LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name:- SAHIL KUMAR

Roll No: 2230189

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

Spring 2024-2025

Table of Content

Exp No.	Title	Date of Experiment	Date of Submission	Remarks
1.	Resume using HTML/CSS	07|01|2025	13|01|2025	
2.	Cat and Dog Classification	14|01|2025	20|01|2025	
3.	Regression Analysis for Stock Prediction	21|01|2025	27|01|2025	
4.	Conversational Chatbot with Any Files	27|01|2025	09|02|2025	
5.				
6.				
7.				
8.				
9.	Open Ended 1			
10.	Open Ended 2			

Experiment Number	4
Experiment Title	Conversational Chatbot with Any Files
Date of Experiment	27|01|2025
Date of Submission	09|02|2025

1. **Objective:-**

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

2. **Procedure:- (Steps Followed)**

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

3. **Code:-**

Index.html file:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <link rel="stylesheet" href="styles/styles.css" />
    <title>AI Chatbot</title>
  </head>
  <body>
    <div class="container fade-in">
      <h1 class="title">AI Chatbot</h1>

      <div class="file-upload slide-in">
        <h2>Upload File</h2>
        <div class="upload-area" id="uploadArea">
          <input type="file" id="fileInput"
accept=".txt,.pdf,.csv" />
          <div class="upload-icon">📄</div>
          <p>Drag & drop or tap to upload</p>
        </div>
        <div id="preUploadPreview" class="pre-upload-preview">
          <div class="file-info">
            <div class="file-type-icon" id="fileTypeIcon"></div>
            <div class="file-name" id="preUploadFileName"></div>
          </div>
          <button class="remove-preview"
onclick="removePreUploadFile()">
            ×
          </button>
        </div>
        <div id="filePreview" class="file-preview">
          <div class="file-preview-header">
            <div class="file-preview-name" id="fileName"></div>
            <button class="remove-file" onclick="removeFile()">
              Remove File
            </button>
          </div>
          <div class="file-preview-content"
id="fileContent"></div>
        </div>
        <button onclick="uploadFile()"
class="pulse">Upload</button>
      </div>

      <div class="chat-container slide-in">
        <div class="model-selection">
          <h3>Select AI Model</h3>
          <select id="modelSelect" class="fancy-select">
            <option value="gemini">Gemini</option>
            <option value="groq">Groq</option>
            <option value="ollama">Ollama (Local)</option>
          </select>
        </div>

        <h3>Ask a Question</h3>
        <textarea
          id="questionInput"
          placeholder="Type your question here..."
          class="fancy-input"
        ></textarea>
        <button onclick="askQuestion()" class="send-button">
          <span>Send</span>
          <div class="button-icon">→</div>
        </button>

        <div class="loading" id="loading">
          <div class="loading-dots">
            <div class="dot"></div>
            <div class="dot"></div>
            <div class="dot"></div>
          </div>
          Processing...
        </div>

        <div id="response" class="response-area"></div>
      </div>
    </div>

    <script src="./styles/script.js"></script>
  </body>
</html>
```

Styles.css file:

```
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

@keyframes slideIn {
  from {
    transform: translateY(20px);
    opacity: 0;
  }
  to {
    transform: translateY(0);
    opacity: 1;
  }
}

@keyframes pulse {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.05);
  }
  100% {
    transform: scale(1);
  }
}

@keyframes bounce {
  0%,
  100% {
    transform: translateY(0);
  }
  50% {
    transform: translateY(-5px);
  }
}

@keyframes loadingDots {
  0%,
  80%,
  100% {
    transform: scale(0);
  }
  40% {
    transform: scale(1);
  }
}

body {
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  max-width: 800px;
  margin: 0 auto;
```

```css
  padding: 20px;
  background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
  min-height: 100vh;
}

.container {
  background-color: white;
  padding: 30px;
  border-radius: 16px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
  animation: fadeIn 0.8s ease-out;
}

.title {
  text-align: center;
  color: #2c3e50;
  margin-bottom: 30px;
  font-size: 2.5em;
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1);
}

