

**Database Systems Project**

**Submitted To :** MS Kainat nazir

**Program :** BS Computer Science

**Batch & Section :** BSCSev-F24-B

**Project Name :** FlateMate

**Submitted By :**

Muhammad Ayan Ali (242746)

Muhammad Bilal Nasir (242755)

# **Project Title**

**Flatemate** (A Hostel Booking Platform) +

**Flatemate\_admin** (Admin View For Bookings And Hostel Management)

FlateMate is a hostel booking Platform (Website) that allows users to reserve rooms online through a user-friendly interface. It features a Search bar to filter hostel according to your needs, pop-up booking form, real-time data storage using PHP and MySQL, and an admin dashboard for managing guests, bookings, hostels, and room allocations effectively via a local XAMPP server.

# **Scope Of Project**

**(The scope of this project involves creating a comprehensive database management system tailored to meet the specific needs of that you decided. The project will encompass the design and implementation, ensuring a functional and efficient database system) 🡨 YOURS.**

The **FlateMate** Website is developed to provide a digital solution for hostel room bookings across various cities. The software offers the following functionalities:

* Enables users to view hostels, check availability, and book rooms through an interactive web interface.
* Stores and processes user booking data securely using PHP and MySQL.
* Allows hostel administrators to manage bookings, guest information, and room allocations.
* Provides a dedicated admin dashboard with real-time insights into hostel activity, earnings, and guest history.
* Automates room allocations and updates guest counts through backend database triggers.

This system is implemented using **HTML, CSS, JavaScript (Frontend)**, **PHP (Backend)**, and **MySQL (Database)**, all hosted on a local **XAMPP server**.

# **Functional Requirements**

The functional requirements of the system are as follows:

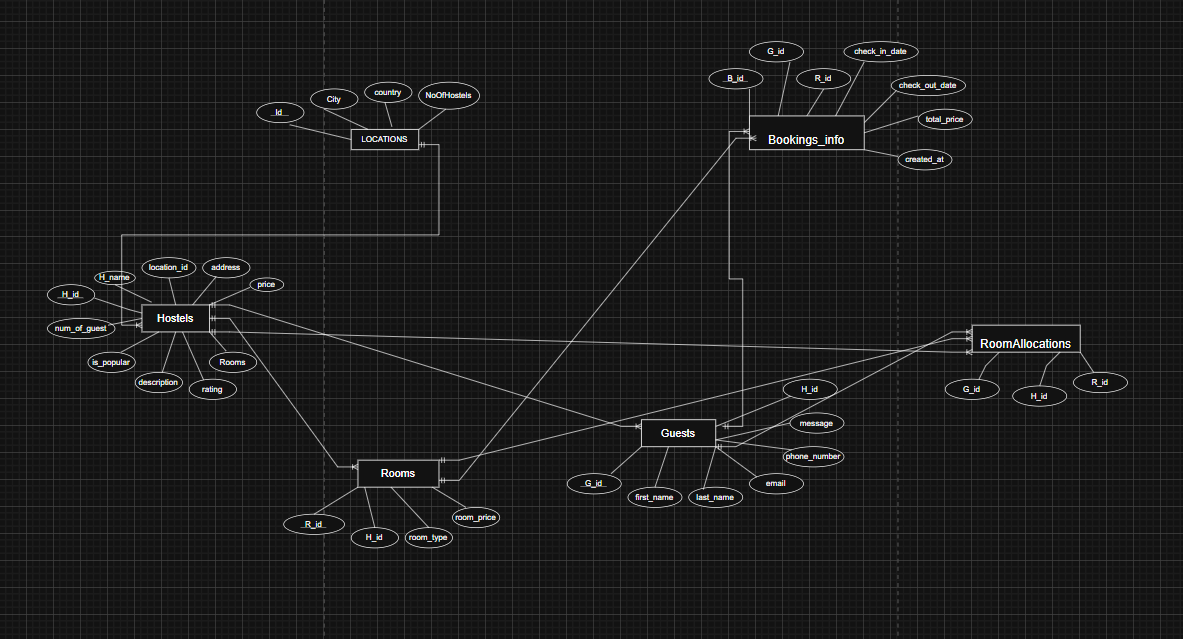
1. **User Booking Interface**
   * Users can view available hostels and room details.
   * A pop-up form captures user details and booking preferences.
   * The form data is sent to the backend via a PHP script for processing.

