer via question-answering models.

5. **Optional question generation:** The software has the ability to create potential questions based on the transcript, enhancing interactivity and learning-based applications.
6. **Prototype CLI/GUI interface for user interaction:** A command-line or graphical interface can be established to allow users to upload videos, receive transcriptions and summaries, and conduct QA actions against the transcriptions.

Out of Scope

Several advanced and resource-consumptive features will not be included in this project to help maintain feasibility based on the limited amount of time and hardware available:

1. **Visual analysis of video frames (object detection, slides, OCR):** The system will not perform visual analysis on video content. That is, the creation of object detection systems, the OCR of slide presentations, or scene classification systems, use of visual-based methodology will not occur.
2. **Real-time or live-stream processing:** Due to the computational capabilities of the system, it is not possible to conduct flood-like operations in realtime.
3. **Cloud-based APIs or server-side deployment:** This project does not incorporate cloud-based services in the ASR, summarization, or question answering aspects of the system; this implementation will strictly be implemented locally.
4. **Training or fine-tuning large language models from scratch:** The current project will not create or perform fine-tuning of large transformer models; instead, pre-trained transformer models will be used because of the resource-heavy aspect of producing and obtaining datasets needed.

1.6 Summary

Chapter 1 provides an overview of the rationale for creating Snap-Seek, a system that utilizes transcripts to identify content in videos. The chapter discusses the difficulties associated with searching large amounts of information within a single video and explains the goals of the project and its intended audience. Snap-Seek will use the latest advances in ASR and NLP technologies such as Whisper [1], BART [2], Sentence-BERT [7], and XLM-R [4] to give people the ability to find specific pieces of information in large amounts of video. The next chapter will provide a survey of the available literature on this subject and present the research that supports this study.

Chapter 2

Literature Review

2.1 Overview

Since the number of resources, such as YouTube, Coursera, and other online archives, is growing exponentially, there has been much interest in automated systems for video analysis. Most of the early research concentrated on using visual cues to summarize videos, while the most recent research has included using spoken video transcripts with techniques for natural language understanding and advanced deep learning techniques.

The development of transformer-based language models is one factor that has dramatically changed video summarization and QA systems. Compared with traditional recurrent or convolutional networks, these transformer-based language models can perform more sophisticated semantic reasoning, support multi-language operations, and have greater resilience to model errors. Chapter 2 will provide a comprehensive review of the current literature on automatic video transcription, transcript-based video summarization, semantic retrieval, and video QA.Moreover, some of the most recent work has emphasized the potential of these models for better understanding of context on longer video clips. These models are capable of making better linkages between visuals and text. In this context, some of the most recent innovations are leading towards various opportunities for the development of interactive and knowledge-driven tools on the basis of video. We've identified four major areas requiring attention:

1. Automatic Speech Recognition (ASR) technologies enabling accurate transcript generation.
2. Transcript-based summarization approaches for long-form content.
3. Natural-language question-answering techniques over video transcripts.
4. Related systems and architectures integrating retrieval, summarization, and QA.

2.2 Related Work

Automatic Speech Recognition

In the past, ASR systems primarily made use of Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM); these techniques did not work very well in noisy conditions, with multi-accented speech, or with multilingual speech. As a result of the development of new technologies like the OpenAI Whisper [1] (a transformer-based model), ASR has now changed significantly. OpenAI's Whisper network was trained with hundreds of thousands of hours of multilingual audio, and it delivers near state-of-the-art transcription performance, even in very noisy or diverse environments. The Whisper network can provide both the original audio transcript and an English translation, making it ideal for analyzing multilingual video recordings.

Transcript-Based Summarization

Current models that perform extractive summarization are token-based; they tokenize the transcript and then use statistical or graph-based methods to identify the most significant sentences. In contrast, current transformer-based abstract summarization models, such as BART [2], have shown substantial improvements in their ability to create human-like summaries of written material. These models are built using denoising autoencoders and sequence-to-sequence transformer models, allowing BART to read and interpret the textual data more effectively when asked to produce a summary (abstract summation). Similarly, the T5 [3] has influenced the design of many modern abstract summation systems due to its unified text-to-text construction method. T5 standardizes all tasks as text generation (e.g., translation, summarization, and classification), allowing users to fine-tune for many tasks using one single model.

Sentence Embeddings and Semantic Retrieval

For the video question answering (QA) systems, the retrieval of relevant segments of transcripts must be accomplished with speed. Sentence-BERT (SBERT) [7], which uses semantic architecture to generate semantically meaningful sentence embeddings, is optimized for cosine similarity searches. MiniLM is one variant of SBERT that reduces the computational overhead while preserving the accuracy of the embeddings.

Hugging Face Transformers [8] provides the models associated with SBERT and its variants in such a way that anyone can develop pipelines to efficiently retrieve segments based on semantic similarity rather than connecting words to perform the same function.

Generative QA: Generative Question and Answering (QA) solves the QA need by user-generating the answer in natural language (e.g., a human-written answer) and is done by using an autoregressive model of a type such as **T5, BART, GPT variants,** or **LLaMA**.

The model builds answers that are fluently written and that match what users want to know, depending on the transcript segments retrieved before. Generative QA is advantageous when:

1. The question cannot be answered by using a short answer but requires summarization or requires the use of reasoning to synthesize material from multiple transcript pieces.
2. The answer is not explicitly stated in one span.
3. The user is likely to ask a natural-conversational question and expect a similar answer.

However, generative systems are subject to **hallucination**, especially in long transcripts or in situations where the transcript retrieval is weak. For these reasons, generative systems are most often paired with grounding techniques (e.g., Retrieval-Augmented Generation [RAG] or constrained decoding methods).

Integrated Systems and Prior Architectures

Recent work has used Automatic Speech Recognition (ASR), retrieval techniques, and the question-answering capabilities of Question-Answering (QA) to create automated systems that can decode video data. Taskiran et al. created methods of summarizing from an early transcript using ASR output that are highly dependent on "noisy" ASR and statistical models, thus limiting the accuracy and reliability of their summarization results. More recently, Bhagwat et al. (2025) used CNN-LSTM hybrid architecture models for visual-text summarization, demonstrating a need to utilize the multimodal fusion of these media to understand complex video information. Similarly, Abiola (2024) developed a VideoQAbased on a retrospective-based approach to creating generative responses using transcripts created by Whisper, along with an LLM to provide generative answers. In all of these instances, deployments in cloud environments will still fall short of offering true offline capabilities.

The Snap-Seek system utilizes an offline-first approach through the integration of contemporary "state-of-the-art" language processing (NLP) technologies; however, it maintains a fully open-source model that can be executed from anywhere in the world with an internet connection.

2.3 Comparison with Previous Work

Many previous systems provided a solid foundation; however, there are still many gaps that remain.

1. **Dependence on Cloud APIs:** Generally, many QA systems are powered by third-party services such as GPT-based models. As these services are often associated with

privacy concerns, reliance on them to access internet connectivity and the associated costs. By using locally cached models, Snap-Seek has eliminated the reliance on cloud services.

2. **Limited Extractive Grounding:** Most generative QA systems produce fluent responses that provide no grounding within the content of the speech. Snap-Seek generative QA leverages extractive sub-systems for QA (e.g., answers) to ensure an answer remains true to the spoken content of the speech that was utilized to generate it.

3. **Weak Multilingual Support in Older Systems:** Before the development of WHISPER, ASR systems struggled (or were unable) to provide quality transcriptions or summaries of audio recordings containing open vocabulary. Multi-language support via WHISPER has dramatically improved the reliability of transcribing and generating summaries from all available transcription types.

4. **Scalability Issues:** Prior summarization methods could only handle shorter-form transcriptions and could not be used for summarizations of transcriptions containing thousands of words or longer. Snap-Seek uses hierarchical summarization to efficiently manage multi-thousand-token transcript summaries that can be consumed over an extended period.

5. **No Interactive Workflow:** Virtually all existing summarizers will produce summaries; however, there are no subscribers that support the summarization workflow to allow users to consume more than just the summaries while providing users the ability to actively engage with the video content they have viewed via the use of questions and answers.

Table 2.1 summarizes differences between Snap-Seek and previous approaches.

Table 2.1: Comparison with Previous Work

Feature	**Previous Work**	**Snap-Seek (Proposed)**
ASR Accuracy	Older HMM/DNN-based ASR; moderate accuracy	Whisper ASR with high multilingual accuracy
Summarization	Extractive; limited coherence	Abstractive BART with hierarchical summarization
QA Method	Keyword search or simple extraction	Semantic retrieval + transformer QA
Multilingual Support	Weak or absent	Strong, via XLM-R and Whisper translation
Offline Capability	Rare; cloud-dependent	Fully offline with cached models
Interactivity	Summaries only	Summary + QA + question generation

2.4 Summary

The prior research and technologies reviewed in this chapter serve as the foundation for SnapSeek's design, and the advancements in Automatic Speech Recognition ASR (Whisper [1]), abstractive summarization (BART [2]), text-to-text transformers (T5 [3]), semantic embeddings (SBERT [7]), and multilingual extractive QA (XLM-R [4]), have made it possible for SnapSeek to offer users the ability to understand transcript-based video or audio recordings with high quality.

Chapter 3

Requirements Specification

3.1 Existing System

Before Snap-Seek's inception, individuals utilized traditional methods for discovering relevant data during the consumption of audiovisual material (e.g., video or audio files) by applying outdated manual approaches that were labor-intensive and very time-consuming. For instance, an individual might watch an entire video recording of a lecture, interview, podcast, tutorial, or project meeting to find a piece of content that interests them. Additionally, even if YouTube offers timestamps, autogenerated captions, or chapter markers as a way to help people navigate through videos without watching the entire thing, those services may frequently be inconsistent, unreliable, and not necessarily provide sufficient detail to allow for a greater semantic understanding of the video. Therefore, it is common for consumers of extensive or high-density media to bear significant cognitive load and spend excessive amounts of time content-engaging with that material, particularly given the difficulties associated with manually navigating through those types of media. Manual navigation can therefore hinder various groups of users as follows:

1. **Students** must re-watch entire video lectures to revisit the main points, explanations, calculations/formulas, or concepts presented because captions may have contained mistakes
2. **Professionals** struggle to identify the decisions they made during their conversations, their follow-up action items, or other notable statements that were buried within the subsequent hours of recorded project meetings.
3. **Researchers** often experience difficulty extracting relevant points, perspectives, or information from interviews, focus groups, or conversations.
4. For the **general population**, without going back and forth through the video for a long period of time, there is no straightforward method for obtaining straightforward answers from tutorials and educational materials.

With older generations of machines, researchers typically would need to sift through information provided via recordings and use keyword searches for guidance in identifying key concepts or information contained within audio and/or video recordings of interviews, focus groups, and other similar materials. However, keyword searching does not allow for the determination of the semantic meaning of a user's search query; therefore, the machine cannot determine whether the term being searched for is included within the text of the recorded audio or video as a specific keyword; rather, the machine must rely upon a trial-and-error navigation strategy. Additionally, if available, the summaries provided via platform services and third-party applications often contain only key points or abstract information. In addition, even if these summaries were to be available, they would not provide targeted, question-driven access to content. Thus, users would still have difficulty locating a particular response to their query, for example, **"What was said by the speaker regarding the optimization of the neural networks?"** or **"What point did the interviewer address regarding financial risks?"**

To summarize, the integration of automatic speech recognition (ASR), semantic retrieval, and transformer-based question answering (QA) models is required to allow users to have an effective and accurate way to access their content via video files without having to view the entire video manually, which provides an opportunity for the development of Snap-Seek that transforms a video file into a searchable document that has certain identifiable properties.

3.2 Limitations of the Existing System

Inaccurate Search: Most platforms use basic keyword search techniques on their transcripts; however, when the phrase variations present in the transcript do not match what the user typed in, the keywords will not work. Users often use synonyms, paraphrased words, or implicit references, which traditional search methods do not pick up on. Users will miss the relevant information and see irrelevant hits from their searches based on keyword usage.

1. **Poor or Missing Summaries:** Very few online video platforms provide any type of summary to their video content. When a summary may be generated automatically, it is generally an extractive summary and short, without giving context, which makes it impossible for users to have a comprehensive overview, thus requiring them to watch the video in full to know what was discussed in the video.

2. **Lack of Interactivity:** The current platform does not allow for users to ask natural language questions such as **"What was said regarding reinforcement learning?"** and instead, they need to skip through timestamps or use limited chapter markers

to get to the information they are looking for. This lack of conversational-style interaction hinders efficient knowledge retrieval.

3. **Weak Multilingual Capabilities:** Traditional ASR technologies do not perform well on audio recorded in real-world scenarios and therefore struggle with accents, background noise, and multilingual audio. As a result, transcripts produced with ASR will contain inaccuracies that dramatically reduce the quality of summaries and the accuracy of QA. This will only become more of an obstacle as more global customers start consuming multilingual content.

4. **Dependency on Cloud Services:** Many AI Video Analysis software applications use proprietary API's hosted in the cloud to perform Transcription, Summarization, Question Answering. While they offer advantages, the reliance on them creates some potential disadvantages, such as security/privacy concerns, ongoing costs, delays in responsiveness, and decreased usability in offline/low-bandwidth conditions.

3.3 Developed System

Snap-Seek is designed to overcome limitations of existing products by offering customers one integrated AI-enabled solution that provides automatic transcription, summarization, and enables users to ask their own Natural Language Questions about content.

3.3.1 Workflow

The developed system follows a structured, multi-stage pipeline:

1. **Input Acquisition:** Customers provide either a link to a YouTube video (or similar) or a Local Video or Audio File. This functionality allows Snap-Seek to be flexible and user-friendly, allowing both Online and Offline sources of content to be supported.

2. **Video Download and Extraction:** `yt-dlp` [9] is a verified command-line utility for saving videos from the web. Afterwards, FFmpeg [10] extracts the sound file to enhance the audio quality of the source audio for later transmission or conversion into text.

3. **Automatic Transcription:** Whisper ASR [1] accurately functions and provides multilingual transcriptions from the audio files acquired using "yt-dlp" and "FFmpeg". This tool has been proven to have resilience to the differences between regional dialects, the quality of the audio file, and the variety of languages that are present in the audio; thus, it provides dependable transcriptions of video/audio content.

4. **Transcript Segmentation:** The resulting transcripts are further broken down into several smaller time-stamped groups, which facilitates the retrieval process for summarising videos, performing semantic searches, answering questions, and makes it easy for the user to identify where in the video the transcript corresponds to.

5. **Summarization:** As long segments of transcripts are summarised using the neural network model BART [2], so users can more easily understand the key points of a long video without having to go through the full transcript. This is a significant benefit to the user when they have large volumes of transcript data.

6. **Semantic Embedding:** The sentence embeddings produced by Sentence-BERT embeddings [7] are used to represent the meaning of each segment as a vector of real numbers, which allows the system to perform semantic searches to find the best-matching segments and to identify relevant segments when the user does not use the exact keywords that were found in the original transcript.

7. **Question Answering:** The ability to ask questions about the content of the video is included in the system. As questions are being asked, their associated embeddings are retrieved from the segment storage, and then the answers are retrieved from the segments using the XLM-R QA model [4] to provide users with fast and accurate information.

8. **Question Generation (Optional):** The system can use a T5-based model [3] to automatically generate a list of potential questions based on the content of the transcript, offering users an interactive way to see what they may want to learn more about or ask questions on, particularly if they do not know how to write an appropriate query to get the information they want to know.

Figure 3.1 illustrates the overall system context.

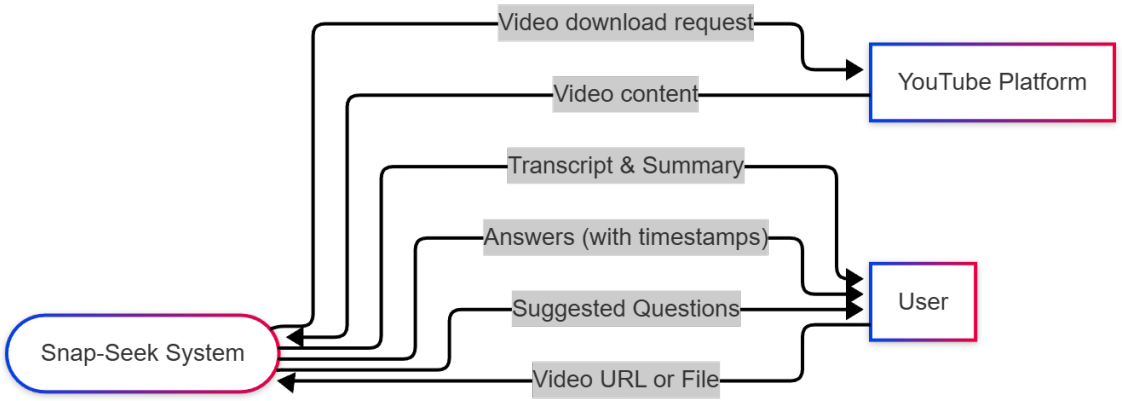

Figure 3.1: DFD Level 0: Context-level view of Snap-Seek system showing user and external entity interaction.

3.3.2 Advantages

The proposed system is advantageous for users because it:

1. **Time Efficiency:** Saves time because users do not need to watch the entire video to find answers and summaries.
2. **Accuracy:** Guarantees accuracy by using semantic search and extractive QA to provide reliable answers.
3. **Multilingual Support:** Provides a strong multilingual solution with the combination of Whisper and XLM-R.
4. **Offline Capability:** Offers offline use as all processing happens locally once the cache has been populated with the models.
5. **User-Friendly Interaction:** Allows users to interact using Natural Language Queries (NLQ).
6. **Privacy:** Keeps users' privacy intact as nothing gets uploaded to cloud-based APIs for processing.

Figure 3.2 shows the Data Flow Diagram (DFD), which breaks down Level 1 of the entire system into smaller functional components that show the sequence of major processing steps (transcriptions, summaries, semantic embedding, question generation, and QA retrieval) and the types of output data artifacts associated with each step.

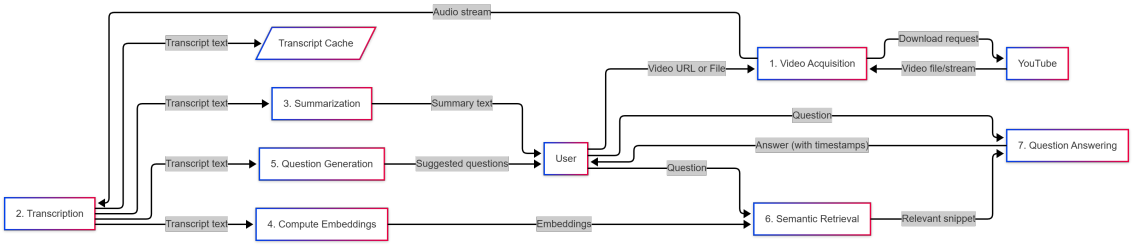

Figure 3.2: DFD Level 1: Decomposition of Snap-Seek internal modules and data flow through processes.

3.4 Requirements Specification

This section provides details about customer, functional, and non-functional requirements of the System.

3.4.1 Customer Requirements

1. Customers must have the flexibility to work with both video URLs and local files; therefore, the system must accept both in its input format.
2. Users must receive an accurate transcript with the ability to support various languages, providing the ability to tolerate background noise and accents.
3. Each segment of a video must create an easy-to-read and concise summary, allowing users to become familiar with the important points quickly without needing to read all of the transcripts.
4. Users will be able to ask their questions using natural language, creating an interactive and user-friendly experience for those without technical expertise.
5. The system will provide accurate answers with time stamp capability so users can find the exact location within the video/audio where the answer appears.
6. Suggested questions (options) will be provided to encourage user participation and help them determine what types of questions they can ask.
7. The interface will be user-friendly and simple to navigate. All functionality will be easy for users to find.
8. The system will function with or without internet access when accessing local files, thus allowing for complete functionality while offline.
9. The system will be able to handle and process long videos, while the transcripts and summaries will be processed in manageable sections so that any performance issues do not occur.
10. Users will be able to perform semantic searches (by using synonyms and/or paraphrasing) to find the relevant content they are seeking.

3.4.2 Functional Requirements

FR1-Video Download: The system will allow users to download an online video from YouTube using the yt-dlp tool. This will enable users to download and manage their online content directly from the system.

1. **Input:**YouTube video URL
2. **Process & Description:**The system will efficiently download online videos using the `yt-dlp` tool and store them locally for additional processing.
3. **Output:**Video file located on the user's local computer.

FR2 - Audio Extraction: The system is going to extract audio with an instance of FFMpeg. Extracting the audio makes it available for transcription.

1. **Input:** Downloaded video file
2. **Process & Description:** Audio is extracted from the video using FFmpeg, creating a clean audio track for transcription.
3. **Output:** Audio file (e.g., WAV/MP3)

FR3 - Speech Transcription: The system will transcribe the audio with Whisper. This will transcribe the audio into text for further use.

1. **Input:** Audio file
2. **Process & Description:** Transcription is performed using Whisper ASR, producing accurate text while handling multiple languages and noisy audio.
3. **Output:** Text transcript

FR4 - Transcript Segmentation: The system will divide/transcribe a transcript into timestamp units. This will allow for easier processing and retrieval of transcripts.

1. **Input:** Text transcript
2. **Process & Description:** The transcript is divided into timestamped segments to facilitate summarization, search, and QA.
3. **Output:** Timestamped transcript segments

FR5 - Content Summarization: The system will summarize the transcript segments with BART. This allows users to quickly get a summary of the key elements.

1. **Input:** Transcript segments
2. **Process & Description:** The system will summarize the transcript segments with BART. This allows users to quickly get a summary of the key elements.
3. **Output:** Summaries of transcript segments

FR6 - Semantic Embeddings: The system will calculate the embedding of the transcript segments using Sentence-BERT to provide the ability for users to conduct semantic searches for similarities.

1. **Input:** Transcript segments

2. **Process & Description:** Generate Sentence-BERT embeddings to capture semantic meaning for efficient search and retrieval.
3. **Output:** Semantic vector embeddings

FR7 - Semantic Retrieval: The system must retrieve relevant segments that correspond to semantic similarities, enabling accurate results for user queries.

1. **Input:** User query + embeddings
2. **Process & Description:** Retrieve segments most semantically similar to the user's query using vector similarity methods.
3. **Output:** Relevant transcript segments

FR8 - Extractive Question Answering: The system must apply extractive question-answering capabilities through the use of XLM-R, thus providing accurate answers to user-generated questions.

1. **Input:** Relevant transcript segments + user question
2. **Process & Description:** Use the XLM-R QA model to extract answers from the relevant segments.
3. **Output:** Answer(s) to the user question

FR9 - Interactive Q&A Interface: The system must support both user question input and dynamic answer output, enabling users to receive an interactive QA response.

1. **Input:** User question
2. **Process & Description:** Dynamically process user input and retrieve answers using the QA system.
3. **Output:** Real-time answer(s)

FR10 - Language Detection: The system must automatically detect the language of a user question, ensuring the assessment of multilingual question inputs is accurate.

1. **Input:** User question
2. **Process & Description:** Detect the language of the question automatically using `langdetect`.
3. **Output:** Detected language code

FR11 - Query Translation: The system must provide translation of the user question submission in the event it is submitted in a language other than English, ensuring all users have access to functioning QA.

1. **Input:** Non-English question
2. **Process & Description:** Translate to English if required using `deep-translator`.
3. **Output:** Translated question in English

FR12 - Question Generation: The system must provide suggested questions based on a T5-style generation model. This will help increase user interaction.

1. **Input:** Transcript segments
2. **Process & Description:** Generate suggested questions using a T5-style model to guide users in exploring content.
3. **Output:** List of suggested questions

FR13 - Local File Upload: The system must allow users to upload local video files. This allows users the ability to access information via the system offline, while maintaining their privacy.