.fade-in {
  animation: fadeIn 0.8s ease-out;
}

.slide-in {
  animation: slideIn 0.8s ease-out;
}

.file-upload {
  margin-bottom: 30px;
  padding: 20px;
  border-radius: 12px;
  background-color: #f8f9fa;
  transition: transform 0.3s ease;
}

.upload-area {
  border: 2px dashed #007bff;
  border-radius: 8px;
  padding: 20px;
  text-align: center;
  margin: 15px 0;
  cursor: pointer;
  transition: all 0.3s ease;
  position: relative;
  overflow: hidden;
}

.upload-area:hover {
  border-color: #0056b3;
  background-color: rgba(0, 123, 255, 0.05);
}

.upload-area input[type="file"] {
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  opacity: 0;
  cursor: pointer;
}

.upload-icon {
  font-size: 2em;
  margin-bottom: 10px;
  animation: bounce 2s infinite;
}

.chat-container {
  margin-top: 30px;
}

.model-selection {
  margin-bottom: 25px;
}

.fancy-select {
  width: 100%;
  padding: 12px;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  background-color: white;
  font-size: 1em;
  transition: all 0.3s ease;
```

```css
}

.fancy-select:focus {
  border-color: #007bff;
  box-shadow: 0 0 0 3px rgba(0, 123, 255, 0.25);
  outline: none;
}

textarea.fancy-input {
  width: 100%;
  height: 120px;
  padding: 15px;
  border: 2px solid #e0e0e0;
  border-radius: 12px;
  font-size: 1em;
  transition: all 0.3s ease;
  resize: vertical;
}

textarea.fancy-input:focus {
  border-color: #007bff;
  box-shadow: 0 0 0 3px rgba(0, 123, 255, 0.25);
  outline: none;
}

button {
  background: linear-gradient(135deg, #007bff 0%, #0056b3 100%);
  color: white;
  border: none;
  padding: 12px 25px;
  border-radius: 8px;
  cursor: pointer;
  font-size: 1em;
  font-weight: 600;
  transition: all 0.3s ease;
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 8px;
  width: 100%;
  max-width: 200px;
}
```

```css
button:hover {
  transform: translateY(-2px);
  box-shadow: 0 5px 15px rgba(0, 123, 255, 0.3);
}

.send-button {
  width: auto;
  margin-top: 10px;
}

.button-icon {
  transition: transform 0.3s ease;
}

.send-button:hover .button-icon {
  transform: translateX(5px);
}

.loading {
  display: none;
  text-align: center;
  margin: 20px 0;
  color: #666;
}

.loading-dots {
  display: flex;
  justify-content: center;
  gap: 8px;
  margin-bottom: 8px;
}

.dot {
  width: 8px;
  height: 8px;
  background-color: #007bff;
  border-radius: 50%;
  animation: loadingDots 1.4s infinite ease-in-out both;
}

.dot:nth-child(1) {
  animation-delay: -0.32s;
}
```

```css
.dot:nth-child(2) {
  animation-delay: -0.16s;
}

.response-area {
  margin-top: 20px;
  padding: 20px;
  border-radius: 12px;
  min-height: 100px;
  background-color: #ffffff;
  border: 2px solid #e0e0e0;
  transition: all 0.3s ease;
  line-height: 1.6;
  font-size: 16px;
  overflow-wrap: break-word;
  max-height: 500px;
  overflow-y: auto;
}

.response-area:not(:empty) {
  animation: slideIn 0.3s ease-out;
}

.response-area h1,
.response-area h2,
.response-area h3,
.response-area h4 {
  margin-top: 24px;
  margin-bottom: 16px;
  font-weight: 600;
  line-height: 1.25;
  color: #2c3e50;
}

.response-area h1 {
  font-size: 2em;
  border-bottom: 2px solid #eaecef;
  padding-bottom: 0.3em;
}
.response-area h2 {
  font-size: 1.5em;
  border-bottom: 1px solid #eaecef;
  padding-bottom: 0.3em;
```

```css
}
.response-area h3 {
  font-size: 1.25em;
}
.response-area h4 {
  font-size: 1em;
}

.response-area p {
  margin-bottom: 16px;
  line-height: 1.6;
}

.response-area ul,
.response-area ol {
  margin-bottom: 16px;
  padding-left: 24px;
}

