Mandem Pradeep

Assignment 1: Web Application Mimicking Google Sheets

Tech Stack:

Frontend: HTML, CSS, JavaScript (React for UI)

Backend: Flask (Python)

Database: SQLite (for storing spreadsheet data)

Libraries: Pandas (for data operations), OpenPyXL (Excel handling)

```
from flask import Flask, request, jsonify, render_template
import pandas as pd

app = Flask(_name_)

# Store spreadsheet data
spreadsheet_data = pd.DataFrame(columns=['A', 'B', 'C', 'D', 'E'])

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

@app.route('/update_cell', methods=['POST'])
def update_cell():
    data = request.json
    row, col, value = data['row'], data['col'], data['value']

    global spreadsheet_data
    spreadsheet_data.at[row, col] = value
    return jsonify({"message": "Cell updated successfully"})
```

```
@app.route('/calculate', methods=['POST'])
def calculate():
    data = request.json
    function, range_cells = data['function'], data['range']

    values = [spreadsheet_data.at[row, col] for row, col in range_cells if
pd.notna(spreadsheet_data.at[row, col])]

    if function == 'SUM':
        result = sum(values)
    elif function == 'AVERAGE':
        result = sum(values) / len(values) if values else 0
    elif function == 'MAX':
        result = max(values)
    elif function == 'MIN':
        result = min(values)
    elif function == 'COUNT':
        result = len(values)
    else:
        result = "Invalid function"

    return jsonify({"result": result})

if _name_ == '_main_':
    app.run(debug=True)
```

Assignment 2: Support Agent Chatbot for CDP

Tech Stack:

Backend: Flask (Python)

Libraries: BeautifulSoup (for web scraping), Requests (for fetching documentation), OpenAI GPT API (for natural language processing) Storage: JSON or SQLite (for storing pre-fetched documentation)

```
from flask import Flask, request, jsonify
import requests
from bs4 import BeautifulSoup

app = Flask(_name_)

CDP_DOCS = {
    "segment": "https://segment.com/docs/",
    "mparticle": "https://docs.mparticle.com/",
    "lytics": "https://docs.lytics.com/",
    "zeotap": "https://docs.zeotap.com/home/en-us/"
}

def fetch_documentation(cdp_name):
    """Fetch documentation from the given CDP URL."""
    url = CDP_DOCS.get(cdp_name.lower())
    if not url:
        return "CDP documentation not found."

    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    text_content = soup.get_text()
    return text_content[:1000]  # Limit text size for demo purposes

@app.route('/ask', methods=['POST'])
def ask_chatbot():
    data = request.json
    question = data.get("question")
```

```
    cdp = data.get("cdp").lower()

    if cdp not in CDP_DOCS:
        return jsonify({"answer": "Invalid CDP selected. Please choose from Segment, mParticle, Lytics, or Zeotap."})

    doc_text = fetch_documentation(cdp)
    return jsonify({"answer": f"Here’s relevant information from {cdp} documentation:\n{doc_text}"})

if _name_ == '_main_':
    app.run(debug=True)
```