

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

CCDS24-1273

CCDS24-1273: Video-Enhanced Learning Companion & Co-Pilot –
Video to Text Approach (2nd Gen)

Nicole Kaira Almonte Imatong

Supervisor: Ong Chin Ann

COLLEGE OF COMPUTING AND DATA SCIENCE

Submitted in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Engineering (Computer Engineering/Science)
Nanyang Technological University

2025

ABSTRACT

This project presents CCDS24-1273: Video-Enhanced Learning Companion & Co-Pilot (2nd Gen), an AI-driven lecture chatbot that transforms video learning into an interactive and context-aware experience. Built on Azure's AI ecosystem, it integrates Azure Video Indexer and Azure OpenAI within a Retrieval-Augmented Generation (RAG) framework. Two key modules were developed: a Temporal Identifier for timestamp-aware retrieval and a Document-Scope Identifier for multi-lecture reasoning. Evaluated using the RAGAS framework on test questions, the enhanced system achieved higher precision, faithfulness, and relevancy compared to the baseline. These results highlight the potential of RAG-based learning companions to improve engagement, contextual understanding, and scalability in modern education.

ACKNOWLEDGEMENTS

I would like to extend my sincere appreciation to Mr. Ong Chin Ann, my supervisor, for his invaluable guidance, continuous encouragement, and constructive feedback throughout this project. His expertise in artificial intelligence has been instrumental in shaping the technical direction and academic rigor of this work. His mentorship and support have been pivotal to the successful completion of this Final Year Project.

I would also like to express my gratitude to the College of Computing and Data Science (CCDS) at Nanyang Technological University for providing the necessary facilities and resources that enabled this research. Special thanks are extended to the students who participated in the user testing sessions, whose insightful feedback contributed significantly to enhancing the system's design, performance, and user experience.

TABLE OF CONTENTS

Chapter 1: Introduction	10
1.1 Background.....	10
1.2 Objective.....	11
1.3 Scope.....	12
Chapter 2: Literature Review.....	13
2.1 Role of AI chatbot in modern education.....	13
2.2 Azure Video Indexer for Video to Text Transformation.....	13
2.3 Temporal Understanding and Timestamp Aware Retrieval	14
2.4 Multiple Docs Resource and Scalable Retrieval Systems	15
2.5 Query Reformulation and Expansion For Retrieval Effectiveness.....	17
2.6 RAGAS for Evaluation.....	17
2.7 RAG and Contextual Retrieval.....	18
2.8 Summary.....	19
Chapter 3: Methodology.....	21
3.1 Overview.....	21
3.2 Baseline RAG System	21
3.2.1 The Knowledge Base Component.....	21
3.2.2 The Retriever Component	22
3.2.3 The Integration and Generator Component.....	22
3.2.4 Limitation of the Baseline RAG System.....	22
3.3 Temporal Identifier Module.....	23
3.3.1 Temporal Identifier Control Flow	23
3.3.2 Metadata Addition on the Segments.....	24
3.3.3 Temporal LLM Prompt.....	24
3.4 Document-Scope Identifier Module	25
3.4.1 Single Document and Multiple Document Identifier Control Flow	25
3.4.2 LLM Prompt Design for Document-Scope Identification	25
3.4.3 Single Document Query Processing.....	26
3.4.4 Multi Document Query Processing	26
3.4.5 Concurrent Sub-Query Execution	26
3.5 Evaluation.....	26
3.5.1 Overview of Evaluation Framework	26
3.5.2 Temporal Test Case Preparation	27
3.5.3 Single Document Test Case Preparation.....	28
3.5.4 Multi Document Test Case Preparation	30

3.5.5	Evaluation Procedure.....	31
Chapter 4:	Implementation.....	32
4.1	Overview.....	32
4.2	System Architecture.....	32
4.3	Core System Components.....	33
4.3.1	Component Services.....	34
4.4	Video Upload and Processing Flow.....	38
4.4.1	Sequence Diagram.....	38
4.5	Chat Processing and AI Intelligence.....	40
4.5.1	Sequence Diagram.....	40
4.5.2	LLM Prompts.....	42
4.6	User Interface.....	44
4.6.1	Video Chatbot Page.....	44
4.6.2	Video Landing Page.....	45
4.6.3	Course Management Page.....	46
4.6.4	Video Upload Page.....	47
Chapter 5:	Results and analysis.....	48
5.1	RAGAS Temporal Questions Results.....	48
5.1.1	Temporal Qualitative Results.....	48
5.1.2	Temporal Qualitative Analysis.....	49
5.2	RAGAS Single Docs and Multi Docs Question Results.....	50
5.2.1	Single Docs Quantitative Analysis.....	50
5.2.2	Multi Docs Quantitative Analysis.....	52
5.2.3	Single Docs and Multi Docs Quantitative Analysis.....	53
5.3	User Evaluation.....	55
5.3.1	Temporal Queries Satisfaction.....	56
5.3.2	Single Docs Queries Satisfaction.....	57
5.3.3	Multi Docs Queries Satisfaction.....	58
5.3.4	Overall Satisfaction.....	59
5.3.5	Future Enhancement.....	60
Chapter 6:	Limitation and Future Development.....	61
6.1	Integration of Tutorial and Assignment Contexts.....	61
6.2	Lecture Slide and PDF Upload Functionality.....	61
Chapter 7:	Conclusion.....	62
References	63
Appendix	65
Appendix A	65

Appendix B	67
Appendix C	70
Appendix D	74
Appendix E	78
Appendix F	83
Appendix G	92
Appendix H	100
Appendix I	119
Appendix J	138

LIST OF FIGURES

Figure 1: Overview of Azure Video Indexer Architecture for Audio and Visual Analysis. Extracted from [4].	14
Figure 2: Overview of the QD-RAG pipeline, where complex questions are decomposed into sub-queries, each retrieved separately before answer synthesis. Extracted from [8]	16
Figure 3: Overview of RAG paradigms, techniques for better RAG. Extracted from [15]	19
Figure 4: Video-to-Response Workflow of the Baseline RAG System	21
Figure 5: Baseline RAG + Temporal Identifier Module System	23
Figure 6: Baseline RAG + Temporal Query Module + Document-Scope Identifier Module	25
Figure 7: RAGAS Metrics definition. Extracted from [13].	27
Figure 8: System Architecture of Video-Enhanced Learning Companion	32
Figure 9: Backend Directory Structure	33
Figure 10: Video upload and Processing Sequence Diagram	38
Figure 11: Chat Processing and Retrieval Flow with Doc Scope Identifier and Temporal Routing	40
Figure 12: Temporal and Document Scope Prompt Template	42
Figure 13: Prompt Template for Response Generation Using Video Context	43
Figure 14: Video Chatbot Page	44
Figure 15: Video Chatbot Landing Page	45
Figure 16: Video Chatbot Course Management Page	46
Figure 17: Video Chatbot Upload Page	47
Figure 18: Temporal Quantitative RAGAS Results Bar Chart	48

Figure 19: Responses generated by the Baseline RAG and the Temporal Identifier RAG for a timestamp-based query.....	49
Figure 20: Single Docs Quantitative RAGAS Results Bar Chart.....	50
Figure 21: Multi Docs Quantitative RAGAS Results Bar Chart	52
Figure 22: Single Document Routing and Query Variants Generation	53
Figure 23: Comparison of Baseline RAG and Temporal Identifier RAG Outputs.....	54
Figure 24: Multi Document Routing and Query Variants Generation.....	54
Figure 25: Comparison of Baseline RAG and Temporal Identifier RAG for Multi-Document Query	55
Figure 26: Temporal Queries Satisfaction Survey Question 1.....	56
Figure 27: Temporal Queries Satisfaction Survey Question 2.....	56
Figure 28: Single Docs Queries Satisfaction Survey Question 1	57
Figure 29: Single Docs Queries Satisfaction Survey Question 2.....	57
Figure 30: Multi Docs Queries Satisfaction Survey Question 1	58
Figure 31: Multi Docs Queries Satisfaction Survey Question 2.....	58
Figure 32: Overall Satisfaction Survey Question 1.....	59
Figure 33: Overall Satisfaction Survey Question 2.....	59
Figure 34 Future Enhancement Survey Question	60

LIST OF TABLES

Table 1: Temporal Reasoning Question Dataset.....	27
Table 2: Single Docs Question Dataset.....	29
Table 3: Multi Docs Question Dataset.....	30
Table 4: File layered architecture.....	34
Table 5: Internal services and their functions	34
Table 6: Baseline RAG vs Temporal RAG time taken.....	49
Table 7: Baseline RAG vs Docs Scope RAG time taken (single docs).....	51
Table 8: Baseline RAG vs Docs Scope RAG time taken (multi docs)	53

1.1 Background

The introduction of **generative AI models** such as ChatGPT has fundamentally reshaped the artificial intelligence landscape. Their adaptability and conversational intelligence make them ideal candidates for deployment in education. When thoughtfully integrated, these models can serve as powerful tools to enhance student learning through personalized tutoring, homework support, conceptual explanations, test preparation, team collaboration, and even mental health support [1]. In line with this vision, the LLM (Large Language Model) based Learning Companion and Co-Pilot project was conceived to bridge the gap between static lecture materials and interactive learning. With the rise of video-based content as a preferred mode of instruction, students increasingly face difficulties locating specific moments within long lecture recordings. Unlike text-based materials, videos lack inherent searchability and dynamic Q&A capabilities.

To address this challenge, the **first-generation project (CCDS24-0653)** developed a system that enabled users to interact with lecture video content through a chatbot powered by Retrieval-Augmented Generation (RAG). This system was designed to extract contextual information from lecture videos using Azure AI Video Indexer and serve relevant answers through a Large Language Model hosted on Azure OpenAI. The project integrated baseline RAG, a hybrid retrieval model combined with an LLM-based document transformation process, which demonstrated promising results across multiple evaluation metrics [2].

However, limitations were also observed. User testing and Retrieval-Augmented Generation Assessment (RAGAS) evaluations highlighted gaps in the chatbot's ability to manage nuanced temporal comprehension. In addition, the current pipeline is only able to retrieve information from a single lecture video. Therefore, further testing is essential to evaluate the pipeline on an expanded knowledge base. These limitations opened new directions for refinement in both the pre-processing stage and retrieval strategy of the pipeline.

1.2 Objective

The primary objectives of this project are framed around four key aims that target current limitations of the video-to-text chatbot pipeline. These enhancements are organized under two main processes within the system:

Video Indexing Flow Enhancements

Aim 1: Enhance efficiency of uploading long videos to Azure Video Indexer.

- Analyze the limitations of Azure Video Indexer when processing long duration videos through its API.

Aim 2: Improve context granularity and retrieval accuracy within the knowledge base.

- Explore additional metadata fields from Azure Video Indexer insights such as timestamps, key topics, and scene changes.
- Integrate these fields into existing chunks to enhance contextual richness and retrieval performance.

Query Processing Flow Enhancements

Aim 1: Temporal Understanding by strengthening the system's ability to answer time-sensitive questions.

- Design evaluation test cases for different temporal question types.
- Benchmark temporal reasoning performance before and after enhancements.
- Identify weaknesses in chunking and retrieval strategies.
- Implement timestamp-aware chunking and advanced retrieval strategies.

Aim 2: Retrieval Expansion by enabling flexible retrieval across single or multiple resources efficiently.

- Redesign the retrieval pipeline for scalable and efficient retrieval across multiple lecture resources.
- Design evaluation test cases for single- and multi-resource queries.
- Improve query formulation (e.g., rewriting, expansion, reformulation)

- Enhance retrieval methods to ensure balanced and contextually relevant chunks.

1.3 Scope

This project builds upon the previously completed baseline system. The scope of this project is to enhance the existing pipeline through improvements in video indexing and query processing. The earlier phase of the project established the foundational pipeline for a video to text chatbot system using Azure services. [2] It implemented the video indexing flow to process lecture videos through transcription, Optical Character Recognition (OCR), and frame analysis, and constructed a knowledge base in Azure Cosmos DB with embedded and indexed chunks for retrieval. A user interface was developed to allow students to query lecture content, and the system was evaluated using the RAGAS framework alongside user testing to measure response quality and temporal reasoning performance. [2]

Building on this baseline, the scope of the current phase extends to enhancements in both the video indexing flow and the query processing flow. On the indexing side, the project seeks to improve the handling of long lecture videos through the API. The evaluation will be conducted using the SC1007 lecture series by Dr. Loke, where each video lasts approximately 65 to 85 minutes. On the query side, the project focuses on strengthening temporal understanding through timestamp-aware chunking and retrieval strategies, as well as expanding retrieval capabilities to support both single resource and multi-resource queries. Query reformulation and representation techniques will also be explored to improve retrieval effectiveness.

2.1 Role of AI chatbot in modern education

Introducing potent technology such as AI chatbots in the education system can help many students to grasp content more quickly. In this research paper, it was able to state the benefits of AI chatbots for both students and educators. It was discussed that chatbots assist students by providing homework and study support, personalised learning and skill development. [3] As for the educators, the AI-powered chatbots save them time and enhance teaching strategies. [3]

AI-powered chatbots provide real-time feedback, suggest tailored learning materials, and adapt question difficulty based on student performance. Additionally, as chatbots offer an accessible, interactive environment, and nonjudgmental setting for asking inquiries, they lessen students' reluctance to ask for assistance. Thus, AI-powered chatbots enable students to take charge of their education, increasing flexibility and student-centeredness.

2.2 Azure Video Indexer for Video to Text Transformation

Azure AI Video Indexer (AVI) provides an automated pipeline that applies Azure AI Speech and Vision to extract transcripts using automatic speech recognition (ASR), optical character recognition (OCR), and metadata from videos. As shown in Figure 1, it combines ASR to generate transcripts of spoken audio and OCR to detect any text shown on slides or visuals [4]. AVI runs dozens of AI models on each video for speech, on-screen text, faces, scene changes, to produce a structured index of the content.

with its timestamp, the system maintains narrative continuity and improves retrieval of time-sensitive information. Specialized lecture video systems now utilize vector-based search with timestamp indexing, allowing users to directly retrieve the precise moment in a video where their query is discussed [6]. Moreover, storing chunk-level time markers as metadata allows the chatbot to not only fetch relevant content but also point users to the specific timeframe in the video [7].

Overall, incorporating temporal understanding from detecting temporal intent in queries to chunking transcripts with timestamp alignment, is shown to enhance the accuracy and relevance of answers for questions involving event timing or chronology.

2.4 *Multiple Docs Resource and Scalable Retrieval Systems*

Lecture chatbots often need to draw information from multiple documents or videos to answer comprehensive queries. This makes multi-document retrieval and fusion a crucial component of scalable RAG systems. Research on multi-resource RAG has explored splitting complex questions and retrieving them from different sources in parallel. For instance, one method (QD-RAG) decomposes a complex query into simpler sub-questions, retrieves evidence for each sub-query separately, then aggregates the results for the final answer [8]. Such query decomposition ensures comprehensive retrieval across multiple lectures or documents.

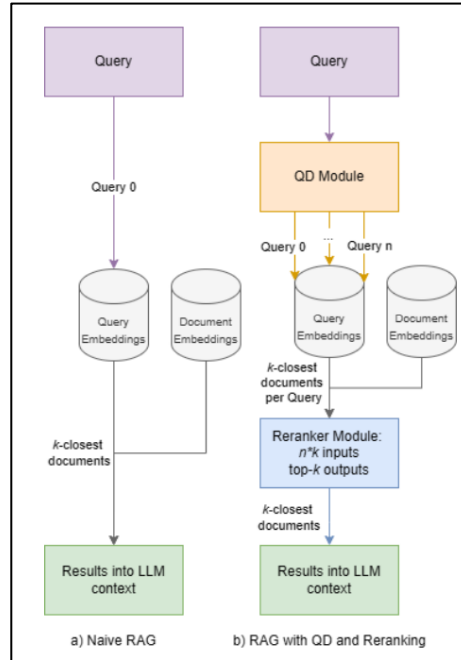


Figure 2: Overview of the QD-RAG pipeline, where complex questions are decomposed into sub-queries, each retrieved separately before answer synthesis.

Extracted from [8]

The subsequent challenge is to combine these separate results into a coherent answer. Hybrid retrieval (dense and sparse) and rank fusion techniques address this by merging evidence from different retrievers. In particular, Reciprocal Rank Fusion (RRF) has proven to be a simple yet powerful approach for combining ranked lists from multiple searches. Documents that consistently appear near the top of any list accumulate higher scores, and the fused list is sorted by these combined scores [9]. Notably, RRF often outperforms more complex fusion methods while requiring minimal tuning [9]. In a hybrid retrieval setting, this means results from a dense vector search and a keyword search can be merged to improve coverage and precision. Studies report that using RRF to blend multiple search strategies yields more accurate and robust results in RAG pipeline [9].

Therefore, for a lecture chatbot dealing with multi-video queries, adopting parallel retrieval with sub-query splitting and using rank fusion (like RRF) to integrate results can enhance scalability and answer completeness.

2.5 *Query Reformulation and Expansion for Retrieval Effectiveness*

Improving the retrieval step of a Retrieval-Augmented Generation (RAG) system often involves reformulating the user’s query to better align with the stored knowledge base. Users’ natural questions may be vague, overly complex, or expressed in ways that differ from the phrasing used in the documents. Query rewriting and expansion techniques aim to bridge this gap. One class of approaches employs large language models (LLMs) to generate alternative or related versions of a query, thereby broadening the search scope. For example, Hypothetical Document Embeddings (HyDE) prompts an LLM to synthesize a hypothetical document that would answer the user’s question and then uses the embedding of that generated document to retrieve semantically similar real documents [10]. This method enriches query context and has been shown to outperform direct query embeddings in zero-shot dense retrieval, approaching the performance of fine-tuned retrievers [10].

Additionally, specialized retrieval frameworks such as Self-Query Retriever leverage LLMs to translate natural-language questions into structured queries, for instance, a semantic vector search combined with metadata filters [11]. This approach allows the retriever to target document attributes such as lecture title, date, or speaker when implied by the query, ensuring that the search surfaces the most contextually relevant segments.

Overall, techniques such as HyDE and LLM-based query decomposition or rewriting have demonstrated improved retrieval recall and contextual relevance for the generation module [10]. Integrating such query expansion and reformulation strategies into the lecture chatbot’s pipeline enables the system to retrieve richer and more precise supporting information, ultimately enhancing the quality of generated answers.

2.6 *RAGAS for Evaluation*

Evaluating a RAG system is inherently complex because it requires assessing both the quality of the retrieved context and the quality of the generated response. RAGAS (Retrieval-Augmented Generation Assessment) is a recently proposed framework that provides a comprehensive and reference-free methodology for evaluating such

systems [12]. Instead of depending solely on exact-match comparisons with a ground-truth answer, RAGAS employs a set of metrics powered by large language model (LLM) judgments to evaluate multiple performance dimensions.

At the retrieval level, RAGAS measures context relevancy, which reflects the precision of retrieved passages, and context recall, which indicates completeness of the retrieved information required to answer the query [13]. High context relevancy means that most retrieved chunks are topically relevant with minimal noise, whereas context recall determines whether critical facts were missed. At the generation level, RAGAS evaluates faithfulness and answer relevancy. Faithfulness assesses the factual accuracy of the generated answer with respect to the retrieved context, specifically the proportion of statements in the response that are supported by the evidence. Answer relevancy measures how completely and directly the response addresses the user’s query [13].

All metrics range from zero to one, with higher values reflecting better performance. Together, these metrics provide a holistic view of the system’s effectiveness. A key advantage of RAGAS is its ability to operate with minimal human annotation because it leverages LLMs to automatically compare the generated answer and retrieved context for factuality and relevance [13].

This automation enables faster and more scalable evaluation that aligns with the rapid development cycle of modern LLM-based applications. By adopting RAGAS for the lecture chatbot, improvements in retrieval performance, measured through context precision and recall, and in answer quality, measured through faithfulness and relevancy, can be quantitatively tracked as new modules are introduced.

2.7 *RAG and Contextual Retrieval*

Retrieval-Augmented Generation (RAG) combines large language models with external knowledge retrieval to improve factual accuracy [14]. Recent research has improved the retrieval process through multi-document retrieval, query rewriting, re-ranking, and ensemble fusion techniques. These strategies provide the generator with richer and more relevant information, effectively reducing hallucinations and

improving complete answers. As shown in Figure 3, the RAG ecosystem has evolved from naive to modular architecture, integrating various optimization techniques to strengthen retrieval performance.

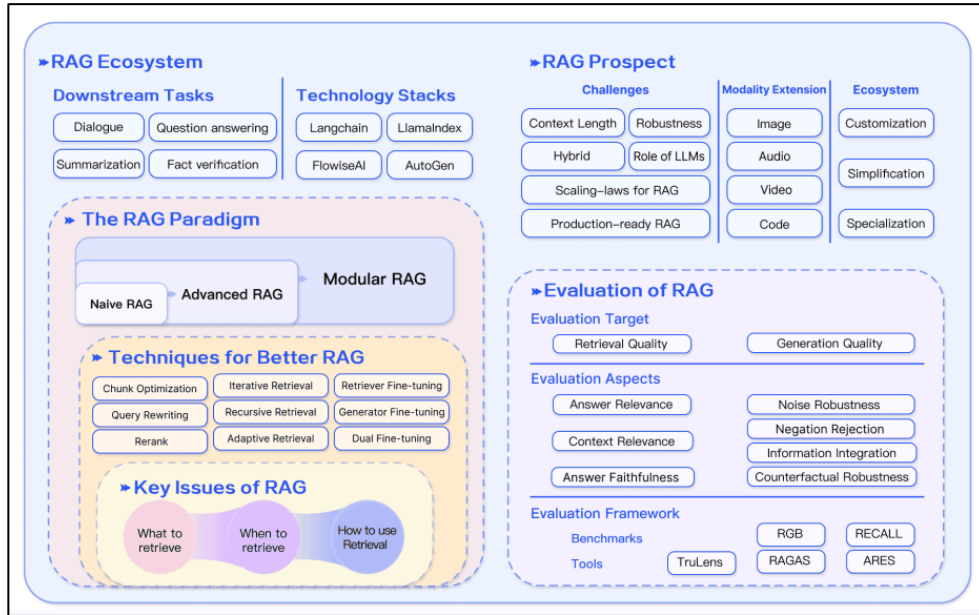


Figure 3: Overview of RAG paradigms, techniques for better RAG. Extracted from [15]

This framework directly aligns with the current project, which progresses from a baseline RAG design to a modular system that integrates a Temporal Identifier and Document Scope Identifier module. Techniques such as hybrid retrieval, query rewriting, and RRF reranking are applied to enhance contextual alignment and multi-video reasoning. By extending RAG to the video modality through Azure Video Indexer, the project leverages temporal and cross-video metadata to enable timestamp-aware retrieval. Using the RAGAS evaluation framework, the system assesses faithfulness, relevance, and contextual accuracy, resulting in a student-facing AI chatbot capable of synthesizing insights across multiple lectures to deliver richer and more efficient learning experiences.

2.8 Summary

From the reviewed studies, this project builds upon the baseline RAG system by continuing the use of Azure AI Services to enhance video understanding and chatbot

responses. Azure OpenAI and Azure Video Indexer will be utilized to support these improvements through transcript analysis, scene segmentation, and intelligent query processing. The project further investigates temporal and multi-document retrieval to strengthen the system's ability to handle complex and time-sensitive queries. The enhanced system will then be evaluated using the RAGAS framework and compared with the baseline implementation to assess overall performance and effectiveness.

3.1 Overview

This section provides a comprehensive overview of the design choices and the rationale behind integrating modern technologies into the baseline RAG system. It highlights the frameworks and strategies adopted in the development of the LLM-based Learning Companion and Co-Pilot generation 2.

3.2 Baseline RAG System

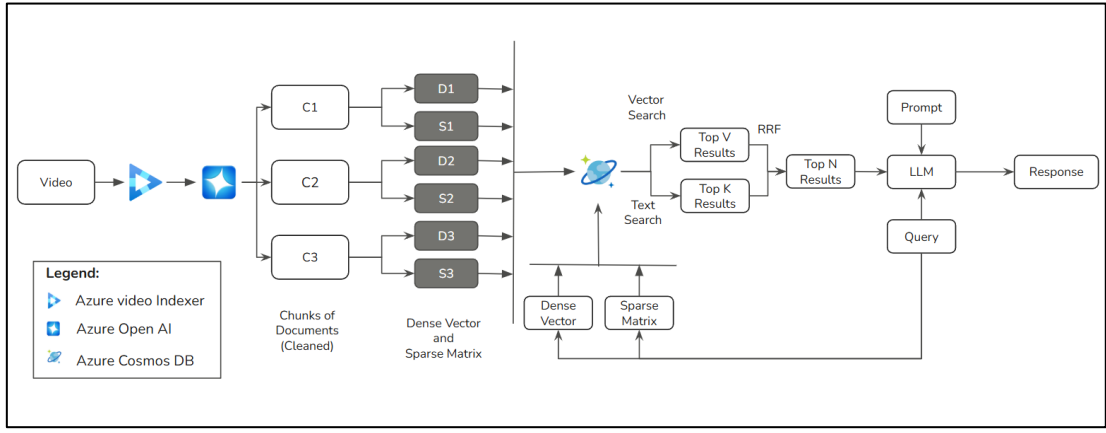


Figure 4: Video-to-Response Workflow of the Baseline RAG System

The baseline RAG system provides a solid foundation for video-to-text question answering. Figure 4 illustrates the pipeline architecture of the baseline RAG system, which consists of four main components: the knowledge base, retriever, integration layer, and generator.

3.2.1 The Knowledge Base Component

Within the knowledge base component, four key processes are carried out. First, the uploaded lecture video is indexed by Azure AI Video Indexer, which segments the video into prompt-ready contents containing transcripts, OCR, and tag metadata. Next, these segments undergo a transcript-cleaning process using an LLM on Azure OpenAI to improve readability and contextual accuracy. The cleaned segments are then transformed into dense and sparse embeddings to support both semantic and keyword-based retrieval. Finally, the processed embeddings and their corresponding metadata

are stored in Azure Cosmos DB for MongoDB vCore, which functions as the vector database for subsequent retrieval tasks.

3.2.2 The Retriever Component

Within the retriever component, two retrieval techniques are employed to identify segments in the knowledge base that are semantically similar to the user's query. These are semantic vector search and keyword-based search. The results from both retrieval methods are ranked and combined using a Weighted Reciprocal Rank Fusion (RRF) algorithm, where the text-based search is assigned a lower weight relative to the semantic search. The top five ranked results are then selected and passed to the Generator component as contextual input for the LLM.

3.2.3 The Integration and Generator Component

Within the integration component, it acts as the orchestrator that coordinates the interaction amongst all the modules and manages data exchange across the pipeline. This component creates a combined prompt consisting of both the user query and the retrieved context before forwarding it to the Generator component in Azure OpenAI. It also applies prompt-engineering techniques to automate the construction of effective prompts, ensuring the LLM receives sufficient context to generate accurate and relevant response.

Within the generator component, the LLM hosted on Azure OpenAI creates the final output based on the augmented prompt provided by the integration component. The generated response is then returned to the user through the application interface.

3.2.4 Limitation of the Baseline RAG System

This baseline design demonstrates reliable response relevance and contextual alignment. However, it has its limitations. It struggles to handle temporal queries that requires timestamp alignment, lacks scalability for multi-document retrieval across several lectures, and exhibits shallow evaluation depth. Building on this foundation, this project extends the baseline system to incorporate query-time temporal understanding and multi-video reasoning, addressing the precise gaps identified in the baseline implementation.

3.3 Temporal Identifier Module

3.3.1 Temporal Identifier Control Flow

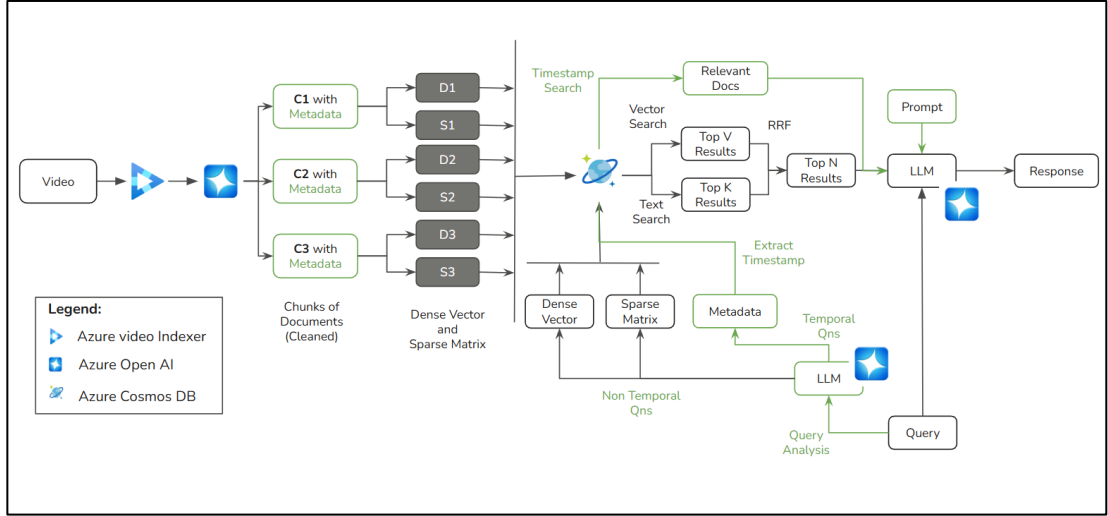


Figure 5: Baseline RAG + Temporal Identifier Module System

The Temporal Identifier module extends the baseline RAG pipeline. Its purpose is to determine whether a user’s query requires time-sensitive retrieval. This module introduces modifications to the knowledge base component mainly the chunk metadata and the retriever component by adding an LLM-based Temporal Identifier prompt hosted on Azure OpenAI of the baseline system.

When a user submits a query, it is first passed through a temporal LLM classifier and extractor. This module returns a structured output containing the time elements. Whether the query is temporal in terms of anchor timestamp or time ranges explicitly referenced.

If the query is temporal, the system pre-filters candidate chunks based on the extracted time window. A tolerance margin or ± 2 -minute tolerance window is applied around anchor points or within specified ranges to capture relevant context. Retrieval then proceeds inside this narrowed window using the hybrid retriever, dense and sparse retrieval is performed. The retrieved results are reranked with weighted RRF. If the query is not temporal, the system simply runs the baseline hybrid retrieval followed by weighted RRF without temporal adjustments. In both cases, the LLM prompt receives the top-k ranked chunks and minimal system instructions to include

timestamps when available. If the query is non-temporal, the system gracefully falls back to the baseline hybrid retrieval.

3.3.2 Metadata Addition on the Segments

To support timestamp-aware retrieval, each chunk generated is enriched with metadata fields rather than only sentence-level timestamps. This process was implemented within the knowledge base component of the pipeline. Referring to Figure 5 above, these changes can be observed where the cleaned segments now include metadata. Besides the textual content derived from Azure AI Video Indexer (transcript, OCR and tags), additional attributes such as start time, end time and the video ID are appended as metadata to every chunk object before storing into the vector database.

The metadata design enables fine-grained alignment between user queries and their corresponding temporal locations in the video. Consequently, during retrieval, the system can directly filter chunks by their time window, ensuring that only segments overlapping the relevant timestamp range are returned.

3.3.3 Temporal LLM Prompt

The temporal LLM prompt is designed to enforce precise behavior from the extractor. The model must detect and normalize all forms of temporal references: explicit timestamps such as “at 19:00” and ranges “between 10 and 15 minutes” of the lecture video. The expected output specifies normalized anchors in “hh:mm:ss” format. If no reliable temporal signal is found, the model will not return any anchors. This ensures that the downstream pipeline either uses temporal filtering or defaults back to baseline retrieval seamlessly.

This design ensures consistent temporal intent detection, standardized timestamp normalization and seamless fallback to baseline retrieval, ultimately improving precision and stability in time-stamp query processing.

