Sri Lanka Institute of Information Technology

Faculty of Computing

Department of Software Engineering

Year 4 Semester 2

SE4010 – Current Trends in Software Engineering

AI/ML Assignment

2025 May

IT21321368 – Samarasinghe.V.R

TABLE OF CONTENTS

1. Introduction 3
2. Justification of LLM Choice 4
3. Justification of Development Approach 5
4. System Diagram 6
5. Challenges and Lessons Learned 6
6. Use of GenAI Tools (Prompts and Outputs) 8
7. Conclusion 11
8. References 12
9. Jupyter Notebook 12
10. Video Demonstration 12

1. Introduction

In recent years, Large Language Models (LLMs) have revolutionized the way information is retrieved, processed, and presented in educational environments. As part of the SE4010 – Current Trends in Software Engineering course, this assignment focuses on applying LLMs to build a domain-specific chatbot capable of answering questions based on course lecture content. The main objective was to create a CTSE Lecture Notes Chatbot that enhances learning by offering an interactive, citation-based question-answering system derived from PDF lecture notes.

This chatbot is implemented within a Jupyter Notebook and utilizes the LangChain framework, which provides tools to integrate LLMs with external data sources. The system is designed around the Retrieval-Augmented Generation (RAG) approach, ensuring that answers are not generated from the model's general knowledge but are grounded strictly in the provided lecture material. This mitigates the risk of hallucinated content and improves reliability in academic contexts.

Google's Gemini 2.0 Flash Lite model was chosen as the core LLM due to its lightweight nature, fast response time, and compatibility with LangChain. Supporting technologies include PyPDFLoader for PDF document ingestion, RecursiveCharacterTextSplitter for content segmentation, Chroma for vector-based document retrieval, and GoogleGenerativeAIEmbeddings for generating semantic embeddings. Together, these tools allow the chatbot to retrieve relevant chunks of lecture content, generate responses, and cite source pages accurately.

This report documents the reasoning behind the technical choices made, challenges encountered during development, the role of generative AI prompts, and the educational value of the final system.

2. Justification of LLM Choice

For this chatbot implementation, Google's Gemini 2.0 Flash Lite model was selected as the underlying large language model. This decision was based on several key considerations:

1. **Performance to Cost Ratio**: Gemini 2.0 Flash Lite offers strong performance capabilities while being cost-effective for educational projects, making it suitable for this assignment's scope and budget constraints.
2. **Contextual Understanding**: The model demonstrates good comprehension of academic content, particularly technical subject matter relevant to software engineering concepts.
3. **Response Quality**: Gemini 2.0 Flash Lite generates coherent, well-structured responses appropriate for educational contexts, maintaining appropriate formality for academic content.
4. **API Accessibility**: Google provides reliable API access through their generative AI services, with straightforward implementation in Python using the langchain-google-genai library.
5. **Temperature Control**: The implementation uses a temperature setting of 0.2, which balances creativity with factual consistency - crucial for educational applications where accuracy is paramount.
6. **Integration Compatibility**: Gemini models work seamlessly with Google's embedding models (models/embedding-001), providing consistency in the RAG pipeline from embeddings to generation.

While other models like OpenAI's GPT models could have provided similar functionality, Gemini 2.0 Flash Lite offered the best balance of capabilities, cost, and integration ease for this specific educational application.

3. Justification of Development Approach

The chatbot implements a Retrieval-Augmented Generation (RAG) architecture, which was chosen for specific pedagogical and technical advantages:

1. **Knowledge Grounding**: RAG ensures answers are grounded in the actual lecture content rather than the model's pre-trained knowledge, preventing hallucinations or incorrect information.
2. **Source Citations**: The implementation automatically provides page citations ([Page: X]) for information, enhancing academic credibility and helping students locate relevant material in the notes.
3. **Chunking Strategy**: The lecture notes are processed using RecursiveCharacterTextSplitter with a chunk size of 1000 characters and 100-character overlap. This granularity provides:
   - Sufficient context for the model to understand concepts
   - Precise enough retrieval to locate specific information
   - Overlap to prevent loss of context across section boundaries
4. **Vector Database Implementation**: Chromadb was selected as the vector store due to:
   - Persistence capabilities for saving embeddings between sessions
   - Efficient similarity search functionality
   - Compatibility with Google's embedding model
5. **Top-k Retrieval**: The retriever is configured to fetch the top 4 most relevant chunks for each query, providing sufficient context while limiting noise.
6. **Prompt Engineering**: A specialized prompt template instructs the model to:
   - Act as a teaching assistant for CTSE specifically
   - Base answers exclusively on provided context
   - Acknowledge knowledge gaps rather than fabricating information
   - Include source citations for traceability

This approach creates a self-contained knowledge system that accurately represents course content while maintaining academic rigor through proper attribution.

