Web Application for Digitalizing Group Accommodation

To develop a web application for digitalizing the hospitality process for group accommodation, follow these steps:

# 1. Setup Environment

Ensure you have the following tools and frameworks installed:

- Python 3.x

- Flask (for web framework)

- pandas (for CSV handling)

- HTML/CSS/JavaScript (for frontend)

- Bootstrap (for responsive design)

- Jinja2 (for templating)

#2. Project Structure

Create a project structure as follows:

group\_accommodation

│

├── app.py

├── templates

│ ├── index.html

│ ├── result.html

├── static

│ ├── styles.css

├── uploads

#3. CSV Files

Place your CSV files (Group Information and Hostel Information) in the `uploads` directory.

#4. Flask Application (app.py)

from flask import Flask, request, render\_template, redirect, url\_for

import pandas as pd

import os

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = 'uploads'

@app.route('/', methods=['GET', 'POST'])

def index():

if request.method == 'POST':

group\_file = request.files['group\_file']

hostel\_file = request.files['hostel\_file']

if group\_file and hostel\_file:

group\_path = os.path.join(app.config['UPLOAD\_FOLDER'], 'group\_info.csv')

hostel\_path = os.path.join(app.config['UPLOAD\_FOLDER'], 'hostel\_info.csv')

group\_file.save(group\_path)

hostel\_file.save(hostel\_path)

return redirect(url\_for('allocate\_rooms'))

return render\_template('index.html')

@app.route('/allocate', methods=['GET'])

def allocate\_rooms():

group\_df = pd.read\_csv(os.path.join(app.config['UPLOAD\_FOLDER'], 'group\_info.csv'))

hostel\_df = pd.read\_csv(os.path.join(app.config['UPLOAD\_FOLDER'], 'hostel\_info.csv'))

allocations = allocate(group\_df, hostel\_df)

result\_df = pd.DataFrame(allocations, columns=['Group ID', 'Hostel Name', 'Room Number', 'Members Allocated'])

result\_csv = os.path.join(app.config['UPLOAD\_FOLDER'], 'allocation\_result.csv')

result\_df.to\_csv(result\_csv, index=False)

return render\_template('result.html', tables=[result\_df.to\_html(classes='data')], titles=result\_df.columns.values)

def allocate(group\_df, hostel\_df):

allocations = []

for index, group in group\_df.iterrows():

group\_id = group['Group ID']

members = group['Members']

gender = group['Gender']

hostel\_rooms = hostel\_df[(hostel\_df['Capacity'] >= members) & (hostel\_df['Gender'] == gender)]

if not hostel\_rooms.empty:

room = hostel\_rooms.iloc[0]

allocations.append([group\_id, room['Hostel Name'], room['Room Number'], members])

hostel\_df = hostel\_df.drop(room.name)

return allocations

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

# 5. Frontend (HTML Templates)

\*\*index.html\*\*

<html>

<head>

<title>Group Accommodation</title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<div class="container">

<h2>Upload CSV Files</h2>

<form action="/" method="post" enctype="multipart/form-data">

<label for="group\_file">Group Information CSV:</label>

<input type="file" name="group\_file" id="group\_file" required>

<br>

<label for="hostel\_file">Hostel Information CSV:</label>

<input type="file" name="hostel\_file" id="hostel\_file" required>

<br>

<input type="submit" value="Upload">

</form>

</div>

</body>

</html>

\*\*result.html\*\*

<html>

<head>

<title>Room Allocations</title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<div class="container">

<h2>Room Allocations</h2>

{% for table in tables %}

{{ table|safe }}

{% endfor %}

<a href="{{ url\_for('static', filename='allocation\_result.csv') }}" download>Download CSV</a>

</div>

</body>

</html>

\*\*styles.css\*\*

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.container {

width: 50%;

margin: 0 auto;

text-align: center;

}

form {

display: flex;

flex-direction: column;

}

input[type="file"], input[type="submit"] {

margin: 10px 0;

}

table.data {

width: 100%;

border-collapse: collapse;

}

table.data th, table.data td {

border: 1px solid #ddd;

padding: 8px;

}

table.data th {

background-color: #f2f2f2;

}

# 6. Running the Application

1. \*\*Run the Flask app\*\*:

```bash

python app.py

```

2. \*\*Access the web application\*\*:

Open your web browser and go to `http://127.0.0.1:5000`.

# 7. Submission

Create a public repository on GitHub, push your code, and provide the link as required.

# Documentation

\*Logic\*:

- Upload CSV files for group and hostel information.

- Read CSV files using pandas.

- Allocate rooms ensuring group members stay together, within room capacities, and gender-specific accommodations.

- Display the allocation result on the web page and provide an option to download the CSV file.

\*Usage\*:

- Open the web application.

- Upload the Group Information CSV and Hostel Information CSV.

- Click upload to process the allocation.

- View the results on the results page and download the CSV file.

\*Instructions to Run\*:

1. Install Python and necessary libraries (`Flask`, `pandas`).

2. Clone the repository.

3. Navigate to the project directory.

4. Run the Flask app using `python app.py`.

5. Access the web application via your browser.

This should cover all the requirements and provide a comprehensive solution for the task.