Palestine Polytechnic University

College of Information Technology and Computer Engineering Department of Computer Science

AI YouTube Video Summarizer

Project Team:

Abdallah Kusbah

Wesam Dwaik

Salahaldin AbdalNabi

Supervisor:

I. Suzan Sultan

This project is submitted to fulfill the requirements for a Bachelor's degree in Computer Science and Information Technology.

It reflects the knowledge and expertise gained over the course of the program.

Jun, 2025

Contents

List of Figures

Acknowledgment

In the name of Allah, the Most Kind and Merciful, who gave us strength, knowledge, and helped us finish this project, we dedicate it to the souls of the martyrs and our brothers and sisters in Gaza. We also want to thank I. Suzan Sultan for helping us a lot in this project. We are very grateful to our families too, who always supported us and encouraged us. Their love and help made us who we are. To all our family members, your belief in us was very important, and we thank you from the bottom of our hearts.

Abstract

This project was developed to help students save time when studying from YouTube videos. Many students, including ourselves, often have limited time to prepare for exams, and watching long videos in full is not always realistic. This motivated us to design a system that makes it quicker and easier to capture the key points of a video.

The website we built extracts the transcript from a YouTube video and transforms it into concise, well-structured notes. Users can also search for a specific word and jump directly to the exact moment it appears, instead of manually navigating through the entire video. In addition, the system offers a feature to compare two videos, highlighting both similarities and differences. This is particularly helpful when multiple videos cover the same subject.

To achieve these functions, we relied on artificial intelligence (AI) techniques. By combining Python with AI-based transcript processing, the system automatically generates summaries, enables word-level search, and supports video comparisons. Through this integration of AI, our project empowers students and other users to study more efficiently without the need to watch every minute of a video.

الملخص:

تم تطوير هذا المشروع لمساعدة الطلاب على توفير الوقت عند الدراسة من مقاطع فيديو YouTube. فالكثير من الطلاب، بمن فيهم نحن، غالبًا ما يكون لديهم وقت محدود للتحضير للامتحانات، ومشاهدة مقاطع الفيديو الطويلة بالكامل لا يكون أمرًا واقعيًا دائمًا. لذلك جاءت فكرتنا بتصميم نظام يجعل الوصول إلى النقاط الأساسية في الفيديو أسرع وأسهل.

الموقع الذي قمنا بتطويرها يقوم باستخراج النص (Transcript) من فيديو على YouTube وتحويله إلى ملاحظات مختصرة ومنظمة. كما يمكن للمستخدم البحث عن كلمة معينة والانتقال مباشرةً إلى اللحظة التي تظهر فيها، بدلاً من التنقل يدويًا عبر الفيديو بالكامل. بالإضافة إلى ذلك، يوفر النظام ميزة مقارنة مقطعي فيديو، حيث يُظهر أوجه التشابه والاختلاف بينهما، وهو أمر مفيد بشكل خاص عندما تتوفر عدة مقاطع فيديو حول الموضوع نفسه.

ولتحقيق هذه الوظائف، اعتمدنا على تقنيات الذكاء الاصطناعي (AI). فمن خلال استخدام لغة Python مع تقنيات معالجة النصوص القائمة على الذكاء الاصطناعي، يستطيع النظام توليد ملخصات بشكل تلقائي، ودعم البحث على مستوى الكلمات، وتنفيذ مقارنات بين الفيديوهات. ومن خلال هذا الدمج للذكاء الاصطناعي، يمكّن مشروعنا الطلاب وغيرهم من المستخدمين من الدراسة بشكل أكثر كفاءة دون الحاجة إلى مشاهدة كل دقيقة من الفيديو.

Chapter 1

Introduction:

In this chapter, we will introduce the general problem space and the challenges associated with video content. We will then discuss the specific difficulties faced by different types of users and explain the overall aim of the project. Furthermore, we will present the idea behind the project and highlight previous systems that offer similar services. In addition, we will review the available alternatives in the field. Finally, we will outline the scope of the project and its limitations in order to clearly define what this work can and cannot achieve.

1.1 Problem Space and Challenges:

YouTube is becoming more popular every day, and videos are getting longer and more detailed. This often consumes a lot of time, especially when someone is only looking for a specific piece of information. To solve this, we built a tool that makes it easier and faster to find what you need inside a video. It can generate summaries and let users search within the transcript, so they can jump straight to the context they're interested in without watching the entire video.

1.2 Challenges Faced by Users:

1. **Time-consuming video watching:**
    - There is a lot of educational videos and there is a the long and short video they might talk about the same topic but different approach there might be unnecessary information in video or missing , we made it possible by to the user to get the summarize for each one or compare them.
2. **Searching for specific information within videos:**
    - Some users are looking for specific information in the video , we made it possible by adding the search bar to search about a specific word in the context and the user can scroll in the video timeline and see each second and the text.
3. **Difficulty in comparing content from two different videos:**
    - Users are looking to watch two videos or more, they might be talk about the same topic and the user is looking to see what is the difference between them , one video is long and the other is short there is curious users that are wondering what is the difference between the short and long video , other thing there is users are looking to compare two topics and see the difference between them.

1.3 Project Aim:

This project introduces a website called (YouTube AI Summarizer), designed to help users quickly understand the main ideas of a video without watching the whole thing. The tool works by pulling the transcript of a YouTube video and then turning it into a clear summary. Besides summaries, the site also lets users compare two or more videos, making it easier to spot similarities and differences. This feature is especially useful when dealing with multiple lectures or tutorials on the same topic. The project is meant for students who need fast study material, instructors who want to prepare teaching content more efficiently, and even content creators who want a quick overview of existing material. In short, it helps save time, simplify video learning, and improve productivity.

1.4 Project Idea:

This project aims to help the youtube users to analyze videos , compare them, extract specific information and ask about the video context all of that is without watching the full videos.

- **Summarize** Summarize YouTube video content to highlight key points.
- **Answer** Answer user questions about the content of the video.
- **Compare** two or more videos on similar topics to provide insights and differences.

Goal: The project goal is to help users to make it easier to search for specific information or compare between videos or summarize them with the artificial intelligence

1.5 Previous systems Offering Similar Services :

1. **NoteGPT:**
    - **YouTube Video Summarization:** The platform uses AI to summarize YouTube videos into concise points, enabling users to quickly understand key concepts without watching the entire video. It supports custom summary lengths, catering to quick overviews or detailed insights.
    - **Content Transcription:** NoteGPT provides accurate transcriptions of videos, transforming spoken content into text for easier reference and review.
    - **AI Chat Assistant:** Users can interact with an AI assistant for deeper explanations, clarifications, or discussions about the content, making it a more engaging experience.
2. **Monica.im YouTube Summary:**
    - **YouTube Video Summarization:** Monica provides real-time or post-view summaries of YouTube videos. Users can input video links or generate highlights and key timestamps while watching.
    - **Multi language Support:** The tool supports over 100 languages, enabling users to summarize videos in different languages easily.
    - **Interactive AI Chat:** Users can engage in conversations about the video content, asking specific questions or requesting additional details about topics covered.

Target Audience for the Project:

1. **Content Creators:**
    - In this case , there might be a content creator who is looking for a new topic to talk about in his video or blog post , so he will search on the internet , youtube ,we offered him by taking the youtube video link and use our site to search and ask summarize the video , that will help him to prepare a context to talk about in his blog post or video in other platform.
2. **Students:**
    - The most targeted group are the student because we have been through this , watching and searching for videos that explain a topic for us , the summarization helps to get the get the goal of the video and it's content in short way , that way consumes short time and is very effective one more thing is finding two or more videos that talked about the same topic but different length our project helps to summarize the video context and answer the student questions that will save a lot of time for studying.

3. **Academic Instructors:**

    - Academic teachers might need to search for missing information about a lesson they are getting ready for , they can ask for a summary or ask about keywords or the lesson goals or any other thing about the video.

1.6 Available Alternatives:

1. **Manual Summarization:**
   An alternative to automated summarization is the traditional method, where a person watches the entire video and then writes a summary. This usually involves taking notes and preparing either a detailed explanation or a short overview of the content. While this approach can be very accurate, it is also slow and requires a lot of time and effort.

2. **AI-Powered Summarization (Proposed System):**
   Our project uses automated techniques to make the video summarization process faster and more efficient. By analyzing the transcripts of videos, the system produces clear summaries without the need for manual work. This saves time and makes it possible to handle a large number of videos more easily, while still keeping the results accurate and useful.

Comparison Between Manual and AI-Powered Summarization:

- **Efficiency:** The manual method relies on human effort, whereas the AI-powered approach automates the entire process, ensuring faster results.
- **Scalability:** Manual summarization becomes impractical for handling large datasets, while AI can process vast amounts of data simultaneously.

1.7 Scope And Limitation:

The project is limited to YouTube videos only. Other video platforms are not supported, and the system focuses exclusively on processing content from YouTube.

1.8 Gantt Table:

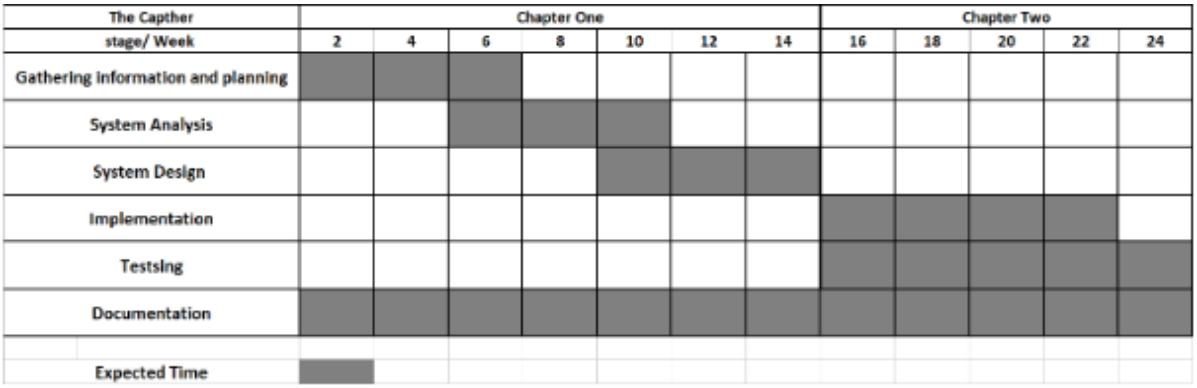


Figure 1.1: Gantt Table

Chapter 2

Functional and Non-Functional Requirements

2.1 Introduction:

In this chapter, we will talk about the actors directly involved and affected by the site. We will also talk about the Functional Requirements and the scenario for each job, as well as the Non-Functional Requirements for this site.

