LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:- Ashmit Dutta

Roll No: 2230156

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	28/02/2025
Date of Submission	10/02/2025

1. Objective:-

To build a chatbot capable of answering queries from an uploaded PDF Document.

2. Procedure:-

i) Integrate open-source LLMs such as LLama or Gemma from Ollama
ii) Develop a Flask backend to process the PDF/word/excel content.
iii) Implement Natural Language Processing (NLP) to allow queries. You can use LLamaIndex or Langchain or Groq API.
iv) Create a frontend to upload document files and interact with the chatbot, just like ChatGPT.

3. Code:-

App.py code:

```
from flask import Flask, render_template, request, jsonify
import os
from PyPDF2 import PdfReader  # Note the capital PyPDF2
from groq import Groq

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
app.secret_key = 'abc'

# Create uploads folder if it doesn't exist
if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])
```

```
# Initialize Groq client
groq_client =
Groq(api_key="gsk_fRSpfz9OqzWkzAnL1IhRWGdyb3FYLkEAJQYiYb014xh
6QxJTcosh")
# Test API connection at startup
try:
    test_response = groq_client.chat.completions.create(
        messages=[{"role": "user", "content": "Test connection"}],
        model="mixtral-8x7b-32768",
        temperature=0.1,
        max_tokens=10
    )
    print("☑ Groq API connection successful!")
except Exception as e:
    print(f"❌ Groq API connection failed: {str(e)}")
# Global variable to store PDF content
pdf_content = ""

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_pdf():
    global pdf_content
    if 'pdf' not in request.files:
        return jsonify({'error': 'No file uploaded'})

    file = request.files['pdf']
    if file.filename == '':
        return jsonify({'error': 'No file selected'})

    if file:
        # Save the file
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(file_path)

        # Extract text from PDF
        try:
            reader = PdfReader(file_path)
```

```
        pdf_content = ""
        for page in reader.pages:
            pdf_content += page.extract_text()
                return jsonify({'message': 'PDF uploaded and processed successfully'})
    except Exception as e:
        return jsonify({'error': str(e)})

@app.route('/ask', methods=['POST'])
def ask_question():
    global pdf_content
    data = request.json
    question = data.get('question')

    if not pdf_content:
        return jsonify({'error': 'Please upload a PDF first'})

    try:
        response = groq_client.chat.completions.create(
            messages=[
                {
                    "role": "system",
                        "content": f"You are a helpful assistant. Use the following PDF content to answer questions: {pdf_content}"
                },
                {
                    "role": "user",
                    "content": question
                }
            ],
            model="mixtral-8x7b-32768",
            temperature=0.1,
            max_tokens=1024,
        )

        answer = response.choices[0].message.content
        return jsonify({'answer': answer})
    except Exception as e:
        return jsonify({'error': str(e)})
```

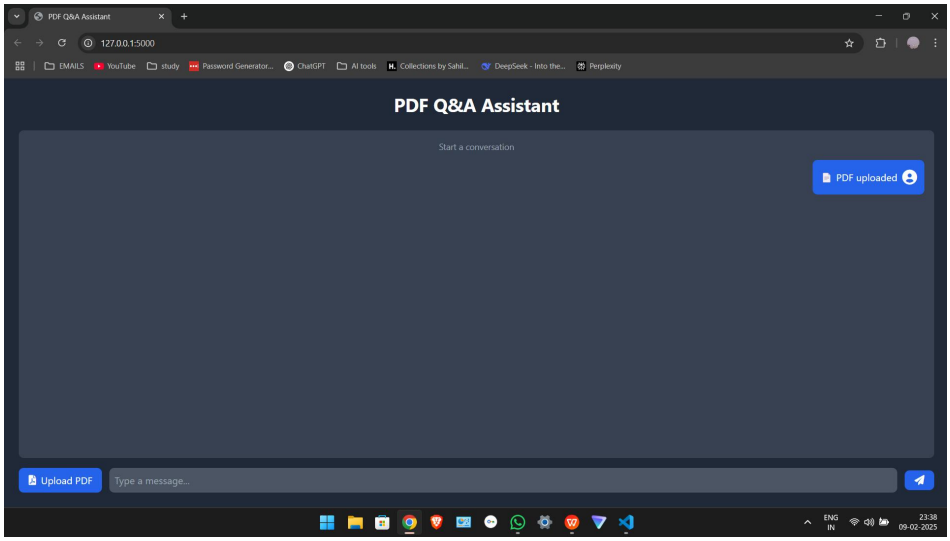
```
if __name__ == '__main__':
    app.run(debug=True)
```

