LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:-Pruthibiraj Nayak

Roll No: 2230183

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Experiment Number	4
Experiment Title	Conversational Chatbot with Any Files
Date of Experiment	04.02.2025
Date of Submission	09.02.2025

1. Objective:-

To build a chatbot capable of answering queries from an uploaded PDF/Word/Excel document.

2. Procedure:- (Steps Followed)

1. Integrate open-source LLMs such as LLama or Gemma from Ollama
2. Develop a Flask backend to process the PDF/word/excel content.
3. Implement Natural Language Processing (NLP) to allow queries. You can use LLamaIndex or Langchain
4. Create a frontend to upload document files and interact with the chatbot, just like OpenAI interface
5. Provide an option to choose the LLM model from a dropdown list. 6. Display the chatbot responses on the webpage.

Code:-

FLASK CODE

```
from flask import Flask, request, jsonify, send_from_directory
import os
from PyPDF2 import PdfReader

app = Flask(__name__)

# Global variable to store the processed PDF text
pdf_text = ""

# Ensure the uploads folder exists
UPLOAD_FOLDER = 'uploads'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

# Route to serve the HTML page
@app.route('/')
def index():
    return send_from_directory(os.getcwd(), 'index.html')

# Route to upload PDF and process its content
@app.route('/upload_pdf', methods=['POST'])
def upload_pdf():
    global pdf_text
    file = request.files['file']
```

```
    file_path = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(file_path)

    # Extract text from the uploaded PDF
    pdf_text = extract_text_from_pdf(file_path)
    return jsonify({'message': 'PDF uploaded and processed successfully!'})

# Route to get chatbot response based on uploaded PDF
@app.route('/get_response', methods=['POST'])
def get_response():
    if not pdf_text:
        return jsonify({'error': 'No PDF content available. Please upload a
PDF first.'})

    query = request.json.get('message')
    response = process_query_with_pdf_content(query)
    return jsonify({'response': response})
# Function to extract text from PDF
def extract_text_from_pdf(pdf_file):
    reader = PdfReader(pdf_file)
    text = ""
    for page in reader.pages:
        text += page.extract_text()
    return text
# Function to process the query with PDF content
def process_query_with_pdf_content(query):
    return f"Answer based on the PDF: {pdf_text[:200]}"  # This is a
placeholder response

if __name__ == '__main__':
    app.run(debug=True, port=5001)
```

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chat with PDF Bot</title>
    <style>
        body { background: lightblue !important; }
    </style>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h2>Chat with the PDF Bot</h2>
```

```
        <!-- File upload section -->
        <div class="upload-section">
            <input type="file" id="pdf-file" accept=".pdf"/>
            <button onclick="uploadPdf()">Upload PDF</button>
        </div>
        <div id="chat-box">
            <!-- Chat content will appear here -->
        </div>
        <div class="chat-input">
            <textarea id="user-input" placeholder="Ask a
question..."></textarea>
            <button onclick="sendMessage()">Send</button>
        </div>
    </div>
    <script>
        function uploadPdf() {
            const fileInput = document.getElementById('pdf-file');
            const file = fileInput.files[0];
            if (!file) {
                alert("Please select a PDF file first.");
                return;
            }
            const formData = new FormData();
            formData.append('file', file);
            fetch('/upload_pdf', {
                method: 'POST',
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                alert(data.message);
            })
            .catch(error => {
                alert('Error uploading PDF');
            });
        }
        function sendMessage() {
            const userMessage = document.getElementById('user-input').value;
            const chatBox = document.getElementById('chat-box');

            if (userMessage.trim() === '') return;
            // Append the user's message
            chatBox.innerHTML += `<div class="user-message">User:
${userMessage}</div>`;

            // Send query to the server for response
            fetch('/get_response', {
                method: 'POST',
```

