



INTERNSHIP REPORT

PAKISTAN CIVIL AVIATION AUTHORITY

SUBMITTED BY

NAVEEN KHAN

SUBMITTED TO

SIR ADEEM KHAN

DATED

13 - 08 - 2025

CHATBOT INTEGRATION PROJECT



INTRODUCTION

Newly launched CATI Chatbot. CATI has developed its own intelligent chatbot to make communication faster, easier and more interactive, so that it can instantly answer your questions, share important updates, and guide you to the right information, helping us serve you better.

Developed By:

Naveen khan D/o Masood Ahmed khan Roll# 21CS059 from Computer Systems Engineering Department, MUET Jamshoro

Supervised By:

Sir Adeem Khan

(Office IT Incharge)

Civil Aviation Training Institute Hyderabad.



CONTENTS

Topic Title	Page#
Title Of Project	Cover Page
Introduction	1
Abstract	2
1. Introduction	3
2. Motivations	3
3. What prompted the project?	3
4. Situation Analysis	4
5. Objectives	4
6. Designing phase of project	5
7. Development Phase of Project	6
8. Implementation of Tools in this Project	9
9. Advantages	14
10. Conclusion	14



ABSTRACT

Our project is about the development of an AI-based chatbot for the Civil Aviation Training Institute, Hyderabad. This project is relevant to the internship program assigned by our Supervisor, Sir Adeem Khan (IT Incharge Officer), at Civil Aviation Training Institute, Hyderabad. The duration of the internship was 4 weeks. The tools and technologies used in making this project include .NET Core SDK 3.1, Visual Studio IDE, Bot Framework Emulator, and IIS (Internet Information Services) for deployment. The main objective of this project is to create a chatbot that can answer user queries, provide important information, and assist in communication through automated responses.

To begin the project, we first analyzed and planned how the chatbot would be developed and completed within the given time. We divided the work into two main phases: the design phase and the implementation phase. In the designing phase, we structured the conversation flow, decided on intents, and planned how the bot would interact with users. We also finalized the design of prompts, responses for better user interaction.

In the implementation phase, we created the chatbot project in Visual Studio using the Bot Framework template, implemented dialog logic, and configured the bot to run locally for testing in Bot Framework Emulator. After ensuring proper functionality, the bot was published to IIS for hosting, so that it could be accessed via a web endpoint. We tested and debugged the chatbot to ensure it was working efficiently and handling user queries without errors. After successful debugging, the project was demonstrated, and snapshots of its working are included later in this report.



1. Introduction

This project is relevant to our summer internship program. It is assigned to us by the IT incharge Officer of Civil Aviation Training Institute (CATI) Hyderabad, Sir Adeem khan, He is also our supervisor of the project. Internship periods start from the 1st July 2025 to 31st July 2025, we have joined the internship program with the aim of learning the development of Ai based projects. The objective was to design and implement a functional chatbot capable of providing quick responses, sharing information, and improving communication within the institute. Through this project, we enhanced our technical skills, gained practical experience, and learned to align our work with the requirements of the institutes.

2. Motivations

The motivation behind developing this chatbot project was to create an efficient, interactive, and user-friendly system that assist in providing instant responses, streamline communication, and reduce the manual workload within the institute. With the increasing reliance on digital solutions, it became essential to introduce an AI-based system that could operate 24/7 and address queries promptly. This project was also driven by our desire to explore modern AI technologies, enhance our problem-solving skills, and contribute to the technological growth of the Civil Aviation Training Institute (CATI) Hyderabad under the guidance of **Sir Adeem Khan**. By working on this project, we aimed to bridge the gap between theoretical learning and real-world application, ensuring that our work adds value to both the institute and our professional growth.

3. What prompted the Project?

Our project is about creating a chatbot for the Civil Aviation Training Institute, Hyderabad, aimed is to providing instant, user-friendly responses to queries and improving communication by delivering quick access to relevant information without constant human assistance.



4. Situation Analysis

After receiving the chatbot project, we analyzed how it could be developed and completed effectively. For this purpose, we created a basic plan outlining the bot's features, conversation flow, and integration requirements. We designed the bot to be interactive, accurate, and user-friendly, ensuring it could handle queries efficiently and provide relevant information to users in real-time.

4.1 Project Challenges

As with any new work, challenges are an inevitable part of the process. Similarly, when I started working on this project, I faced several difficulties. Initially, I had no clear idea about how to integrate the chatbot effectively. Another major challenge was selecting the right tools and technologies that could easily integrate with the .NET 4.8 version. Integrating the Bot Framework with .NET 4.8 proved to be quite challenging, requiring research, experimentation, and careful configuration to ensure smooth functionality.