.response-area li {
  margin-bottom: 8px;
}

.response-area code {
  background-color: #f6f8fa;
  padding: 2px 6px;
  border-radius: 4px;
  font-family: "Consolas", "Monaco", "Courier New", monospace;
  font-size: 0.9em;
  color: #476582;
}

.response-area pre {
  background-color: #f6f8fa;
  padding: 16px;
  border-radius: 8px;
  overflow-x: auto;
  margin: 16px 0;
  border: 1px solid #e0e0e0;
}

.response-area pre code {
  background-color: transparent;
  padding: 0;
  border-radius: 0;
  color: #476582;
  display: block;
  line-height: 1.5;
}

.response-area blockquote {
  border-left: 4px solid #007bff;
  padding: 12px 16px;
  margin: 16px 0;
  background-color: #f8f9fa;
  color: #2c3e50;
  font-style: italic;
}

.response-area strong {
  font-weight: 600;
  color: #2c3e50;
}

.response-area em {
  font-style: italic;
}

.response-area table {
  border-collapse: collapse;
  width: 100%;
  margin: 16px 0;
}

.response-area th,
.response-area td {
  border: 1px solid #e0e0e0;
  padding: 8px 12px;
  text-align: left;
}

.response-area th {
  background-color: #f6f8fa;
  font-weight: 600;
}
```

```css
.response-area tr:nth-child(even) {
  background-color: #f8f9fa;
}

.response-area::-webkit-scrollbar {
  width: 8px;
  height: 8px;
}

.response-area::-webkit-scrollbar-track {
  background: #f1f1f1;
  border-radius: 4px;
}

.response-area::-webkit-scrollbar-thumb {
  background: #cbd5e0;
  border-radius: 4px;
}

.response-area::-webkit-scrollbar-thumb:hover {
  background: #a0aec0;
}

.notification {
  position: fixed;
  bottom: 20px;
  right: 20px;
  padding: 15px 25px;
  border-radius: 8px;
  background: white;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
  transform: translateY(100px);
  opacity: 0;
  transition: all 0.3s ease;
}

.notification.show {
  transform: translateY(0);
  opacity: 1;
}

.notification.success {
  border-left: 4px solid #28a745;
```

```css
}

.notification.error {
  border-left: 4px solid #dc3545;
}

@media (max-width: 600px) {
  body {
    padding: 10px;
  }

  .container {
    padding: 15px;
  }

  .title {
    font-size: 1.8em;
    margin-bottom: 20px;
  }

  .file-upload {
    padding: 15px;
  }

  .upload-area {
    padding: 15px;
  }

  .upload-icon {
    font-size: 1.5em;
  }

  button {
    width: 100%;
    max-width: none;
    padding: 10px 20px;
  }

  .fancy-select,
  textarea.fancy-input {
    padding: 10px;
  }
  .notification {
    left: 10px;
    right: 10px;
    bottom: 10px;
    text-align: center;
  }
}

@media (max-width: 400px) {
  .container {
    padding: 10px;
  }

  .title {
    font-size: 1.5em;
  }

  .upload-area {
    padding: 10px;
  }
}

.fancy-input {
  width: calc(100% - 30px) !important;
  margin: 10px 0;
}

.file-preview {
  margin: 15px 0;
  padding: 15px;
  background-color: #f8f9fa;
  border: 1px solid #e0e0e0;
  border-radius: 8px;
  display: none;
}

.file-preview.show {
  display: block;
  animation: slideIn 0.3s ease-out;
}

.file-preview-header {
  display: flex;
```

```css
  justify-content: space-between;
  align-items: center;
  margin-bottom: 10px;
}

.file-preview-name {
  font-weight: 600;
  color: #2c3e50;
  word-break: break-all;
}

.remove-file {
  background: #dc3545;
  color: white;
  padding: 5px 10px;
  border-radius: 4px;
  cursor: pointer;
  font-size: 0.9em;
  max-width: 100px;
}

