**Visitors Hostel (OS1)**

Kaif Gowani Mandalapu Vijaya Durga Saladi Manikyamba Harshitha

**AI-Driven Approach for Fusion**

**1. Using Copilot**

Task 3: Database Schema and Backend Code

For Visitors hostel there are three users Intender , caretaker , incharge . The intended use of the system by its users is as follows: where An intender can book rooms by submitting a booking form in the required format. They can also book rooms offline by calling and informing the caretaker. They can view the status of their booking

requests(confirmed, pending or cancelled) and receive notifications regarding approval or rejection of bookings. Intenders can also modify or cancel their booking requests and receive a final confirmation. They can also check history of bookings and print booking info. The Intender also settles the bill offline by paying the required payable amount to the caretaker and collects the bill. The caretaker will review the bookings requested by the intenders and he may cancel the booking or he can forward the request to the In-Charge for approval of the booking.Caretaker also manages the offline room bookings and updates the system accordingly.He also keeps track of the guests check in,settle the payment(checking-out)and updates the database.He also manages the inventory and gets alerts if a particular inventory goes below the threshold limit.He also gets alerts regarding the guests who didn’t check-in. Incharge is responsible for receiving notifications of requests from caretaker and accepting / rejecting booking request , to manage offline bookings , to manage inventory

,check bills , meal bookings , Access summarized reports , generate reports ." give database schema for this

**Response :**

**Tables and Their Structures**

1. **Users**

○ user\_id (Primary Key): Integer

○ user\_type: Enum ('Intender', 'Caretaker', 'Incharge')

○ name: Varchar

○ email: Varchar

○ phone: Varchar

○ password: Varchar

2. **Rooms**

○ room\_id (Primary Key): Integer

○ room\_number: Varchar

○ type: Varchar

○ capacity: Integer

○ status: Enum ('Available', 'Booked', 'Under Maintenance')

3. **Bookings**

○ booking\_id (Primary Key): Integer

○ user\_id (Foreign Key): Integer

○ room\_id (Foreign Key): Integer

○ booking\_date: DateTime

○ check\_in\_date: DateTime

○ check\_out\_date: DateTime

○ status: Enum ('Pending', 'Confirmed', 'Cancelled')

○ approval\_date: DateTime

○ approval\_by (Foreign Key to Users): Integer

○ payment\_status: Enum ('Pending', 'Paid')

4. **Payments**

○ payment\_id (Primary Key): Integer

○ booking\_id (Foreign Key): Integer

○ amount: Decimal

○ payment\_date: DateTime

○ payment\_method: Varchar

5. **Notifications**

○ notification\_id (Primary Key): Integer

○ user\_id (Foreign Key): Integer

○ booking\_id (Foreign Key): Integer

○ message: Text

○ notification\_date: DateTime

○ status: Enum ('Unread', 'Read')

6. **Inventory**

○ inventory\_id (Primary Key): Integer

○ item\_name: Varchar

○ quantity: Integer

○ threshold\_limit: Integer

7. **Inventory\_Alerts**

○ alert\_id (Primary Key): Integer

○ inventory\_id (Foreign Key): Integer

○ alert\_date: DateTime

○ message: Text

○ status: Enum ('Unresolved', 'Resolved')

8. **Guests**

○ guest\_id (Primary Key): Integer

○ booking\_id (Foreign Key): Integer

○ name: Varchar

○ check\_in\_date: DateTime

○ check\_out\_date: DateTime

○ status: Enum ('Checked In', 'Checked Out', 'No Show')

9. **Reports**

○ report\_id (Primary Key): Integer

○ user\_id (Foreign Key): Integer

○ report\_date: DateTime

○ summary: Text

**Relationships**

● Users to Bookings: One-to-Many (One user can have multiple bookings) ● Rooms to Bookings: One-to-Many (One room can be booked multiple times) ● Bookings to Payments: One-to-One (Each booking has one payment) ● Users to Notifications: One-to-Many (One user can receive multiple notifications) ● Inventory to Inventory\_Alerts: One-to-Many (One inventory item can have multiple alerts)

● Bookings to Guests: One-to-Many (One booking can have multiple guests) ● Users to Reports: One-to-Many (One user can generate multiple reports)

2. Backend Code - Develop complete working Backend Code in Django for any one actor.

https://drive.google.com/file/d/1bQaCiU8iCBnQ2\_edqy-y6qbggEbQME5y/view?usp=sharing **2. Using Chatgpt**

