International Islamic University, Islamabad (IIUI)
Faculty of Computing and Information Technology (FCIT)
Department of Software Engineering
PhD-SE F24, S25, MSSE F24, S25, Advances in Artificial Intelligence
Assignment No. 1, March 9, 2025

Name: ……………………………. Registration No: ……………….. [Total Marks 40]

This assignment is due on Monday 17th March, 2025. The solution to the assignment should be submitted in soft copy on Google Classroom. Late submission will result in marks deduction.

Assignment: Build a RAG Chatbot with FAISS for Course Management

Objective:
Your task is to develop a **Retrieval-Augmented Generation (RAG) chatbot** using **Google Gemini 2.0 (experimental)** and **FAISS**. The chatbot should retrieve information from multiple **local documents stored in a folder**, instead of relying solely on the LLM's internal knowledge.

Problem Statement:

Many of you, especially those who are teachers, maintain a **folder for a course** that includes:

- **Lecture slides** (PDF, PPTX, etc.)
- **Lecture notes** (TXT, DOCX, PDF)
- **Books & reference materials** (PDF, EPUB, etc.)
- **Student attendance records** (Excel, CSV)
- **Assignments, quizzes, and semester projects**

Other students also maintain similar folders with important documents related to their coursework.

Your goal is to build a chatbot that **enables users to retrieve specific information** from these documents. Example use cases include:

- A teacher checking a **particular student's attendance record**.
- A student retrieving a **solution to a quiz**.
- Searching for **specific content from lecture slides or notes**.

Requirements:

1. Set Up Your Environment

- Install required libraries:

```
!pip install -U langchain langchain-community langchain-google-genai faiss-cpu pypdf pandas openpyxl
```

- Configure **Google Gemini API** and FAISS.

2. Load Documents from a Folder

- Your chatbot should load **all files from a specified folder**.
- Support multiple file types:
    - **PDFs (books, lecture slides, notes)** → Use pypdf
    - **TXT, DOCX (notes, assignments)** → Use TextLoader
    - **Excel, CSV (attendance records, grades)** → Use pandas

3. Chunk & Index Documents

- **Split documents into meaningful chunks** (e.g., 500 characters per chunk with overlap).
- **Convert text into vector embeddings** using HuggingFaceEmbeddings.
- **Store embeddings in FAISS** for fast retrieval.

4. Implement the RAG Chatbot

- Use **Google Gemini 2.0** for response generation.
- The chatbot should:
    - Retrieve relevant document chunks based on user queries.
    - Pass retrieved information to Gemini for response.

5. Query the Chatbot

- Implement an interactive CLI (or a simple web app) to allow users to **ask questions** about their documents.
- Example queries:
    - *"Show me the attendance record of John Doe from week 3."*
    - *"Retrieve the solution for Quiz 2."*
    - *"Find the notes on machine learning from my lecture slides."*

Deliverables:

- **Python script or Jupyter Notebook** implementing the chatbot.
- **A demo video or screenshots** showing:
    - How documents are loaded and indexed.

    - Example queries and chatbot responses.

- **Assignment report** designed according to the evaluation criteria explaining:
    - How you processed different file types.
    - How retrieval and response generation work.
    - Any challenges faced and how you overcame them.

Evaluation Criteria:

Criteria	**Description**	**Weight**
Document Processing	Ability to load and process various file types	20%
Vectorization & Indexing	Efficient chunking, embedding, and FAISS usage	20%
Chatbot Functionality	Accuracy and relevance of retrieved responses	25%
User Interaction	Ease of use and well-structured interface	15%
Report & Demo	Clarity of explanation and demonstration	20%

Bonus (Extra 10%)

- **Deploy as a simple web app** using Streamlit or Flask.
- Implement **metadata filtering** (e.g., search attendance only in Excel files).