.remove-file:hover {
  background: #c82333;
}

.file-preview-content {
  max-height: 150px;
  overflow-y: auto;
  padding: 10px;
  background: white;
  border: 1px solid #e0e0e0;
  border-radius: 4px;
  font-family: monospace;
  font-size: 0.9em;
  white-space: pre-wrap;
  word-wrap: break-word;
}

@media (max-width: 600px) {
  .file-preview-header {
    flex-direction: column;
    gap: 10px;
    align-items: flex-start;
  }

  .remove-file {
    width: 100%;
    max-width: none;
    text-align: center;
  }
}

.pre-upload-preview {
  display: none;
  margin: 10px 0;
  padding: 10px;
  background-color: #fff;
  border: 1px solid #e0e0e0;
  border-radius: 8px;
  animation: slideIn 0.3s ease-out;
}

.pre-upload-preview.show {
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.file-info {
  display: flex;
  align-items: center;
  gap: 10px;
}

.file-type-icon {
  font-size: 1.5em;
  color: #2c3e50;
}

.file-name {
  font-size: 0.9em;
  color: #2c3e50;
  word-break: break-all;
}

.remove-preview {
  background: none;
  border: none;
  color: #dc3545;
  cursor: pointer;
  padding: 5px;
  font-size: 1.2em;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: color 0.3s ease;
}

.remove-preview:hover {
  color: #c82333;
}

@media (max-width: 600px) {
  .pre-upload-preview {
    padding: 8px;
  }

  .file-type-icon {
    font-size: 1.2em;
  }

  .file-name {
    font-size: 0.8em;
  }
}
```

script.js file

```javascript
let fileContent = "";

const uploadArea = document.getElementById("uploadArea");
const filePreview = document.getElementById("filePreview");
const fileNameElement = document.getElementById("fileName");
const                    fileContentElement                    =
document.getElementById("fileContent");
const                    preUploadPreview                      =
document.getElementById("preUploadPreview");
```

```
const preUploadFileName =
document.getElementById("preUploadFileName");
const fileTypeIcon = document.getElementById("fileTypeIcon");

document
  .getElementById("fileInput")
  .addEventListener("change", handleFileSelect);

function handleFileSelect(e) {
  const file = e.target.files[0];
  if (file) {
    showPreUploadPreview(file);
  }
}

function getFileIcon(fileType) {
  const icons = {
    "text/plain": "📝",
    "application/pdf": "📄",
    "text/csv": "📊",
    default: "📄",
  };
  return icons[fileType] || icons["default"];
}

function showPreUploadPreview(file) {
  const icon = getFileIcon(file.type);
  fileTypeIcon.textContent = icon;
  preUploadFileName.textContent = file.name;
  preUploadPreview.classList.add("show");
}

function removePreUploadFile() {
  document.getElementById("fileInput").value = "";
  preUploadPreview.classList.remove("show");
}

["dragenter", "dragover", "dragleave",
"drop"].forEach((eventName) => {
  uploadArea.addEventListener(eventName, preventDefaults, false);
});

function preventDefaults(e) {
  e.preventDefault();
  e.stopPropagation();
}

["dragenter", "dragover"].forEach((eventName) => {
  uploadArea.addEventListener(eventName, highlight, false);
});

["dragleave", "drop"].forEach((eventName) => {
  uploadArea.addEventListener(eventName, unhighlight, false);
});

function highlight(e) {
  uploadArea.classList.add("highlight");
}

function unhighlight(e) {
  uploadArea.classList.remove("highlight");
}

uploadArea.addEventListener("drop", handleDrop, false);

function handleDrop(e) {
  const dt = e.dataTransfer;
  const files = dt.files;

  if (files.length > 0) {
    document.getElementById("fileInput").files = files;
    showPreUploadPreview(files[0]);
  }
}

function showFilePreview(file) {
  fileNameElement.textContent = file.name;

  const reader = new FileReader();
  reader.onload = function (e) {
    const content = e.target.result;
    fileContentElement.textContent =
        content.slice(0, 500) + (content.length > 500 ? "..." :
"");
    filePreview.classList.add("show");
  };
  reader.readAsText(file);
}

function removeFile() {
  document.getElementById("fileInput").value = "";
  fileContent = "";
  filePreview.classList.remove("show");
  preUploadPreview.classList.remove("show");
  showNotification("File removed successfully", "success");
}