4. System Diagram

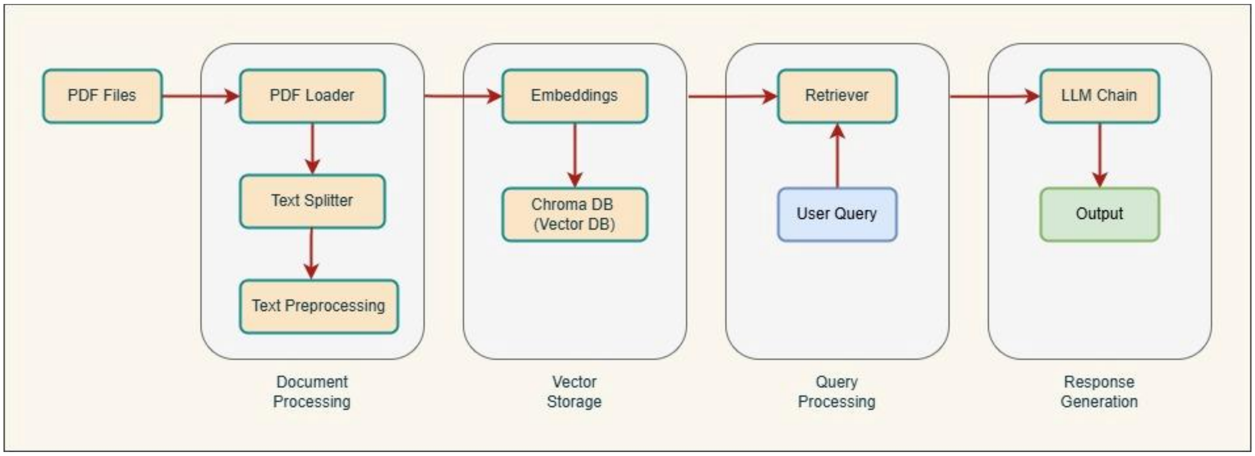


Figure 1 - System Diagram

5. Challenges and Lessons Learned

1. Model Selection and API Compatibility

Challenge: Initial integration attempts with gemini-pro resulted in 404 errors due to API version and model compatibility issues.

Lesson: The successful implementation uses gemini-2.0-flash-lite, demonstrating the importance of selecting appropriate model versions that are consistently available in the current API.

2. Resource Quota Management

Challenge: Encountered ResourceExhausted: 429 errors when exceeding API quota limits.

Lesson: The working code implements focused queries with controlled parameters (temperature=0.2, retrieving only top 4 chunks) to optimize API usage and stay within quota limits.

3. Robust Error Handling

Challenge: Early implementations lacked comprehensive error handling.

Lesson: The final code includes proper try-except blocks with user-friendly error messages, particularly in the ask() method to gracefully handle API failures.

4. Document Processing Optimization

Challenge: Large documents needed efficient processing to avoid excessive token usage.
Lesson: Implemented chunk size of 1000 with 100 token overlap, creating appropriate context windows that balance comprehensive information with efficient API usage.

5. Structured Response Processing

Challenge: Raw API responses required post-processing for user readability.

Lesson: The implemented solution extracts and highlights page sources with regex and provides clean, formatted answers using Markdown display.

