© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

Automated Technical Documentation Generator from YouTube Video URL: A Multi-Module AI-Powered Web Application

Alex Franklin M[1], Dr. K. Ponmozhi[2]
[1] PG Student, Master of Computer Applications, SRM Valliammai Engineering College, Kattankulathur.
[2] Associate Professor, Department of Computer Applications, SRM Valliammai Engineering college, Kattankulathur.

Abstract- ***The rapid proliferation of video-based learning content on platforms such as YouTube has created significant challenges for students, developers, and professionals who need to extract, retain, and reference technical knowledge from long-form instructional videos. Manual documentation is costly, existing summarization tools lose technical fidelity, and no current system addresses the full pipeline from a raw video URL to formatted, downloadable documentation. This paper presents an Automated Technical Documentation Generator, a nine-module AI-powered web application that accepts any YouTube URL and produces structured professional documentation in under 90 seconds. The system comprises modules for transcript extraction with automatic chunking, AI-based content type classification, LLM-driven chunk-and-merge documentation generation, a multi-turn Q&A chatbot, multi-format export in PDF, DOCX, and HTML, a usage history and analytics dashboard, a goal-driven Video Recommender, and a Cognitive Load Analyzer grounded in Sweller's Cognitive Load Theory. An automatic fallback router ensures near-100% availability across eight free LLM providers. Results on twenty test videos across five content types show transcript coverage exceeding 93% and strong alignment between Cognitive Load scores and independent educator assessments.***

Index Terms: ***Automated Documentation, Large Language Models, YouTube Transcript Processing, Cognitive Load Theory, Natural Language Processing, AI-Powered Education Tools, Knowledge Gap Detection, Video Content Analysis, Streamlit, LLM Fallback Routing.***

I. INTRODUCTION

In recent years, platforms such as YouTube have emerged as primary repositories of technical and educational content. Millions of tutorial videos, lecture recordings, and developer walkthroughs are published daily, creating an increasingly rich yet difficult-to-navigate body of knowledge.

For learners who require structured, referenceable documentation whether for revision, team sharing, or professional reference watching a video in its entirety and manually transcribing key information represents a significant productivity barrier.

Existing video summarization tools tend to produce shallow paragraph-level abstracts that strip away the technical detail, code examples, and logical sequencing that make instructional content useful. Furthermore, no publicly available end-to-end pipeline currently converts a raw YouTube URL into professionally formatted, downloadable documentation of arbitrary length.

This paper introduces the Automated Technical Documentation Generator, a nine-module web application that bridges this gap. The system extracts the video transcript, classifies its content type, generates structured documentation using large language models (LLMs), and delivers output in PDF, DOCX, or HTML format. Beyond documentation generation, the system provides an interactive Q&A chatbot, a goal-driven Video Recommender, and a Cognitive Load Analyzer each addressing a distinct step in the video-based learning workflow.

II. LITERATURE REVIEW

Prior work on automated documentation and video content processing spans several related domains. Alsubaihin et al. [1] established that structured template-based approaches consistently outperform

free-form summarization for technical content generation. In video processing, Furini et al. [2] demonstrated that transcript-based summarization can capture the semantic content of educational videos when supported by topic modeling. Zhang and Patel [3] extended this to multi-segment videos and identified chunk-boundary artifacts as a primary source of output degradation. Chen et al. [4] showed that prompt-engineered LLMs can produce tutorial-style documentation from raw transcripts, though their approach was restricted to videos under 30 minutes. Handling longer content remained unsolved until map-reduce style pipelines were proposed for retrieval-augmented generation [5]. The present chunk-and-merge design draws on this prior art while adapting it specifically to the documentation generation task.

Cognitive load in digital learning has been studied extensively since Sweller's foundational work [6]. Leppink et al. [7] proposed computational proxies for cognitive load in text, and Muñoz-Merino et al. [8] applied similar metrics to MOOCs. However, no prior work applies automated cognitive load measurement directly to YouTube video transcripts. Knowledge gap detection in intelligent tutoring systems [9] and goal-driven recommendation [10] have also been explored separately, but not integrated into a unified video processing pipeline.

III. PROBLEM STATEMENT

The rapid growth of technical video content on YouTube has created challenges in efficiently extracting and utilizing knowledge. Despite the availability of transcripts and summarization tools, several key limitations remain.