async function uploadFile() {
  const fileInput = document.getElementById("fileInput");
  const file = fileInput.files[0];
  const button = event.target;

  if (!file) {
    showNotification("Please select a file first", "error");
    return;
  }

  button.disabled = true;
  button.innerHTML = "Uploading...";

  const formData = new FormData();
  formData.append("file", file);

  try {
    const response = await fetch("/upload", {
      method: "POST",
      body: formData,
    });

    const data = await response.json();
    if (data.success) {
      fileContent = data.text;
      showFilePreview(file);
      preUploadPreview.classList.remove("show");
      showNotification("File uploaded successfully!", "success");
    } else {
      showNotification("Error: " + data.error, "error");
    }
  } catch (error) {
```

````
    showNotification("Error uploading file: " + error, "error");
  } finally {
    button.disabled = false;
    button.innerHTML = "Upload";
  }
}

function renderMarkdown(text) {
  text = text
    .replace(/### (.*$)/gim, "<h3>$1</h3>")
    .replace(/## (.*$)/gim, "<h2>$1</h2>")
    .replace(/# (.*$)/gim, "<h1>$1</h1>")
    .replace(/\*\*(.*?)\*\*/g, "<strong>$1</strong>")
    .replace(/\*(.*?)\*/g, "<em>$1</em>")
    .replace(/```([\s\S]*?)```/g, "<pre><code>$1</code></pre>")
    .replace(/`([^`]+)`/g, "<code>$1</code>")
    .replace(/^\s*-\s(.+)/gim, "<li>$1</li>")
    .replace(/(<li>.*<\/li>)/gims, "<ul>$1</ul>")
    .replace(/\n/g, "<br>");

  return text;
}

async function askQuestion() {
  if (!fileContent) {
    showNotification("Please upload a file first", "error");
    return;
  }

                          const            question            =
document.getElementById("questionInput").value;
  const model = document.getElementById("modelSelect").value;
  const loadingDiv = document.getElementById("loading");
  const responseDiv = document.getElementById("response");
  const button = event.target;

  if (!question) {
    showNotification("Please enter a question", "error");
    return;
  }

  loadingDiv.style.display = "flex";
  responseDiv.innerHTML = "";
````

```javascript
  button.disabled = true;

  try {
    const response = await fetch("/chat", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        question: question,
        context: fileContent,
        model: model,
      }),
    });

    const data = await response.json();
    if (data.response) {
      responseDiv.innerHTML = renderMarkdown(data.response);
    } else {
      responseDiv.innerHTML = "Error: " + data.error;
    }
  } catch (error) {
    responseDiv.innerHTML = "Error: " + error;
  } finally {
    loadingDiv.style.display = "none";
    button.disabled = false;
  }
}

function showNotification(message, type) {
  const notification = document.createElement("div");
  notification.className = `notification ${type}`;
  notification.textContent = message;

  document.body.appendChild(notification);

  setTimeout(() => {
    notification.classList.add("show");
  }, 10);

  setTimeout(() => {
    notification.classList.remove("show");
    setTimeout(() => {
      notification.remove();
    }, 300);
  }, 3000);
}

document.querySelectorAll("button").forEach((button) => {
  button.addEventListener("click", function (e) {
    const ripple = document.createElement("span");
    const rect = button.getBoundingClientRect();

    ripple.className = "ripple";
    ripple.style.left = `${e.clientX - rect.left}px`;
    ripple.style.top = `${e.clientY - rect.top}px`;

    button.appendChild(ripple);

