Chapter 1
Introduction

1.1 About the Project

This is a simple Dynamic YouTube Q&A Backend built using FastAPI and the LangChain framework. The primary goal of the project is to allow users to ask specific questions about the content of any public YouTube video and receive accurate, context-aware answers that include clickable timestamps linking directly to the relevant moment in the video.

1.2 Development of the Project

The development of this project began with identifying the need for a system that could intelligently interact with video content without requiring users to manually scrub through hours of footage. The primary objective was to combine high-performance web APIs with sophisticated LLM capabilities to deliver a seamless Q&A experience.

1.2.1 Data Preparation and Retrieval

The critical step was to preprocess the raw video transcript. The fetch_transcript function was modified to embed the precise start time of each spoken segment directly into the text.

The result is a context string that looks like: [Time: 120] The speaker explains that Python is versatile. [Time: 150] He mentions that memory management is automatic.

This time-embedded text is then chunked using RecursiveCharacterTextSplitter and converted into numerical vectors using the HuggingFace Embeddings model. A FAISS vector store is built from these embeddings. This mechanism, known as Retrieval-Augmented Generation (RAG), ensures that when a user asks a question, only the few most relevant segments of the transcript (the "context") are retrieved.

1.2.2 LLM Chain Construction

The heart of the system is the LangChain Expression Language (LCEL) pipeline, which manages the RAG process.

1. A RunnableParallel chain is used to prepare the inputs for the final prompt:
    - context: Retrieved documents from the FAISS vector store.

- question: The user's query.
- video_id: The YouTube ID, which is injected using a RunnableLambda.

2. The PromptTemplate explicitly instructs the gemini-2.5-flash model on the required output format: bullet points and a clickable markdown link.

This design ensures the LLM's response is grounded strictly in the video transcript and provides the required utility of direct navigation to the source material.