2.2 Users Concerned with the Site:

1. Users
2. AI Model

2.3 Functional requirements for site elements:

User Requirements:

1. **Access Summaries:** Input YouTube video links to receive automated summaries.
2. **Access Compare Videos:** Compare two or more videos on the same or related topics.
3. **Ask Questions:** Submit questions about video content and get AI-generated answers.
4. **Save and Organize:** Save the video URL and image, then save it in my notes Page.

AI Model:

1. **Summarize Transcript:** After extracting the transcript, the system summarizes it using AI.
2. **Answer Questions:** Users ask questions about the YouTube video, and the AI provides answers based on the extracted transcript.
3. **Compare Videos:** Choose two videos from your notes. The system extracts transcripts and identifies differences in topics, keywords, and main points.
4. **Extract Transcript:** Enter a YouTube video link to automatically receive its transcript

   Extracting the Thumbnail: When a YouTube video link is provided, the system extracts the video ID from the URL using the `urllib.parse` library. It then displays the corresponding thumbnail image to the user alongside the video details.

2.4 Use Case Tables:

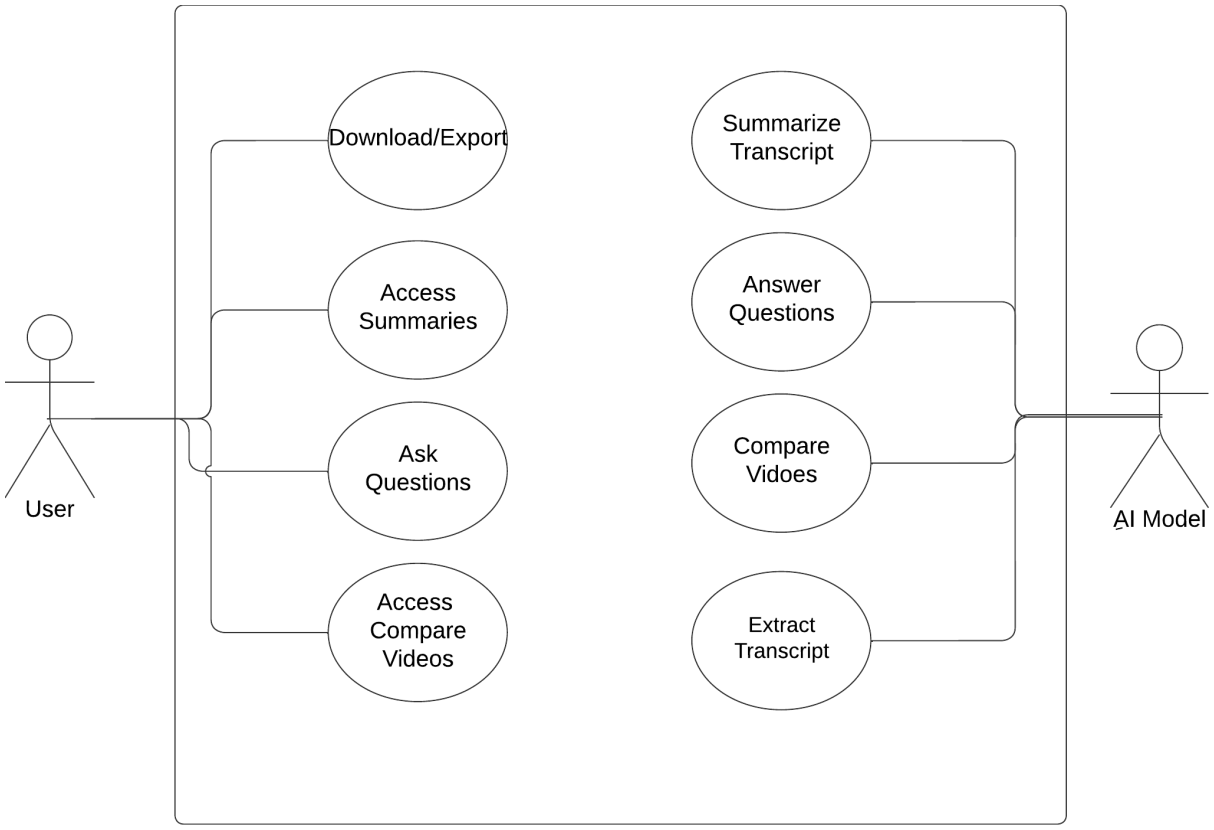


Figure 2.1: Use-Case

Main Actor	User
The Goal	Generate video summaries from YouTube links.
Prerequisites	AI Summarization Engine
The Effect	Summarize video content quickly.
Scenario	**Scenario:** User inputs video URL. AI processes the video and generates a summary. User can save or download the summary.

Table 2.1: Table 1: Access Summaries

Main Actor	User
The Goal	Compare two or more videos on related topics.
Prerequisites	AI Summarization Engine
The Effect	Identify key differences between videos.
Scenario	**Scenario:** User selects 'Compare Videos'. AI compares the videos. Differences and similarities are highlighted.

Table 2.2: Table 2: Compare Videos

Main Actor	User
The Goal	Submit questions about video content for AI-generated answers.
Prerequisites	AI Summarization Engine
The Effect	Receive quick answers from video content.
Scenario	**Scenario:** User accesses 'Ask AI'. User inputs video URL and types a question. AI processes the transcript and generates answers.

Table 2.3: Table 3: Ask Questions

Main Actor	User
The Goal	Save video summaries for future reference.
Prerequisites	Database
The Effect	Summaries are categorized and stored.
Scenario	**Scenario:** User views summary. Video URL and transcript are stored.

Table 2.4: Table 4: Save and Organize

Main Actor	User
The Goal	Download transcript as PDF.
Prerequisites	Database
The Effect	Summarization is available offline.
Scenario	**Scenario:** User views the transcript. User selects 'Download'. The summary is downloaded.

Table 2.5: Table 5: Download/Export

Main Actor	AI Model
The Goal	Extract transcript for URL.
Prerequisites	AI has access to YouTube video transcripts via an API.
The Effect	AI automates the process of extracting transcript.
Scenario	**Scenario:** User inputs a YouTube video link. AI extracts the transcript from the video using an API. AI cleans and structures the text for readability. AI provides the full transcript to the user.

Table 2.6: Table 6: Extract Transcript

Main Actor	AI Model
The Goal	Process the summaries from transcript.
Prerequisites	Transcripts
The Effect	AI automates the process of summarizing.
Scenario	**Scenario:** AI receives the extracted transcript. AI analyzes key points, topics, and essential details. AI generates a concise summary of the transcript. AI delivers the summary to the user in an easy-to-read format.

Table 2.7: Table 7: Summarize Transcript

Main Actor	AI Model
The Goal	Process the answer of questions from user.
Prerequisites	Transcripts and user questions.
The Effect	AI automates the process of Q&A.
Scenario	**Scenario:** User asks a question related to the video content. AI searches the transcript for relevant information. AI generates a clear, good answer based on the transcript.

Table 2.8: Table 8: Answer Questions

Main Actor	AI Model
The Goal	Show the differences between two videos.
Prerequisites	Transcripts and video.
The Effect	AI automates the process of comparing videos.
Scenario	**Scenario:** User selects two videos from my notes. AI extracts and processes the transcripts of both videos. AI analyzes and compares differences in: - Purpose and Focus - Learning Approach - Target Audience - Keywords - Conclusion AI provides a structured comparison report.

Table 2.9: Table 9: Compare Videos

2.5 Non-Functional Requirements:

1. **Performance**

    - The system should generate responses in less than 5 seconds for transcript extraction.
    - The system should handle at least 50 concurrent users without performance degradation.
    - The system should query the database in under 500 ms.

2. **Accuracy**

    - The chatbot should provide accurate and relevant responses related to the video content.
    - The chatbot should give accurate reference to video timestamps.
    - The chatbot should classify the topics in the video correctly.

3. **Usability**

    - The system has a usable interface.
    - The system interface is simple and easy to use.
    - The system provides instructions on how to use it for the end user.

4. **Validation**

    - If the URL is not from YouTube, the system should:

        – Reject the input with an error message: *" Please enter a valid YouTube URL."*

Chapter 3

System Design and Analysis

3.1 Introduction:

In this chapter, we will show the important parts of our project. We will show the design of the pages and how the project works. We will also explain how we take the words from the video, how the chatbot gives answers, and how we show the time of the important parts in the video.