1. Design Database Schema - Use the Problem statement to create the database schema.

**-- USERS TABLE: Stores information about intenders, caretakers, and in-charge CREATE TABLE Users (**

**user\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**username VARCHAR(100) UNIQUE NOT NULL,**

**password VARCHAR(255) NOT NULL,**

**role ENUM('intender', 'caretaker', 'incharge') NOT NULL,**

**email VARCHAR(255) UNIQUE NOT NULL,**

**phone\_number VARCHAR(15),**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**-- ROOMS TABLE: Stores information about rooms**

**CREATE TABLE Rooms (**

**room\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**room\_number VARCHAR(50) UNIQUE NOT NULL,**

**room\_type VARCHAR(50) NOT NULL,**

**capacity INT NOT NULL,**

**status ENUM('available', 'booked', 'offline') NOT NULL,**

**price\_per\_day DECIMAL(10, 2) NOT NULL**

**);**

**-- BOOKINGS TABLE: Stores booking requests by intenders**

**CREATE TABLE Bookings (**

**booking\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**intender\_id INT NOT NULL,**

**room\_id INT NOT NULL,**

**booking\_date DATE NOT NULL,**

**check\_in\_date DATE NOT NULL,**

**check\_out\_date DATE NOT NULL,**

**status ENUM('confirmed', 'pending', 'cancelled') DEFAULT 'pending', total\_amount DECIMAL(10, 2),**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**FOREIGN KEY (intender\_id) REFERENCES Users(user\_id), FOREIGN KEY (room\_id) REFERENCES Rooms(room\_id) );**

**-- PAYMENTS TABLE: Stores payment information**

**CREATE TABLE Payments (**

**payment\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**booking\_id INT NOT NULL,**

**intender\_id INT NOT NULL,**

**payment\_date DATE NOT NULL,**

**amount\_paid DECIMAL(10, 2) NOT NULL,**

**payment\_method ENUM('cash', 'online') NOT NULL,**

**FOREIGN KEY (booking\_id) REFERENCES Bookings(booking\_id), FOREIGN KEY (intender\_id) REFERENCES Users(user\_id) );**

**-- INVENTORY TABLE: Manages inventory items**

**CREATE TABLE Inventory (**

**item\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**item\_name VARCHAR(100) NOT NULL,**

**quantity INT NOT NULL,**

**threshold INT NOT NULL,**

**last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP**

**);**

**-- MEAL BOOKINGS TABLE: Tracks meal orders**

**CREATE TABLE MealBookings (**

**meal\_booking\_id INT AUTO\_INCREMENT PRIMARY KEY, intender\_id INT NOT NULL,**

**meal\_type ENUM('breakfast', 'lunch', 'dinner') NOT NULL, booking\_date DATE NOT NULL,**

**quantity INT NOT NULL,**

**total\_cost DECIMAL(10, 2),**

**FOREIGN KEY (intender\_id) REFERENCES Users(user\_id) );**

**-- ALERTS TABLE: Logs alerts for inventory or no-shows**

**CREATE TABLE Alerts (**

**alert\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**alert\_type ENUM('inventory', 'no-show') NOT NULL,**

**alert\_message TEXT NOT NULL,**

**alert\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**-- REPORTS TABLE: Tracks generated reports**

**CREATE TABLE Reports (**

**report\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**report\_type ENUM('booking\_summary', 'financial\_summary', 'inventory\_status') NOT NULL,**

**generated\_by INT NOT NULL,**

**report\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (generated\_by) REFERENCES Users(user\_id)**

**);**

**-- BOOKING HISTORY VIEW: Tracks booking history**

**CREATE VIEW BookingHistory AS**

**SELECT**

**b.booking\_id,**

**b.intender\_id,**

**b.room\_id,**

**r.room\_number,**

**b.booking\_date,**

**b.check\_in\_date,**

**b.check\_out\_date,**

**b.status,**

**b.total\_amount**

**FROM Bookings b**

**JOIN Rooms r ON b.room\_id = r.room\_id;**

**-- GUEST CHECK-IN TRACKING TABLE**

**CREATE TABLE CheckIns (**

**checkin\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**booking\_id INT NOT NULL,**

**check\_in\_time TIMESTAMP,**

**check\_out\_time TIMESTAMP,**

**FOREIGN KEY (booking\_id) REFERENCES Bookings(booking\_id) );**

2. Backend Code - Develop complete working Backend Code in Django for any one actor.

**https://drive.google.com/file/d/1RhmlewxPLIY0UcBtdFkhqDp3KFPNu2aX/view?usp=shari ng**