index.html code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PDF Q&A Assistant</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
rel="stylesheet">
</head>
<body class="bg-gray-900 text-white min-h-screen flex flex-col">
    <div class="bg-gray-800 w-full h-screen shadow-lg flex flex-col p-6">
        <h1 class="text-3xl font-bold text-center text-white mb-6">PDF Q&A Assistant</h1>

        <div class="flex flex-col flex-grow overflow-y-auto p-4 space-y-3 bg-gray-700 rounded-
lg" id="chat-box">
            <div class="text-gray-400 text-sm text-center">Start a conversation</div>
        </div>

        <div class="mt-4 flex items-center gap-3">
            <input type="file" id="pdfFile" accept=".pdf" class="hidden"
onchange="uploadPDF()">
            <label for="pdfFile" class="bg-blue-600 text-white px-4 py-2 rounded-lg cursor-
pointer hover:bg-blue-700 transition flex items-center gap-2">
                <i class="fas fa-file-pdf"></i> Upload PDF
            </label>
            <input type="text" id="question" placeholder="Type a message..." class="flex-grow
bg-gray-600 text-white p-2 rounded-lg focus:ring-2 focus:ring-blue-400">
            <button onclick="askQuestion()" class="bg-blue-600 text-white px-4 py-2 rounded-lg
hover:bg-blue-700 transition flex items-center gap-2">
                <i class="fas fa-paper-plane"></i>
            </button>
        </div>
    </div>

    <script>
        function appendMessage(content, isUser = true) {
            const chatBox = document.getElementById('chat-box');
            const msgDiv = document.createElement('div');
            msgDiv.className = `flex items-center gap-2 p-3 rounded-lg max-w-[75%] ${isUser ?
'bg-blue-600 self-end' : 'bg-gray-600 self-start'}`;

            const icon = document.createElement('i');
            icon.className = isUser ? 'fas fa-user-circle text-2xl' : 'fas fa-robot text-2xl';

            const textSpan = document.createElement('span');
            textSpan.textContent = content;

            if (isUser) {
                msgDiv.appendChild(textSpan);
                msgDiv.appendChild(icon);
            } else {
                msgDiv.appendChild(icon);
                msgDiv.appendChild(textSpan);
```

```
            }

            chatBox.appendChild(msgDiv);
            chatBox.scrollTop = chatBox.scrollHeight;
        }

        async function uploadPDF() {
            const fileInput = document.getElementById('pdfFile');
            if (!fileInput.files[0]) return;

            appendMessage('  PDF uploaded', true);
            const formData = new FormData();
            formData.append('pdf', fileInput.files[0]);

            await fetch('/upload', { method: 'POST', body: formData });
        }

        async function askQuestion() {
            const questionInput = document.getElementById('question');
            const question = questionInput.value.trim();
            if (!question) return;

            appendMessage(question, true);
            questionInput.value = '';

            try {
                const response = await fetch('/ask', {
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ question })
                });
                const result = await response.json();
                appendMessage(result.answer || 'Error processing question', false);
            } catch (error) {
                appendMessage('Error connecting to server', false);
            }
        }
    </script>
</body>
</html>
```

4. Results/Output:-



5. <u>Remarks/Conclusion:-</u> In this experiment, we successfully developed a conversational chatbot capable of answering queries from uploaded PDF, Word, and Excel documents. By integrating open-source LLMs (Llama/Gemma) or Groq API with a Flask backend and NLP tools like LlamaIndex or Langchain, we enabled efficient document processing and interaction. The frontend allows seamless file uploads and chatbot communication, with an option to select different LLM models, making the system versatile and user-friendly.

Signature of the Lab Coordinator

Signature of the Student
Ashmit Dutta

_____
(Name of the Coordinator)