# Hospitality Process Digitalization Web Application

## Introduction

This web application is designed to digitalize the hospitality process for group accommodation. It allows users to upload CSV files containing group and hostel information and allocates rooms efficiently while ensuring group members with the same ID stay together and adhere to hostel capacities and gender-specific accommodations.

## Features

- Upload two CSV files: one for group information and one for hostel information.

- An algorithm to allocate rooms based on group IDs, gender, and room capacity.

- Display allocated rooms and provide a downloadable CSV file with the allocation details.

## Technologies Used

- Backend: Flask (Python)

- Frontend: HTML, CSS, JavaScript

- Data Processing: Pandas (Python)

## Usage

### Running the Application

### Prerequisites

1. \*\*Python\*\*: Ensure you have Python installed. You can download it from [python.org](https://www.python.org/downloads/).

2. \*\*pip\*\*: Ensure you have pip installed. It usually comes with Python, but you can also install it using the instructions [here](https://pip.pypa.io/en/stable/installation/).

### Steps to Run the Application

1. \*\*Clone the Repository:\*\*

```sh  
 git clone <repository\_url>  
 cd hospitality\_process\_digitalization  
 ```

2. \*\*Set Up a Virtual Environment and Install Dependencies:\*\*

- Create and activate a virtual environment:

```sh  
 python -m venv venv  
 source venv/bin/activate # On Windows: venv\Scripts\activate  
 ```

- Install required packages:

```sh  
 pip install Flask pandas  
 ```

3. \*\*Start the Flask Application:\*\*

```sh  
 python backend/app.py  
 ```

4. \*\*Open Your Web Browser and Navigate to:\*\*

```  
 http://127.0.0.1:5000  
 ```

5. \*\*Uploading CSV Files:\*\*

- \*\*Upload Groups CSV File:\*\*

- Click on the "Choose File" button under "Upload Groups CSV."

- Select your `groups.csv` file from your local directory.

- Click the "Upload" button.

- \*\*Upload Hostels CSV File:\*\*

- Click on the "Choose File" button under "Upload Hostels CSV."

- Select your `hostels.csv` file from your local directory.

- Click the "Upload" button.

6. \*\*Allocating Rooms:\*\*

- After both files are uploaded, click the "Allocate Rooms" button.

- The application will process the files and allocate rooms based on the criteria.

- A CSV file named `allocation.csv` will be generated and downloaded automatically.

## Project Structure

```  
hospitality\_process\_digitalization/  
│  
├── backend/  
│ ├── app.py  
│ ├── uploads/  
│ │ └── (directory for uploaded files)  
│ └── requirements.txt  
│  
└── frontend/  
 ├── index.html  
 └── (additional frontend files if needed)  
```

## Backend (app.py)

Here is the complete `app.py` file:

```python  
from flask import Flask, request, jsonify, send\_file  
import pandas as pd  
import os  
  
app = Flask(\_\_name\_\_)  
UPLOAD\_FOLDER = 'uploads'  
os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)  
  
@app.route('/upload\_groups', methods=['POST'])  
def upload\_groups():  
 file = request.files['file']  
 filepath = os.path.join(UPLOAD\_FOLDER, 'groups.csv')  
 file.save(filepath)  
 return jsonify({"message": "Groups file uploaded successfully."})  
  
@app.route('/upload\_hostels', methods=['POST'])  
def upload\_hostels():  
 file = request.files['file']  
 filepath = os.path.join(UPLOAD\_FOLDER, 'hostels.csv')  
 file.save(filepath)  
 return jsonify({"message": "Hostels file uploaded successfully."})  
  
@app.route('/allocate\_rooms', methods=['GET'])  
def allocate\_rooms():  
 groups\_df = pd.read\_csv(os.path.join(UPLOAD\_FOLDER, 'groups.csv'))  
 hostels\_df = pd.read\_csv(os.path.join(UPLOAD\_FOLDER, 'hostels.csv'))  
   
 # Implement allocation logic here  
 # ...  
  
 # Save the result to a new CSV file  
 output\_filepath = os.path.join(UPLOAD\_FOLDER, 'allocation.csv')  
 allocation\_df.to\_csv(output\_filepath, index=False)  
   
 return send\_file(output\_filepath, as\_attachment=True, download\_name='allocation.csv')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)  
```

## Frontend (index.html)

Here is the complete `index.html` file:

```html  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Hospitality Process Digitalization</title>  
</head>  
<body>  
 <h1>Upload CSV Files</h1>  
 <form id="groups-form">  
 <label for="groups-file">Upload Groups CSV:</label>  
 <input type="file" id="groups-file" name="file" required>  
 <button type="submit">Upload</button>  
 </form>  
   
 <form id="hostels-form">  
 <label for="hostels-file">Upload Hostels CSV:</label>  
 <input type="file" id="hostels-file" name="file" required>  
 <button type="submit">Upload</button>  
 </form>  
  
 <button id="allocate-btn">Allocate Rooms</button>  
  
 <script>  
 document.getElementById('groups-form').addEventListener('submit', function(event) {  
 event.preventDefault();  
 let formData = new FormData();  
 formData.append('file', document.getElementById('groups-file').files[0]);  
 fetch('/upload\_groups', {  
 method: 'POST',  
 body: formData  
 }).then(response => response.json())  
 .then(data => alert(data.message));  
 });  
  
 document.getElementById('hostels-form').addEventListener('submit', function(event) {  
 event.preventDefault();  
 let formData = new FormData();  
 formData.append('file', document.getElementById('hostels-file').files[0]);  
 fetch('/upload\_hostels', {  
 method: 'POST',  
 body: formData  
 }).then(response => response.json())  
 .then(data => alert(data.message));  
 });  
  
 document.getElementById('allocate-btn').addEventListener('click', function() {  
 fetch('/allocate\_rooms').then(response => response.blob())  
 .then(blob => {  
 let url = window.URL.createObjectURL(blob);  
 let a = document.createElement('a');  
 a.href = url;  
 a.download = "allocation.csv";  
 document.body.appendChild(a);  
 a.click();  
 a.remove();  
 });  
 });  
 </script>  
</body>  
</html>  
```

## Requirements File (requirements.txt)

Create a `requirements.txt` file in the `backend` directory with the following content:

```  
Flask  
pandas  
```

## Conclusion

By following these instructions, you can set up and run the web application to digitalize the hospitality process for group accommodation. This guide covers the project structure, the necessary code for the backend and frontend, and the steps to run the application.