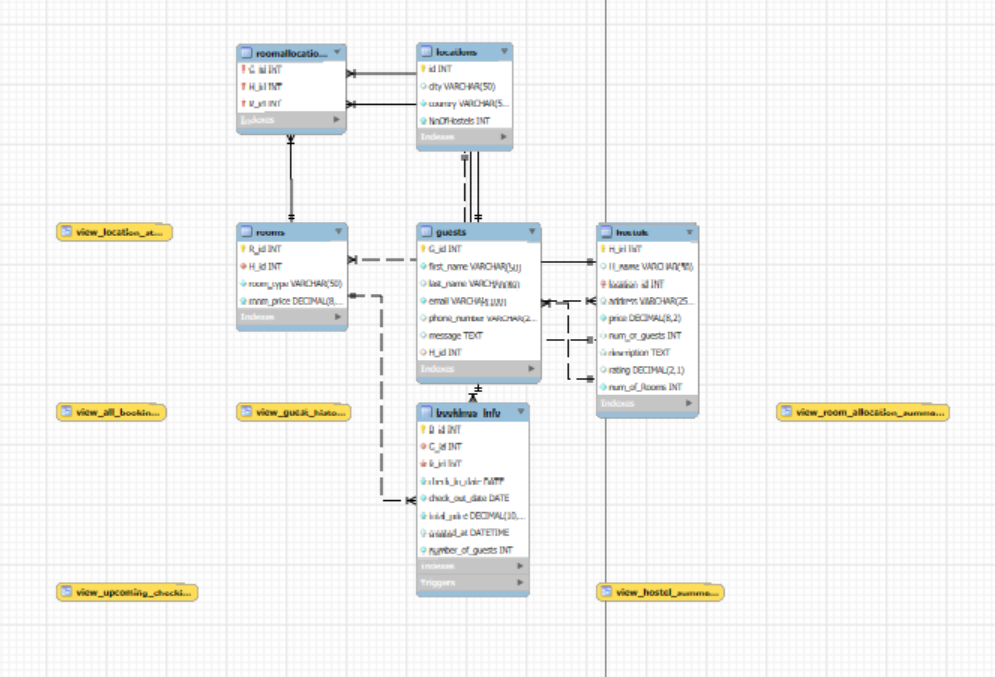
1. **Database Interaction**
   * The system stores guest, booking, and room allocation data in MySQL.
   * Triggers handle automatic updates, such as incrementing guest counts and assigning rooms.
2. **Admin Dashboard**
   * Admins can view bookings, guest history, earnings, and room allocation summaries.
   * Dynamic views provide up-to-date statistics and summaries.
   * Navigation between views is handled via query parameters.
3. **Automation**
   * Room allocations and booking confirmations are automatically handled after form submission.
   * Admins do not need to manually assign rooms or track guest counts.

**YOURS**

* **Data Management: Ability to add, update, delete, and retrieve data.**
* **Data Integrity and Validation: Ensure data consistency and accuracy.**
* **Scalability and Performance: Design the system to handle increasing amounts of data efficiently.**
* **Frontend Interface: Develop a user-friendly interface for interacting with the database**