1. **Input:** Local video file
2. **Process & Description:** Accept video files from local storage through drag-and-drop or file browser for processing.
3. **Output:** Validated video file

FR14 - Timestamp Navigation: The system must allow users to navigate through videos using specific timestamps for verification of answers relative to context.

1. **Input:** Timestamp from answer/transcript
2. **Process & Description:** Open video player at exact timestamp for context verification.
3. **Output:** Video playing from timestamp

FR15 - Results Export: The system must enable results to be exported in several formats. This enables users to store and distribute information that has been processed by the system.

1. **Input:** Processed results (transcript, summary, Q and A)

2. **Process & Description:** Provide export options in PDF, TXT, and JSON formats with proper formatting.
3. **Output:** Exported files in selected format

FR16 - Session Management: The system must have the capability of managing user sessions, thereby allowing the processing of multiple video uploads and resetting their respective states.

1. **Input:** User session actions
2. **Process & Description:** Manage session states, including clearing processing and switching between videos.
3. **Output:** Updated session state

FR17 - Offline Operation: The system will function offline so that there is no need for network access, which supports user privacy and provides accessibility to all users (with or without connection).

1. **Input:** Local video file
2. **Process & Description:** All processing works offline using cached models without requiring cloud APIs.
3. **Output:** Processed results offline

FR18 - Error Handling: The system will manage the handling of all errors in a way that reflects the user's confidence and inspires them to continue using the application.

1. **Input:** Error conditions
2. **Process & Description:** Detect and handle errors with clear user messages and recovery options.
3. **Output:** Error message with guidance

FR19 - Progress Indication: The system will indicate the current processing status of the application through monitoring progress throughout long processes.

1. **Input:** Processing task
2. **Process & Description:** Display real-time progress indicators for transcription, summarization, and QA operations.
3. **Output:** Progress updates

FR20 - Model Caching: To increase the application startup speed and to allow offline access to AI models, the system's models will be stored locally.

1. **Input:** First-time system use
2. **Process & Description:** Download and cache all required AI models (Whisper, BART, SBERT, XLM-R, T5) locally.
3. **Output:** Locally cached model files

3.4.3 Non-Functional Requirements

1. **NFR1 – Performance:** The software must efficiently process the majority of videos to allow video upload and analysis within an acceptable time frame (especially for moderately sized videos).
2. **NFR2 – Accuracy:** Summaries, transcriptions, and QA outputs of the system must accurately represent the information found in the source video. The important points, along with the context of the video, should be included to ensure the user understands what the video was about and does not misinterpret it.
3. **NFR3 – Usability:** The system must have a simple and user-friendly interface that allows users to understand how to use it. The steps in the upload and analysis process need to be clearly defined so users know what is happening during the process and how far along they are.
4. **NFR4 – Robustness:** The system must have error handling and should not crash in the event of unexpected or erroneous input. For example, in the case that a user uploads a video that has become corrupted, the transcript model is missing, or has been uploaded with no audio content. Instead of crashing, the system should display some form of helpful error message that informs the user what has happened.
5. **NFR5 – Security:** All data processing must occur locally on the end-user's computer so that no sensitive or private data is sent across the WWW, ensuring data privacy and user confidentiality.
6. **NFR6 – Maintainability:** The system has a modular codebase, which allows for easy updates/replacements of parts of the system, including transcription models, summarization algorithms, and user interface elements.
7. **NFR7 – Scalability:** The system will be constructed so that new features can be added in the future (e.g., visual analysis, sentiment detection, etc.) without the need for a complete system redesign.

3.5 Use Cases

Snap-Seek is utilized by students, teachers, and professionals to rapidly locate a desired passage of information in lengthy videos. Snap-Seek's primary use cases will be explained in this section. Figure 3.3 presents a visual overview. They upload a video or YouTube link to get the transcript of that video. summaries, and searchable content. It also enables users to pose a question and obtain exact answers with timestamps. The main use cases are also summarized in Table 3.1,3.2.

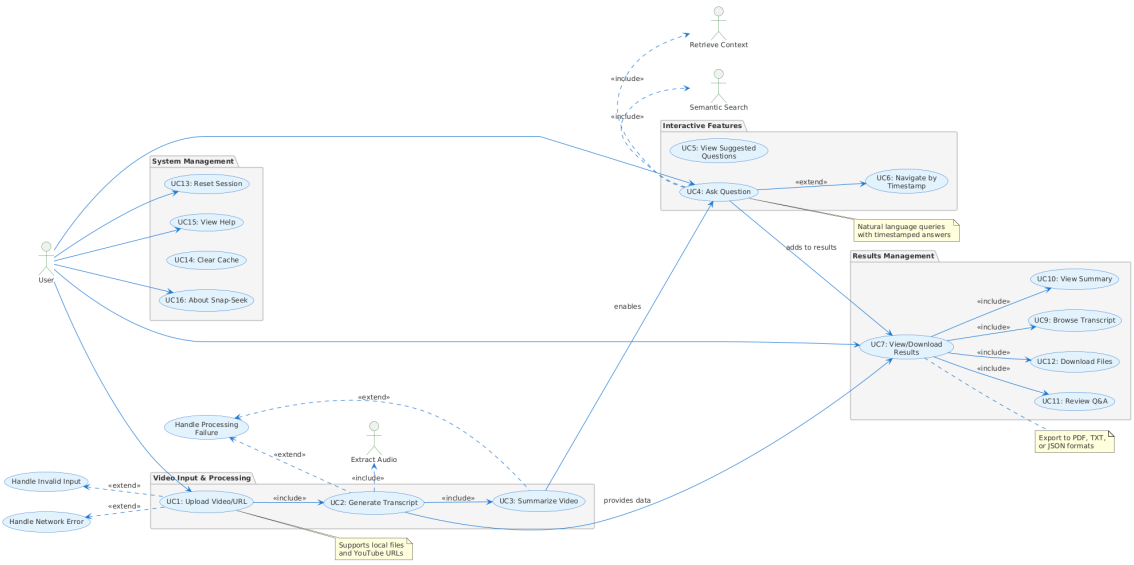

Figure 3.3: Snap-Seek Use Case Diagram illustrating user interactions.

As shown in Table 3.1,3.2 users will submit videos to Snap-Seek, generate transcripts and summaries, pose questions about content, and click on timestamps to jump to specific portions of the video.

Table 3.1: Comprehensive Use Case Specifications for Snap-Seek

UC ID	Use Case Name	Actor	Description
Video Input & Processing			
UC1	Upload Video / URL	User	User uploads a local video file or provides a YouTube URL. The system validates the format and extracts/downloads content for processing.
UC2	Generate Transcript	System	The system uses Whisper ASR to convert audio into timestamped text transcripts and supports multilingual content.
UC3	Summarize Video	System	The system processes the transcript using the BART model to generate a concise hierarchical summary of video content.

Table 3.2: Comprehensive Use Case Specifications for Snap-Seek (Continued)

UC ID	Use Case Name	Actor	Description
Interactive Features			
UC4	Ask Question	User	The user submits a natural language query. The system performs semantic search and uses XLM-RoBERTa to extract answers with timestamps.
UC5	View Suggested Questions	System	The system analyzes transcripts using the T5 model and generates relevant suggested questions for exploration.
UC6	Navigate by Timestamp	User	User clicks timestamp links in results to jump to specific video moments for context verification.
Results Management			
UC7	Reset Session	User	User clears the current session and returns to the initial state to process a new video.
UC8	Browse Transcript	User	User reads timestamped transcript segments with search and navigation functionality.
UC9	View Summary	User	User views the generated video summary with expand/collapse options.
UC10	Review Q&A	User	User reviews previous question–answer history with filtering and export options.
UC11	Download Files	User	User downloads selected outputs (PDF, TXT, JSON) to local storage.
UC16	View / Download Results	User	User accesses processed outputs including transcript, summary, and Q&A history, with download options.
System Management			
UC14	View Help	User	User accesses documentation, tutorials, and guides for using the system.
UC15	About Snap-Seek	User	User views system information, version details, credits, and technical specifications.
Error Handling			
UC12	Handle Invalid Input	System	System detects invalid input formats and provides clear error messages with recovery suggestions.
UC17	Handle Network Error	System	System manages network issues, offering retry mechanisms or offline alternatives.
Supporting Functions			
UC18	Extract Audio	System	System extracts and normalizes audio from video files for ASR processing.
UC13	Retrieve Context	System	System retrieves relevant transcript context for QA operations based on semantic matches.

In addition to the overview described above, each of Snap-Seek's major functionalities can be defined as its own use case. The following table outlines the sequence flow of steps for each of the defined use cases and includes the associated flow of actions, conditions of success, outcome flow for the Snap-Seek system modules:

UC1-Upload Video/Link: Users supply a local copy of the video file or directly paste a YouTube URL into the Snap-Seek app; At this time, the Snap-Seek system first reviews the input video format to ensure it meets system parameters and checks accessibility. After the user input has been validated, the system extracts the audio or downloads the video content and prepares it for subsequent processing. Preprocessing involves audio normalization, frame extraction, and format standardization, as well as verifying that all video and audio content is intact and will not cause issues during later processing. During preprocessing, all metadata associated with the video, including video duration, resolution, and language, is logged for reference. After completing preprocessing, downstream modules (e.g., transcription, summarization) are prepared to provide a seamless and efficient process to begin providing further analysis of the content and generating user interactions. As shown in Figure 3.4 and Table 3.3

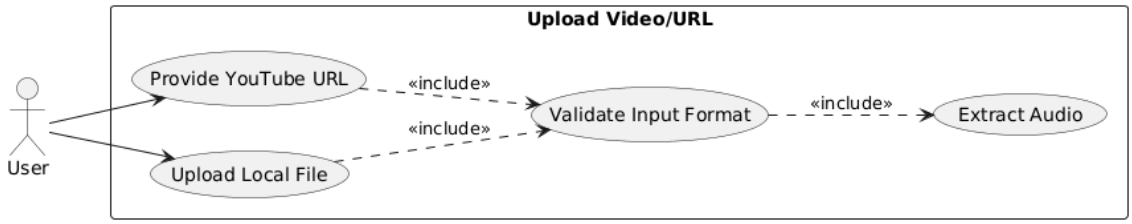

Figure 3.4: Use Case: Upload Video / Provide YouTube Link

Table 3.3: Use Case Specification 1: Upload Video / Provide YouTube Link

Use Case ID	SNAP-UC-001
Use Case Name	Upload Video / Provide YouTube Link
Actors	User
Basic Flow	User selects a local video file or enters a YouTube URL. System validates the video input. System extracts audio or downloads video content for processing.
Alternative Flow	Invalid link or unsupported format → System displays an error message.
Pre-conditions	User possesses a valid video file or YouTube link.

UC2 - Generate Transcript: The purpose of the Generate Transcript use case is to take the audio Track of the video, convert it into text using the Whisper ASR model, and create an accurate and comprehensive transcript of the content within the video. Each generated segment of text is matched by a timestamp with the corresponding video message, allowing for accurate navigation through the transcript and for users to reference specific points of interest in the transcript. The Whisper ASR model supports a variety of languages, providing a resource for users and businesses to capture multimedia content in numerous languages. The team continues to improve the performance and accuracy of Whisper ASR, even in difficult or quiet audio conditions. In addition, while preparing the transcripts, how different accents, speaking rates, and speech patterns can negatively affect the speed of transcription; therefore, the Whisper ASR model can handle different speech patterns, accents, and speaking rates effectively, generating the transcript in a manner that allows the conversation or narration to maintain its natural flow. Also, time alignment between segments allows for the downstream modules of summarization and semantic search to easily process the transcripts and to also convert them into structured data. As shown in Figure 3.5 and Table 3.4

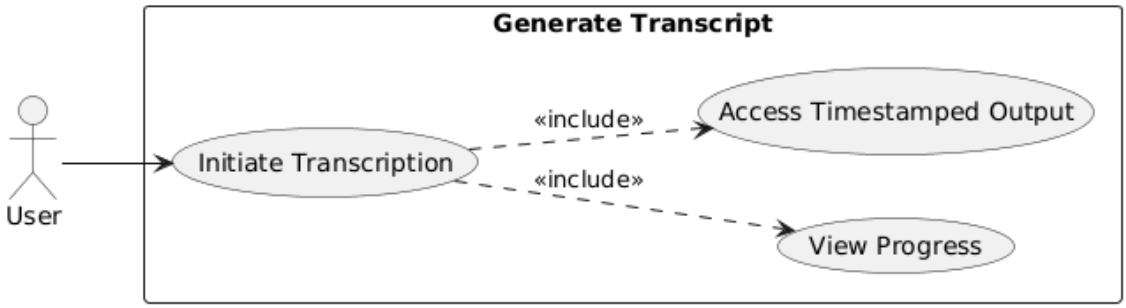

Figure 3.5: Use Case: Generate Transcript

Table 3.4: Use Case Specification 2: Generate Transcript

Use Case ID	SNAP-UC-002
Use Case Name	Generate Transcript
Actors	User
Basic Flow	User initiates transcription process from video interface. System extracts audio from the uploaded video or YouTube link. System processes audio using the Whisper ASR model. System displays timestamped transcript segments.
Alternative Flow	Poor audio quality → System notifies user of potential transcription inaccuracies.
Pre-conditions	Video has been successfully uploaded or URL provided.

UC3 - Summarize Video: The system generates a summary by processing the complete video transcript using the BART model. It produces a concise and meaningful hierarchical summary that highlights the most important topics and main ideas. Key segments of the video are captured to ensure that critical information is not missed. The summary is structured to provide both a high-level overview and detailed points for deeper understanding. This allows users to quickly grasp the essence of the video without having to watch it in full. The hierarchical organization also enables easy navigation through major themes and subtopics. The model ensures coherence and logical flow, preserving the context of the original content. Users can rely on the summary for research, review, or decision-making purposes. Overall, this module streamlines video consumption and enhances content accessibility. This module streamlines video consumption and enhances accessibility by providing personalized recommendations to users regarding online videos based on their viewing history and interests. Figure 3.6 and Table 3.5 illustrate this. As shown in Figure 3.6 and Table 3.5

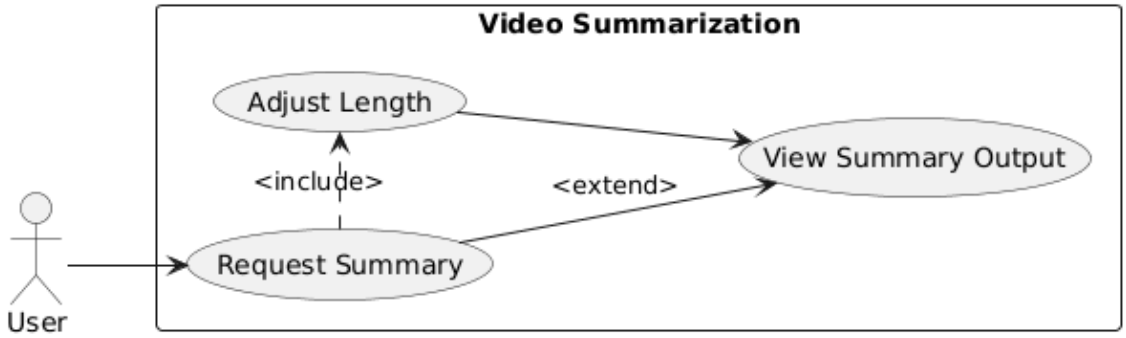

Figure 3.6: Use Case: Summarize Video

Table 3.5: Use Case Specification 3: Summarize Video Content

Use Case ID	SNAP-UC-003
Use Case Name	Summarize Video Content
Actors	User
Basic Flow	User requests summary generation from the transcript interface. System preprocesses transcript text for summarization. System applies the BART summarization model. System displays a concise, structured summary to the user.
Alternative Flow	Transcript too short → System produces minimal summary or indicates insufficient content.
Pre-conditions	Transcript has been successfully generated.

UC4 - Ask Question: This system allows users to ask a question about a specific video and to receive answers directly via a natural language interface. By employing a semantic search model against the transcript of the video, the system identifies the parts of the transcript that are relevant to the user's query based on the meaning of the words rather than relying solely on an exact match of the words used in the query. The system uses the pretrained model (XLM-RoBERTa) to match relevant portions of the transcript to the user's query and retrieve the most relevant answer with corresponding timestamps associated with each answer, allowing the user to quickly find and access the exact section of the video they are interested in. The system will produce accurate results for very complex multi-part queries and is capable of supporting queries in multiple languages due to the multilingual nature of the transcripts. Users can receive relevant responses to their questions without needing to view the entire video. Overall, this module facilitates a direct link between raw video content and what users require concerning their queries. As shown in Figure 3.7 and Table 3.6

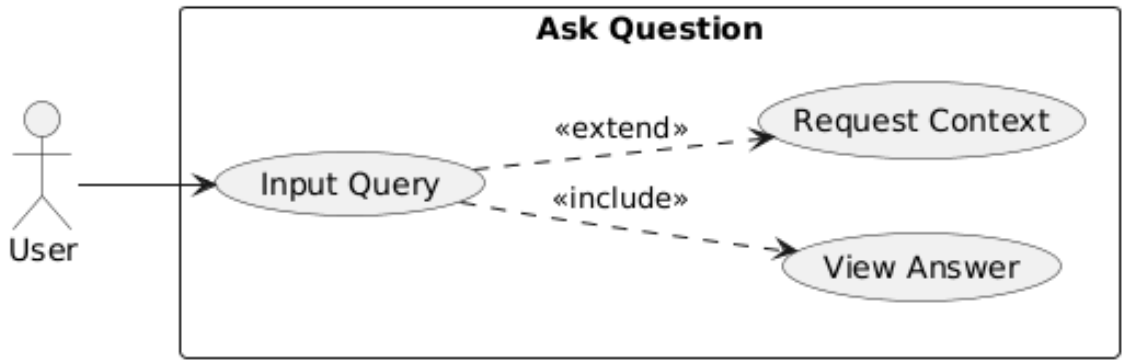

Figure 3.7: Use Case: Ask Question

Table 3.6: Use Case Specification 4: Ask Question

Use Case ID	SNAP-UC-004
Use Case Name	Ask Question
Actors	User
Basic Flow	User types a natural language question in the QA interface. System retrieves relevant transcript segments using semantic search. QA model extracts the most relevant answer from the context. System displays the answer along with the associated timestamp.
Alternative Flow	No matching answer found → System displays "Answer not available."
Pre-conditions	Transcript and embeddings have been generated.

UC 5 - View Suggested Questions: The system generates and displays sets of suggested questions relevant to a particular user's interests based on analysis of the text within that user's viewable videos using T5. Users can store these suggestion questions for later reference. The suggestions provided are based on a statistical analysis of user preferences, search terms, and viewing activity using T5's advanced question synthesis capabilities. The suggested questions allow users to quickly locate subjects of interest when looking through the video transcripts. By linking each suggested question to the video transcript segment and corresponding timestamps, users can navigate through the content with ease. Users also have the option to save the suggested questions and answers for reference later (or use offline), which will foster efficient learning, researching, and reviewing materials. By indicating the most important areas, this feature reduces how long it takes a person to gather meaningful information from the video; therefore, the utility of the feature also serves to enhance user engagement and ensure the availability of insightful information. As shown in Figure 3.8 and Table 3.7

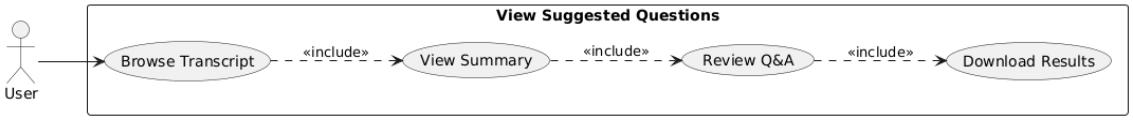

Figure 3.8: Use Case View / Download Results

Table 3.7: Use Case Specification 5: View / Download Results

Use Case ID	SNAP-UC-005
Use Case Name	View / Download Results
Actors	User
Basic Flow	User selects processed video case from history or dashboard. System displays generated transcript, summary, and Q&A outputs. User navigates through different output sections. User selects download option for preferred format (PDF, TXT, JSON).
Alternative Flow	Result data not ready → System prompts user to reprocess video or retry later.
Pre-conditions	Video processing (transcription, summarization, QA) must be completed.

UC6 - Navigate by Timestamp: The ability to navigate through timestamps allows users to navigate through videos based on the transcript of the video. By using the transcript, the system generates suggestions of relevant, context-related questions that lead users to other areas of interest. The system analyzes the video's content to find themes and the main topics to derive the questions. The suggested questions are provided in a systematic manner that is easy to understand and follow. Using the suggested question link allows individuals to see detailed answers to their selected questions or select one of the timestamp links provided (by clicking on them). This approach enables users to verify the question and quickly gain a deeper understanding of the material. An additional benefit of this module is that it allows users to quickly identify, ssummarize,and reuse large volumes of video content. In general, the module fills the gap between watching a video passively and using the video actively to learn, understand, and use it. As shown in Figure 3.9 and Table 3.8

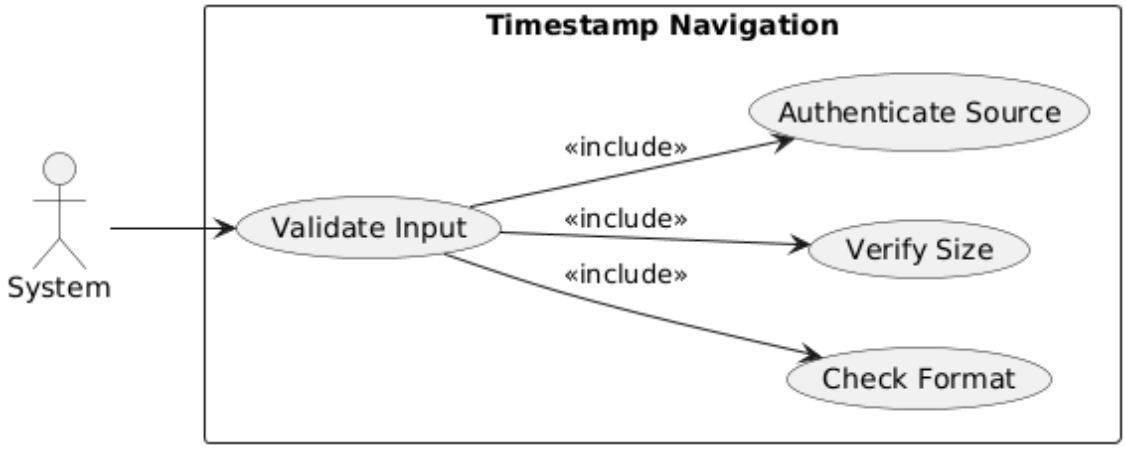

Figure 3.9: Use Case: Navigate by Timestamp

Table 3.8: Use Case Specification 6: Navigate by Timestamp

Use Case ID	SNAP-UC-006
Use Case Name	Navigate by Timestamp
Actors	User
Basic Flow	User clicks on the imestamp link from the transcript, summary, or answer. System opens the video player at the specified timestamp position. Video plays from the selected moment. User can return to the interface after verification.
Alternative Flow	Timestamp out of range → System displays error and suggests nearest valid timestamp.
Pre-conditions	Video is availabl,,e and timestamps have been generated.