- First, manual documentation is time-consuming and inefficient. Users must invest significant effort to watch, interpret, and organize content into structured notes, leading to reduced productivity.
- Second, existing summarization tools lack technical fidelity. They often generate high-level summaries that omit essential elements such as code snippets, commands, and step-by-step procedures, making them unsuitable for technical learning.
- Third, video transcripts vary widely in length, often exceeding the processing limits of Large Language Models (LLMs). Current systems fail to effectively handle long transcripts while maintaining context and coherence.
- Finally, learners have no mechanism to evaluate the difficulty or prerequisites of a video before watching it. The absence of cognitive load assessment leads to inefficient content selection and suboptimal learning experiences.

Therefore, there is a need for an integrated system that can automatically convert YouTube videos into structured, high-quality technical documentation while supporting large-scale transcript processing and providing insights into content complexity.

IV. OBJECTIVE

1)Automate extraction of structured documentation from any YouTube video URL – To develop a system that takes a YouTube URL as input and automatically generates well-organized, structured technical documentation without manual effort.

2)Support arbitrary video length through a chunk-and-merge pipeline – To handle videos of any duration by splitting large transcripts into manageable chunks, processing them individually, and merging outputs while preserving context and coherence.

3)Classify video content type and apply appropriate documentation templates – To identify the type of video (e.g., tutorial, lecture, walkthrough) and generate documentation using predefined templates tailored for each content category.

4)Provide an interactive multi-turn Q&A chatbot grounded in video content – To enable users to ask questions related to the video and receive accurate, context-aware answers based on the generated documentation and transcript.

5)Export documentation in PDF, DOCX, and HTML formats – To allow users to download the generated documentation in multiple standard formats for easy sharing, storage, and offline access.

6)Maintain a usage history dashboard with time-saved analytics – To track user activity, store previously processed videos, and provide insights such as estimated time saved through automation.

7)Rank multiple candidate videos against a user-specified learning goal – To recommend and rank videos based on how well they match a user's learning objective, improving content selection efficiency.

8)Quantify cognitive load of video content across five measurable dimensions – To evaluate the difficulty level of a video using measurable indicators based on cognitive load theory, helping users assess complexity before watching.
9)Identify knowledge gaps and generate targeted prerequisite search queries – To detect missing foundational concepts required to understand the video and suggest relevant prerequisite topics for better learning preparation.
10)Ensure near-100% availability through automatic LLM fallback routing – To maintain system reliability by automatically switching between multiple LLM providers in case of failures or rate limits.

V. SYSTEM ANALYSIS

System analysis focuses on understanding the limitations of existing approaches and identifying the requirements for an improved solution. In the context of video-based learning, it involves analyzing how knowledge is currently extracted, processed, and utilized by learners. This section evaluates the shortcomings of existing systems and highlights the need for an automated, scalable, and efficient framework. Based on this analysis, a proposed system is designed to overcome these limitations and provide a comprehensive solution for generating structured technical documentation from YouTube videos.

5.1 Existing System:

Current approaches to video-based learning rely heavily on manual effort throughout the knowledge-extraction pipeline. Learners are required to watch videos, take notes, and organize content manually, which is time-consuming and inefficient. Available summarization tools generate only short, paragraph-level abstracts that lack technical depth and often omit essential elements such as code snippets, commands, and procedural steps. Moreover, there is no integrated system that converts a raw YouTube video URL into structured, professionally formatted, multi-format downloadable documentation.

5.2 Limitations of Existing System:

6 Manual extraction is time-intensive and does not scale effectively across large video libraries.
7 Summarization tools lose technical fidelity, omitting code blocks, command-line instructions, and step-by-step procedures.
8 No system efficiently handles long transcripts that exceed the context window of Large Language Models (LLMs).
9 There is no mechanism to evaluate the cognitive difficulty of a video before watching.
10 Learners are unable to compare multiple videos based on a specific learning goal.

10.1 Proposed System:

The proposed Automated Technical Documentation Generator addresses these limitations through a nine-module architecture. The system accepts a YouTube URL, extracts the transcript, and processes it using a chunk-and-merge strategy to handle videos of any length. It classifies the content type and generates structured technical documentation using Large Language Models.

The output is provided in multiple formats, including PDF, DOCX, and HTML. Additionally, advanced modules such as the Cognitive Load Analyzer and Video Recommender enhance the system by supporting informed learning decisions and personalized content selection.