3.4 Document-Scope Identifier Module

3.4.1 Single Document and Multiple Document Identifier Control Flow

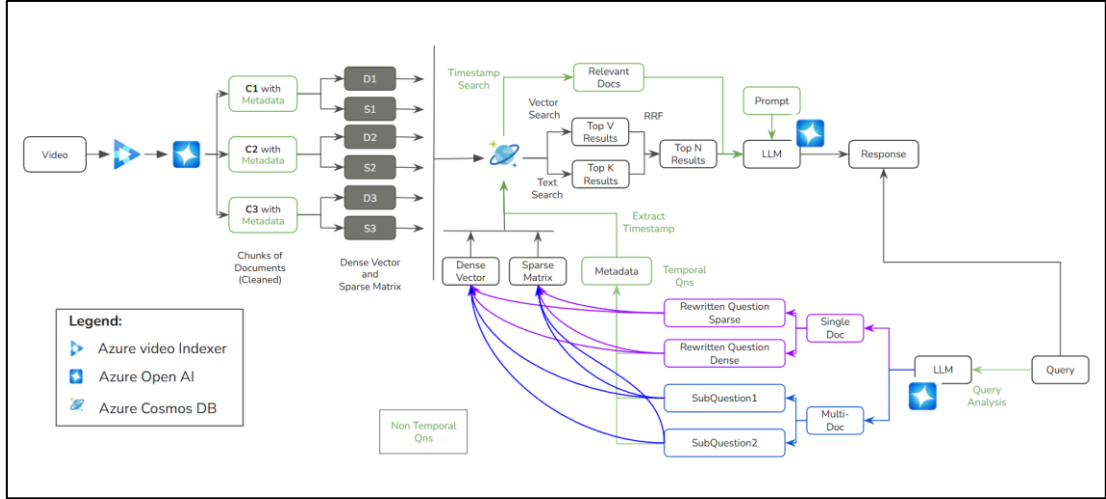


Figure 6: Baseline RAG + Temporal Query Module + Document-Scope Identifier Module

The baseline system was initially limited to retrieving context from one video at a time. Therefore, this project explored extending the system to allow users to query and obtain contextual information from multiple videos. Several modifications were made to the baseline RAG system to ensure both the knowledge base construction and the retrieval process could support multi-video retrieval for each query.

In addition, an LLM hosted on Azure OpenAI was integrated into the baseline system to identify whether a query required a single document or multiple documents for retrieval. A single document query referred to questions that required context from one lecture video, whereas a multi-document query involved information from two or more lecture videos. This design was implemented to reduce noise and improve retrieval precision, ensuring unrelated material from other videos was not retrieved.

3.4.2 LLM Prompt Design for Document-Scope Identification

When the user submits a query, it is processed through a single-multi document classifier module, which returns a structure output containing three fields which are document retrieval type, videos of interest and sub-queries generated. This structured

output allowed the system to adapt its retrieval strategy based on the identified query type.

3.4.3 Single Document Query Processing

For single document queries, the LLM rewrites the query into two optimized forms, one focusing on meaning (semantic retrieval) and another focusing on keywords (sparse retrieval). Both rewritten queries are used to search through the chunks in the database. The results from both searches are merged and ranked to ensure that the most relevant segments are returned to the LLM for the final response generation.

3.4.4 Multi Document Query Processing

For multiple document queries, the LLM decomposes the main question into smaller sub-questions, each representing a distinct lecture or aspect of the query. Each sub-question performs its retrieval independently and the results are combined and re-ranked using the Reciprocal Rank fusion (RRF) method. This ensures a balanced and contextually accurate aggregation of results across multiple videos.

3.4.5 Concurrent Sub-Query Execution

To enhance performance, both sub-query retrievals were executed concurrently, minimizing the overall retrieval time and maintaining consistency in ranking. This parallel execution strategy significantly improved system responsiveness.

3.5 Evaluation

3.5.1 Overview of Evaluation Framework

For testing, virtual lectures delivered by Dr. Loke Yuan were used as sample material. The lecture, part of the SC1007: Data Structures and Algorithms course, is a core module for all CCDS freshmen. It is available in the following YouTube playlist ([Link](#)). A total of five videos were selected from the series, with runtimes ranging between 65 and 85 minutes each.

The evaluation aimed to assess the effectiveness of the proposed modules against the baseline RAG system using the RAGAS framework. It is divided into two distinct experiments, each targeting a core enhancement of the system. The first experiment

focuses on temporal question answering, testing whether the pipeline can detect, filter, and answer time-sensitive queries within a single lecture video. The second experiment evaluates knowledge-based expansion, where multiple videos from the same course are stored in one collection to assess retrieval and answering performance across lectures. Both experiments were assessed using the RAGAS framework, which provides structured metrics for retrieval and answer quality. To ensure comparability across experiments, the following four RAGAS criteria were used as illustrated in Figure 7 below.

Metric	Definition
Context Precision	The fraction of retrieved context chunks that are relevant to the query.
Context Recall	The fraction of all relevant chunks that were successfully retrieved from the knowledge base.
Faithfulness	This checks whether the generated answer is factually consistent with the retrieved context with no hallucinations.
Answer Relevancy	This measures how well the generated answer addresses the user’s query

Figure 7: RAGAS Metrics definition. Extracted from [13].

3.5.2 Temporal Test Case Preparation

This experiment was used to test temporal reasoning performance. Questions in this set explicitly reference timestamps, short ranges, or relative markers. Ground truth was annotated with the correct time ranges and content.

The aim was to validate whether the temporal query module correctly identifies timestamp references, retrieves the correct time-aligned chunks, and produces answers that quote or match the annotated timing. The table below are five of the temporal questions with the ground truth used for this experiment. The complete list of questions and ground truth is provided in Appendix A.

Table 1: Temporal Reasoning Question Dataset

No.	Question	Ground Truth
-----	----------	--------------

1	What is mentioned at 33 minutes of the lecture?	At the 33-minute mark, the lecturer mentioned that the module will mainly cover problem types such as searching, graph problems, and combinatorial problems involving permutations.
2	What was discussed at 27:00 of the lecture?	The lecturer discussed the Fibonacci sequence, using it as an example to illustrate algorithmic thinking and recursive problem-solving.
3	When was the difference between algorithm and program discussed?	The difference between an algorithm and a program was discussed around 21:00
4	What concept is explained around 45:00 into the lecture?	Around the 45-minute mark, the lecturer explains stable sorting algorithms, highlighting that they preserve the relative order of repeated elements during sorting, using student marks across modules as an example.
5	At what point in the lecture does it start discussing How do we solve the different searching, graph and combinatorial problems?	It begins discussing how to solve different searching, graph, and combinatorial problems around the 53-minute mark, introducing data structures and algorithmic strategies like brute force, divide and conquer, greedy, and more.

3.5.3 *Single Document Test Case Preparation*

This experiment tested retrieval within a single video under a course with multiple videos in it. The questions in this set are crafted to ensure the retrieval from only one

video. The Ground truth was annotated with the correct contextual segments corresponding to the specific video selected.

The aim was to validate whether the single document component of the pipeline correctly identifies the video of interest, rewrites the main questions into semantic and keyword focused questions, retrieves the appropriate contextual chunks, and generates answers that accurately reflect or match the ground truth. The table below are five of the single document questions with their corresponding ground truth. The complete list of questions and ground truth entries is provided in Appendix B.

Table 2: Single Docs Question Dataset

No.	Question	Ground Truth
1	According to the lecturer in Sc1007_videolecture, is the lab graded, and how is attendance handled?	The lab itself is not graded, but attendance is still taken. Skipping labs may hurt performance in lab tests.
2	In Sc1007_videolecture, what penalty does the lecturer state for plagiarism in assignments?	Assignments are checked for plagiarism; getting caught results in zero marks for that assignment.
3	List the three essential characteristics of an algorithm mentioned by the lecturer in Sc1007_videolecture.	Correctness, precision (unambiguous steps), and finiteness (must terminate in finite time).
4	When is the combined lab test and quiz scheduled, and what does the quiz cover mentioned in Sc1007_videolecture.	Week 14 (April 20): a combined two-hour lab test and quiz. The MCQ tests concepts from Weeks 1–13, including analysis of algorithms.
5	In the factorial example with a while loop mentioned in Lecture2_Sc1007, how does	It scales linearly with N; time complexity is linear ($O(N)$).

	computational time scale with N?	
--	----------------------------------	--

3.5.4 Multi Document Test Case Preparation

This experiment tested retrieval across multiple videos within the same course. The questions in this set are designed to require information from two or more lecture videos. The Ground truth was annotated with the correct contextual segments drawn from each relevant video, ensuring that cross-lecture relationships and comparison were accurately represented.

The aim was to validate whether the multi-document component of the pipeline correctly identifies all relevant videos, decomposed the main question into sub-queries, retrieved the appropriate chunks from each video, and synthesized them into a coherent and faithful response. The table below presents five of the multi-document questions with their corresponding ground truth. The complete list of questions and ground truth entries is provided in Appendix C.

Table 3: Multi Docs Question Dataset

No.	Question	Ground Truth
1	Across the two lectures Sc1007_Videolecture and Lecture2_videolecture, which topics are planned after the analysis of algorithms and when might hash tables be covered?	Both lectures mention moving on to hash tables and graph problems. Lecture2_Sc1007 notes hash tables will be covered on Monday (or Wednesday) depending on time.
2	"Which lecture between Sc1007_Videolecture and Lecture2_videolecture, previews asymptotic notation and which one defines Big O,	Sc1007_videolecture previews that asymptotic notation (e.g., Big O) will be covered; Lecture2_Sc1007 defines Big O, Big Omega, and Big Theta and explains ignoring constants and focusing on growth order.

	Big Omega, and Big Theta in detail?"	
3	Summarize how recursion impacts time and space across the Sc1007_Videolecture and Lecture2_videolecture lectures using examples given.	Sc1007_videolecture uses Fibonacci to show naive recursion can be exponential in time; Lecture2_Sc1007 shows factorial recursion is linear time but uses extra stack memory compared to iteration.
4	From both lectures Sc1007_Videolecture and Lecture2_videolecture, what is the difference between an algorithm and a program, and how is efficiency evaluated?	Sc1007_videolecture explains an algorithm is a well-defined procedure while a program is its implementation; Lecture2_Sc1007 explains efficiency is evaluated via time/space complexity and asymptotic growth, not raw runtime constants.
5	Combine the examples: which method is most efficient for summing $1..N$, and how is this justified by the complexity principles in both Sc1007_Videolecture and Lecture2_videolecture lectures.	Using the arithmetic series formula $N/2 * (1+N)$ is most efficient (constant-time operations), justified by Lecture2_Sc1007's principle of focusing on lowest order of growth and ignoring constants.

3.5.5 Evaluation Procedure

Each test case was first executed on the baseline RAG system, and the RAGAS results were collected. The same test cases were then re-evaluated using the Baseline + Temporal Query Module and the Baseline + Document-Scope Identifier Module to ensure a fair comparison under identical conditions. The average response time per query was also recorded to assess performance efficiency.

For each configuration, the RAGAS metrics (Context Precision, Context Recall, Faithfulness, and Response Relevancy) were averaged across all use case test questions. These averaged values were used to compare overall performance and analyze the improvements achieved by introducing temporal and document-scope awareness into the retrieval pipeline.

CHAPTER 4: IMPLEMENTATION

4.1 Overview

The following section provides a technical overview of the services and components used in the Video-Enhanced Learning Companion & Co-Pilot (Generation 2). This version builds upon the baseline RAG framework by introducing temporal context awareness, document scope identification and retrieval, and intelligent orchestration across modular AI services. The subsequent subsections detail the system architecture, core system components and two main processing flows, and user interface that collectively power the end-to-end functionality of the Video-Enhanced Learning companion.

4.2 System Architecture

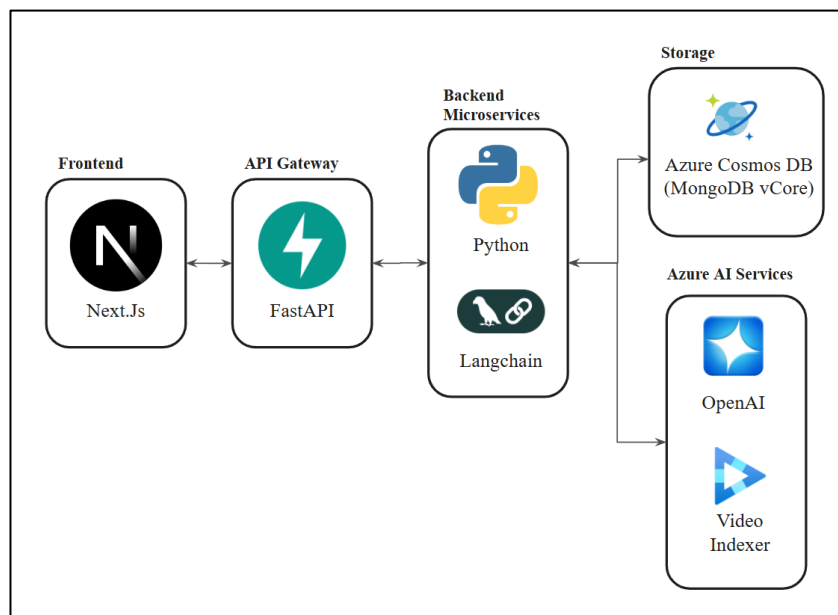


Figure 8: System Architecture of Video-Enhanced Learning Companion

This diagram above presents the overall system architecture integrating both frontend and backend components. The Next.js frontend provides user interaction features such as course management, video uploads, and chat. Requests pass through the FastAPI gateway, which manages routing and API communication with the Python microservices. These microservices perform video analysis, embedding generation, and RAG processing. Data is stored in Azure Cosmos DB (MongoDB vCore), ensuring scalable and flexible document-based storage. The Azure AI Services layer provides the system's intelligence. Azure Video Indexer automatically extracts multimodal insights such as transcripts, speaker segmentation, OCR text, and scene boundaries from uploaded lecture videos, while Azure OpenAI powers large language model processing for transcript cleaning, temporal understanding, document scope identification, question reformulation, and conversational response generation.

4.3 Core System Components

```

backend/
├── main.py                # FastAPI application entry point
├── requirements.txt       # Python dependencies
├── utils.py              # Utility functions and prompt templates
├── EmbeddingService.py   # Text embedding service
├── brokerservice/        # Video processing orchestration
│   ├── brokerService.py  # Main broker service
│   ├── model.py          # Data models
│   ├── repository.py     # Database operations
│   ├── router.py         # API endpoints
│   └── status.py         # Status definitions
├── chatservice/          # Conversational AI service
│   ├── chatService.py    # Core chat logic
│   ├── model.py          # Chat data models
│   ├── repository.py     # Chat database operations
│   ├── router.py         # Chat API endpoints
│   └── utils.py          # Chat utilities
├── databaseservice/      # Database connection management
│   └── databaseService.py # MongoDB singleton service
├── transcriptservice/    # Transcript processing
│   ├── TranscriptService.py # Transcript cleaning & processing
│   └── repository.py      # Transcript database operations
├── videoindexerclient/   # Azure Video Indexer integration
│   ├── Consts.py         # Constants and configuration
│   ├── model.py          # Video data models
│   ├── repository.py     # Video database operations
│   ├── router.py         # Video API endpoints
│   ├── utils.py          # Video utilities
│   ├── VideoIndexerClient.py # Azure Video Indexer client
│   └── VideoService.py   # Video processing service
├── evaluation_results/   # Evaluation data storage
│   └── evaluation_results_*.json # Evaluation result files

```

Figure 9: Backend Directory Structure

As shown in Figure 9 above, each service is structured to have four main files which are the service, repository, model and router. This consistent organization promotes separation of concerns, ensuring that each file serves a distinct purpose while minimizing interdependencies between layers. Such layered architecture enhances maintainability, scalability, and readability of the overall backend system.

Table 4: File layered architecture

File Name	Description
Model	Defines the data models and schemas used throughout the application.
Repository	Manages all database access operations by abstracting interactions with the data layer.
Service	Implements the core business logic of the application, coordinating interactions between layers.
Router	Defines API endpoints and manages HTTP request-response handling through FastAPI's routing system.

4.3.1 Component Services

Table 5 below shows the main purpose, functions and features of each internal service.

Table 5: Internal services and their functions

Services	Descriptions
FastAPI Application	<p>Purpose: Main application entry point</p> <p>Function: Routes requests to appropriate services</p> <p>Feature: CORS support and background tasks</p>
Broker Service	<p>Purpose:</p> <p>Acts as the central orchestration service within the video processing pipeline. It manages communication between multiple backend</p>

	<p>services to ensure smooth video upload, processing, and status tracking.</p> <p>Functions:</p> <ul style="list-style-type: none"> • Coordinates workflows between the Video Indexer, Transcript, and Database services. • Manages video uploads, course validation, and registration. • Tracks processing status and updates video records in the database.
Embedding Service	<p>Purpose:</p> <p>Generates text embeddings and enables semantic retrieval for contextually relevant information within the chatbot system.</p> <p>Functions:</p> <ul style="list-style-type: none"> • Text Embedding Generation: Utilizes Azure OpenAI to convert textual data into high-dimensional vector representations.
Chat Service	<p>Purpose:</p> <p>Provides an AI-powered conversational question-answering system that interprets user intent and generates contextually relevant responses.</p> <p>Functions:</p> <ul style="list-style-type: none"> • Question Processing: Handles user prompts and transforms them into structured queries. • Response Generation: Synthesizes contextually grounded answers using retrieved video and transcript data. • Context Management: Maintains and utilizes chat history to preserve conversational continuity. <p>Advanced Features</p> <ul style="list-style-type: none"> • Temporal Identifier: Supports time-sensitive retrieval to handle queries referencing specific timestamps.

	<ul style="list-style-type: none"> • Document Scope Identifier: Distinguishes between single document and multi document retrieval contexts for efficient query routing. • Multi-Video Support: Enables cross-video search and synthesis for comprehensive answers.
Database Service	<p>Purpose:</p> <p>Manages MongoDB connections across all services to ensure efficient and reliable data access.</p>
Transcript Service	<p>Purpose:</p> <p>Handles the extraction, cleaning, and structuring of video transcripts to prepare them for retrieval and analysis.</p> <p>Functions:</p> <ul style="list-style-type: none"> • Orchestrates transcript cleaning and chunking processes. • Utilizes Azure OpenAI for AI-powered transcript refinement. • Maps timestamps to corresponding transcript segments for precise contextual retrieval. <p>Core Functionalities:</p> <ul style="list-style-type: none"> • Extracts raw transcripts from video insights. • Performs AI-driven cleaning and formatting for readability. • Maps timestamps to transcript text for temporal alignment. • Processes and chunks large transcripts for efficient embedding and retrieval.
Video Indexer Client Service	<p>Purpose:</p> <p>Integrates with Azure Video Indexer to automate video analysis, metadata extraction, and insight generation for the system.</p> <p>Key Components:</p>

	<ul style="list-style-type: none"> • VideoIndexerClient.py – Handles communication with the Azure Video Indexer API. • VideoService.py – Manages video upload, processing, and result handling workflows. <p>Core Functionalities:</p> <ul style="list-style-type: none"> • Uploads and indexes videos through the Azure Video Indexer API. • Extracts thumbnails, transcripts, and video insights automatically. • Generates prompt content for downstream AI services. • Creates embeddable widget URLs for video playback and visualization. • Manages secure authentication and access tokens for Azure services.
--	---

4.4 Video Upload and Processing Flow

4.4.1 Sequence Diagram

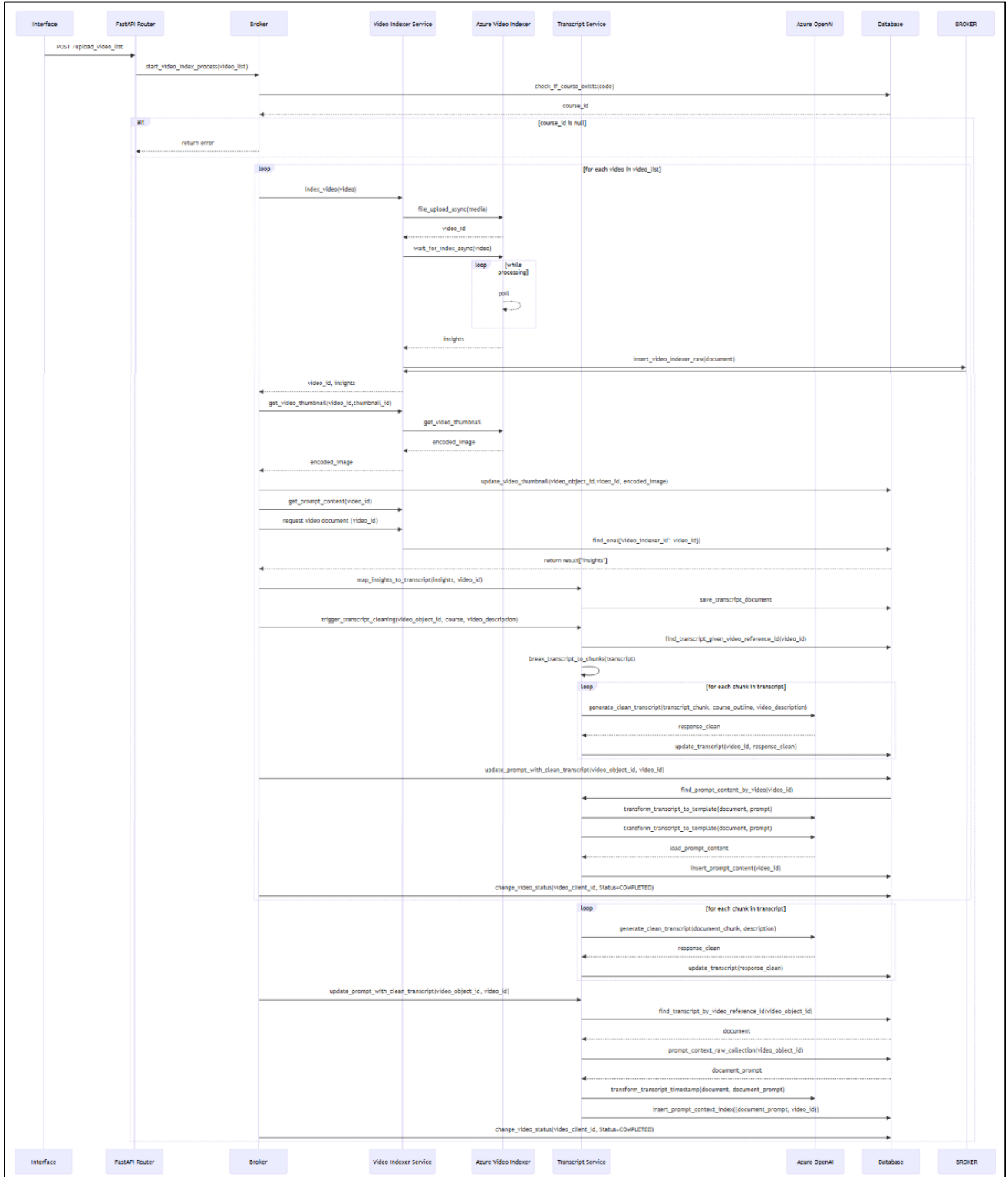


Figure 10: Video upload and Processing Sequence Diagram

This sequence diagram in Figure 10 illustrates the complete workflow of the video uploading and analysis pipeline, which begins when the user uploads videos through the interface. The FastAPI Router receives the upload request and delegates the task to the Broker Service, which orchestrates the entire backend process. The broker first validates the course information in the Cosmos DB and, for each uploaded video, triggers the Video Indexer Service to interact with the Azure Video Indexer. During this process, the video is uploaded, indexed, and periodically polled until the processing is complete. Once finished, the video insights, transcripts, and thumbnails are retrieved and stored in the database.

The Transcript Service then processes these insights to generate, clean, and segment transcripts. Each transcript chunk is refined using Azure OpenAI, which performs AI-based cleaning and context enhancement. The cleaned transcript is subsequently stored and used to generate structured prompt templates for future chat interactions. Finally, the broker updates the video's status to COMPLETED in the database, indicating that the video is ready for retrieval and chat-based question answering.

This architecture ensures reliable, automated video processing that transforms raw uploads into structured, searchable, and context-enriched learning data, enabling seamless integration with the chat system for later use.

4.5 Chat Processing and AI Intelligence

4.5.1 Sequence Diagram

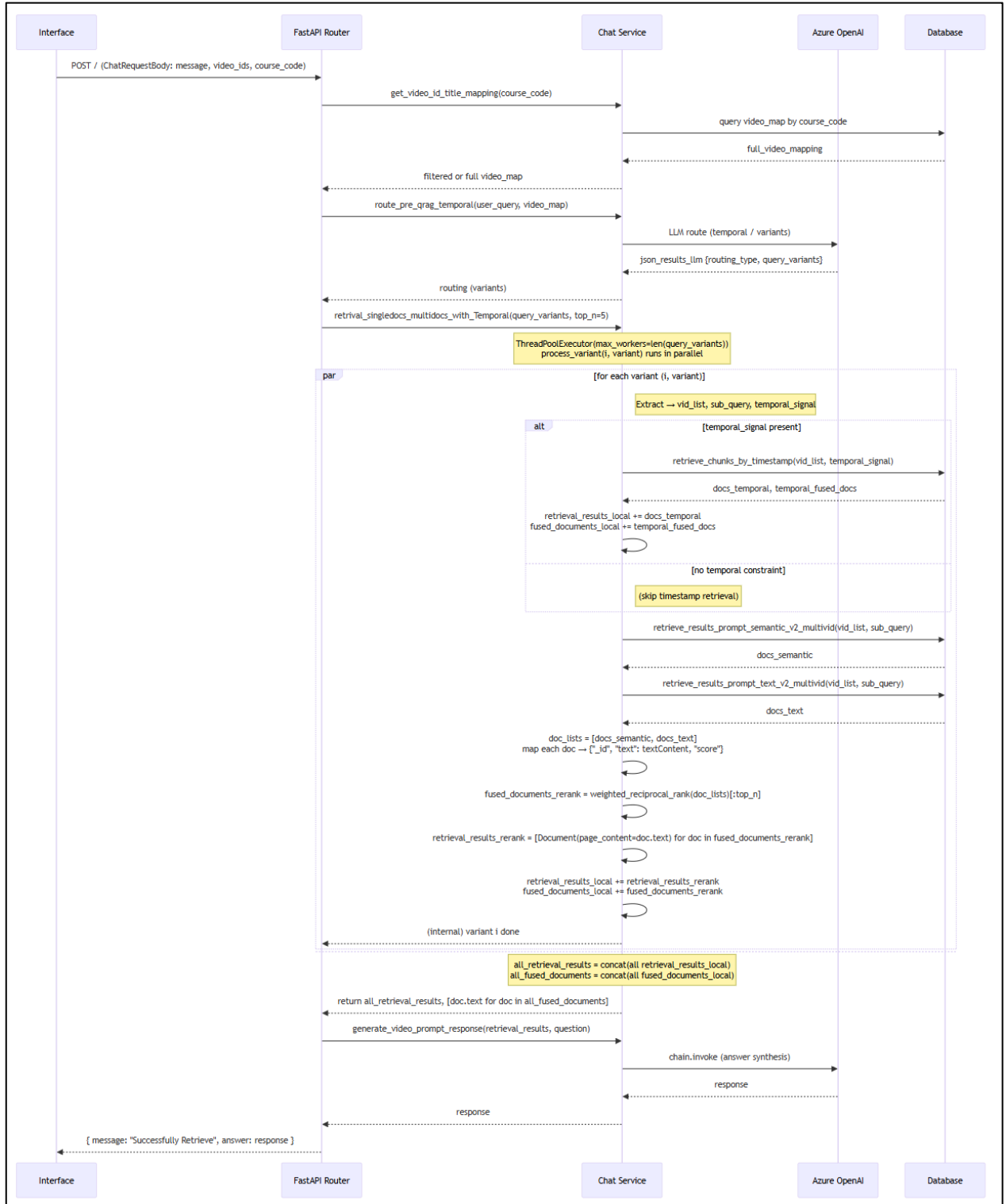


Figure 11: Chat Processing and Retrieval Flow with Doc Scope Identifier and Temporal Routing

This sequence diagram in Figure 11 illustrates the end-to-end workflow of the Chat Service, which handles user prompts and generates intelligent responses using temporal understanding and the document scope identifier mechanism. When a user submits a message through the interface, the request is sent to the FastAPI Router, which delegates processing to the Chat Service. The system first retrieves the relevant video mappings from Cosmos DB to ensure the language model can accurately map the user's video of interest to the corresponding record in the database. It then performs temporal understanding and document scope routing using Azure OpenAI to determine whether the query requires single video, multi video, or time-sensitive retrieval.

The diagram further details the internal logic of the `retrieval_single docs_multidocs_with_Temporal()` function, where multiple query variants are executed concurrently through a thread pool. Each variant may invoke timestamp-based retrieval, semantic search, and text-based search against the database. The retrieved results are normalized and fused using a Weighted Reciprocal Rank algorithm to ensure the most contextually relevant chunks are selected. Finally, the fused results are passed back to the OpenAI model, which synthesizes the final response using contextual reasoning. The resulting answer is returned through the FastAPI Router to the interface, completing the query cycle.

This flow ensures that each user query dynamically adapts to the context of multiple videos, timestamps, and retrieval modes, producing coherent and context-aware responses.

4.5.2 LLM Prompts

```
def get_prompt_preQrag_temporal():
    return """
SYSTEM ROLE:
You are a lightweight PRE-QRAG router and question rewriter for a lecture-video RAG system.

INPUTS:
- user_query = {user_query}
- video_map = {video_map} # array of [{"name": "...", "video_id": "..."}]

STEP 1 - CLASSIFY
Routing_type:
- "SINGLE_DOC": answerable from one specific lecture.
- "MULTI_DOC": needs ≥2 lectures. If unsure OR no lecture explicitly mentioned, choose "MULTI_DOC".

STEP 2 - MAP LECTURES TO video_id(s)
Resolve case-insensitive names/aliases (and "lecture N" → Nth entry in video_map) to video_id(s).
- If no lecture explicitly named → set top-level video_ids to **all** IDs in video_map (order-preserving).
- SINGLE_DOC → exactly 1 id. MULTI_DOC → ≥1 ids (deduped, order-preserving).

STEP 3 - QUESTION REWRITING
- SINGLE_DOC: produce **exactly 2** variants:
  1) Dense-optimized (semantic). 2) Sparse-optimized (keyword-heavy).
  Each variant's "video_ids" = [that single mapped id].
- MULTI_DOC: produce **exactly 2** decomposed into distinct sub-questions.
  Each sub-question should target a distinct aspect of the query, not duplicates.
  Each variant's "video_ids" = all related ids to each sub-question; if none specified, use **top-level video_ids** (i.e., all videos).

CONSTRAINTS
- Do **not** invent facts or lecture names. Queries must stay grounded in the original question.
- "video_ids" must be valid IDs from video_map.
- Top-level "video_ids" must equal the union (deduped, order-preserving) of all IDs appearing in query_variants[*].video_ids.
- Return **valid JSON only** (no comments/markdown/trailing commas).

STRICT OUTPUT (return ONLY this JSON object):
{{
  "routing_type": "SINGLE_DOC" | "MULTI_DOC",
  "user_query": "{user_query}",
  "video_ids": [...],
  "query_variants": [
    {{ "video_ids": [...], "question": "...", "temporal_signal": ["hh:mm:ss"] }},
    {{ "video_ids": [...], "question": "...", "temporal_signal": [] }}
  ]
}}
    """
```

Figure 12: Temporal and Document Scope Prompt Template

In Figure 12, it shows the system prompt used for the temporal and document Scope module, which enables the model to identify whether a user query targets a single lecture or multiple lectures while also recognizing temporal intent. It guides the model to rewrite each query into semantic and keyword-focused variants for hybrid retrieval, attach relevant timestamps when needed, and return the output in a structured JSON format. This ensures accurate, context-aware routing for both single video and multi-video queries.

```
def prompt_template_test():
    return """
You are an AI assistant that answers questions based on detailed video context. The context includes:

**Instructions:**

1. Understand the User's Question:
    - Carefully read the user's query to determine what information they are seeking.

2. Use Relevant Context:
    - Search through the provided context to find information that directly answers the question.

3. Compose a Clear and Concise Answer:
    - Provide the information in a straightforward manner.
    - Ensure the response is self-contained and understandable without needing additional information.
    - If unsure of question, ask the user to clarify again in a polite manner.
    - If unable to find answer in context, say that you are unable to find an answer in a polite manner.

4. Formatting Guidelines:
    - Begin your answer by addressing the user's question.

**History:**

{history}

**Context:**

{context}

**User's Question:**

{input}

Your Answer:
    """
```

Figure 13: Prompt Template for Response Generation Using Video Context

Figure 13 shows the prompt template used for generating answers within the Chat Service. The template provides structured instructions that guide the language model to understand the user's question, extract relevant information from the provided video context, and compose a clear and concise response. It also defines formatting rules, ensuring that the generated answers remain self-contained, polite, and contextually grounded. By including placeholder's context and user input, this prompt enables consistent and coherent response generation across different chat interactions.