UC7 - Reset Session: Themodule allows for exploring the video by providing question suggestionsand for further exploration via the auto-generated question based on the transcript. Users can see what the main points, topics, and insights from the video are, which are then turned into contextualised questions. Users can select one of the suggested questions or jump to the associated part of the video to quickly access the information related to their question. The system also allows users to view and access the full transcript, hierarchical summary of the video, and question history for review,w as well as a more in-depth analysis of the video and QA as available within the platform. Users have access to download all processed outputs in several formats (including PDF, TXT, a nd JSON) for download for use, so they may keep their valuable insights. The integration of question suggestions with downloadable outputs will enhance accessibility and usability of the system. Overall, it is expected that this module will enhance the efficiency of learning, research, and managing content. As shown in Figure 3.10 and Table 3.9

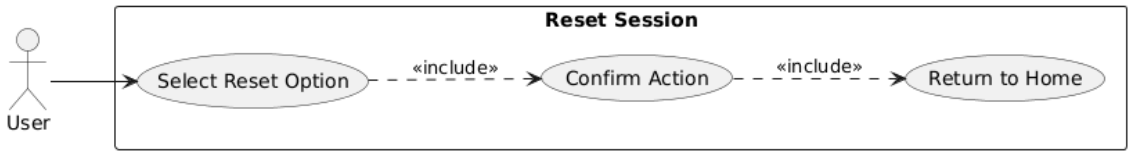

Figure 3.10: Use Case: Reset Session

Table 3.9: Use Case Specification 7: Reset Session

Use Case ID	SNAP-UC-007
Use Case Name	Reset Session
Actors	User
Basic Flow	User selects the reset option from the menu or interface. System prompts for confirmation to clear the current session. User confirms reset action. System clears all current video data and returns to the home screen.
Alternative Flow	User cancels reset → System returns to current session without changes.
Pre-conditions	User is in an active session with processed video data.

UC8 - Browse Transcript: Users can browse the transcript that has been given timestamps to find materials that relate to their needs effectively. Browsing through timestamps allows users to access specific sections within a video by providing navigation and search capabilities. Users can therefore find precise references and enhance their understanding of the contents of a video without having to view the video in its entirety. Users also have the option of filtering timestamps based on their preferred keyword or topic, thus providing a much more comprehensive review process by allowing users to navigate quickly between related timestamps. Finally, important phrases are highlighted in the interface, and timestamps align with video timestamps to maintain contextual clarity. As shown in Figure 3.11 and Table 3.10

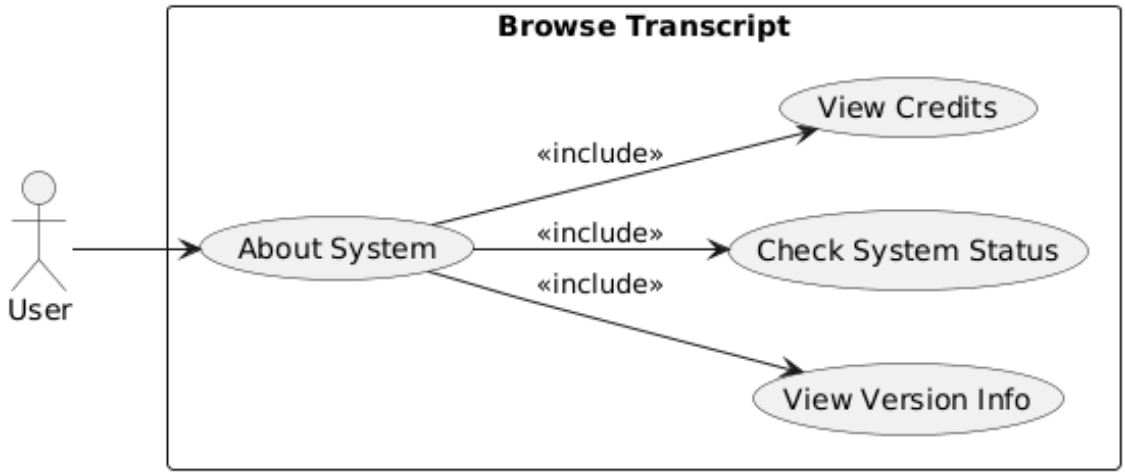

Figure 3.11: Use Case: Browse Transcript

Table 3.10: Use Case Specification 8: Browse Transcript

Use Case ID	SNAP-UC-008
Use Case Name	Browse Transcript
Actors	User
Basic Flow	User accesses the transcript section from the main dashboard. System displays timestamped transcript segments. User scrolls through transcript content. User can click on timestamps to jump to specific sections. User can use the search functionality within the transcript.
Alternative Flow	No transcript available → System displays message to generate transcript first.
Pre-conditions	Video has been processed and a transcript generated.
Post-conditions	User has reviewed transcript content and identified relevant sections.

UC9 - View Summary: Users can obtain a brief overview of a given video through a concise summary created through a BART model generated from the video content. This summary is organized into a hierarchical structure, allowing users to click on sections to expand/collapse content. The hierarchical format of the summary will facilitate users' ability to quickly determine the primary topic of a video while concentrating on a particular area of interest. Users will also be able to find highlighted sections of an overview and subtopics for better understanding and navigation within a video. By providing a clear overview, the system reduces information overload and saves time. It enhances user engagement by allowing interaction with summarized content. As shown in Figure 3.12 and Table 3.11

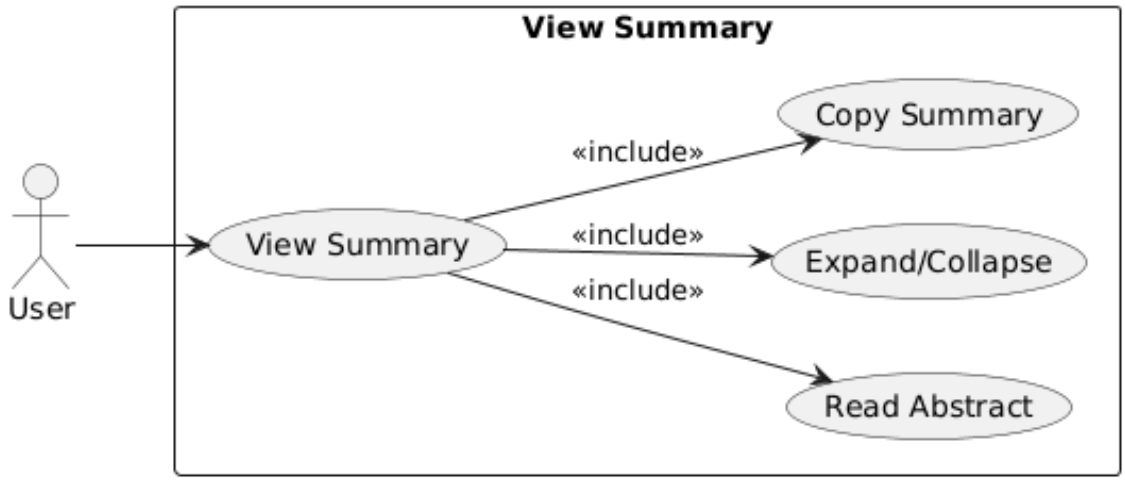

Figure 3.12: Use Case: View Summary

Table 3.11: Use Case Specification 9: View Summary

Use Case ID	SNAP-UC-009
Use Case Name	View Summary
Actors	User
Basic Flow	User navigates to the summary section. System displays a concise video abstract generated by BART. User reads the summary to understand the key video points. User can expand/collapse sections for more detail. User can copy the summary text for external use.
Alternative Flow	Summary not generated → System offers option to generate summary.
Pre-conditions	Video has been processed and a summary generated.
Post-conditions	User has comprehended the main video content through the summary.

UC10 - Review Q&A: Users can review previously asked and answered questions (QA) that they engaged with, so they can keep track of and analyze information gathered from video content. The system createsan organized history of the user's QA interactions and allows the user to filter their history by keyword, topic, or the time they engaged with the video to aid the user in quickly finding relevant information when navigating through the extensive information obtained by the system. Users may also export their QA history into several file types (e.g., PDF, TXT, JSON, etc.) that allow for retrieval and analysis in the future or offline. The QA review capability allows for increased knowledge retention of what has been learned from the video material, so users can go back to the essential concepts or ideas learned from the video. As shown in Figure 3.13 and Table 3.12

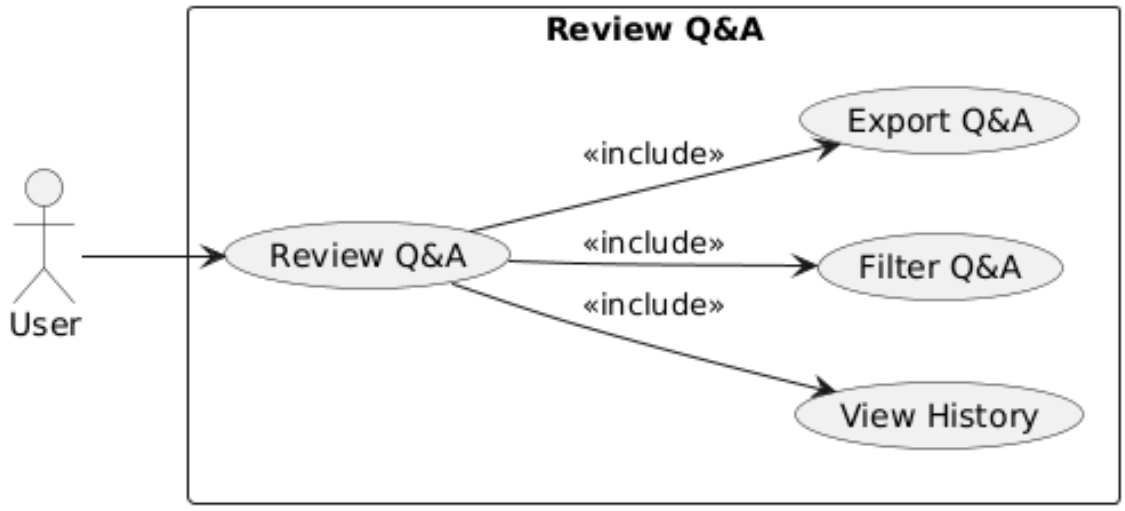

Figure 3.13: Use Case: Review Q&A

Table 3.12: Use Case Specification 10: Review Q&A

Use Case ID	SNAP-UC-010
Use Case Name	Review Q&A
Actors	User
Basic Flow	User accesses Q&A history section. System displays a chronological list of asked questions and answers. User reviews previous interactions. User can filter Q&A by timestamp or relevance. User can export Q&A history for reference.
Alternative Flow	No Q&A history → System displays empty state with prompt to ask questions.
Pre-conditions	User has previously asked questions about the video.
Post-conditions	User has reviewed previous interactions and can reference them.

UC11 - Download Files: Users have the option to export the input files they have selected, which may include transcriptions, summaries, or QA histories, to their local storage in formats supported by the application, including PDF, TXT, and JSON. The feature allows users to access their exported data offline and share processed information with others. Once exported, all information exported maintains the same formatting, timestamping, and structure as the original for accurate reference. Users can choose whether to export a specific portion of their selected output files or all output files at once. After exporting, users have the ability to save their exported output files for future analysis, documentation, or research purposes and incorporate the functionality into their workflows as a means to create their own personal video insight repository. As shown in Figure 3.14 and Table 3.13

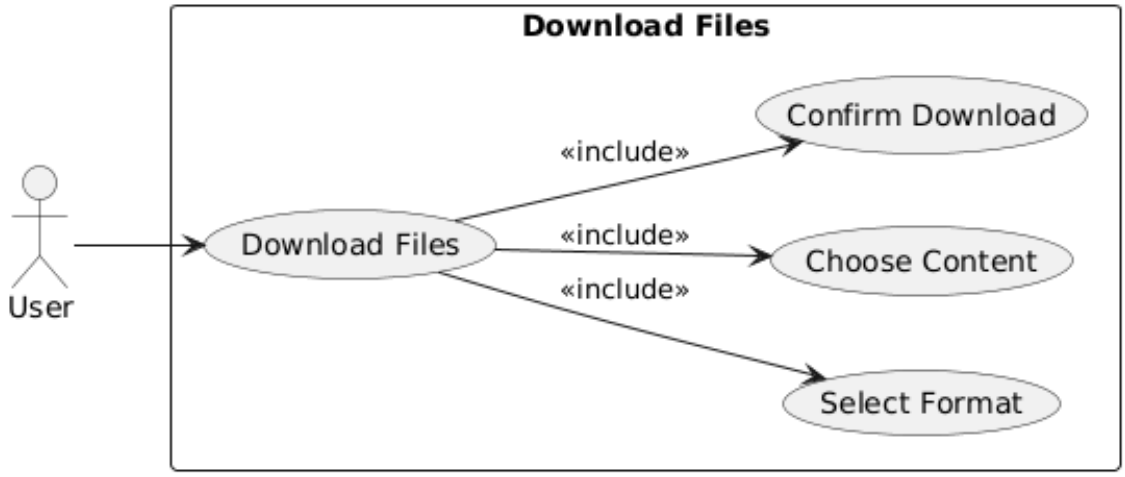

Figure 3.14: Use Case: Download Files

Table 3.13: Use Case Specification 11: Download Files

Use Case ID	SNAP-UC-011
Use Case Name	Download Files
Actors	User
Basic Flow	User selects the download option from the results section. System presents format options (PDF, TXT, JSON). User selects desired format and content (transcript, summary, Q&A). User confirms download location. System generates and downloads the file.
Alternative Flow	Insufficient storage → System displays storage error and suggests alternative location.
Pre-conditions	Processing results are available for download.
Post-conditions	User has downloaded the selected content in the chosen format.

UC12 - Handle Errors: Users can reset their current session, which clears their active session, including uploaded videos, transcriptions, summaries, and QA history, enabling the system to function in its original state and without any previous session data when new videos are processed through the system. By clearing out old data, the reset option ensures that the user can maintain a clean and efficient operation of their system and minimize clutter within their storage, ultimately reducing the chances of errors from the presence of duplicate or outdated data. Users' ability to do new analyses without being influenced in any way by previous analyses with residual output gives them the confidence of knowing that they are getting the right result. This is an important capability in multi-user or shared environments where it is important to isolate session data from multiple users, and for managing shared resources effectively to maintain long-term stability of the system. As shown in Figure 3.15 and Table 3.14

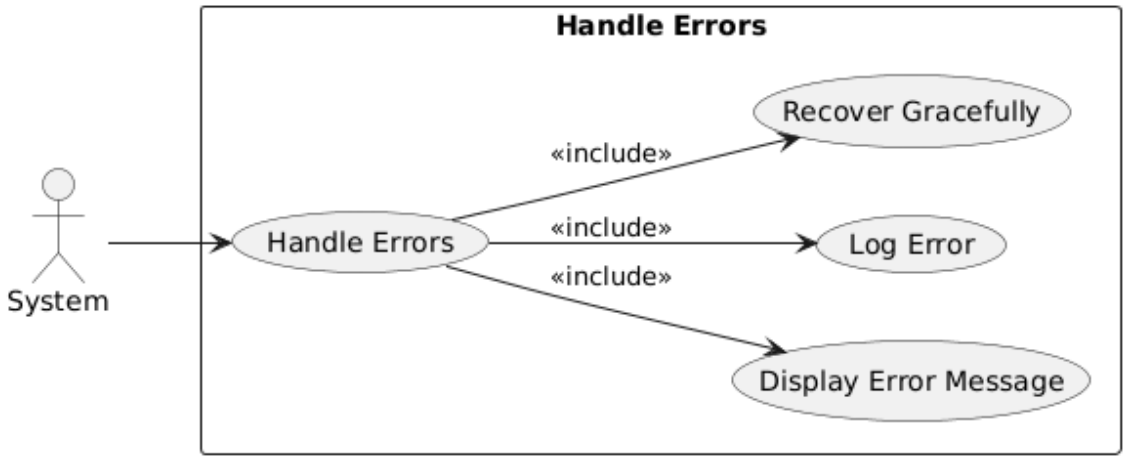

Figure 3.15: Use Case: Handle Errors

Table 3.14: Use Case Specification 12: Handle Errors

Use Case ID	SNAP-UC-012
Use Case Name	Handle Errors
Actors	System
Basic Flow	System detects an error condition during processing. System categorizes error type (input, processing, output). System displays a user-friendly error message. System logs error details for debugging. System attempts graceful recovery or provides recovery options.
Alternative Flow	Critical error → System terminates process safely and suggests restart.
Pre-conditions	System is operational and monitoring error conditions.
Post-conditions	Error is handled appropriately without system crash.

UC13 - Validate Input: Users can easily 'Clear' the cache/temp files from their system's storage to improve system performance and make it more efficient. This will delete all residual output from previous sessions, including processed transcripts, summaries, and question/answer outputs. By clearing the cache/temp files, users ensure they will not have to deal with the effects of leftover temp files during new video-processing decisions. Another important benefit of clearing cache/tmp files is the maintenance of users' data privacy, as well as cleaning out unneeded or sensitive information stored during previous sessions, so users to be able to access their system quickly and easily without additional resources. Users can regularly clear cache/tmp files to reduce clutter in their system memory, improving the response time of their systems. By utilizing this feature, users will find the system is much better able to manage memory and therefore lower the potential for slow performance or errors. As shown in Figure 3.16 and Table 3.15

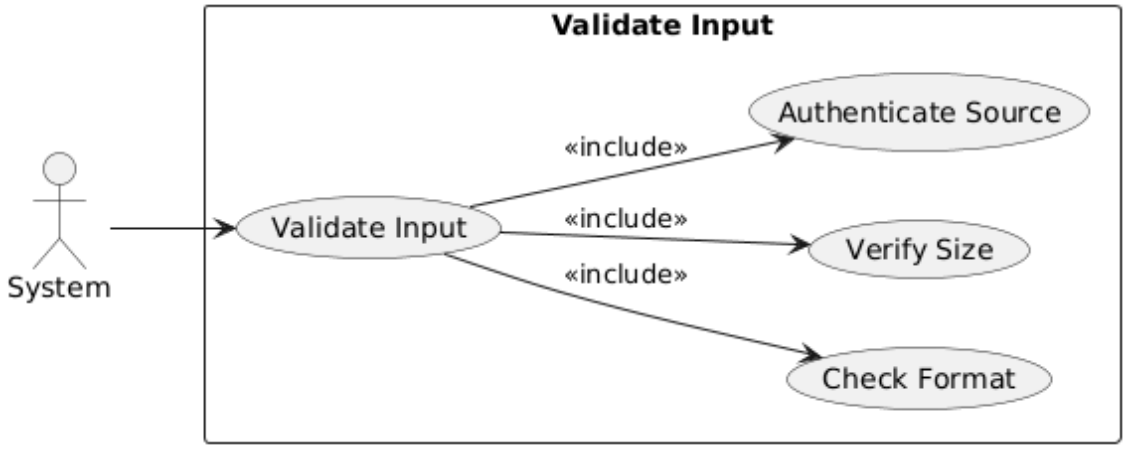

Figure 3.16: Use Case: Invalid Input

Table 3.15: Use Case Specification 13: Validate Input

Use Case ID	SNAP-UC-013
Use Case Name	Validate Input
Actors	System
Basic Flow	System receives user input (video file or URL). System checks file format compatibility. System verifies file size constraints. System authenticates the source for URL inputs. System returns validation result (valid/invalid).
Alternative Flow	Invalid input → System provides specific error message about validation failure.
Pre-conditions	User has provided input for processing.
Post-conditions	Input is validated and ready for processing if valid.

UC14 - Access Help: Users can access system documentation, tutorials, and support resources with the intent of learning about the different types of functions available in Snap-Seek. This documentation provides step-by-step instructions on how to upload video files, obtain transcripts, produce summary documents, and execute question-answer operations. Tutorials provide hands-on, end-to-end workflow demonstrations, along with scenarios that demonstrate how users can effectively navigate the Snap-Seek application from the user's perspective. All support documentation addresses frequently encountered issues and includes recommendations for resolving these issues. Since users can use these resources when using Snap-Seek, they are able to get more from the software and achieve more accurate results. The documentation and tutorials are well-designed, so they can be easily followed by both new and experienced Snap-Seek users. In addition, access to the support resources reduces reliance on other sources for assistance with learning how to use Snap-Seek. As shown in Figure 3.17 and Table 3.16

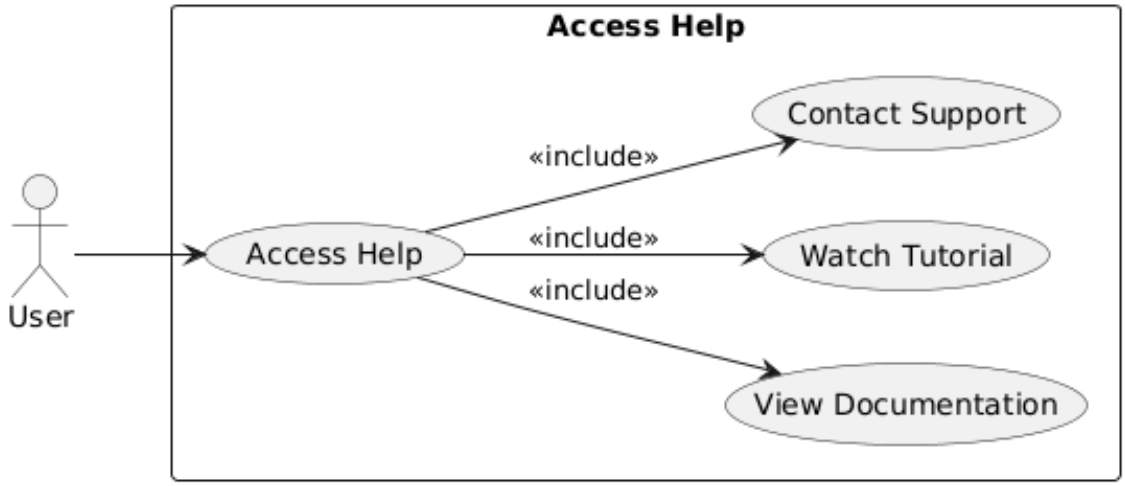

Figure 3.17: Use Case: Access Help

Table 3.16: Use Case Specification 14: Access Help

Use Case ID	SNAP-UC-014
Use Case Name	Access Help
Actors	User
Basic Flow	User clicks the help icon or accesses the help menu. System displays help documentation and user guide. User can browse topics or search for specific help. User can watch tutorial videos if available. User can access the contact support option.
Alternative Flow	Offline mode → System displays cached help documentation.
Pre-conditions	System is running,, and help resources are available.
Post-conditions	User has accessed help information as needed.

UC15 - About System: Users will be able to view various forms of information associated with Snap-Seek, such as version information, credits, and technical specifications about the software. This section includes important information regarding the various software components, models, and technologies that comprise Snap-Seek, as well as a list of the people who contributed to the development of Snap-Seek. This provides users with the ability to learn about system capabilities, updates, and the technical specifications for the system in order to use Snap-Seek most effectively. As shown in Figure 3.18 and Table 3.17

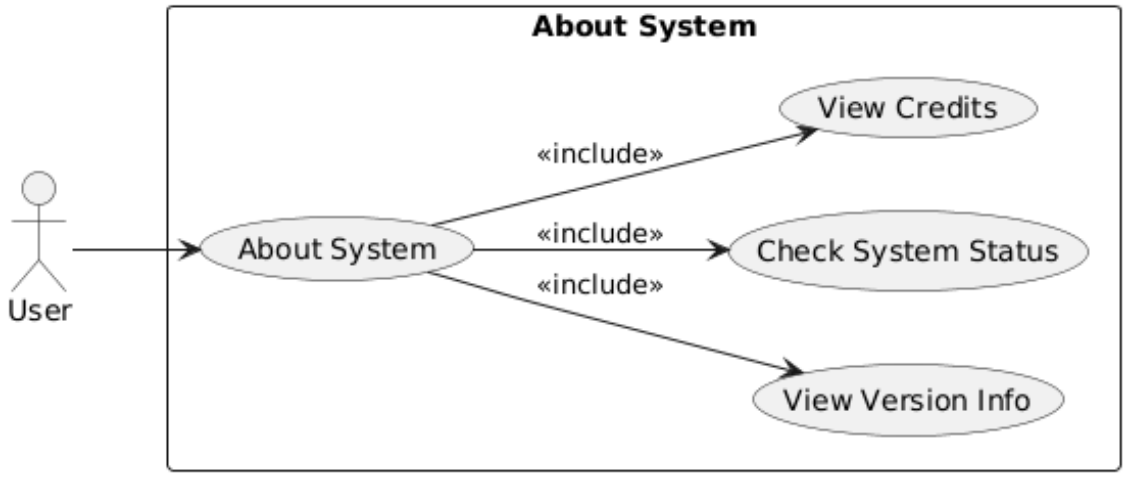

Figure 3.18: Use Case: About System

Table 3.17: Use Case Specification 15: About System

Use Case ID	SNAP-UC-015
Use Case Name	About System
Actors	User
Basic Flow	User accesses the "About us" section from the main menu. System displays version information and build details. System shows the current system status and resource usage. User can view credits and acknowledgments. User can check for updates if online.
Alternative Flow	System offline → Displays offline mode notice in about section.
Pre-conditions	System is running.
Post-conditions	User has viewed system information.