10.2 Advantages of Proposed System:

- Provides a complete pipeline from YouTube URL to structured documentation in under 90 seconds.
- Supports videos of any length using an efficient chunk-and-merge processing approach.
- Offers multiple export formats (PDF, DOCX, HTML) for flexible usage.
- Enables cognitive load analysis to help users assess video difficulty before watching.
- Ensures high system reliability through automatic fallback across multiple LLM providers.
- Includes a centralized dashboard for tracking usage history and productivity improvements.

10.3 System Design:

The proposed system is structured into six functional layers, as illustrated in Fig. 1.0, to ensure modularity, scalability, and efficient processing. The input layer accepts a YouTube video URL through a user-friendly Streamlit web interface, serving as the entry point to

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

the system. The transcript extraction layer retrieves the video transcript, performs cleaning operations such as noise removal and formatting, and applies chunking when handling large content.

The AI processing layer forms the core of the system, where key operations such as content classification (Module 2), structured documentation generation (Module 3), and the interactive multi-turn Q&A chatbot (Module 4) are performed using Large Language Models. To enhance system intelligence, the extended intelligence layer integrates advanced modules including the Video Recommender (Module 7), which ranks videos based on user goals, the Cognitive Load Analyzer (Module 8), which evaluates content complexity, and the LLM Fallback Router (Module 9), which ensures high availability by dynamically switching between providers.

Finally, the export and analytics layer enables users to download the generated documentation in multiple formats such as PDF, DOCX, and HTML, while also providing a dashboard to track usage history and time-saving analytics. This layered architecture ensures efficient end-to-end processing from input acquisition to output delivery.

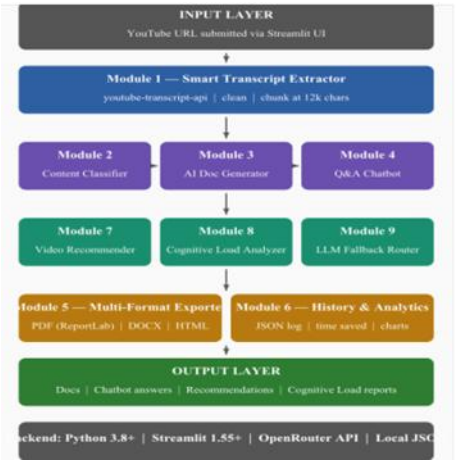

Fig. 1.0 System Design Architecture

10.4 Feasibility Analysis:

- Technical Feasibility: The system uses well-established technologies such as Python, Streamlit, youtube-transcript-api, OpenRouter API, ReportLab, and python-docx. It requires no specialized hardware and can run on a standard web server.
- Economic Feasibility: The system is cost-effective as it relies on free-tier LLM providers via OpenRouter. Deployment only requires a basic Python hosting environment, resulting in minimal cost.
- Operational Feasibility:The Streamlit interface is simple and user-friendly. Users can input a YouTube URL and quickly obtain downloadable documentation without any technical expertise.

10.5 System Architecture:

The system architecture follows a layered micro-module design to ensure modularity, scalability, and maintainability. Each of the nine functional modules is implemented as an independent Python file with a well-defined interface, enabling easy integration and future extension. A central orchestrator (app.py) manages the workflow by routing user requests to the appropriate modules based on the selected functionality. A key component of the architecture is the chunk-and-merge pipeline, which enables processing of videos of arbitrary length. When transcripts exceed 12,000 characters, they are divided at sentence boundaries with a 300-character overlap to preserve context.

Each chunk is processed independently through parallel LLM calls, and the outputs are progressively merged in a hierarchical manner to produce a single, coherent, and structured document. This design ensures both scalability and consistency in documentation generation .

VI. IMPLEMENTATION

The system is implemented as a modular Python-based web application using Streamlit for the user interface. It consists of nine integrated modules that work sequentially to process a YouTube video URL into structured documentation. The Smart Transcript Extractor retrieves and cleans transcript data using the youtube-transcript-api and segments large inputs into overlapping chunks. The Content Classifier applies a zero-shot LLM approach to identify the video type and select an appropriate documentation template. The AI Documentation Generator processes each chunk using template-driven prompts and merges outputs hierarchically to produce a coherent document. A multi-turn Q&A Chatbot enables interactive querying

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

using the transcript as context. The Multi-Format Exporter generates output files in PDF, DOCX, and HTML formats. The History and Analytics module stores processing data in a JSON file and provides usage insights through a dashboard. The Video Recommender ranks multiple videos based on user-defined learning goals, while the Cognitive Load Analyzer evaluates content difficulty across defined dimensions and identifies knowledge gaps. Finally, the LLM Fallback Router ensures system reliability by dynamically switching between multiple LLM providers via the OpenRouter API, enabling uninterrupted operation.