5. Objectives

There are many objectives of our project few are discuss below:

- a) create a functional and interactive chatbot for Civil Aviation Training Institute (CATI).
- b) To provide quick and accurate responses to user queries.
- c) To improve accessibility to institute-related information.
- d) To ensure compatibility with the .NET 4.8 framework.
- e) To design a reliable, user-friendly, and scalable solution for future upgrades.

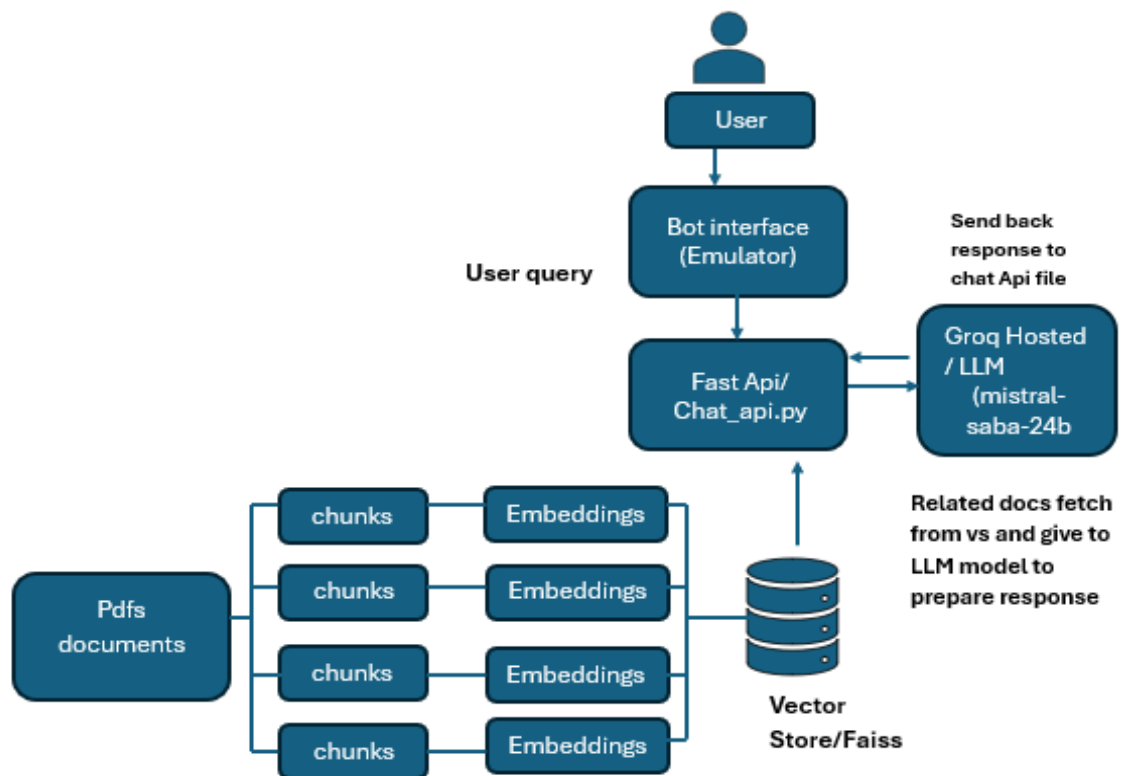


6. Designing Phase Of Project

6.1 Stages

s.no	Stages	Library/Tools
1	Text extraction	PyMuPDF (fitz)
2	Cleaning	Re
3	Text processing(chunking)	Langchain text splitter
4	Embeddings	HuggingFace + all-MiniLM-L6-v2
5	Vector Store	FAISS via langchain_community
6	LLM Inference	Groq API (mistral-saba-24b model)
7	Backend API Framework	FastAPI
8	Chatbot framework	Microsoft Bot Framework (.NET C#)

6.2 System Flow Diagram



6.2 Workflow

- a) User sends a query through the Bot Interface (Emulator).
- b) The Microsoft Bot Framework forwards the query to FastAPI (chat_api.py).
- c) FastAPI retrieves relevant document chunks from the Vector Store (FAISS).
- d) The Groq-hosted LLM (meta-llama/llama-4-maverick-17b-128e-instruct) processes the retrieved chunks + user query.
- e) The LLM response is sent back to the Bot Interface for the user.