4.6 User Interface

4.6.1 Video Chatbot Page

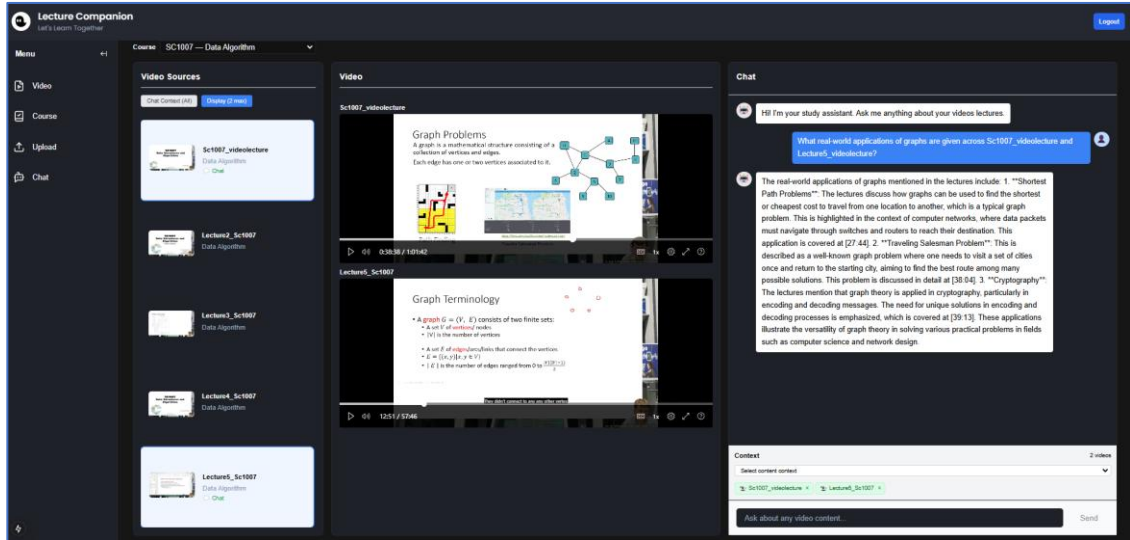


Figure 14: Video Chatbot Page

The Video Chatbot page integrates interactive learning and intelligent assistance within a single interface. Users can select one or multiple videos as context sources, allowing the chatbot to retrieve relevant information across lectures. The interface is divided into three main sections: the video list panel on the left, the video player area in the center, and the chat interface on the right. This layout enables users to watch lecture videos while simultaneously interacting with the chatbot, asking questions related to specific segments or across multiple videos. The chatbot dynamically references the selected video contexts to generate accurate, time-aligned responses, while maintaining a conversational flow. This design promotes an active learning experience, enabling students to review lectures, clarify concepts, and engage with course material in real time.

4.6.2 Video Landing Page

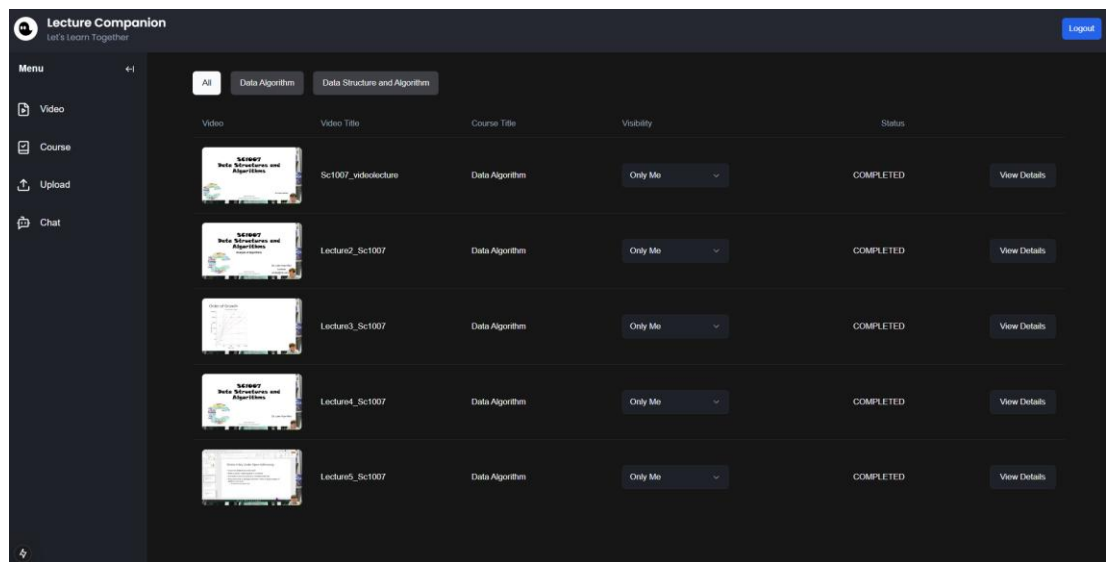


Figure 15: Video Chatbot Landing Page

The video landing page was designed to provide users with an organized overview of their uploaded and processed lecture videos within the Lecture Companion platform. The interface adopts a clean, dashboard-style layout with a dark theme for better visual comfort and focus. A left-hand navigation menu allows quick access to key features such as Video, Course, Upload, and Chat, ensuring consistent navigation throughout the platform.

At the center, videos are presented in a structured tabular view containing attributes such as Video Title, Course Title, Visibility, and Status. Each entry includes a “View Details” button that links to the individual video’s analysis or transcript page. A filter bar at the top enables users to toggle between different course categories improving discoverability and content management.

4.6.3 Course Management Page

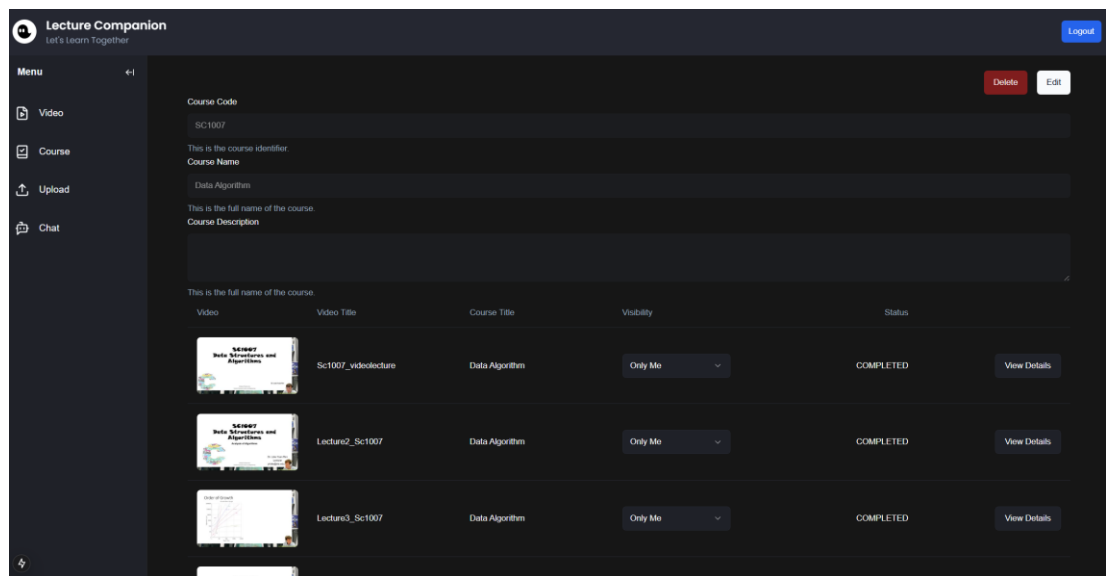


Figure 16: Video Chatbot Course Management Page

The Course Management page was designed to allow users to view, edit, and manage course-level information in an intuitive and consistent interface. The layout follows a two-section structure, the upper section displays editable course metadata such as Course Code, Course Name, and Course Description, while the lower section lists all associated lecture videos in a table format. Each video entry includes details like Title, Course Title, Visibility, and Status, along with a “View Details” button for deeper interaction. The interface uses a dark theme to maintain visual harmony with the rest of the platform, emphasizing readability and focus. The Edit and Delete buttons are positioned prominently to allow quick course updates or removal, with clear visual cues for action safety.

4.6.4 Video Upload Page

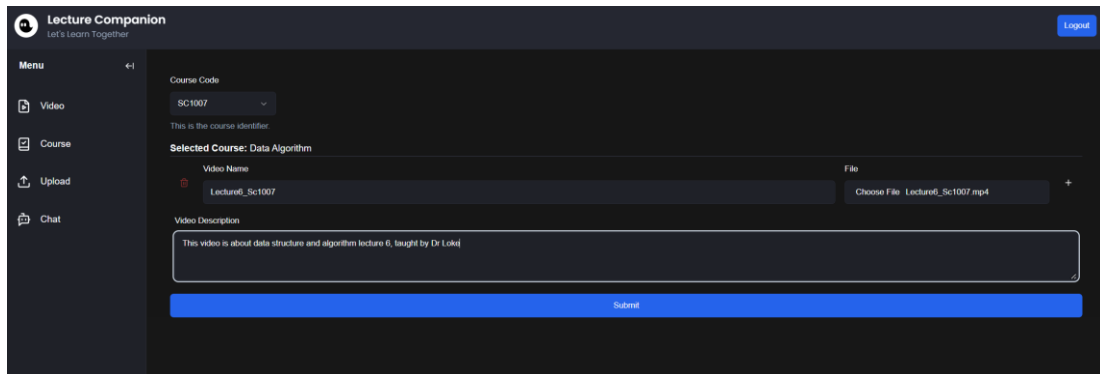
The screenshot shows the 'Lecture Companion' web application interface. On the left is a dark sidebar with a 'Menu' section containing icons and labels for 'Video', 'Course', 'Upload', and 'Chat'. The main content area has a dark background. At the top left of this area is the 'Lecture Companion' logo and tagline 'Let's Learn Together', and at the top right is a 'Logout' button. The form consists of several fields: a 'Course Code' dropdown menu with 'SC1007' selected; a text label 'This is the course identifier.'; a 'Selected Course: Data Algorithm' label; a 'Video Name' input field with 'Lecture0_Sc1007' entered; a 'File' section with a 'Choose File' button and the filename 'Lecture0_Sc1007.mp4'; and a 'Video Description' text area containing the text 'This video is about data structure and algorithm lecture 6, taught by Dr. Lokid'. A prominent blue 'Submit' button is at the bottom of the form.

Figure 17: Video Chatbot Upload Page

The Upload Video page was designed to provide a straightforward and efficient interface for adding new lecture videos to the platform. The layout follows a form-based structure, allowing users to select an existing course, input a video name, attach a video file, and include a short description. Each input field is clearly labeled to ensure ease of use and prevent upload errors. The page maintains the platform's dark-themed, minimalist design, with a bright blue "Submit" button to highlight the main call-to-action. Upon submission, the form data and video file are processed through a backend API, where metadata such as course association and file name are stored in the database. This design emphasizes clarity, accessibility, and responsiveness, ensuring that users can quickly upload and manage their lecture videos within the Lecture Companion system.

5.1 RAGAS Temporal Questions Results

5.1.1 Temporal Qualitative Results

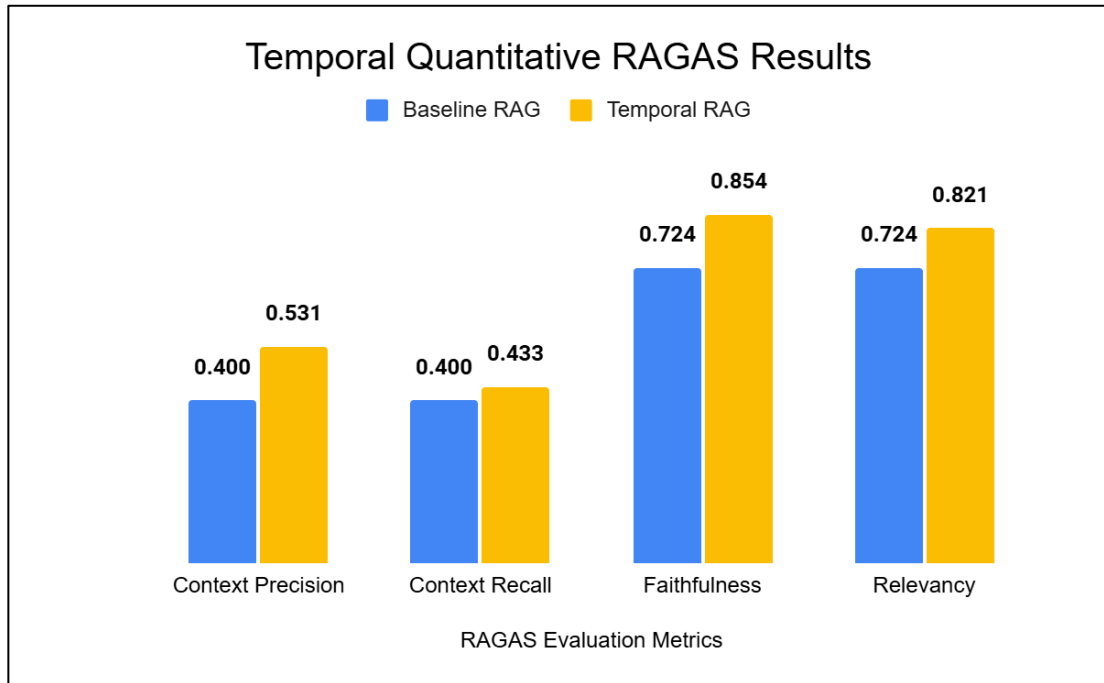


Figure 18: Temporal Quantitative RAGAS Results Bar Chart

The quantitative results demonstrate that the Temporal RAG outperforms the Baseline RAG across all four RAGAS evaluation metrics when tested on 20 temporal-based questions. Specifically, Context Precision improved from 0.400 to 0.531, and Context Recall increased slightly from 0.400 to 0.433, indicating that the Temporal RAG retrieves more accurate and relevant contextual chunks with higher coverage. The Faithfulness metric rose significantly from 0.724 to 0.854, suggesting that the generated responses align more closely with the retrieved context and ground truth. Similarly, Relevancy improved from 0.724 to 0.821, showing that the answers produced are more pertinent to the questions. Overall, the numerical gains reflect stronger temporal understanding and retrieval precision. The full results are provided in Appendix D and E.

Table 6: Baseline RAG vs Temporal RAG time taken

Pipeline	Time Taken (Seconds)
Baseline RAG	3.68
Temporal RAG	4.93

In terms of timing, the results show that the Baseline RAG achieves a slightly faster response time compared to the Temporal RAG, with an average of 3.68 seconds versus 4.93 seconds. This minor delay is a reasonable trade-off for the substantial gains in retrieval quality and overall response reliability.

5.1.2 Temporal Qualitative Analysis

<p>Question:</p> <p>What is mentioned at 33 minutes of the lecture?</p> <p>Ground Truth:</p> <p>At the 33-minute mark, the lecturer mentioned that the module will mainly cover problem types such as searching, graph problems, and combinatorial problems involving permutations.</p>			
<p>Baseline RAG answer: I am unable to find an answer for the specific content mentioned at 33 minutes of the lecture. The provided transcript does not include any information from that timestamp. If you have any other questions or need information from a different part of the lecture, feel free to ask!</p>			
Context Precision	Context Recall	Faithfulness	Relevancy
0	0	0.666667	0
<p>Temporal Identifier RAG Answer: At 33 minutes of the lecture, the speaker discusses the various types of problems covered in the module, including searching, graph problems, and combinatorial problems, which involve permutations and combinations. They also mention that in the next level, topics such as sorting strings, processing, and geometric and numerical problems will be introduced. This section emphasizes the breadth of topics that will be explored in the course. This information is "Covered at 33:19."</p>			
Context Precision	Context Recall	Faithfulness	Relevancy
0.5	1	1	0.959078

Figure 19: Responses generated by the Baseline RAG and the Temporal Identifier RAG for a timestamp-based query.

From a qualitative standpoint, the enhancements introduced in Temporal Identifier RAG have proven highly effective in addressing the temporal reasoning limitations of the baseline system. As shown in Figure 19, the comparison between the responses generated by the Baseline RAG and the Temporal Identifier RAG for a timestamp-based query demonstrates the improvement in contextual retrieval. The Temporal

RAG successfully identified and retrieved the relevant segment of the lecture corresponding to the 33-minute mark, enabling it to generate a precise and time-aligned response.

The improved performance across all metrics highlights that the new temporal-aware retrieval mechanism successfully captured time-dependent information and selected more contextually aligned chunks. These results indicate that the integration of temporal awareness into the RAG pipeline has significantly strengthened both retrieval accuracy and response quality, leading to a more reliable and contextually intelligent system. Despite taking marginally longer per query, the performance uplift clearly demonstrates that the modifications introduced in the Temporal RAG deliver a substantial qualitative advantage.

5.2 RAGAS Single Docs and Multi Docs Question Results

5.2.1 Single Docs Quantitative Analysis

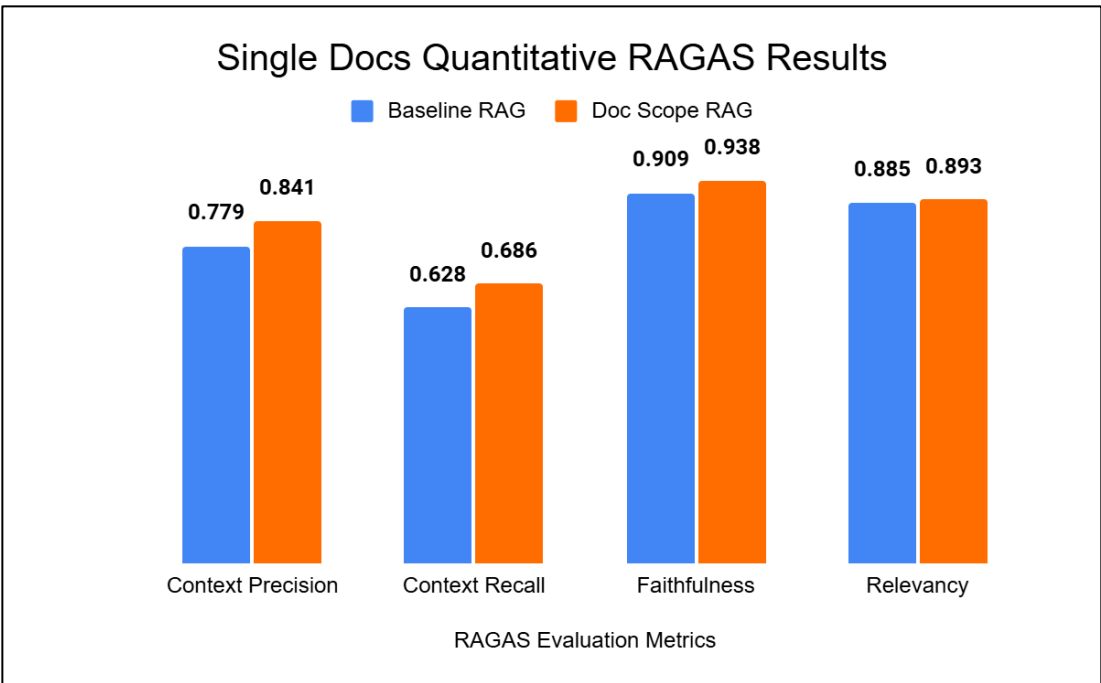


Figure 20: Single Docs Quantitative RAGAS Results Bar Chart

The evaluation of the Doc Scope RAG against the Baseline RAG on thirty single document questions demonstrates consistent performance improvements across most

RAGAS metrics. Context Precision increased from 0.704 to 0.730, showing that Document-Scope RAG retrieved more accurate and relevant chunks. Context Recall remained nearly constant, moving slightly from 0.506 to 0.508, indicating that while the coverage of retrieved context did not expand significantly, it maintained stability despite additional query reformulation steps. Notably, Faithfulness improved from 0.810 to 0.879, suggesting that the answers generated were more consistent with the retrieved evidence and less prone to hallucination. Similarly, Relevancy rose from 0.760 to 0.826, reflecting better alignment between the user queries and the generated responses. The full results are provided in Appendix F and G.

Table 7: Baseline RAG vs Docs Scope RAG time taken (single docs)

Pipeline	Time Taken (Seconds)
Baseline RAG	4.80
Docs Scope RAG	7.35

In terms of processing efficiency, the single and multi-RAG recorded an average response time of 7.35 seconds per question, compared to 4.80 seconds for the Baseline RAG, indicating an increase of approximately 2.5 seconds. The marginal increase in latency represents a worthwhile trade-off for the greater contextual accuracy and faithfulness achieved by the single and multi-RAG, highlighting the efficiency performance balance of the new pipeline.

5.2.2 Multi Docs Quantitative Analysis

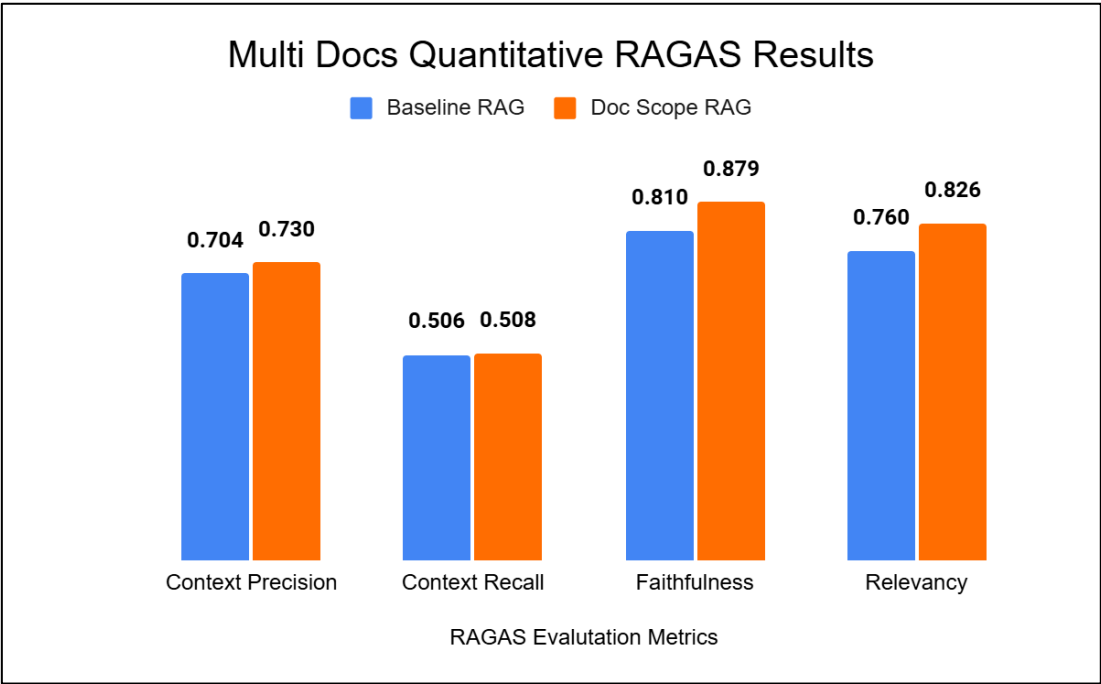


Figure 21: Multi Docs Quantitative RAGAS Results Bar Chart

The quantitative evaluation of the Doc Scope RAG compared to the Baseline RAG on 30 multi-document questions shows consistent improvements across all major RAGAS metrics. Context Precision increased from 0.704 to 0.730, reflecting better retrieval accuracy and more precise context selection from multiple sources. Context Recall remained stable, moving slightly from 0.506 to 0.508, suggesting that while the coverage of retrieved information was maintained, the new retrieval logic did not compromise the breadth of contextual evidence. Significant gains were observed in Faithfulness, which rose from 0.810 to 0.879, indicating that the responses generated were more grounded in the retrieved context and less prone to hallucination. Likewise, Relevancy improved from 0.760 to 0.826, showing that the answers were more semantically aligned with the user’s questions. The full results are provided in Appendix H and I.

Table 8: Baseline RAG vs Docs Scope RAG time taken (multi docs)

Pipeline	Time Taken (Seconds)
Baseline RAG	4.88
Doc Specific RAG	7.48

In terms of timing, the Doc Scope RAG took an average of 7.48 seconds per question, compared to 4.88 seconds for the Baseline RAG. This additional 2.6 seconds can be attributed to the extra pre-retrieval query reformulation and multi-document fusion processes. Despite the longer response time, the measurable improvements in precision, faithfulness, and relevancy make this delay a worthwhile trade-off, as it leads to more contextually rich and accurate responses, especially valuable for multi-document reasoning tasks.

5.2.3 Single Docs and Multi Docs Quantitative Analysis

Single Routing Result – Question	
Routing Type:	SINGLE_DOC
User Query:	In Lecture5_Sc1007, what topics are covered in this lecture?
Mapped Video:	zwb6lqhpz1
Generated Query Variants	
Variant 0:	
Question:	Can you summarize the main concepts and key topics discussed throughout Lecture5_Sc1007 Data Structures and Algorithms?"
Video ID:	zwb6lqhpz1
Retrieved:	3 context chunks
Variant 1:	
Question:	List all topics covered in Lecture5_Sc1007, such as data structures, algorithms, or examples mentioned.
Video ID:	zwb6lqhpz1
Retrieved:	3 context chunks

Figure 22: Single Document Routing and Query Variants Generation

Figure 22 shows the routing result of a single document query within the system. The user's question is first classified under the SINGLE_DOC routing type and mapped to the corresponding lecture video ID. The system then generates two reformulated query variants. One semantically focused and one dense or keyword

focused. Each variant retrieves the top three context chunks from the mapped lecture video for downstream processing.

<p>Question:</p> <p>In Lecture5_Sc1007, what topics are covered in this lecture?</p> <p>Ground Truth:</p> <p>Graph Representation, specifically Adjacency Matrix and Adjacency List (mentioned in 0:09:48).</p>			
<p>Baseline RAG answer: In Lecture5_Sc1007, the topics covered include the traversal of graphs, specifically focusing on two methods: breadth-first search and depth-first search. The lecturer mentions that while there are many ways to traverse graphs, only these two methods will be introduced in the lecture. This is highlighted when the lecturer states, "we only introduce the breadth-first search and depth-first search in the next lecture" and elaborates on the nature of these traversals at timestamps (57:07.48) and (57:29.48).</p>			
Context Precision	Context Recall	Faithfulness	Relevancy
0	0	1	0.9029132
<p>Temporal Identifier RAG Answer: In Lecture5_Sc1007, the topics covered include an introduction to graphs, how to represent them, and specifically two representations: the adjacency matrix and the adjacency list. Additionally, there is mention of traversing graphs, with a focus on breadth-first search and depth-first search, which may be covered in the next lecture if time allows. This is outlined at various points in the lecture, particularly at "Covered at 09:04" for the introduction to graphs and "Covered at 09:35" for the representations. The traversal methods are mentioned towards the end, with a specific note on their introduction in the next lecture at "Covered at 57:07"</p>			
Context Precision	Context Recall	Faithfulness	Relevancy
0.916667	0.5	1	0.981168

Figure 23: Comparison of Baseline RAG and Temporal Identifier RAG Outputs

It presents a side-by-side comparison of responses generated by the Baseline RAG and the Temporal Identifier RAG for the same question. It includes the question, ground-truth reference, both generated answers, and their corresponding RAGAS evaluation metrics.

Multi Routing Result – Question	
Routing Type:	MULTI_DOC
User Query:	What guidance is given about tutorials, labs, and coding across Sc1007_Videolecture and Lecture2_videolecture?
Mapped Videos:	zwb6lqhpz1 (Sc1007_Videolecture) tg2fy923h1 (Lecture2_videolecture)
Generated Query Variants	
Variant 0:	
Question:	"What guidance is provided on tutorials in Sc1007_Videolecture and Lecture2_videolecture?"
Video IDs:	zwb6lqhpz1, tg2fy923h1
Retrieved:	3 context chunks
Variant 1:	
Question:	"What information is available regarding labs and coding in Sc1007_Videolecture and Lecture2_videolecture?"
Video IDs:	zwb6lqhpz1, tg2fy923h1
Retrieved:	3 context chunks

Figure 24: Multi-Document Routing and Query Variants Generation

Figure 24 displays the result of the multi-document routing process for a multi-document query. The user’s question is classified under the MULTI_DOC routing type, mapping it to two relevant lecture videos. The system then generates two reformulated query variants.

Question: What guidance is given about tutorials, labs, and coding across Sc1007_videolecture and Lecture2_videolecture ?

Ground Truth: In Sc1007_videolecture, the lecturer explained that tutorials focus on concepts while labs focus on practice, and in Lecture2_videolecture the lecturer explained that complexity analysis is taught in lecture but coding is applied in labs.

Baseline RAG answer: In Lecture5_Sc1007, the topics covered include the traversal of graphs, specifically focusing on two methods: breadth-first search and depth-first search. The lecturer mentions that while there are many ways to traverse graphs, only these two methods will be introduced in the lecture. This is highlighted when the lecturer states, "we only introduce the breadth-first search and depth-first search in the next lecture" and elaborates on the nature of these traversals at timestamps (57:07.48) and (57:29.48).

Context Precision	Context Recall	Faithfulness	Relevancy
0.588889	0.5	1	0.969175

Temporal Identifier RAG Answer: "In the Sc1007_videolecture, several key points are made regarding tutorials, labs, and coding: 1. **Tutorials**: The tutorials are designed to help students clarify doubts they may have encountered during lab sessions. There is an opportunity to meet biweekly for further clarification, and students are encouraged to reach out via email if they still have questions after these sessions. This is emphasized at the timestamp ""Covered at 13:58"". 2. **Labs**: Attendance in labs is important, although the lab itself is not graded. However, not attending may negatively impact performance in the lab tests. This is highlighted at ""Covered at 08:17"". Additionally, lab tests will assess the knowledge gained in labs, and attendance will be taken, but it does not guarantee a good performance in tests if labs are skipped (""Covered at 08:34"". 3. **Coding**: The emphasis is placed on learning how to code rather than just reading solutions. Students are encouraged to implement solutions themselves to truly understand the material. This guidance is provided at ""Covered at 01:00:53"". The importance of understanding time complexity and space complexity in coding is also mentioned, particularly in relation to algorithm analysis (""Covered at 57:39"". Overall, the guidance stresses the importance of active participation in tutorials and labs, as well as the necessity of hands-on coding practice to grasp the concepts effectively."

Context Precision	Context Recall	Faithfulness	Relevancy
1	1	1	0.935147

Figure 25: Comparison of Baseline RAG and Temporal Identifier RAG for Multi-Document Query

Figure 25 presents a comparison between the Baseline RAG and Temporal Identifier RAG outputs for the same multi-document question. It includes the original query, ground-truth reference, generated responses, and corresponding RAGAS evaluation metrics.

From a qualitative perspective, both the single document and multi-document testing results highlight the strong impact of the newly introduced Document Scope Identifier module in improving contextual understanding, reasoning accuracy, and response coherence compared to Baseline RAG.

5.3 User Evaluation

A user evaluation survey was conducted with 13 NTU students from the College of Computing and Data Science (CCDS) to evaluate the application and gather feedback through Google Forms. The majority had previously taken the SC1007: Data

Structures and Algorithms module, which served as the primary video dataset for the system. Their familiarity with the course content allowed for more relevant and contextual feedback on the chatbot’s retrieval and reasoning capabilities.

The survey comprised four sections: temporal queries, single document queries, multi-document queries, and overall satisfaction. Responses were rated on a five-point Likert scale, where 1 indicated “least satisfied” and 5 indicated “most satisfied”. The results of each section are presented in the following subsections.