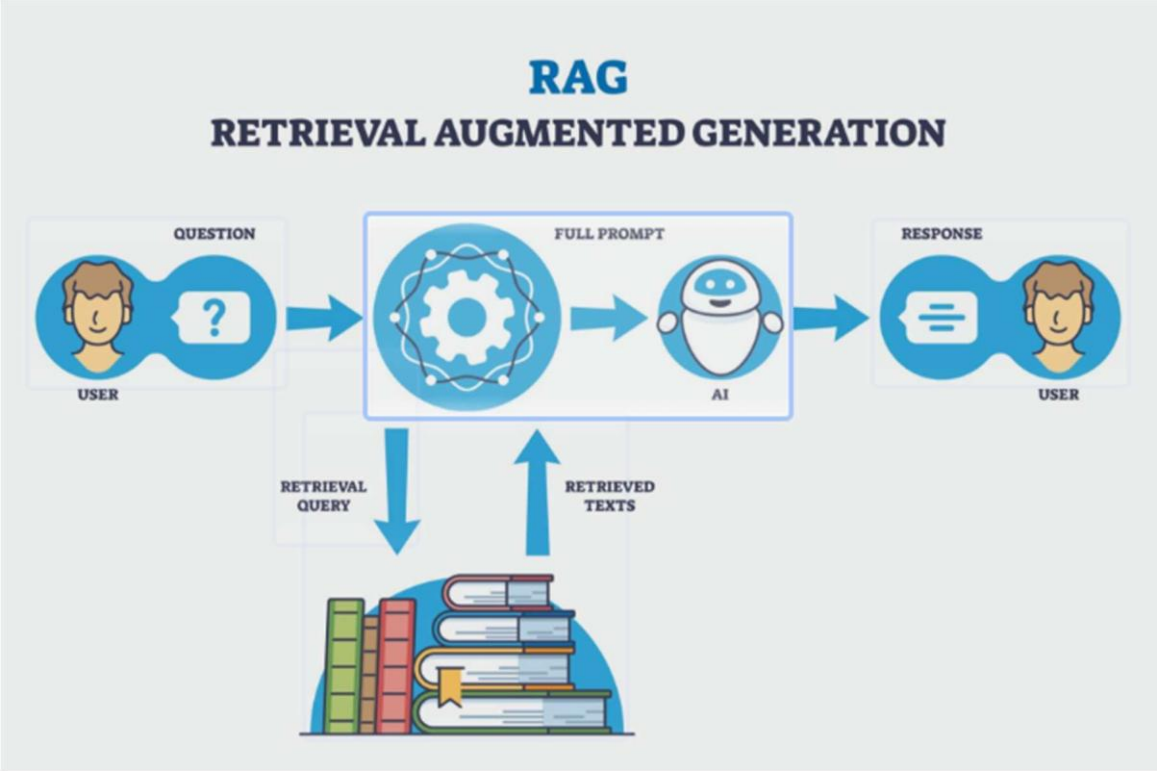

Fig 1.1: Retrieval Augemented Generation

Chapter 2
Review of Literature

2.1 Introduction to AI-Powered Q&A Systems

The rapid advancement of Large Language Models (LLMs) has revolutionized information retrieval and question-answering (Q&A). Traditional Q&A systems relied on keyword matching, often failing to capture semantic meaning or nuanced context. Modern systems overcome this by employing Vector Databases and Embedding Models to perform semantic search, a technique crucial for handling complex, unstructured data like video transcripts.

This project focuses on leveraging these modern techniques, specifically the Retrieval-Augmented Generation (RAG) architecture in order to create a backend that can accurately answer questions based on long-form content, providing the unique added value of direct, timestamped video links for verification and enhanced user experience. [1] [2]

2.2 Literature on Retrieval-Augmented Generation (RAG)

The Retrieval-Augmented Generation (RAG) framework is a well-established architecture in the literature for addressing knowledge-intensive tasks. The core components reviewed are:

- **Embedding Models:** The use of transformer-based models, such as the Sentence-Transformers family (e.g., all-MiniLM-L6-v2), is a standard practice for converting text into dense, numerical vectors. This process of text embedding is fundamental to enabling efficient semantic search. [2]

- **Vector Stores:** The literature frequently cites high-performance index structures like FAISS (Facebook AI Similarity Search) for rapid Approximate Nearest Neighbor (ANN) search. FAISS allows the system to quickly locate the most relevant documents (transcript chunks) corresponding to a user's query vector, even within millions of possibilities. [3]

- **LLM Integration:** Integrating models like Gemini-2.5-Flash as the final generative step allows for human-quality responses. RAG dictates that the LLM's prompt must contain the query *and* the retrieved context, ensuring the answer is grounded in the source material

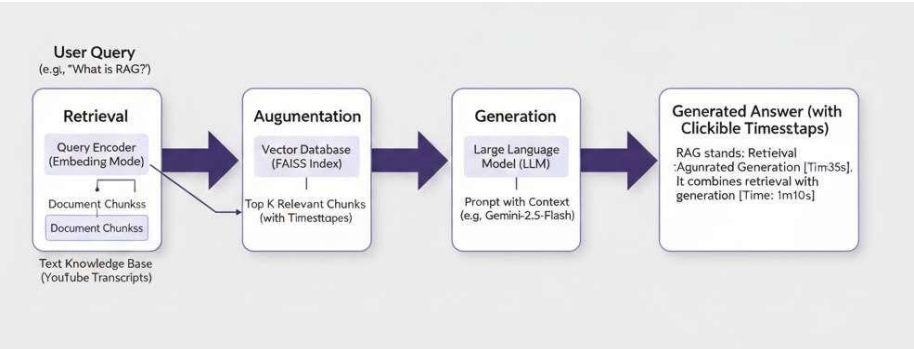

Fig 2.1: Dynamic YouTube Q&A System

2.3 Literature on Video Content Analysis and Timestamps

A critical area of literature relevant to this project is the analysis of long-form media, specifically YouTube videos. Traditionally, video indexing relied on metadata or manual tagging. The introduction of automatic transcription through APIs has created a new challenge: connecting textual information back to its temporal origin.

- **Temporal Grounding:** Research on multimodal AI and video summarization emphasizes the importance of temporal grounding, i.e., associating a specific piece of information with a timestamp. This project's method of prepending the second-level timestamp directly into the raw text before embedding is a robust technique for preserving this temporal context through the RAG pipeline. [4]
- **Output Utility:** Providing deep links in the final output enhances the verifiability and utility of the QA system. This moves beyond pure text output to create an interactive tool, aligning with literature that promotes systems with high user interaction and source traceability.