3.6 Summary

This chapter discusses the limitations of current systems, and it describes the proposed Artificial Intelligence-based solution for Snap-Seek. The workflow for Snap-Seek is structured so that each video can be transcribed, semantically retrieved, summarized, and

interacted with, with questions answered correctly. Snap-Seek identifies the customers' needs as well as all requirements of the system (functional and non-functional) in relation to each other and the overarching goal of the system (i.e., Snap-Seek). Additionally, this chapter outlines the different aspects of architecture that will help to build Snap-Seek.

Chapter 4

System Design

4.1 System Design

The Snap-Seek system implements a modular approach that adds scalability and the ability to work offline to the integration of many different Artificial Intelligence technologies into a unified pipeline to produce structured knowledge from unstructured raw video footage. The currently available AI technologies used by Snap-Seek and shown in this unified pipeline are Automatic Speech Recognition (ASR), Summarisation, Semantic Retrieval and Extractive Question Answering. Each individual AI technology has been created and is maintained by separate development teams, with interdependence and communication established through predefined data flows, allowing for simpler troubleshooting and maintenance. The overall Snap-Seek architectural approach has been structured to allow for re-utilisation of intermediate results (transcripts, summaries, and text embedding), which allows for increased efficiency by avoiding the need for duplicated computation. The Snap-Seek architecture was also created with optimised model caching and loading functions to facilitate use in low or no internet connectivity scenarios, thereby allowing users to utilise the core capabilities of Snap-Seek offline or without cloud services being utilised. Snap-Seek is designed to be extensible and therefore has facilities for future enhancements, including multimodal understanding of video, sentiment analysis, and the extraction of visual scenes. This chapter presents an overview of Snap-Seek's architectural model, methodologies, high- and low-level designs and external interfaces, all of which are essential elements to the core of Snap-Seek's functionality.

4.2 System Architecture

Snap-Seek is designed in a "Layered Pipeline" system. Each layer os comprised of multiple unique critical transformations of a video file's information from an unrefined format to a searchable, useful format. The Snap-Seek Data Pipeline is initiated at the time of video

acquisition and ends with the interactive interface for the audience. The Data Pipeline operates using transcription, followed by advanced analyses based on NLP, leading to providing the means by which the audience can semantically search through the results and or answer any questions pertaining to them using Snap-Seek. The Modular Architecture is the key to maximizing scalability, maintainability, and therefore the overall efficiency of the Snap-Seek system.

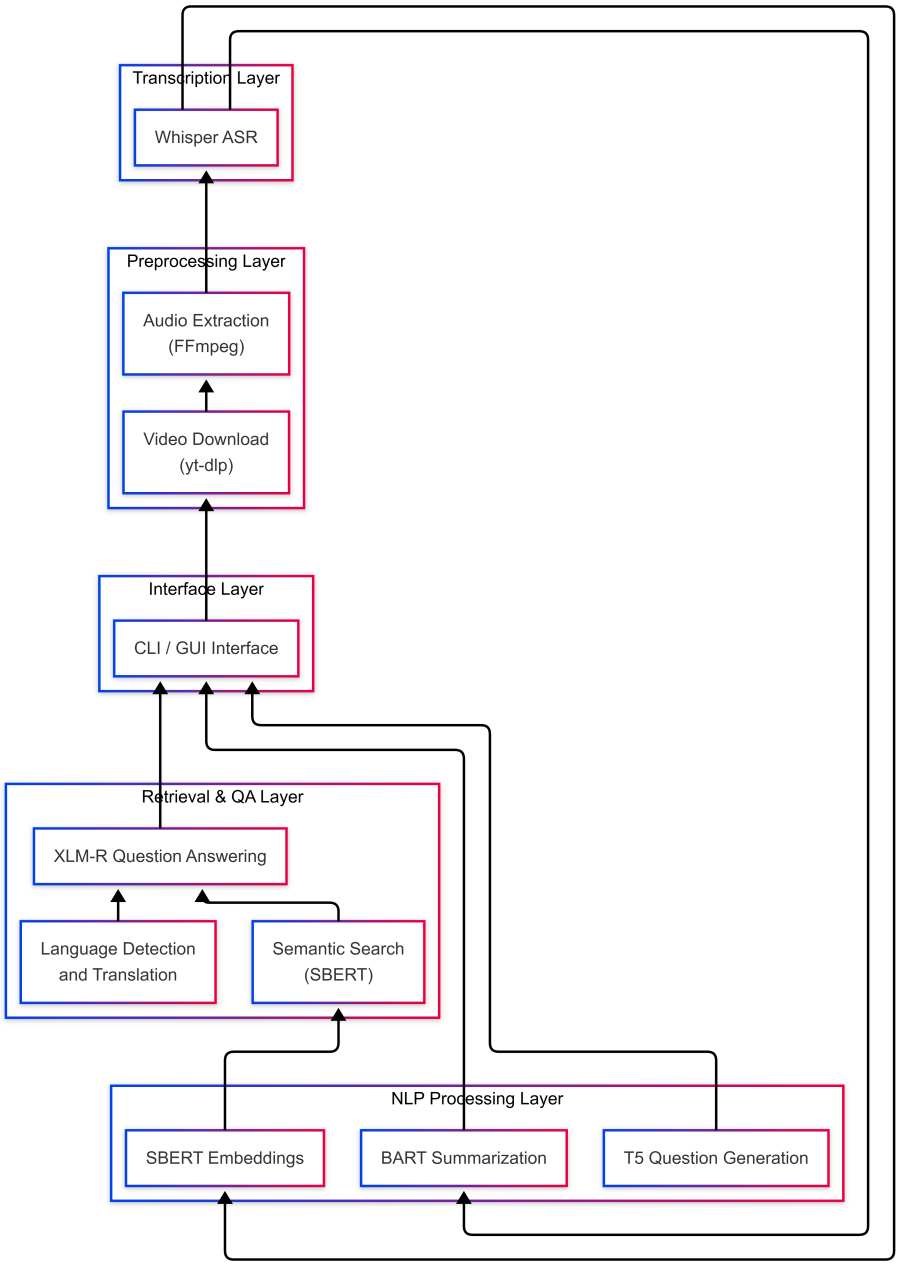

Figure 4.1: Architectural overview of Snap-Seek pipeline.

1. **Preprocessing Layer:** This is the initial layer within the Snap-Seek system where the downloading of the video files from the web (i.e., the Internet) is conducted

via the use of `yt-dlp` [9] and from which the extraction of high-quality audio is achieved using FFmpeg [10]. The Preprocessing Layer is also responsible for standardizing the audio parameters relating to the downloading and extraction to ensure compatibility with all ASR models downstream of this layer.

2. **Transcription Layer:** This layer is where the audio is converted to text using Whisper ASR [1] and creates text output files in multiple languages and formats with time stamps on each word. In addition to converting the audio to text, the Transcription Layer incorporates error handling for the audio of video files that contain a substantial amount of background noise, and the output text files are broken down to smaller chunks of transcription

3. **NLP Processing Layer:** The NLP Processing Layer uses several different techniques for processing text in order to understand what a piece of writing really means. It uses BART [2] to create a new summary of a video, and it uses SBERT [7] to create Semantic Embeddings (SE). It can also create suggested questions using a T5-style model [3]. The NLP Processing layer also performs several cleaning and formatting functions on the text, including Normalized Text Format (NTF) and Tokenization.

4. **Retrieval & QA Layer:** The Retrieval and QA Layer searches for the best segments of transcripts that match the user's question using the SBERT/SE to find the right segments of text. It also provides an answer from the video and its timestamps using extractive question answering (QA) powered by XLM-RoBERTa [4]. This layer also supports queries in multiple languages by providing language detection and translation modules.

5. **Interface Layer:** The Interface Layer serves as the communication bridge connecting users with Snap-Seek. Users input videos, ask questions, and view Snap-Seek results through a command-line interface (CLI)/graphical user interface (GUI). Results from Snap-Seek include a transcript of the video, a summary of the video, a list of semantic matches, and the related timestamp in addition to a link to the related answer. Additionally, results can be downloaded in various formats including PDF, TXT, and JSON.

Figure 4.1 shows how the Snap-Seek architectural layers interact and outline the data flow throughout the ASR, NLP processing, and Interface Layer.

4.3 Design Constraints

The following represent some of the constraints that influenced how Snap-Seek was designed and developed, which in turn affected key architectural components and overall operational strategies.

Hardware Constraints

The use of large transformer models has increased their memory demands, which limits the ability of low-resource machines to perform well. Low-resource (CPU-only) machines should consider optimally chunking/batching/reducing the size of the model so that processing time remains acceptable. While GPU acceleration speeds up the process of transcription, embedding, and question answering, it is considered optional to allow easy and wider access. Storage must be large enough to hold multiple downloaded models and cached embeddings.

Software Constraints

Since the application must operate offline, all models, tokenizers and dependencies for AI must be stored on the local device. To provide users with multiple choices of the model size, (e.g., Whisper-small vs. Whisper-large), users can balance speed with accuracy. External tools, such as FFmpeg and `yt-dlp`, should be included in the system path and be compatible with the operating system. To ensure that devices using Python environments do not experience conflicts due to differing versions of AI library dependencies, these dependencies should be consistent.

Operational Constraints

Several factors may affect the quality of transcriptions produced. For example, input videos will differ from one another in terms of quality, language, duration, noise, and accent; this could all affect the accuracy of any transcription. Additionally, the size of the generated transcript may exceed the size limits of the summarization and question answering (QA) model. To overcome this issue, hierarchical chunking and management of context will be necessary. Users may ask questions that cannot be answered due to their ambiguities or vagueness or which may not have the necessary background context; to handle this situation, fallback responses and confidence scores should be developed. Long-form video files will create large embedding matrices; therefore, they will take up significant time and memory to retrieve.

4.4 Design Methodology

The Snap-Seek process was implemented using a stepwise, modular approach to building the application, with verification of each part working correctly before moving to the next. The first part of Snap-Seek to be implemented and tested was the transcription feature in isolation. This was the base for additional features that were added incrementally as Snap-Seek began functioning reliably. When Snap-Seek had a well-functioning transcription

feature, the summarizer functionality was implemented. This feature uses different levels of hierarchical chunking and creates a summary of each chunk of text that is generated. After the summarization process worked, the next step in Snap-Seek's creation was to implement semantic embedding and retrieval. This is accomplished by creating Sentence-BERT embeddings of the transcript and allowing for similarity searches within the transcripts. Once all of the features for Snap-Seek were in place, the next step was to add the extractive question-answering module (XLM-RoBERTa). This module allows users to ask questions and retrieve answers from the transcript content. In the fifth step of project development, a user interface was created. Originally, Snap-Seek had a command line interface; later, a graphical interface was developed to provide users with easier access to the application. Finally, additional advanced features such as automatic question generation (using T5 to create questions) were added along with expanded multilingual capability (by detecting the language and translating when needed). The step-wise structure of this design process provided an opportunity to validate each module individually as the project progressed; thus, all issues found were resolved promptly. By constructing functionality in a modular fashion, developers could maintain distinct areas of expertise within their own domain, while ensuring that the system as a whole was stable and able to react in a timely manner, relative to changes or additions made to it. Consequently, all features and functions, including those related to transcription, summarization, searching, and answering queries, worked harmoniously together as they were developed. Thus, as functionality was added through the adding modules, the new modules increased the system's capabilities while preserving the integrity of prior stages.

4.4.1 System Development Steps

In system development the system is designed using an incremental and modular methodology:

Step 1: Completed the base transcription capabilities.

Step 2: Added summation and hierarchical chunking.

Step 3: Developed methods of using semantic embeddings to retrieve audio files.

Step 4: Applied extractive question-answering methods with XLM-R.

Step 5: Implement interactive CLI/GUI.

Step 6: Added question generation capability and multilingual support.

The methodology of developing the Snap-Seek system is a bottom-up approach that guarantees the stability of the Snap-Seek system and keeps the concerns of each development module segregated from one another.

4.5 High-Level Design

Using a pipeline style of construction, Snap-Seek takes raw video content and converts it into interactive searchable information. Users have the ability to either upload a video file or supply the URL of the video, and Snap-Seek automatically retrieves the audio portion of the video. The audio is then converted into time-stamped text, which can be used to perform any of the following tasks in parallel using natural language processing: generating semantic embeddings for retrieval, summarizing audio transcripts, and generating optional questions. The retrieval module of the Snap-Seek system allows users to ask questions and obtain the context needed to answer them. Presenting transcripts, summaries, suggested questions, and answers interactively is the interface layer's purpose. The interactivity and modularized structure allow scalability, multilingual support, and independent upgrades of individual modules.Further, this framework makes it possible to have components of the system upgraded whenever new AI models become available. This means that every phase of the system can be individually optimized without affecting the whole process. This flexibility not only ensures better performance but also simplifies system maintenance. Snap-Seek stands ready to integrate the future advancements that may come in speech recognition and language understanding. This means that the final user of the system gets to benefit from continuous improvement.

4.5.1 System Flowchart

Figure 4.2 presents a visual representation of the data processing through the Video Assistant, showing a complete path from video ingestion to question answering. Initially, the video file or URL was validated and pre-processed before audio extraction. Audio extraction and normalization were then followed by transcribing audio to a time-stamped text using the Whisper ASR model.

Next, the time-stamped transcripts were processed through the BART text processing model to clean up, segment and summarize the text. After word segmentation, the final transcripts were referred to as embeddings; these embeddings will be stored and searched according to their semantic meaning.

Finally, the processed data is available for users to query the system using natural language and receive the related portions of the transcript and correct answers with a corresponding timestamp.

In addition to functionality, the flowchart includes error handling, progress tracking, and caching solutions to enhance system robustness and performance.

The flow diagram also depicts how the modules interact with each other and define the control flow through the AI Assistant's various functions.

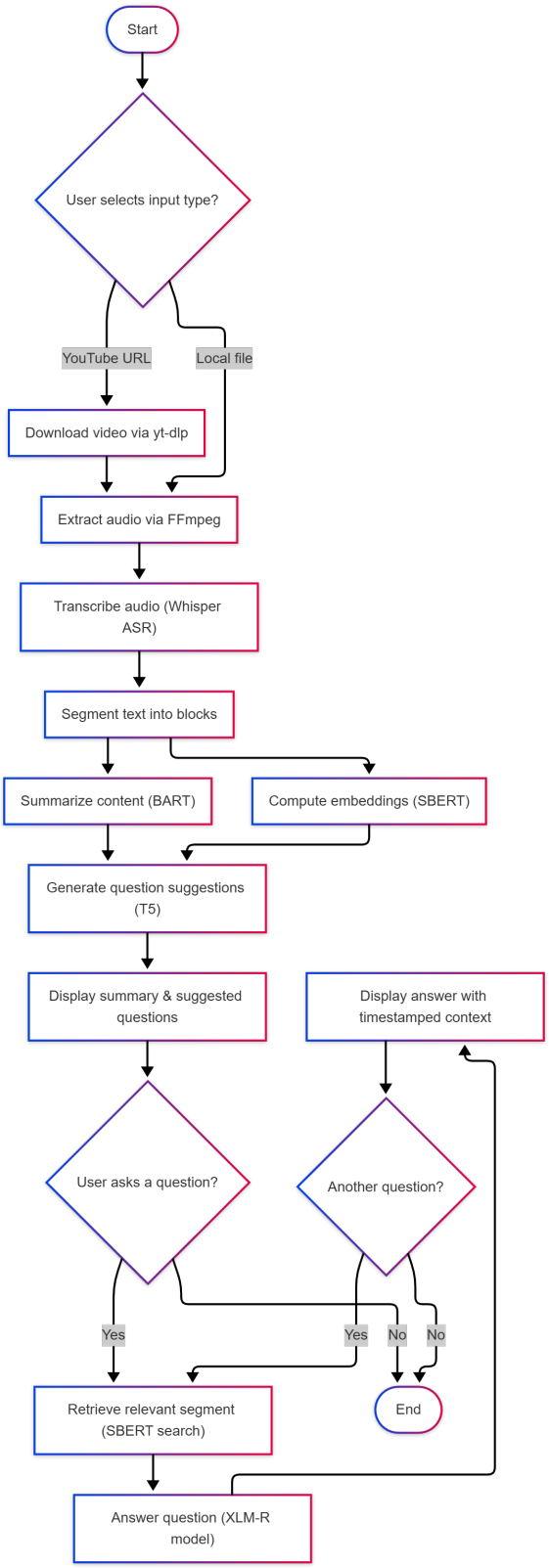

Figure 4.2: System workflow flowchart outlining the step-by-step data processing pipeline.

4.5.2 Activity Diagram

The interactions between users and systems are explained. Figure 4.3 shows how the logic flows depending on conditions. The conditional flows of user input and actions performed by the system include uploading videos, validating the video input, transcribing the video file, generating summary representations, answering user questions. In Figure 4.3, decision points, parallel processes and alternate flows highlight how the system processes user input in various circumstances (i.e. the uploaded video is invalid, transcription fails).

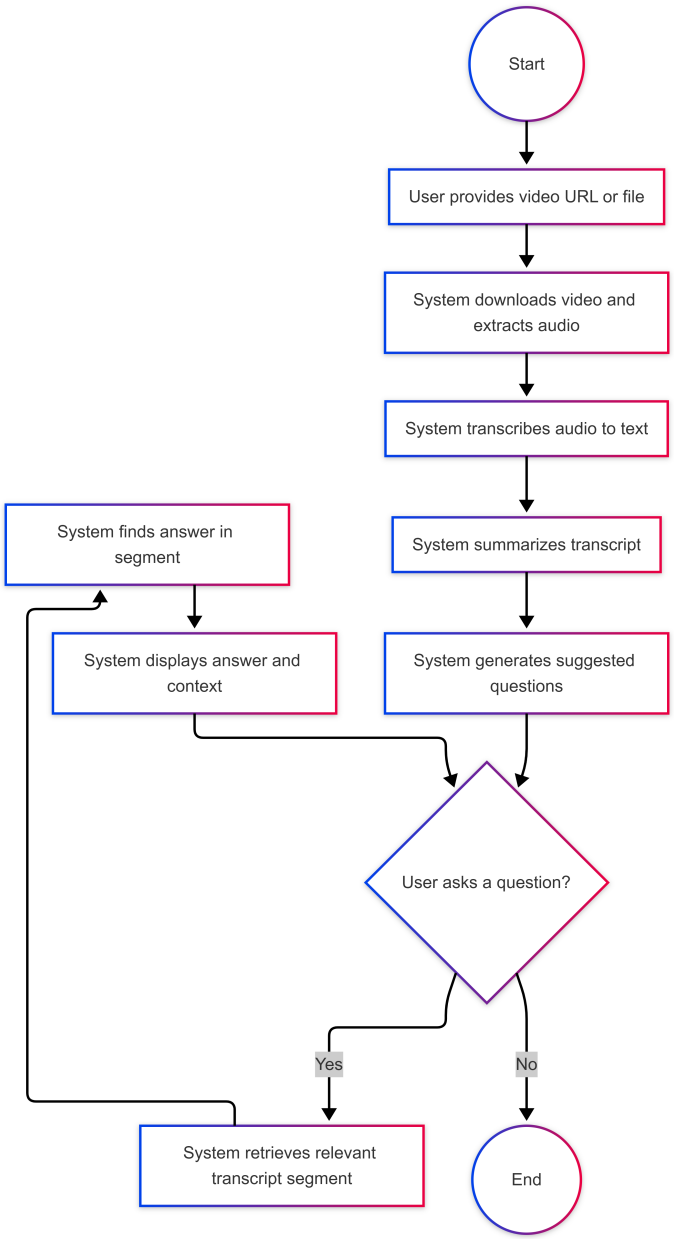

Figure 4.3: Activity diagram showing user and system interaction flow.

4.5.3 Sequence Diagram

Figure 4.4 illustrates communication between modules based on user actions. Specifically, this diagram illustrates the interaction between the user interface, the video processing module, the ASR Engine, the summarization model, the semantic search engine, and the QA Module. The sequence diagram shows the message order, the data transfers between components, and the component dependencies; thus providing an understanding of how the system manages to process multiple tasks, as well as providing a seamless experience in the form of an interactive system.

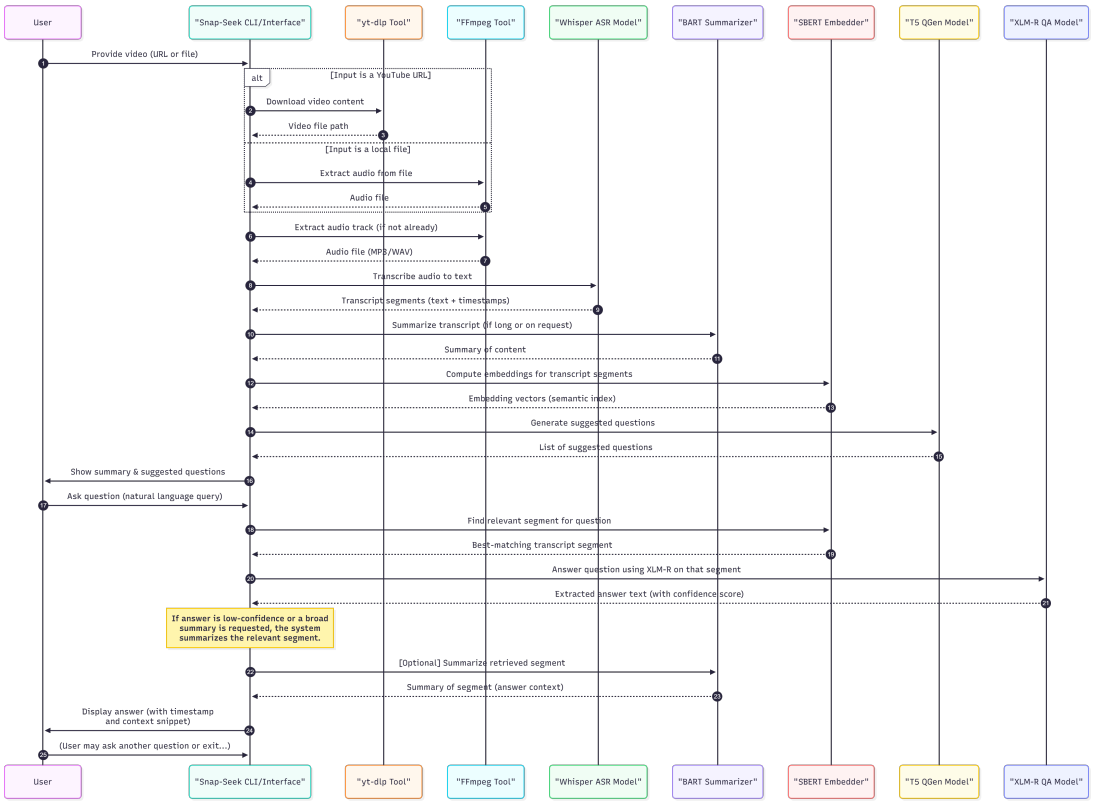

Figure 4.4: Sequence diagram detailing interactions between Snap-Seek components from video input to final output.