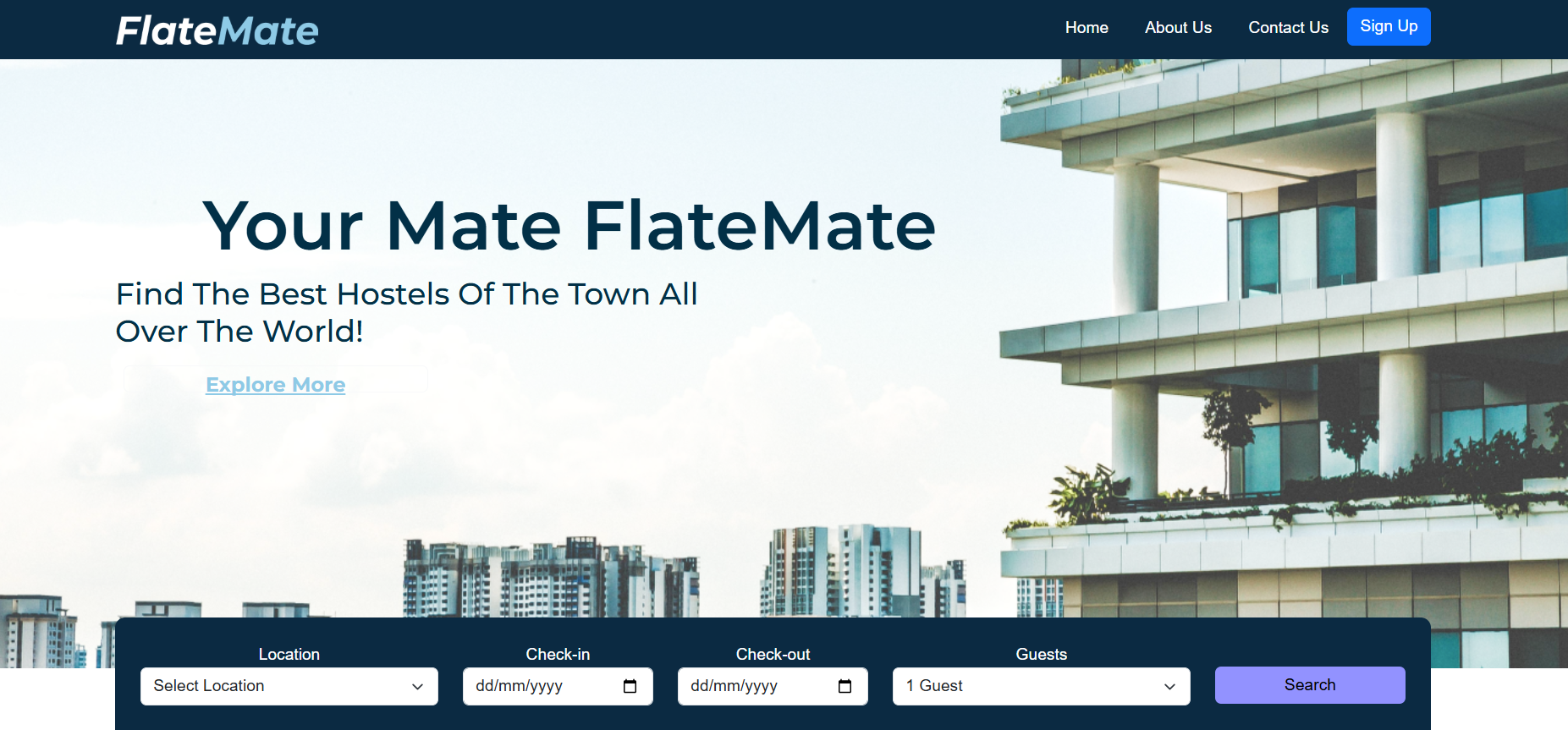
# **Entity-Relationship Diagram (ERD)**



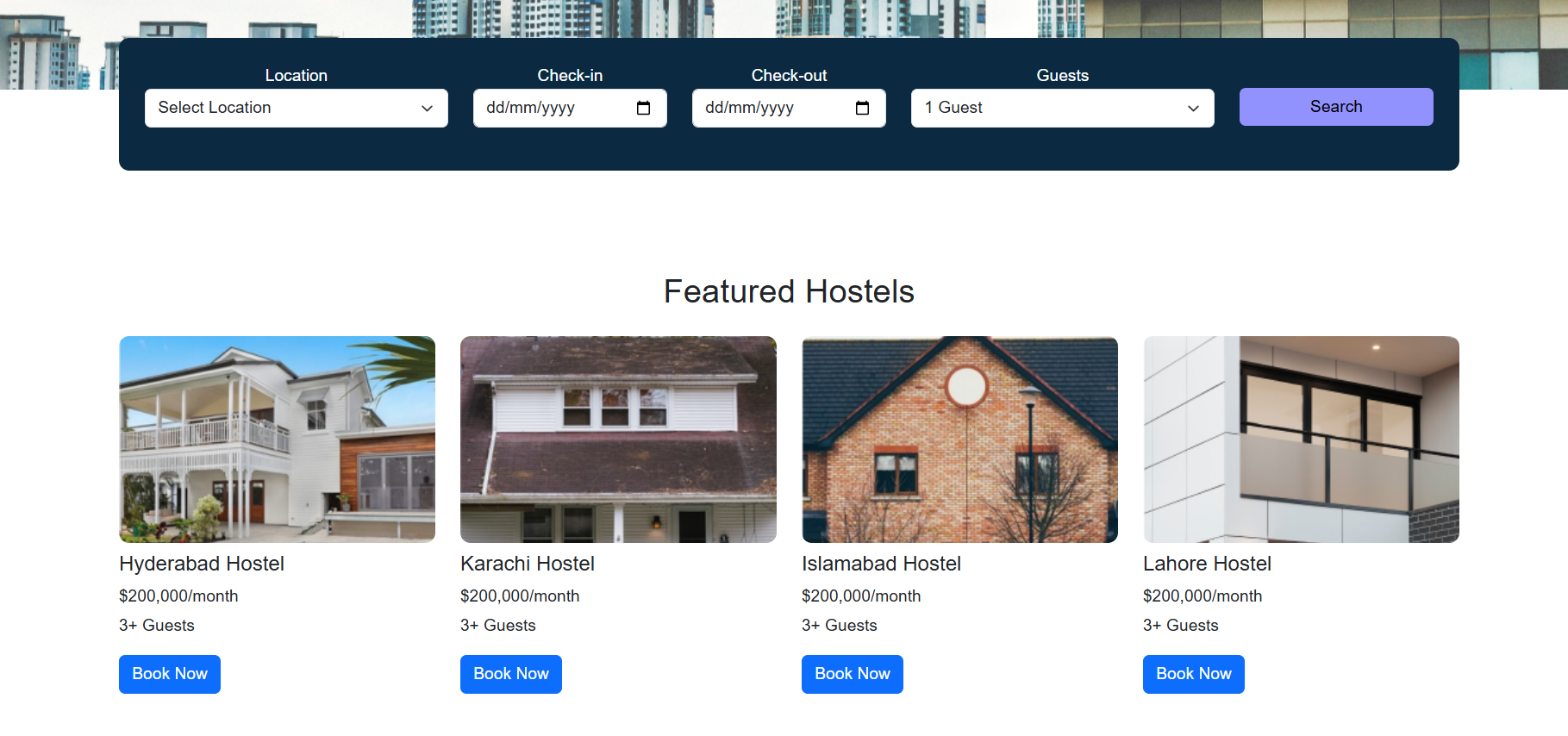