5.3.1 Temporal Queries Satisfaction

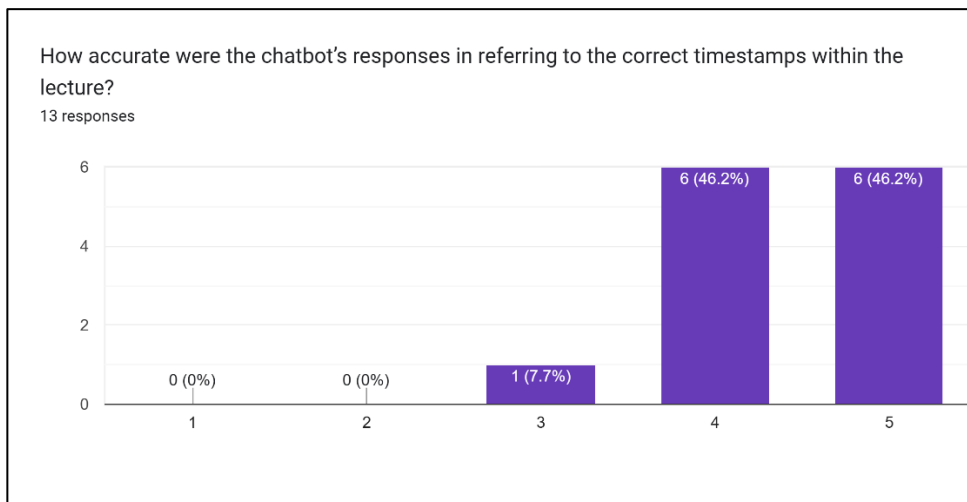


Figure 26: Temporal Queries Satisfaction Survey Question 1

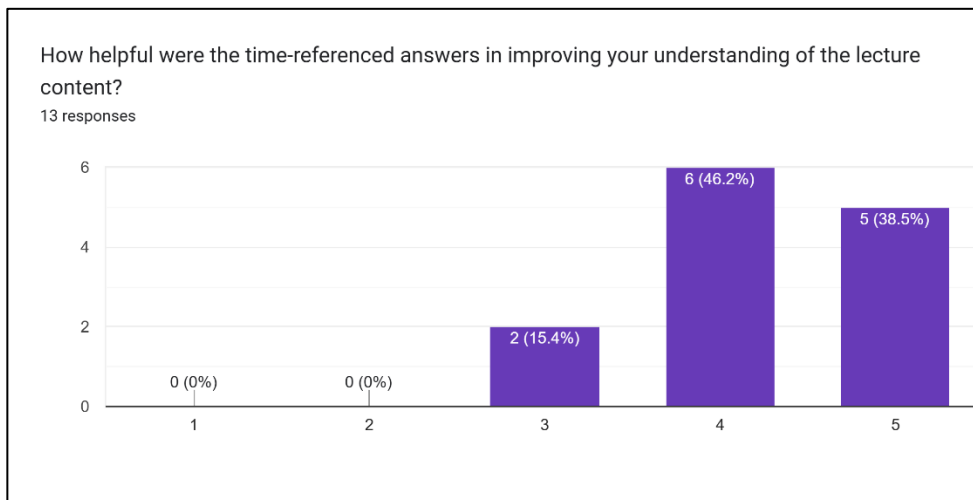


Figure 27: Temporal Queries Satisfaction Survey Question 2

5.3.2 Single Docs Queries Satisfaction

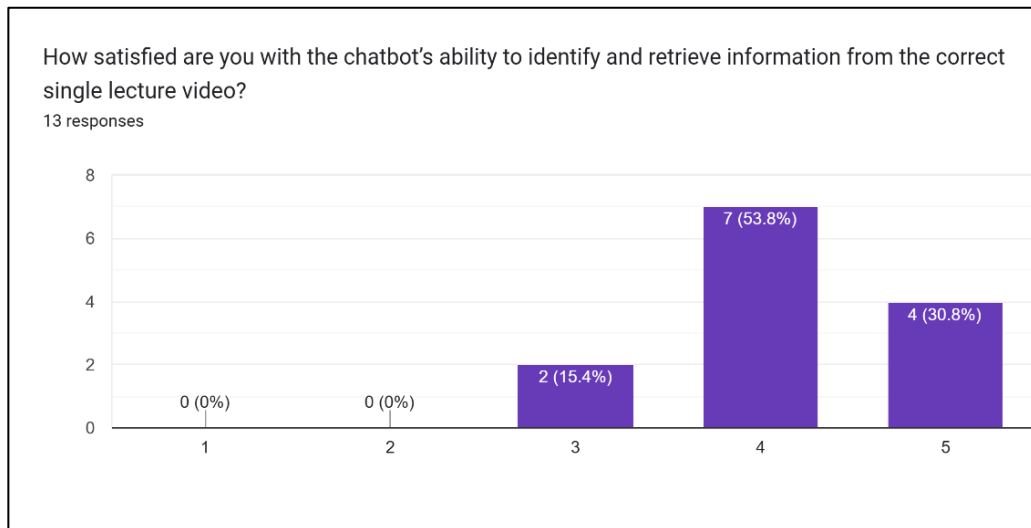


Figure 28: Single Docs Queries Satisfaction Survey Question 1

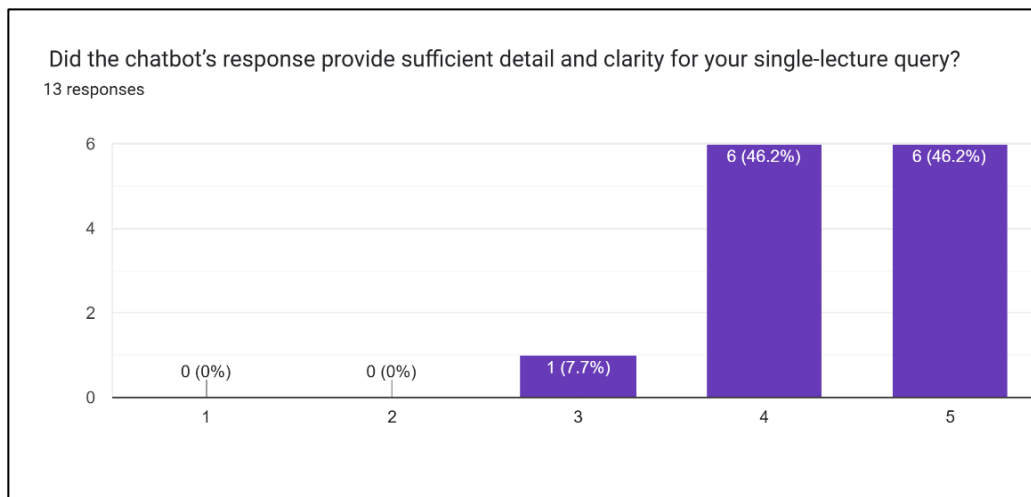


Figure 29: Single Docs Queries Satisfaction Survey Question 2

5.3.3 Multi Docs Queries Satisfaction

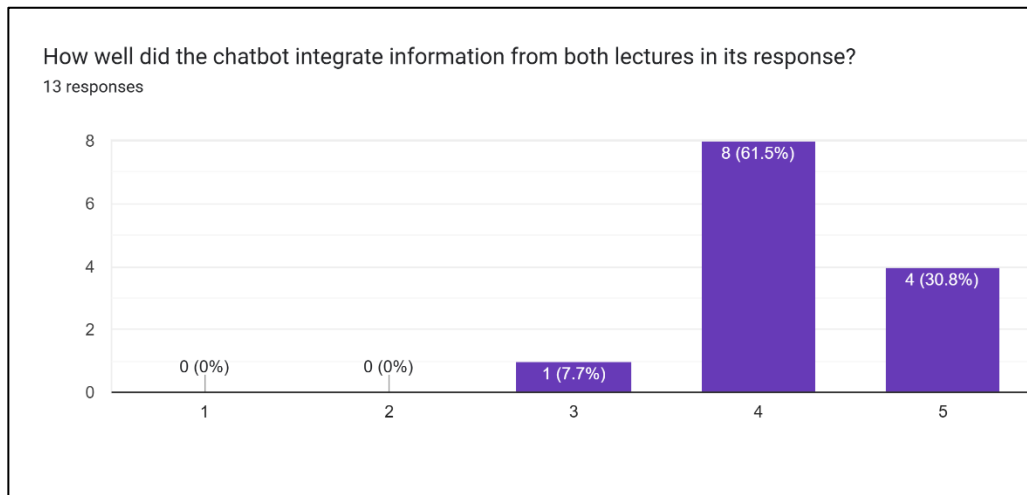


Figure 30: Multi Docs Queries Satisfaction Survey Question 1

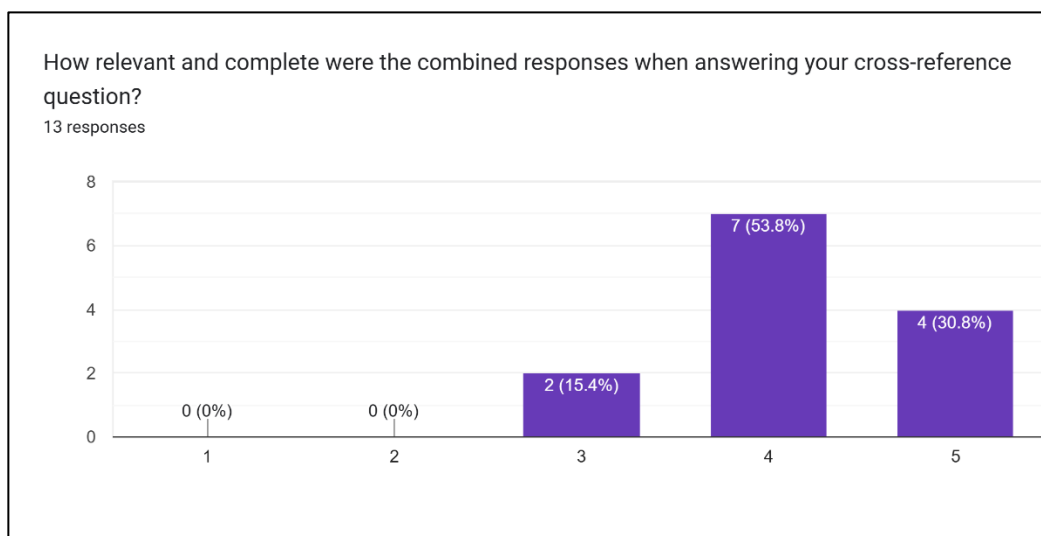


Figure 31: Multi Docs Queries Satisfaction Survey Question 2

5.3.4 Overall Satisfaction

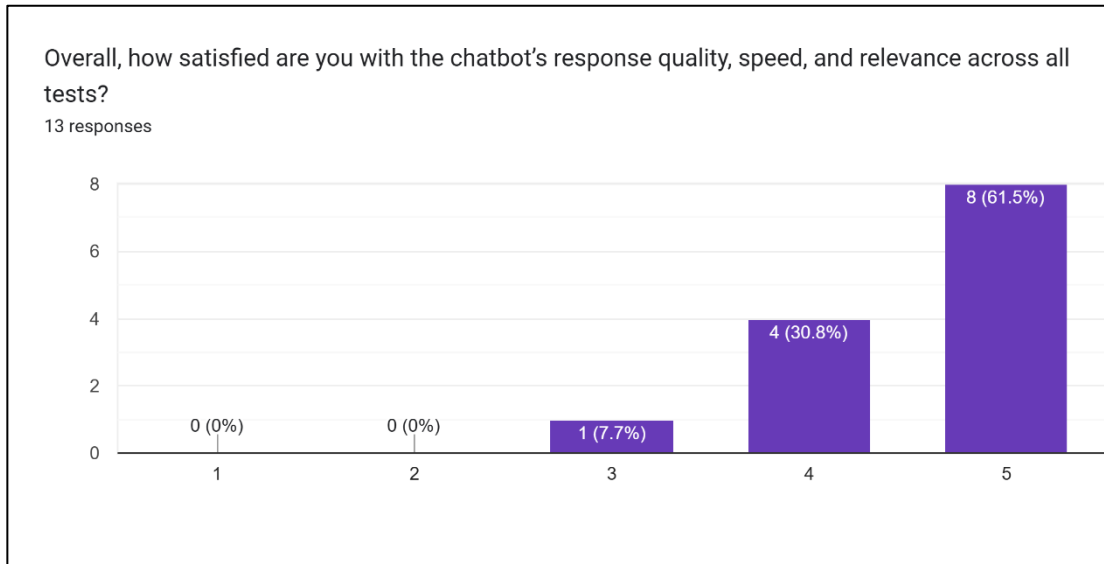


Figure 32: Overall Satisfaction Survey Question 1

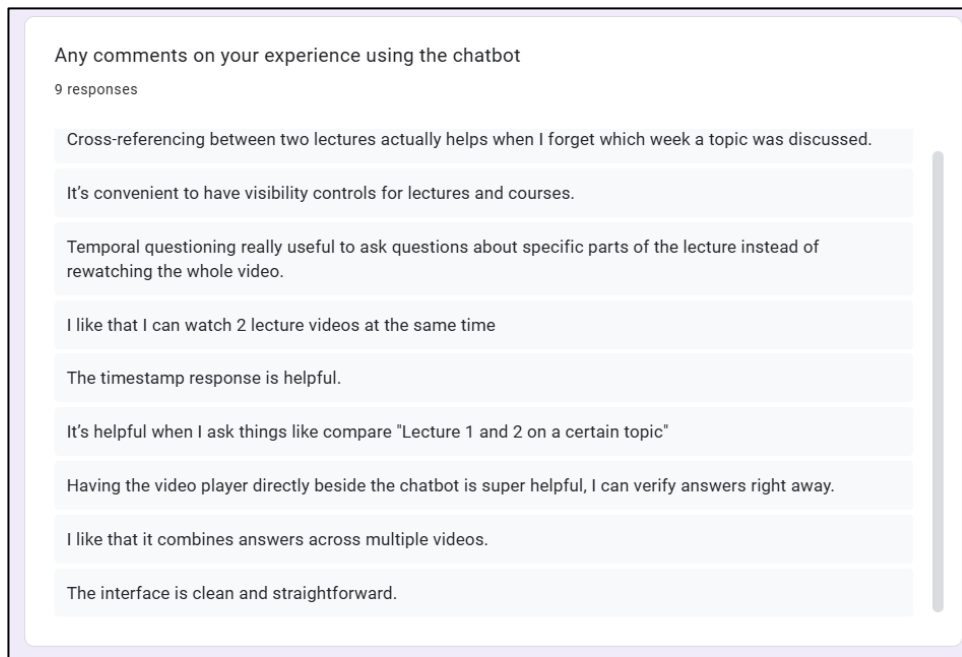


Figure 33: Overall Satisfaction Survey Question 2

To further interpret user experience, participants were asked to provide open-ended comments on their overall impressions of the system. As shown in Figure 33, the responses were highly positive, highlighting specific features that users found beneficial. Many participants appreciated the ability to cross-reference between two lecture videos, describing it as helpful for recalling when a specific topic was

discussed. Others emphasized that temporal questioning made it convenient to locate particular lecture segments without rewatching entire videos. The timestamp-based responses and side-by-side video and chatbot layout were also noted as useful, allowing students to verify answers quickly and interactively.

Overall, the user feedback indicates that the enhanced Document-Scope Identifier feature, which allows retrieval across multiple videos, contributed significantly to user satisfaction. Compared to the baseline system that supported only single video retrieval, the improved version provided greater flexibility for comparative and cross-lecture queries.

5.3.5 *Future Enhancement*



What improvements or additional features would you suggest for future versions of this chatbot?

11 responses

- It would be better if it mentioned the slide title or topic name together with the timestamp.
- Support lecture slide recognition, where i can add in my pdf slides for context
- Show transcript text alongside the video when answering. So i can read what exacly the prof explained
- Cross-video queries take a bit longer to load, maybe show a progress indicator
- Generate short quizzes based on a lecture segment.
- Add a feature to summarize an entire lecture automatically.
- Make up quizzes to test our understanding
- Response is slightly slow when comparing multiple videos.
- Integrate the video transcript under the chatbot for reference.

Figure 34 Future Enhancement Survey Question

Participants generally expressed satisfaction with the chatbot's temporal referencing and cross-video retrieval capabilities. However, they suggested several improvements to enhance usability and learning support. These include adding slide or topic titles alongside timestamps, supporting lecture slide recognition by uploading PDF slides, and displaying transcript text next to the video for easier reference. Some also noted that cross-video queries load slightly slower and recommended adding a progress indicator. Other proposed features include automatic lecture summarization, short

quizzes or question generation, and transcript integration beneath the chatbot interface. These insights will guide the next phase of system enhancement.

CHAPTER 6: LIMITATION AND FUTURE DEVELOPMENT

6.1 *Integration of Tutorial and Assignment Contexts*

A key direction for future development involves extending the system's knowledge base to incorporate tutorial sheets, lab assignments, and lecture slides. By integrating these supplementary learning materials, the chatbot can generate more holistic and context-aware responses that connect theoretical concepts from lectures with their corresponding applications in practical exercises. For instance, when a student uploads a tutorial question, the system can retrieve related explanations or examples discussed in the relevant lecture segments. This feature not only reinforces conceptual understanding but also allows students to bridge the gap between lecture theory and hands-on practice. Additionally, aligning tutorial content with lecture discussions can help identify learning gaps and provide targeted assistance, ultimately enhancing personalized learning outcomes.

6.2 *Lecture Slide and PDF Upload Functionality*

Another promising enhancement is the implementation of a document upload feature that allows both instructors and students to upload supplementary materials such as lecture slides, tutorial notes, or reference readings in PDF format. The system can leverage Optical Character Recognition (OCR) and semantic embedding techniques to extract text and key concepts from these documents, transforming them into searchable knowledge entries within the chatbot's retrieval framework. This integration expands the system's context base beyond video transcripts, enabling multi-modal retrieval that combines visual and textual content. As a result, the chatbot can generate richer, more comprehensive responses that draw from multiple types of learning materials. Furthermore, this development supports scalability, as new lecture materials can be continuously added to the system without requiring extensive manual preprocessing.

CHAPTER 7: CONCLUSION

In conclusion, this project successfully demonstrated the integration of Large Language Models with video-to-text processing using Azure's AI ecosystem, resulting in a functional and scalable Learning Companion and Co-Pilot. Through the implementation of hybrid retrieval and LLM-based document transformation, coupled with systematic evaluation using the RAGAS framework, the system achieved strong contextual relevance and reliability in lecture-based question answering. The results highlight the potential of Retrieval-Augmented Generation in improving educational interactivity and accessibility for students. At the same time, the findings also reveal areas that can be further enhanced, particularly in handling complex, cross-topic, and time-dependent queries. Moving forward, the insights and outcomes from this project provide a solid foundation for future advancements in AI-driven learning environments, where intelligent systems can not only answer questions but also support personalized understanding, adaptive learning, and deeper engagement in digital education.

REFERENCES

- [1] A. v. d. B. J. C. R. a. L. W. v. M. S. Groothuijsen, "AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning," 2024.
- [2] W. X. Lee, "LLM-based Learning Companion & Co-Pilot – A Video to Text Approach," Singapore, 2025.
- [3] M. G. a. L. M. L. Labadze, "Role of AI chatbots in education: systematic literature review," *Int. J. Educ. Technol. High. Educ.*, vol. 20, no. 56, 2023.
- [4] M. Corporation, "Azure Video Indexer Overview," 2025.
- [5] N. Zeng, H. Hou, F. R. Yu, S. Shi and Y. T. He, "SceneRAG: Scene-level Retrieval-Augmented Generation for Video Understanding," arXiv, Shenzhen, 2025.
- [6] V. Wrobel, S. Kazemi, F. Hammerschmidt, T. Hannemann, G. Stange, R. Geislinger and S. M. Yimam, "LECTURE4ALL: A Lightweight Approach to Precise Timestamp Detection in Online Lecture Videos," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 2025.
- [7] T. Varshney and A. Surla, "An Easy Introduction to Multimodal Retrieval-Augmented Generation for Video and Audio," 2024.
- [8] P. J. L. Ammann, J. Golde and A. Akbik, "Question Decomposition for Retrieval-Augmented Generation," Berlin, Germany (Humboldt University of Berlin), 2025.
- [9] J. Wang, "Better RAG Results with Reciprocal Rank Fusion and Hybrid Search," 2024.
- [10] L. Gao, X. Ma, J. Lin and J. Callan, "Zero-Shot Dense Retrieval without Relevance Labels," arXiv, 2022.

- [11] X. Chen, G. Tang, Y. He and Y. Zhang, "Self-Retrieval: End-to-End Information Retrieval with One Large Language Model," arXiv, 2024.
- [12] S. Es, J. James, L. Espinosa-Anke and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," arXiv, 2023.
- [13] S. Es, "Evaluating RAG Applications with RAGAS," Medium – Data Science Publication, 2023.
- [14] Y. X. X. G. K. J. J. P. Y. B. Y. D. J. S. Q. G. M. W. & H. W. unfan Gao, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv, 2023.
- [15] P. E. Guide, "Retrieval Augmented Generation (RAG) for LLMs," 2024.

APPENDIX

Appendix A

Temporal Reasoning Question Full Dataset

No.	Question	Ground Truth
1	What is mentioned at 33 minutes of the lecture?	At the 33-minute mark, the lecturer mentioned that the module will mainly cover problem types such as searching, graph problems, and combinatorial problems involving permutations.
2	What was discussed at 27:00 of the lecture?	The lecturer discussed the Fibonacci sequence, using it as an example to illustrate algorithmic thinking and recursive problem-solving.
3	When was the difference between algorithm and program discussed?	The difference between an algorithm and a program was discussed around 21:00
4	What concept is explained around 45:00 into the lecture?	Around the 45-minute mark, the lecturer explains stable sorting algorithms, highlighting that they preserve the relative order of repeated elements during sorting, using student marks across modules as an example.
5	At what point in the lecture does it start discussing How do we solve the different searching, graph and combinatorial problems?	It begins discussing how to solve different searching, graph, and combinatorial problems around the 53-minute mark, introducing data structures and algorithmic strategies like brute force, divide and conquer, greedy, and more.
6	What does the lecturer say right after the algorithm Vs program at 23 minutes in the video?	Right after explaining the difference between an algorithm and a program around the 23-minute mark, the lecturer introduces a simple example of summing numbers from 1 to N using different algorithmic approaches, such as a for loop, a mathematical formula, and recursion.
7	Does the lecturer explain graph problems before or after Combinatorial problems?	The lecturer explains graph problems before combinatorial problems.
8	What was discussed before Learning outcomes?	Before discussing the learning outcomes, the lecturer went through the course schedule.

9	What topic is discussed right after Algorithm Design Strategies?	Right after algorithm design strategies, the lecturer summarizes the overview of the lecture and discusses what will be taught in the next few weeks in this module.
10	Which topic comes just before the explanation of the Sorting Problem?	The combinatorial problem is discussed just before the explanation of the sorting problem.
11	Which week or lecture covers the topic Hash Tables?	The topic of Hash Tables is covered in Week 8 of the lecture.
12	During which part of the lecture (start/middle/end) is Computer Science Programme Structure discussed?	The Computer Science Programme Structure is discussed at the beginning of the lecture.
13	What is covered in the last 5 minutes of the lecture?	In the last 5 minutes of the lecture, the lecturer emphasizes the importance of practicing coding independently rather than relying on provided solutions, highlighting that true understanding comes from implementing algorithms yourself. He also concludes the lecture and ends the live stream.
14	What was discussed at the start of the lecture?	At the start of the lecture, Dr. Loke gave an introduction to the module, explained the format of the live stream and lecture notes, and outlined the topics to be covered, including the analysis of algorithms, hash tables, and graph problems in the second half of the module.
15	What was discussed at the end of the lecture?	A Question-and-answer session with the students, where the lecturer discussed how labs and tutorials could support their learning in the module.

16	What was said 2 minutes before Problem Type was introduced?	Two minutes before the Problem Type was introduced, the Fibonacci Sequence was discussed.
17	When did the lecturer mention the learning outcomes of this course?	The lecturer mentioned the learning outcomes of the course between 07:00 and 09:45
18	What topic is discussed between 23:00 and 26:00?	Between 23:00 and 26:00, an example on the arithmetic series is discussed.
19	When did the lecturer mention the overview of the lecture?	The lecturer mentioned the overview of the lecture 57:00 onwards.
20	What topics were discussed between the 2 to 8 minutes of the lecture?	The lecturer discussed the course schedule and the learning outcomes of the course.

Appendix B

Single Docs Question Full Dataset

No.	Question	Ground Truth
1	According to the lecturer in Sc1007_videolecture, is the lab graded, and how is attendance handled?	The lab itself is not graded, but attendance is still taken. Skipping labs may hurt performance in lab tests. (Reference: Sc1007_videolecture)
2	In Sc1007_videolecture, What penalty does the lecturer state for plagiarism in assignments?	Assignments are checked for plagiarism; getting caught results in zero marks for that assignment. (Reference: Sc1007_videolecture)
3	List the three essential characteristics of an algorithm mentioned by the lecturer in Sc1007_videolecture.	Correctness, precision (unambiguous steps), and finiteness (must terminate in finite time). (Reference: Sc1007_videolecture)

4	When is the combined lab test and quiz scheduled, and what does the quiz cover mentioned in Sc1007_videolecture.	Week 14 (April 20): a combined two-hour lab test and quiz. The MCQ tests concepts from Weeks 1–13, including analysis of algorithms. (Reference: Sc1007_videolecture)
5	In the factorial example with a while loop mentioned in Lecture2_Sc1007, how does computational time scale with N?	It scales linearly with N; time complexity is linear ($O(N)$). (Reference: Lecture2_Sc1007)
6	In Lecture2_Sc1007, for a nested loop that sums all elements in an $N \times N$ matrix, what is the time complexity?	Quadratic time, $O(N^2)$; more generally $M \times N$ operations for an $M \times N$ matrix. (Reference: Lecture2_Sc1007)
7	What cases should be considered when analyzing selection structures (if/else) for time complexity mentioned in Lecture2_Sc1007?	Best case, worst case, and average case (with probabilities). (Reference: Lecture2_Sc1007)
8	In Lecture2_Sc1007, what is the recurrence for counting comparisons in a simple recursive array scan, and what does it solve to?	$W(N) = 1 + W(N-1)$ with base $W(1)=1$; it solves to linear growth, i.e., N total comparisons. (Reference: Lecture2_Sc1007)
9	In a binary-tree traversal that makes two recursive calls per node, how does the number of operations grow with the number of levels K , mentioned in Lecture2_Sc1007?	It follows a geometric series $(1 + 2 + 4 + \dots + 2^{(K-1)})$, i.e., exponential growth in K . (Reference: Lecture2_Sc1007)
10	In Lecture3_Sc1007, what topics does the lecturer plan to cover after analysis of algorithms?	Sequential search, binary search, and hash tables found in (mentioned in 0:03:12).
11	What does the lecturer emphasize as more important than exact running time when analyzing algorithms in Lecture3_Sc1007?	The order of growth (number of operations). (mentioned in 0:06:27).
12	what example mentioned in Lecture3_Sc1007, is used to explain why factorial time complexity is impractical?	An algorithm with N , operations grows exponentially fast, making it infeasible. (mentioned in 0:07:07).
13	In Lecture3_Sc1007, why are constants ignored in asymptotic analysis?	Because for large input sizes, constants (like +100) become negligible compared to growth rate. (mentioned in 0:08:44).
14	What are the three asymptotic notations introduced by the lecturer in Lecture3_Sc1007?	Big O, Big Omega, and Big Theta. (mentioned in 0:11:31).

15	What is the best-case time complexity of sequential search in a linked list in Lecture3_Sc1007?	Constant time, Big Theta(1). (mentioned in 0:40:07).
16	In Lecture3_Sc1007, what is the worst-case time complexity of sequential search?	Linear time, Big Theta(N). (mentioned in 0:42:03).
17	In Lecture3_Sc1007, what is the average-case complexity of sequential search?	Linear time, Big Theta(N). (mentioned in 0:46:32).
18	What improvement does the lecturer suggest over binary search for faster lookups in Lecture4_Sc1007?	Using a hashtable to achieve constant time search. (mentioned in 0:08:15).
19	In Lecture4_Sc1007, what trade-off is mentioned when using hash tables?	Sacrificing space to achieve faster (constant time) search. (mentioned in 0:09:00).
20	In Lecture4_Sc1007, what is hashing defined as?	Mapping data of arbitrary size to a fixed size array using a hash function. (mentioned in 0:13:20).
21	What problem arises when two values map to the same slot in a hash table discussed in In Lecture4_Sc1007?	A collision. (mentioned in 0:14:30).
22	In Lecture4_Sc1007, why is it recommended to make the table size a prime number?	To avoid clustering when using modular arithmetic. (mentioned in 0:27:13).
23	Mentioned in Lecture4_Sc1007, what collision resolution technique stores multiple keys in a linked list at the same slot?	Closed addressing (separate chaining). (mentioned in 0:31:01).
24	In Lecture4_Sc1007, what is linear probing in open addressing?	Checking the next slot sequentially when a collision occurs. (mentioned in 0:52:01).
25	What is secondary clustering in quadratic probing talked about In Lecture4_Sc1007?	Keys with the same initial hash follow the same probe sequence, causing clustering. (mentioned in 0:58:25).
26	In Lecture4_Sc1007, why is double hashing preferred over quadratic probing?	It varies the probe sequence per key, reducing clustering. (mentioned in 1:02:15).
27	In Lecture5_Sc1007, what topics are covered in this lecture?	Graph Representation, specifically Adjacency Matrix and Adjacency List (mentioned in 0:09:48).

28	In Lecture5_Sc1007, when will the lecturer release the assignment?	The Assignment would be released this Wednesday or next Monday, and students should start practicing on assignment 4 early. The assignment link is only valid for two weeks. (mentioned in 0:10:00).
29	In Lecture5_Sc1007, what two components make up a graph?	Vertices and edges. (mentioned in 0:10:03).
30	In Lecture5_Sc1007, what makes a tree a special kind of graph?	It has no cycles. (mentioned in 0:15:00).