7. Development Phase of Project

In this phase, we transformed our planned system design into an actual, functional chatbot system. At this stage, we focused on implementing the backend logic, integrating AI capabilities, and establishing communication between the bot interface and the backend API. We thoroughly tested the chatbot's core features, such as message handling, API communication, and response generation, as well as checking for compatibility with the .NET 4.8 environment and Bot Framework Emulator.

7.1 Development Environment Setup

For integration of chatbot these tools are need to install

- a) Visual Studio Code for backend code writing.
- b) Python for backend
- c) Visual Studio IDE (2019) for integrating with bot framework
- d) Bot Framework Emulator for local testing.
- e) .NET Framework 4.8 packages and Bot Framework SDK for C#.

7.2 Knowledge Base Creation

In this phase, we prepared the main knowledge source for our chatbot using organization-related PDF documents. First, we extracted the complete text from the PDFs, then cleaned it by removing unnecessary spaces, formatting, and irrelevant characters. Using LangChain's RecursiveCharacterTextSplitter, we split the text into smaller, overlapping chunks to ensure



accurate and relevant retrieval during queries. Finally, the processed chunks were stored in a `knowledge_base.jsonl` file, serving as the permanent data source for the chatbot.

Required library

- a) **json** – Store processed chunks in JSONL format.
- b) **fitz** – Extract text from PDFs.
- c) **os** – Access files and directories.
- d) **re** – Clean text using regular expressions.
- e) **RecursiveCharacterTextSplitter** – Split text into small, overlapping chunks for better search accuracy.

7.3 Vector Store Development

After preparing the knowledge base, I built a vector store using FAISS(Facebook AI Similarity Search) for fast and efficient similarity search. Each text chunk from `knowledge_base.jsonl` was converted into numerical embeddings using HuggingFace's `all-MiniLM-L6-v2` model. These embeddings were stored in a FAISS index that allows the chatbot to quickly find the most relevant chunks for a given user query. This retrieved chunks are later passed to the LLM model (Mistral via Groq API) to generate accurate, context-aware answers.

Required library

- a) **langchain_community.vectorstores.FAISS** : Store and retrieve vector embeddings efficiently.
- b) **HuggingFaceEmbeddings** : Convert text chunks into semantic vector representations.
- c) **os** : Manage file paths for storing and loading the FAISS index.

7.4 Backend Api Integration

In this phase, the backend API is developed using FastAPI to handle incoming user queries and generate intelligent responses. The `/ask` endpoint is created to receive POST requests containing user messages. This endpoint processes the queries by leveraging the Groq API integrated with the Mistral language model (`mistral-saba-24b`) for generating contextually relevant AI responses.

The API is designed to interface smoothly with the Bot Framework by accepting input from the chatbot frontend, querying the vector database for relevant information, and returning



concise, accurate replies. Proper error handling and response formatting are implemented to ensure robustness and ease of integration.

This backend service acts as the core logic layer, connecting the knowledge base, AI model, and frontend chatbot interface, facilitating real-time, intelligent interactions.

Required library

- a) **os**: Manage file paths and check vector store existence
- b) **requests**: Call external Groq API for AI responses
- c) **fastapi**: Create API endpoints and handle HTTP requests
- d) **pydantic**: Validate input data schemas
- e) **langchain_community.vectorstores.FAISS**: Load and search vector embeddings efficiently
- f) **langchain.embeddings.HuggingFaceEmbeddings**: Generate/load text embeddings with HuggingFace models

7.5 Bot Framework Integration

In this phase, the chatbot backend is connected with the Microsoft Bot Framework to enable seamless interaction between the user and the AI model. An EchoBot.cs class is created within the .NET 4.8 project to handle incoming messages from users via the Bot Framework Emulator or other channels.

The bot captures user input and forwards it to the FastAPI backend /ask endpoint, which processes the query and returns an AI-generated response. This response is then sent back through the Bot Framework to the user interface.

7.6 Testing And Debugging

During this phase, the chatbot system undergoes thorough testing to verify seamless communication between the frontend, backend API, and Bot Framework components. The Bot Framework Emulator is primarily used to simulate user interactions and test message exchanges in a controlled environment.

Common challenges faced during testing include:

- a) **API Routing Issues**: Incorrect endpoint URLs or controller mismatches causing 404 errors.