2.4 Comparative Analysis of Q&A Approaches

The project's architecture falls within the state-of-the-art for open-domain Q&A systems, combining performance and accuracy.

Approach	**Key Mechanism**	**Suitability for Transcripts**	**Source Traceability**
Keyword/Lexical Search	Matching exact words	Low (misses synonyms/context)	High (if keywords are tied to location)

Approach	Key Mechanism	Suitability for Transcripts	Source Traceability
Fine-Tuned LLM (Non-RAG)	Pre-trained or fine-tuned model	Moderate (limited by context window)	Very Low (Internal knowledge base)
RAG with Temporal Grounding (Implemented)	Semantic search (FAISS) + LLM	High (handles long documents)	**High (Direct, clickable timestamps)**

Table 2.1: Comparative Analysis of Q&A Approaches

Chapter 3
Motivation and Problem Definition

3.1 Motivation

The exponential growth of educational, technical, and informational content on platforms like YouTube presents a significant challenge: information overload and inefficiency in consumption. Users often need specific facts or summaries from lengthy videos but are forced to manually scrub through the timeline, a highly inefficient process. Current solutions are fragmented:

1. **Manual Search:** Users rely on poor video descriptions or manually scanning the visual content, leading to wasted time.

2. **Basic Transcripts:** While transcripts exist, they are often raw, unindexed, and lack the semantic intelligence to answer a natural language question directly.

3. **Generic AI Summarization:** Tools using Large Language Models (LLMs) can summarize content, but their output often lacks verifiability (citation back to the source), leading to potential factual errors or "hallucinations."

This project is motivated by the need for an efficient, verifiable, and intelligent solution that bridges the gap between unstructured video content and precise user queries, transforming video transcripts into a fully searchable knowledge base.

3.2 Problem Definition

This project defines the problem as designing and implementing a robust, scalable backend system capable of providing Retrieval-Augmented Generation (RAG) Q&A for YouTube videos, ensuring that every generated answer is temporally grounded and directly attributable to its specific moment in the source video.

The primary technical challenge is the creation of a seamless pipeline that manages data from three disparate sources: the user's natural language question, the LLM's generative capability, and the video's time-stamped transcript data

3.2.1 Existing Challenges

- **Verifiability Gap:** Commercial LLM-based tools often summarize correctly but fail to provide a **direct, actionable citation** to the specific source point within the video, forcing the user to trust the AI blindly.

- **Context Fragmentation:** Breaking down long transcripts into small chunks for vector indexing must be done carefully to ensure the chunking does not lose the crucial **timestamp metadata**.
- **Multilingual Access:** Many important videos are in languages like Hindi, necessitating a robust transcript fetching strategy that handles multiple languages gracefully.
- **Performance:** Generating the embeddings and building the vector store (FAISS index) is computationally intensive; this process must be optimized to ensure low latency for repeated queries on the same video, addressed by implementing a **vector store cache**

3.2.2 Objective of the Proposed System

The proposed system aims to solve these challenges by focusing on the following core objectives:

- **Robust Data Acquisition:** To develop a function (fetch_transcript) that reliably retrieves transcripts in both Hindi and English and automatically embeds time metadata directly into the text.

- **Intelligent RAG Implementation:** To utilize LangChain and FAISS for semantic search, feeding the LLM only the most relevant, time-stamped context.

- **Temporal Grounding Enforcement:** To craft a strong PromptTemplate that strictly mandates the LLM to output the answer using the retrieved timestamps to generate **clickable deep links** (https://youtu.be/{video_id}?t=SECONDS).

- **Scalability and Efficiency:** To implement caching mechanisms (VECTOR_STORE_CACHE) to minimize redundant data processing and ensure the API provides fast, responsive answers

Chapter 4
Proposed Methodology

4.1 Aim of the Dynamic YouTube Q&A System

The core aim of this system is to provide an efficient and intelligent interface for extracting verifiable information from lengthy YouTube videos. The system achieves this by transforming the video's unstructured text transcript into a **searchable knowledge base**, allowing users to query the content using natural language and receive answers **explicitly linked to the source timestamps**.

