Course: ITAI-2277 — Artificial Intelligence Resource

Project Title: *AI Course Copilot – RAG-Powered Study & Project Assistant*

Team: Oscar Cortez · Judith Barrios · Ruben Valenzuela Alvarado · Darnell Newman

GitHub Repository (Phase 05):

https://github.com/oscarecortez361/ai-course-copilot-phase03/tree/main/phase05

Phase 05- Technical report

1. Executive Summary

This technical report documents the design, implementation, and final delivery of the AI Course Copilot, a Retrieval-Augmented Generation (RAG) prototype developed as the capstone project for ITAI-2277. The goal of the system is to help students interact with approved course materials by asking natural language questions and receiving grounded, citation-based answers.

The project was completed in five phases: project proposal, data pipeline and exploratory analysis, model development and evaluation, integrated system prototype, and final project delivery. By the end of Phase 05, the team delivered a fully working prototype in Google Colab with a Gradio interface, a vector-based retrieval backend using FAISS, preprocessed ITAI-2277 documents, and supporting documentation including a user guide and this technical report.

2. Introduction

In modern AI and data science courses, students must navigate large volumes of course materials: syllabi, assignment descriptions, grading rubrics, policies, and project guidelines. Finding a specific piece of information often requires manually searching through multiple PDFs and web pages. This can be time-consuming and frustrating, especially close to deadlines.

The AI Course Copilot was designed to address this pain point. Instead of searching manually, students can ask questions in natural language and receive answers based only on approved course materials. The system follows the Retrieval-Augmented Generation (RAG) paradigm, combining dense retrieval with a large language model (LLM) to generate accurate, context-grounded responses with explicit citations.

3. Problem Statement and Objectives

The problem addressed by this project can be summarized as follows:

- Students struggle to quickly locate relevant information across multiple course documents.
- There is a risk of misunderstanding grading policies, assignment requirements, or deadlines.
- Generic LLMs may hallucinate or provide incorrect answers that are not aligned with the syllabus.

The main objectives of the AI Course Copilot are:

1. Provide a single interface where students can ask course-related questions.
2. Restrict answers to content found in instructor-approved documents.
3. Return answers with clear citations indicating the source document.
4. Serve as a prototype that can be extended to other courses or programs in the future.

4. Methodology and Project Phases

The project followed a structured, phased methodology:

• Phase 01 – Project Proposal:

The team defined the problem, proposed a RAG-based solution, and outlined the high-level architecture and milestones. Roles and responsibilities were assigned, and initial tools and technologies were selected.

• Phase 02 – Data Pipeline and Exploratory Analysis:

The team collected ITAI-2277 course materials (syllabus, assignment descriptions, grading policies, capstone details) and curated them into an allow-listed dataset. Text was extracted from PDFs and online resources, cleaned, normalized, and split into semantically coherent chunks. Basic statistics were computed to understand document length, term frequency, and coverage.

• Phase 03 – Model Development and Evaluation:

A retrieval pipeline was built using SentenceTransformers to generate embeddings and FAISS for vector search. Baseline retrieval was evaluated using simple test queries, and the team iteratively adjusted chunk sizes, top-k values, and prompt wording to improve answer quality and reduce noise.

• Phase 04 – Integrated System Prototype:

Retrieval, generation, and user interface components were integrated in a Google Colab notebook. A Gradio interface was implemented to allow interactive querying of the system. The prototype was tested with realistic student questions and refined based on observed behavior.

• Phase 05 – Final Project Delivery:

The team finalized bug fixes, cleaned the code, completed documentation, prepared a user guide, and created the final presentation and project package. This report summarizes the technical aspects of the final system.

5. System Architecture

The system follows a classic Retrieval-Augmented Generation pipeline with the following stages:

1. User Query Input:

The student types a natural language question in the Gradio interface.

2. Query Embedding:

The query is converted into a dense vector representation using a pretrained SentenceTransformer model.

3. Vector Retrieval with FAISS:

The query embedding is used to search a FAISS index built over chunked course documents. The system retrieves the top-k most similar chunks.