****

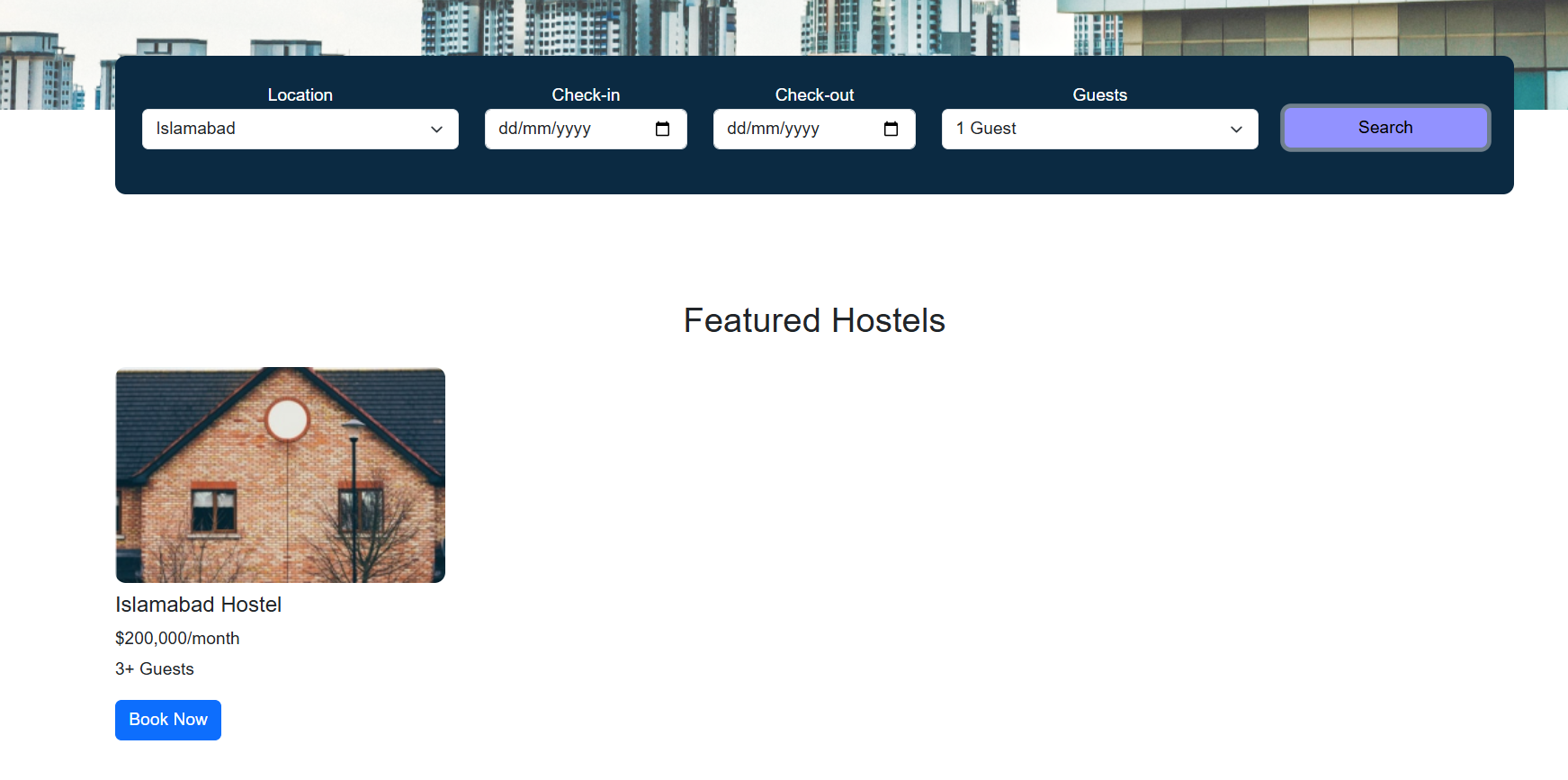
# **Screenshots Of Project**

**Landing Page**