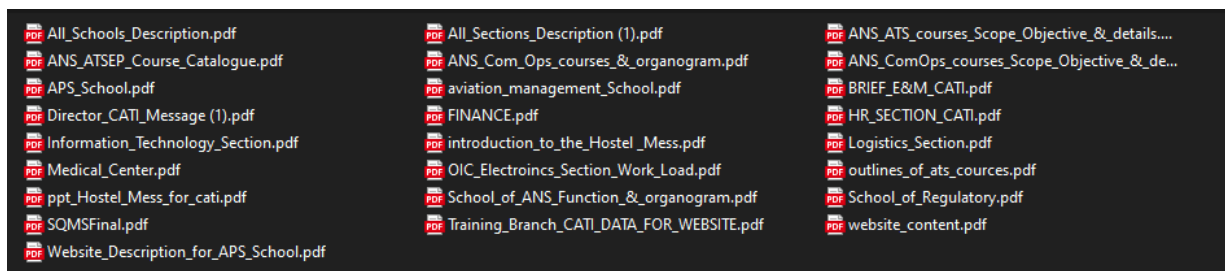


- b) **CORS Errors:** Cross-Origin Resource Sharing problems blocking frontend requests to backend.
- c) **Data Serialization Errors:** Mismatches in JSON request/response formats leading to parsing failures.
- d) **Latency and Timeout:** Slow API responses affecting user experience.
- e) **Debugging Distributed Components:** Tracing errors across separate FastAPI and .NET services can be complex.

8. Implementation of Project

8.1 Dataset collection

We collect the data in the form of pdf



8.2 Knowledge base creation

In this stage I done text extraction from above pdf and I clean that text from white spacing and then I divided complete text into smaller chunks and this chunks I stored in knowledgebase.



8.3 vector store

Now this chunks convert into docs and convert into vector embedding (numerical formate) in order to store in vector database.

```
#####Load chunks from jsonl file#####



def load_chunks(jsonl_file):
    documents = []
    with open(jsonl_file, 'r', encoding='utf-8') as f:
        for line in f:
            item = json.loads(line)
            documents.append(Document(page_content=item['text']))
    return documents

#####Convert docs into vector embeddings#####
# create faiss for fast similarity search

def build_vector_store(docs, faiss_path="vector_store"):
    # embeddings = GPT4AllEmbeddings() # this help to convert text into number

    embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
    db = FAISS.from_documents(docs, embeddings) #in docs ,list of object is in document formate with page cont
    db.save_local(faiss_path)
    print(f"✅ FAISS vector store saved to: {faiss_path}")
```

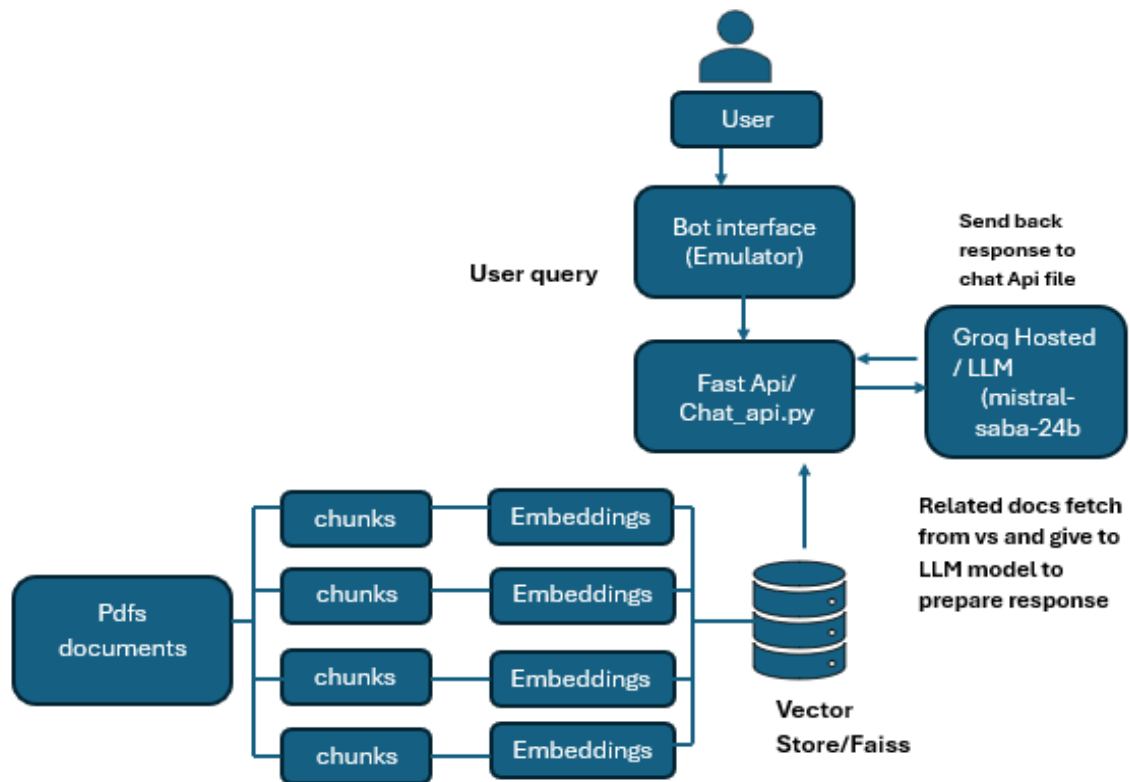
After store embedding in vectore store these files are created in file structure