4. Context Assembly:

Retrieved chunks and their metadata (e.g., document title, section) are concatenated into a context window that fits within the LLM's token limit.

5. LLM Answer Generation:

A large language model is prompted with instructions to answer based only on the provided context, and to avoid external knowledge.

6. Citation Injection:

The final answer includes a list of source documents referenced in the response so that students can verify the information.

7. Response Display:

The formatted answer and citations are displayed in the Gradio UI.

6. Implementation Details

The final prototype is implemented entirely in a Google Colab notebook. Key implementation aspects include:

- Language and Libraries:

Python is used as the primary language. Key libraries include: sentence-transformers for

embeddings, faiss-cpu for vector search, pandas and numpy for data handling, and gradio for the web interface.

- Data Preparation:

Processed text chunks and metadata are loaded from files prepared in earlier phases. Each chunk contains a snippet of course text and associated labels such as document name and category (e.g., syllabus, grading policy).

- FAISS Index:

A flat FAISS index is built from the document embeddings. This is sufficient for the size of the dataset and allows for fast similarity search during demo.

- Prompt Design:

The system prompt instructs the LLM to answer only from the retrieved context, to admit when it does not know an answer, and to include citations in a consistent format.

- Gradio Interface:

The interface is configured with an input textbox, a submit button, and an output area. The backend function wraps the RAG pipeline: it receives a question, runs retrieval and generation, and returns the final answer with sources.

7. Testing and Evaluation

Since the system is a course prototype, evaluation focused on functional behavior rather than large-scale quantitative benchmarks. The team tested the system using representative queries such as:

- "How is ITAI-2277 graded?"
- "What is the final capstone project requirement?"
- "What are the attendance and make-up policies?"

For each query, the team manually verified:

- Whether the retrieved answer matched the syllabus or assignment text.
- Whether the citations pointed to the correct source documents.
- Whether the system avoided hallucinating information not present in the course materials.

Through iterative testing, the team adjusted parameters like the number of retrieved chunks and the prompt wording. The result is a prototype that consistently returns accurate, grounded answers for core course questions and gracefully indicates uncertainty when the answer is outside the dataset.

8. Limitations

The current system has several limitations:

- Scope Restricted to One Course:

  The dataset only covers ITAI-2277 materials. The system cannot answer questions about other courses or external topics.

- Limited Dataset Size:

  The number of documents is relatively small, which simplifies retrieval but may miss some edge cases.

- No Advanced Re-Ranking:

  The prototype uses a simple top-k retrieval approach without a separate cross-encoder re-ranking layer.

- Dependence on LLM Behavior:

  While prompts encourage grounded answers, the system still relies on the underlying LLM and may occasionally produce verbose or partially redundant responses.

- Colab-Based Deployment:

The prototype runs in Google Colab and is not deployed as a persistent web service.

9. Future Improvements

If the project were to be extended, several improvements could be implemented:

- Multi-Course Support:

Expanding the dataset to include materials from multiple ITAI courses and adding filters by course or module.

- Advanced Retrieval and Re-Ranking:

Integrating a hybrid retrieval approach (combining dense and keyword-based search) and adding a cross-encoder re-ranker to further improve relevance.

- Enhanced User Interface:

Providing additional UI features such as query history, saved answers, and clickable links to original documents.

- Deployment as a Web Application:

Containerizing the application and deploying it on a cloud platform so that students can access it without Colab.

- Analytics and Feedback:

Logging anonymous usage statistics and feedback to iteratively refine retrieval quality and prompt design.

10. Conclusion

The AI Course Copilot project successfully delivered a working RAG-based prototype that demonstrates how retrieval and generation can be combined to help students navigate complex course materials. Through successive phases, the team defined the problem, built a data pipeline,

implemented a retrieval model, integrated an LLM, and wrapped the system in an accessible Gradio interface.

Although the prototype is limited in scope and runs only in a development environment, it provides a solid foundation for future work. The project also gave the team practical experience with real-world AI workflows: data preparation, vector search, prompt engineering, system integration, and documentation. Overall, the AI Course Copilot meets the objectives of the capstone assignment and illustrates the potential of RAG systems in educational settings.