3.2 Website Interface Design:

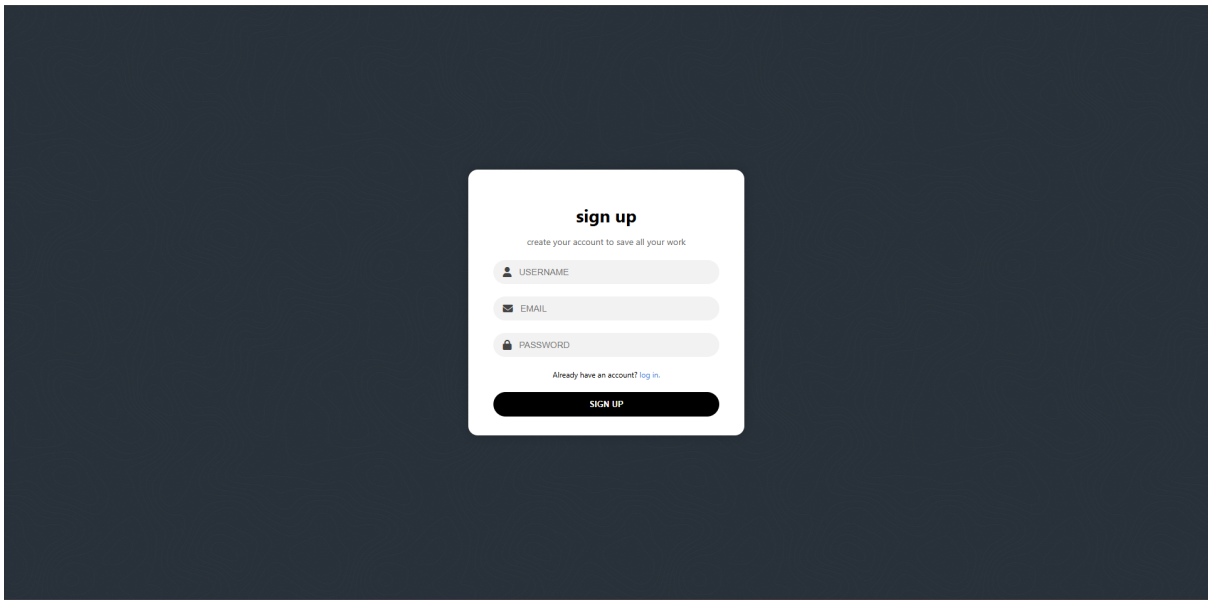


Figure 3.1: SingUp Page

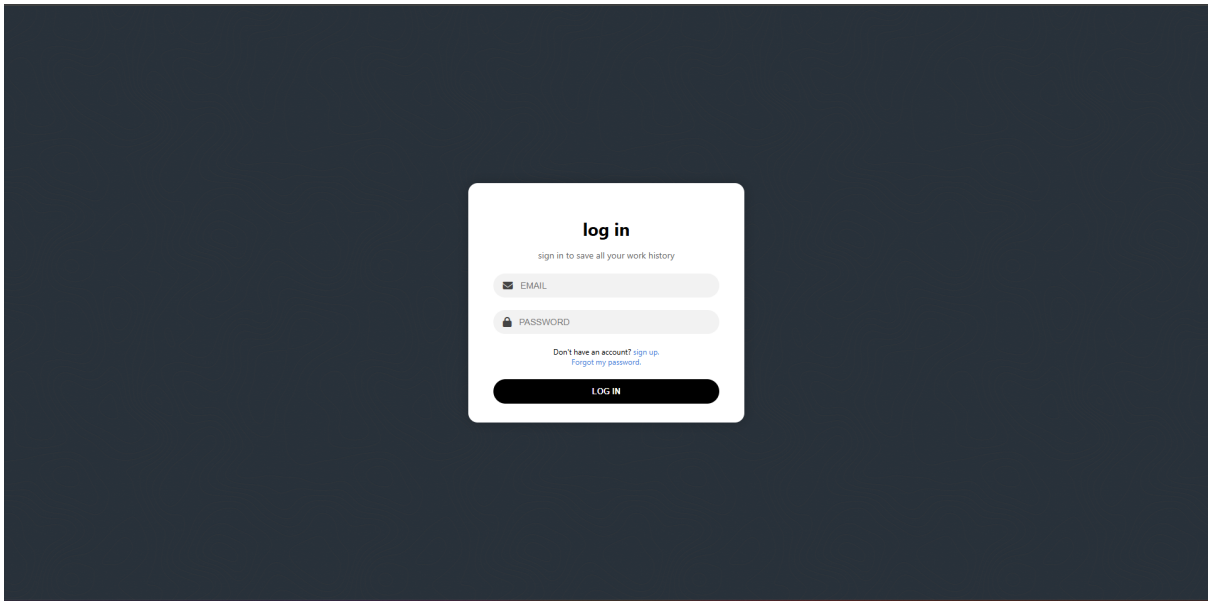


Figure 3.2: Login Page

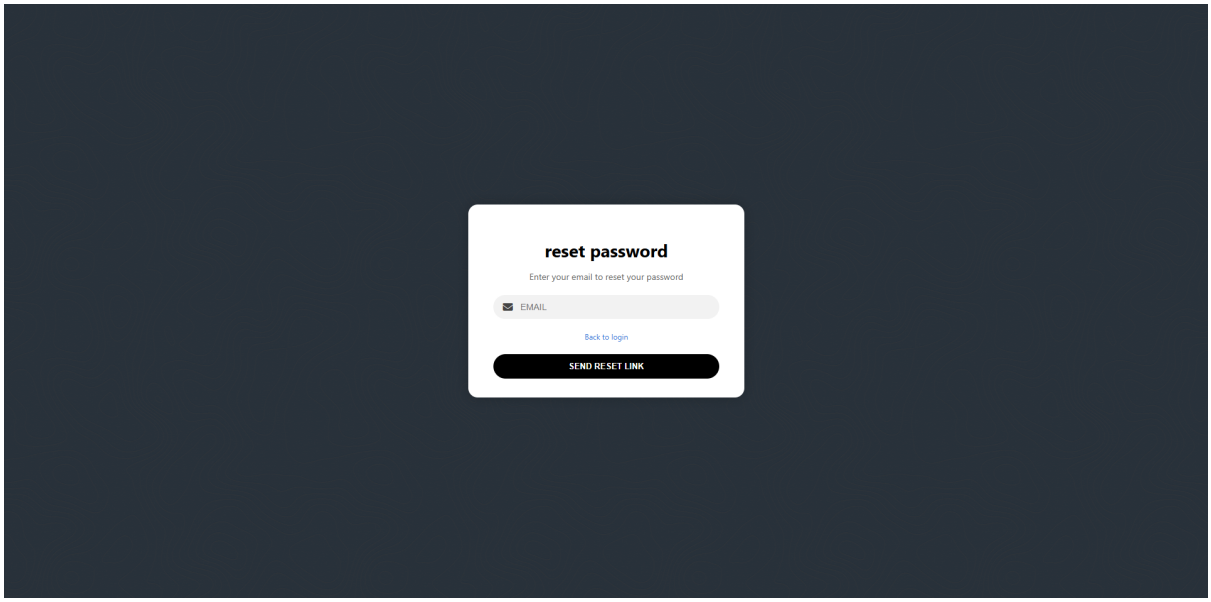


Figure 3.3: Rest Password Page

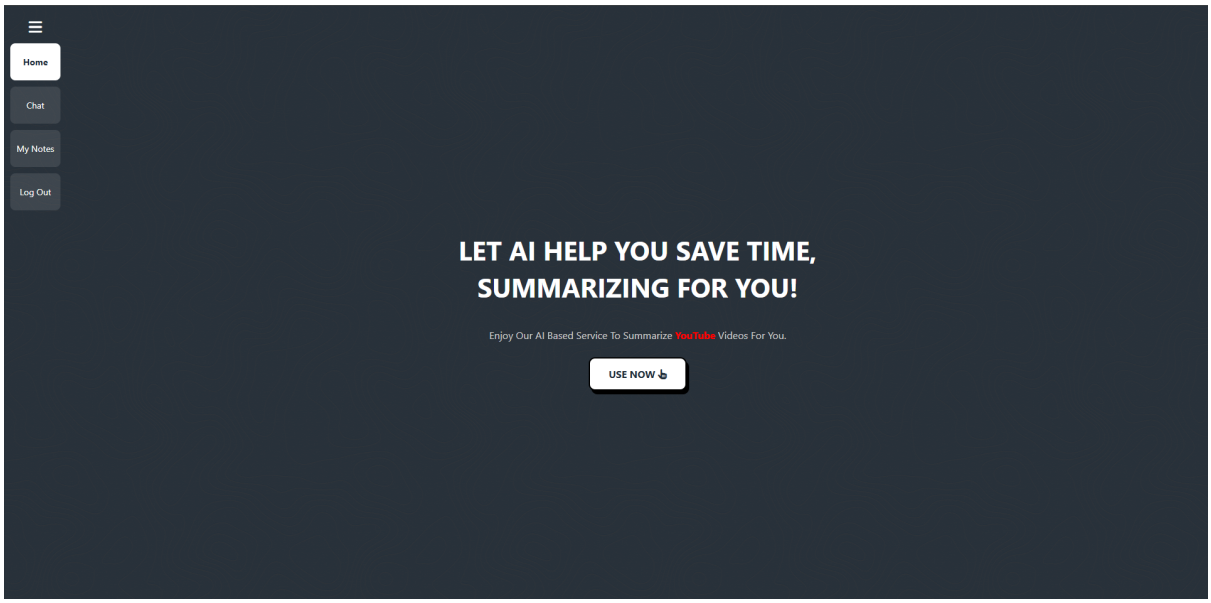


Figure 3.4: Home Page

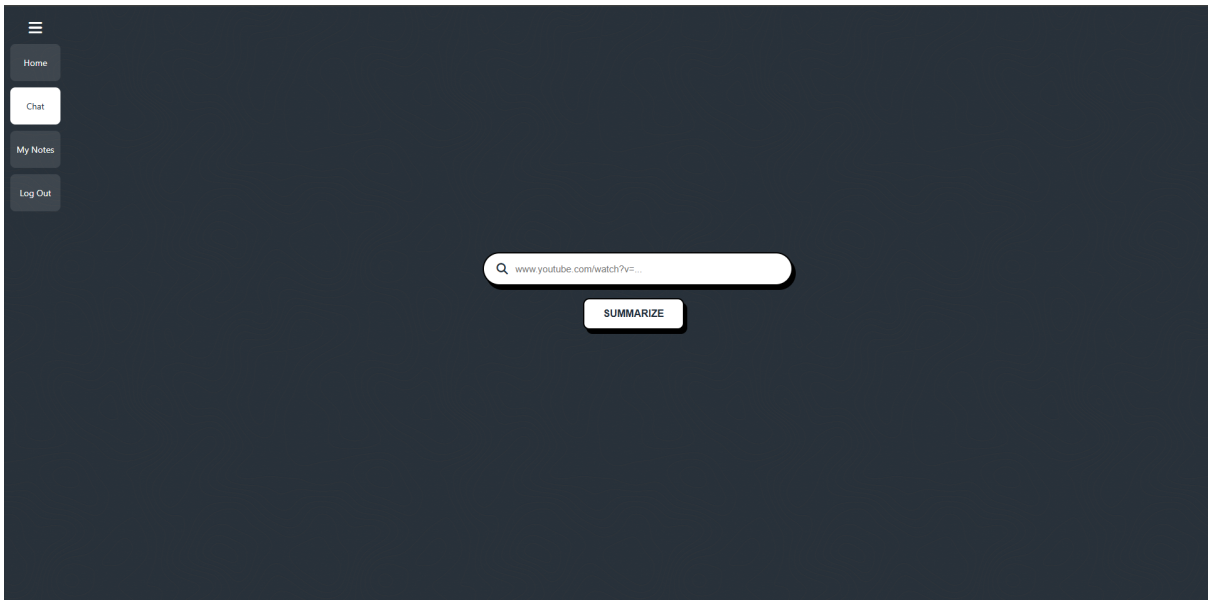


Figure 3.5: URL Page

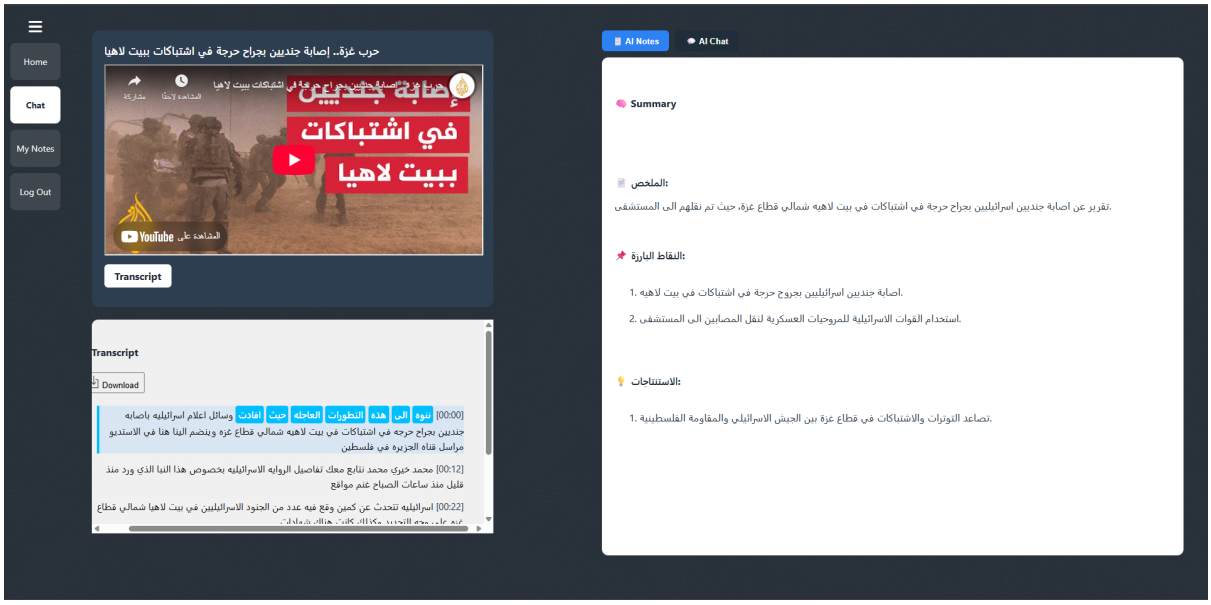


Figure 3.6: Chat Page

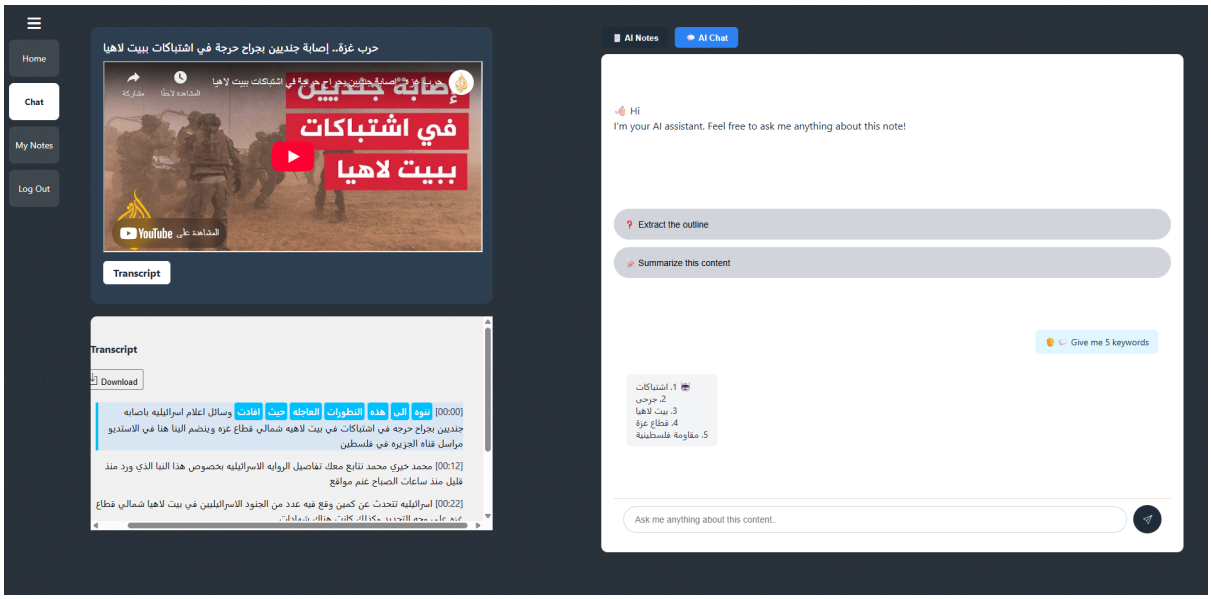


Figure 3.7: AiChat Page