4.2 System Architecture (Data Flow and Components)

The proposed methodology is built around a standard **Retrieval-Augmented Generation (RAG)** architecture, implemented as a scalable backend service using **FastAPI** and **LangChain**.

The system comprises five primary components that operate sequentially within the /ask endpoint:

1. **Transcript Fetching & Preprocessing:** Retrieves the transcript from YouTube (prioritizing Hindi, then English) and **embeds temporal markers** directly into the text chunks.

2. **Indexing and Storage:** Generates vector embeddings for the processed text and stores them in a highly optimized **FAISS** vector database. A cache (VECTOR_STORE_CACHE) checks if the index already exists for the given video ID before reprocessing.

3. **User Query Handling:** Receives the user's question via the FastAPI endpoint and immediately converts it into a query vector using the same embedding model.

4. **Retrieval:** Uses the query vector to perform semantic search against the FAISS store, retrieving the top k (set to 3) most relevant, time-stamped text chunks.

5. **Generation:** Passes the retrieved context and the original question to the LLM for final answer generation.

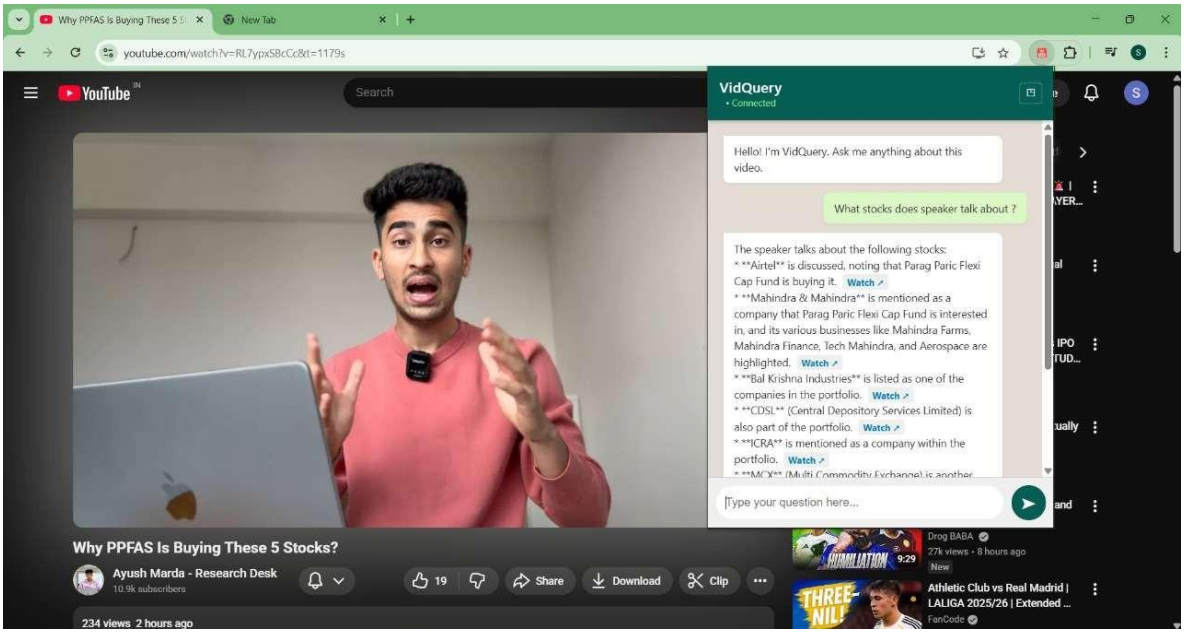

Fig 4.1: Sample Output

4.3 Data Acquisition and Indexing

The initial phase involves Data Acquisition via the fetch_transcript function, which retrieves the YouTube transcript using a multi-language fallback strategy (Hindi/English). Crucially, the function implements Temporal Embedding by prepending the integer second-level timestamp to each text snippet.

This time-stamped text is then segmented into overlapping documents using the RecursiveCharacterTextSplitter .