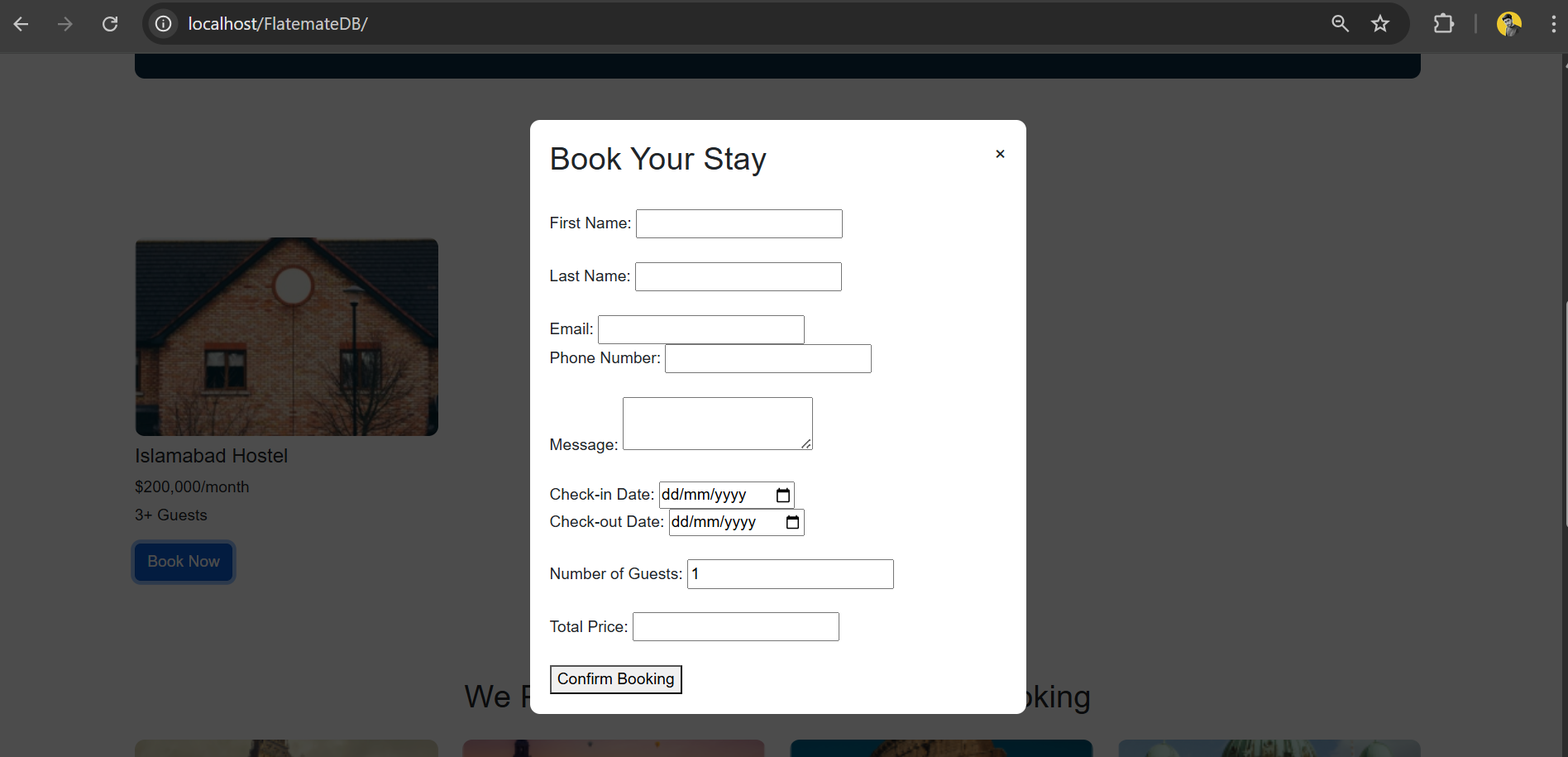
****

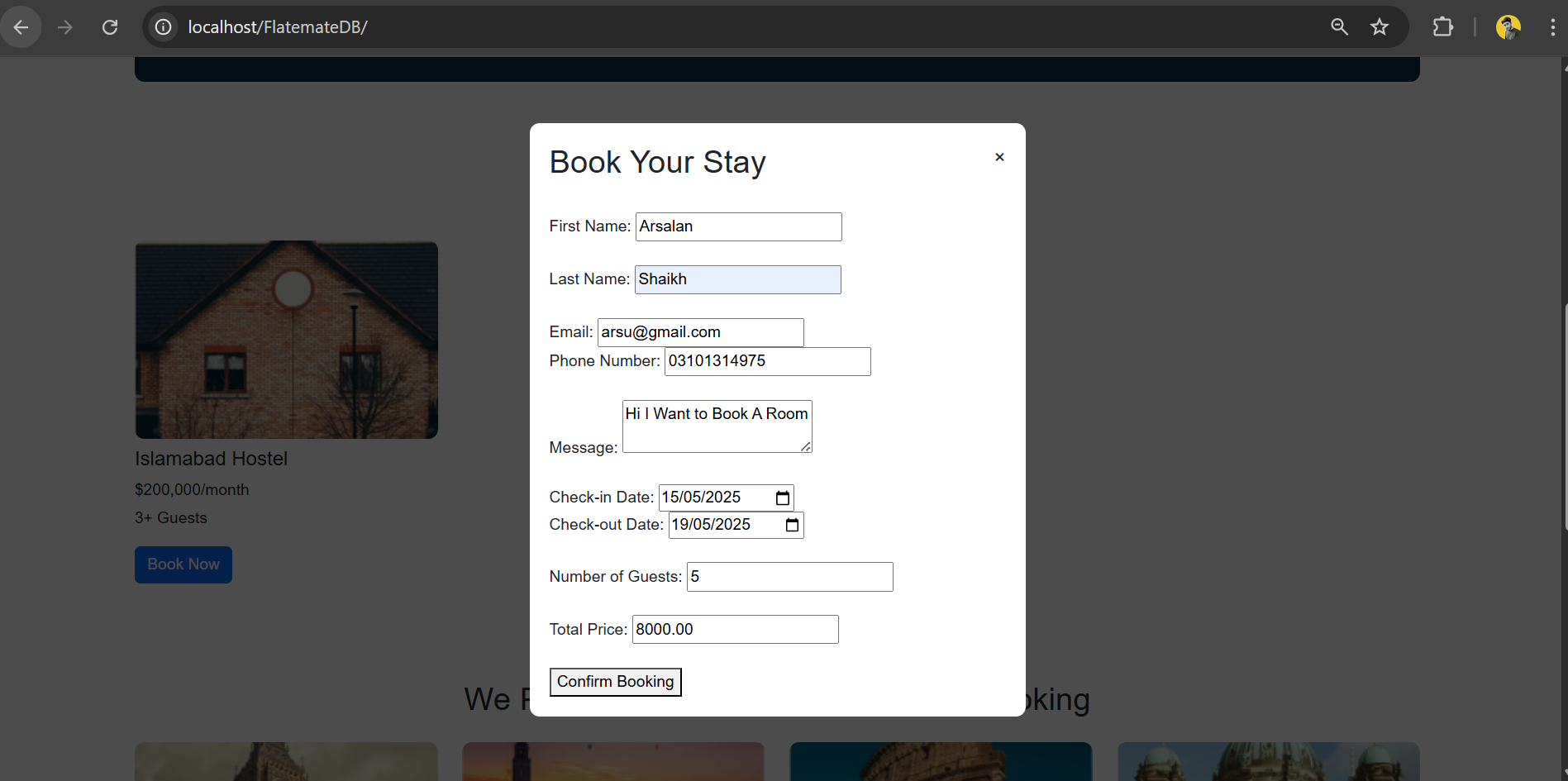
**Search Bar Feature**

