
# North South University

# Department of Electrical & Computer Engineering

## MILESTONE-1

## CSE299

## Section-20

## Course Faculty:Dr. Nabeel Mohammed [NBM]

**Submitted by**

Sanjana Aktar Maria Id:2132533042

Afifa Imran

Nazib Riasat Omio

# Introduction

The RAG Chatbot is a document retrieval and response generation system designed to extract information from various file formats (PDFs, DOCX, TXT, and images) and provide relevant answers to user queries. The system leverages Retrieval-Augmented Generation (RAG) techniques, combining document retrieval using FAISS (Facebook AI Similarity Search) and response generation using a local LLM (Large Language Model).

# System Design

## Architecture

The system is divided into the following components:

**Document Processing:**

- Extracts text from PDFs, DOCX, TXT, and image files.
- Uses OCR (Optical Character Recognition) for image files.

**Embedding Generation:**

- Converts extracted text into embeddings using a pre-trained Sentence Transformer model.

**Indexing and Retrieval:**

- Builds a FAISS index for efficient similarity search.
- Retrieves relevant documents based on user queries.

**Response Generation:**

- Uses a local LLM (GPT4All) to generate responses based on retrieved documents.

**Interactive Chat Interface:**

- Allows users to interact with the chatbot and input queries or files.

# Algorithms

## Text Extraction

- PDFs: Uses PyMuPDF (fitz) to extract text from PDF files.
- DOCX: Uses python-docx to extract text from DOCX files.
- TXT: Reads text directly from TXT files.
- Images: Uses Tesseract OCR (via pytesseract) to extract text from image files.

## Embedding Generation

- Uses the Sentence Transformer model (all-MiniLM-L6-v2) to convert text into embeddings.
- Embeddings are stored as numpy arrays for efficient indexing.

## Indexing and Retrieval

- FAISS is used to build an index of document embeddings.
- The index supports efficient similarity search using the L2 distance metric.
- New documents are added to the index dynamically, and the index is saved for future use.

## Response Generation

- The GPT4All model (orca-mini-3b-gguf2-q4_0.gguf) is used to generate responses.
- The model is prompted with the retrieved documents and the user query to generate a context-aware response

# Libraries Used

## Core Libraries

- PyMuPDF (fitz): For extracting text from PDF files.
- python-docx: For extracting text from DOCX files.
- Pillow (PIL): For image processing.
- pytesseract: For OCR-based text extraction from images.
- Sentence Transformers: For generating text embeddings.
- FAISS: For building and querying the document index.
- GPT4All: For local LLM-based response generation.
- pickle: For serializing and deserializing data (e.g., documents, tracked files).

## Additional Libraries

- os: For file and directory operations.
- numpy: For handling embeddings and numerical operations.