	index.faiss	7/23/2025 10:05 AM	FAISS File	646 KB
	index.pkl	7/23/2025 10:05 AM	PKL File	241 KB

8.4 Backend Api Integration

The backend API is built using FastAPI to receive user questions via the /ask endpoint. It queries the FAISS vector store to find relevant documents based on the user input. The combined context and question are sent to the Groq API with the Mistral language model to generate an AI response. Finally, the API returns the answer to the frontend for display.





8.5 Bot Framework echo bot template

The bot's core logic resides in the `EchoBot.cs` class, which is responsible for receiving user messages, forwarding these messages to an AI backend API, and returning the AI-generated responses to the user in real-time.

How It Works

- When a user sends a message, the bot captures the input text via the overridden method `OnMessageActivityAsync`.
- The bot then constructs a JSON payload containing the user's question and sends it as an HTTP POST request to a **FastAPI** backend endpoint (`http://localhost:8000/ask`).
- The FastAPI service, hosting the AI logic and PDF-trained knowledge base, processes the question and returns an AI-generated answer in JSON format.
- Upon receiving the response, the bot parses the JSON, extracts the answer text, and sends it back to the user through the Bot Framework.
- Error handling is included to manage any API failures or exceptions, ensuring that the user receives meaningful feedback in case of issues.



Results

cati mission

Just now

CATI's Mission

To deliver world-class civil aviation training that equips trainees with the skills, knowledge and competencies required across diversified disciplines, in line with our forward-looking Vision 2030. By providing high-quality instruction, state-of-the-art facilities (including our 24/7 Medical Center), and a supportive learning environment, CATI strives to develop competent professionals who advance the safety, efficiency and innovation of the civil aviation sector.

Just now

atc courses

A minute ago

Here are some ATC courses:

1. **Basic ATC Course (ICAO-051):** An introductory course for air traffic control students.
2. **Aerodrome Control Training (ICAO-052):** A course focused on aerodrome control procedures.
3. **Approach Control (Procedure) Training (ICAO-053):** A course on approach control procedures.
4. **Area Control (Procedure) Training (ICAO-055):** A course on area control procedures.
5. **ATC Assistant (Refresher) Course:** A refresher course for ATC assistants.

These courses are designed to provide students with the necessary knowledge and skills to work as air traffic controllers. If you're interested in learning more about a specific course, I can try to provide more information.

A minute ago



RFFS – Radio Frequency Fixed Services

In the context of our company's aviation and communications work, **RFFS** refers to the set of radio services that operate on fixed radio stations (i.e., stations that are permanently installed at a specific location). These services are a core part of the **Aeronautical Fixed Services (AFS)** that we manage and maintain, and they fall under the broader umbrella of **Aeronautical Radio Frequency Spectrum Management (ARFSM)**.

Key Points

Aspect	Details
What RFFS Are	Fixed-station radio services used for voice, data, and telemetry in aviation.
Relation to AFS	RFFS is a subset of the Aeronautical Fixed Services that we oversee, including HF, VHF, and EVHF communications.
Why It Matters	Proper management of RFFS ensures reliable, interference-free communication

Just now

9. Advantages

- Provides instant, real-time user interaction
- Easily integrates with your existing .NET website
- Uses free/open-source tools to reduce costs
- Offers 24/7 support and improves user accessibility
- Simple to update by adding new documents
- Utilizes advanced AI for better understanding of questions
- Includes error handling for reliable performance

10. Conclusion

A real-time AI chatbot integrated into a website enhances how users access knowledge efficiently. It provides quick, accurate answers based on the organization's own documents, improving user experience. This reduces the need to search through lengthy files or wait for



manual help. Overall, it makes information accessible anytime, supporting timely and effective communication.