```
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ message: userMessage })
            })
            .then(response => response.json())
            .then(data => {
                // Append the bot's response
                chatBox.innerHTML += `<div class="bot-message">Bot:
${data.response}</div>`;
                chatBox.scrollTop = chatBox.scrollHeight;
            })
            .catch(error => {
                chatBox.innerHTML += `<div class="bot-message">Bot: Sorry,
there was an error.</div>`;
            });
            // Clear the input field
            document.getElementById('user-input').value = '';
        }
    </script>
</body>
</html>
```

CSS SHEET

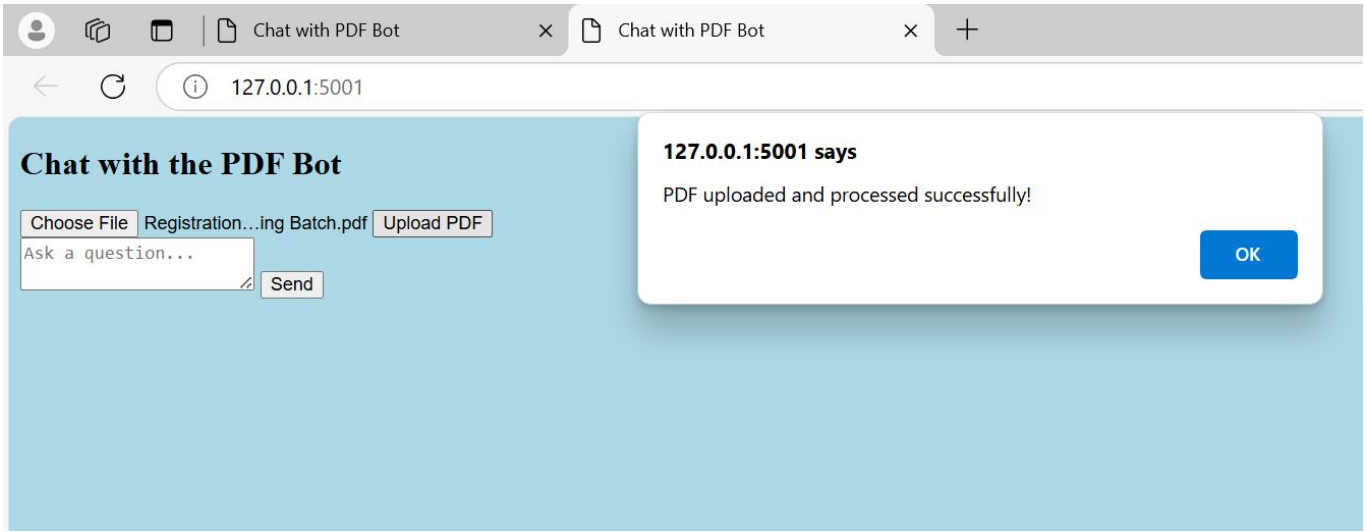
```
/* Import Google Font */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&displa
y=swap');

/* General Styling */
body {
    font-family: 'Poppins', sans-serif;
    background: linear-gradient(135deg, #ffdde1, #ee9ca7, #ffdde1);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

/* Container */
.container {
    background: linear-gradient(135deg, #ffe6f7, #e3f2fd);
    padding: 20px;
    border-radius: 15px;
    box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.2);
    text-align: center;
    width: 50%;
    max-width: 500px;
```

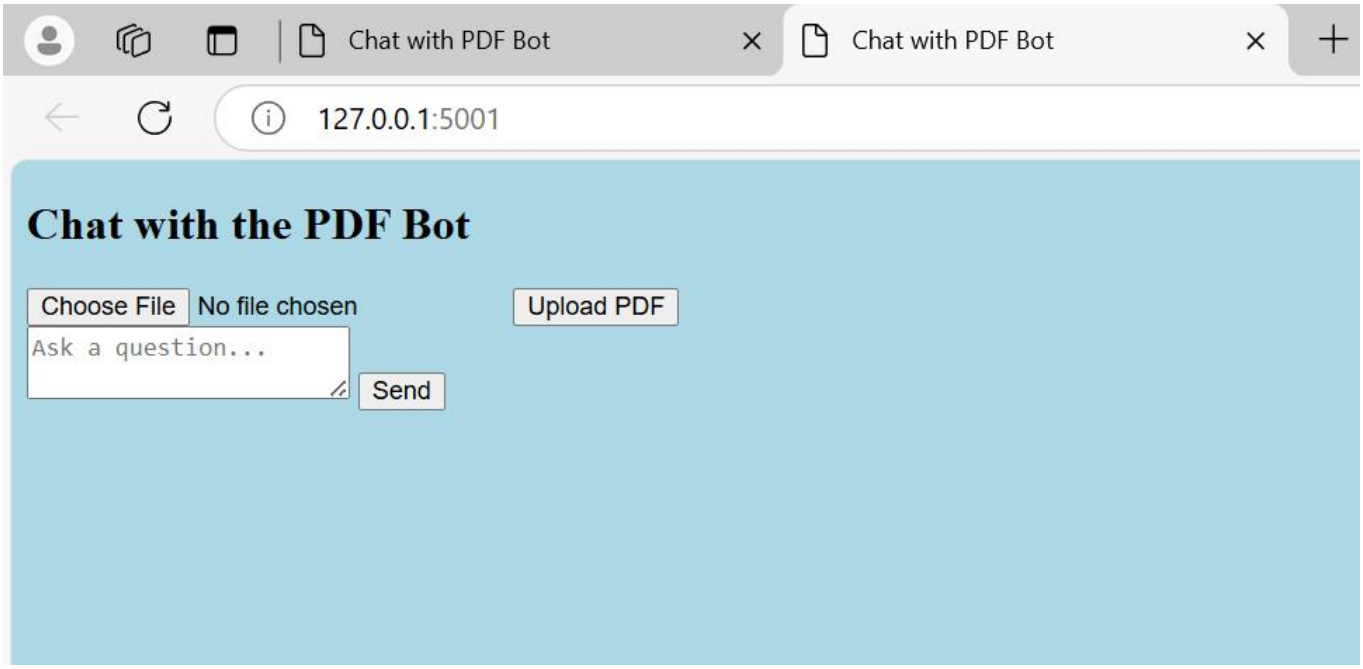
```
    display: flex;
    flex-direction: column;
    align-items: center;
}

