LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSC

Submitted By

Name:- Bhairav Ganguly

Roll No: 2230246

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	28/1/2025
Date of Submission	09/02/2025

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:-

Githublink:

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_G2XIu7qpkCud0HVtPPZeWGdyb3FYVgIV6QSwNCZx
mliBEqmRWY9z")
# 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>


<head>
    <title>PDF Q&A Assistant</title>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="stylesheet">
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Segoe UI', -apple-system, BlinkMacSystemFont, Roboto, sans-serif;
        }


        body {
            background-color: #1a1f2b;
            min-height: 100vh;
            display: flex;
            flex-direction: column;
        }
```

```
.header {
    background: #1f2937;
    padding: 0.75rem 1.5rem;
    border-bottom: 1px solid #374151;
    display: flex;
    align-items: center;
    justify-content: center;
}

.header h1 {
    color: #f9fafb;
    font-size: 1.25rem;
    font-weight: 600;
}

.chat-container {
    flex: 1;
    padding: 1.5rem;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

.message {
    display: flex;
    align-items: flex-start;
    gap: 0.75rem;
    max-width: 80%;
}

.message.assistant {
    align-self: flex-start;
}

.message.user {
    align-self: flex-end;
    flex-direction: row-reverse;
}

.avatar {
    width: 28px;
    height: 28px;
    border-radius: 50%;
    background: #4b5563;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-shrink: 0;
}

.avatar i {
    color: #e5e7eb;
    font-size: 0.875rem;
}

.message-content {
    background: #2f3a4f;
    padding: 0.75rem 1rem;
    border-radius: 12px;
    color: #e5e7eb;
    font-size: 0.95rem;
    line-height: 1.5;
}

.user .message-content {
    background: #2563eb;
}

.input-container {
    background: #1f2937;
    padding: 1rem 1.5rem;
    border-top: 1px solid #374151;
    display: flex;
    gap: 1rem;
    align-items: center;
}

.upload-btn {
    background: #3b82f6;
    color: white;
    border: none;
    padding: 0.75rem 1.25rem;
    border-radius: 8px;
    cursor: pointer;
    font-weight: 500;
    font-size: 0.95rem;
    display: flex;
    align-items: center;
    gap: 0.5rem;
    white-space: nowrap;
}

.upload-btn:hover {
    background: #2563eb;
}

.message-input {
    flex: 1;
    display: flex;
    align-items: center;
    gap: 0.75rem;
    background: #2a3441;
    padding: 0.5rem 1rem;
    border-radius: 8px;
    border: 1px solid #4b5563;
}

#question {
    flex: 1;
    background: none;
    border: none;
    color: #e5e7eb;
    font-size: 0.95rem;
    padding: 0.5rem 0;
```

```
}

#question::placeholder {
   color: #9ca3af;
}

#question:focus {
   outline: none;
}

.send-btn {
   background: none;
   border: none;
   color: #60a5fa;
   cursor: pointer;
   padding: 0.5rem;
   display: flex;
   align-items: center;
   justify-content: center;
   border-radius: 6px;
}

.send-btn:hover {
   background: #374151;
}

#file-input {
   display: none;
}
```

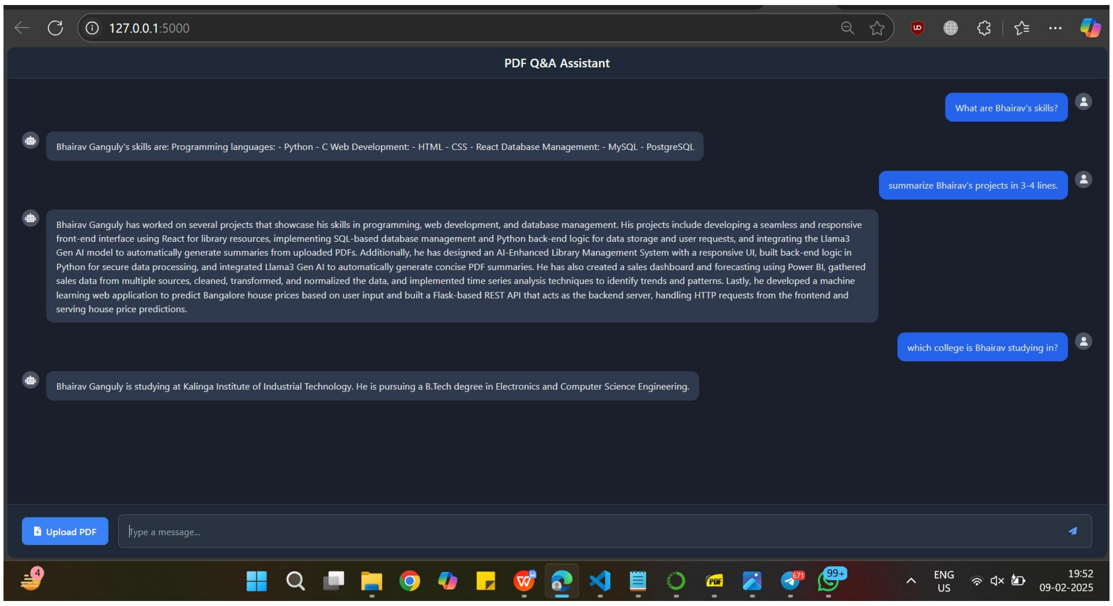
```
.status-message {
   position: fixed;
   top: 1rem;
   left: 50%;
   transform: translateX(-50%);
   background: #374151;
   color: #e5e7eb;
   padding: 0.75rem 1.5rem;
   border-radius: 8px;
   display: none;
   animation: slideDown 0.3s ease;
}