Appendix C

Multi Docs Question Full Dataset

No.	Question	Ground Truth
1	Across the two lectures Sc1007_Videolecture and Lecture2_videolecture, which topics are planned after the analysis of algorithms and when might hash tables be covered?	Both lectures mention moving on to hash tables and graph problems. Lecture2_Sc1007 notes hash tables will be covered on Monday (or Wednesday) depending on time. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
2	"Which lecture between Sc1007_Videolecture and Lecture2_videolecture, previews asymptotic notation and which one defines Big O, Big Omega, and Big Theta in detail?"	Sc1007_videolecture previews that asymptotic notation (e.g., Big O) will be covered; Lecture2_Sc1007 defines Big O, Big Omega (Omega), and Big Theta (Theta) and explains ignoring constants and focusing on growth order. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
3	Summarize how recursion impacts time and space across the Sc1007_Videolecture and Lecture2_videolecture lectures using examples given.	Sc1007_videolecture uses Fibonacci to show naive recursion can be exponential in time; Lecture2_Sc1007 shows factorial recursion is linear time but uses extra stack memory compared to iteration. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
4	From both lectures Sc1007_Videolecture and Lecture2_videolecture, what is the difference between an algorithm and a program, and how is efficiency evaluated?	Sc1007_videolecture explains an algorithm is a well-defined procedure while a program is its implementation; Lecture2_Sc1007 explains efficiency is evaluated via time/space complexity and asymptotic growth, not raw runtime constants. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
5	Combine the examples: which method is most efficient for summing 1..N, and how is this justified by the complexity	Using the arithmetic series formula $N/2 * (1+N)$ is most efficient (constant-time operations), justified by Lecture2_Sc1007's

	principles in both Sc1007_Videolecture and Lecture2_videolecture lectures.	principle of focusing on lowest order of growth and ignoring constants. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
6	What searching approaches are mentioned in Sc1007_Videolecture and Lecture2_videolecture, and how do their time complexities differ?	Linear search (Sc1007_videolecture) is $O(N)$; hash-based lookup (Lectures 1 & 2 plan) targets $O(1)$ average time with extra space. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
7	Considering the course logistics from Sc1007_Videolecture and Lecture2_videolecture lectures, what assessments contribute to the final grade?	Assignments (two parts totaling 40%), two lab tests (20percent respectively), and a final quiz (20%), as outlined in Sc1007_videolecture; Lecture2_Sc1007 reiterates upcoming topics tied to assessments release (e.g., after hash tables). (Reference: Sc1007_videolecture + Lecture2_Sc1007)
8	Using both lectures Sc1007_Videolecture and Lecture2_videolecture, explain why constants are ignored in asymptotic analysis and give an example where $N^2 + 100$ and N^2 are treated the same.	Lecture2_Sc1007 explicitly shows that as N grows large, additive constants (like $+100$) and constant factors have negligible impact on growth, so $N^2 + 100$ and N^2 are both $\Theta(N^2)$. (Reference: Lecture2_Sc1007 (previewed in Sc1007_videolecture))
9	Across the Sc1007_Videolecture and Lecture2_videolecture videos, which graph-related topics are planned and what real-world path problem is used as an example?	Planned topics include BFS, DFS, backtracking, permutations, dynamic programming, and matching; Google Maps shortest path is given as a real-world example (Sc1007_videolecture). (Reference: Sc1007_videolecture + Lecture2_Sc1007)
10	What guidance is given about coding vs. concepts across the Sc1007_Videolecture and Lecture2_videolecture videos, and how should students prepare?	Lectures focus on concepts; implementation practice is expected in labs/assignments. Students should practice coding themselves rather than relying on solutions. (Reference: Sc1007_videolecture + Lecture2_Sc1007)
11	Which graph-related problems are planned in Sc1007_Videolecture, and what real-world application is used in Lecture5_videolecture?	In Sc1007_videolecture, the lecturer planned to cover graph-related problems such as BFS, DFS, backtracking, and dynamic programming (mentioned in 0:04:26), while in Lecture5_videolecture a real-world MRT/Google Maps shortest path example was discussed (mentioned in 0:27:01).
12	Which lecture between Sc1007_Videolecture and Lecture3_videolecture previews asymptotic notation, and which defines Big O, Big Omega, and Big Theta in detail?	In Sc1007_videolecture, the lecturer previewed asymptotic notation (mentioned in 0:26:40), while in Lecture3_videolecture the lecturer defined Big O, Big Omega, and Big Theta formally (mentioned in 0:11:31).
13	How is search complexity contrasted in Lecture3_videolecture sequential search and Lecture4_videolecture binary search?	In Lecture3_videolecture, the lecturer explained that sequential search takes $\Theta(N)$ in the worst case (mentioned in 0:42:03),

		whereas in Lecture4_videolecture binary search was shown to achieve $\Theta(\log N)$ (mentioned in 0:06:06).
14	What assessments are mentioned in Sc1007_Videolecture and Lecture2_videolecture videos and how do they count toward the final grade?	In Sc1007_videolecture, the lecturer explained that assignments contribute 40%, lab tests contribute 20percent respectively, and the quiz contributes 20% (mentioned in 0:07:54), while in Lecture2_videolecture the lecturer also discussed lab test and quiz details (mentioned in 0:03:45).
15	Why are constants ignored in asymptotic analysis, and what example is provided across the Sc1007_Videolecture and Lecture3_videolecture videos?	In Sc1007_videolecture, the lecturer explained that constants are negligible for large inputs (mentioned in 0:26:26), and in Lecture3_videolecture the lecturer gave the example that $N^2 + 100$ is treated the same as N^2 (mentioned in 0:08:44).
16	How are linked lists used differently in Lecture4_videolecture hash tables and Lecture5_videolecture adjacency lists?	In Lecture4_videolecture, the lecturer explained that closed addressing uses linked lists to handle collisions (mentioned in 0:31:01), whereas in Lecture5_videolecture adjacency lists use linked lists to store neighboring vertices (mentioned in 0:37:14).
17	How is efficiency of summing $1..N$ compared in both Sc1007_Videolecture and Lecture3_videolecture ?	In Sc1007_videolecture, the lecturer explained that using the formula $N(N+1)/2$ provides constant time efficiency (mentioned in 0:24:34), while in Lecture3_videolecture growth rate analysis also showed that the formula is the most efficient method (mentioned in 0:06:45).
18	What graph representations are mentioned across Lecture3_videolecture and Lecture5_videolecture?	In Lecture3_videolecture, the lecturer introduced graphs conceptually (mentioned in 0:33:38), and in Lecture5_videolecture the lecturer explained adjacency matrices and adjacency lists in detail (mentioned in 0:30:01).
19	What real-world applications of graphs are given across Sc1007_videolecture and Lecture5_videolecture?	In Sc1007_videolecture, the lecturer gave examples such as the traveling salesman problem and cryptography (mentioned in 0:37:32), while in Lecture5_videolecture examples included MRT shortest path problems and computer networks (mentioned in 0:27:01).
20	What is the average-case complexity of search mentioned in Lecture3_videolecture and Lecture4_videolecture?	In Lecture3_videolecture, the lecturer explained that sequential search has an average-case complexity of $\Theta(N)$ (mentioned in 0:46:32), while in Lecture4_videolecture the lecturer explained that hash table search averages $\Theta(1)$ when the load factor is constant (mentioned in 0:46:41).

21	What guidance is given about tutorials, labs, and coding across Sc1007_videolecture and Lecture2_videolecture ?	In Sc1007_videolecture, the lecturer explained that tutorials focus on concepts while labs focus on practice (mentioned in 0:08:50), and in Lecture2_videolecture the lecturer explained that complexity analysis is taught in lecture but coding is applied in labs (mentioned in 0:03:28).
22	How is graph connectivity defined in Lecture5_videolecture compared to complexity bounds in Lecture3_videolecture ?	In Lecture5_videolecture, the lecturer explained that a graph is connected if a path exists between any two vertices (mentioned in 0:21:53), while in Lecture3_videolecture the lecturer explained how algorithms are bounded using Big O, Big Omega, and Big Theta (mentioned in 0:11:31).
23	How do adjacency lists relate to hash tables as explained across Lecture4_videolecture and Lecture5_videolecture?	In Lecture4_videolecture, Closed addressing uses linked lists (mentioned in 0:31:01), whereas in Lecture5_videolecture, Adjacency list uses linked lists to store neighbors (mentioned in 0:37:14).
24	What are the main considerations when analyzing if/else in Lecture2_videolecture versus collisions in Lecture4_videolecture?	In Lecture2_videolecture, it mentions the best and worst cases in if/else (mentioned in 0:18:20), while in Lecture4_videolecture, it mentioned collisions and clustering (mentioned in 0:27:13).
25	Which parts of the module are described as concept-heavy, and how do labs balance this across Sc1007_videolecture and Lecture2_videolecture ?	In Sc1007_videolecture, he mentioned tutorials focus on concepts, labs on coding practice (mentioned in 0:08:50). In Lecture2_videolecture, he mentioned complexity analysis is theory, coding tested in labs (mentioned in 0:03:28).
26	How are nested loops explained in Lecture2_videolecture and Lecture3_videolecture, and what time complexity do they lead to?	In Lecture2_videolecture, summing NxN matrix gives $O(N^2)$ (mentioned in 0:16:02). In Lecture3_videolecture, general polynomial complexities like $O(N^2)$ (mentioned in 0:07:00).
27	What is the difference between an algorithm and a program, and how is efficiency evaluated across Sc1007_Videolecture and Lecture2_videolecture ?	In Sc1007_videolecture, the lecturer explained that an algorithm is a procedure while a program is its implementation (mentioned in 0:21:07), and in Lecture2_videolecture efficiency was evaluated using time and space complexity (mentioned in 0:07:20).
28	In Lecture2_videolecture and Lecture4_videolecture, what considerations are made for selection structures and hash table collisions?	In Lecture2_videolecture, the lecturer emphasized analyzing best and worst cases for if/else statements (mentioned in 0:18:20), while in Lecture4_videolecture the lecturer explained how collisions and clustering issues affect hash tables (mentioned in 0:27:13).
29	How does load factor in Lecture4_videolecture relate to loop growth in Lecture2_videolecture ?	In Lecture4_videolecture, the lecturer explained that the load factor affects the efficiency of hash tables (mentioned in 0:20:30),

		while in Lecture2_videolecture the lecturer described how loop complexity grows with the input size N (mentioned in 0:06:27).
30	What distinguishes trees from general graphs across the lectures Lecture2_videolecture and Lecture5_videolecture?	In Lecture2_videolecture, the lecturer explained recursive binary tree traversal (mentioned in 0:20:01), while in Lecture5_videolecture the lecturer emphasized that trees are acyclic graphs (mentioned in 0:15:00).

Appendix D

Baseline RAG - RAGAS Evaluation Scores for Temporal with Output Response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
1	What is mentioned at 33 minutes of the lecture?	0	0	0.666667	0
		I am unable to find an answer for the specific content mentioned at 33 minutes of the lecture. The provided transcript does not include any information from that timestamp. If you have any other questions or need information from a different part of the lecture, feel free to ask!			
2	What was discussed in 27:00 of the lecture?	0	0	0.333333	0
		I am unable to find an answer for the specific content discussed at 27:00 of the lecture. If you have any other questions or need information from a different part of the lecture, feel free to ask!			
3	When was the difference between algorithm and program discussed?	1	1	0.857143	0.929078
		<p>The difference between an algorithm and a program was discussed at several points in the video. Specifically, it was covered in detail at the following timestamps:</p> <ul style="list-style-type: none"> - At 21:07.52, the speaker states, "Generally speaking, they are not the same thing," while introducing the topic. - Further elaboration is provided at 21:20.16, where the speaker explains that an algorithm is a procedure or idea to solve a problem, while a program is the implementation of that algorithm in a programming language. <p>These timestamps highlight the key distinctions made between algorithms and programs in the lecture.</p>			
4	What concept is explained around 45:00 into the lecture?	0	0	0.75	0.923041
		Around 45:00 into the lecture, the instructor discusses the analysis of algorithms, specifically focusing on time complexity and space complexity. He mentions that he will cover best case, worst case, average efficiency, and the importance of understanding the order of growth of an algorithm, including how to represent these concepts using big O notation. This is covered at "Covered at 57:20".			

5	At what point in the lecture does it start discussing How do we solve the different searching, graph and combinatorial problems?	0.2	1	0.5	0.959778
		The lecture begins discussing how to solve different searching, graph, and combinatorial problems around the timestamp of **33:19** . At this point, the speaker mentions that there are many kinds of problems in the module, including searching, graph problems, and combinatorial problems, which encompass permutation and combination issues. This sets the stage for further exploration of these topics. Covered at **[33:19]** .			
6	What does the lecturer say right after the algorithm Vs program at 23 minutes in the video?	0	0	0.5	0.923193
		Right after discussing the difference between an algorithm and a program at 23 minutes, the lecturer emphasizes the importance of practicing the implementation of algorithms. He states that as you progress to higher levels, algorithms become harder to implement, and if you don't practice today, you are unlikely to realize those algorithms in the future. This is covered at [23:09].			
7	Does the lecturer explain graph problems before or after Combinatorial problems ?	0.805556	1	1	0.936003
		The lecturer explains graph problems after discussing combinatorial problems. Specifically, combinatorial problems are mentioned first when the lecturer talks about various types of problems, including searching and graph problems, and then transitions into graph problems later in the lecture. This is covered at "33:19.44" and "56:30.88". Before the Learning Outcomes, the discussion focused on the importance of understanding and practicing programming concepts. It emphasized that after learning the concepts, students need to practice in labs, at home, or during lab sessions to develop their programming skills. The speaker highlighted that programming skills can only be developed through practice, stating, "No matter how I teach you all the possible techniques in the lecture, if you do not practice, you will not learn anything" (Covered at 09:30). Additionally, the speaker mentioned that pointers are crucial in C programming, followed by linked lists, stacks, queues, and other data structures (Covered at 14:22).			
8	What was discussed before Learning outcomes?	0.7	1	0.583333	0.965422
		Before the Learning Outcomes, the discussion focused on the importance of understanding and practicing programming concepts. It emphasized that after learning the concepts, students need to practice in labs, at home, or during lab sessions to develop their programming skills. The speaker highlighted that programming skills can only be developed through practice, stating, "No matter how I teach you all the possible techniques in the lecture, if you do not practice, you will not learn anything" (Covered at 09:30). Additionally, the speaker mentioned that pointers are crucial in C programming, followed by linked lists, stacks, queues, and other data structures (Covered at 14:22).			
9		0.2	0	0.9	0.999999

	What topic is discussed right after Algorithm Design Strategies?	The topic discussed right after Algorithm Design Strategies is the analysis of algorithms. This includes an introduction to time complexity and space complexity, as well as discussions on best case, worst case, average efficiency, and the importance of understanding the order of growth of an algorithm, including how to represent them using big O notation. This is mentioned towards the end of the lecture, specifically at the timestamp "Covered at 57:20".			
10	Which topic comes just before the explanation of the Sorting Problem?	1	1	1	0.9871
		The topic that comes just before the explanation of the Sorting Problem is the Minimum Vertex Cover Problem. This is discussed at the timestamp (42:57.56), where it is mentioned that they need to find the minimum number of checkpoints to allocate at particular junctions in a traffic network. The sorting problem is introduced shortly after that, starting at (43:24.36). So, the transition from the Minimum Vertex Cover Problem to the Sorting Problem occurs between these timestamps.			
11	Which week or lecture covers the topic Hash Tables?	0.8875	0	0.833333	0.935736
		The topic of Hash Tables is covered in the second half of the module, specifically mentioned after discussing linked lists and binary search trees. The relevant content indicates that hash tables will be introduced following the discussion on searching algorithms and before graph problems. This is covered at "03:37.48" and "04:05.08". If you need more specific details about the week or assignments related to Hash Tables, please let me know!			
12	During which part of the lecture (start/middle/end) is Computer Science Programme Structure discussed?	0.916667	0	0.4	0.945464
		The discussion about the Computer Science Programme Structure occurs primarily in the middle part of the lecture. Specifically, it is mentioned around the timestamps of **11:40** to **12:20**, where the speaker talks about the software development components and prerequisites for the courses. This section provides insights into the structure and expectations of the program. Covered at **11:40** to **12:20**.			
13	What is covered in the last 5 minutes of the lecture?	0.25	1	0.769231	0.965178
		In the last 5 minutes of the lecture, the speaker discusses the representation of time complexity and space complexity using asymptotic notation, indicating that this topic will be covered in the next lecture. They also address a question about solving recursive relations and optimizing time complexity, mentioning that it involves a trade-off where more space is used to store results, which can improve efficiency. Additionally, they emphasize the importance of practicing coding to truly learn the material, stating that simply reading solutions will not be beneficial for understanding how to implement algorithms. This part of the lecture concludes with a reminder to practice independently in labs and assignments. This content is covered at the following timestamps:			

		- Time complexity and space complexity representation: "Covered at 58:04" - Discussion on recursive relations and optimization: "Covered at 58:24" - Importance of practice: "Covered at 01:00:07" and "Covered at 09:30"			
14	What was discussed at the start of the lecture?	1	0	0.916667	0.93949
		At the start of the lecture, the instructor introduced the topic of algorithms and the analysis of algorithms. They mentioned that the lecture would be conducted through a live stream and that there would be a slight delay in communication. The instructor also highlighted that the lecture notes were already uploaded and encouraged students to refer to them if anything was missed during the lecture. Additionally, they mentioned that questions could be posted in the chat, but responses might be delayed due to the communication lag. This introduction sets the stage for the lecture content that follows. This information is covered at the following timestamps: - Introduction to the lecture topic: "Covered at 00:47" - Mention of lecture notes and communication delay: "Covered at 01:30" and "Covered at 01:56".			
15	What was discussed at the end of the lecture?	0.2	1	0.9	0.821998
		At the end of the lecture, the speaker mentioned that the following topics would be covered in the upcoming weeks, specifically highlighting that on Wednesday, they would discuss the analysis of algorithms. This includes introducing time complexity and space complexity, as well as discussing best case, worst case, average efficiency, and the importance of understanding the order of growth of an algorithm, including how to represent these concepts using big O notation. This information is "Covered at [57:14]".			
16	What was said 2 minutes before Problem Type was introduced?	0	0	0.8	0.952878
		Two minutes before the introduction of Problem Type, the discussion focused on the importance of understanding algorithms and their efficiency. Specifically, it was mentioned that while it's crucial to know how to implement and realize an algorithm to solve a problem, the expectation at the next level is to select the most efficient algorithm. This was covered at the timestamps of 13:18.40 to 13:39.32, where the speaker emphasized the need to focus on implementation first before selecting the best solution.			
17	When did the lecturer mention the learning outcomes of this course?	0	0	0.6	0.920453
		The lecturer mentioned the learning outcomes of the course at the beginning of the video. Specifically, the learning outcomes include selecting appropriate data structures, implementing algorithms to solve real-world problems using C programming, and conducting complexity analysis of algorithms. This information is covered at the timestamp "00:10.56".			
18		0	0	0.5	0

	What topic is discussed between 23:00 and 26:00?	I am unable to find an answer for the specific topic discussed between 23:00 and 26:00 in the provided context. If you have any other questions or need information on a different part of the video, feel free to ask!			
19	When did the lecturer mention the overview of the lecture?	0.5	0	1	0.896504
		The lecturer provided an overview of the lecture during the introduction, specifically mentioning the focus on algorithms and their analysis. This was covered at "00:47.25" when the lecturer stated, "So today I'll give an introduction to the algorithm and analysis of algorithms through this channel." Additionally, the overview of the course structure and topics was discussed around "15:03.28," where the lecturer mentioned spending time on time complexity and space complexity, along with introducing algorithms and their strategies.			
20	What topics were discussed between the 2 to 8 minutes of the lecture?	0.333333	1	0.666667	0.978055
		<p>Between the 2 to 8 minutes of the lecture, the instructor discussed the following topics:</p> <ol style="list-style-type: none"> 1. The structure of the lecture, mentioning that it would cover an introduction to algorithms and analysis of algorithms. 2. The format of the lecture, indicating that it is a live stream and that the content may vary slightly each time it is presented. 3. The availability of lecture notes and slides for reference, which will be uploaded in the coming week. 4. The instructor encouraged students to post questions in the chat, although they may not be answered immediately due to a delay in communication. <p>This information is primarily covered at the following timestamps:</p> <ul style="list-style-type: none"> - Introduction to the lecture and its structure: Covered at 00:47.25 to 01:12.60 - Availability of notes and slides: Covered at 01:30.20 to 01:46.32 - Encouragement to ask questions: Covered at 01:56.92 to 02:24.76. 			

Appendix E

Temporal RAG - RAGAS Evaluation Scores for Temporal with Output Response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
1	What is mentioned at 33 minutes of the lecture?	0.5	1	1	0.959078
		At 33 minutes of the lecture, the speaker discusses the various types of problems covered in the module, including searching, graph problems, and combinatorial problems, which involve permutations and combinations. They also mention that in the next level, topics such as sorting strings, processing, and geometric and numerical problems will be introduced. This section emphasizes the breadth of topics that will be explored in the course.			

		This information is "Covered at 33:19."			
2	What was discussed in 27:00 of the lecture?	0.5	1	0.8	0.900145
		<p>In the lecture around 27:00, the speaker discusses the Fibonacci sequence, explaining how it is generated. The sequence starts with 1, 1, and each subsequent number is the sum of the two preceding numbers. For example, the next numbers are calculated as follows: $1 + 1 = 2$, then $1 + 2 = 3$, followed by $2 + 3 = 5$, and so on. The speaker also mentions the goal of finding the Nth term of the Fibonacci sequence and introduces the concept of using a recursive function to achieve this. This recursive approach involves calling the function for $N - 1$ and $N - 2$ until reaching the base cases of N equal to 1 or 2, where the function returns 1. This part of the discussion emphasizes the recursive nature of the Fibonacci sequence calculation.</p>			
		This information is covered at [27:03].			
3	When was the difference between algorithm and program discussed?	1	0	0.75	0.9359
		<p>The difference between an algorithm and a program was discussed at several points in the video. Specifically, it was covered in detail at the following timestamps:</p> <ul style="list-style-type: none"> - At 21:07.52, the speaker states, "Generally speaking, they are not the same thing," and elaborates on the distinction between an algorithm and a program. - Further clarification is provided at 21:20.16, where the speaker explains that an algorithm is a procedure or idea to solve a problem, while a program is the implementation of that algorithm in a programming language. 			
		These timestamps highlight the key differences between the two concepts.			
4	What concept is explained around 45:00 into the lecture?	1	1	0.857143	0.874386
		<p>Around 45:00 into the lecture, the concept of stability in sorting algorithms is explained. The speaker discusses how stable sorting algorithms maintain the relative order of records with equal keys. For example, if two students receive the same score, a stable sorting algorithm will keep their original order intact when sorting. This is illustrated with an example involving students' scores across different modules, where the order of students with the same score remains unchanged after sorting. This ensures that the results of previous sorts are preserved. This explanation is covered at "Covered at 45:08".</p>			
5	At what point in the lecture does it start discussing How do we solve the	0.583333	0	1	0.819036
		<p>The lecture begins discussing how to solve different searching, graph, and combinatorial problems around the timestamp of 33:19. At this point, the speaker mentions that there are many kinds of</p>			

	different searching, graph and combinatorial problems?	problems in the module, including searching, graph problems, and combinatorial problems, which encompass permutation and combination problems. This sets the stage for further exploration of these topics. Covered at [33:19] .			
6	What does the lecturer say right after the algorithm Vs program at 23 minutes in the video?	0.638889	1	0.666667	0.925883
		Right after discussing the difference between an algorithm and a program at 23 minutes, the lecturer explains that the next step after understanding a problem is to sort it out with an algorithm, which is a well-defined step-by-step procedure. He then mentions that you implement this algorithm in a programming language like C, Java, or Python to write a program that takes input and produces output. This is highlighted in the transcript at "Covered at 21:20".			
7	Does the lecturer explain graph problems before or after Combinatorial problems ?	1	0	1	0.894661
		The lecturer explains graph problems after discussing combinatorial problems. Specifically, the mention of combinatorial problems occurs around 33:19 and the focus on searching and graph problems is introduced later at 56:30 . Thus, combinatorial problems are covered first, followed by graph problems. Covered at 33:19 and 56:30 .			
8	What was discussed before Learning outcomes?	0	0	0.857143	0.930412
		Before the Learning Outcomes, the discussion focused on the importance of understanding and practicing programming concepts. It emphasized that after learning the concepts, students need to practice in the lab, at home, or during lab sessions to develop their programming skills. The speaker highlighted that programming skills can only be developed through practice, stating, "No matter how I teach you all the possible techniques in the lecture, if you do not practice, you will not learn anything" (Covered at 09:30). Additionally, the lecture covered the significance of pointers and various data structures, including linked lists, stacks, and queues, as essential components of C programming (Covered at 14:22).			
9	What topic is discussed right after Algorithm Design Strategies?	0.25	1	0.857143	0.995246
		The topic discussed right after Algorithm Design Strategies is the analysis of algorithms. This includes an introduction to time complexity and space complexity, as well as discussions on best case, worst case, average efficiency, and the importance of understanding the order of growth of an algorithm, including how to represent them using big O notation. This is mentioned towards the end of the lecture, specifically at the timestamp "Covered at 57:20".			
10		1	0	1	0.9871

	Which topic comes just before the explanation of the Sorting Problem?	<p>The topic that comes just before the explanation of the Sorting Problem is the Minimum Vertex Cover Problem. This is discussed at the timestamp of **42:57.56**, where it is mentioned that they need to find the minimum number of checkpoints to allocate at particular junctions for checking cars in a traffic network. The sorting problem is introduced shortly after that, starting at **43:24.36**.</p> <p>So, the transition is from the Minimum Vertex Cover Problem to the Sorting Problem.</p>			
11	Which week or lecture covers the topic Hash Tables?	1	0	1	0.935773
		<p>The topic of Hash Tables is covered in the second half of the module, specifically mentioned after discussing linked lists and binary search trees. The relevant content about hash tables is introduced around the timestamp of **03:37.48**, where it states, "The second half of this module will cover hash tables and graph problems." Additionally, hash tables are further discussed in the context of lab sessions, particularly in Lab 5, which is scheduled after the introduction of hash tables.</p> <p>For more details, you can refer to the content covered at **03:37.48**.</p>			
12	During which part of the lecture (start/middle/end) is Computer Science Programme Structure discussed?	0.8875	1	0.8	0.942464
		<p>The Computer Science Programme Structure is discussed in the middle part of the lecture. Specifically, it is mentioned around the timestamps of **11:40** to **12:20**, where the focus is on software development and the prerequisites for the courses. This section outlines the theoretical and practical components of the program, indicating the structure and expectations for students. Covered at **[11:40]**.</p>			
13	What is covered in the last 5 minutes of the lecture?	0	0	0.75	0
		<p>I'm unable to find an answer regarding the content covered in the last 5 minutes of the lecture, as the provided transcript does not include that specific information. If you have any other questions or need further clarification, feel free to ask!</p>			
14	What was discussed at the start of the lecture?	0.804167	0.666667	1	0.93949
		<p>At the start of the lecture, the instructor introduced the topic of algorithms and the analysis of algorithms. They mentioned that the lecture would be conducted through a live stream and that there would be a slight delay in communication. The instructor also noted that they had uploaded lecture notes and encouraged students to refer to them if anything was missed during the lecture. Additionally, they invited students to post questions in the chat, although they might not be able to respond immediately due to the delay. This introduction sets the stage for the lecture content that follows.</p> <p>This information is covered at the following timestamps:</p>			

		- Introduction to the lecture topic: "Covered at 00:47" - Mention of lecture notes and communication: "Covered at 01:30" and "Covered at 01:56"			
15	What was discussed at the end of the lecture?	0.2	1	0.857143	0.853839
		At the end of the lecture, the speaker mentioned that the following topics would be covered in the upcoming weeks. Specifically, they indicated that on Wednesday, they would discuss the analysis of algorithms, including time complexity and space complexity. They planned to cover concepts such as best case, worst case, average efficiency, and the importance of understanding the order of growth of an algorithm, as well as how to represent these concepts using big O notation. This information is "Covered at [57:14]".			
16	What was said 2 minutes before Problem Type was introduced?	0	0	0.571429	0.788746
		To answer your question, two minutes before the introduction of the Problem Type, the speaker was discussing the structure of the lecture and how it would be conducted. Specifically, they mentioned that they would be giving an introduction to algorithms and the analysis of algorithms, and that the lecture would be a live stream with slight variations each time it is taught. They also noted that lecture notes and slides would be available for reference. This information is covered at the following timestamps: - Introduction to the algorithm and analysis of algorithms: Covered at **00:47** - Mention of the lecture being a live stream and variations: Covered at **01:18** - Availability of lecture notes and slides: Covered at **01:33** If you need more specific details or further clarification, feel free to ask!			
17	When did the lecturer mention the learning outcomes of this course?	0.75	0	0.8	0.93293
		The lecturer mentioned the learning outcomes of the course at the beginning of the video. Specifically, the learning outcomes include selecting appropriate data structures, implementing algorithms to solve real-world problems using C programming, and conducting complexity analysis of algorithms. This information is covered at the timestamp "00:10.56".			
18	What topic is discussed between 23:00 and 26:00?	0	0	0.75	0
		I'm unable to find an answer regarding the specific topic discussed between 23:00 and 26:00 in the provided context. The transcript only covers the introduction and initial parts of the lecture. If you have any other questions or need information from a different section, feel free to ask!			
19		0.5	1	0.833333	0.901867

	When did the lecturer mention the overview of the lecture?	The lecturer provided an overview of the lecture during the introduction, specifically mentioning the focus on algorithms and their analysis. This is covered at "00:47.25" when the lecturer states, "So today I'll give an introduction to the algorithm and analysis of algorithms through this channel." Additionally, the overview of the course structure and topics is discussed around "15:03.28," where the lecturer outlines the important concepts and topics that will be covered, including time complexity and space complexity.			
20		0	0	0.923077	0.906097
	What topics were discussed between the 2 to 8 minutes of the lecture?	<p>During the lecture between the 2 to 8 minutes mark, the speaker primarily discusses the following topics:</p> <ol style="list-style-type: none"> 1. Introduction to the Lecture: The speaker introduces the topic of algorithms and the analysis of algorithms, indicating that this will be the focus of the lecture. This is mentioned at the timestamp of 00:47.25. 2. Lecture Format and Content: The speaker explains that the lecture is a live stream and that each session may cover different parts of the material. This is highlighted at 01:18.04 and 01:20.12. 3. Availability of Lecture Materials: The speaker mentions that lecture notes and slides have already been uploaded and that more materials will be provided in the coming week. This is covered at 01:30.20 and 01:46.32. 4. Interaction with Students: The speaker encourages students to post questions in the chat, although they may not be able to respond immediately due to a delay. This is discussed at 01:56.92 and 02:10.32. 5. Contact Information: The speaker provides their name, email address, and office information for students to reach out with questions, which is mentioned at 02:30.40 and 02:34.96. <p>Overall, the segment focuses on setting the stage for the lecture, explaining the format, and encouraging student interaction.</p>			

Appendix F

Baseline RAG - RAGAS Evaluation Scores for Single Docs queries with Output Response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
1	According to the lecturer in Sc1007_videolecture, is the lab graded, and how is attendance handled?	0.8875	0.6666667	1	0
		According to the lecturer in the Sc1007_videolecture, the lab is not graded. However, attendance is still taken, and while it is not guaranteed that you will do well in the lab test if you choose not to attend, it is ultimately your choice whether to participate or not. The lecturer emphasizes that many people keep asking about the grading of the lab, confirming that it is not graded, but attendance is recorded, and it may impact your performance in the lab test. This information is covered at [08:17.24] and [08:27.80].			
2	In Sc1007_videolecture, What penalty does the lecturer state for plagiarism in assignments?	0.8333333	1	1	0.933432
		In the Sc1007_videolecture, the lecturer states that if a student is caught committing plagiarism, they will receive zero marks for their assignment. This is emphasized to encourage students not to copy their friends' answers directly, as it undermines their learning. This information is covered at *(05:22.72)*.			
3	List the three essential characteristics of an algorithm mentioned by the lecturer in Sc1007_videolecture.	0.8333333	0.5	0.875	0.967018
		<p>The three essential characteristics of an algorithm mentioned by the lecturer in the Sc1007_videolecture are:</p> <ol style="list-style-type: none"> 1. Correctness: The output must be correct and consistent for every given input instance. For example, if the input is 5, the output should consistently be 5 when the same input is provided again. Covered at *(19:14.04)*. 2. Precision: Every step of the algorithm must be well-defined and clear, without any ambiguous terms. For instance, instead of saying "some items," it should specify the exact number, like "N cubed items." Covered at *(17:53.08)*. 3. Finiteness: The algorithm must terminate in a finite number of steps, meaning it cannot run indefinitely. Covered at *(18:39.24)*. <p>These characteristics ensure that an algorithm is effective and reliable in producing the desired output.</p>			
4		1	0.6666667	0.5714286	0.9493112

	When is the combined lab test and quiz scheduled, and what does the quiz cover mentioned in Sc1007_videolecture.	<p>"The combined lab test and quiz for the SC1007 course is scheduled for April 20, which is Week 14 of the module. This test will include multiple-choice questions and programming components, and it will cover concepts from Week 1 to Week 13, particularly focusing on the analysis of algorithms, linked lists, binary trees, hash tables, graphs, dynamic programming, and matching problems.</p> <p>This information is detailed in the video at the following timestamps:</p> <ul style="list-style-type: none"> - The date of the lab test and quiz is mentioned at ""Covered at 05:38"". - The topics covered in the quiz are discussed at ""Covered at 06:25""." 			
5	In the factorial example with a while loop mentioned in Lecture2_Sc1007, how does computational time scale with N?	0.75	1	1	0.9530181
		<p>In the factorial example discussed in Lecture2_Sc1007, the computational time scales linearly with the problem size N. Specifically, the algorithm requires N iterations in the while loop, which means that if you increase your problem size by 10 times, the computational time will also roughly increase by about 10 times. This is because the time complexity is a linear function of N, as stated at "Covered at 15:28.92".</p> <p>Additionally, the constants involved (C0 and C1) are considered negligible when N becomes very large, such as in the case of increasing N to a thousand or a million. The focus is primarily on the linear growth of the time complexity with respect to N, which is emphasized at "Covered at 15:43.56".</p>			
6	In Lecture2_Sc1007, for a nested loop that sums all elements in an N×N matrix, what is the time complexity?	1	1	1	0.961835
		<p>In Lecture2_Sc1007, the time complexity for a nested loop that sums all elements in an NxN matrix is described as quadratic. Specifically, when you have two for loops (one for rows and one for columns), the time complexity is $O(N^2)$. This is because for each of the N rows, you perform N operations to sum the elements, resulting in a total of $N * N$ operations, which simplifies to N^2.</p> <p>This concept is covered at the following timestamps:</p> <ul style="list-style-type: none"> - "The function increases quadratically with n if $m == n$" is mentioned at 17:16.56. - The explanation of how the computational time increases with the problem size is elaborated at 18:00.44. <p>Thus, the overall time complexity for summing all elements in an NxN matrix is $O(N^2)$.</p>			
7		0.8875	1	0.9090909	0.9287461