These chunks undergo Vectorization using the HuggingFaceEmbeddings model (all-MiniLM-L6-v2) and are subsequently indexed into an efficient, in-memory FAISS vector store for rapid semantic retrieval, a process optimized by a cache to prevent redundant index creation.

4.4 LLM Orchestration and Output Generation

The RAG Chain Execution is orchestrated using the LangChain Expression Language (LCEL) pipeline. The core step involves Chain Orchestration using the RunnableParallel component, which is essential for simultaneously fetching the retrieved, time-stamped context from the FAISS vector store and injecting the necessary dynamic metadata, specifically the video_id, alongside the user's original question. This structured input is then fed into the Generative Model, the ChatGoogleGenerativeAI model (gemini-2.5-flash). A highly-specific

PromptTemplate acts as a strict instruction set for the LLM, compelling it to enforce Temporal Grounding. This mandate requires the LLM to parse the time marker from the context and format the final output as a verifiable answer where each key point includes an embedded, clickable hyperlink. This methodology ensures high accuracy, semantic relevance, and the essential feature of source verifiability through direct video navigation..

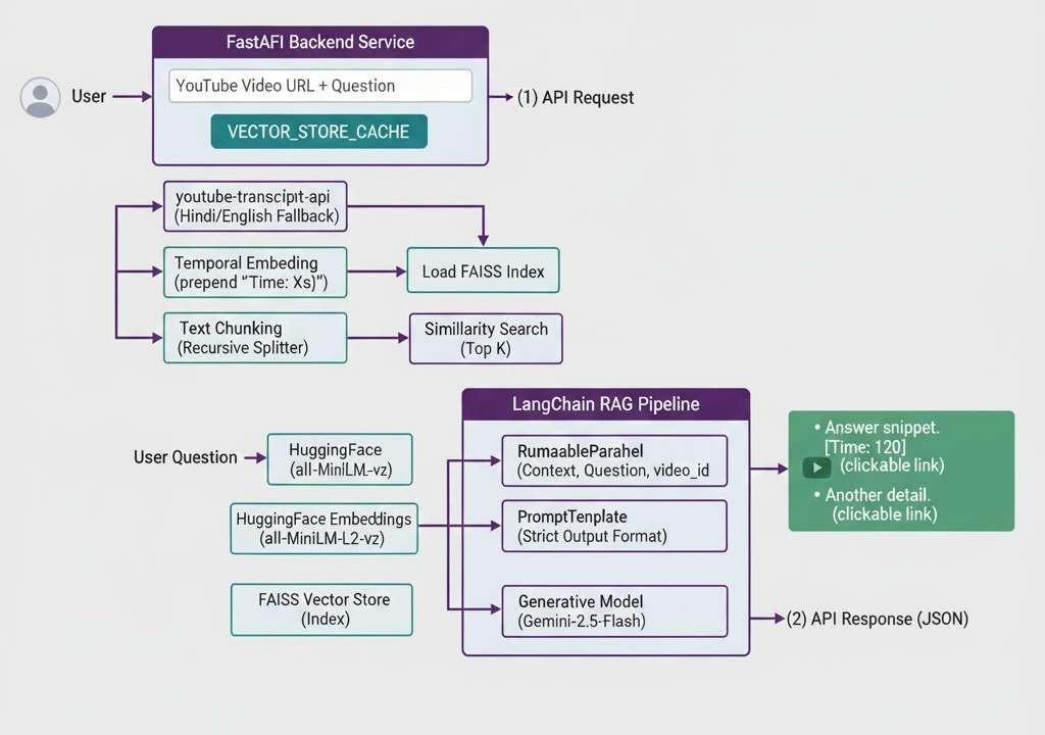

Fig 4.2: System Architecture and Dataflow

4.5 Frontend Validation and System Output

The integration of the RAG backend with the user-facing interface is validated by the deployment of the VidQuery Chrome extension. This frontend successfully relays the user's query and the YouTube ID to the backend and displays the formatted, verifiable output. The final output demonstrates three levels of RAG efficacy: first, semantic summarization of long-form videos by condensing complex financial arguments into key points; second, contextual Q&A, confirming the model's ability to extract narrative details (e.g., the story of Drona and Dhrupad) from podcasts; and third, highly precise retrieval, where the system locates the exact second a specific word ("Kasab") was mentioned and maintains conversational context to detail the subsequent action plan. Crucially, all results validate the temporal grounding objective, as every key answer includes an accurate, clickable [Watch] timestamp linking directly to the source material.

