Project Report: AI-Powered Chatbot Web Application

1. Executive Summary

This repository contains the code for an AI-Powered Chatbot Web Application, developed as an internship project by Saif Ullah Awan. The application features a .NET frontend, a Python backend for AI processing and file handling, and uses MongoDB for persistent chat history storage.

2. Project Overview

This project aims to provide an intelligent, interactive chatbot solution capable of engaging in conversational dialogue, processing uploaded documents (PDFs and images), and maintaining a persistent chat history. It showcases a modern, decoupled architecture designed for scalability and maintainability.

3. Features

- Interactive Chat Interface: A user-friendly web interface for real-time text-based conversations.

- AI-Powered Responses: Integrates a pre-trained large language model (DialoGPT) to generate intelligent and contextually relevant replies.

- Document Text Extraction: Supports uploading PDF and image (PNG, JPG, JPEG) files, from which the chatbot can extract and present the contained text.

- Persistent Chat History: Automatically saves all chat messages, uploaded file details, and chatbot replies into a MongoDB database.

- Chat History Retrieval: Provides an endpoint to easily fetch and display the complete history of interactions.

- Scalable Architecture: Designed with decoupled components to allow for independent scaling of frontend and backend services.

- Secure File Handling: Implements secure practices for managing uploaded files.

4. Technologies Used

Frontend (.NET Web Application)
- Framework: ASP.NET Core Razor Pages

- Language: C#
- Libraries: HttpClient, Microsoft.AspNetCore.Mvc
- Database Driver: MongoDB.Driver

Backend (Python Flask API)
- Language: Python 3.x
- Framework: Flask
- AI Model: DialoGPT via transformers library
- PDF Processing: PyPDF2
- Image Processing (OCR): Pillow (PIL), pytesseract
- File Handling: werkzeug.utils.secure_filename

Database
- Type: MongoDB
- Purpose: Stores ChatHistory logs

5. Architecture

The application follows a decoupled architecture:
- Frontend (.NET): Handles UI, user input, and API communication.
- Backend (Python): Processes input, runs AI model, handles file extraction and storage.
- MongoDB: Stores all chat interactions persistently.

6. Setup and Installation

1. Clone the Repository: https://github.com/saifxawan/Pythonchatbot
2. Set up MongoDB and configure the connection string in appsettings.json
3. Configure and run Python backend using Flask
4. Install Python dependencies and Tesseract for OCR
5. Run .NET frontend using dotnet run

7. Usage

Ensure both Python backend and .NET frontend are running.
Visit the .NET frontend URL in browser, interact with chatbot, upload files, and view chat responses. MongoDB handles persistent chat history.

8. Project Structure

```
.
├── ChatbotWebApp/
│   ├── Controllers/
```

```
│   ├── Models/
│   ├── Services/
│   ├── Pages/
│   └── appsettings.json
└── PythonBackend/
    ├── chatbot.py
    ├── uploads/
    └── venv/
```

GitHub: https://github.com/saifxawan/Pythonchatbot

LinkedIn: https://www.linkedin.com/in/saif-ullah-awan-404953373/

Project proposal link: https://blue-joann-81.tiiny.site/