    setTimeout(() => ripple.remove(), 600);
  });
});
```

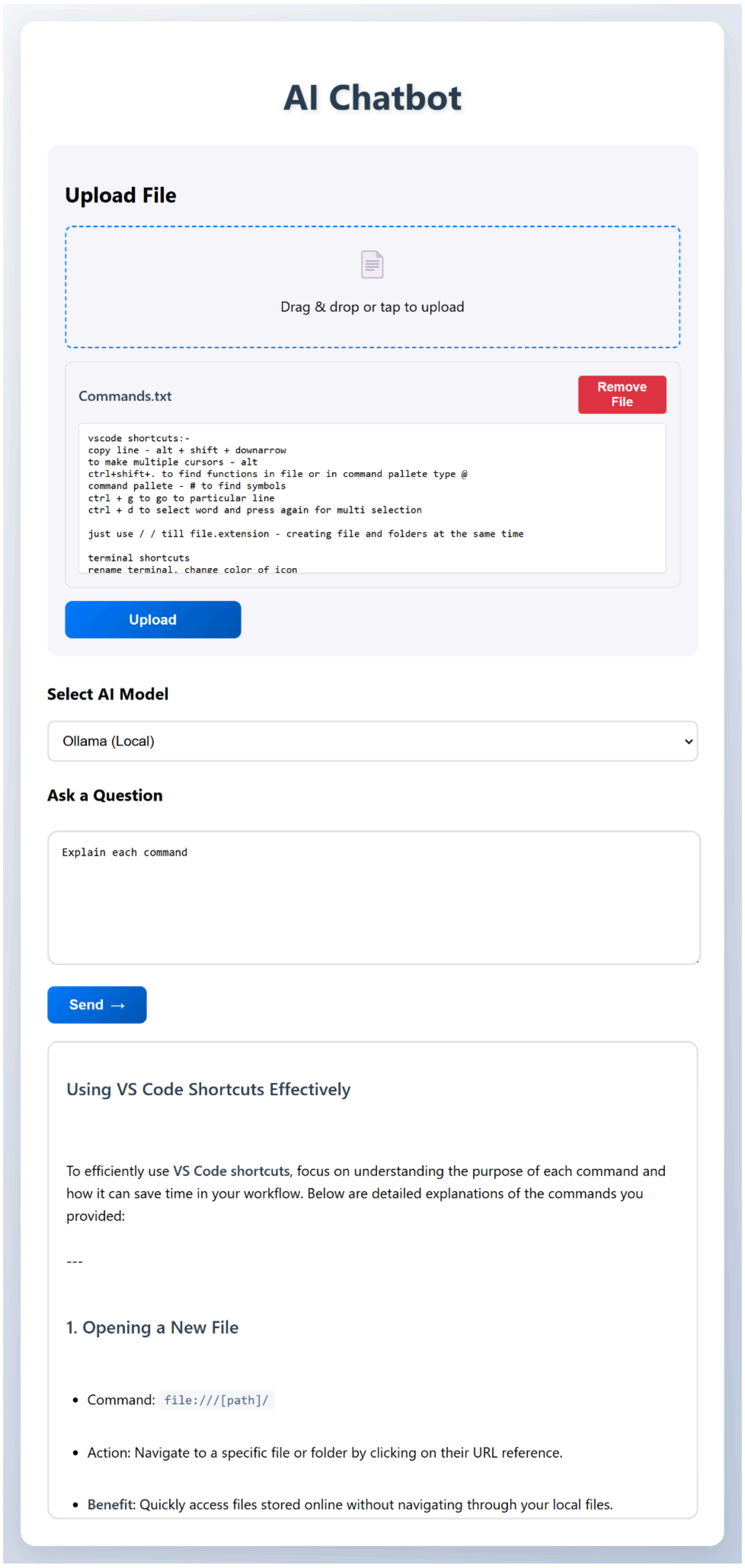
app.py:-

```
from flask import Flask, request, jsonify, render_template
import os
from werkzeug.utils import secure_filename
import pandas as pd
import PyPDF2
import google.generativeai as genai
from groq import Groq
from flask_cors import CORS
from dotenv import load_dotenv
import requests

load_dotenv('.env.local')

app        =        Flask(__name__,        static_folder='styles',
template_folder='.')
CORS(app)
```

```python
UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'csv'}
OLLAMA_API_URL = "http://localhost:11434/api/generate"