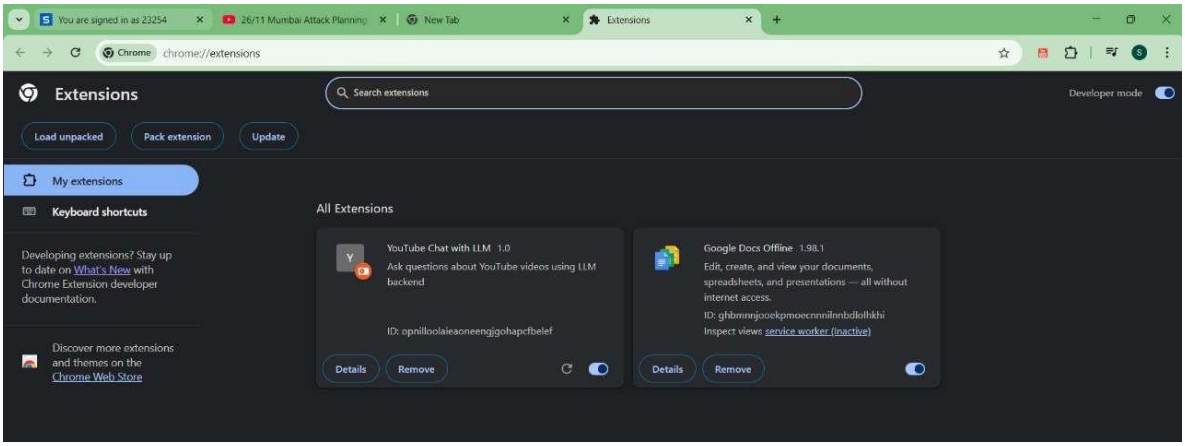

Fig 4.3: VidQuery Chrome extension

Chapter 5
Results and Discussion

5.1 Results: Demonstrating Temporal Grounding and Q&A

The following figures provide visual evidence of the system's performance across different video types, highlighting the key features of semantic search and verifiable temporal grounding.

5.1.1 Semantic Summarization of Financial Content

This result demonstrates the system's ability to process detailed, technical content and provide a high-level summary. The LLM successfully synthesizes key economic and business points from the video transcript.