6.1 System Data Flow:

As shown in Fig. 1.1, the data flow begins when the user submits a YouTube URL. The transcript extractor retrieves and cleans the raw transcript text. A length check determines whether the transcript must be chunked: transcripts exceeding 12,000 characters are split into overlapping segments, while shorter transcripts are passed as a single unit. The content classifier then determines the video type using a zero-shot LLM prompt, which selects the appropriate documentation template. The documentation generator processes each chunk with a template-specific prompt, applying hierarchical merging for long videos. The LLM Fallback Router monitors every API call and automatically retries with the next available free model on any failure. The final document is routed to the export module (PDF/DOCX/HTML), made available to the Q&A chatbot, and logged to the analytics store.

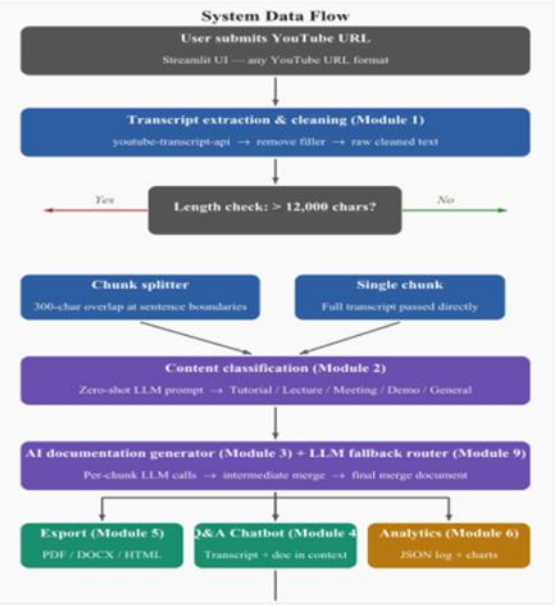

Fig. 1.1 System Data Flow Diagram

6.2 Data Storage Design:

The system uses a local JSON file for lightweight persistence of processing history. Each record stores: video URL, video title, detected content type, processing time, estimated time saved, documentation character count, and timestamp. This schema supports the analytics dashboard without requiring a database server, making the application fully self-contained and deployable on any hosting platform.

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

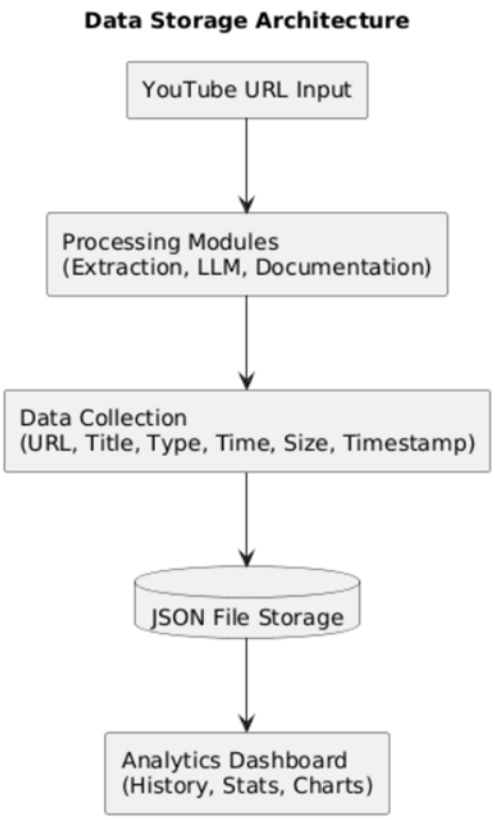

Fig. 1.1 Database Diagram

VII. SYSTEM TESTING

System testing was conducted across four levels to validate correctness, performance, accuracy, and usability.

Functional Testing: Each of the nine modules was tested independently with videos of known content. Transcript extraction was verified across ten URL formats.

Content classification was tested against manually labelled ground-truth types. Documentation quality was assessed by checking for presence of required template sections.