	What cases should be considered when analyzing selection structures (if/else) for time complexity mentioned in Lecture2_Sc1007?	<p>When analyzing selection structures (if/else statements) for time complexity, the following cases should be considered:</p> <ol style="list-style-type: none"> 1. Best-case analysis: This is the scenario where the algorithm performs the least number of operations. For example, if the condition in the if statement is true, only one operation is executed. 2. Worst-case analysis: This represents the scenario where the algorithm performs the maximum number of operations. For instance, if the condition is false and the else block is executed, two operations are performed. 3. Average-case analysis: This involves considering the probabilities of different inputs and their corresponding operations. It takes into account the likelihood of the condition being true or false and calculates the expected number of operations based on these probabilities. <p>These considerations help in understanding how the algorithm behaves under different circumstances, which is crucial for performance evaluation. This information is covered at [22:57].</p>			
8	In Lecture2_Sc1007, what is the recurrence for counting comparisons in a simple recursive array scan, and what does it solve to?	1	1	0.75	0.9628058
		<p>In Lecture2_Sc1007, the recurrence for counting comparisons in a simple recursive array scan is given as $W(N) = 1 + W(N - 1)$. This means that for each recursive call, one comparison is made, and then the function calls itself with a reduced problem size of $N - 1$.</p> <p>The solution to this recurrence can be derived by recognizing that it sums up the number of operations until the base case is reached. Specifically, it results in $W(N) = N$, as you will perform one comparison for each element in the array until you reach the base case when N equals 1, where only one operation is performed.</p> <p>This is summarized in the transcript where it states, "So you repeat the procedure again and again. So it's $1 + 1 + 1 + 1 + 1 + 1$ until the last one" and concludes with "Eventually, your N will be 3, 2, 1" leading to $W(1) = 1$.</p> <p>This explanation is covered at [40:20].</p>			
9		1	1	1	0.9103162

	In a binary-tree traversal that makes two recursive calls per node, how does the number of operations grow with the number of levels K, mentioned in Lecture2_Sc1007?	<p>In a binary-tree traversal that makes two recursive calls per node, the number of operations grows exponentially with the number of levels $\lfloor K \rfloor$. Specifically, the recurrence relation for the number of operations can be expressed as $\lfloor W(K) = 1 + 2W(K - 1) \rfloor$. This leads to a geometric series of operations: $\lfloor 1 + 2 + 4 + 8 + \dots + 2^{\lfloor K - 1 \rfloor} \rfloor$.</p> <p>The total number of operations can be simplified to $\lfloor 2^K - 1 \rfloor$, indicating that as the number of levels $\lfloor K \rfloor$ increases, the number of operations grows exponentially. This exponential growth is significant because it means that if the problem size increases, the number of nodes and thus the operations required will grow rapidly, which can be concerning in terms of efficiency.</p> <p>This concept is covered in the lecture at the timestamp "Covered at 48:19".</p>			
10	In Lecture3_Sc1007, what topics does the lecturer plan to cover after analysis of algorithms?	0	0	1	0.8704736
		<p>In Lecture3_Sc1007, the lecturer plans to cover hash tables after the analysis of algorithms. They mention that once the hash table is covered, there will be tutorial and lab sessions starting from the following week, and the next assignment will also be about the hash table. This is indicated in the transcript where the lecturer states, "Once I cover the hash table, I think next week we will have the tutorial and lab sessions" and "I will also release the assignment after I finish the hash table, because the next assignment, Assignment 4, is about the hash table" (Covered at 04:11.36).</p>			
11	What does the lecturer emphasize as more important than exact running time when analyzing algorithms in Lecture3_Sc1007?	0.5888889	0.5	0.8571429	0.931202
		<p>In Lecture3_Sc1007, the lecturer emphasizes that when analyzing algorithms, the order of growth of the number of operations is more important than the exact running time. This is because the exact time taken can vary based on the specifications of the computer being used, and what truly matters is how the number of operations scales with the size of the input. The lecturer states, "we are more interested in the order of growth, specifically how many operations you need to execute in this algorithm" and highlights that the order is crucial, especially for algorithms with factorial growth, which can lead to significant increases in problem size. This is covered at [06:29].</p>			
12	what example mentioned in Lecture3_Sc1007, is used to explain why factorial time complexity is impractical?	0.9166667	0.5	1	0.9928602
		<p>In Lecture3_Sc1007, the example used to explain why factorial time complexity is impractical is the traveling salesman problem. It is mentioned that this problem has a time complexity of n factorial, which grows very quickly and becomes unmanageable as the problem size increases. This is highlighted when discussing the implications of</p>			

		algorithms with such high growth rates, particularly in comparison to other algorithms. Covered at [01:03:47].			
13	In Lecture3_Sc1007, why are constants ignored in asymptotic analysis?	0.8875	0.5	0.875	0.9572285
		<p>In Lecture3_Sc1007, constants are ignored in asymptotic analysis because the focus is on the growth rate of functions as the input size becomes very large. The idea is that for sufficiently large inputs, the impact of constant factors becomes negligible compared to the growth of the function itself. This is illustrated when discussing Big-Oh notation, where it is stated that if $f(n)$ is bounded above by some constant multiple of $g(n)$ for all large n, then $f(n)$ is said to be in $O(g(n))$ (Covered at [27:20]).</p> <p>Additionally, the lecturer mentions that constants can be any real number, including fractions, as long as they fulfill the criteria of the definition (Covered at [27:26]). This emphasizes that the specific values of constants do not affect the overall classification of the function's growth rate in asymptotic terms.</p>			
14	What are the three asymptotic notations introduced by the lecturer in Lecture3_Sc1007?	0.75	0.5	1	0.9937455
		The three asymptotic notations introduced by the lecturer in Lecture3_Sc1007 are Big O, Big Theta, and Big Omega. This is mentioned at the timestamp "Covered at 11:24".			
15	What is the best-case time complexity of sequential search in a linked list in Lecture3_Sc1007?	0.7555556	0.5	1	0.9653527
		The best-case time complexity of sequential search in a linked list is constant, represented as $\Theta(1)$. This occurs when the search key is the first item in the list, allowing you to find it immediately without needing to check any other items. This is covered at [40:19].			
16	In Lecture3_Sc1007, what is the worst-case time complexity of sequential search?	1	0.5	1	0.9835882
		In Lecture3_Sc1007, the worst-case time complexity of sequential search is described as linear, specifically Big Theta N ($\Theta(N)$). This means that in the worst-case scenario, you may need to check every item in the list before finding the search key or determining that it is not present. This is covered at [42:19].			
17	In Lecture3_Sc1007, what is the average-case complexity of sequential search?	1	0.5	1	0.9860982
		In Lecture3_Sc1007, the average-case complexity of sequential search is concluded to be $\Theta(N)$. This means that, on average, you will need to check a linear number of items in the list to find the search key. This conclusion is drawn from the fact that			

		whether the key is in the list or not, the search process involves checking through the items linearly. This is covered at [50:45].			
18	What improvement does the lecturer suggest over binary search for faster lookups in Lecture4_Sc1007?	0.95	0.5	1	0.9540585
		The lecturer suggests using a hash table as an improvement over binary search for faster lookups. While binary search improves search time to $\Theta(\log_2 N)$, a hash table can achieve constant time search, $O(1)$, by storing data in a specific way that may require more memory. This approach represents a trade-off between space and time, where you sacrifice space to gain better time complexity. This is covered at [08:15] and [08:54].			
19	In Lecture4_Sc1007, what trade-off is mentioned when using hash tables?	0.95	0.5	1	0.987457
		In Lecture4_Sc1007, the trade-off mentioned when using hash tables is between space and time. Specifically, to achieve better time complexity for search operations (from linear to constant time), you may need to sacrifice space by using more memory to create a larger hash table. This is highlighted when the speaker states, "You sacrifice space to achieve better time complexity" and "you need to use more space and create a very large hash table to store the data" (Covered at 09:00 and 01:09:49).			
20	In Lecture4_Sc1007, what is hashing defined as?	1	1	0.8	0.9767562
		In Lecture4_Sc1007, hashing is defined as the process of using a hash function to map data of arbitrary size to a fixed size. This involves creating a mapping that allows for the storage of data in a hash table, which is a fixed-size array or table. The hash function is crucial as it determines how the data is distributed within the hash table. This definition is covered at "Covered at 13:21".			
21	What problem arises when two values map to the same slot in a hash table discussed in In Lecture4_Sc1007?	1	0.5	1	0.8511195
		The problem that arises when two values map to the same slot in a hash table is known as a "collision." When a collision occurs, it means that two different data items are trying to occupy the same slot or index in the hash table. This situation requires a method to resolve the collision, which is an important aspect of managing hash tables. This concept is discussed in the video at multiple points, including "Covered at 14:19" and "Covered at 19:37."			
22	In Lecture4_Sc1007, why is it recommended to make the table size a prime number?	0.95	1	1	0.9145546
		In Lecture4_Sc1007, it is recommended to make the table size a prime number to avoid clustering of data, which can lead to collisions. A prime number should not be too close to any power of 2, as this helps to ensure a more even distribution of keys across the hash table. If the table size is not chosen carefully, all data may map to the			

		same location, resulting in inefficient performance. This is highlighted in the discussion about hash functions and their design, specifically at the timestamps: "Covered at 27:13" and "Covered at 27:48".			
23	Mentioned in Lecture4_Sc1007, what collision resolution technique stores multiple keys in a linked list at the same slot?	0.6388889	0.5	1	0.9598957
		The collision resolution technique that stores multiple keys in a linked list at the same slot is known as **closed addressing hashing** , also referred to as **separate chaining** . In this method, all keys that map to the same location are stored in a linked list, and the head pointer of this linked list is stored in the hash table. When searching for a key, the hash function directs you to the appropriate location, where you can then traverse the linked list to find the key. This technique is covered at [31:40].			
24	In Lecture4_Sc1007, what is linear probing in open addressing?	0.9166667	0.5	1	0.9067774
		In Lecture4_Sc1007, linear probing is a method used in open addressing for handling collisions in hash tables. When a collision occurs (i.e., when two keys hash to the same index), linear probing resolves this by checking the next available slot in a sequential manner.			
		For example, if you want to insert a key and it hashes to an index that is already occupied, you simply add one to the index and check the next slot. If that slot is also occupied, you continue to add one until you find an empty slot. This process is straightforward, as described in the lecture: "If a collision occurs at this location, I try to add one" (Covered at 53:01).			
		However, linear probing can lead to clustering, where multiple keys end up occupying adjacent slots, which can slow down search times (Covered at 55:15).			
25	What is secondary clustering in quadratic probing talked about In Lecture4_Sc1007?	1	0.5	1	0.9644336
		Secondary clustering in quadratic probing refers to a situation where, despite using a non-sequential probing method, the keys are still inserted in a fixed pattern. This occurs because if the keys are hashed to the same initial location, they will follow the same probing sequence to find an alternate slot. As a result, when searching for a key, the search will also follow this fixed pattern, leading to clustering of keys in certain slots. This can make searching inefficient as it creates a predictable pattern of occupied slots, which is problematic in terms of performance.			
		This concept is explained in the video at the timestamp "Covered at 58:09".			

26	In Lecture4_Sc1007, why is double hashing preferred over quadratic probing?	0.3333333	1	0.625	0.9897717
		<p>In Lecture4_Sc1007, double hashing is preferred over quadratic probing because it provides a more effective way to avoid clustering. With double hashing, each key generates a different probe sequence, which reduces the likelihood of forming clusters of occupied slots. This means that the search time does not become linear, making it more efficient compared to quadratic probing, where the probing sequence can still lead to clustering under certain conditions. This advantage of double hashing helps maintain better performance in terms of search time.</p> <p>This information is covered at [01:02:02.72].</p>			
27	In Lecture5_Sc1007, what topics are covered in this lecture?	0	0	1	0.9029132
		<p>In Lecture5_Sc1007, the topics covered include the traversal of graphs, specifically focusing on two methods: breadth-first search and depth-first search. The lecturer mentions that while there are many ways to traverse graphs, only these two methods will be introduced in the lecture. This is highlighted when the lecturer states, "we only introduce the breadth-first search and depth-first search in the next lecture" and elaborates on the nature of these traversals at timestamps (57:07.48) and (57:29.48).</p>			
28	In Lecture5_Sc1007, when will the lecturer release the assignment?	0	0	0	0
		<p>I am unable to find an answer regarding the release date of the assignment in the provided context. If you have any other questions or need further clarification, feel free to ask!</p>			
29	In Lecture5_Sc1007, what two components make up a graph?	0.8041667	0.5	1	0.9476233
		<p>In Lecture5_Sc1007, a graph consists of two components: the vertex and the edge. The vertex represents the nodes of the graph, while the edge represents the links that connect these vertices. This is covered at [10:37.72].</p>			
30	In Lecture5_Sc1007, what makes a tree a special kind of graph?	0.75	1	1	0.9527877
		<p>In Lecture5_Sc1007, a tree is considered a special kind of graph because it is defined as a graph that has no cycles. This means that if any siblings in the tree are connected, it would form a cycle, which disqualifies it from being a tree. Therefore, a tree must maintain the property of being acyclic, making it a unique case within the broader category of graphs. This concept is discussed at the timestamp "15:57.16" where it is emphasized that a tree is a special graph with no cycles.</p>			

Appendix G

Docs Scope RAG - RAGAS Evaluation Scores for Single Docs queries with output response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
1	According to the lecturer in Sc1007_videolecture, is the lab graded, and how is attendance handled?	0.966666667	1	1	0
		In the Sc1007_videolecture, the lecturer states that if a student is caught committing plagiarism, they will receive zero marks for their assignment. This is emphasized to encourage students not to copy their friends' answers directly, as it undermines their learning. This information is covered at 05:22.72 .			
2	In Sc1007_videolecture, What penalty does the lecturer state for plagiarism in assignments?	0.366666667	1	1	0.933588285
		<p>The three essential characteristics of an algorithm mentioned by the lecturer are:</p> <ol style="list-style-type: none"> Correctness: The output must be correct and consistent for every given input instance. For example, if the input is 5, the output should always be 5 when the same input is provided again. Covered at [19:14]. Precision: Every step of the algorithm must be well-defined and systematic, without any ambiguous terms. For instance, instead of saying "some items," it should specify the exact number, like "N cubed items." Covered at [18:23]. Finiteness: The algorithm must terminate in a finite number of instructions and cannot run indefinitely. Covered at [19:00]. <p>These characteristics ensure that an algorithm is effective and reliable in solving problems.</p>			
3	List the three essential characteristics of an algorithm mentioned	1	0.5	0.875	0.967017995
		In the factorial example using a while loop, the computational time scales linearly with the problem size N. Specifically, the time complexity is described as a linear function of N, meaning that if you increase the problem size by 10 times, the computational time will also			

	by the lecturer in Sc1007_videolecture.	<p>roughly increase by about 10 times. This is because the while loop runs N iterations, and the constant time operations (C0 and C1) become negligible as N grows larger.</p> <p>This information is covered at [15:28.92] and [15:43.56].</p>			
4	When is the combined lab test and quiz scheduled, and what does the quiz cover mentioned in Sc1007_videolecture.	1	0.666666667	1	0.938614461
		<p>When analyzing selection structures (if/else statements) for time complexity, the lecture mentions that you should consider three cases: the best-case, worst-case, and average-case scenarios.</p> <ul style="list-style-type: none"> - The best-case scenario is when the condition is met in a way that results in the least number of operations being executed. - The worst-case scenario is when the condition leads to the maximum number of operations being executed. - The average-case scenario takes into account the probabilities of different outcomes and their associated costs in terms of operations. <p>This comprehensive analysis helps in understanding how the algorithm performs under different conditions, which is crucial for real-time systems and resource allocation decisions.</p> <p>This information is covered at [22:57.20] and [20:13.04].</p>			
5	In the factorial example with a while loop mentioned in Lecture2_Sc1007, how does computational time scale with N?	1	1	1	0.953018135
		<p>In a binary-tree traversal that makes two recursive calls per node, the number of operations grows exponentially with the number of levels (K). Specifically, the recurrence relation for the number of operations can be expressed as $W(K) = 1 + 2W(K - 1)$. This leads to a geometric series where the total number of operations can be derived as $1 + 2 + 4 + 8 + \dots + 2^{(K - 1)}$, which sums up to $(2^K - 1)$ operations.</p> <p>As (K) increases, the number of nodes in the tree grows in powers of 2, resulting in an exponential growth in the number of operations. This means that if you increase the level of (K) significantly, the number of operations will grow exponentially, which can be quite substantial.</p> <p>This concept is discussed in the video at various timestamps, including the explanation of the recurrence relation and the geometric series at "Covered at 44:40" and "Covered at 49:19".</p>			
6		0.8875	1	1	0.981701878

	In Lecture2_Sc1007, for a nested loop that sums all elements in an N×N matrix, what is the time complexity?	<p>In Lecture2_Sc1007, the time complexity for a nested loop that sums all elements in an N×N matrix is described as quadratic. Specifically, when you have two for loops (a nested for loop), the time complexity increases to N squared. This is because for each row (N iterations), you perform a summation operation for each column (another N iterations), resulting in a total of N * N operations.</p> <p>This is covered at [16:30.92] and [17:23.60], where it is explained that the function increases quadratically with N when summing elements in a 2D matrix.</p>			
7	What cases should be considered when analyzing selection structures (if/else) for time complexity mentioned in Lecture2_Sc1007?	0.81	1	0.875	0.912835103
		<p>"When analyzing selection structures (if/else statements) for time complexity, you should consider three cases: the best-case, worst-case, and average-case scenarios. This means evaluating the performance of the algorithm under different conditions based on the input values.</p> <p>Specifically, the best-case scenario looks at the minimum time required for the algorithm to complete, the worst-case scenario examines the maximum time, and the average-case scenario considers the expected time over all possible inputs. This comprehensive analysis helps in understanding how the algorithm behaves under various conditions.</p> <p>This information is covered at [22:57]."</p>			
8	In Lecture2_Sc1007, what is the recurrence for counting comparisons in a simple recursive array scan, and what does it solve to?	0.876666667	0.666666667	0.857142857	0.965455169
		<p>In Lecture2_Sc1007, the recurrence for counting comparisons in a simple recursive array scan is given as $W(N) = 1 + W(N - 1)$. This means that for each recursive call, one comparison is made, and then the function calls itself with a reduced problem size of $(N - 1)$ until it reaches the base case.</p> <p>The solution to this recurrence can be derived using backward substitution. It sums up to $W(N) = N$, indicating that the total number of comparisons made is equal to the size of the array (N) when counting occurrences of a specific element. This is covered at [39:54].</p>			
9	In a binary-tree traversal that makes two recursive calls	1	0.5	0.9	0.893399374
		<p>In a binary-tree traversal that makes two recursive calls per node, the number of operations grows exponentially with the number of levels (K). Specifically, the recurrence relation for</p>			

	per node, how does the number of operations grow with the number of levels K, mentioned in Lecture2_Sc1007?	<p>the number of operations can be expressed as $W(K) = 1 + 2W(K - 1)$. This leads to a geometric series where the total number of operations can be derived as $1 + 2 + 4 + 8 + \dots + 2^{(K - 1)}$, which sums up to $(2^K - 1)$ operations.</p> <p>As (K) increases, the number of nodes in the tree grows in powers of 2, resulting in an exponential increase in the number of operations required. This exponential growth is described as $(2^{(K - 1)})$ for the time complexity of the traversal, indicating that if the number of levels increases significantly, the total operations will grow rapidly, which can be quite substantial.</p> <p>This concept is covered at multiple points in the lecture, particularly at timestamps (49:19.64) and (49:49.24).</p>			
10	In Lecture3_Sc1007, what topics does the lecturer plan to cover after analysis of algorithms?	0.416667	0.5	0.6	0.867856
		<p>In Lecture3_Sc1007, the lecturer, Dr. Loke Yuan Ren, plans to cover the hash table after completing the analysis of algorithms. He also mentions that he will try to cover sequential search and binary search if possible during the lecture. This is indicated at the timestamp "Covered at 03:12.29" for the analysis of algorithms and "Covered at 04:11.36" for the hash table.</p>			
11	What does the lecturer emphasize as more important than exact running time when analyzing algorithms in Lecture3_Sc1007?	0.805556	0.5	1	0.931202
		<p>In Lecture3_Sc1007, the lecturer emphasizes that when analyzing algorithms, the order of growth is more important than the exact running time. This is because the exact time taken can vary based on the specifications of the computer being used and the time each primitive operation takes. Instead, the focus should be on how many operations need to be executed in the algorithm, which is crucial for understanding its efficiency. This point is covered at "06:29.88" and "07:07.28".</p>			
12	what example mentioned in Lecture3_Sc1007, is used to explain why factorial time	1	0.5	1	0.994908
		<p>In Lecture3_Sc1007, the example used to explain why factorial time complexity is impractical is the "traveling salesman problem," which is described as having a time complexity of $n \text{ factorial}(n!)$. This example illustrates how quickly the number of operations</p>			

	complexity is impractical?	can grow, making algorithms with factorial time complexity infeasible for larger problem sizes. This is covered at [01:03:47.64].			
13	In Lecture3_Sc1007, why are constants ignored in asymptotic analysis?	1	0.5	0.909091	0.98595
		<p>In Lecture3_Sc1007, constants are ignored in asymptotic analysis because the focus is on the growth rate of functions as the input size becomes very large. The idea is that when analyzing the efficiency of algorithms, the exact values of constants become less significant compared to how the functions behave as they approach infinity. This is illustrated when the lecturer mentions that constants like 0.1, 0.5, or 0.33333 can be used as long as they fulfill the criteria of being constant, emphasizing that the specific value does not affect the overall growth classification of the function. This is covered at [30:43].</p> <p>Additionally, the lecturer points out that when deriving time complexity, it is more important to understand the order of growth rather than the exact linear function, which is why simplifications are often made to focus on the dominant term. This is discussed around [54:35].</p>			
14	What are the three asymptotic notations introduced by the lecturer in Lecture3_Sc1007?	0.805556	0.5	1	0.993745
		The three asymptotic notations introduced by the lecturer in Lecture3_Sc1007 are big O, big Theta, and big Omega. This was mentioned at the timestamp "Covered at 11:24".			
15	What is the best-case time complexity of sequential search in a linked list in Lecture3_Sc1007?	0.804167	0.5	0.833333	0.965353
		The best-case time complexity of sequential search in a linked list, as discussed in Lecture3_Sc1007, is constant time, denoted as $\Theta(1)$. This occurs when the search key is the first item in the list, allowing you to find it immediately without needing to check any other items. This is covered at [40:50].			
16	In Lecture3_Sc1007, what is the worst-case	1	0.5	0.833333	0.983588
		In Lecture3_Sc1007, the worst-case time complexity of sequential search is described as Big Theta N ($\Theta(N)$). This occurs when you need to search through all the elements in the list, particularly if the search key is at the last position or not present at all. The explanation			

	time complexity of sequential search?	highlights that in the worst-case scenario, you would need to run $N - 1$ iterations plus one additional operation, leading to a linear function. This is summarized at the timestamp "Covered at 42:19".			
17	In Lecture3_Sc1007, what is the average-case complexity of sequential search?	1	1	1	0.986304
		In Lecture3_Sc1007, the average-case complexity of sequential search is concluded to be $\Theta(N)$. This means that, on average, the time complexity is linear with respect to the number of items being searched. This information is covered at [50:45].			
18	What improvement does the lecturer suggest over binary search for faster lookups in Lecture4_Sc1007?	0.8875	0.5	1	0.973435
		The lecturer suggests using a hash table as an improvement over binary search for faster lookups. While binary search can improve search time to $\Theta(\log_2 N)$, a hash table can achieve constant time complexity, $O(1)$, for search operations. This is accomplished by storing data in a way that may require more space but allows for efficient lookups, insertions, and deletions of key-value pairs. This approach represents a trade-off between space and time, where you sacrifice space to achieve better time complexity. This information is covered at [08:15] and [09:00].			
19	In Lecture4_Sc1007, what trade-off is mentioned when using hash tables?	0.966667	0.5	1	0.987457
		In Lecture4_Sc1007, the trade-off mentioned when using hash tables is between space and time. Specifically, to achieve better time complexity for search operations (improving from linear to constant time), you need to sacrifice space by using more memory to create a larger hash table. This trade-off allows for efficient lookup, insertion, and deletion of key-value pairs. This concept is covered at multiple points, including "This kind of algorithm or approach is a trade-off between space and time" at [08:54] and "However, to achieve this, you need to sacrifice space" at [01:09:49].			
20	In Lecture4_Sc1007, what is hashing defined as?	1	1	1	0.976756
		In Lecture4_Sc1007, hashing is defined as the process of using a hash table to map data of arbitrary size to a fixed size. This involves creating a mapping through a hash function, which allows for the organization of data into a fixed-size array or table. This definition is covered at "Covered at 13:21".			

21	What problem arises when two values map to the same slot in a hash table discussed in In Lecture4_Sc1007?	1	0.5	1	0.92984
		<p>The problem that arises when two values map to the same slot in a hash table is known as a "collision." When a collision occurs, it means that two different data items are trying to map to the same location or index in the hash table. This situation requires a method to resolve the collision, as it can affect the efficiency of data retrieval. The need to address collisions is a significant aspect of designing hash tables, as mentioned in the lecture.</p> <p>This concept is covered at multiple points in the lecture, including the explanation of collisions at timestamps (14:19.80) and (19:37.76).</p>			
22	In Lecture4_Sc1007, why is it recommended to make the table size a prime number?	1	1	1	0.97215
		<p>In Lecture4_Sc1007, it is recommended to make the table size a prime number to avoid clustering and ensure a more uniform distribution of keys. A prime number helps to minimize the chances of collisions, as it prevents the hash function from being influenced by factors of the table size. If the table size is not a prime number, certain data distributions may lead to clustering, where multiple keys map to the same index, resulting in collisions. This is particularly important when using modular arithmetic with real data, as a non-prime size can lead to inefficient probing and searching within the hash table.</p> <p>This concept is covered at multiple points, including the explanation that "to avoid this, we usually choose the size of the table to be a prime number" (27:13.80) and the note that "you also need to choose a prime number to ensure that the number you use is not a factor of the slot size" (01:03:18.68).</p>			
23	Mentioned in Lecture4_Sc1007, what collision resolution technique stores multiple keys in a linked list at the same slot?	0.466667	0.5	1	0.958045
		<p>The collision resolution technique that stores multiple keys in a linked list at the same slot is known as "closed addressing hashing", also referred to as "separate chaining". In this method, all keys that map to the same location are stored in a linked list, and the head pointer of this linked list is stored in the hash table. When searching for a key, the hash function directs you to the appropriate location, where you can then traverse the linked list to find the key. This technique is covered at [31:40].</p>			
24	In Lecture4_Sc1007,	0.75	0.5	1	0.923295
		<p>In Lecture4_Sc1007, linear probing is described as a method used in open addressing for handling collisions in a hash table. When a collision occurs (i.e., when a slot is already</p>			

	what is linear probing in open addressing?	<p>occupied), linear probing involves checking the next available slot in a sequential manner. Specifically, if the first insertion is at index (I), the next slot to check would be $(I + 1) \mod H$, where (H) is the total number of slots in the hash table. This process continues until an empty slot is found for the new key.</p> <p>The concept is introduced as straightforward, where you simply add one to the index of the occupied slot to find the next available one. This method is covered in detail at the timestamp "Covered at 51:48" and further explained with an example of inserting keys at "Covered at 52:01".</p>			
25	What is secondary clustering in quadratic probing talked about In Lecture4_Sc1007?	0.8875	0.5	1	0.964589
		<p>Secondary clustering in quadratic probing refers to a situation where, despite using a non-sequential probing method, the keys are still inserted in a fixed pattern. This occurs because if the probing sequence is fixed, the keys will always be inserted in that particular order. Consequently, when searching for a key, the search will also follow this fixed pattern, leading to clustering of keys. This clustering can negatively impact the efficiency of the search process, as it may cause multiple keys to be located in close proximity, making it take longer to find a specific key.</p> <p>This concept is discussed in the video at the timestamp "Covered at 58:14".</p>			
26	In Lecture4_Sc1007, why is double hashing preferred over quadratic probing?	0.333333	1	0.6	0.963067
		<p>Double hashing is preferred over quadratic probing because it reduces the likelihood of forming clusters in the hash table. In double hashing, each key generates a different probe sequence, which means that the search time does not become linear and is less likely to lead to clustering of keys. This is highlighted in the lecture when it states that "every key will get a different kind of probe to look for alternate slots" and that this approach "will not form clusters" (Covered at 01:01:55.76).</p> <p>In contrast, quadratic probing can still lead to clustering, as it may not sufficiently disperse the keys when collisions occur. Therefore, double hashing is favored for its efficiency in maintaining a manageable search time and avoiding the issues associated with clustering.</p>			
27	In Lecture5_Sc1007, what topics are	0.916667	0.5	1	0.981168
		<p>In Lecture5_Sc1007, the topics covered include an introduction to graphs, how to represent them, and specifically two representations: the adjacency matrix and the adjacency list. Additionally, there is mention of traversing graphs, with a focus on breadth-first search and depth-first search, which may be covered in the next lecture if time allows. This is outlined at</p>			

	covered in this lecture?	various points in the lecture, particularly at "Covered at 09:04" for the introduction to graphs and "Covered at 09:35" for the representations. The traversal methods are mentioned towards the end, with a specific note on their introduction in the next lecture at "Covered at 57:07".			
28	In Lecture5_Sc1007, when will the lecturer release the assignment?	0.45	0.75	0.857143	0
		The lecturer will release the next assignment after finishing the portion on graphs, which may be either this Wednesday or next Monday. He encourages students to start working on the assignment by logging in, looking at the questions, and trying out a little bit. The assignment invitation link will only last for two weeks. This information is covered at [09:55].			
29	In Lecture5_Sc1007, what two components make up a graph?	0.966667	0.5	1	0.947646
		In Lecture5_Sc1007, a graph consists of two components: the vertex and the edge. The vertex represents the nodes of the graph, while the edge represents the links that connect these vertices. This is covered at [10:37.72].			
30	In Lecture5_Sc1007, what makes a tree a special kind of graph?	0.854167	1	1	0.952788
		A tree is considered a special kind of graph because it is defined as a graph that has no cycles. This means that in a tree structure, you cannot find any closed loops or cycles connecting the vertices. If any siblings in a tree are connected, it would form a cycle, which would disqualify it from being a tree. Therefore, a tree is characterized by its acyclic nature, making it a unique case within the broader category of graphs. This concept is explained in the lecture at "Covered at 15:00".			