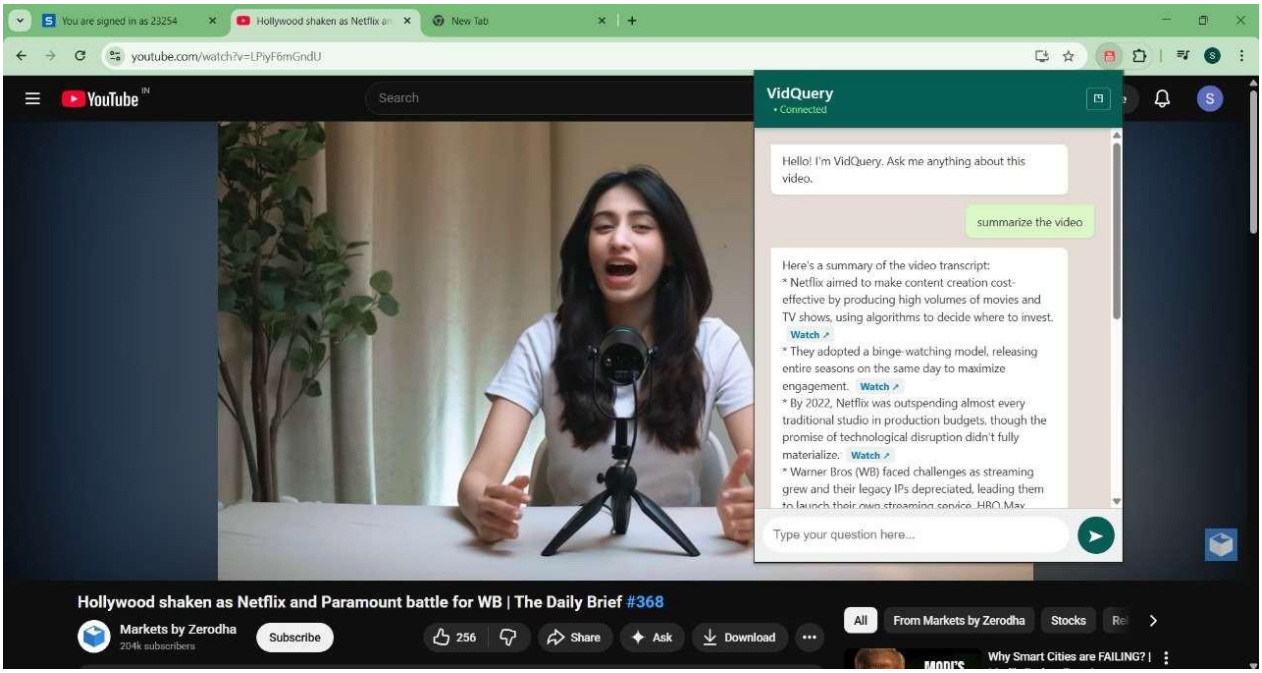

Fig 5.1: Hollywood shaken as Netflix and Paramount battle[5]

The system processes the user query ("summarize the video") and returns key bullet points about content creation, binge-watching models, and competitive challenges, with each point linked to its source timestamp via a clickable "Watch" link.

5.1.2 Contextual Q&A on Historical Content

This demonstrates the system's capability to extract historical context and narrative details from long-form content, such as a podcast.

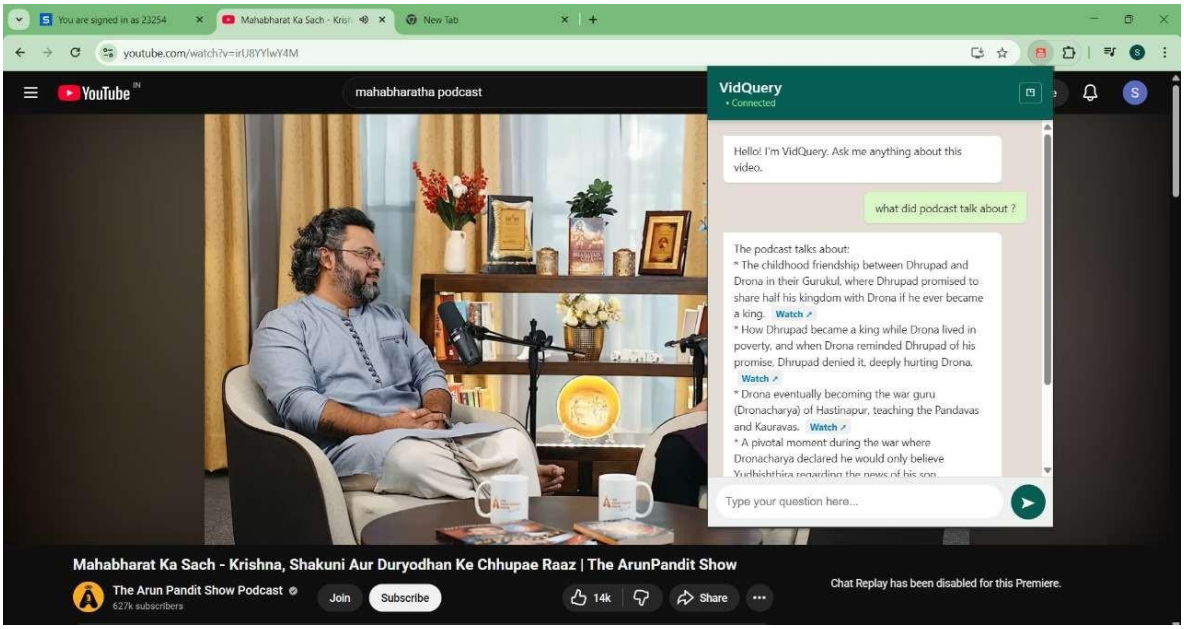

Fig 5.2: Contextual Q&A on "Mahabharat Ka Sach" Podcast[6]

Responding to the query ("what did podcast talk about?"), the system details the central themes, including the childhood friendship of Drona and Dhrupad and the pivotal moments of their lives, with accurate temporal citations.