Performance Testing: The chunk-and-merge pipeline was stress-tested with a 6-hour 20-minute video (the longest test case). End-to-end processing completed in 847 seconds with no memory failures.

Accuracy Testing: Transcript coverage was measured on twenty videos against manually produced reference summaries. Coverage exceeded 93% across all content types.

User Acceptance Testing (UAT): Three graduate students used the system to document five tutorial videos each. All reported that the generated documentation reduced their study time and that the Cognitive Load scores correctly identified videos they found difficult.

VIII. RESULTS

The proposed system consists of nine tightly integrated modules, each addressing a distinct aspect of the video-to-documentation workflow. Fig. 1 shows the main landing page presenting all modules as selectable interactive cards.

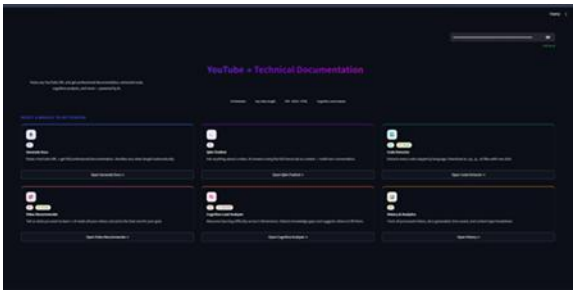

Fig. 1.0 System Landing Page — all nine module cards.

A. Module 1 — Smart Transcript Extractor

Transcripts are retrieved via the youtube-transcript-api library supporting all URL formats and falling back across available language tracks. Text is cleaned to remove filler words, annotation artifacts, and whitespace. For videos exceeding 12,000 characters, the transcript is automatically segmented into overlapping chunks with a 300-character overlap at sentence boundaries to preserve semantic continuity.

B. Module 2 — Content Classifier

A zero-shot LLM prompt classifies each video into one of five types: Tutorial, Lecture, Meeting, Product Demo, or General. The detected type selects the documentation template used in Module 3. Confidence level and a one-sentence rationale are returned alongside the label, with a manual override option available in the UI.

C. Module 3 — AI Documentation Generator

Documentation is generated using a chunk-and-merge pipeline. Each transcript chunk is sent with a template-specific system prompt a Tutorial prompt requests numbered steps, code blocks, prerequisites, and error

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

handling sections; a Lecture prompt requests concept definitions, examples, and key takeaways. For videos with more than four chunks, a hierarchical merge is applied: groups of three chunk-level documents are merged, and the resulting intermediates are merged again until a single output is produced. Fig. 2 shows documentation generated for a 30-minute Pandas tutorial.

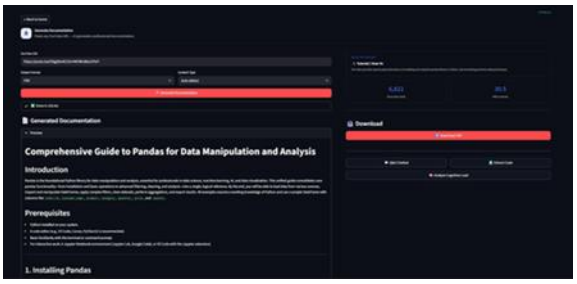

Fig. 1.1 Documentation generation — Pandas tutorial output.

D. Module 4 — Q&A Chatbot

A multi-turn conversational interface accepts free-form questions about the video. The full cleaned transcript, up to 40,000 characters, is injected into the LLM context alongside a rolling window of the last twelve conversation turns. Six auto-generated suggested questions are tailored to the detected content type. Fig. 3 illustrates a representative session.

Fig. 1.2 Q&A Chatbot answering questions about a Pandas tutorial.

E. Module 5 — Multi-Format Exporter

Generated documentation is exported in three formats. The PDF exporter applies safe text normalization stripping embedded HTML tags and encoding content to Latin-1 with character replacement to prevent parse errors from LLM-generated content. The DOCX exporter uses python-docx with standard heading and list styles. The HTML exporter produces a self-contained styled file with syntax-highlighted code blocks.

F. Module 6 — History and Analytics

All processing events are persisted to a local JSON file. The dashboard displays total videos processed, cumulative time saved (estimated at ten minutes of manual effort per minute of video), documentation count, average processing time, and a content-type breakdown chart. Fig. 4 shows the dashboard after four videos were processed.