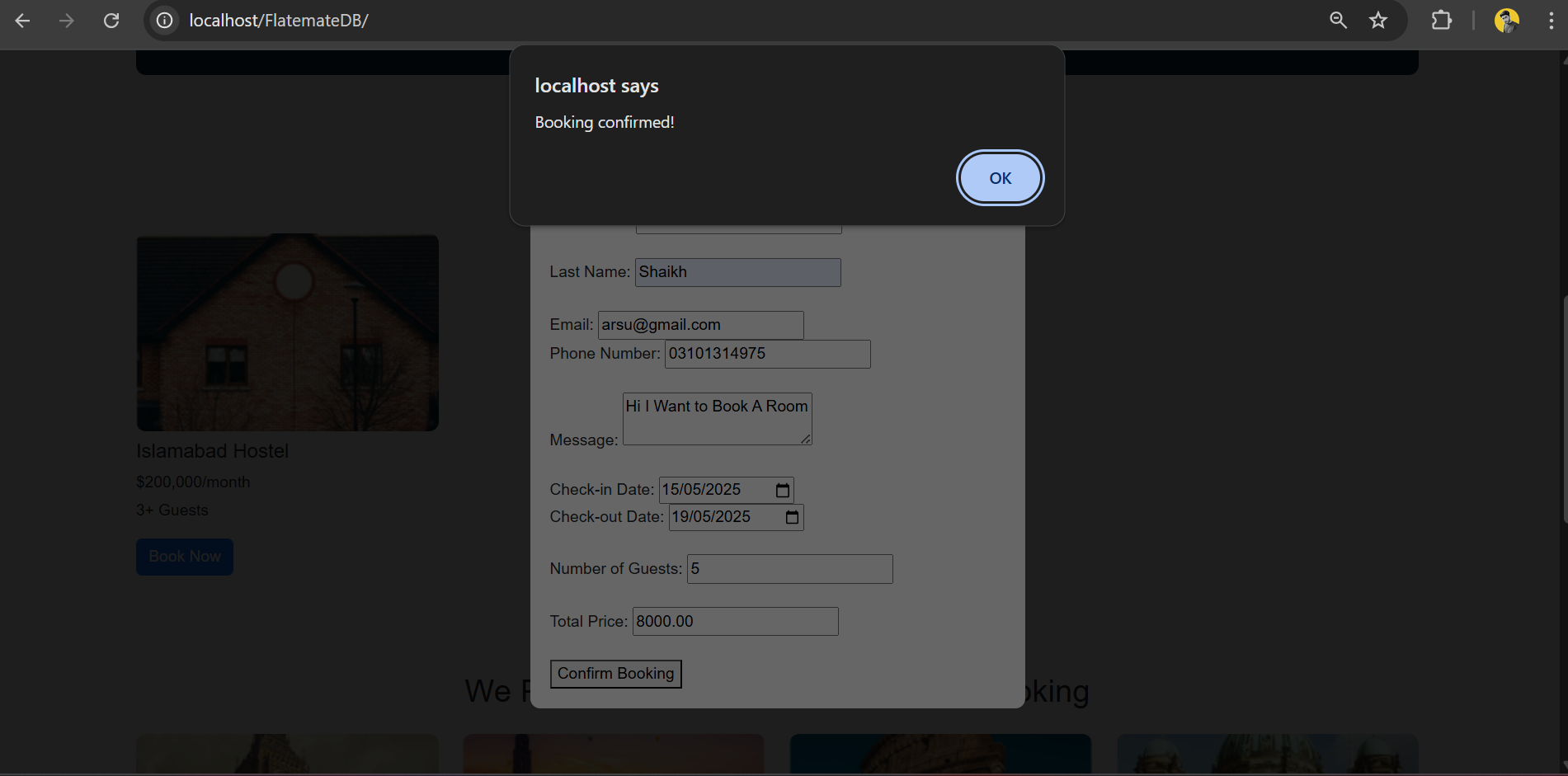
****

****

**Booking Form**

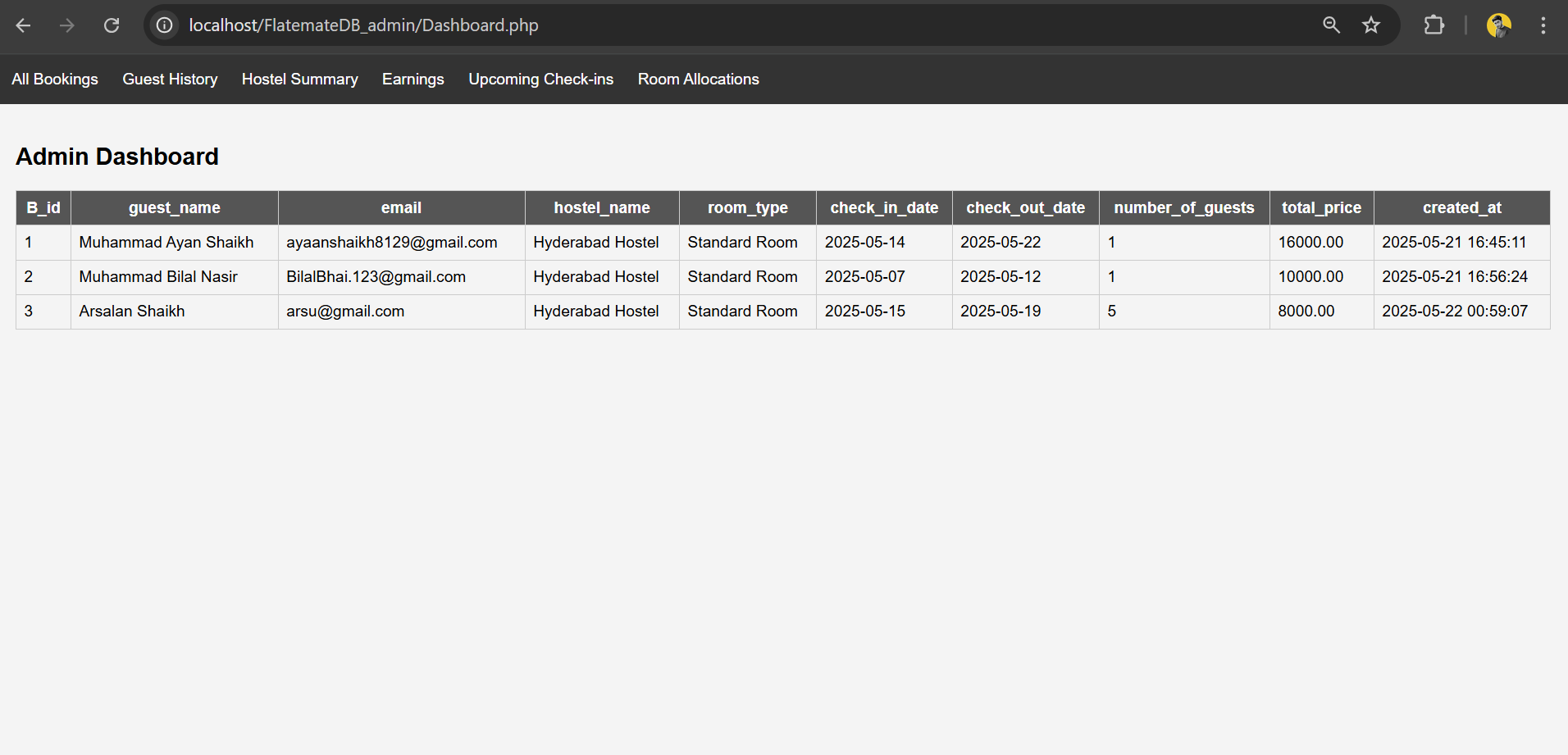