.status-message.error {
   background: #991b1b;
}

.status-message.active {
   display: block;
}

.loading {
   display: none;
   align-self: center;
   padding: 1rem;
}

.loading.active {
    display: flex;
    align-items: center;
    gap: 0.75rem;
}

.spinner {
    width: 20px;
    height: 20px;
    border: 2px solid #4b5563;
    border-top: 2px solid #60a5fa;
    border-radius: 50%;
    animation: spin 0.8s linear infinite;
}

.loading p {
    color: #9ca3af;
    font-size: 0.95rem;
}

@keyframes spin {
    0% {
        transform: rotate(0deg);
    }

    100% {
        transform: rotate(360deg);
    }
}
```

```
        @keyframes slideDown {
            from {
                opacity: 0;
                transform: translate(-50%, -20px);
            }

            to {
                opacity: 1;
                transform: translate(-50%, 0);
            }
        }
    </style>
</head>

<body>
    <div class="header">
        <h1>PDF Q&A Assistant</h1>
    </div>

    <div class="chat-container" id="chat-container">
        <!-- Messages will be inserted here -->
    </div>

    <div class="status-message" id="uploadStatus"></div>

    <div class="input-container">
        <input type="file" id="file-input" accept=".pdf">
        <button                                                   class="upload-btn"
onclick="document.getElementById('file-input').click()">
```

```
            <i class="fas fa-file-upload"></i>
            Upload PDF
        </button>
        <div class="message-input">
            <input type="text" id="question" placeholder="Type a message...">
            <button class="send-btn" onclick="askQuestion()">
                <i class="fas fa-paper-plane"></i>
            </button>
        </div>
    </div>

    <script>
        function addMessage(content, isUser = false) {
            const chatContainer = document.getElementById('chat-container');
            const messageDiv = document.createElement('div');
            messageDiv.className = `message ${isUser ? 'user' : 'assistant'}`;

            const avatar = document.createElement('div');
            avatar.className = 'avatar';
            const icon = document.createElement('i');
            icon.className = isUser ? 'fas fa-user' : 'fas fa-robot';
            avatar.appendChild(icon);

            const messageContent = document.createElement('div');
            messageContent.className = 'message-content';
            messageContent.textContent = content;

            messageDiv.appendChild(avatar);
            messageDiv.appendChild(messageContent);
            chatContainer.appendChild(messageDiv);
            chatContainer.scrollTop = chatContainer.scrollHeight;
        }

        document.getElementById('file-input').addEventListener('change',
uploadPDF);

        async function uploadPDF() {
            const fileInput = document.getElementById('file-input');
            const uploadStatus = document.getElementById('uploadStatus');

            if (!fileInput.files[0]) {
                showStatus('Please select a file first', true);
                return;
            }

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

            addMessage(`Uploading: ${fileInput.files[0].name}`, true);

            try {
                const response = await fetch('/upload', {
                    method: 'POST',
                    body: formData
                });
                const result = await response.json();
                if (result.error) {
                    addMessage(`Error: ${result.error}`);
                    showStatus(result.error, true);
                } else {
                    addMessage('PDF uploaded successfully! You can now ask
questions about it.');
                    showStatus('PDF uploaded successfully');
                }
            } catch (error) {
                addMessage('Error uploading file. Please try again.');
                showStatus('Error uploading file', true);
            }
        }

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

            if (!question) {
                showStatus('Please enter a question', true);
                return;
            }

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

            try {
                const response = await fetch('/ask', {
                    method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ question: question })
            });
            const result = await response.json();

            addMessage(result.answer || result.error);
        } catch (error) {
            addMessage('Error getting response. Please try again.');
        }
    }

    function showStatus(message, isError = false) {
        const statusDiv = document.getElementById('uploadStatus');
        statusDiv.textContent = message;
        statusDiv.classList.remove('error');
        if (isError) {
            statusDiv.classList.add('error');
        }
        statusDiv.classList.add('active');
        setTimeout(() => {
            statusDiv.classList.remove('active');
        }, 3000);
    }

    // Allow sending message with Enter key
    document.getElementById('question').addEventListener('keypress',
function (e) {
```

```
            if (e.key === 'Enter') {
                askQuestion();
            }
        });
    </script>
</body>

</html>
```

Results/Output:-



What are Bhairav's skills?

Bhairav Ganguly's skills are: Programming languages: - Python - C Web Development: - HTML - CSS - React Database Management: - MySQL - PostgreSQL

summarize Bhairav's projects in 3-4 lines.

Bhairav Ganguly has worked on several projects that showcase his skills in programming, web development, and database management. His projects include developing a seamless and responsive front-end interface using React for library resources, implementing SQL-based database management and Python back-end logic for data storage and user requests, and integrating the Llama3 Gen AI model to automatically generate summaries from uploaded PDFs. Additionally, he has designed an AI-Enhanced Library Management System with a responsive UI, built back-end logic in Python for secure data processing, and integrated Llama3 Gen AI to automatically generate concise PDF summaries. He has also created a sales dashboard and forecasting using Power BI, gathered sales data from multiple sources, cleaned, transformed, and normalized the data, and implemented time series analysis techniques to identify trends and patterns. Lastly, he developed a machine learning web application to predict Bangalore house prices based on user input and built a Flask-based REST API that acts as the backend server, handling HTTP requests from the frontend and serving house price predictions.

which college is Bhairav studying in?

Bhairav Ganguly is studying at Kalinga Institute of Industrial Technology. He is pursuing a B.Tech degree in Electronics and Computer Science Engineering.

4. **Remarks/Conclusion:-** 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 Student	Signature of the Lab Coordinator
Bhairav Ganguly	______________________________
(Name of the Student)	(Name of the Coordinator)