Hotel Room Reservation System

Problem Statement (As given by Unstop)

Design a hotel room reservation system that allows users to book rooms with the following constraints:

- Hotel has **97 rooms across 10 floors**
- Floors **1–9** have **10 rooms each**
- Floor **10** has **7 rooms**
- Booking must indicate room occupancy
- Previously booked rooms must be reflected correctly
- Room state transitions must be supported
- Visualization is required
- Additional customer interactions such as vacating & resetting should work correctly

Approach & Interpretation

To build an interactive and functional system consistent with real hotel workflows, the following assumptions and interpretation were made:

- Each room can be in two states:
    - **Available (Green)**
    - **Occupied (Red)**
- Booking is performed **manually** by specifying a room number
- Vacating a room returns it to **Available**
- Reset clears all reservations
- Random allocates one available room
- Visualization reflects hotel layout with:
    - **10th floor on top**
    - **1st floor at bottom**

System Architecture

The solution is implemented using **2-layer architecture**:

Frontend Layer

- HTML
- CSS
- JavaScript

Responsibilities:

✓ Visualization
✓ User interactions (Book, Random, Reset, Vacate)
✓ Popup Modal for Vacating
✓ Rendering floors & room occupancy

Backend Layer

- Python
- FastAPI

Responsibilities:

✓ Booking validation
✓ Vacating logic
✓ Random allocation
✓ State synchronization

Core Functional Features

✓ Book Room

- User enters room number (strict mode)
- Validation: room exists + not already occupied
- Single room booking (Booking group size 1)

✓ Random Booking

- Allocates one free room randomly

✓ Vacate Booking

- Click room → modal → confirm → unbook

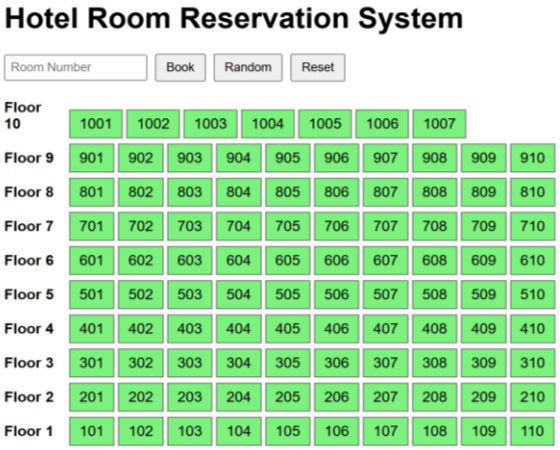
✓ Reset

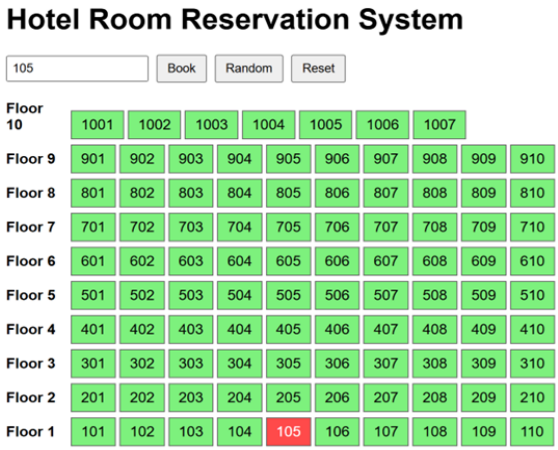
- Sets all rooms to green

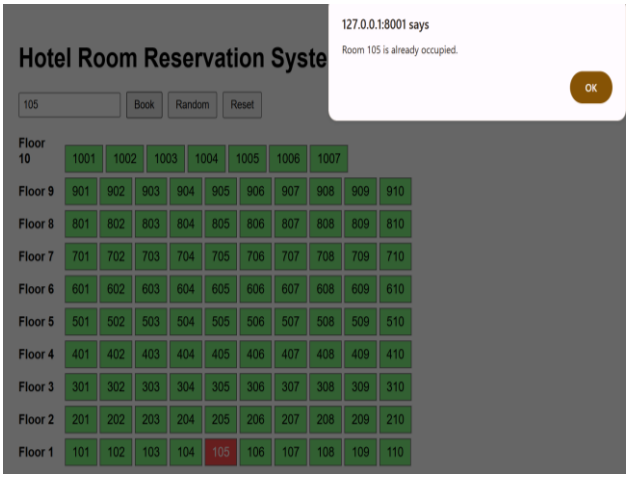
Hotel Layout & Visualization

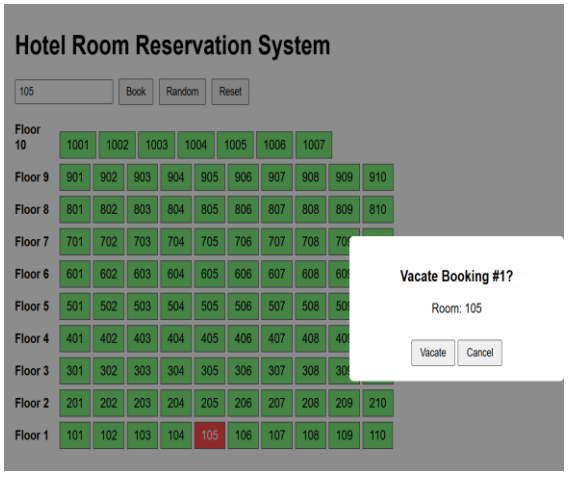
UI Rendering Principles:

- ➢ Top-down floors
- ➢ Horizontal left→right per floor
- ➢ Room color coded (Green/Red)
- ➢ Click interactions support vacate modal









Validation Rules:

Invalid room number	Reject
Room occupied	Reject
Room available	Allow booking
No rooms available	Reject random booking
Vacating	Turns red → green

Tech Stack:

Frontend	HTML, CSS, JS
Backend	Python

How to Run (Local Setup)

Backend

- cd backend
- pip install -r requirements.txt
- python -m uvicorn main:app –reload
- Service runs at: http://127.0.0.1:8000

Frontend

- Cd frontend
- `python -m http.server 8001`
- Open browser: http://127.0.0.1:8001/index.html

User Interaction Flow

```
1. Start application
2. UI renders all rooms green
3. User types room number → Book → changes to red
4. User clicks red room → modal → Vacate
5. User triggers Random → allocates new room
6. User presses Reset → clears state
```

Edge Cases Handled

- Book non-existing room (Reject)
- Book occupied room (Reject)
- Random when full (Reject)

- Vacate non-existing booking (Safe no-op)

Future Scope

Potential improvements include:

- Customer Info + Profile
- Check-in/Check-out timestamps
- Pricing & Billing
- Booking history
- Search room
- Occupancy analytics
- SQLite / PostgreSQL storage
- Admin panel
- Mobile UI
- Cloud deployment (Render + Vercel)
- Authorization & Security

s