****

****

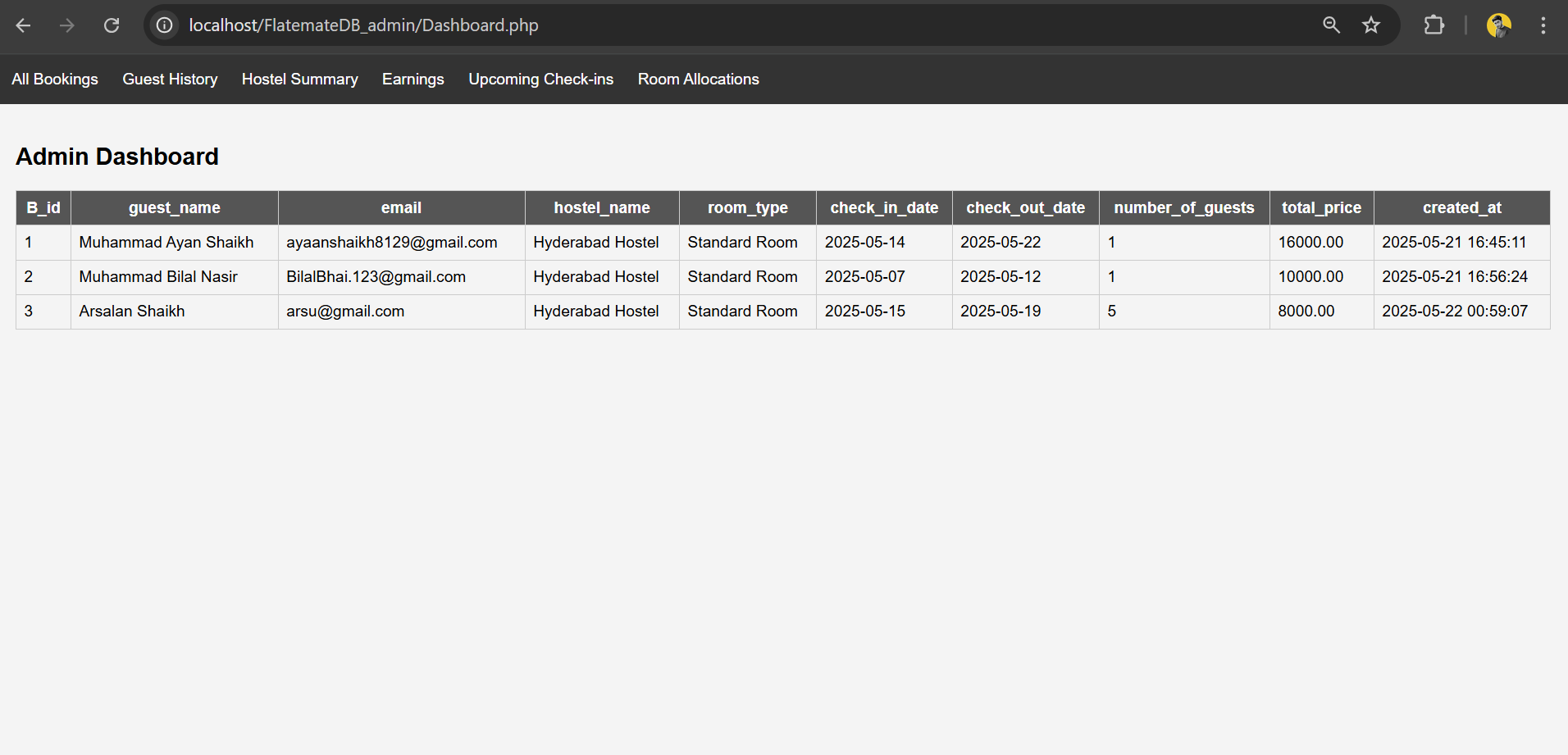
****

**Admin Dashbaord**

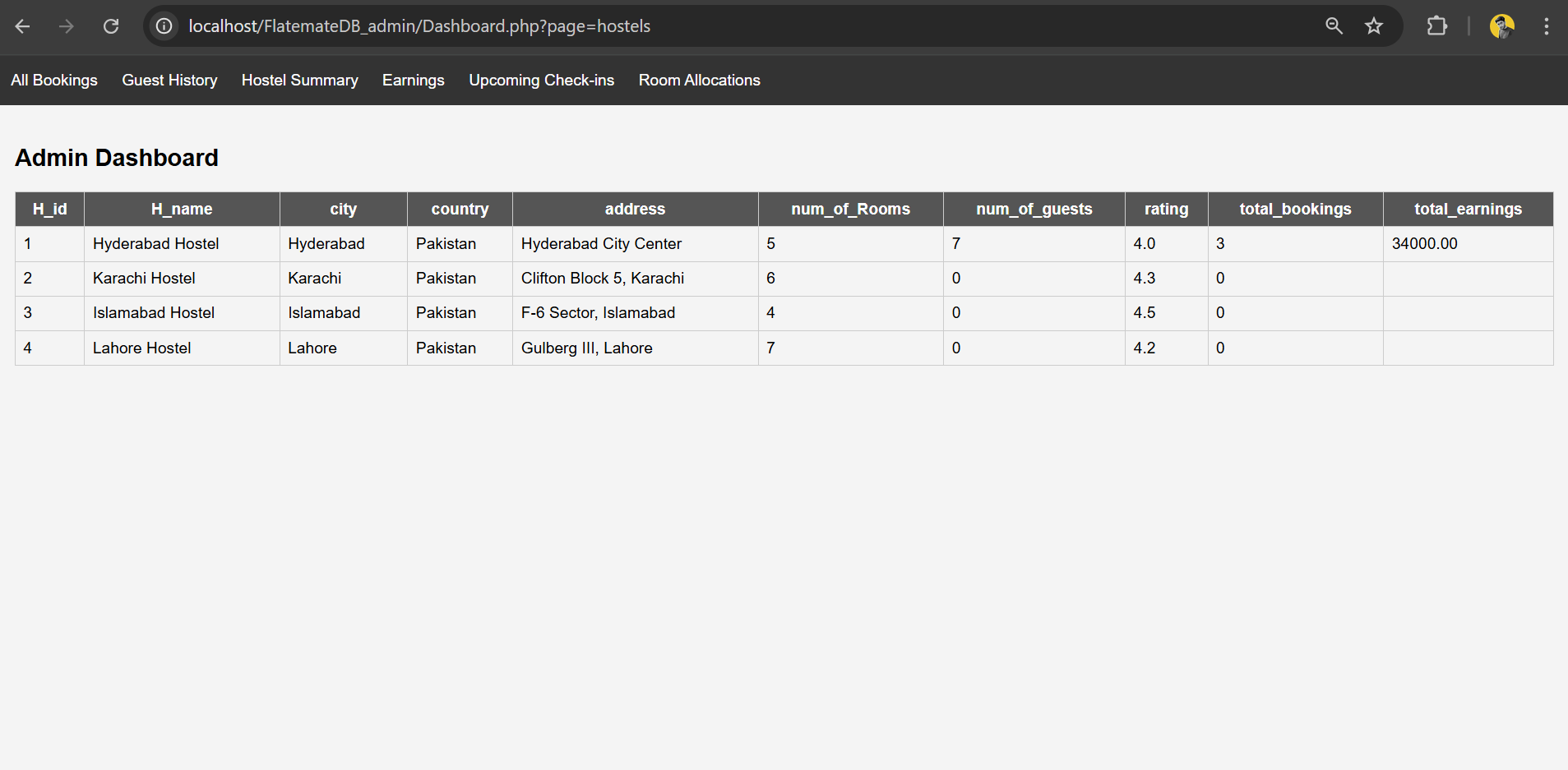