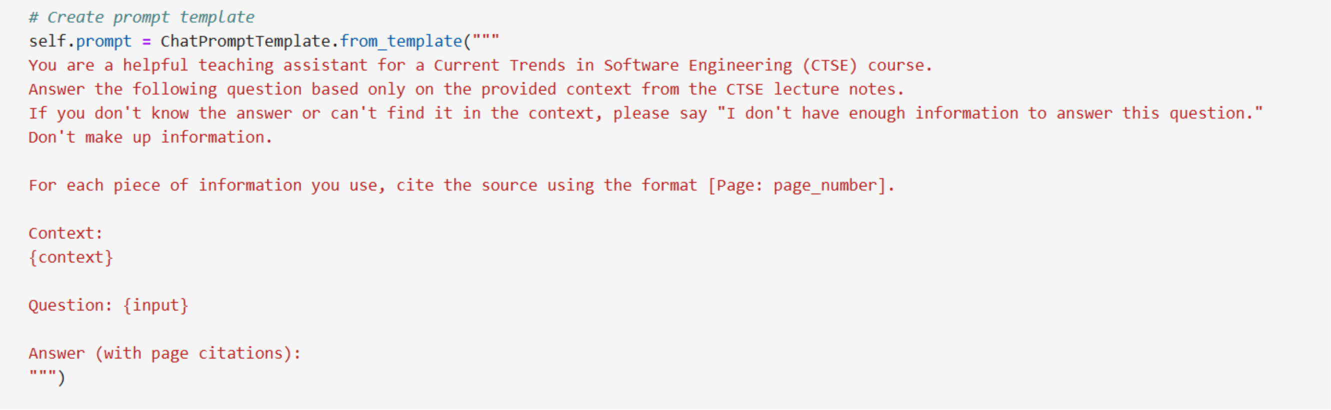
These lessons demonstrate how technical challenges with the Gemini API were systematically addressed to create a robust, user-friendly educational tool.

6. Use of GenAI Tools (Prompts and Outputs)

Generative AI (GenAI) tools played a central role in the development of the CTSE Lecture Notes Chatbot. Specifically, Google's Gemini 2.0 Flash Lite model was used in conjunction with the LangChain framework to generate accurate, context-aware answers to student questions. The chatbot follows a Retrieval-Augmented Generation (RAG) approach, meaning the LLM does not answer from memory but is instead guided by retrieved content from the uploaded CTSE lecture notes.

To ensure reliable and factual answers, a custom prompt was designed. This prompt restricts the model to only answer questions based on the context it receives and to clearly cite the source using the format [Page: number]. This reduces hallucinations and maintains academic transparency.

Prompt Template Used:

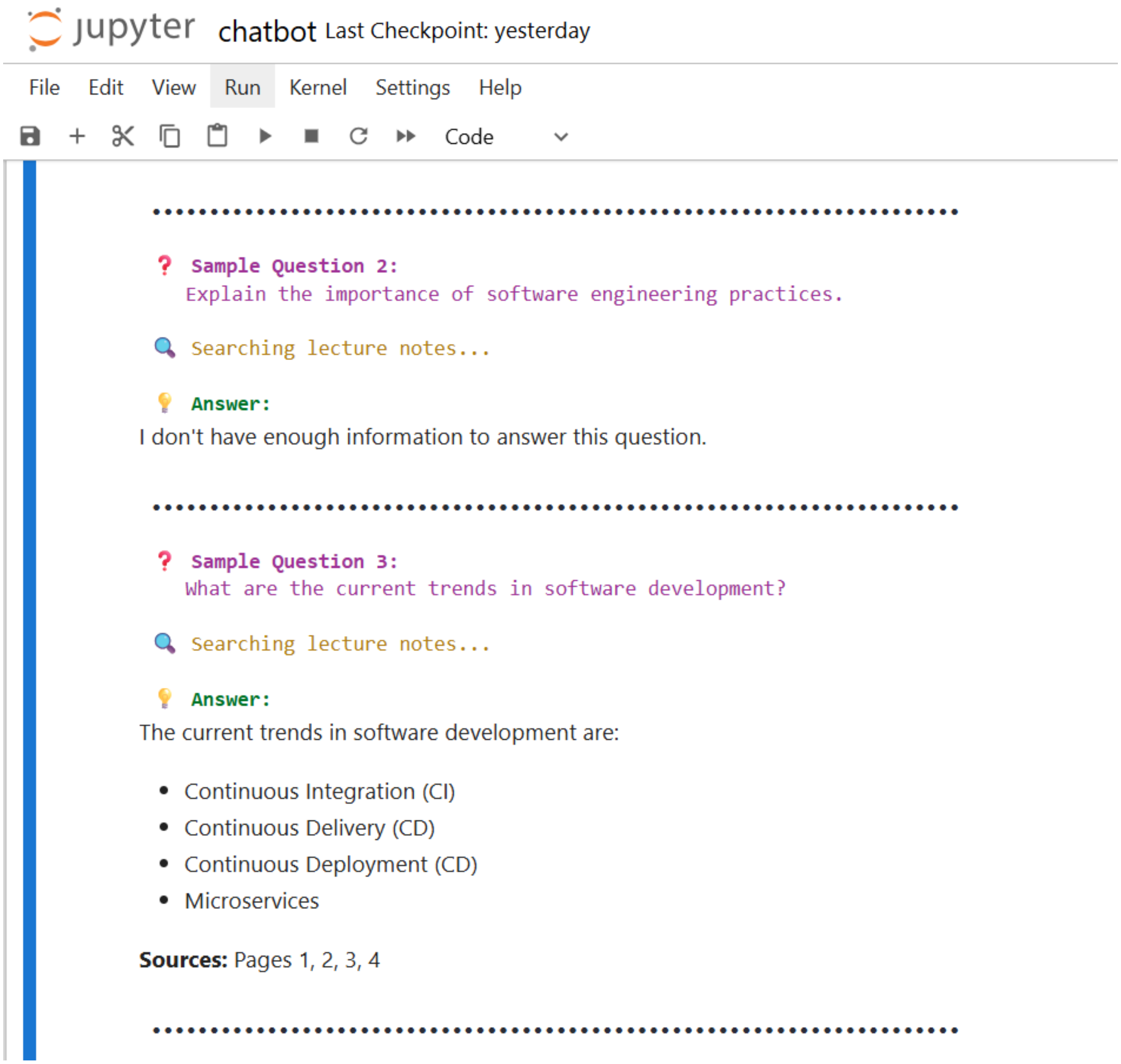


```
# Create prompt template
self.prompt = ChatPromptTemplate.from_template("""
You are a helpful teaching assistant for a Current Trends in Software Engineering (CTSE) course.
Answer the following question based only on the provided context from the CTSE lecture notes.
If you don't know the answer or can't find it in the context, please say "I don't have enough information to answer this question."
Don't make up information.

For each piece of information you use, cite the source using the format [Page: page_number].

Context:
{context}

Question: {input}

Answer (with page citations):
""")
```

Figure 2 - Prompt template

This prompt was embedded within a LangChain chain using create_stuff_documents_chain() to generate well-structured, citation-based responses.

Example Question and Output



••

```
❓ Sample Question 2:
  Explain the importance of software engineering practices.

🔍 Searching lecture notes...

💡 Answer:
```

I don't have enough information to answer this question.

••

```
❓ Sample Question 3:
  What are the current trends in software development?

🔍 Searching lecture notes...

💡 Answer:
```

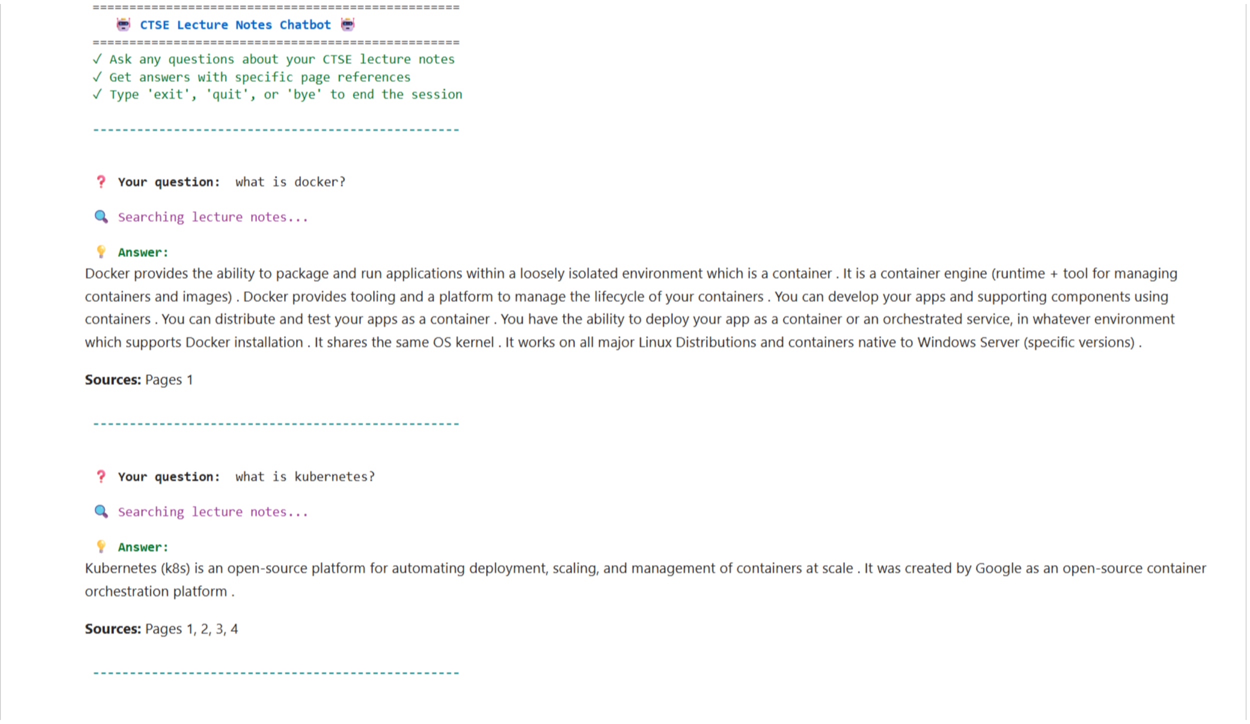
The current trends in software development are:

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Continuous Deployment (CD)
- Microservices

Sources: Pages 1, 2, 3, 4

••

Figure 3 - Sample Questions



```
==================================================
   🤖 CTSE Lecture Notes Chatbot 🤖
==================================================
✓ Ask any questions about your CTSE lecture notes
✓ Get answers with specific page references
✓ Type 'exit', 'quit', or 'bye' to end the session

--------------------------------------------------


❓ Your question:  what is docker?

🔍 Searching lecture notes...

💡 Answer:
Docker provides the ability to package and run applications within a loosely isolated environment which is a container . It is a container engine (runtime + tool for managing
containers and images) . Docker provides tooling and a platform to manage the lifecycle of your containers . You can develop your apps and supporting components using
containers . You can distribute and test your apps as a container . You have the ability to deploy your app as a container or an orchestrated service, in whatever environment
which supports Docker installation . It shares the same OS kernel . It works on all major Linux Distributions and containers native to Windows Server (specific versions) .

Sources: Pages 1

--------------------------------------------------


❓ Your question:  what is kubernetes?

🔍 Searching lecture notes...

💡 Answer:
Kubernetes (k8s) is an open-source platform for automating deployment, scaling, and management of containers at scale . It was created by Google as an open-source container
orchestration platform .

Sources: Pages 1, 2, 3, 4

--------------------------------------------------
```

Figure 4 - Asked Questions

7. Conclusion

The CTSE Lecture Notes Chatbot successfully demonstrates the application of RAG techniques for educational purposes. By combining Google's Gemini 2.0 Flash Lite model with carefully engineered retrieval mechanisms, the system provides accurate, cited responses to queries about course content.

The implementation highlights how modern LLM technologies can enhance educational experiences by providing accessible, accurate information retrieval from course materials. The chatbot not only answers questions but also guides students to relevant sections in their lecture notes through consistent page citations.

Future enhancements could include a graphical user interface, multi-document support for additional course materials, and integration with learning management systems. The current implementation provides a solid foundation that could be extended for broader educational applications.

8. References

1. LangChain Documentation. (2024). Retrieval Augmented Generation (RAG). https://python.langchain.com/docs/tutorials/rag/
2. Google Generative AI Python SDK. (2024). https://github.com/google/generative-ai-python
3. ChromaDB Documentation. (2024). Building RAG applications with Chroma. https://docs.trychroma.com/
4. Recursive Character Text Splitter Documentation. (2024). LangChain Text Splitting. https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/recursive_text_splitter

9. Jupyter Notebook

Jupyter notebook code drive link - https://drive.google.com/file/d/1ZkSpIBbJLcFXmvbXfvCXMP6S2CKTubhC/view?usp=drive_link

10. Video Demonstration

Demo Video Link - https://drive.google.com/file/d/1fOuJUq7KC71K25wbU4o9emjEre_QxgjS/view?usp=drive_link