Appendix H

Baseline RAG - RAGAS Evaluation Scores for Multi Docs queries with Output Response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
--------	----------	-------------------	----------------	--------------	-----------

1	Across the two lectures Sc1007_Videolecture and Lecture2_videolecture, which topics are planned after the analysis of algorithms and when might hash tables be covered?	0.833333	0.333333	0.833333	0.928236
		<p>In the lectures Sc1007_videolecture and Lecture2_videolecture, the topics planned after the analysis of algorithms include hash tables and graph problems. Specifically, the analysis of algorithms will be covered in the upcoming lecture, and following that, hash tables will be introduced. This is mentioned in the context of the course structure, where it states that the second half of the module will cover hash tables and graph problems, with hash tables being introduced after discussing searching algorithms and binary search trees.</p> <p>The hash tables are specifically mentioned to be covered in detail during a lecture that focuses on them, which is indicated to occur after the analysis of algorithms session. This is confirmed by the statement that the lecture will cover hash tables first, and if time allows, the time complexity of binary search will also be discussed.</p> <p>For more details, you can refer to the timestamps:</p> <ul style="list-style-type: none"> - Analysis of algorithms is mentioned at "Covered at 57:20" - Hash tables are introduced at "Covered at 01:19" in Lecture2_videolecture. 			
2	"Which lecture between Sc1007_Videolecture and Lecture2_videolecture, previews asymptotic notation and which one defines Big O, Big Omega, and Big Theta in detail?"	1	0.666667	0.583333	0.915971
		<p>The lecture that previews asymptotic notation is Lecture3_Sc1007, where it introduces the concept of asymptotic notations, including big O, big Omega, and big Theta, at 11:24.12.</p> <p>In contrast, Lecture2_Sc1007 provides a more detailed definition of these notations. It explains that big O represents functions that grow at a lower or the same rate as a given function, while big Omega represents functions that grow at a higher rate or the same rate, and big Theta represents functions that grow at the same rate. This detailed explanation can be found at 01:01:51.68 for the introduction of the notations and further elaboration on their definitions at 01:05:41.68.</p>			

		In summary, Lecture3_Sc1007 previews asymptotic notation, while Lecture2_Sc1007 defines big O, big Omega, and big Theta in detail.			
		0.2	0	0.833333	0.925445
3	Summarize how recursion impacts time and space across the Sc1007_Videolecture and Lecture2_videolecture lectures using examples given.	<p>In the Sc1007_videolecture and Lecture2_videolecture, recursion is discussed in terms of its impact on time and space complexity.</p> <p>1. Time Complexity: The lectures indicate that when dealing with recursive functions, the time complexity is often not constant. This is because recursive calls can lead to multiple evaluations of the same subproblems, which can increase the overall time taken to compute a result. The lecturer mentions that there are ways to optimize recursive functions, suggesting that while recursion can be less efficient, it can be improved through techniques like memoization, where results of expensive function calls are stored for reuse. This is highlighted when the lecturer states, "You can optimize it" and "you will find that you can improve efficiency" (Covered at 58:46).</p> <p>2. Space Complexity: The space complexity of recursive functions is also a significant concern. Each recursive call typically requires additional space on the call stack, which can lead to increased memory usage. The lecturer notes that when using recursion, "you will put it in global storage as a global variable," indicating that the space is occupied due to the nature of recursive calls (Covered at 59:06). Furthermore, the space complexity is described as being easier to analyze than time complexity, as it primarily involves counting the variables and arrays declared in the algorithm (Covered at 01:09:27).</p> <p>In summary, recursion can complicate both time and space complexity due to the nature of multiple calls and the use of stack space, but optimizations can be applied to mitigate these issues.</p>			

4	From both lectures Sc1007_Videolecture and Lecture2_videolecture, what is the difference between an algorithm and a program, and how is efficiency evaluated?	0.755556	0.5	1	0.915884
		<p>The difference between an algorithm and a program is that an algorithm is a well-defined step-by-step procedure to solve a problem, while a program is a concrete implementation of that algorithm in a specific programming language. An algorithm serves as the idea or blueprint for solving a problem, whereas a program is the actual code that executes the algorithm. This distinction is emphasized in the lecture, where it is stated that "an algorithm is just a procedure; you can think of it as an idea to solve the problem" and that "a computer program is an instance, or concrete representation of an algorithm" (Covered at 21:20).</p> <p>Regarding efficiency, the lectures indicate that while understanding how to implement an algorithm is crucial, there is also an expectation to know how to select the most efficient algorithm at higher levels of study. Initially, the focus is on implementing and realizing the algorithm to solve the problem effectively, rather than immediately selecting the best solution (Covered at 13:18). The evaluation of efficiency typically involves determining which algorithm is the most efficient for a given problem, but this is a more advanced consideration that comes after mastering the implementation of algorithms (Covered at 13:24).</p> <p>In summary, an algorithm is a theoretical concept, while a program is its practical application, and efficiency is evaluated based on the ability to select and implement the most effective algorithm for a problem.</p>			
5	Combine the examples: which method is most efficient for summing 1..N, and how is this justified by the complexity principles in both Sc1007_Videolecture and Lecture2_videolecture lectures.	1	0.666667	1	£0.89
		<p>To determine the most efficient method for summing numbers from 1 to N, we can analyze the examples provided in the Sc1007_videolecture and the complexity principles discussed in the Lecture2_videolecture.</p> <p>In the Sc1007_videolecture, three methods for summing numbers from 1 to N are presented. The most efficient method is identified as the one that uses</p>			

		<p>the formula for the arithmetic series, which is $\frac{N}{2} \times (1 + N)$. This method is efficient because it requires only a single multiplication, one addition, and one division, resulting in a constant time complexity of $O(1)$ (Covered at [24:39]).</p> <p>In contrast, the other two methods involve either a for loop or a recursive function. The for loop iterates through all numbers from 1 to N, leading to a linear time complexity of $O(N)$ (Covered at [24:46]). The recursive method also has a higher time complexity due to the increasing number of recursive calls, which can lead to exponential growth in the number of operations required (Covered at [25:51]).</p> <p>The justification for the efficiency of the arithmetic series formula is rooted in the principles of time complexity discussed in the Lecture2_videolecture. Here, the analysis of algorithms emphasizes the importance of understanding the number of operations required as the input size grows. For the arithmetic series, the operations remain constant regardless of N, while the other methods' operations increase linearly or exponentially with N (Covered at [51:05] and [52:12]).</p> <p>In summary, the most efficient method for summing numbers from 1 to N is the arithmetic series formula, justified by its constant time complexity compared to the linear and exponential complexities of the other methods.</p>			
6	What searching approaches are mentioned in Sc1007_Videolecture and Lecture2_videolecture, and how do their time complexities differ?	0.833333	0.666667	0.8	0.895981
		<p>In the Sc1007_videolecture, two searching approaches are mentioned: linear search and hash table search.</p> <p>1. Linear Search: This approach involves checking each item in a sequence to find a key. The time complexity for linear search is:</p> <ul style="list-style-type: none"> Worst-case complexity: $O(n)$, where n is the number of items in the sequence. This occurs when the key is not present or is the last item checked. 			

		<p>- Best-case complexity: $O(1)$, which happens if the first item is the key. This is covered at [34:47].</p> <p>2. Hash Table Search: This method uses a hash function to map keys to specific indices, allowing for constant time complexity when searching for an item. The time complexity for hash table search is:</p> <p>- Average-case complexity: $O(1)$, as it allows for direct access to the index where the item is stored. However, there are trade-offs in terms of storage space. This is discussed at [35:34].</p> <p>In the Lecture2_videolecture, the time complexities of these searching methods are summarized as follows:</p> <p>- Sequential Search (similar to linear search):</p> <p>- Worst-case complexity: $O(n)$</p> <p>- Best-case complexity: $O(1)$ if the key is the first item.</p> <p>This is mentioned at [01:08:33].</p> <p>In summary, linear search has a time complexity of $O(n)$ in the worst case, while hash table search can achieve $O(1)$ on average, making it significantly more efficient for searching in large datasets.</p>			
7	Considering the course logistics from Sc1007_Videolecture and Lecture2_videolecture lectures, what assessments contribute to the final grade?	0.916667	0.5	1	0.934504
		<p>In the SC1007 course, the assessments that contribute to the final grade include:</p> <p>1. Assignments: These account for a total of 40%, with Dr. Owen's part contributing 20% and another 20% from the other instructor's part. Covered at [07:30].</p> <p>2. Lab Tests: There are two lab tests, each contributing 20% to the final grade. The first test is scheduled for Week 1 and the second for Week 14. Covered at [07:54].</p>			

		<p>3. **Final Quiz**: This also contributes 20% to the final grade. Covered at [07:54].</p> <p>Overall, attendance for the tests is compulsory, and failing to attend can result in receiving zero marks for that portion of the grade. Covered at [08:06].</p>			
8	Using both lectures Sc1007_Videolecture and Lecture2_videolecture, explain why constants are ignored in asymptotic analysis and give an example where $N^2 + 100$ and N^2 are treated the same.	0.95	0.666667	1	0.882199
		<p>In asymptotic analysis, constants are ignored because they become negligible as the problem size (N) increases. The focus is on the growth rate of the algorithm's time complexity rather than the exact number of operations or the time taken for each operation. This is because, for large values of N, the highest order term dominates the performance of the algorithm.</p> <p>For example, in the case of the time complexities $N^2 + 100$ and N^2, as N grows larger, the +100 becomes insignificant compared to N^2. This is illustrated in the lecture where it is stated that "the +100 is negligible" when considering large problem sizes, and that "if you compare $13N$ squared and $100N$ squared, although $100N$ squared is slower, the growth is about the same" (Covered at 57:22).</p> <p>Thus, both $N^2 + 100$ and N^2 are treated the same in asymptotic analysis because their growth rates are equivalent as N approaches infinity.</p>			
9		0	0	1	0.880745
	Across the Sc1007_Videolecture and Lecture2_videolecture videos, which graph-related topics are planned and what real-world path problem is used as an example?	<p>In the Sc1007_videolecture, the planned graph-related topic is the traveling salesman problem, which is a well-known problem in computer science. This problem involves a set of cities that need to be visited once, with the goal of returning to the starting city while finding the shortest possible route. The lecturer explains that to find the best solution, one must try all possible combinations of routes, which is a key characteristic of this problem. This is covered at "Covered at 38:08".</p>			

		Additionally, the lecturer mentions that the traveling salesman problem is a classic example of a network or graph problem, emphasizing its significance in the field of computer science.			
10	What guidance is given about coding vs. concepts across the Sc1007_Videolecture and Lecture2_videolecture videos, and how should students prepare?	1	0.666667	1	0.897499
		<p>In the Sc1007_videolecture, the guidance emphasizes the importance of not just reading solutions but actively learning to implement algorithms yourself. The instructor stresses that understanding how to code is crucial, as merely knowing the solution does not provide insight into how it was derived. Students are encouraged to practice coding to develop their skills, as programming can only be honed through practice. This is highlighted at the timestamp "Covered at 01:00:53.12" and "Covered at 09:09.12".</p> <p>Additionally, the lecture discusses the need for students to be proficient in coding before advancing to higher-level courses. It is mentioned that the focus will be on realizing algorithms rather than solely on finding the most efficient solution, as long as students can solve problems effectively. This is covered at "Covered at 12:58.28".</p> <p>To prepare, students should engage in practical exercises, both in labs and at home, to apply the concepts learned in lectures. They should also familiarize themselves with theoretical concepts related to algorithm design and analysis, as these will be essential for future courses. This preparation is underscored at "Covered at 10:02.80".</p> <p>Overall, students are advised to balance their understanding of theoretical concepts with practical coding skills to succeed in their studies.</p>			
11	Which graph-related problems are planned in Sc1007_Videolecture , and what real-world application	0	0	0.8	0.835193
		In the Sc1007_videolecture, one of the graph-related problems discussed is the traveling salesman problem. This problem involves a set of cities that need to be visited once, with the goal of returning to the starting city while			

	is used in Lecture5_videolecture?	<p>finding the shortest path. It is highlighted as a well-known graph problem in computer science, where all combinations must be tried to determine the best solution. This is covered at "Covered at 38:04".</p> <p>As for real-world applications mentioned in Lecture 5, one application of the traveling salesman problem is in cryptography. In this context, encoding and decoding messages must ensure that all patterns have a unique solution, which is crucial for secure communication. This application is discussed around "Covered at 39:13".</p>			
12	Which lecture between Sc1007_Videolecture and Lecture3_videolecture previews asymptotic notation, and which defines Big O, Big Omega, and Big Theta in detail?	0.866667	0	0.777778	0.905617
		<p>The lecture that previews asymptotic notation is Sc1007_videolecture. It mentions that the upcoming topics will include the analysis of algorithms and the representation of order of growth using big O notation, among other concepts. This is covered at (57:20.60).</p> <p>On the other hand, Lecture3_Sc1007 defines Big O, Big Omega, and Big Theta in detail. It introduces these notations and explains their meanings, specifically stating that they represent sets of functions based on their order of growth. This is covered at (11:24.12) and (11:31.24).</p>			
13	How is search complexity contrasted in Lecture3_videolecture sequential search and Lecture4_videolecture binary search?	1	0.5	0.470588	0.943779
		<p>In the context of search complexity discussed in the lectures, the sequential search and binary search have distinct characteristics:</p> <ol style="list-style-type: none"> Sequential Search: <ul style="list-style-type: none"> The worst-case time complexity is $O(n)$, where n is the number of elements in the array. This occurs when the key is either the last item or not present at all, requiring a comparison with every element. This is highlighted in Lecture3 at (03:46.44), where the time complexity of sequential search is discussed. Binary Search: 			

		<p>- In contrast, binary search operates on a sorted array and has a much better time complexity of $O(\log n)$. This means that with each comparison, the search space is halved, making it significantly more efficient than sequential search for large datasets. The binary search time complexity is mentioned in Lecture4 at (01:25.56), where the lecture transitions from discussing sequential search to binary search.</p> <p>Thus, the key difference lies in their time complexities: sequential search is linear $O(n)$, while binary search is logarithmic $O(\log n)$, making binary search more efficient for larger datasets.</p>			
14	What assessments are mentioned in Sc1007_Videolecture and Lecture2_videolecturevideos and how do they count toward the final grade?	0.583333	0.25	0.923077	0
		<p>In the _ Sc1007videolecture, the assessments mentioned include assignments, lab tests, and a final quiz. The breakdown of how these assessments count toward the final grade is as follows:</p> <ul style="list-style-type: none"> - Assignments: 40% total (20% from Dr. Owen's part and 20% from another part) (Covered at 07:30). - Lab Tests: There are two lab tests, each contributing 20% to the final grade. The upcoming test in Week 1 is 20%, and another in Week 14 is also 20%, making a total of 40% from lab tests (Covered at 07:54). - Final Quiz: This contributes an additional 20% to the final grade (Covered at 07:54). <p>Attendance for the tests is compulsory, and failing to attend will result in receiving zero marks for that portion of the grade (Covered at 08:06).</p> <p>Unfortunately, I could not find specific information regarding assessments in the Lecture2_videolecture as it was not included in the provided context.</p>			
15	Why are constants ignored in asymptotic analysis, and what example is provided	0.2	1	0.866667	0.933745
		<p>In asymptotic analysis, constants are ignored because the focus is on the growth rate of functions as the input size approaches infinity. This means</p>			

	across the Sc1007_Videolecture and Lecture3_videolecture videos?	<p>that while constants may affect the performance for small input sizes, they become less significant as the input grows larger. The goal is to understand how an algorithm's performance scales, rather than the exact number of operations it performs.</p> <p>An example provided in the context is the Big O notation, where a function like $f(n) = 4n + 3$ is considered to be in $O(n)$ because, as n becomes very large, the term $4n$ dominates the behavior of the function, and the constant 3 becomes negligible. This is discussed in the Lecture3 video, specifically at the timestamp "29:53.32" where it mentions that constants are not significant in the context of asymptotic growth.</p> <p>Thus, the essence of asymptotic analysis is to simplify the representation of an algorithm's efficiency by focusing on the most impactful terms as the input size increases, effectively ignoring constants.</p>			
16	How are linked lists used differently in Lecture4_videolecture hash tables and Lecture5_videolecture adjacency lists?	0.8875	0.5	0.583333	0.929107
		<p>In Lecture 4, linked lists are used in the context of hash tables. Specifically, when discussing hash tables, the lecture introduces a searching algorithm that utilizes linked lists to handle collisions. This means that when multiple elements hash to the same index, they are stored in a linked list at that index, allowing for linear search through the list to find the desired element. This is covered at "03:58.76" where it mentions introducing another type of searching algorithm that uses hash tables.</p> <p>In Lecture 5, linked lists are employed as part of the adjacency list representation of graphs. Each vertex in the graph has a linked list that stores its connections to other vertices. This allows for efficient representation of sparse graphs, where the number of edges is much less than the maximum possible. The lecture explains that for each vertex, a linked list is used to represent the connections, and this is discussed at "09:35.08" where it mentions using an array to represent vertices and linked lists for connections.</p>			

		In summary, linked lists in Lecture 4 are primarily used for collision resolution in hash tables, while in Lecture 5, they are used to represent connections in the adjacency list of a graph.			
17	How is efficiency of summing 1..N compared in both Sc1007_Videolecture and Lecture3_videolecture ?	1	0.5	0.75	0
		<p>The efficiency of summing numbers from 1 to N is discussed in the Sc1007_videolecture, where different algorithms are compared. The most efficient method mentioned is a direct formula that calculates the sum in one line of code: $\frac{N}{2} \times (1 + N)$. This method is highlighted as efficient because it requires only one multiplication, one addition, and one division, leading to an instant result. This is covered at [24:39].</p> <p>In contrast, the other methods, such as using a for loop or recursive functions, involve more iterations or recursive calls, which makes them less efficient as the number of iterations grows larger. This is explained at [26:21].</p> <p>Unfortunately, I do not have access to the content of Lecture3_videolecture, so I cannot provide a comparison with that specific lecture. If you have details from Lecture3_videolecture, I would be happy to help analyze the differences in efficiency further.</p>			
18	What graph representations are mentioned across Lecture3_videolecture and Lecture5_videolecture?	0.916667	0.5	0.666667	0
		<p>In Lecture 5, two graph representations are specifically mentioned: the adjacency matrix and the adjacency list. These representations are introduced as common ways to represent graphs, with their advantages and disadvantages discussed in relation to different applications. This is covered at multiple points, including at timestamps (09:35.08) and (30:22.04).</p> <p>Unfortunately, I do not have access to the content of Lecture 3, so I cannot provide information on the graph representations mentioned in that lecture. If</p>			

		you have specific details or timestamps from Lecture 3, I would be happy to help analyze them!			
19	What real-world applications of graphs are given across Sc1007_videolecture and Lecture5_videolecture?	0.583333	1	0.88	0.952388
		<p>The real-world applications of graphs mentioned in the lectures include:</p> <ol style="list-style-type: none"> Traveling Salesman Problem: This is a well-known graph problem where you have a set of cities to visit, and you want to find the shortest path that visits each city once and returns to the starting point. This problem is significant in optimization and logistics. Covered at [38:04]. Computer Networks: Graphs are used to model computer networks where computers are nodes and connections between them are edges. This application involves finding the best path for data transfer through routers and switches. Covered at [27:48]. Electrical Circuits: In electrical engineering, graphs represent components as nodes and connections as edges, which helps in optimizing the layout of circuits on a PCB (Printed Circuit Board). Covered at [28:53]. Maps and Navigation: Graphs can represent routes and locations, allowing for the search of the shortest or cheapest path between points, such as traveling from one location to another. Covered at [27:44]. <p>These applications illustrate the versatility of graph theory in various fields, including logistics, computer science, and engineering.</p>			
20	What is the average-case complexity of search mentioned in Lecture3_videolecture and Lecture4_videolecture?	0.916667	1	1	0.907016
		<p>The average-case complexity of the sequential search is $\Theta(N)$, as mentioned in the context of Lecture3 and Lecture4. This means that, on average, you will need to check a linear number of items in the list to find the search key. This conclusion is drawn from the analysis presented in the lectures, where it is stated that the average case is always linear due to the nature of the search</p>			

		<p>process.</p> <p>This information is specifically covered at "Covered at 50:45" in Lecture3 and "Covered at 52:20" in Lecture4.</p>			
		0.588889	0.5	1	0.969175
21	<p>What guidance is given about tutorials, labs, and coding across Sc1007_videolecture and Lecture2_videolecture ?</p>	<p>In the SC1007 video lecture, guidance regarding tutorials, labs, and coding is provided as follows:</p> <ol style="list-style-type: none"> 1. Tutorials: The tutorials are designed to clarify doubts that students may encounter during lab sessions. There is an opportunity to meet biweekly to discuss these doubts, and if students still have questions, they are encouraged to reach out via email for further assistance. This is covered at "Covered at (13:58.60)". 2. Labs: The lab sessions focus on practical application and implementation of the concepts learned in lectures. Students are expected to practice coding and apply their knowledge of data structures and algorithms. The importance of understanding how to implement solutions rather than just reading them is emphasized, as it helps in grasping the underlying concepts better. This is highlighted at "Covered at (01:00:53.12)". 3. Coding: The course emphasizes the importance of coding proficiency. Students are expected to implement algorithms and data structures themselves, rather than relying solely on provided solutions. The instructor mentions that they will not focus on whether the solution is the most efficient, as long as it solves the problem correctly. This is discussed at "Covered at (12:38.92)". <p>Overall, the guidance stresses the importance of active participation in tutorials and labs, along with a strong emphasis on coding practice and implementation skills.</p>			

22	How is graph connectivity defined in Lecture5_videolecture compared to complexity bounds in Lecture3_videolecture ?	0.5	0.25	0.636364	0
		<p>In Lecture 5, graph connectivity is defined in terms of vertices and edges. A graph consists of a set of vertices (or nodes) and a set of edges that connect these vertices. The minimum number of edges can be zero, while the maximum number of edges is determined by the formula $\frac{V \times (V - 1)}{2}$, where V is the number of vertices. This means that in a fully connected graph, each vertex can connect to every other vertex, leading to a maximum of 10 edges for 5 vertices, as illustrated in the lecture (Covered at 13:44).</p> <p>Unfortunately, I do not have access to the content of Lecture 3, so I cannot provide a comparison regarding complexity bounds. If you have specific details or excerpts from Lecture 3, I would be happy to help analyze or compare them!</p>			
23	How do adjacency lists relate to hash tables as explained across Lecture4_videolecture and Lecture5_videolecture?	1	0.5	1	0
		<p>The context provided does not contain specific information about how adjacency lists relate to hash tables as explained in Lecture4_videolecture and Lecture5_videolecture. Therefore, I am unable to find an answer to your question. If you have any other questions or need clarification, please feel free to ask!</p>			
24	What are the main considerations when analyzing if/else in Lecture2_videolecture versus collisions in Lecture4_videolecture?	0	0	0.636364	0.847626
		<p>In Lecture 2, the focus is on understanding the concepts of if/else statements and their implications in programming. The main considerations include how to effectively implement these conditional statements to control the flow of a program. The importance of practice in applying these concepts is emphasized, as highlighted at "Covered at 09:09" and "Covered at 09:30".</p> <p>In contrast, Lecture 4 discusses collisions in the context of hash tables. The main considerations here involve understanding how collisions occur when two or more data items map to the same index in a hash table. The lecture covers the need to resolve these collisions and the challenges associated with</p>			

		<p>deleting keys from a hash table, as mentioned at "Covered at 14:19" and "Covered at 14:49".</p> <p>In summary, while Lecture 2 focuses on the implementation and practice of conditional statements, Lecture 4 addresses the complexities of data mapping and collision resolution in hash tables.</p>			
25	Which parts of the module are described as concept-heavy, and how do labs balance this across Sc1007_videolecture and Lecture2_videolecture ?	1	1	1	0.860234
		<p>The module SC1007 is described as having a significant focus on theoretical concepts, particularly in the areas of algorithms and data structures. The lecture emphasizes that understanding these concepts is crucial before moving on to practical applications. For instance, the analysis of algorithms, including time and space complexity, is highlighted as a key theoretical component that prepares students for more advanced topics in subsequent modules (Covered at 10:02).</p> <p>In terms of balancing this theoretical focus with practical application, the labs are designed to reinforce the concepts learned in lectures. The instructor mentions that after understanding the concepts, students need to practice in the lab to develop their programming skills (Covered at 09:12). This hands-on practice is essential for applying the theoretical knowledge gained during lectures, ensuring that students can implement algorithms and data structures effectively.</p> <p>Overall, while the lectures provide a strong conceptual foundation, the labs serve to bridge the gap between theory and practice, allowing students to apply what they have learned in a practical setting.</p>			
26	How are nested loops explained in Lecture2_videolecture and Lecture3_videolecture , and	0.25	1	0.642857	0.935946
		<p>In the Lecture2 video, nested loops are explained through an example where the outer loop runs from 1 to N, and the inner loop runs based on the current value of the outer loop variable. Specifically, the inner loop performs I^2 squared iterations for each iteration of the outer loop. The total number of</p>			

	<p>what time complexity do they lead to?</p>	<p>operations is calculated as the sum of squares: 1 squared + 2 squared + 3 squared + ... + N squared. This leads to a time complexity of N cubed ($O(N^3)$) for the algorithm being analyzed. This explanation is covered at "51:43.80" to "53:30.80".</p> <p>In the Lecture3 video, while the specific details of nested loops are not explicitly mentioned in the provided transcript, it is indicated that the analysis of algorithms, including time complexity, will be discussed further. The focus will be on understanding how to represent time complexity using asymptotic notation, which includes the implications of nested loops. This is mentioned at "58:04.40" to "58:11.24".</p> <p>Overall, the nested loops lead to a time complexity of $O(N^3)$ as explained in Lecture2.</p>			
27	<p>What is the difference between an algorithm and a program, and how is efficiency evaluated across Sc1007_Videolecture and Lecture2_videolecture ?</p>	1	0.5	0.8	0.954422
		<p>The difference between an algorithm and a program is primarily in their definitions and purposes. An algorithm is a well-defined step-by-step procedure to solve a problem, while a program is a concrete implementation of that algorithm in a specific programming language. This distinction is highlighted when the speaker mentions that an algorithm is just an idea to solve a problem, and it is the programmer's task to implement it as a program that takes input and produces output. This is covered at 21:20 and 21:53.</p> <p>Regarding the evaluation of efficiency, it is discussed in the context of analyzing algorithms. The analysis focuses on the speed and scalability of algorithms, particularly how their efficiency changes as input sizes grow. The speaker emphasizes the importance of understanding time complexity (the time taken to run the algorithm) and space complexity (the memory used) to identify the most efficient algorithm for a given problem. This is covered at 03:30 and 04:33.</p>			

		In summary, an algorithm is a theoretical concept, while a program is its practical application. Efficiency is evaluated through analysis of time and space complexities, which helps in selecting the most suitable algorithm for a problem.			
28	In Lecture2_videolecture and Lecture4_videolecture, what considerations are made for selection structures and hash table collisions?	0.5	0.5	0.583333	0.933094
		<p>In the context of selection structures and hash table collisions, the following considerations are made:</p> <ol style="list-style-type: none"> 1. **Selection Structures**: In the earlier lectures, particularly when discussing linked lists and binary search trees, the focus was on the time complexity of searching algorithms. The time complexity for sequential search (or linear search) and binary search was highlighted, indicating the importance of understanding how these structures operate in terms of efficiency. This foundational knowledge is crucial when considering how to implement selection structures effectively. Covered at [01:25.56]. 2. **Hash Table Collisions**: When discussing hash tables, it is essential to consider that collisions occur when two or more data items map to the same index in the hash table. This situation is described as a collision, and resolving these collisions is a significant challenge in the design of hash tables. The lecture emphasizes that no matter how a hash table is designed, collisions will inevitably happen due to the mapping of arbitrary-sized data to fixed-size values. Covered at [14:19.80] and [14:30.16]. <p>These considerations are fundamental to understanding how to effectively use selection structures and manage hash table collisions in data structures.</p>			
29	How does load factor in Lecture4_videolecture relate to loop growth in Lecture2_videolecture ?	1	0.5	0.578947	0.873192
		The load factor discussed in Lecture 4 relates to how efficiently a hash table utilizes its slots, which can indirectly affect the performance of algorithms that may involve loops, such as those analyzed in Lecture 3.			