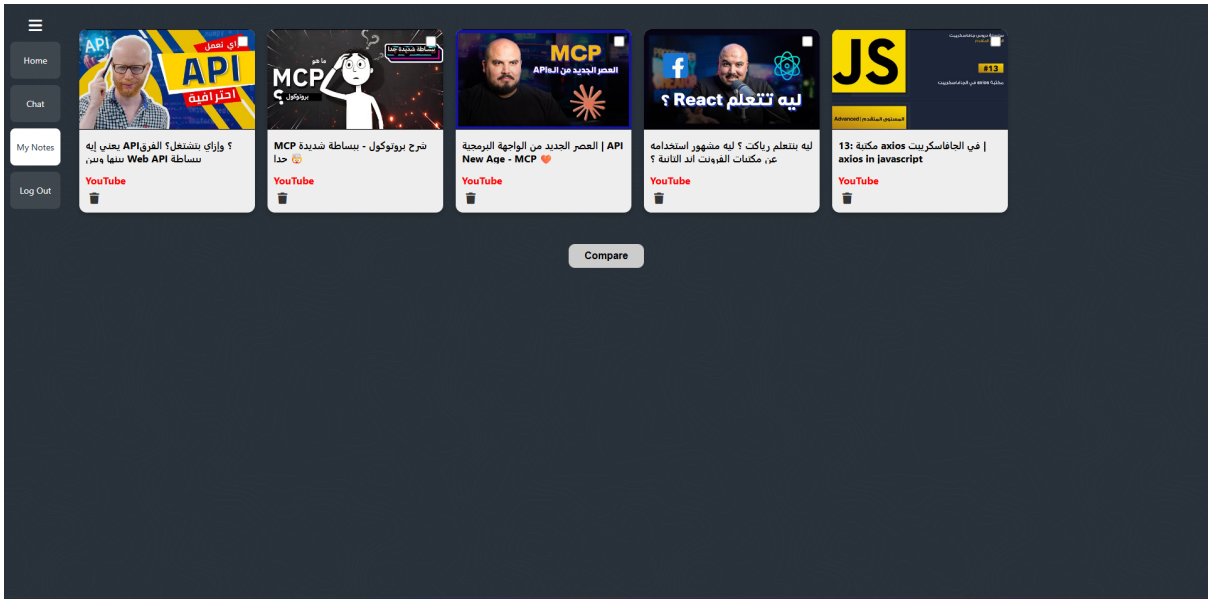


Figure 3.8: Note Page

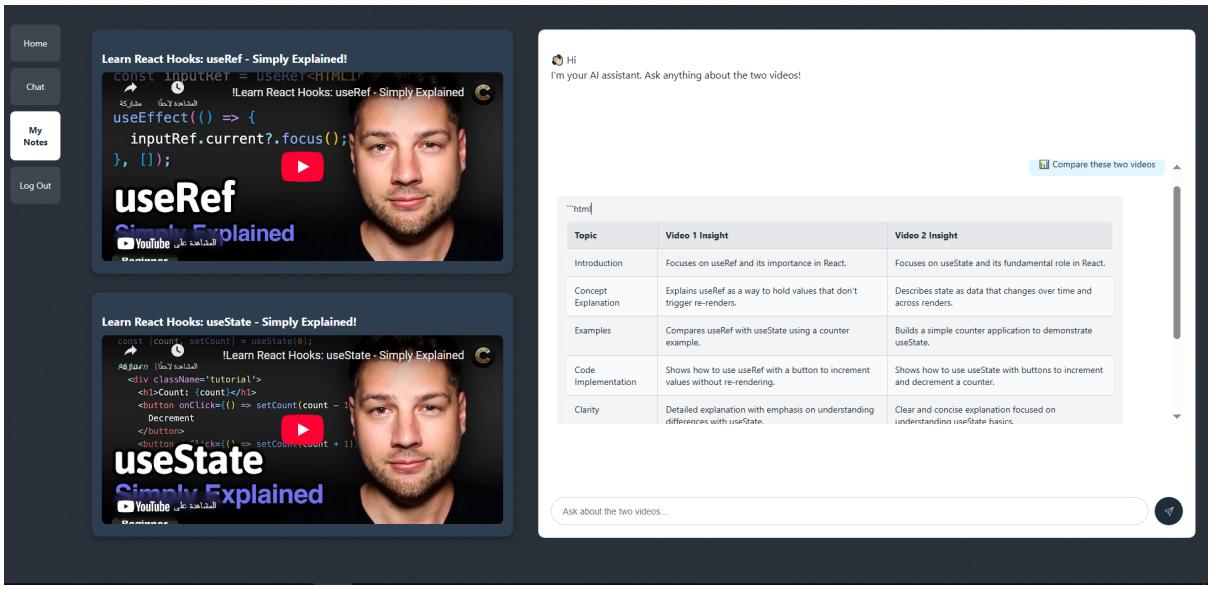


Figure 3.9: Compare Page

3.3 Database:

This table which is made of two tables – the Users table which includes info like name, email, and password for each individual user which has a unique ID and the Summaries table which has what users have put in as sumsary which may be a video link, some text, and date of creation. Each summary is put under a certain user by ID number which also means a single user may have many.