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def init_gemini():
    gemini_api_key = os.getenv('GEMINI_API_KEY')
    if not gemini_api_key:
        raise ValueError("GEMINI_API_KEY not found in environment
variables")
    genai.configure(api_key=gemini_api_key)
    return genai.GenerativeModel('gemini-pro')

def init_groq():
    groq_api_key = os.getenv('GROQ_API_KEY')
    if not groq_api_key:
        raise ValueError("GROQ_API_KEY not found in environment
variables")
    return Groq(api_key=groq_api_key)

try:
    gemini_model = init_gemini()
    groq_client = init_groq()
except Exception as e:
    print(f"Error initializing AI models: {str(e)}")

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower()
in ALLOWED_EXTENSIONS

def extract_text(file_path):
    file_extension = file_path.rsplit('.', 1)[1].lower()

    if file_extension == 'txt':
        with open(file_path, 'r', encoding='utf-8') as file:
            return file.read()

    elif file_extension == 'pdf':
        text = ""
        with open(file_path, 'rb') as file:
            pdf_reader = PyPDF2.PdfReader(file)
            for page in pdf_reader.pages:
                text += page.extract_text()
        return text

    elif file_extension == 'csv':
        df = pd.read_csv(file_path)
        return df.to_string()

def get_ollama_response(prompt):
    payload = {
        "model": "deepseek-r1:1.5b",
        "prompt": prompt,
        "stream": False
    }
    try:
        response = requests.post(OLLAMA_API_URL, json=payload)
        response.raise_for_status()
        return response.json()["response"]
    except requests.exceptions.ConnectionError:
          raise Exception("Could not connect to Ollama. Make sure
it's running (ollama run deepseek-r1:1.5b)")
    except Exception as e:
        raise Exception(f"Ollama API error: {str(e)}")

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

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
             filepath = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
```

```
        file.save(filepath)

        try:
            text_content = extract_text(filepath)
            return jsonify({'success': True, 'text':
text_content})
        except Exception as e:
            return jsonify({'error': str(e)}), 500

    return jsonify({'error': 'Invalid file type'}), 400

@app.route('/chat', methods=['POST'])
def chat():
    data = request.json
    if not data or 'question' not in data or 'context' not in
data or 'model' not in data:
        return jsonify({'error': 'Missing required fields'}), 400

    question = data['question']
    context = data['context']
    model_choice = data['model']

    prompt = f"""Context: {context}

Question: {question}

Please provide a clear, well-structured response following these
guidelines:
- Use markdown formatting for better readability
- Break down your answer into relevant sections
- Use bullet points or numbered lists where appropriate
- Add headings using markdown (e.g., ### Section Title) for
different parts of your answer
- Format any code snippets using markdown code blocks
- Use bold or italic text to emphasize important points

Answer:"""

    try:
        if model_choice == 'gemini':
            response = gemini_model.generate_content(prompt)
            answer = response.text
        elif model_choice == 'ollama':
            answer = get_ollama_response(prompt)
        else:
            completion = groq_client.chat.completions.create(
                messages=[
                         {"role": "system", "content": "You are a
helpful assistant that provides well-structured,
markdown-formatted responses with clear sections, headings, and
formatting."},
                    {"role": "user", "content": prompt}
                ],
                model="mixtral-8x7b-32768",
            )
            answer = completion.choices[0].message.content

        return jsonify({'response': answer})
    except Exception as e:
        return jsonify({'error': str(e)}), 500

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

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



Select AI Model: Ollama (Local)

```
vscode shortcuts:-
copy line - alt + shift + downarrow
to make multiple cursors - alt
ctrl+shift+. to find functions in file or in command pallete type @
command pallete - # to find symbols
ctrl + g to go to particular line
ctrl + d to select word and press again for multi selection

just use / / till file.extension - creating file and folders at the same time

terminal shortcuts
rename terminal. change color of icon
```

```
Explain each command
```

Using VS Code Shortcuts Effectively

To efficiently use VS Code shortcuts, focus on understanding the purpose of each command and how it can save time in your workflow. Below are detailed explanations of the commands you provided:

---

1. Opening a New File

- Command: `file:///[path]/`
- Action: Navigate to a specific file or folder by clicking on their URL reference.
- Benefit: Quickly access files stored online without navigating through your local files.

5. Remarks:-

Git - Github Repo

Signature of the Student	Signature of the Lab Coordinator
____________________	______________________________
(Name of the Student)	(Name of the Coordinator)