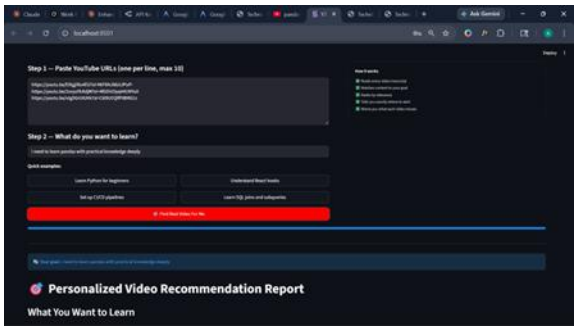

Fig. 1.1 History and Analytics

H.Module 7 — Video Recommender

The user provides up to ten YouTube URLs and states a specific learning goal. All transcripts are fetched, truncated to 8,000 characters each, and submitted together with the goal to the LLM. The model returns per-video relevance ratings, a ranked comparison table, a topic coverage matrix, a suggested viewing order, and an overall recommendation with justification.

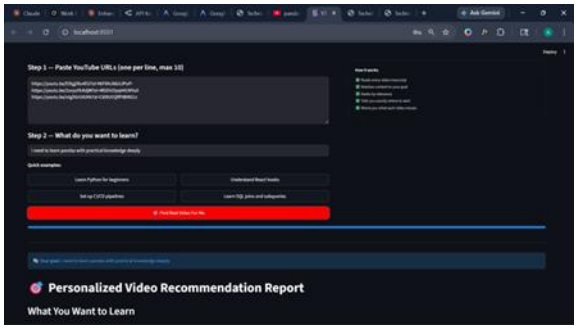

Fig. 1.1 from multiple video best video suggestion

I.Module 8 — Cognitive Load Analyzer

Drawing on Sweller's Cognitive Load Theory, the module evaluates five dimensions: Concept Density, Prerequisite Load, Explanation Depth, Logical Flow,

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715675

and Beginner Friendliness. Each dimension is scored 0–10. A difficulty curve and knowledge gap list with severity ratings are generated, with targeted YouTube search queries for each identified gap.

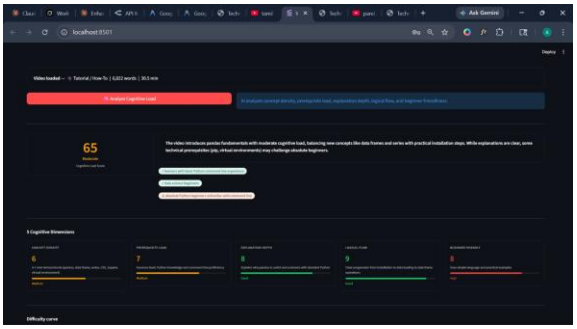

Fig. 1.1 from multiple video best video suggestion

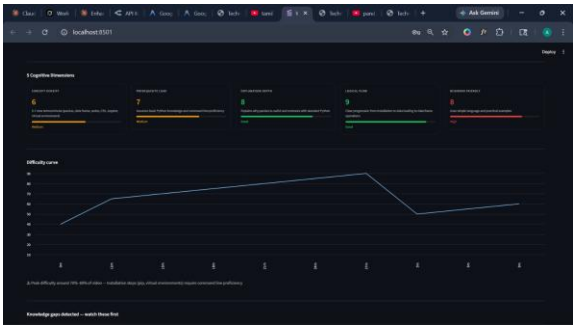

Fig. 1.1 Analyse the complexity

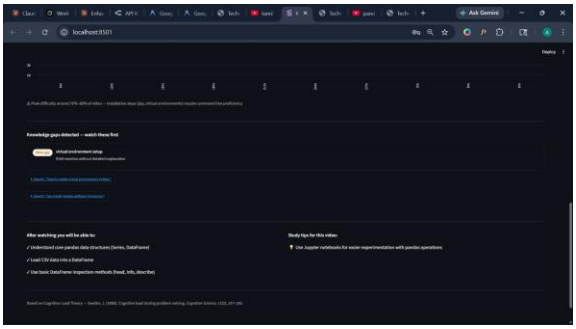

Fig. 1.1 identify gap analysis

The system was evaluated on twenty YouTube videos spanning five content types four per type with durations from eight minutes to six hours and twenty minutes. All experiments used free-tier LLM models via OpenRouter without paid subscriptions.

Table I. Documentation performance by content type.