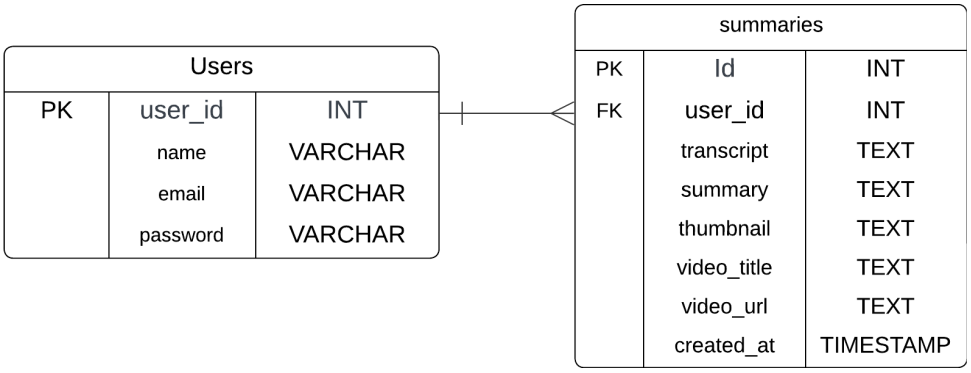


Figure 3.10: Database

3.4 Summarization Feature:

This part gives a small summary of the video, The user adds a YouTube link, The system takes the words from the transcript, GPT reads these words. It writes a Summary, Highlights and Key Insights, This helps the user.

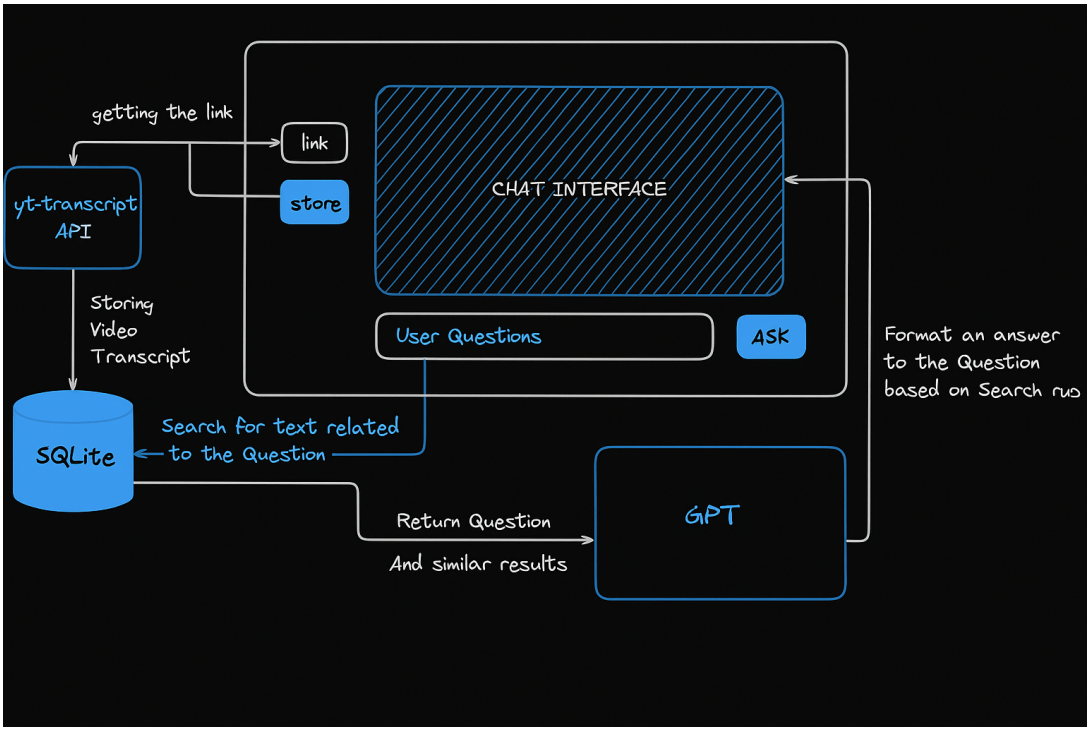


Figure 3.11: Summarization process in the system

3.5 Comparison Feature:

This part gives a small summary of the video, The user adds a YouTube link, The system takes the words from the transcript, GPT reads these words. It writes a Summary, Highlights and Key Insights, This helps the user.

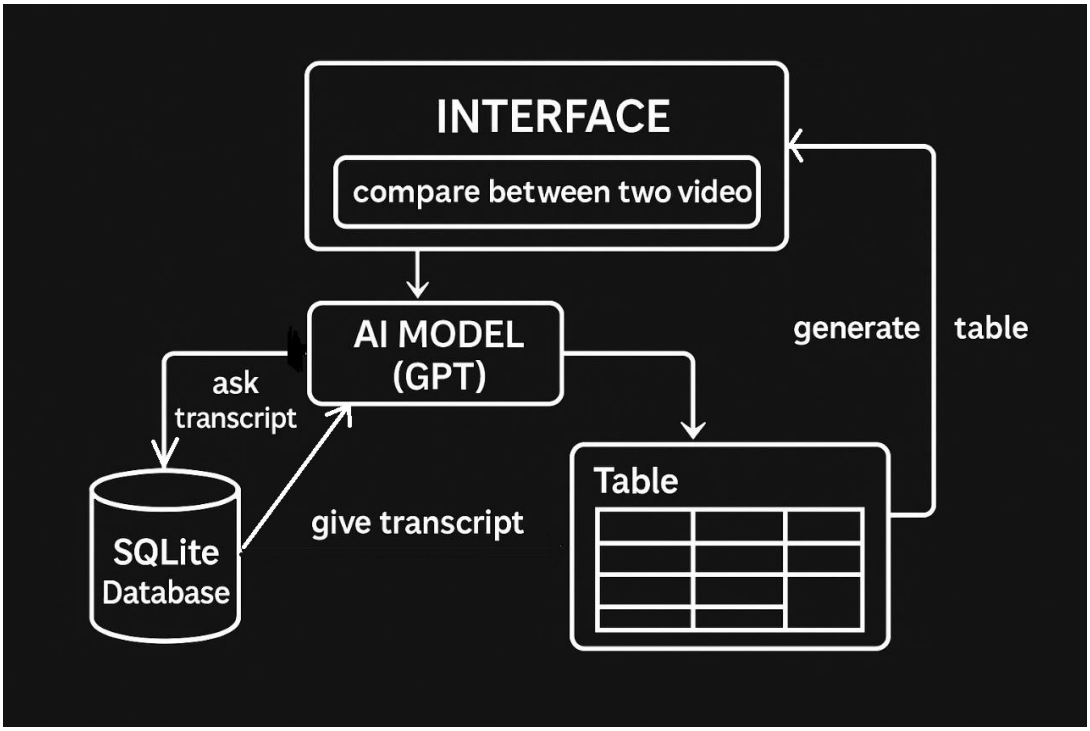


Figure 3.12: Comparison process in the system

3.6 Transcript Extraction from YouTube Videos:

We used in the project the library (YouTube Transcript API) which depends on sending (HTTP Requests) to YouTube and prepares the automatic or manually added translation inside the video when the (video ID) is available. The library extracts the full transcript if it exists and returns it in the form of a list containing time parts of the texts.

3.7 Extracting the Thumbnail from a YouTube Video:

In order to extract the image, first we need the (video ID) from the URL using the urllib.parse library. We use the urlparse() function to divide the link into sections and extract the ID section. We write in this link (https://img.youtube.com/vi/¡video$_id > /hqdefault.jpg$)

Chapter 4

Implementation and Evaluation

4.1 Introduction:

In this chapter we present what we did in the implementation of our project and the tools we used to put it together. We chose certain technologies and libraries which we found to do best in the YouTube video, transcript extraction, and AI summary generation tasks. Each tool we picked out for their simplicity, performance and because they helped us out with real issues that came up during the development stage. Also in this chapter we describe the integration of these tools into our system and we present the code which we used for the transcript handling, response generation and management of user data. Also we report on the issues we had and how we went about solving them.

4.2 Handling Alternatives

This section explores the technologies we used in the project and compares them with possible alternatives.

Subject	Technology Used	Alternative	Comparison
Database	SQLite3	MySQL / PostgreSQL	SQLite3 is lightweight, serverless, and easy to integrate for small projects, while MySQL and PostgreSQL are more powerful but require server setup and administration.
Framework	Flask	Django	Flask is simple and flexible, ideal for quick prototyping, while Django is heavier but provides built-in features such as authentication and admin dashboards.
AI Model	ChatGPT (OpenAI)	LLaMA / Google PaLM	ChatGPT offers strong API support and reliable summarization quality, while LLaMA and PaLM are powerful but harder to deploy and require more resources.
Library	urllib.parse	Regex	urllib.parse is built-in and reliable for URL parsing, while regex can work but is error-prone and harder to maintain.
Library	YouTubeTranscriptApi	YouTube Data API	YouTubeTranscriptApi is free and lightweight for extracting captions, while YouTube Data API is more official but requires API keys and setup.
Library	langdetect	langid.py	langdetect is simple and fast for language detection, while langid.py offers similar accuracy but is slightly slower and heavier.
Language	Python	JavaScript (Node.js)	Python has a rich ecosystem for AI/ML and NLP, while Node.js is strong for real-time apps but has fewer ready-made NLP libraries.