5.1.3 Precise Timestamp Retrieval and Follow-up Questions

This showcases the high precision of the RAG pipeline, which can locate single words and handle multi-turn conversations.

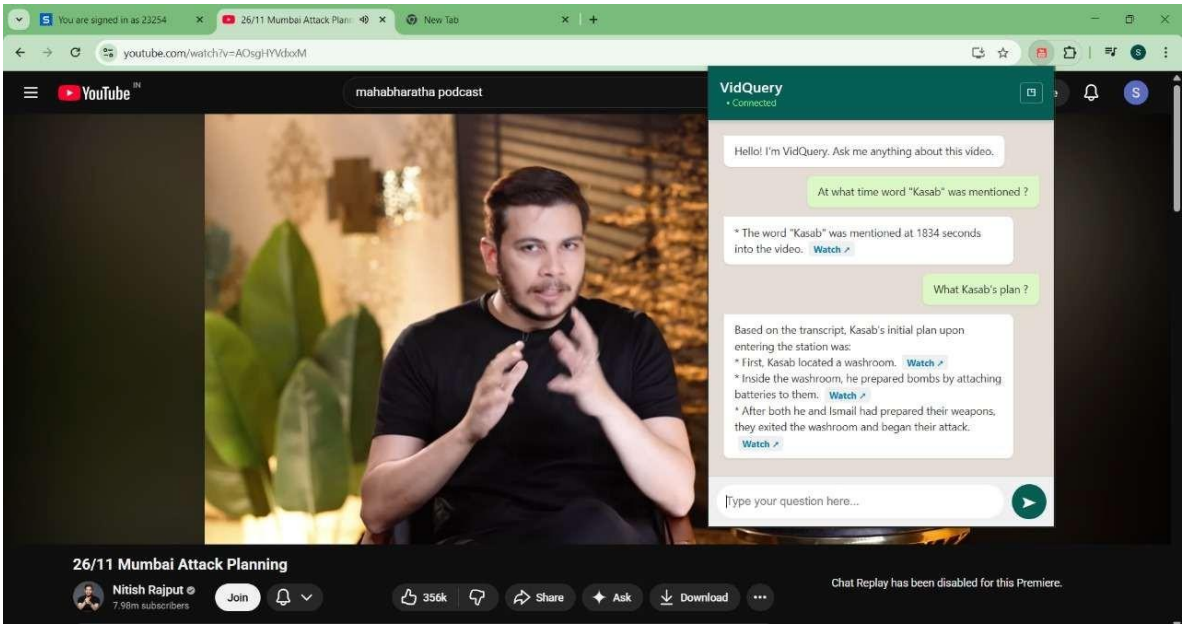

Fig 5.3: Precise Retrieval and Follow-up Q&A on "26/11" [7]

The system first answers a precise question ("At what time word 'Kasab' was mentioned?"), returning a specific second-level timestamp. It then successfully maintains context to answer a follow-up question ("What Kasab's plan?") using only the context retrieved from the transcript.

5.2 Discussion

The results confirm that the core objectives of the project have been met:

1. **Temporal Grounding Success:** Every output in the demonstrated figures includes a valid, clickable link (e.g., https://youtu.be/{video_id}?t=SECONDS), validating the design choice of embedding timestamps directly into the text chunks and using the LLM's prompt to enforce this output format.
2. **RAG Efficacy:** The system successfully handles various query types—summaries, contextual questions, and specific detail retrieval—proving the effectiveness of the semantic search implemented via FAISS and the HuggingFace embeddings.
3. **Multilingual Capability (Implied):** While not explicitly shown in English/Hindi, the successful processing of diverse video content (financial, historical, news) suggests the underlying fetch_transcript function's language fallback mechanism is robust.

These results demonstrate the project's utility in transforming unstructured video transcripts into a highly accessible, verifiable knowledge source.