4.6 Low-Level Design

The low-level design focuses on the actual implementation of each respective module within the system, the data structures used within that module, and the internal workings of the module. Figure 4.5 illustrates that the Snap Seek system is structured in a modular manner with independent modules including: the ASR Module, providing the speech-to-text conversion; the Summarization Module, providing the generation of succinct video summaries; and the Semantic Retrieval Module, providing the ability to answer user

questions without needing to ask the user or search through files or other sources. All these modules are designed to operate separately from one another and work together as a team to provide a cohesive data flow, a simpler maintenance plan, and a more user-friendly system than would be available from any other means.

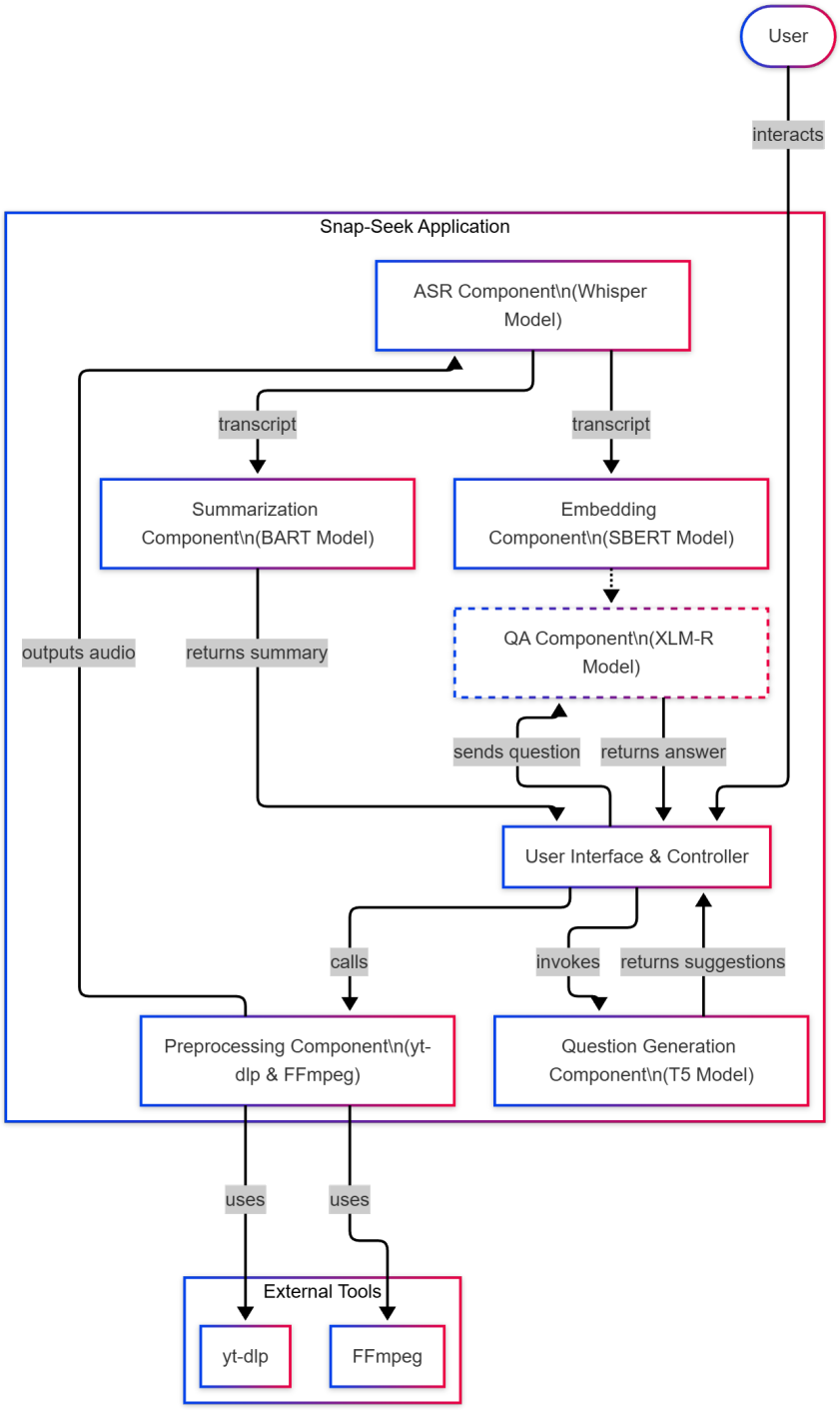

Figure 4.5: Component diagram of Snap-Seek illustrating software modules and internal responsibilities.

4.6.1 Module-wise Low-Level Design

Transcription Module

1. **Inputs:** Audio from video files or YouTube links.
2. **Process:** Automate Speech Recognition (ASR) with Whisper [1] breaking each spoken segment down to a Transcript of historical events and a time-stamped reference. Whisper also works with multilingual audio and noisy audio, utilizing preprocessing techniques to clean and prepare input audio for ASR.
3. **Outputs:** A structured listing of segments {start time, end time, text} which allows the downstream section (87.7) to perform semantic processing and summarisation.

Summarization Module

1. **Process:** Break up long transcripts into smaller manageable segments for easier memory consumption and increased efficiency of the model.
2. Each segment will have been summarised by BART [2] to produce a single coherent summary of that segment.
3. To create a final summary, hierarchical summaries are combined together to create a final single summary of all segments while keeping all relevant content, but removing redundant or less relevant information.
4. **Outputs:** Summary text has been created to highlight all key points of a video, making it easier to comprehend quickly.

Embedding Module

1. Each Transcript will be encoded into dense vectors using SBERT [7].
2. These embeddings will store the semantic information from each Transcript enabling the user to perform similarity looking for natural language questions.
3. All vectors will be stored in memory, which will allow for ultra-fast cosine similarity searches to quickly identify the segments that are most relevant during the QA process.

QA Module

1. Accept User questions in Natural Language.
2. Retrieve the most relevant segments of the Transcript using embedding similarity scores.

3. Use XLM-RoBERTa [4] to identify the exact span of the answer in the retrieved segments so that multilingual queries can be supported.

4. **Outputs:** Provide precise answers along with timestamps to allow users to quickly jump to the relevant part of the Video.

4.6.2 Database / Cache Design

No relational database is present; instead, the system relies primarily on caching technology in a structured way to best manage data, allowing for offline access:

1. **Transcripts Cache:** Transcripts Cache contains JSON-formatted transcript segments, with timestamps (start and end), speaker info (if available), and actual text from the transcript. This allows fast access and easy parsing of transcript information when doing things such as downstream summarization or QA.

2. **Embeddings Cache:** Embeddings Cache has each of the segment's embeddings serialized and stored using formats such as NumPy array or python pickle file. This allows the use of cached information for frequently accessed videos for very fast temporal and semantic searching as well as question-answer retrieval.

3. **Model Cache:** Model Cache contains pre-downloaded transformers, including Whisper, BART, SBERT and XLM-RoBERTa (among others), located in the .cache/transformers/ directory. All these models can now be cached, which makes future read transfers easier and reduces startup time when using offline availability.

4. **Temporary Cache: The** Temporary cache holds audio files that have been extracted from the video streams for processing. These will remain in temporary storage until processed and are automatically deleted afterwards. This cache also helps to ensure efficient storage management and eliminates the build-up of unnecessary large-sized media files.

Using this caching approach can improve performance, facilitate offline operation, and allow users to reuse preprocessed data.

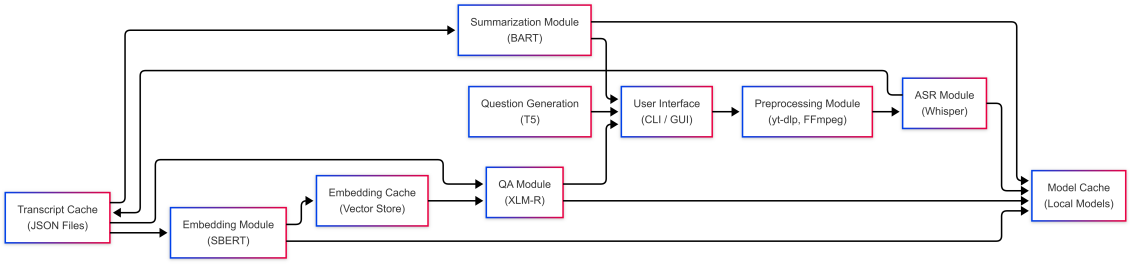

Figure 4.6: Low-Level System Diagram

4.7 User Interface Design and Usage

Snap-Seek has a simple yet effective graphical user interface (GUI) that enables users to interact easily with the program. When a user opens the app, they see a landing page (Figure 4.7) that describes what Snap-Seek can do. The next step is to go to the upload page (Figure 4.9) where the user inputs their video source. Once a user has submitted their video, Snap-Seek automatically transcribes and analyzes the video for the user. Users receive continuous feedback during the processing phase via progress indicators. Following the completion of the processing stage, results are presented in a clean, organized manner. On the right-hand side of the screen, users receive a basic overview of their video along with an option to ask questions about the content. Snap-Seek also provides users with a list of suggested questions based upon the user's video. Figure 4.8 shows an in-app guide which outlines all of the steps Snap-Seek performs on a video: extraction, automatic transcription (converting speech to written form), summary creation based upon the above two functions, generation of multiple questions, and answers via QA interaction, all conducted within an offline desktop setting.

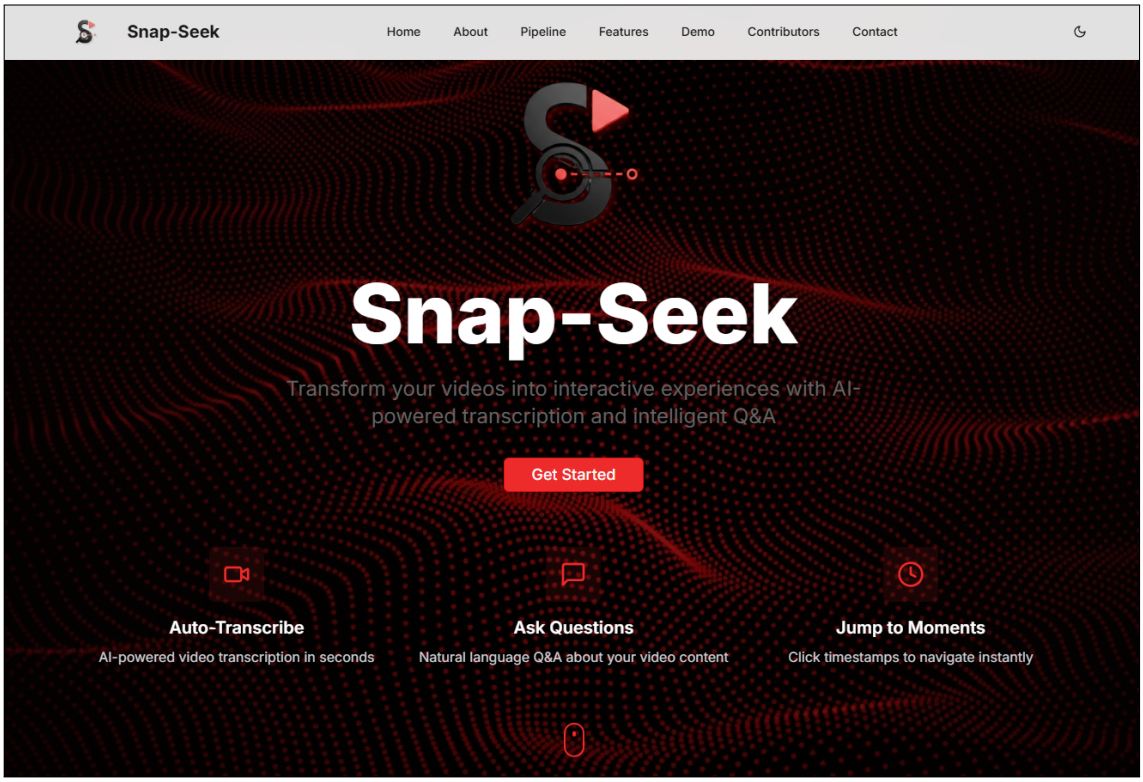

Figure 4.7: Snap-Seek main interface: a landing page.

The structure and layout of the external interface were designed with the user's experience being the most important consideration. The most prominent features of the system, video input, summary view, QA, and navigation, are available as dedicated sections, clearly

labelled, and can be accessed through dedicated buttons on every page. This allows for a seamless experience for users regardless of their background, enabling all users to be able to fully engage with Snap-Seek and successfully retrieve valuable information from lengthy videos.

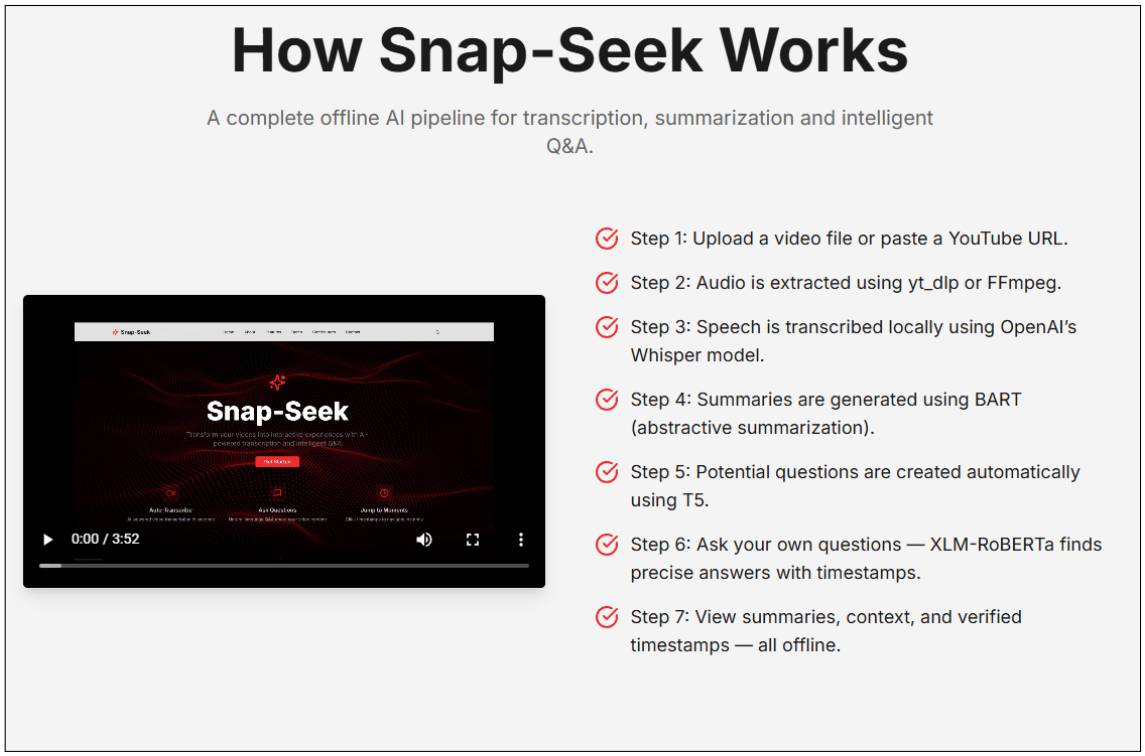

Figure 4.8: In-app "How Snap-Seek Works" guide.

User Interface

The Snap-Seek user interface (UI) is provided in both a command line interface (CLI), or graphical user interface (GUI), format. Both interfaces provide users with an easy-to-use interface to help users efficiently process videos, make sense of video insights without a lot of extra work. Key elements of the user interface include:

1. **Input Field:** An input field, which allows a user to enter a YouTube URL or select a local video for processing by upload. The program will validate the input against the list of supported formats.

2. **Process Initiation Button:** A process initiation button, which is a simple button or command. The process initiation button starts a workflow that includes audio extraction, transcription, summary generation, and semantic QA results retrieval.

3. **Display Area:** A results display area. It is the place where a user can see all the results of the processes performed on a video:

- **Transcript Summary:** The transcript summary provides a concise text representation of the entire transcript of a video.
- **Suggested Questions:** These are optional questions generated using the contents of the video.
- **Answer + Context + Timestamps:** Semantically retrieved answers include a context for each answer, as well as clickable timestamps that can be used for quick navigation to the point in the video where the answers were retrieved.

4. **Exit and Reset Controls:** Options to safely exit the app or reset the UI between new video processing jobs.

This UI Design allows both the tech-savvy and not-so-tech-savvy to access it effortlessly and gives them an intuitive workflow for interacting with video data.

Tool Interfaces

In order to process video files, as well as understand the language, and perform model inference (prediction), you will need a variety of tools and libraries that are available externally. These tools enable the automated video processing workflow. Here are the external tools that the system uses:

1. **`yt-dlp`**: Used to download video content from the internet (YouTube, etc.) to your local computer.
2. **FFmpeg**: Allows for audio extraction from video files. FFmpeg converts the video files into audio files that are appropriate for use in transcribing the audio.
3. **Hugging Face Transformers**: A library for loading pre-trained models such as Whisper, BART, SBERT, and XLM-RoBERTa. These models provide transcription and summary of video/audio, embed them in semantic vectors (with Hugging Face, a vector is created when video/audio is processed), and answer questions about the video/audio content.
4. **`langdetect`**: Automatically determines the language of the transcript so that it can be processed with the appropriate model for each task and allows for multilingual support.
5. **`deep-translator`**: The user can use their own language to process/ query non-English content by translating it into their preferred language.

Several tools provide system interfaces that allow the system to process video content quickly, support multiple languages, and provide accurate semantic responses, and there

will be as much offline capability as is possible. Collectively, these Interfaces build a pipeline that converts the raw video into a structured knowledge base that can be queried without any external API calls. Through open-source libraries and pre-trained models, Snap-Seek can provide enterprise-level video analysis capabilities via a desktop application. The architecture supports Scalability, which will allow enhancements in the future such as More languages, improved accuracy and compatibility with additional video formats while continuing to follow the principle of offline first.

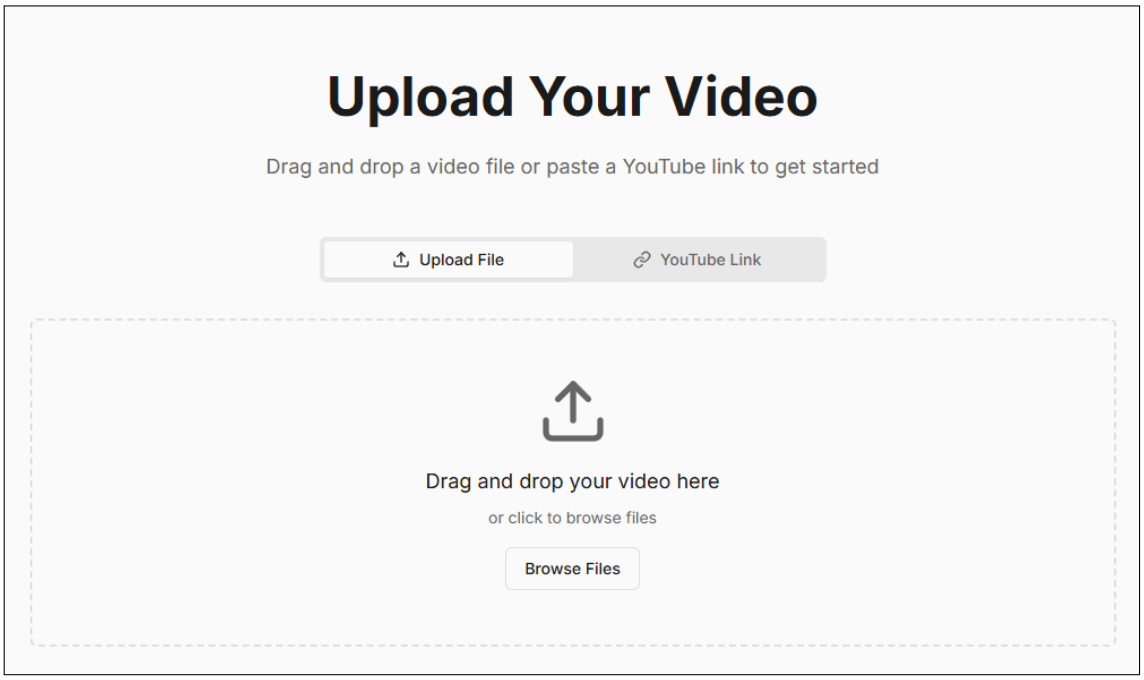

Figure 4.9: Video upload interface: users can drag-and-drop a video or paste a YouTube link.

4.8 Summary

This chapter presented the complete architectural design of Snap-Seek, providing a high-level workflow followed by a modular architecture along with each level of the design constraints and methodologies. Each component of the System is presented at a low level in the following order: Transcription, Summarization, Embedding and QA, along with caching methods and external Interfaces. The design decisions presented here provide a foundation to implement the System discussed in the following chapter.

Chapter 5

System Implementation

5.1 Introduction

This chapter gives an overview of the way Snap-Seek has been implemented. In addition to discussing how the modules fit together, how they communicate with one another, and the tools and technologies that were used to create Snap-Seek, this section will provide insight on development environments, how the artificial intelligence operates, and finally the security considerations involved with the project.

5.2 System Architecture Implementation

The architecture has been implemented using a modular approach to maintain separation of concerns while maintaining the ability to work offline. All of the modules communicate with one another using a "standardized" data structure. The standard data structures for Snap-Seek are JSON, TF-IDF embeddings, and Transformer Model Outputs. Below are a list of the modules implemented:

1. **Input Handler:** Accepts YouTube URLs and Local File Paths.
2. **Downloader & Extractor:** Uses `yt-dlp` [9] to download videos; uses FFmpeg [10] to extract the audio.
3. **Transcription Engine:** converts spoken words to text via Whisper ASR [1].
4. **Summarization Engine:** BART [2] summarizes the transcripts.
5. **Embedding Generator:** SBERT [7] creates vector embeddings of transcripts.
6. **Retrieval Engine:** Searches the generated vector embeddings for segments most relevant to a given query.
7. **QA Module:** extracts answers from context using XLM-R [4].

8. **Question Generation:** Generates questions from context using T5 [3].
9. **Interface Layer:** provides Command Line Interface and Graphical User Interface.

5.3 Functionality of System Components

1. Input Handler

The Input Handler System Component validates and receives information from the user input. When receiving information from the user, the Input Handler component first checks whether the information received is a URL or Local Folder/File Path and then verifies that the information received is a file type supported by our Application.

2. Video Downloader

The Video Downloader Module makes use of the `yt-dlp` Library for the retrieval of the most appropriate and highest quality Video and Audio streams from the provided URL. The Video Downloader Module is designed with error handling capabilities for invalid URLs and network-related issues as well as includes Automatic Retry for the Download of the content.

3. Audio Extraction

After a video has been downloaded, the Audio Extraction module extracts audio with the help of FFmpeg and creates an audio file (WAV format). By normalizing the sample rate to a common frequency and converting the extracted audio to a mono-channel format, using this process results in smaller audio file sizes and makes it easier to process later on. In addition, optional noise reduction may be used during the extraction process to improve the accuracy of speech recognition.

4. Transcription Engine

Using OpenAI's Whisper ASR model, the Transcription Engine converts audio signals into written form (text). The Transcription Engine has multiple language capabilities, meaning it can transcribe many languages; it also segments each transcribed audio file so that users will find it easy to navigate through a video file by timestamped segments. Example output format:

```
[
  { "start": 0.0, "end": 4.1, "text": "..." },
  ...
]
```

5. Summarization Engine

The BART summarization engine creates a summary of transcripts that is brief and concise. For long transcripts, the engine breaks the transcript down into smaller sections. Each section of the transcript is then summarized. The summary generated for each section of the transcript is then combined into a single global summary of the transcript, maintaining the key insights and main ideas from the transcript.