		<p>In Lecture 4, the load factor is defined as the ratio of the number of data items (N) to the number of slots (H) in a hash table. A load factor of 1 indicates that all slots are filled, while a load factor of 0.5 means that only half of the slots are in use. This load factor helps evaluate the efficiency of the hash table and can influence the average search time, which remains $O(1)$ under optimal conditions. This is covered at [20:30].</p> <p>In Lecture 3, the analysis of algorithms focuses on the order of growth, particularly how the number of operations in an algorithm increases with the input size. For example, if an algorithm has a factorial growth rate, it will require significantly more operations as the input size increases, which can lead to inefficiencies in execution time. This is discussed at [07:14].</p> <p>While the load factor itself does not directly relate to loop growth, both concepts emphasize the importance of efficiency in data structures and algorithms. A well-designed hash table with an appropriate load factor can lead to faster lookups, which can be crucial when loops are involved in processing data, as they may need to access elements from the hash table frequently.</p> <p>In summary, the load factor in hash tables affects the efficiency of data retrieval, which can impact the overall performance of algorithms that utilize loops, as discussed in Lecture 3.</p>			
30	What distinguishes trees from general graphs across the lectures Lecture2_videolecture and Lecture5_videolecture?	0.833333	0.5	0.647059	0.948424
		<p>In the lectures, trees are distinguished from general graphs primarily by their structural properties.</p> <p>1. Cycles: A tree is defined as a special type of graph that has no cycles. This means that in a tree, there are no closed loops; every pair of vertices is connected by exactly one path. This is emphasized in Lecture 5, where it is</p>			

		<p>stated, "A tree can be considered a special case... it is a special graph with no cycles" (Covered at 14:57).</p> <p>2. Connectivity: Trees are connected graphs, meaning there is a path between any two vertices. This property is inherent to trees, while general graphs can be disconnected.</p> <p>3. Degree of Vertices: In a tree, the degree of a vertex (the number of edges incident to it) can vary, but the overall structure must maintain the acyclic property. In Lecture 5, it is mentioned that "the degree of a vertex is the number of edges incident to it" (Covered at 14:28), which applies to both trees and general graphs, but the implications of this degree are different in trees due to their acyclic nature.</p> <p>4. Special Case of Graphs: Trees are often described as a special case of graphs, which can include various types such as directed, undirected, weighted, and unweighted graphs. This distinction is made clear when the lecturer states, "A tree has no cycles... it is a special graph" (Covered at 15:27).</p> <p>In summary, the key distinctions are the absence of cycles, the requirement of connectivity, and the specific structural properties that define trees as a subset of graphs.</p>
--	--	---

Appendix I

Docs Scope RAG - RAGAS Evaluation Scores for Multi Docs queries with Output Response

Qn No.	Question	Context Precision	Context Recall	Faithfulness	Relevancy
1		0.833333	0.333333	0.857143	0.925862

	Across the two lectures Sc1007_Videolecture and Lecture2_videolecture, which topics are planned after the analysis of algorithms and when might hash tables be covered?	<p>In the Sc1007_videolecture, after the analysis of algorithms, the topics planned include hash tables and graph problems. Specifically, hash tables will be introduced following the analysis of algorithms, which is scheduled for this Wednesday. The lecture will cover time complexity and space complexity, discussing various efficiency cases and big O notation before moving on to hash tables and graph problems. This is covered at [57:20] and [03:37].</p> <p>If you have any further questions or need more details, feel free to ask!</p>			
2	"Which lecture between Sc1007_Videolecture and Lecture2_videolecture, previews asymptotic notation and which one defines Big O, Big Omega, and Big Theta in detail?"	0.8875	0.666667	0.333333	0.884463
		<p>The lecture that previews asymptotic notation is Sc1007_videolecture. It mentions that the analysis of algorithms will cover topics such as time complexity and space complexity, including the use of big O notation, which is discussed in more detail later on. This is covered at 57:39.</p> <p>On the other hand, the lecture that defines Big O, Big Omega, and Big Theta in detail is Lecture2_Sc1007. It introduces these three asymptotic notations and explains their significance in terms of the order of growth of functions. This is covered at 01:01:51.</p>			
3	Summarize how recursion impacts time and space across the Sc1007_Videolecture and Lecture2_videolecture lectures using examples given.	0.805556	0	0.954545	0.926572
		<p>Recursion significantly impacts both time and space complexity, as discussed in the Sc1007_videolecture and Lecture2_videolecture.</p> <p>1. Time Complexity: In the Sc1007_videolecture, it is noted that with recursive calls, the number of calls increases exponentially. For example, when N equals 4, there are five recursive calls made. As N increases, the number of calls grows rapidly, leading to a time complexity of approximately $\backslash(1.6^N\backslash)$ or sometimes referred to as $\backslash(2^N\backslash)$, indicating exponential growth in time required to complete the function. This is highlighted at "Covered at [30:16]" and "Covered at [31:08]".</p> <p>2. Space Complexity: The space complexity is affected by how data is</p>			

		<p>stored during recursion. The lecture mentions that recursive functions often use global variables to store results, which means that space is always occupied due to the declaration of these variables. This trade-off is discussed in terms of optimizing space to improve efficiency, as seen in "Covered at [58:46]". The recursive function's reliance on global storage can lead to increased space usage, which is a critical consideration when designing algorithms.</p> <p>3. **Comparison with Iterative Approaches**: The lecture contrasts recursive methods with iterative approaches, stating that iterative methods typically require linear time complexity (N iterations) and are more efficient in terms of time compared to their recursive counterparts, which can be significantly slower due to repeated calculations. This is mentioned at "Covered at [31:24]".</p> <p>In summary, recursion can lead to exponential time complexity and increased space usage due to the nature of recursive calls and data storage, making it essential to analyze and optimize these factors when designing algorithms.</p>				
4	From both lectures Sc1007_Videolecture and Lecture2_videolecture, what is the difference between an algorithm and a program, and how is efficiency evaluated?	0.804167	0.666667	1	0.948353	<p>The difference between an algorithm and a program is that an algorithm is a well-defined step-by-step procedure or set of rules to solve a problem, while a program is a concrete implementation of that algorithm in a specific programming language. An algorithm serves as the conceptual framework for solving a problem, whereas a program is the actual code that executes the algorithm. This distinction is highlighted when the speaker mentions that "an algorithm is just a procedure; you can think of it as an idea to solve the problem," and that "a computer program is an instance, or concrete representation of an algorithm" (Covered at 22:00).</p>

		<p>Regarding the evaluation of efficiency, it is primarily assessed through two aspects: time complexity and space complexity. Time complexity measures the amount of time an algorithm takes to run, while space complexity refers to the amount of memory used. The speaker emphasizes that understanding how efficiency changes as input sizes grow is crucial, and that sometimes there is a trade-off between time and space complexity when selecting an algorithm (Covered at 04:41 and 05:11). The analysis of algorithms helps identify the most efficient algorithm for a given problem, taking into account the resources available (Covered at 04:23 and 06:01).</p>			
5	<p>Combine the examples: which method is most efficient for summing 1..N, and how is this justified by the complexity principles in both Sc1007_Videolecture and Lecture2_videolecture lectures.</p>	0.8875	1	0.869565	£0.91
		<p>To determine the most efficient method for summing numbers from 1 to N, we can analyze the examples provided in both the Sc1007_videolecture and Lecture2_videolecture.</p> <p>In the Sc1007_videolecture, three methods are discussed for summing numbers from 1 to N:</p> <ol style="list-style-type: none"> Iterative Method: This involves using a for loop to iterate from 1 to N, adding each number to a sum variable. This method has a time complexity of $O(N)$ because it requires N iterations to compute the sum. Covered at [23:46]. Recursive Method: This method recursively calls itself, reducing N by 1 each time until it reaches the base case of N equal to 1. This also has a time complexity of $O(N)$ due to the N recursive calls made. Covered at [24:59]. Formula Method: The most efficient method is using the arithmetic series formula: $\text{Sum} = \frac{N(N + 1)}{2}$. This method has a time complexity of $O(1)$ because it computes the sum in a constant time, regardless of the size of N. Covered at [24:39]. <p>The justification for the efficiency of the formula method is rooted in the principles of algorithm analysis discussed in the Lecture2_videolecture. The analysis of algorithms focuses on evaluating the speed and scalability of different solutions, particularly how efficiency changes as input sizes grow. It emphasizes</p>			

		<p>the importance of understanding time complexity (the time taken to run the algorithm) and space complexity (the memory used). Covered at [03:30].</p> <p>In summary, the formula method for summing numbers from 1 to N is the most efficient due to its $O(1)$ time complexity, while the iterative and recursive methods both have $O(N)$ time complexity, making them less efficient as N increases. This analysis aligns with the principles of algorithm efficiency discussed in the lectures.</p>			
6	What searching approaches are mentioned in Sc1007_Videolecture and Lecture2_videolecture, and how do their time complexities differ?	0.5	0.666667	0.583333	0
		<p>In the Sc1007_videolecture, two searching approaches are mentioned: linear search and a more efficient method that can achieve logarithmic time complexity, although the latter is not explicitly named in the provided context.</p> <p>1. Linear Search: This is described as the most intuitive method where you check every single item in a sequence to find a key. The time complexity for linear search is $O(n)$, meaning it takes linear time relative to the number of items being searched. This is covered at [34:02].</p> <p>2. Logarithmic Time Complexity Search: While the specific algorithm achieving logarithmic time complexity is not detailed in the transcript, it is mentioned that there exists an alternative that can achieve this efficiency. This is indicated at [33:03].</p> <p>Unfortunately, I do not have information from the "Lecture2_videolecture" to compare the searching approaches and their time complexities. If you have specific details or transcripts from that lecture, I can help analyze them further.</p>			
7	Considering the course logistics from Sc1007_Videolecture and Lecture2_videolecture lectures, what assessments contribute to the final grade?	0.805556	0.5	1	0.934656
		<p>In the SC1007 course, the assessments that contribute to the final grade include assignments, two lab tests, and a final quiz. These components are designed to assess whether students have truly learned the material covered in the lectures and tutorials. The course emphasizes the importance of understanding concepts such as pointers, linked lists, stacks, queues, and other data structures, as well as the analysis of algorithms, including time and space complexity.</p>			

		<p>This information is covered at multiple points in the video, specifically when discussing the assessment components and their significance in evaluating student learning. For instance, it is mentioned that "All of these assess whether you really learned the material" at 14:14.28.</p>			
8	<p>Using both lectures Sc1007_Videolecture and Lecture2_videolecture, explain why constants are ignored in asymptotic analysis and give an example where $N^2 + 100$ and N^2 are treated the same.</p>	0.325	0.5	0.923077	0.882199
		<p>In asymptotic analysis, constants are ignored because the focus is on the growth rate of an algorithm as the input size (N) becomes very large. This means that while constants may affect the performance for small input sizes, they become negligible when considering how the algorithm behaves as N approaches infinity.</p> <p>For example, in the context of the algorithms discussed in the lectures, we can compare the functions $N^2 + 100$ and N^2. As N increases, the term +100 becomes insignificant relative to N^2. Thus, both functions are treated the same in terms of their growth rate, which is quadratic.</p> <p>This concept is illustrated in the transcript where it mentions that when the problem size increases significantly, the constant (like +100) is negligible compared to the dominant term (N^2). Specifically, it states, "the constant is not important when your problem size increases to such a large number. The +100 is negligible" (Covered at 55:44).</p> <p>Therefore, in asymptotic notation, both $N^2 + 100$ and N^2 would be classified as $O(N^2)$, indicating they have the same order of growth despite the constant difference.</p>			
9	<p>Across the Sc1007_Videolecture and Lecture2_videolecture videos, which graph-related topics are planned and what real-world path problem is used as an example?</p>	0.166667	0	0.777778	0.927559
		<p>In the Sc1007_videolecture, graph-related topics include the introduction to graph problems, specifically the traveling salesman problem. This problem is described as a scenario where you have a set of cities to visit, and you want to find the shortest path that allows you to visit each city once and return to the starting city. The complexity of this problem arises from the need to try all possible combinations to determine the best solution. This is covered at "Covered at 38:08".</p>			

		<p>Additionally, the lecture mentions that graph problems are significant in mathematics and computer science, indicating their importance in various applications. The traveling salesman problem serves as a classic example of a real-world path problem in this context.</p> <p>If you need more details about other graph-related topics or specific applications, please let me know!</p>			
10	What guidance is given about coding vs. concepts across the Sc1007_Videolecture and Lecture2_videolecture videos, and how should students prepare?	1	0.666667	0.846154	0.886732
11		0.333333	0.5	0.833333	0.953858

	Which graph-related problems are planned in Sc1007_Videolecture , and what real-world application is used in Lecture5_videoleecture?	<p>In the Sc1007_Videolecture, several graph-related problems are planned to be covered, including:</p> <ol style="list-style-type: none"> Traversal Problems: The lecture will introduce two main methods of traversing graphs: breadth-first search and depth-first search. These methods are essential for systematically exploring graph structures. This is mentioned as being part of the next lecture on Wednesday, where the instructor will discuss these traversal techniques in detail. Covered at [57:07]. The Traveling Salesman Problem: This is a well-known graph problem where the objective is to find the shortest possible route that visits a set of cities and returns to the origin city. The complexity of this problem arises from the need to evaluate all possible combinations to determine the best solution. Covered at [38:08]. Minimum Vertex Cover Problem: This problem involves finding the minimum number of checkpoints needed at junctions in a traffic network to ensure that all cars pass through these checkpoints. This is a practical application of graph theory in traffic management. Covered at [42:57]. <p>In terms of real-world applications discussed in Lecture5_videoleecture, the instructor mentions the use of graph algorithms in computer networks. Specifically, when transferring data across a network, routers and switches utilize graph algorithms to determine the best path for data packets to reach their destination. This application highlights the importance of graph theory in optimizing network communications. Covered at [28:31].</p>			
12	Which lecture between Sc1007_Videolecture and Lecture3_videoleecture previews asymptotic notation, and which defines Big O, Big Omega, and Big Theta in detail?	0.876667	0	1	0.921472
		<p>The lecture that previews asymptotic notation is Lecture3_Sc1007. It introduces the three notations: big O, big Theta, and big Omega at Covered at 11:24.</p> <p>In terms of defining Big O, Big Omega, and Big Theta in detail, this is also covered in Lecture3_Sc1007. The definitions and explanations of these notations are provided, including examples of functions that fall under each</p>			

		<p>category, such as Big Omega representing functions that grow at a higher rate than G or at the same rate as G, which is discussed at Covered at 13:24.</p> <p>Thus, both the preview and detailed definitions of asymptotic notation are found in Lecture3_Sc1007.</p>			
13	How is search complexity contrasted in Lecture3_videolecture sequential search and Lecture4_videolecture binary search?	0.691667	1	1	0.913368
		<p>In the context of search complexity discussed in the lectures, the sequential search and binary search have distinct characteristics:</p> <p>1. Sequential Search:</p> <ul style="list-style-type: none"> - The time complexity for sequential search is linear. In the average case, it is $\Theta(N)$, meaning that the number of comparisons grows linearly with the number of items in the list. This is because you may need to check every item in the list to find the key, especially if the data is not sorted. The worst-case scenario also results in $O(N)$ comparisons, where N is the number of items. This is highlighted in the lecture at (50:45.08) and (52:27.16). <p>2. Binary Search:</p> <ul style="list-style-type: none"> - In contrast, binary search operates on sorted data and has a significantly better time complexity. The average case for binary search is $O(\log N)$, which means that the number of comparisons grows logarithmically with the number of items. This efficiency comes from the method of dividing the search space in half with each comparison. The specific details about binary search were mentioned in the context of its time complexity, although the exact timestamp for this in the provided context is not specified. <p>In summary, sequential search has a linear time complexity ($\Theta(N)$), while binary search has a logarithmic time complexity ($O(\log N)$), making binary search much more efficient for large datasets when the data is sorted.</p>			
14	What assessments are mentioned in Sc1007_Videolecture and Lecture2_videolecturevideos	0.876667	0.25	0.866667	0
		<p>In the Sc1007_videolecture, the assessments mentioned include assignments, lab tests, and a final quiz. The grading breakdown is as follows:</p>			

	and how do they count toward the final grade?	<p>- Assignments contribute 40% to the final grade, with each instructor's part accounting for 20% (Covered at 07:30).</p> <p>- There are two lab tests, each contributing 20% to the final grade, making a total of 40% from lab tests (Covered at 07:54).</p> <p>- The final quiz also contributes 20% to the overall grade (Covered at 07:54).</p> <p>Additionally, attendance for the tests is compulsory, and failing to attend will result in receiving zero marks for that portion of the grade (Covered at 08:06).</p> <p>If you have specific information from the Lecture2_videolecture that you would like to know about, please provide the context or details, as I currently do not have that information.</p>			
15	Why are constants ignored in asymptotic analysis, and what example is provided across the Sc1007_Videolecture and Lecture3_videolecture videos?	1	1	1	0.93379
		<p>In asymptotic analysis, constants are ignored because they do not significantly affect the growth rate of functions as the input size becomes very large. The focus is on how functions behave in relation to each other as they approach infinity, rather than their exact values at specific points. This allows for a simplified comparison of algorithms based on their efficiency.</p> <p>An example provided in the video is the comparison of functions where $(F_N = 2N)$, $(G_N = N^2)$, and $(H_N = N^3)$. Here, (F_N) is in $(O(G_N))$ and (G_N) is in $(O(H_N))$, leading to the conclusion that (F_N) is also in $(O(H_N))$. This illustrates that as (N) grows, the constants (like the coefficient 2 in (F_N)) become negligible compared to the growth rates of the polynomial terms (N^2) and (N^3) (Covered at [57:04]).</p> <p>Additionally, when discussing the overall time complexity of two algorithms, if one is linear and the other is logarithmic, the linear algorithm dominates, and the logarithmic growth is considered negligible (Covered at [57:51]).</p>			
16	How are linked lists used differently in Lecture4_videolecture hash tables and	0.81	0.5	1	0.929144
		In Lecture 4, linked lists are used in the context of hash tables through a method known as closed addressing hashing, or separate chaining. In this approach, when multiple keys hash to the same index, they are stored in a linked list at that index.			

	Lecture5_videolecture adjacency lists?	<p>The hash table only stores the head pointer of the linked list, and when searching for a key, the search proceeds along the linked list to find the key. This method allows for efficient handling of collisions in the hash table. This is covered at (31:40.56) to (33:19.80).</p> <p>In contrast, in Lecture 5, linked lists are utilized to represent the adjacency relationships between vertices in a graph. Each vertex has its own linked list that contains all the vertices it is connected to. For example, vertex 1 might have a linked list containing vertices 2, 3, and 4. This representation allows for efficient storage of graph connections, where the space complexity is lower compared to an adjacency matrix. The search time to check if two vertices are connected is linear, as it requires traversing the linked list for the vertex in question. This is discussed at (38:12.20) to (39:41.32).</p> <p>In summary, while both lectures use linked lists to manage collections of data, Lecture 4 focuses on resolving collisions in hash tables, and Lecture 5 focuses on representing connections in graph structures.</p>			
17	How is efficiency of summing 1..N compared in both Sc1007_Videolecture and Lecture3_videolecture ?	0.876667	0.5	0.703704	0.914529
		<p>In the context of summing numbers from 1 to N, the efficiency of different algorithms is discussed in both the Sc1007_videolecture and the Lecture3_videolecture.</p> <p>In the Sc1007_videolecture, three methods are presented for summing numbers from 1 to N:</p> <ol style="list-style-type: none"> Iterative Method: A for loop is used to sum the numbers, which involves N iterations. This method is straightforward but can be less efficient for large N due to the number of iterations required. The efficiency is described as growing larger with more iterations, making it less optimal compared to other methods. Covered at [24:46]. Formula Method: The formula $\frac{N}{2} \times (1 + N)$ is mentioned as a one-liner that provides the answer instantly. This method is the most efficient because it requires only a few arithmetic operations (one 			

		<p>multiplication, one addition, and one division). Covered at [24:39].</p> <p>3. Recursive Method: This method involves recursively calling the function until reaching the base case ($N = 1$). While it is a valid approach, it is less efficient due to the increasing number of recursive calls as N grows. Covered at [24:59].</p> <p>In the Lecture3_videolecture, the discussion focuses on the time complexity of algorithms. It highlights that the recursive approach can lead to exponential time complexity, approximately $\mathcal{O}(1.6^N)$ or even $\mathcal{O}(2^N)$, which is significantly slower than the iterative method. The iterative approach runs in linear time, requiring N iterations, which is more efficient than the recursive method for larger values of N. Covered at [30:32].</p> <p>Overall, the formula method is the most efficient, followed by the iterative method, while the recursive method is the least efficient due to its exponential time complexity.</p>			
18	What graph representations are mentioned across Lecture3_videolecture and Lecture5_videolecture?	0.854167	0.5	0.75	0.96516
		<p>In the context of the Lecture5 video, two graph representations are specifically mentioned: the adjacency matrix and the adjacency list. These representations are introduced as methods to represent graphs, with the lecturer stating, "We will introduce two representations: one is the adjacency matrix, and the other is the adjacency list" (Covered at 09:35).</p> <p>Unfortunately, I do not have access to the content of Lecture3_videolecture, so I cannot provide information on the graph representations mentioned in that lecture. If you have specific details or timestamps from Lecture3, I would be happy to help analyze or compare them!</p>			
19	What real-world applications of graphs are given across Sc1007_videolecture and Lecture5_videolecture?	0.2	0.5	0.769231	0.984376
		<p>In the Sc1007_videolecture and Lecture5_videolecture, several real-world applications of graphs are discussed:</p> <p>1. Traveling Salesman Problem: This is a well-known graph problem where</p>			

		<p>you have a set of cities and want to visit all of them once, eventually returning to the first city. The challenge is to find the best solution among many possible routes. This problem illustrates the complexity of finding the shortest path in a network. Covered at [38:04].</p> <p>2. Cryptography: Graphs are also applied in cryptography, particularly in encoding and decoding messages. The need for unique solutions in encoding ensures that the transmitted message can be decoded correctly without ambiguity. This highlights the importance of graph theory in secure communication. Covered at [39:13].</p> <p>These examples demonstrate how graph theory is integral to solving complex problems in various fields, including logistics and information security.</p>			
20	What is the average-case complexity of search mentioned in Lecture3_videolecture and Lecture4_videolecture?	1	0	0.857143	0.957042
		<p>The average-case complexity of the sequential search mentioned in both Lecture 3 and Lecture 4 is $\Theta(N)$. This means that, on average, the search will require checking about half of the elements in the array or linked list, resulting in a linear time complexity. This information is covered in Lecture 4 at the timestamp "Covered at 05:37.04".</p> <p>In Lecture 3, the same concept is implied, although the specific details about average-case complexity are not explicitly stated in the provided transcript. However, it aligns with the discussion in Lecture 4.</p>			
21	What guidance is given about tutorials, labs, and coding across Sc1007_videolecture and Lecture2_videolecture ?	1	1	1	0.935147
		<p>In the Sc1007_videolecture, several key points are made regarding tutorials, labs, and coding:</p> <p>1. Tutorials: The tutorials are designed to help students clarify doubts they may have encountered during lab sessions. There is an opportunity to meet biweekly for further clarification, and students are encouraged to reach out via email if they still have questions after these sessions. This is emphasized at the timestamp "Covered at 13:58".</p>			

		<p>2. **Labs**: Attendance in labs is important, although the lab itself is not graded. However, not attending may negatively impact performance in the lab tests. This is highlighted at "Covered at 08:17". Additionally, lab tests will assess the knowledge gained in labs, and attendance will be taken, but it does not guarantee a good performance in tests if labs are skipped ("Covered at 08:34").</p> <p>3. **Coding**: The emphasis is placed on learning how to code rather than just reading solutions. Students are encouraged to implement solutions themselves to truly understand the material. This guidance is provided at "Covered at 01:00:53". The importance of understanding time complexity and space complexity in coding is also mentioned, particularly in relation to algorithm analysis ("Covered at 57:39").</p> <p>Overall, the guidance stresses the importance of active participation in tutorials and labs, as well as the necessity of hands-on coding practice to grasp the concepts effectively.</p>				
22	How is graph connectivity defined in Lecture5_videolecture compared to complexity bounds in Lecture3_videolecture ?	0.666667	0.5	1	0.830355	<p>In Lecture 5, graph connectivity is defined in terms of paths between vertices. An undirected graph is considered connected if there is a path from any vertex to any other vertex. Similarly, a directed graph is termed strongly connected if there is a path from any vertex to any other vertex as well. This concept is emphasized when the lecturer states, "You need to check all the vertices; any vertex can reach any other vertex" (Covered at 21:48). The importance of checking every vertex to ensure connectivity is reiterated throughout the lecture.</p> <p>In contrast, Lecture 3 focuses on complexity bounds, particularly using Big-Oh notation to describe the growth rates of functions. The lecturer explains that for a polynomial of degree K, it is represented as big Theta N to the K, indicating how the function behaves as N approaches infinity (Covered at 01:12:27). The discussion revolves around how to determine the upper bounds of functions and the significance of constants in these definitions.</p> <p>In summary, while Lecture 5 deals with the concept of connectivity in graphs,</p>

		Lecture 3 addresses the analysis of algorithmic complexity through bounds, highlighting different aspects of graph theory and computational analysis.			
23	How do adjacency lists relate to hash tables as explained across Lecture4_videolecture and Lecture5_videolecture?	0.876667	0	1	0.948025
		<p>In the context of the lectures, adjacency lists and hash tables serve as data structures for storing relationships and data efficiently.</p> <p>1. **Adjacency Lists**: In Lecture 5, Dr. Loke Yuan Ren explains that adjacency lists are used to represent graphs by storing the connections between vertices. Instead of needing to know the entire graph, you only need to know which vertices are adjacent to each other. This allows for efficient traversal of the graph without constructing the entire structure. The adjacency list stores these relationships, making it easier to visit nodes in a specific order during graph traversal (Covered at 32:20).</p> <p>2. **Hash Tables**: In Lecture 4, the focus is on hash tables, which map data of arbitrary size to fixed-size values using a hash function. The lecture discusses the concept of collisions, where two different data items may map to the same index in the hash table, and how to resolve these collisions. Hash tables allow for efficient data retrieval, with an average search time of $O(1)$ (Covered at 14:19).</p> <p>Both data structures emphasize efficiency in storing and retrieving data. While adjacency lists are specifically tailored for graph representation, hash tables provide a general-purpose method for data storage and retrieval. The underlying principle in both cases is to optimize access to data, whether it be through adjacency relationships in graphs or key-value pairs in hash tables.</p>			
24	What are the main considerations when analyzing if/else in Lecture2_videolecture versus collisions in Lecture4_videolecture?	0	0	0.714286	0.847626
		<p>In Lecture 2, the focus is on the analysis of algorithms, particularly time complexity and space complexity, which are essential when evaluating conditional statements like if/else. The analysis involves understanding how these statements affect the overall efficiency of an algorithm, including the order of growth and asymptotic notation. This is crucial for determining how the performance of an algorithm scales with input size.</p>			

		<p>In contrast, Lecture 4 discusses hashing and the specific issue of collisions in hash tables. The main considerations here include how to handle situations where multiple data items map to the same index in a hash table, which is known as a collision. The lecture emphasizes the need to resolve these collisions effectively to maintain efficient data retrieval and storage. Key points include the design of the hash function and the implications of the load factor on the performance of the hash table.</p> <p>To summarize, while Lecture 2 focuses on the theoretical analysis of algorithm efficiency, including if/else statements, Lecture 4 deals with practical challenges in data structures, specifically the resolution of collisions in hash tables.</p>			
25	Which parts of the module are described as concept-heavy, and how do labs balance this across Sc1007_videolecture and Lecture2_videolecture ?	1	0.5	0.944444	0.842851
26		0.608333	1	1	0

	<p>How are nested loops explained in Lecture2_videolecture and Lecture3_videolecture , and what time complexity do they lead to?</p>	<p>In the Lecture2 video, nested loops are explained in the context of time complexity. The lecturer discusses how a nested loop structure can significantly increase the time complexity of an algorithm. For example, when using two nested for loops, the time complexity becomes worse compared to a single loop. Specifically, if the outer loop runs from 1 to N and the inner loop runs I squared iterations, the total number of operations is calculated as 1 squared + 2 squared + 3 squared + N squared, leading to a time complexity of $O(N^3)$ for this particular example. This is covered at [51:43.80].</p> <p>Unfortunately, I do not have access to the content of Lecture3, so I cannot provide information on how nested loops are explained there or the time complexity they lead to. If you have specific details or timestamps from Lecture3, I can help analyze that information.</p>				
27	<p>What is the difference between an algorithm and a program, and how is efficiency evaluated across Sc1007_Videolecture and Lecture2_videolecture ?</p>	<table><tr><td>1</td><td>0.5</td><td>1</td><td>0.931816</td></tr></table> <p>The difference between an algorithm and a program is primarily in their definitions and roles in problem-solving. An algorithm is a well-defined step-by-step procedure or a set of rules to solve a problem, while a program is a concrete implementation of that algorithm in a specific programming language. This distinction is highlighted when the speaker mentions that "a computer program is an instance, or concrete representation of an algorithm in some programming languages" and that "an algorithm is just a procedure; you can think of it as an idea to solve the problem" (Covered at 21:27.88 and 22:10.68).</p> <p>Regarding the evaluation of efficiency, the analysis of algorithms focuses on understanding how the efficiency of an algorithm changes as the input size grows. This includes evaluating both time complexity, which measures the time taken to run the algorithm, and space complexity, which refers to the memory used (Covered at 04:41.84 and 04:44.92). The speaker emphasizes the importance of analyzing these complexities to identify the most efficient algorithm for a given problem and to understand the trade-offs between different approaches (Covered at 04:33.48).</p> <p>In summary, algorithms are abstract procedures for solving problems, while programs are their specific implementations. Efficiency is evaluated through time</p>	1	0.5	1	0.931816
1	0.5	1	0.931816			

		and space complexities, which help determine the best algorithm to use based on resource constraints.			
28	In Lecture2_videolecture and Lecture4_videolecture, what considerations are made for selection structures and hash table collisions?	1	1	1	0.946334
		<p>In the context of selection structures and hash table collisions discussed in the lectures:</p> <p>1. **Selection Structures**: In Lecture 2, the analysis of selection structures (like if/else statements and switch-case statements) involves counting the number of primitive operations performed. The time complexity is expressed in terms of problem size, which relates to how the performance of the algorithm scales with increasing input size. This is particularly important for recursive functions, which can be more complex to analyze due to the number of function calls made during execution. The time complexity is defined by the total number of primitive operations across all recursive calls (Covered at 11:30).</p> <p>2. **Hash Table Collisions**: In Lecture 4, when discussing hash tables, it is noted that collisions occur when two or more data items map to the same location in the hash table. This situation necessitates collision resolution strategies. The lecture introduces two main methods: closed addressing hashing (also known as separate chaining), where all keys that map to the same location are stored in a linked list, and open addressing hashing, which involves probing for the next available slot. The resolution of collisions is a critical consideration in the design of hash tables (Covered at 31:40).</p> <p>These considerations highlight the importance of analyzing both the efficiency of selection structures and the strategies for managing collisions in hash tables to ensure optimal performance.</p>			
29	How does load factor in Lecture4_videolecture relate to loop growth in Lecture2_videolecture ?	0.416667	0.5	0.857143	0.889822
		<p>The load factor discussed in Lecture 4 relates to how efficiently a hash table utilizes its slots, while the concept of order of growth in Lecture 2 pertains to the performance of algorithms as input sizes increase.</p> <p>In Lecture 4, the load factor is defined as the ratio of the number of data items</p>			

		<p>(N) to the number of slots (H) in a hash table. A load factor of 1 indicates that all slots are filled, while a load factor of 0.5 means that only half of the slots are in use. This helps evaluate how well the hash table is performing and how many slots are utilized, which is crucial for maintaining efficient search times in hashing (Covered at 20:30).</p> <p>In Lecture 2, the order of growth is concerned with how the time complexity of an algorithm scales with increasing input sizes. For example, if an algorithm has a time complexity of N^2, it indicates that the time taken grows quadratically as the input size increases. The focus is on the highest order of growth, as it significantly impacts performance when the problem size becomes large (Covered at 59:15).</p> <p>Both concepts emphasize efficiency: the load factor assesses the efficiency of space utilization in hash tables, while order of growth evaluates the efficiency of time performance in algorithms as input sizes grow. Understanding both helps in designing algorithms and data structures that are optimal for their intended use cases.</p>			
30	What distinguishes trees from general graphs across the lectures Lecture2_videolecture and Lecture5_videolecture?	0.804167	0.5	0.941176	0.922933
		<p>Trees are distinguished from general graphs primarily by their structural properties, particularly the absence of cycles. In the lectures, it is emphasized that a tree is a special type of graph that does not contain any cycles. This is highlighted when it is stated that "A tree can be considered a special case. It's a special graph with no cycles" (Covered at 14:57).</p> <p>Additionally, it is mentioned that if any siblings in a tree are connected, it would form a cycle, which would disqualify it from being a tree (Covered at 15:13). In contrast, general graphs can be cyclic or acyclic, and they can have various structures, including cycles.</p> <p>Furthermore, the concept of connectivity is also relevant. A tree is inherently connected, meaning there is a path between any two vertices, while general graphs can be disconnected. This distinction is crucial in understanding the unique properties and applications of trees compared to more general graph</p>			

		<p>structures.</p> <p>In summary, the key distinctions are:</p> <ol style="list-style-type: none"> 1. Trees have no cycles (Covered at 14:57). 2. Trees are connected (Covered at 15:13). 3. General graphs can be cyclic or acyclic and may not be connected.
--	--	---

Appendix J

This appendix provides an overview of the setup process and execution flow for **VideoLearnRAG library**, the backend library powering the Lecture Companion chatbot system. The steps below outline the environment configuration, installation, and minimal code examples to help replicate or extend the system locally. For full implementation details, configuration templates, and updates, refer to the project's GitHub repository.

Prerequisites

Before running the system, ensure the following requirements are met:

- Python 3.9+
- Azure OpenAI resource and model deployment (with API key and endpoint)
- MongoDB instance (local or hosted)
- Azure Video Indexer account for video indexing

Virtual Environment Setup

To maintain isolated dependencies, create and activate a virtual environment.

- Windows (PowerShell): `python -m venv .env`
Active the environment: `.\env\Scripts\Activate.`
- macOS/Linux (bash/zsh): `python3 -m venv .env`
Active the environment: `source .env/bin/activate`

Installing the Library

Install the library in editable mode to allow local development:

- `pip install -e backend/src/videoLearnRAG`

Environment Configuration

The application uses a .env file to store sensitive credentials.

Start with the provided template (backend/src/videoLearnRAG/.env-template) and fill in the following keys:

```
AZURE_OPENAI_ENDPOINT=  
AZURE_OPENAI_API_KEY=  
YOUR_DEPLOYMENT_NAME=  
OPENAI_API_VERSION=  
MONGODB_CONNECTION_STRING=  
DB_NAME=  
SECRET_KEY=  
  
AZURE_VIDEO_INDEXER_ACCOUNT_ID=  
AZURE_VIDEO_INDEXER_LOCATION=  
AZURE_VIDEO_INDEXER_API_KEY=
```

Minimal Usage Examples

Video Processing (BrokerService)

Pipeline stages:

1. Enter course details to create a course
2. Enter the video details to upload (course_code ,video name, video_description and base64 of video)
3. Validate Course ID existence. If not valid Course ID, raise an exception. No videos will be processed.
4. Registering the video in the database.
5. Sending the video for indexing to Video Indexer.
6. Fetching insights and associating transcripts.
7. Chunking and Transcript cleaning
8. Update the clean transcript into the database
9. Change the indexing status to complete in the DB

```
from videoLearnRAG.brokerservice.brokerService import BrokerService  
from videoLearnRAG.videoindexerclient.model import Video, VideoList
```

```

broker = BrokerService()

broker.add_course('SC1007', 'Intro to Data Structure and Algorithm',
'Fundamentals of computer science')

videos = VideoList(course_code='SC1007', video=[
    Video(video_name='Lecture 1', video_description='Overview',
    base64_encoded_video='<base64>')
])

broker.start_video_index_process(videos)

```

- Chat Generation (ChatService)

Pipeline Stages:

1. Enter query, video_ids for context, course_code
2. Query is pass through temporal + docs scope module
3. Retrieve chunks from DB
4. Pass chunks and query to LLM
5. Get a response

```

import asyncio
from videoLearnRAG.chatservice.chatservice import ChatService

async def main():
    chat = ChatService()
    results, context = await chat.query_evaluation(
        'What is mentioned at the 5-minute mark of each video?',
        ['zwb6lqhpzl', 'tg2fy923h1'],
        'SC1007'
    )
    print(results, context)

```

```
asyncio.run(main())
```