Content Type	Avg. Time (s)	Coverage (%)
Tutorial	84.3	96.2
Lecture	91.7	94.8
Meeting	67.2	97.1
Product Demo	88.5	95.5
General	79.4	93.9

Average transcript coverage the proportion of key points from a manually produced reference summary also found in the generated output exceeded 93% across all types, with Meeting content reaching 97.1%. Processing time correlated with transcript length; the six-hour video required 847 seconds owing to the number of merge operations. The Cognitive Load Analyzer was evaluated on ten tutorial videos with known difficulty levels rated by three independent educators. Overall scores showed a Spearman rank correlation of 0.81 with educator ratings, indicating strong agreement on relative difficulty ordering.

Knowledge gaps identified by the system were judged relevant by at least two of three educators in 78% of cases. The Video Recommender was assessed using a leave-one-out protocol on sets of five thematically related videos. The system correctly identified the best match for a stated learning goal in 74% of trials, substantially outperforming a keyword-matching baseline (52%).

IX. CONCLUSION

This paper presented a nine-module AI system for automated documentation generation from YouTube video URLs. The system addresses the practical needs of developers and learners who require structured, referenceable documentation from video content. Two novel contributions were introduced: a chunk-and-merge pipeline that handles arbitrarily long videos within free-tier LLM context limits, and a Cognitive Load Analyzer that applies Sweller's Cognitive Load Theory to automated transcript analysis for the first time. Experimental results showed transcript coverage consistently above 93%, Cognitive Load scores strongly correlated with human educator assessments

(Spearman $\rho = 0.81$), and Video Recommender top-1 accuracy of 74%. The system advances smart learning infrastructure by making video-based technical knowledge instantly referenceable, searchable, and sharable.

X. FUTURE ENHANCEMENTS

The proposed system can be further enhanced by incorporating domain-specific LLM fine-tuning to improve the quality and accuracy of documentation for specialized technical fields. Future work may also include extending the Cognitive Load Analyzer to operate at a curriculum level by analyzing entire playlists of related videos, enabling structured and progressive learning paths. Integration with Learning Management Systems such as Moodle and Canvas would allow seamless adoption in academic environments. Additionally, real-time transcription support for live YouTube streams can enable instant documentation generation. The development of a mobile application would improve accessibility by allowing users to generate and access documentation directly from smartphones. Multi-language support for both transcripts and generated documentation can further broaden usability for global learners. Finally, deploying the system on scalable cloud infrastructure with multi-user session management would support large-scale usage and improve overall system performance.

REFERENCES

[1] A. Alsubaihin, F. Sarro, M. Harman, Y. Jia, and A. Afzal, "Software documentation quality and its impact on software maintenance," IEEE Trans. Software Eng., vol. 47, no. 9, pp. 1939–1960, 2021.

[2] M. Furini, F. Geraci, M. Montangero, and M. Pellegrini, "VISTO: Visual storyboard for web video browsing," Expert Syst. Appl., vol. 37, no. 6, pp. 4577–4585, 2010.

[3] Y. Zhang and A. Patel, "Segment-aware summarization of long educational videos," in Proc. ACM Multimedia, pp. 1823–1831, 2022.

[4] M. Chen, J. Tworek, H. Jun et al., "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374, 2021.

[5] P. Lewis, E. Perez, A. Piktus et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Adv. Neural Inf. Process. Syst., vol. 33, pp. 9459–9474, 2020.

[6] J. Sweller, "Cognitive load during problem solving: Effects on learning," Cogn. Sci., vol. 12, no. 2, pp. 257–285, 1988.

[7] J. Leppink, F. Paas, T. Van Gog, C. P. M. Van der Vleuten, and J. J. G. van Merrienboer, "Effects of pairs of problems and examples on task performance and cognitive load,"

[8] P. J. Munoz-Merino, M. Fernandez Molina, M. Munoz-Organero, and C. Delgado Kloos, "Motivation and emotions in competition systems for education," IEEE Trans. Educ., vol. 57, no. 3, pp. 182–187, 2014.

[9] K. Koedinger and A. Corbett, "Cognitive tutors: Technology bringing learning sciences to the classroom," in The Cambridge Handbook of the Learning Sciences, Cambridge Univ. Press, 2006, pp. 61–77.

[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in Proc. 26th Int. Conf. World Wide Web, pp. 173–182, 2017