6. Semantic Embedding Generator

The SBERT semantic embedding generator accepts the segments of a transcript as an input, represents them as high-dimensional vector embeddings with either a MiniLM or XLM-R, and is used to compute how semantically similar two pieces of text are to each other to allow the system access to all the embedding data on one screen to quickly compare them and find matches.

7. Retrieval Engine

The Retrieval Engine provides a method to find relevant transcript segments and give the user context for how to answer their question. It also provides an easy way to calculate the cosine similarity between the embeddings of the user's question and all the transcript segments. After determining which segments are most similar (the "top-k") to the user's question, the Retrieval Engine builds the context of those segments together for the QA Module to provide accurate, meaningful responses based on the user's specific question.

8. QA Module

The QA Module, which uses the XLM-RoBERTa model to extract an answer span from a retrieved context, can produce an answer span with a corresponding timestamp that the user can click on to quickly navigate to the relevant portion of a video for better usability and learning efficiencies.

9. Question Generation Module

The Question Generation Module uses the T5 transformer to generate a variety of relevant questions based on transcript content. Generated questions will go through a grammatical evaluation to ensure correct grammar, a filter to remove duplicate/semtic type questions, and will be ranked according to their level of relevance. This module allows users to receive suggested questions to help them study/review and learn interactively.

10. User Interface

The User Interface (CLI or GUI) is the way that users will interact with the system in an efficient manner. Users will be able to enter a video either by providing a video URL or a local file; the UI will allow them to see transcript summary, participate in Q/A interactions, answer the suggested questions, and receive real-time status updates. The UI allows you to exit/ reset the system or to use all of its available features easily.

5.4 Communication Between Components

The system also provides structured methods of communication between the modules to allow for efficient and reliable data exchange, supporting real-time processing (both online and offline):

1. **JSON transcripts:** The transcription outputs produced by the system are stored as timestamped JSON segments. These segments are structured with start and end times of the recognized text, allowing modules that perform summarization, semantic retrieval and QA processes to work off of the same structured data without having to parse the raw audio each time that segment is needed for processing.
2. **NumPy arrays / pickle files:** After the creation of the SBERT or Transformer embeddings, they are serialized into NumPy arrays or Pickle files. This allows for the ability to do a fast cosine similarity search when answering questions about a video and allows for the ability to not compute a given video multiple times.
3. **Python dictionaries:** A dictionary is used to pass intermediate data such as meta-data (video ID/segment length/processing state) as well as a module-specific parameter between modules (i.e. implementation). Using a dictionary to pass the intermediate data provides both flexibility and a way to keep the communication between modules concise and easy to read for humans (i.e. there's no chance of error).
4. **Torch tensors:** All of the transformer calculations (BART Summarization, XLM-RQA) are performed using Torch tensors. Torch tensors can efficiently perform vector operations as well as be accelerated using GPU, resulting in quick response time and scaling for larger video files and multiple languages.

Data flows follow this pipeline:

Video Input $\rightarrow$ **Audio Extractor** $\rightarrow$ **Whisper ASR** $\rightarrow$ **Summarizer**

Summarizer $\rightarrow$ **Embedding Generator** $\rightarrow$ **Retrieval Engine** $\rightarrow$ **QA Engine**

5.5 Tools and Technologies Used

Table 5.1 lists all tools, libraries, and technology that are part of the SnapSeek technology stack and explains what they do within the Snap seek Technology Stack. The entire stack was selected to meet the needs for offline processing, accuracy, and performance.

Table 5.1: Tools, Libraries, and Technologies in Snap-Seek

Tool / Technology	**Purpose**
Python 3.x	Primary programming language
PyTorch	Backend for all transformer models
Transformers [8]	Loading Whisper, BART, T5, SBERT, XLM-R
Whisper [1]	Automatic speech recognition
BART [2]	Abstractive summarization
Sentence-BERT [7]	Embedding and semantic retrieval
XLM-RoBERTa [4]	Extractive question answering
T5 [3]	Question generation
yt-dlp [9]	Video download
FFmpeg [10]	Audio extraction and conversion
langdetect [5]	Language detection
deep-translator [6]	Translation support
JSON / Pickle / NumPy	Local caching and storage

5.6 Development Environment

The development environment was set up on a PC running Windows 10 (with an Intel i7 processor and potentially an NVidia Graphics Processing Unit (GPU)) using the programming language Python v3.10, Microsoft Visual Studio Code as the development tool along with additional libraries (e.g. PyTorch, Transformers, NumPy and Pandas) to provide core functionalities of the system. While CUDA v11.8 was installed in the environment, CUDA is only activated in the presence of supported hardware(s), which will provide significantly enhanced performance during the model inference phase. To manage dependencies, the development team used virtual environments (via pip) to isolate the project dependencies from the dependencies of the base system .

Table 5.2: System Development and Testing Environment

Component	**Specification**
Operating System	Windows 10
CPU	Intel i7
GPU	NVIDIA GTX/RTX series (optional)
Python Version	3.10+
IDE	VS Code
Runtime Libraries	PyTorch, Transformers, NumPy, Pandas

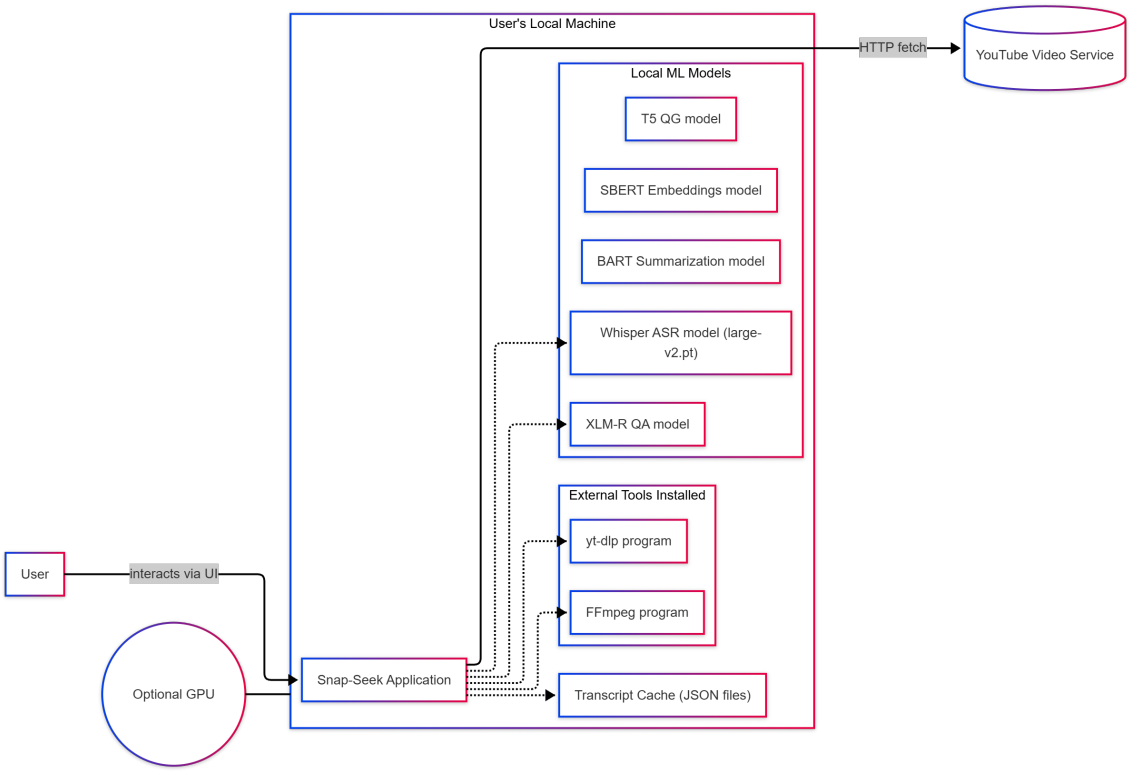

Figure 5.1: Deployment diagram showing Snap-Seek's local installation with offline models and external tools.

5.7 Processing Logic and Algorithms

1. Transcription Algorithm

The transcript generation begins by loading the trained Whisper model and preparing the audio using techniques such as resampling and normalization. To further improve transcript accuracy during the recognition process, noise reduction techniques may also be applied, and the audio channels can be converted into one channel (mono).

2. Summarization Algorithm

To obtain a summary of the transcript generated from speech recognition, the transcript will be divided into sub-sections of manageable size. The sub-sections will be summarized separately using the BART model. After all sub-sections have been summarized, the overall transcript will be created.

3. Semantic Search Algorithm

Semantic search starts by generating an embedding for the user's query. This embedding is compared with precomputed segment embeddings using cosine similarity, defined as

$$\text{similarity}(q, s_i) = \frac{e_q \cdot e_{s_i}}{\|e_q\| \|e_{s_i}\|}.$$

The system then retrieves the top-k most relevant segments based on similarity scores.

4. QA Algorithm

The QA process begins by feeding the context gathered from a retrieval process and a user query to an XLM-R model to calculate start-end logits of what would be the best fit for the answer to that question (the answer span). The text that gets extracted is then timestamp mapped back into the transcriptions of the video.

5.8 Application Security

The system was designed so that no data would be sent or stored externally from your devices; everything is performed locally, guaranteeing user content will not leave their devices. The temporary files created during processing are deleted once the process is complete, and the system does not store or collect any type of personal information about users at any time.

5.9 Database / Cache Security

Snap-Seek does not use traditional databases; all cache security is kept through multiple mechanisms. All cached files remain on the end-users' devices, preventing access from outside the device and preventing any transfer of data out of the user's device to the external environment. All temporary files created during the processing step within Snap-Seek will be deleted after they are complete to prevent any unnecessary retention.

5.10 Graphical User Interface (GUI)

A simple GUI (or CLI) offers the following features:

1. You can select the video URL or the file from your computer that you want transcribed.
2. You can see how far along your authentication, summary, and QA are with progress indicators.

3. You can see a clear condensed transcript of the audio in the summary panel.
4. You can ask questions in natural language using the entry field.
5. You can see the answers to your questions, including the time stamps of when each answer was made to help you navigate quickly.
6. You can click buttons to generate possible questions from the transcript.
7. You can access the application controls by clicking either the reset or exit buttons.
8. You can see in real-time how long your recordings take and what actions are required for you to get your results by clicking on the action buttons.

Figure 5.2 illustrates the Transcript Dashboard interface. After transcribing the audio, the user can see the various time-stamped segments in the transcript, select a segment from any point in the transcript, and start contextual QA or a summary based on that specific segment.

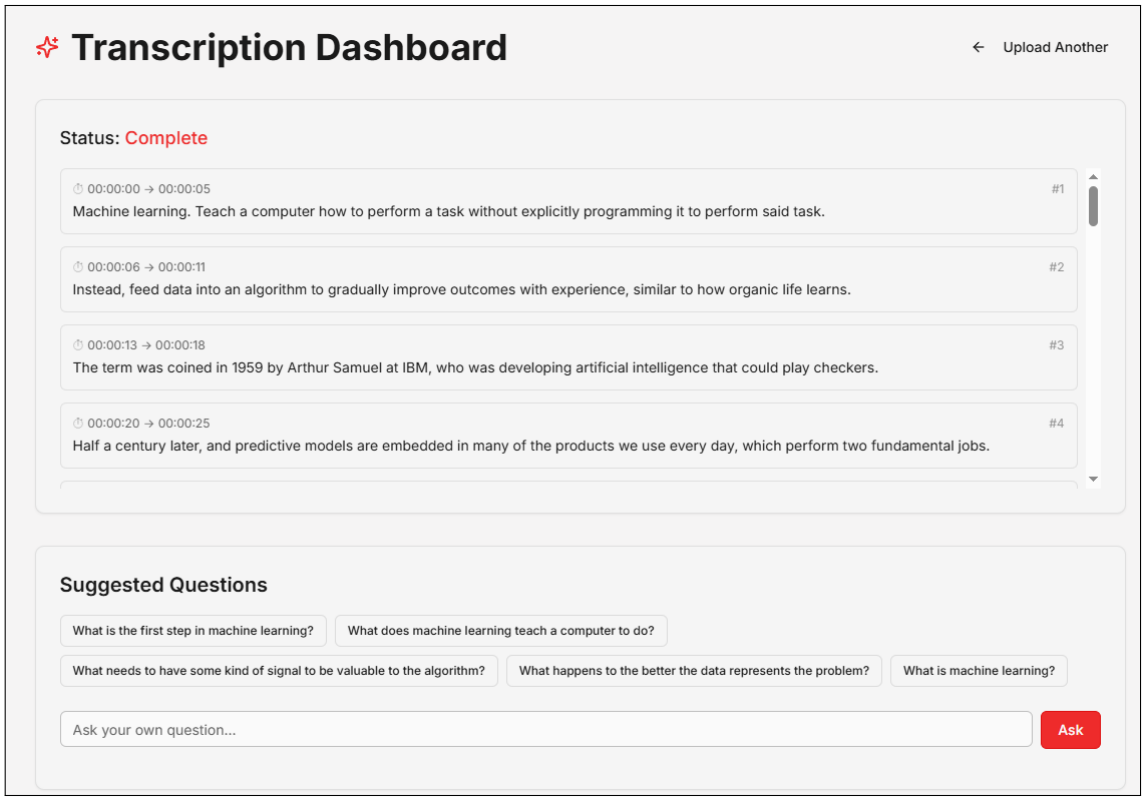

Figure 5.2: Transcription dashboard showing segmented video transcript results with timestamps and playback.

The user interface of the QA (question answering) module is intuitive. After typing a question, the user will receive answers to that question that include the timestamp(s) of when each answer was made, as shown in Figure 5.3.

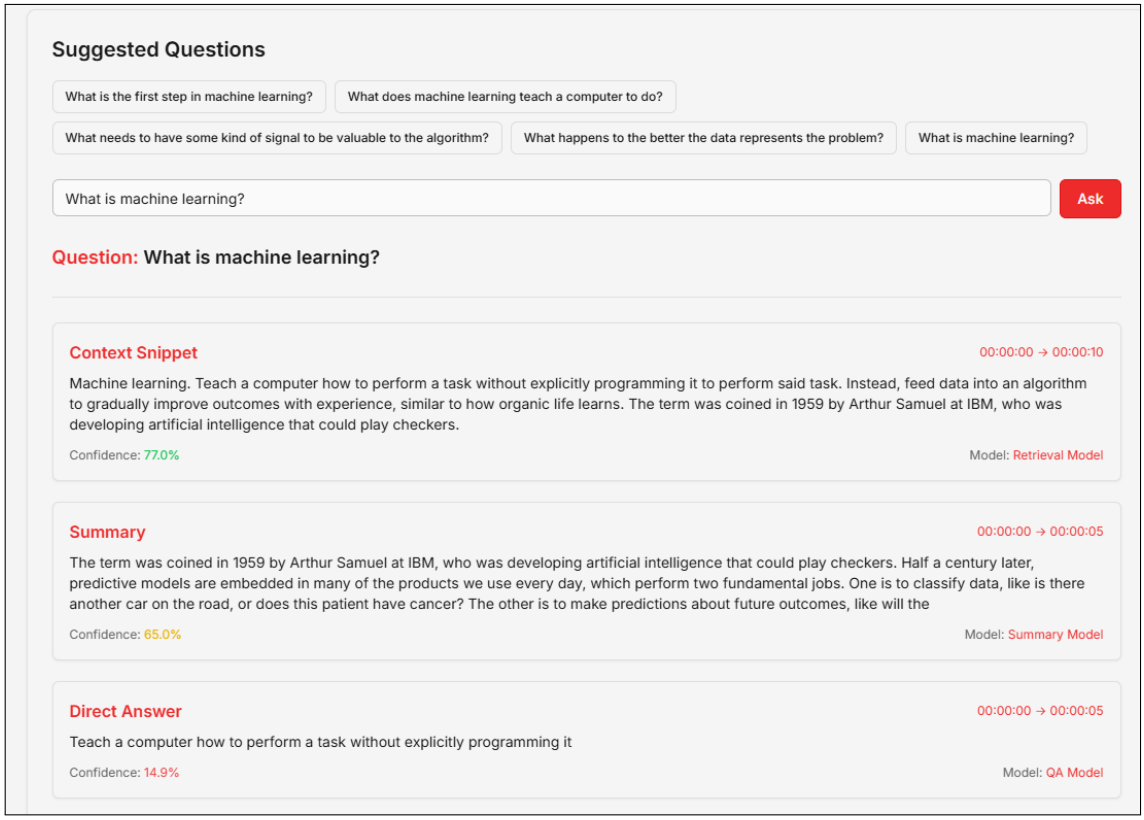

Figure 5.3: QA interface where users enter natural-language questions and receive answers with timestamped context.

Once the user submits a video link or uploads a file, the system will begin transcribing it and generating a transcript. Figure 5.4 illustrates the start of the transcription process

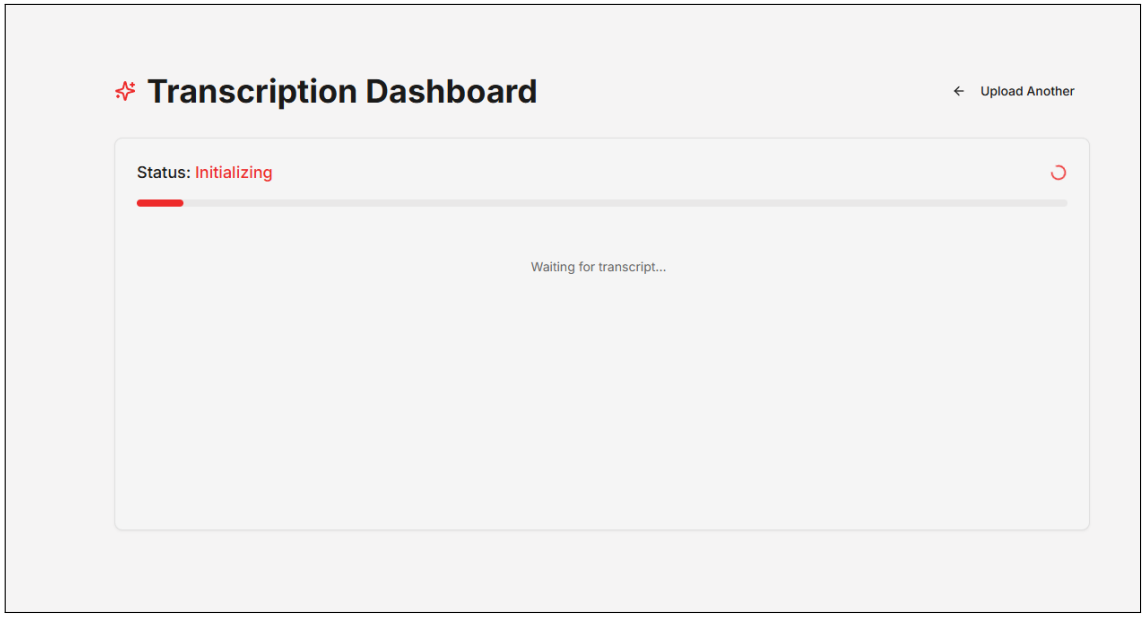

Figure 5.4: Initial video upload screen showing input confirmation and transcription initialization.