Table 4.1: Technologies Used and Alternatives with Comparison

4.3 Technologies and Frameworks:

- **Python**: It is the primary language that will be used to develop the system, the main reason is the flexibility along with the convenience provided by the libraries it has.
- **Flask**: A Python framework used to build the server that receives the video link from the user and returns the digest after processing it, Flask integrates APIs and manages data transfer between client and server.
- **Ollama**: is an open-source AI models provider that runs in the terminal, you can think of it as an online hub that contains open-source AI models that you use to pull down to your local machine some LLMs and then run them. On top of that ollama provide python libraries to enable the integration of LLMs with the python code you write.
- **YouTubeTranscriptApi**: A Python library used to extract transcripts from YouTube videos. It sends HTTP requests to retrieve subtitles or manually added captions if available.
- **urllib.parse**: A Python library used to parse YouTube video URLs accurately to extract video IDs, which helps the system handle requests properly and integrate with external services like YouTube APIs.
- **langdetect**: A Python library used for automatic language detection of the transcript text, ensuring the data sent to the AI model is in the correct language (mainly Arabic or English).
- **OpenAI GPT-4o**: The AI model used to analyze the extracted video transcriptions and generate summaries, answer questions, and compare video contents, it provides fast and accurate results.
- **SQLite3 Database**: A lightweight, file-based database used to store user information, video details, transcripts, and AI-generated summaries securely and efficiently.
- **Frontend Technologies (HTML, CSS, JavaScript)**: These are used to build the user interface, including video playback, transcript display, AI notes, and user interaction components.

4.3.1 AI Model:

Chat GPT 4o-OpenAi

Initially we began with llama3 for our experiments (testing) which we later on switched to openai as the official API to interact with AI models GPT we sent out extracted video transcriptions (Chat GPT 4o) to the model for analysis which in turn gave us smart sumsaries to present to the user thus we achieved an automatic, efficient and fast summarization process.

4.3.2 Frameworks:

In our project, we used Flask as the main framework for building the backend and API. Flask was chosen because it is lightweight, easy to set up, and well-suited for integrating external services. It allowed us to create endpoints that receive YouTube video links, process them, and return the summarized output. The backend was implemented in Python, which served as the programming language to connect all components together. Python was also used to call external libraries such as YouTubeTranscriptApi for transcript extraction and the GPT-4o model for text summarization and comparison. By combining Flask with Python and these libraries, we were able to design a modular system where each component communicates smoothly, making the application easier to maintain and extend.

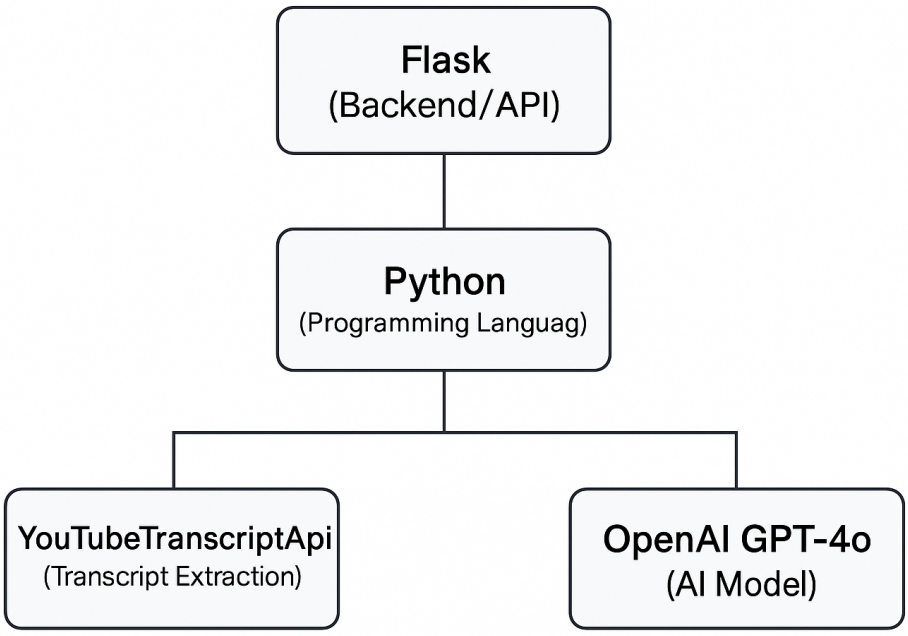


Figure 4.1: Backend Workflow of the Summarization System

4.4 Front-end:

In the interfaces, we used languages such as HTML to create the page appearance, CSS to give the appropriate appearance to the page, and JavaScript for validation and to communicate with the backend.

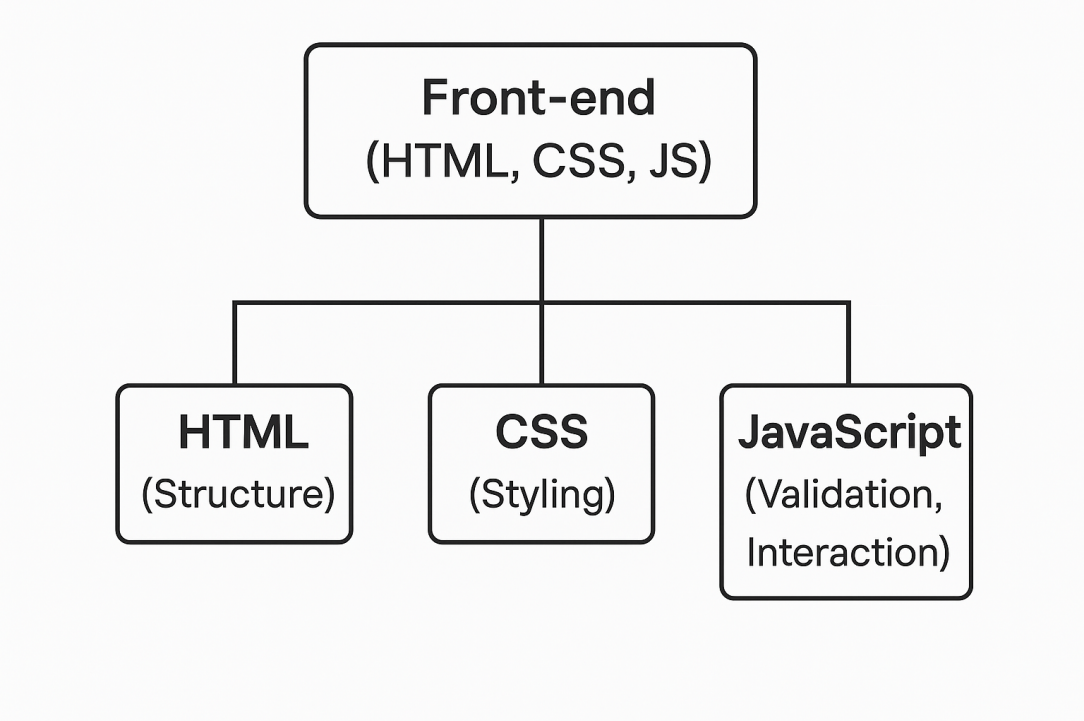


Figure 4.2: Frontend Workflow of the System

4.5 Problems we encountered:

Transcript

- **I**nitially, we used ChatGPT's Whisper ffmpeg to load the audio file and extract the text, but the process didn't work because it was local and took a long time to extract the text, load the audio, and transcribe, which was pretty slow at first. We then tried API facebook/wav2vec2-base-960h, which also suffered from slow extraction. We ended up using a YouTubeTranscriptAPI, which free solutions couldn't solve.
- **S**ome YouTube videos do not provide transcripts, which makes it impossible to extract the text directly. To handle such cases, a paid speech-to-text API is required. We tested free alternatives, but they were very slow and took much longer to process the videos, making them impractical for our project.

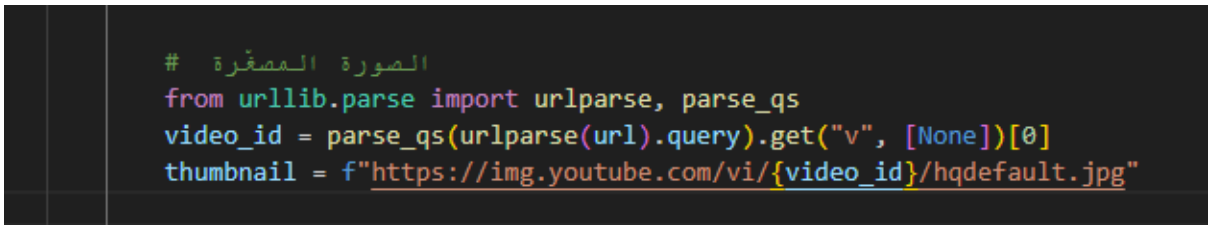
4.6 System features :

```
try:
    transcript_obj = YouTubeTranscriptApi.list_transcripts(video_id)

    # تجيب أي لغة (عربي أو إنجليزي)
    try:
        transcript = transcript_obj.find_transcript(['ar'])
    except:
        try:
            transcript = transcript_obj.find_transcript(['en'])
        except:
            transcript = transcript_obj.find_transcript(['ar', 'en'])

    transcript_data = transcript.fetch()
```

Figure 4.3: Transcript code :
This code uses to specify whether the video is in Arabic or English.