/* Heading */
h2 {
    color: #6a4c93;
    margin-bottom: 15px;
    font-weight: 600;
}

/* File Upload Section */
.upload-section {
    display: flex;
    justify-content: center;
    gap: 10px;
    margin-bottom: 15px;
    width: 100%;
}

input[type="file"] {
    border: none;
    padding: 10px;
    border-radius: 10px;
    background: #f6d5f7;
    color: #6a4c93;
    cursor: pointer;
    font-size: 14px;
}

input[type="file"]::-webkit-file-upload-button {
    background: #ffb6c1;
    border: none;
    padding: 10px;
    border-radius: 10px;
    color: white;
    cursor: pointer;
    transition: 0.3s;
}

input[type="file"]::-webkit-file-upload-button:hover {
    background: #ff6f91;
}

/* Buttons */
button {
    background: #ff9a8b;
    border: none;
```

```
    padding: 10px 15px;
    border-radius: 10px;
    color: white;
    font-size: 14px;
    cursor: pointer;
    transition: 0.3s;
    font-weight: bold;
}

button:hover {
    background: #ff6f61;
}

/* Chat Box */
#chat-box {
    background: #faf3f3;
    height: 250px;
    overflow-y: auto;
    padding: 15px;
    border-radius: 15px;
    margin-bottom: 15px;
    width: 90%;
    box-shadow: inset 0px 2px 5px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    align-items: flex-start;
}

/* Chat Messages */
.user-message, .bot-message {
    padding: 10px;
    border-radius: 10px;
    margin: 5px 0;
    width: fit-content;
    max-width: 80%;
    font-size: 14px;
}

.user-message {
    background: #ffebee;
    align-self: flex-end;
    text-align: right;
    border-radius: 10px 10px 0px 10px;
}

.bot-message {
    background: #e3f2fd;
    align-self: flex-start;
    text-align: left;
```

```
    border-radius: 10px 10px 10px 0px;
}

/* Chat Input */
.chat-input {
    display: flex;
    gap: 10px;
    align-items: center;
    width: 90%;
}

textarea {
    flex: 1;
    padding: 10px;
    border: 2px solid #e0e0e0;
    border-radius: 10px;
    resize: none;
    font-size: 14px;
    background: #f8efff;
    border: none;
}

/* Mobile Responsiveness */
@media (max-width: 600px) {
    .container {
        width: 90%;
    }
}
```

3. Results/Output:- Entire Screen Shot including Date & Time





4. Remarks:-

In this experiment, we built a Conversational Chatbot with PDF Support that enables users to upload PDFs, extract their content, and interact with an AI-powered model for contextual responses. Using Flask

as the backend, LLamaIndex/LangChain for document processing, and Ollama for LLM integration, we created a seamless experience with a visually appealing frontend for file uploads and chat interactions. The chatbot successfully retrieves answers based on the document's content, demonstrating the potential of NLP and LLMs in document-based AI assistants. Future enhancements could include multi-file support, advanced summarization, and real-time response optimization.

Pruthibiraj Nayak (2230183)

_____	_____
(Name of the Student)	(Name of the Coordinator)