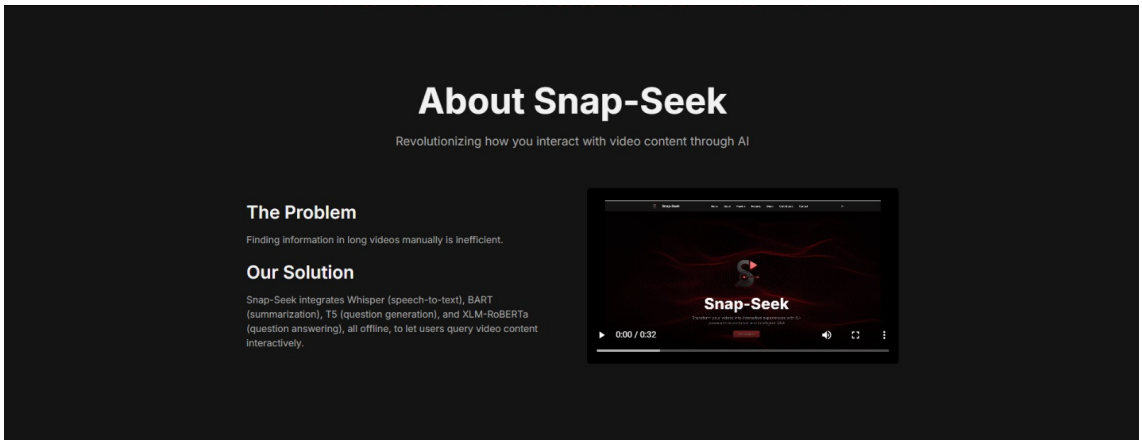

Figure 5.5: About Snap-Seek

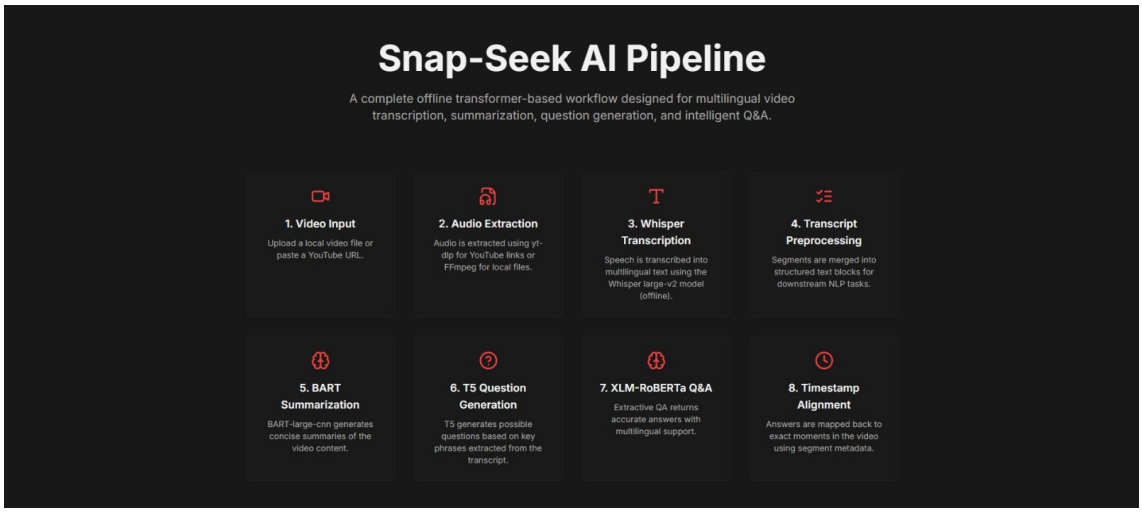

Figure 5.6: Snap-Seek Pipeline

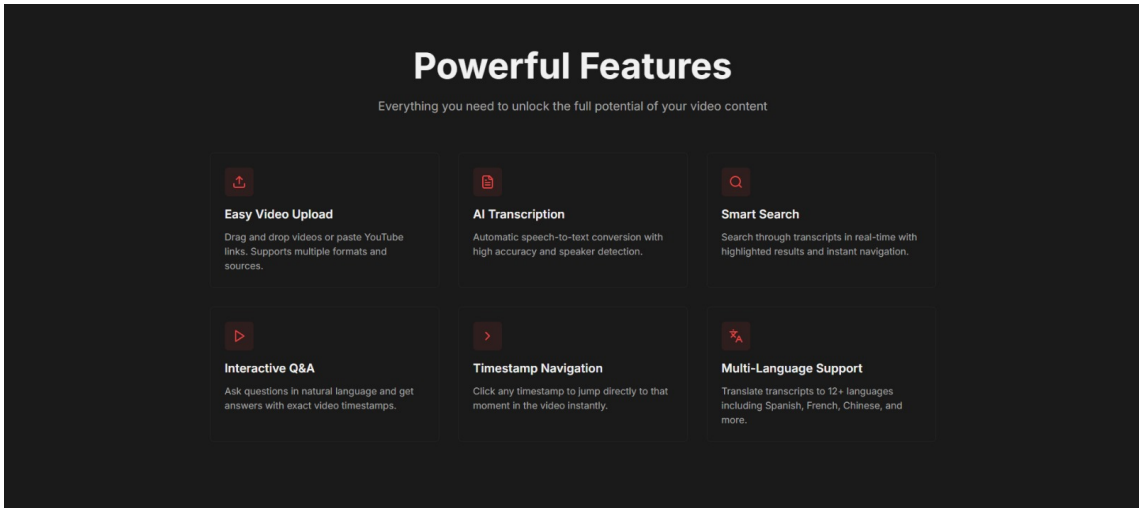

Figure 5.7: Features of Snap-Seek

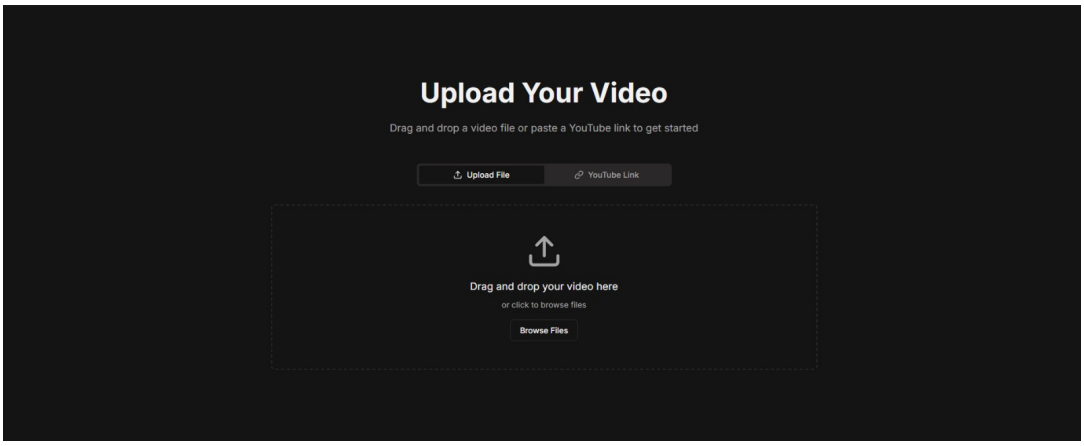

Figure 5.8: Working page

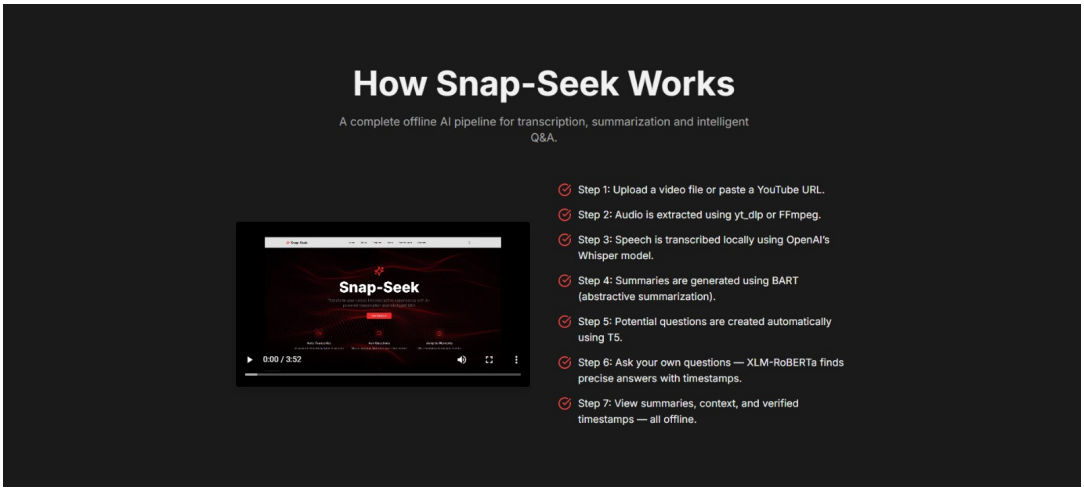

Figure 5.9: Step-by-Step Guidance

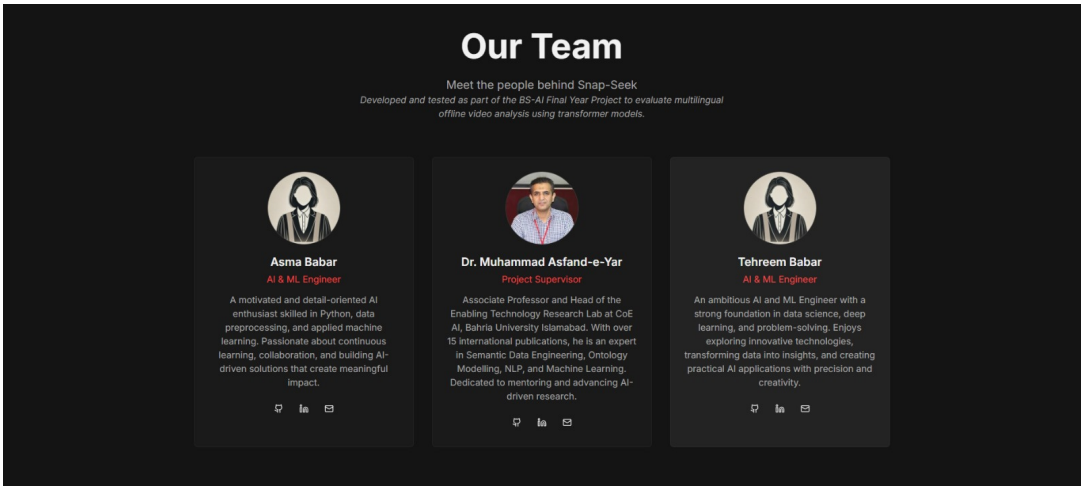

Figure 5.10: Snap-Seek Team

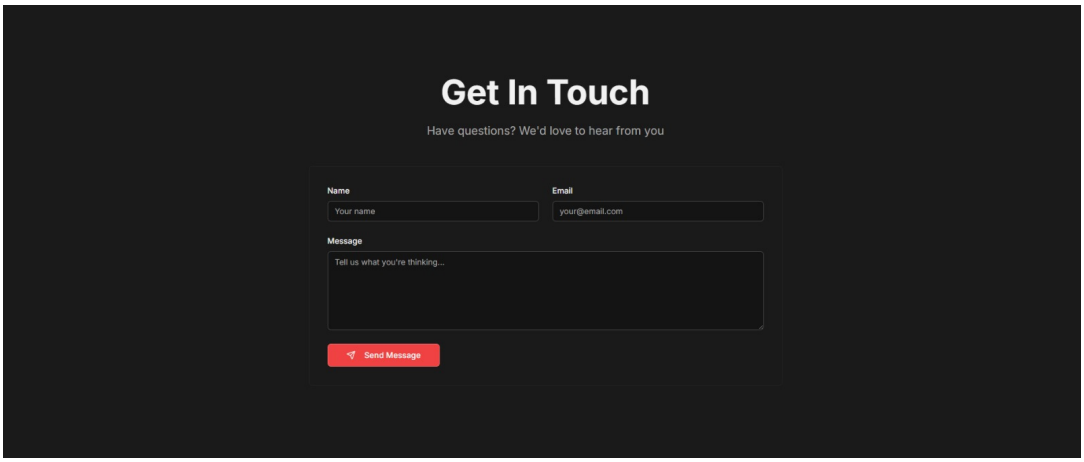

Figure 5.11: Contact Us

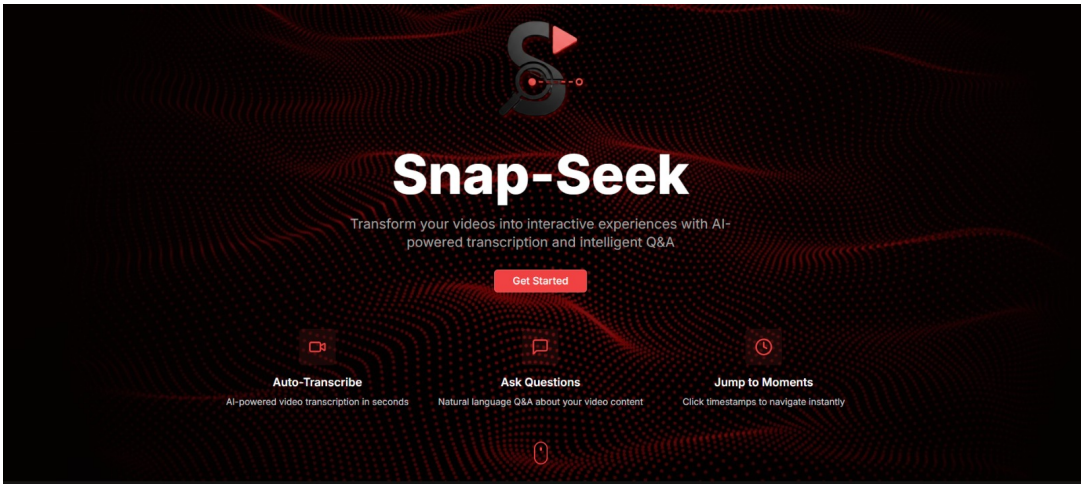

Figure 5.12: Home Page

5.11 Summary

In this chapter, we discuss how we built the Snap-Seek system using a modular offline architecture to provide end-to-end functionality, such as video processing, transcription, summarization, semantic search, and question answering. We also discuss in detail the features of each software component and the underlying technologies used for these components, as well as the algorithms that power the overall pipeline. In addition, we describe how we designed Snap-Seek to operate securely when used in a disconnected environment and the GUI/CLI created for users to easily interact with Snap-Seek.

Chapter 6

System Testing and Evaluation

6.1 Introduction

Testing of Snap-Seek is an essential step in validating whether Snap-Seek meets its functional requirements and non-functional requirements. This chapter provides a detailed overview of the strategies employed for testing, methodologies, and results from the testing performed during the evaluation. Unit testing, integration testing, system testing, and usability testing were performed to verify that Snap-Seek can successfully and reliably operate in real-world user scenarios. The following categories of testing were performed for Snap-Seek:

1. Graphical user interface testing
2. Usability testing
3. Compatibility testing
4. Exception and robustness testing
5. Load and performance testing
6. Security testing
7. Structured test cases

6.2 Graphical User Interface Testing

Graphical user interface (GUI) testing is focused on ensuring that the system's user interface is easy to navigate, responsive, and thus user-friendly. During GUI testing, we looked at the layout of each input field and button, making sure that items displayed correctly on the screen, the transition between stages of processing was seamless, summaries and answers

displayed correctly, and that suggested questions and answers remained visible. As shown in Table 6.1

Participant	Experience	Suggestions for Improvement
Mr Ñasir Ali	Found the dashboard simple and intuitive. Uploading videos and navigating to features was smooth.	Suggested adding theme options (light/dark) for visual comfort.
Mr Rizwan Saeed	Upload interaction was smooth, but he felt uncertain after the upload completed.	Recommended adding a clear "Start Analysis" button after upload.
Mr Javaid Wahab	Results such as scenes and frames were displayed correctly but lacked descriptive context.	Suggested adding small tooltips or hints explaining each output.
Mr Ahmed Raza	Navigation was easy and elements were accessible. Text sizes for results were small.	Suggested increasing font size and improving emphasis for key outputs.

Table 6.1: User Feedback from GUI Testing

Test Observations

The interface is certainly displaying improvement, is easy to use, and is very responsive; the majority of the responses to the questions and answers were displayed within 2-3 seconds. In conclusion, the evidence clearly shows that the system is running efficiently and effectively.

6.3 Usability Testing

Usability testing on Snap-Seek was done with existing users of video-based learning. Participants were asked to upload or link a video, summarize it, ask at least five questions, and then create an additional question using the suggested questions feature in Snap-Seek. As shown in Table 6.2

Framework Aspect	Evaluation	Rating (1–5)
Learnability	Users quickly understood uploading and analysis steps.	4
Efficiency	Some delays were noticed during heavy video processing.	3
Memorability	Users remembered the steps easily after initial use.	4
Errors	Most errors were handled with clear messages.	4
Satisfaction	Overall satisfaction was positive.	3

Table 6.2: Usability Testing

Feedback Summary

Feedback was positive overall, with users indicating that they found the summaries of their videos generated to be accurate and beneficial and perceived the post-experience suggested questions as providing a way to investigate the content in greater detail. One particularly positive aspect of Snap-Seek documented in the feedback was the offline processing ability. Users recognized that this addressed a growing concern about protecting their privacy when using video for education.

Components of the interface evaluated include the transcription dashboard (Figure 5.2), the QA answer panel (Figure 5.3) and feedback on upload status (Figure 5.4). Usability testing confirmed that Snap-Seek provides users with a functional and intuitive experience.

6.4 Compatibility Testing

The testing included various configurations of the system for multiple operating systems, including

1. Windows 10 with only the CPU as the processing unit.
2. Windows 11, which included both the CPU and GPU.
3. various mixed CPU/GPU systems operating Ubuntu 20.04 LTS.

These operating systems, along with the various hardware configurations, were chosen to provide a large cross-section of the environments in which users would typically be using the application and provide some understanding of how the various hardware components would perform under different operating conditions. As shown in Table 6.3

Device/Platform	Status	Comments
Windows – Chrome	Passed	Smooth performance and accurate interface display.
Google Colab	Passed	GPU acceleration supported fast processing.
Jupyter Notebook	Passed	Frame extraction and video display worked perfectly.
VS Code	Passed	Integrated well with Python libraries and video tools.

Table 6.3: Compatibility Testing Results

Results

All of the core features of the software (transcription, summarisation, question generation and answer retrieval) functioned similarly across all of the platforms tested. The systems with GPU acceleration exhibited significant performance improvements, showing nearly a

6× increase in transcription speed and smoother processing of longer tasks. CPU-based systems with only CPU resources performed well on short to moderate bandwidth videos but began to exhibit substantially slower processing times on videos longer than 45 minutes.

6.5 Exception and Robustness Testing

The exception testing process checks that the system performs well and provides a good user experience when it encounters unexpected input or errors in processing. It ensures that the pipeline remains stable, user-friendly, and robust to a wide variety of real-world error conditions, As shown in Table 6.4

Test Scenarios

1. YouTube URL is unreachable or invalid.
2. Local file path is incorrect or not provided at all.
3. Video/Audio files that have been partially downloaded or corrupted.
4. Video codecs that are unsupported or obscure and cannot be decoded by FFmpeg.
5. Video files that contain minimal speech and have long periods of silence between sections.
6. User questions that are vague, incomplete, and/or tend to be too broad.

Test Scenario	Status	Comments
Uploading unsupported video formats	Passed	Displayed an error message for invalid file types.
Video with missing frames	Passed	Skipped unreadable frames and notified user.
Processing long videos (timeout)	Failed	System froze on very long videos; optimization needed.
Invalid input in search	Passed	Clear error message shown; system prevented crash.
Missing file permissions	Passed	Displayed permission error and stopped safely.

Table 6.4: Exception Handling Test Results

Outcome

1. Rather than generic "failure" messages, the system provided robust and clear error messages for every invalid input.

2. If a video file had no significant amount of speech, then the system returned a fallback response (i.e. "No useful transcript available") which prevented unnecessary processing of the file.

3. For ambiguous or incomplete queries made by a user, the QA module would always return answers based on a broader context rather than just providing an empty response.

4. The pipeline remained functional throughout all testing and did not crash even when it encountered corrupted files, unsupported formats or missing data during its operation.

5. Every module returned an appropriate error state if it failed; therefore the errors did not propagate through the system, maintaining the robustness of the overall system.

6.6 Load and Performance Testing

The load testing identified the ability of the system to support the processing of multiple user queries over a prolonged period of time as well as to process a full length of video without experiencing any substantial performance degradation due to heavy load/rate usage patterns, including rapid user repeated query submission.

Transcription Load Testing

The first hour of the lecture was processed properly using both a GPU-enabled device and a CPU-only device. Using the GPU-enabled environment drastically increased the performance of the system by reducing the time that it took to complete the transcription of the hour long lecture from approximately 25 minutes using the CPU to 4 minutes using the GPU. During the testing process, memory utilization remained relatively constant and there were no system freezes or interruptions, validating the pipeline's ability to handle long form content in a reliable manner.

QA Load Testing

The system successfully answered 50 consecutive user requests with no noticeable decrease in overall performance. The SBERT retrieval module was also efficient during the test, with most queries taking less than 0.5 seconds to retrieve the appropriate embedding. The system was able to consistently respond accurately to each of the 50 queries and had no issues meeting the user's needs over an extended interactive session on the system. Further, memory usage and CPU usage were within normal operating limits during the QA Load Test, indicating that the system effectively mitigated the need for excessive resource allocation or use during the time period of the load test. No issues with timeouts

or query failures occurred during the load test. Overall, the results demonstrate that the system is capable of supporting real-time and repeated interactions with users in practical applications of the system. As shown in Table 6.5

Load Level	Number of Videos	Processing Time	System Response
Low Load	1–2 videos	2–4 seconds per frame extraction	Smooth performance
Moderate Load	3–5 videos	4–7 seconds per frame extraction	Minor delay
High Load	6–10 videos	7–12 seconds per frame extraction	Noticeable lag
Peak Load	10+ videos	12–20 seconds per frame extraction	Lag and occasional freezing

Table 6.5: Load Testing Results

6.7 Security Testing

While Snap-Seek operates completely offline, security testing ensures the integrity of the data and model stored with each user locally. As shown in Table 6.6

Key Areas Tested

1. **Local storage safety:** All temporary files are deleted after use.
2. **Model integrity:** To confirm that the Models used were not tampered with or are missing.
3. **User privacy:** All video and transcript files remain on the user's computer.
4. **External connectivity:** No background API calls or Internet connections occur during processing.
5. **Security Vulnerabilities:** No security vulnerabilities have been found in Offline Mode.

Security Aspect	Details
File Security	Videos are temporarily stored and automatically deleted after analysis.
Unauthorized Access Prevention	Restricted access prevents exposure to sensitive system files.
Data Protection	All processing occurs locally, ensuring user privacy.
Script Safety	Input validation prevent malicious injections.

Table 6.6: Security Testing Summary

6.8 Test Cases

To ensure Snap-Seek meets all functional requirements and handles edge cases robustly, a suite of structured test cases was developed. The test design covers each stage of the pipeline from video ingestion and transcription to summarization and Q&A. Table 6.7 outlines key test cases, including their input conditions, expected outcomes, and whether the system passed each test.

Test Case Table

Table 6.7: Key Test Cases Conducted on Snap-Seek

Test Case	**Input**	**Expected Output**	**Status**
Upload Video/URL	Local file or YouTube URL	Video uploaded or loaded successfully	Pass
Generate Transcript	Video audio input	Timestamped transcript generated with high accuracy	Pass
Summarize Content	Transcript text	Concise summary generated via BART model	Pass
Ask Question (Q&A)	Natural language question	Accurate answer extracted with timestamp	Pass
Semantic Search	Search query keyword	Relevant transcript segments retrieved via embeddings	Pass
Navigate by Timestamp	Click timestamp link	Video player jumps to exact timestamp position	Pass
View/Download Results	Select PDF/JSON export	Results file generated and downloaded successfully	Pass
Suggested Questions	Request suggestions	List of relevant questions generated via T5 model	Pass
Multilingual Support	English query on Spanish video	Accurate cross-lingual answer provided	Pass
Error Handling	Invalid YouTube URL	System displays clear validation error message	Pass
Offline Operation	Local file (No Internet)	Full processing (transcription, QA) works locally	Pass
Performance Load	2-hour lecture video	Processing completes within acceptable time (<30 mins)	Pass
Reset Session	"Reset" button click	Current session data clears; returns to upload screen	Pass
Model Caching	First-time system run	AI models downloaded and cached for future offline use	Pass
Progress Indication	Long video processing	Real-time progress bar shows percentage/time remaining	Pass

Example Test Case

To illustrate the testing process, consider the *Invalid URL Test* scenario. In this test, a user provides an incorrectly formatted video link as input. The expected behavior is that Snap-Seek will detect the invalid input and display an appropriate error message. During testing, the system successfully identified the malformed URL and responded with a clear error notification (e.g., **Invalid link: please check the URL**). This outcome matches the expected result, confirming that the system gracefully handles erroneous inputs. The test was therefore marked as **Pass**.