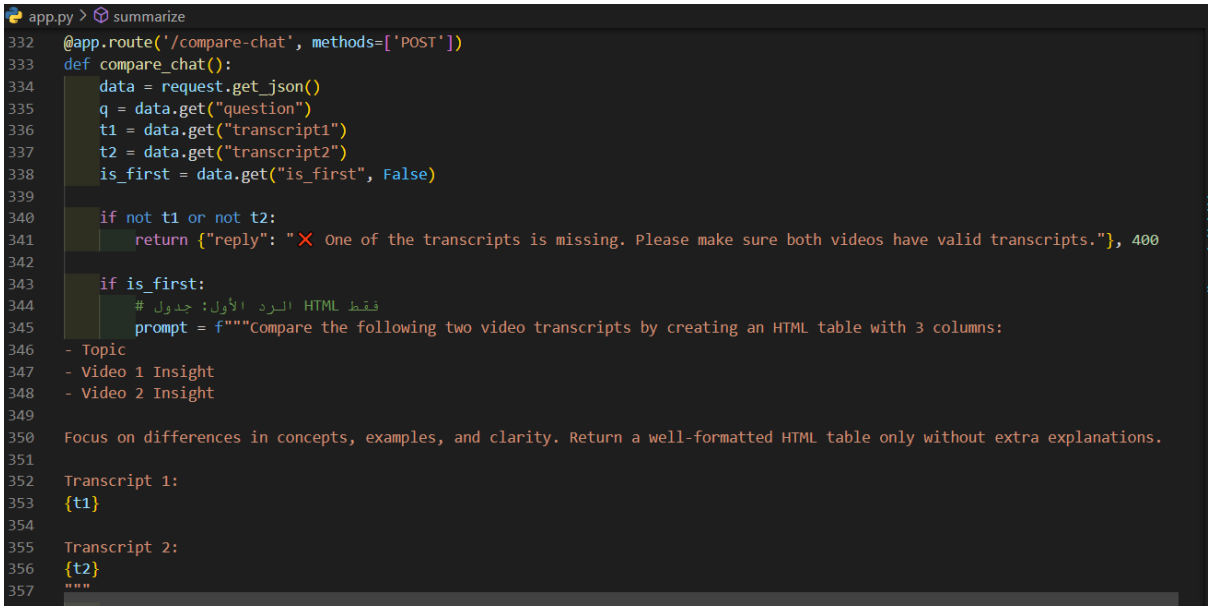
```
if not user_input or not transcript:
    return { "error": " Missing transcript or question." }, 400

try:
    prompt = f"Based on this video transcript:\n\n{transcript}\n\nAnswer this user question:\n{user_input}"
    if lang == "ar":
        prompt = f"بناءً على هذا النص:\n\n{transcript}\n\nأجب على هذا السؤال:\n{user_input}"

    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant answering based on a YouTube transcript."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7,
        max_tokens=400
    )
    reply = response['choices'][0]['message']['content'].strip()
    return { "reply": reply }
```

Figure 4.4: Response code :
This code combines the transcript extracted from the YouTube video with the user's question, and sends both to the OpenAi model to answer the question based on the text alone.



```python
# الصورة المصغّرة
from urllib.parse import urlparse, parse_qs
video_id = parse_qs(urlparse(url).query).get("v", [None])[0]
thumbnail = f"https://img.youtube.com/vi/{video_id}/hqdefault.jpg"
```

Figure 4.5: Thumbnail code :
This code is used to extract the YouTube video thumbnail from the video link.



app.py > summarize

```python
@app.route('/compare-chat', methods=['POST'])
def compare_chat():
    data = request.get_json()
    q = data.get("question")
    t1 = data.get("transcript1")
    t2 = data.get("transcript2")
    is_first = data.get("is_first", False)

    if not t1 or not t2:
        return {"reply": "❌ One of the transcripts is missing. Please make sure both videos have valid transcripts."}, 400

    if is_first:
        # فقط HTML الرد الأول: جدول
        prompt = f"""Compare the following two video transcripts by creating an HTML table with 3 columns:
- Topic
- Video 1 Insight
- Video 2 Insight

Focus on differences in concepts, examples, and clarity. Return a well-formatted HTML table only without extra explanations.

Transcript 1:
{t1}

Transcript 2:
{t2}
"""
```

Figure 4.6: compare code :
In the comparison, we use the script in the first video and the script in the second video from the database, and we make a comparison between the two videos.

```
// تتبع الوقت ومزامنة الكلمات
function startTracking() {
  setInterval(() => {
    if (player && typeof player.getCurrentTime === "function") {
      const currentTime = Math.floor(player.getCurrentTime());
      // تمييز الكلمة حسب الوقت
      document.querySelectorAll(".word").forEach(word => {
        const wordTime = parseInt(word.getAttribute("data-start"));
        if (wordTime === currentTime) {
          word.classList.add("highlighted");
          // word.scrollIntoView({ behavior: "smooth", block: "center" });
        } else {
          word.classList.remove("highlighted");
        }
      });
```

Figure 4.7: Transcript tracking code
This JavaScript function synchronizes the video playback time with the transcript display by highlighting each word as its corresponding timestamp is reached during playback.

4.7 Performance Testing:

In our performance testing, we evaluated how fast and reliable the YouTube summarizer system works under different conditions. We simulated multiple users sending requests at the same time to measure response time, accuracy, and stability. For transcript extraction, the system responded on average in less than 10 seconds for short videos (under 10 minutes) and around 20-30 seconds for longer videos (over 30 minutes). and around 60-120 seconds for longer videos (over 60 minutes). The summarization process produced results within 3–5 seconds per request. Even when testing with up to 20 concurrent users, the system maintained stable performance without crashes. These results show that the system can handle multiple requests simultaneously while still delivering accurate summaries quickly and reliably.

Importance of Performance Testing

The main goal of it is to test our system when it is under high load , to test if there is any crashing or slowing down.

4.8 Comparison Between AI Models:

We have tested two different AI models during the development of our system:

1. **LLaMA AI Model:** At the start it was a very good open source AI model , but we had multiple problems like no quick responses , and it was not accurate , and sometimes our comparing feature did not work with this model. We tried to fix it but we didn't figure it out. That was the first version of our project.
2. **GPT-4o (OpenAI):** Then we found that the GPT-4o paid ai model, is better with our project , more accurate , very quick responses and multitask capabilities is much higher , we had slow responses with the previous ai model and not accurate for summarization and for answering , and with our comparing feature , GPT-4o was much accurate at differences and can answer quickly and reliably.

4.8.1 YouTube URL Input and Validation:

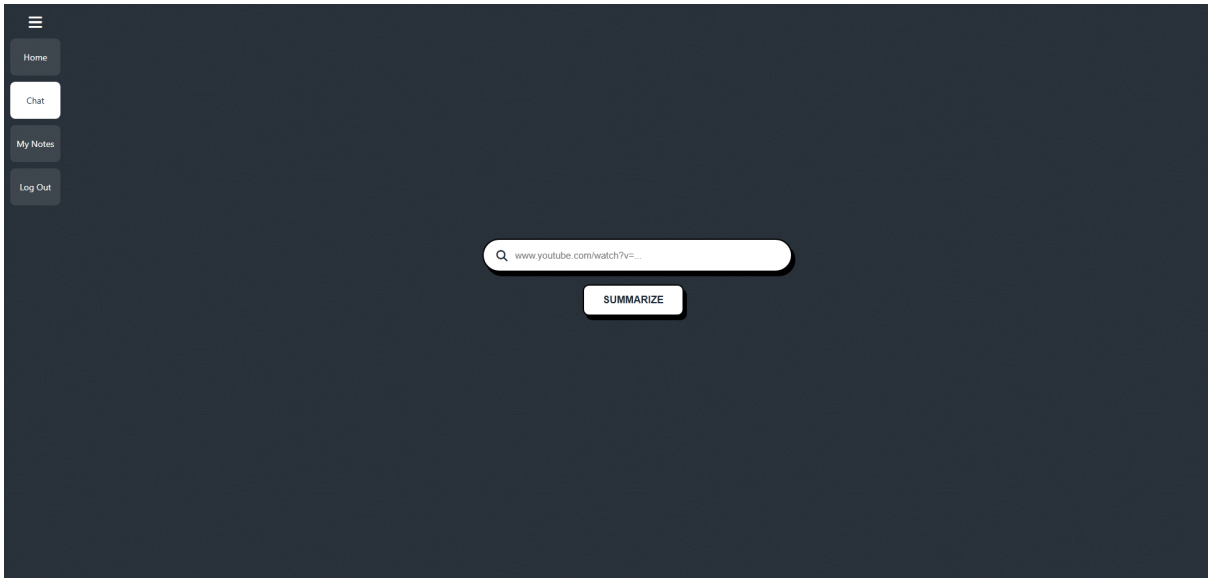


Figure 4.8: YouTube URL Input Interface with Validation

URL Validation:
In this interface the user must enter a valid YouTube video URL, old URLs will not work only the videos with transcript (subtitles). If the link is invalid the button will not be clickable.

Proceeding to Summarization:
When the URL is true the user must click on the summarize button that will take him to the next interface and will talk about it.

4.8.2 Interactive QA Interface:

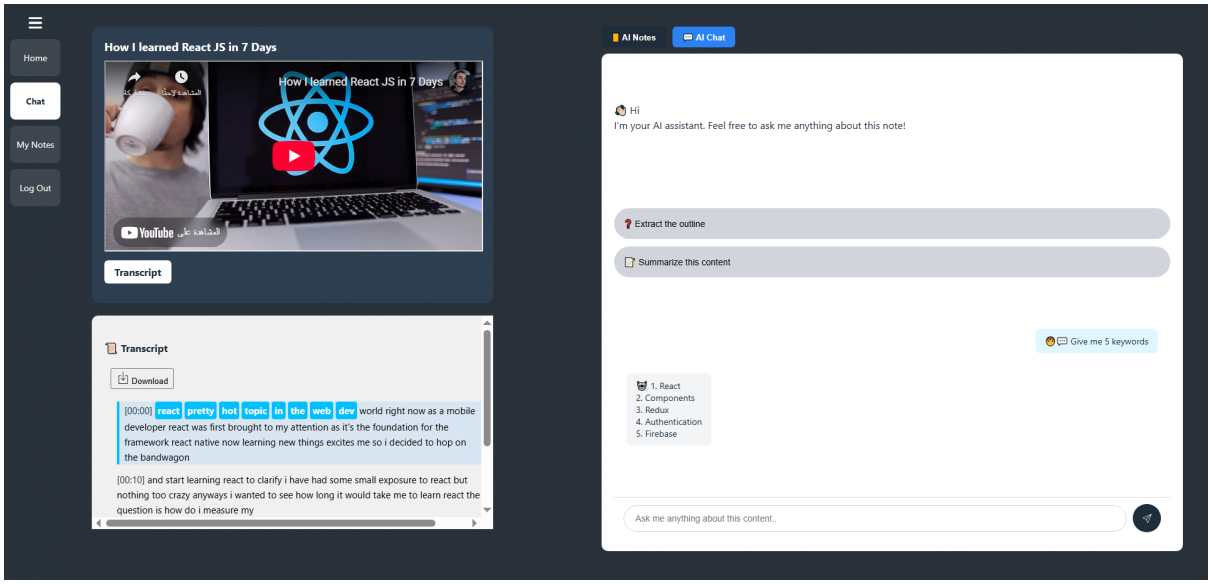


Figure 4.9: AI Chat Interface for Answering Questions Based on YouTube Transcript

AI Notes Section:
When the user clicks on Ai chat button that will move him the the ai chat interface , in this interface the user can ask about anything related to the youtube video , user might ask about .

