Usage

This code implements a Flask (Flask is a web framework used for building web applications in Python) web application that facilitates room allocation based on student and hostel data provided in CSV files and displays it.

The upload function checks if the request has files named ‘group_file.csv’ and ‘hostel_file.csv’. It also checks if any of the files are empty.

If the files are valid CSV files, they are saved to the upload folder with specific filenames (group_file.csv and hostel_file.csv).

The function then calls the ‘allocate_rooms’ function to process the files and generate a room allocation result.

The two csv files should have their names as group_file.csv and hostel_file.csv and their format should be as

group_file.csv :-

StudentID,GroupID,Gender

……

hostel_file.csv :-

HostelName,Gender,Capacity

….

Upon uploading the files, the application checks their validity (CSV format) and presence (group_file.csv and hostel_file.csv).

Logic

- The ‘allocate_rooms ‘ function is called to process the uploaded data.
- It reads the group and hostel data using pandas.
- It loops through each hostel record
- Checks the hostel's capacity and gender preference.
- Filters the group data to select students matching the hostel's gender and limits the number based on the capacity.
- The function creates an allocation record for each hostel with details like:

  Hostel Name

  List of allocated students (each student represented as a dictionary with details)

- Students allocated to a hostel are removed from the main group data to avoid duplicate allocations.
- Finally, the function returns a list containing allocation details for each hostel which is then displayed.

The following part of the code deals with the allocation process :-

```
def allocate_rooms (group_file_path, hostel_file_path):
    # Read the CSV files
    groups = pd.read_csv(group_file_path)
    hostels = pd.read_csv(hostel_file_path)

    allocation = []

    for hostel in hostels.itertuples():
        hostel_capacity = hostel.Capacity
        hostel_gender = hostel.Gender

        # Select students based on gender and availability
        allocated_students = groups[(groups['Gender'] ==
hostel_gender)].head(hostel_capacity)

        allocation.append({
            'Hostel': hostel.HostelName,
            'Allocated': allocated_students.to_dict('records')
        })

        # Remove allocated students from the group
        groups =
groups[~groups['StudentID'].isin(allocated_students['StudentID'])]

    return allocation
```