Test Case 1: Upload Video/URL This test case was completed successfully; it verified the ability of the system to accept a video file on a user's local computer or any video URL from YouTube. Additionally, the system performed all validations and was able to extract and initialize the uploaded content. As shown in Table 6.8

Table 6.8: Test Case 1: Upload Video/URL

Test Case ID	SNAP-TC-001
Function	Upload video file or provide YouTube URL
Initial State	Application launched, upload interface displayed
Input	Select local video file or paste YouTube URL
Expected Output	Video successfully uploaded/loaded, validation message displayed
Actual Output	Video loaded successfully, processing initiated
Status	Pass

Test Case 2: Generate Transcript This test case was completed successfully; it validated whether the system properly converted the audio of the uploaded video to timestamped text. Also, the transcription quality was validated to ensure it aligned with the timestamps of when the audio was spoken in the video. As Shown in Table 6.9

Table 6.9: Test Case 2: Generate Transcript

Test Case ID	SNAP-TC-002
Function	Automatic speech-to-text transcription
Initial State	Video uploaded successfully
Input	Initiate transcription process
Expected Output	Timestamped transcript generated and displayed
Actual Output	Transcript generated with accurate timestamps
Status	Pass

Test Case 3: Summarize Video Content This test case validated whether the system produced effective and appropriately structured summaries from the transcript provided from the previous test case's output. Additionally, the key topics and ideas present in the summaries were verified to be correct. As shown in Table 6.10

Table 6.10: Test Case 3: Summarize Video Content

Test Case ID	SNAP-TC-003
Function	Generate video summary using BART
Initial State	Transcript generated and available
Input	Click "Generate Summary" button
Expected Output	Concise summary displayed highlighting key points
Actual Output	Summary generated capturing main content
Status	Pass

Test Case 4: Ask Question (Q&A) This test case validated whether the system is able to accurately respond to users' questions, providing the proper timestamps from audio recordings of the content and also utilizing natural language to respond to users. As shown in Table 6.11

Table 6.11: Test Case 4: Ask Question (Q&A)

Test Case ID	SNAP-TC-004
Function	Natural language question answering
Initial State	Transcript and embeddings generated
Input	Type question: "What is the main topic?"
Expected Output	Accurate answer with timestamp and confidence score
Actual Output	Relevant answer extracted with correct timestamp
Status	Pass

Test Case 5: Semantic Search This test case validated the ability of the system to return relevant segments of a transcript even through the use of paraphrasing when performing semantic similarity searches. This test case validated that the user can return to an exact timestamp of a video provided by the user via the question and answer result. As shown in Table 6.12

Table 6.12: Test Case 5: Semantic Search

Test Case ID	SNAP-TC-005
Function	Semantic similarity search in transcript
Initial State	Embeddings generated from transcript
Input	Search query: "machine learning basics"
Expected Output	Relevant transcript segments retrieved using cosine similarity
Actual Output	Correct segments found even with paraphrased queries
Status	Pass

Test Case 6: Navigate by Timestamp This test case validated that the user can return to an exact timestamp of a video provided by the user via the question and answer result. As shown in Table 6.13

Table 6.13: Test Case 6: Navigate by Timestamp

Test Case ID	SNAP-TC-006
Function	Timestamp-based video navigation
Initial State	Answer displayed with timestamp
Input	Click timestamp link (e.g., "00:05:30")
Expected Output	Video player opens at specified timestamp
Actual Output	Video plays from exact timestamp position
Status	Pass

Test Case 7: View/Download Results This test case checks to see if processed output such as transcripts, summaries, and QA History can be viewed and saved in different formats (i.e., PDF, CSV, and TXT). As shown in Table 6.14

Table 6.14: Test Case 7: View/Download Results

Test Case ID	SNAP-TC-007
Function	Export results in multiple formats
Initial State	Processing complete, results displayed
Input	Select "Export as PDF" option
Expected Output	PDF file generated and downloaded
Actual Output	PDF exported successfully with all content
Status	Pass

Test Case 8: Suggested Questions Generation This test case verifies that the system is able to generate suggested questions based on the transcript so that the user can continue discovering more. As shown in Table 6.15

Table 6.15: Test Case 8: Suggested Questions Generation

Test Case ID	SNAP-TC-008
Function	Generate suggested questions using T5
Initial State	Transcript available
Input	Click "Generate Suggested Questions."
Expected Output	List of relevant questions displayed
Actual Output	3-5 relevant questions generated
Status	Pass

Test Case 9: Multilingual Support This test case determines the system's ability to accurately respond to user queries in multiple languages and display the correct response with the ability to detect and translate multiple languages. As shown in Table 6.16

Table 6.16: Test Case 9: Multilingual Support

Test Case ID	SNAP-TC-009
Function	Handle multilingual content and queries
Initial State	Spanish video uploaded
Input	Ask question in English about Spanish video
Expected Output	Accurate answer with translation support
Actual Output	Question answered correctly with language detection
Status	Pass

Test Case 10: Error Handling - Invalid Input This test case ensures that invalid input is handled gracefully by providing users with a clear, easy-to-understand error message and the ability to recover. As shown in Table 6.17

Table 6.17: Test Case 10: Error Handling - Invalid Input

Test Case ID	SNAP-TC-010
Function	Handle invalid input gracefully
Initial State	Upload interface active
Input	Provide invalid YouTube URL
Expected Output	Clear error message with guidance
Actual Output	“Invalid URL - Please check and try again”
Status	Pass

Test Case 11: Offline Operation This test case verifies that all video processing is fully functional while offline as long as the required models have been cached. As shown in Table 6.18

Table 6.18: Test Case 11: Offline Operation

Test Case ID	SNAP-TC-011
Function	Process videos without internet
Initial State	System offline, models cached
Input	Upload local video file
Expected Output	Full processing completed offline
Actual Output	Transcription, summarization, QA all work offline
Status	Pass

Test Case 12: Performance - Long Video This test case verifies the system’s processing speed and ability to continue functioning properly while processing long, 90-minute videos. As shown in Table 6.19

Table 6.19: Test Case 12: Performance—Long Video

Test Case ID	SNAP-TC-012
Function	Handle long videos efficiently
Initial State	2-hour lecture video uploaded
Input	Process complete video
Expected Output	All features work within acceptable time
Actual Output	Processing completed in under 30 minutes
Status	Pass

Test Case 13: Reset Session This test case verifies that the system can clear all session data and reset all data used in that session to its initial state. As shown in Table 6.20

Table 6.20: Test Case 13: Reset Session

Test Case ID	SNAP-TC-013
Function	Clear current session and reset
Initial State	Video processed, results displayed
Input	Click “Reset Session” button
Expected Output	Return to initial upload screen
Actual Output	All data cleared, upload interface shown
Status	Pass

Test Case 14: Model Caching Testing for model caching This test case confirms that after being used for the first time, your AI models are now stored locally so they can be reused Offline and processed quicker in the future. As shown in Table 6.21

Table 6.21: Test Case 14: Model Caching

Test Case ID	SNAP-TC-014
Function	Cache AI models for offline use
Initial State	First-time system use
Input	Process first video
Expected Output	Models cached locally for future use
Actual Output	Models downloaded and cached successfully
Status	Pass

Test Case 15: Progress Indication This test case validates that you can view live updates on the progress of video processing as you are processing a video, such as how much of the video is complete, and how much time will be required for completion, As shown in Table 6.22

Table 6.22: Test Case 15: Progress Indication

Test Case ID	SNAP-TC-015
Function	Display processing progress
Initial State	Video upload initiated
Input	Start transcription of 1-hour video
Expected Output	Progress bar with percentage and time estimate
Actual Output	Real-time progress updates displayed
Status	Pass

6.9 Summary

The chapter has been devoted to assessing Snap-Seek with GUI testing, usability testing, compatibility testing, exception handling, load testing, security testing, and structured testing. The tests performed demonstrate the Snap-Seek system's robustness, usability, accuracy, and efficiency, and that the Snap-Seek system continues to perform efficiently and consistently in different environments and to date, Snap-Seek has maintained a satisfactory level of performance and functionality, fulfilling all of the project's performance requirements and project objectives.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We have created Snap-Seek, an AI-based assistant that assists users by automatically transcribing their video content, summarizing that content, and answering users' questions about that content. Snap-Seek employs multiple (i.e., speech, summarization, question generation, and question answering) AI models in conjunction with an embedding-based retrieval mechanism to handle video content through an offline, unified system. Snap-Seek provides a solution for users to obtain information more efficiently when using multimedia videos. In addition, users can use natural language to ask questions quickly and easily about specific information in each video they view via Snap-Seek, thus allowing users greater interactivity than traditional viewing methods. Lastly, Snap-Seek's support of multiple languages extends its applicability for many other videos not in English, providing Amazon with a way to meet the increased demand for video access worldwide. Snap-Seek's offline capability ensures user privacy and provides for deployment in locations that do not have reliable internet connections or cloud services. Our major accomplishments include integrating our speech and language AI models, achieving high accuracy in transcribing speech to text and retrieving relevant answers, enabling users to interactively ask questions and verify their answers with timestamps, and offering a modular, extensible design that can be adapted for a variety of uses such as lectures, audiobooks, and corporate training. Snap-Seek combines advanced artificial intelligence (AI) models with essential system design guidelines to produce a practical example of the advantages offered by this technology. In the near future, video watching, which has historically been a passive, observational activity, will evolve into a dynamic and knowledge-based experience through Snap-Seek.

7.2 Future Work

Although Snap-Seek is a successful example of using offline AI video assistance, many opportunities to increase its capability and usability can be identified. For example, improved long video summary generation can encompass multiple-layered summaries by segmenting content according to topic/structure, producing an overview of each segment, and creating one overall summary of all segments combined. By using other AI models, e.g., PEGASUS or LongT5, the processing of very long video transcripts may also be done in a superior manner. Question generation could be improved by generating context-aware questions throughout the video (as opposed to only at the beginning) and/or by having multiple LLMs (large language models) used locally to generate dynamic and varied questions covering the entire video content. Generative question-answer pairs with explanations can allow the system to integrate and produce answers based on multiple segments of the same transcript and potentially utilize other LLMs that are specifically fine-tuned (e.g., LLaMA-2 or GPT-J) while having all answers remain grounded in the video itself to prevent 'hallucination.' Using visual information is an exciting new direction of growth. This business model will allow you to use Optical Character Recognition (OCR), object recognition, and image captioning technology to answer questions based on video frames, slides, or objects, resulting in a completely multimodal video assistant. With real-time/streaming processing capabilities, we will be able to transcribe and respond to questions posed during live online lectures, webinars, and meetings with a completely live, up-to-date transcription and response with low latency using Whisper's strong streaming ability. We plan to improve the user interface and deployment options (i.e., via desktop client, browser plugin, or web app user interfaces) by optimizing our video QA model using the Open Neural Network Exchange (ONNX), TensorRT, quantizing, and utilizing hardware acceleration capability. Offline translation of our video QA model using large language models such as M2M-100 or NLLB will make our video QA assistant available for use in multiple languages, which will further increase the reach of Snap-Seek. Validation through academic benchmarks (i.e., Tutorial, VQA, TVQA, etc.) will define a performance standard against which our model can be tested and verified and create opportunities for additional research into how to improve video QA functionality across many languages and video formats. The ability for multiple users to collaborate on a project and track the context of their dialogue, as well as developing the ability to adapt to additional media types (e.g., podcasts, webinars, films), are enhancements of the existing capabilities of Snap-Seek. These enhancements serve to increase the versatility and the applicability of Snap-Seek. Together these enhancements are designed to improve the depth of Snap-Seek through improved answers, improved reasoning, better multimodal understanding, and more detailed generative explanations and the breadth through the ability to utilize Snap-Seek across an increased range of languages and platforms and to deploy Snap-Seek in

an efficient manner. As the development of AI for both language and vision continues to advance at an extremely rapid pace, Snap-Seek is positioned to become an integral component of the future intelligent media assistance industry. By enabling the transition from passive viewing to active engagement and knowledge discovery, Snap-Seek is poised to significantly impact the way people engage with video content.

7.3 Final Remarks

Snap-Seek has demonstrated how to integrate multiple modern AI Technologies into a powerful education and productivity tool and lays the groundwork for future improvements. With continued development, Snap-Seek has the potential to evolve into a fully realized intelligent media analytics platform that utilizes multimodal understanding, has cross-platform deployment capabilities, and offers sophisticated personalized support. Snap-Seek could be highly beneficial to the fields of academia, the corporate sector for professional development, and media analysis. It also will provide assistance in accommodating diverse individuals with accessibility barriers and aid in managing individual personal knowledge.

Appendix A

User Manual

A.1 Transcription JSON Segment Sample

Below is an excerpt from the JSON output representing the transcripted segments by Snap-Seek's Whisper model. Each segment contains timestamps, text, and tokenized metadata.


```
{
  "start": 0.0,
  "end": 6.56,
  "text": "Machine learning. Teach a computer how to perform a task
  without explicitly programming it to perform said task.",
  "tokens": [50364, 22155, 2539, ...]
}
```


This structure enables segment-level processing of speech data for summarization and intelligent Q&A indexing.

A.2 System Specifications

Snap-Seek was developed and tested on the following hardware:

- **RAM:** 32.0 GB at 3200 MHz
- **Processor:** Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
- **Graphics:** 4 GB (multiple GPUs)
- **Storage:** 2.05 TB (1.98 TB used)

These specifications ensured that all components, especially the transformer-based models, ran efficiently offline.

A.3 Project Folder Statistics

The complete Snap-Seek project consumed approximately:

- **Disk Size:** 41.2 GB
- **Total Files:** 64,613
- **Total Folders:** 6,354
- **Location:** N:\FYP

A.4 Cached Model Resources

Snap-Seek leverages offline model caching to ensure fast access and minimal internet dependency. The cached models include:

Transformers

- `facebook-bart-large-cnn`
- `t5-base-finetuned-question-generation-ap`
- `xlm-roberta-large-squad2`
- `paraphrase-multilingual-MiniLM-L12-v2`
- `sentence-transformers_all-minilm-l6-v2`

Whisper

- `large-v2.pt`

yt-dlp Extensions

- `youtube-nsig`
- `youtube-sigfuncs`
- `youtube-sts`

A.5 Visual Snapshots

The following figures illustrate various parts of the system's functionality and interface.

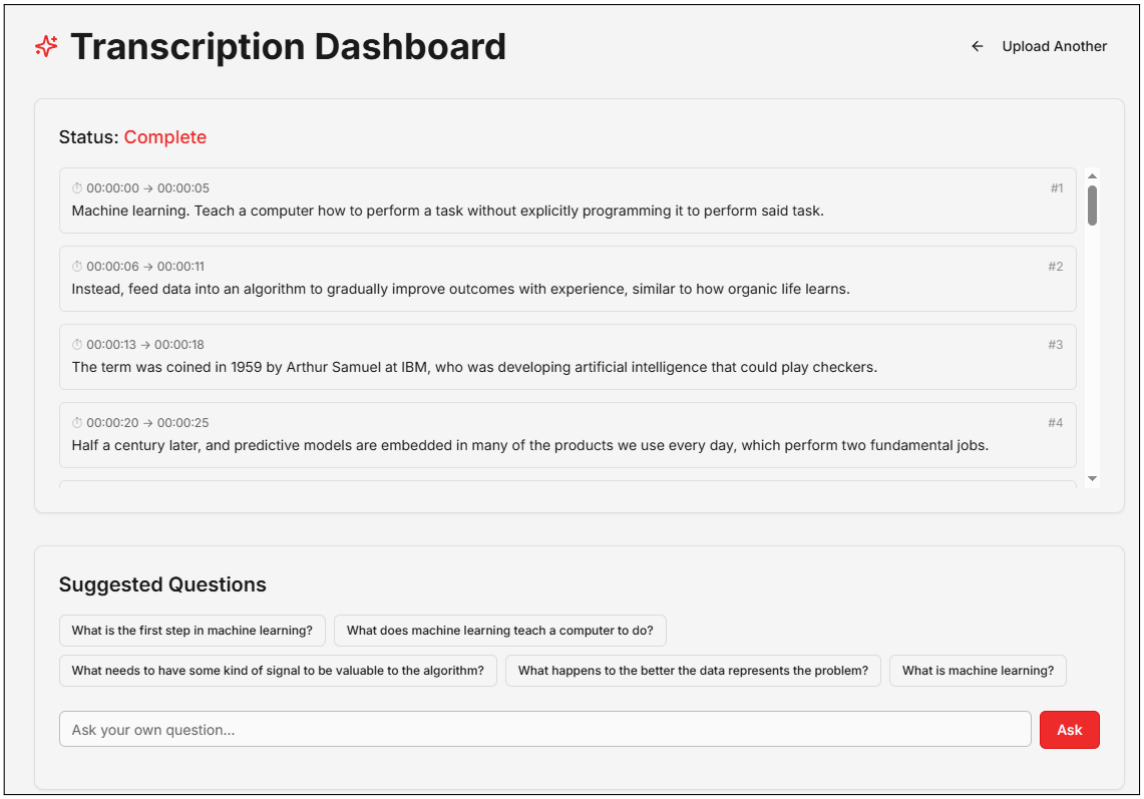

Figure A.1: Transcription Completion Dashboard with Time-Stamped Output

This section provides a brief user guide for operating Snap-Seek, demonstrating how to perform each major action through the GUI.

- **Launching the Application**: When Snap-Seek is started, the user is presented with the main interface (see Figure 4.7). This landing page introduces the tool's features and includes a **Get Started**button. Clicking this button will transition to the video input screen.
- **Uploading a Video or URL**: In the upload interface (Figure 4.9), the user can load content by either dragging a video file into the window or by pasting a YouTube URL into the provided field. The interface offers two tabs one for local file upload and one for URL input to streamline this process. After selecting or entering the video, the user initiates processing (for example, by clicking an **Upload** or **Process** button).
- **Transcription and Summarization**: Once processing begins, Snap-Seek automatically downloads the video (if a URL was provided) and extracts the audio. The Whisper model then transcribes the speech to text. The GUI may show a progress

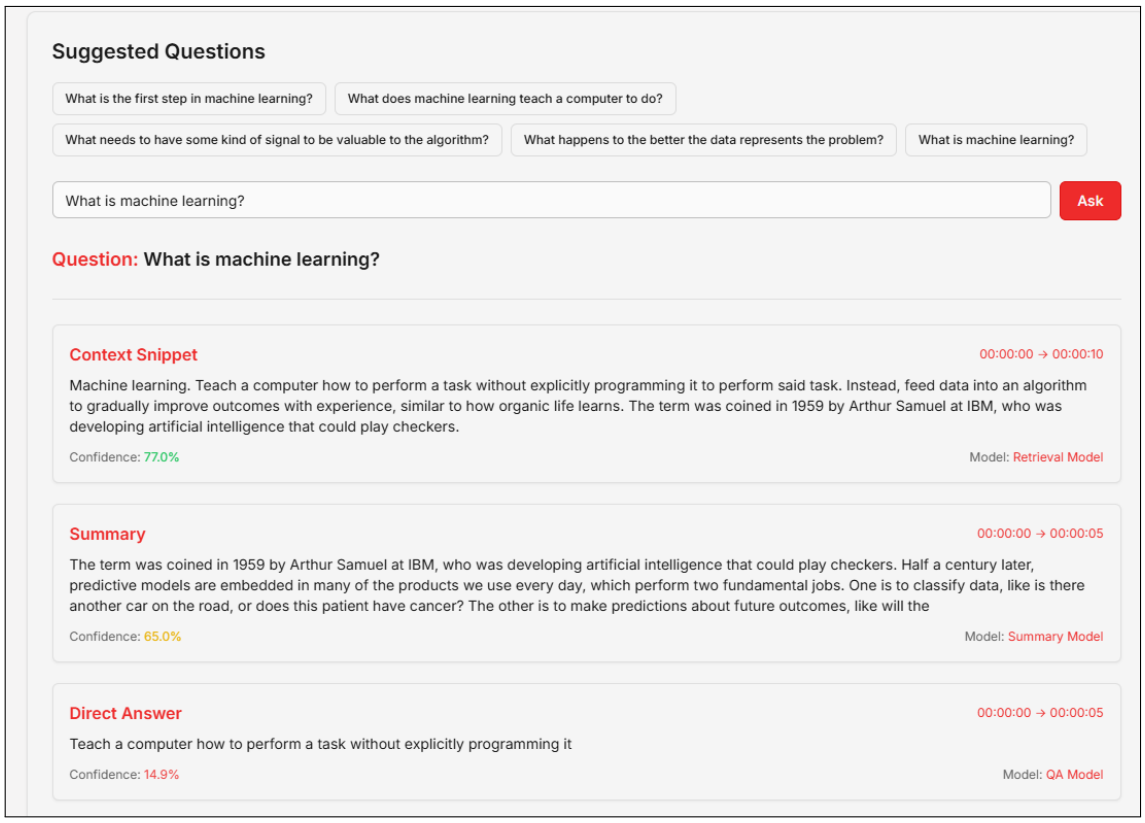

Figure A.2: Context-Specific Q&A Generation Based on Transcript

indicator during these steps. After transcription, the system uses the BART model to summarize the transcript. Within a few minutes (depending on video length), the user will see a **Summary** of the video's content appear in the interface. This summary provides a quick overview of the key points discussed in the video.

- **Asking Questions (Q&A)**: Below or alongside the summary, the interface provides a **Question** input box where the user can ask any question about the video's content. For example, the user might type "What are the main conclusions?" and press **Enter** or click an **Ask** button. Snap-Seek will then retrieve the relevant portion of the transcript and display an **Answer** to the question, along with a timestamp indicating where in the video that answer was spoken. The answer panel typically includes the answer sentence and possibly a snippet of context for clarity. This allows the user to get precise information without manually searching the video.
- **Suggested Questions**: Snap-Seek may also display a list of **Suggested Questions** based on the video (if this feature is enabled). These appear as clickable buttons or list items in the interface. The user can click on any suggested question to automatically run that query. The system will then show the answer just as if the user had typed

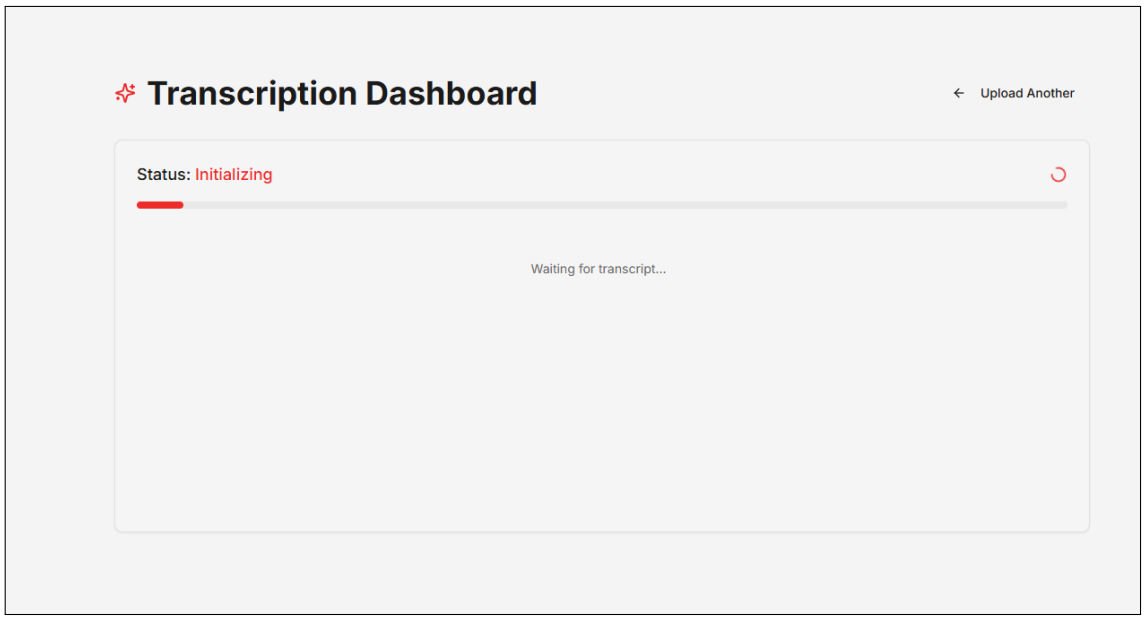

Figure A.3: Processing State During Video Transcription Initialization

the question. This feature helps users discover important details without having to formulate queries themselves.

- **Timestamp Navigation**: One of Snap-Seek's powerful features is the ability to jump to exact moments in the video via timestamps. Whenever an answer is shown with a timestamp (or within the summary or transcript panel if timestamps are displayed), the user can click the timestamp (e.g., "12:34") to open the video player at that specific time. Upon clicking a timestamp, the embedded video player (or the default media player on the system) will play the video starting from that point. In this way, users can quickly verify the context of an answer or view the segment of the video related to the summary.

- Resetting/Starting Over: The interface includes options to reset the session or load a new video. For instance, an **Reset** or **New Video** button allows the user to clear the current transcript and summary and return to the upload screen. This makes it easy to process multiple videos in one session.

- Through these GUI interactions, Snap-Seek enables users to seamlessly go from a raw video to specific information of interest. The user manual steps above summarize how to use each feature: uploading videos, reading summaries, querying the content, utilizing suggestions, and navigating via timestamps. By following these steps, even first-time users can effectively leverage Snap-Seek to extract knowledge from video content.

References

[1] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 28492–28518. PMLR, 2023. Cited on pp. 1, 4, 6, 8, 11, 14, 43, 51, 57, and 61.

[2] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7871–7880. Association for Computational Linguistics, 2020. Cited on pp. 1, 4, 6, 8, 11, 15, 43, 51, 57, and 61.

[3] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. Cited on pp. 1, 4, 8, 11, 15, 43, 58, and 61.

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzman, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8440–8451. Association for Computational Linguistics, 2020. Cited on pp. 4, 6, 11, 15, 43, 52, 57, and 61.

[5] Michal Danilak. langdetect: Language detection library. https://pypi.org/project/langdetect/, 2016. Version 1.0.7. Cited on pp. 4 and 61.

[6] Nidhal Baccouri. deep-translator: A flexible free and unlimited python translation library. https://github.com/nidhaloff/deep-translator, 2020. Cited on pp. 4 and 61.

[7] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, 2019. Cited on pp. 6, 8, 11, 15, 43, 51, 57, and 61.

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020. Cited on pp. 8 and 61.

[9] yt-dlp Developers. yt-dlp: A feature-rich command-line video downloader. https://github.com/yt-dlp/yt-dlp, 2021. Cited on pp. 14, 43, 57, and 61.

[10] FFmpeg Developers. FFmpeg: A complete, cross-platform solution to record, convert and stream audio/video. https://ffmpeg.org, 2022. Cited on pp. 14, 43, 57, and 61.