4.8.3 Summarization Result Interface:

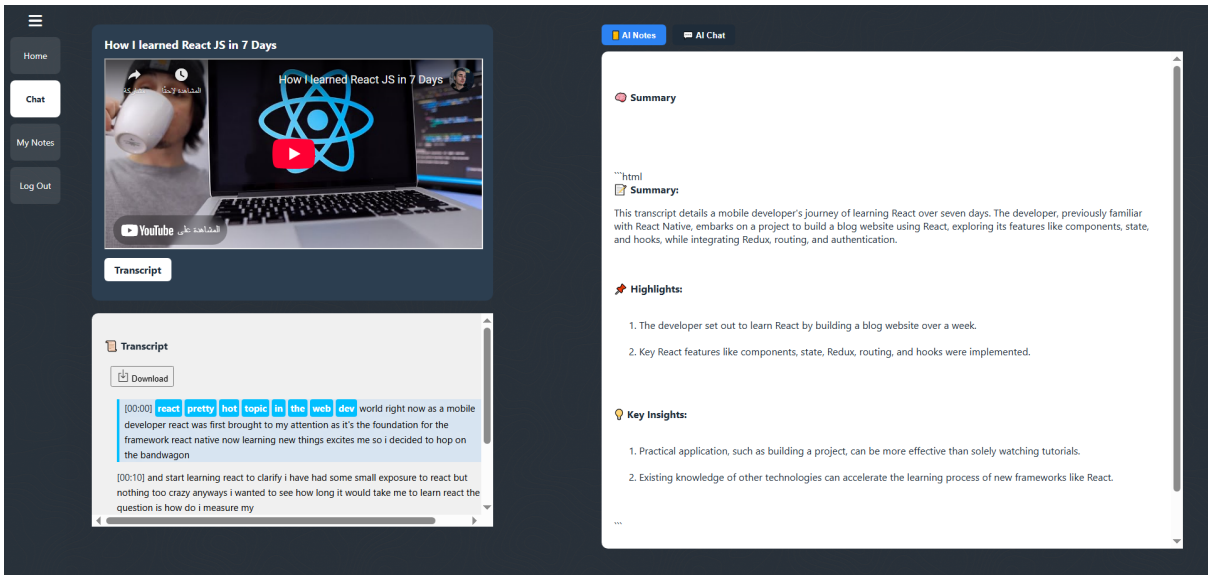


Figure 4.10: Video and Transcript Display Interface

Interface Overview:
This will be the interface when the user click on the summarize button, the left side of the interface it will show the YouTube video thumbnail and you can start the video and scroll in the timeline and there is a box that shows the text in each minute with scrolling and it will highlight the text for the user as shown in the picture, that will make it easier for showing the transcript.

AI Notes Section:
In the right side of the interface there are two options, the first one it will be the AI notes that will include as shown, summary, highlights, key insights.

4.8.4 My Notes Interface with Video Management:

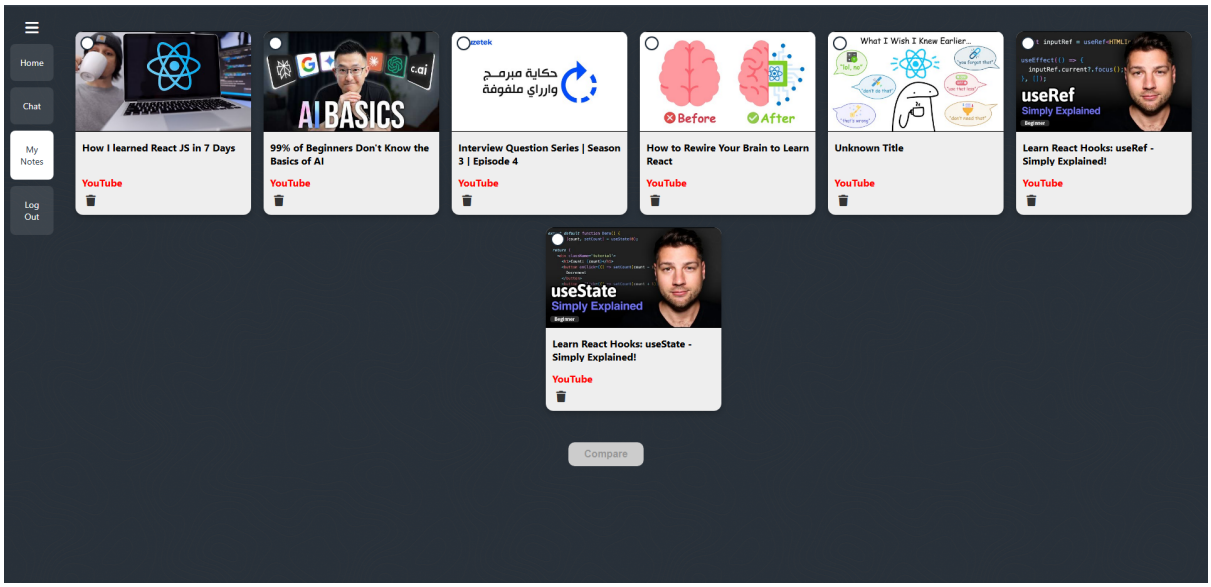


Figure 4.11: My Notes Interface with Video Management

My Notes Interface Overview:
Here is My notes interface, that interface includes the videos that the user entered in the first interface, it's related to the user input.

Comparison and Deletion Features:
There is a compare button, the first state of it is transparent and it's not clickable until the user selects two videos for comparison. In each video there is a delete button that deletes the specific video the user selected.

4.9 Evaluation

Evaluating System Features

In this section, we evaluated our system using each module. Each module has its tasks, and the success or failure of each task was recorded.

A check mark (✓) indicates success, while a cross (✗) indicates failure. Tasks that initially failed but succeeded after retries are marked as (✗ ✗).

Transcript Extraction Module:

Task ID	Task	Status
1.1	Extract transcript from YouTube video URLs	✓
1.2	Handle videos without transcripts gracefully	✗ ✗
1.3	Parse video URLs accurately	✓

Table 4.2: Transcript Extraction Evaluation

Summarization Module:

Task ID	Task	Status
2.1	Generate accurate summaries from transcripts	✓
2.2	Highlight key insights and main points	✓
2.3	Support multiple languages	✓

Table 4.3: Summarization Module Evaluation

Video Comparison Module:

Task ID	Task	Status
3.1	Compare two videos for similarities and differences	✓
3.2	Display comparison results clearly	✓

Table 4.4: Video Comparison Evaluation

4.9.1 Evaluation Statistics

The table below summarizes success and failure rates for the main modules of the system, reflecting its overall reliability and performance:

Module	**Success Rate**	**Failure Rate**
Transcript Extraction	90%	10%
Summarization	95%	5%
Video Comparison	85%	15%

Table 4.5: Module Success and Failure Rates

These results are based on practical testing of the system. For transcript extraction, the system succeeded in 45 out of 50 cases (90%), with failures occurring mainly in videos without subtitles. Summarization achieved a (95%) success rate, producing clear and accurate summaries in most cases, while three videos had incomplete outputs. Video comparison worked well in 34 out of 40 cases (85%), but struggled in 6 cases where the videos had very different structures or topics. Overall, the evaluation shows that the system is reliable and efficient, though future improvements are needed to handle videos without transcripts and to refine comparison accuracy.

Chapter 5

Conclusion

5.1 Summary:

This project successfully developed a website that uses OpenAI's GPT-4o, it's a youtube video ai summarizer and a compare tool to compare between two or more video on youtube , the user can see the youtube video and the timeline and scroll it and see every moment and it's text ,it's important for someone who's searching for specific information in a video to or summarize the whole video script or compare between different videos to see the similarities and differences between them , it's a way to consume short time for researchers or student , video content creators to study or search for information , this project will help them extract their points form the video in the way that they wanted , in the future we are working on doing a new feature that can merge two videos that talked about the same topic , and extract the wanted point using the ai model that we are using , to add more impact we might dive into different ai systems that could help with other things such as develop a mobile application so that the user can use it anywhere without using the web

5.2 Future Work:

Future improvements and additions to the system may include:

- In the future we are working on doing a new feature that can merge two videos that talk about the same topic. This feature will extract the wanted point using the AI model that we are using.
- To add more impact we might dive into different AI systems that could help with other things, such as AI models that can listen to the video and watch it and analyze the audio and the visuals.
- Maybe there will be a video that has graphs, the AI model can extract them as they are and what the user requires will answer.
- We might develop a mobile application so that the user can use it anywhere without using the web.

References

[1] YouTube Transcript API Documentation. https://github.com/jdepoix/youtube-transcript-api

[2] Meta AI, LLaMA Model Card. https://ai.meta.com

[3] OpenAI API Documentation. https://platform.openai.com/docs

[4] YouTube Data API v3. https://developers.google.com/youtube/v3

[5] Python Software Foundation, Python Language Reference. https://www.python.org/doc/

[6] Flask Documentation. https://flask.palletsprojects.com/en/latest/

[7] SQLite Documentation. https://www.sqlite.org/docs.html

[8] Python Software Foundation, *urllib.parse — Parse URLs into components*, Python 3 Documentation. https://docs.python.org/3/library/urllib.parse.html

[9] Mozilla Contributors, *JavaScript Guide*, MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

[10] Shuyo, Nobuyuki, *langdetect: Language detection library for Python*, GitHub repository. https://github.com/Mimino666/langdetect

[11] NoteGPT, *AI-powered tool for summarizing and organizing notes*, Available at: https://notegpt.io/