**All Bookings**

****

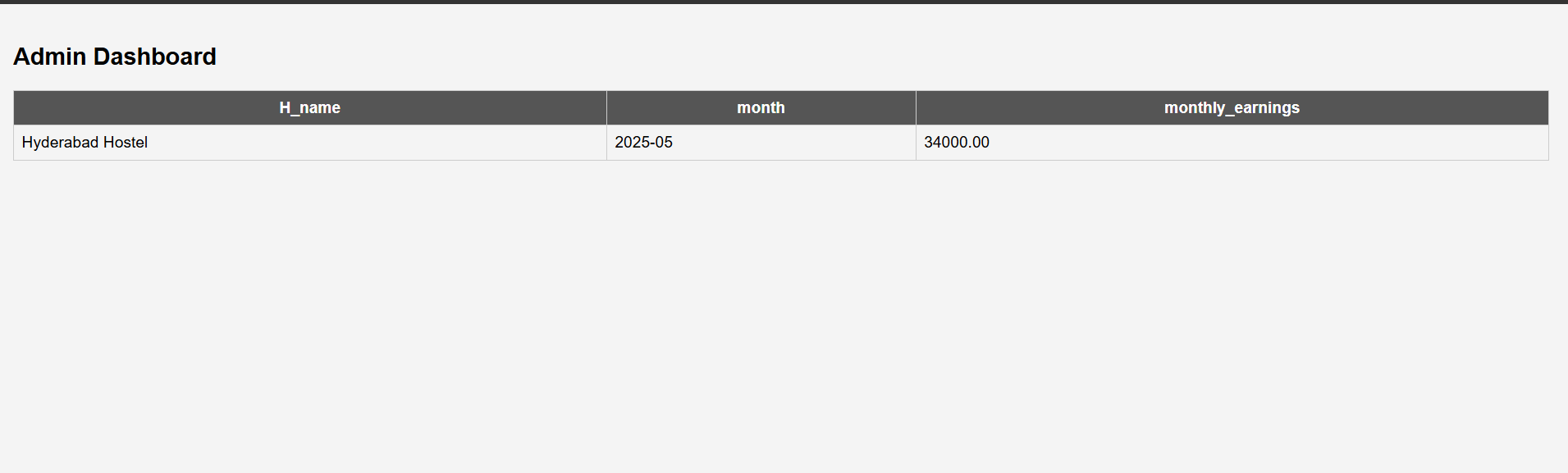
**Guest History**

****

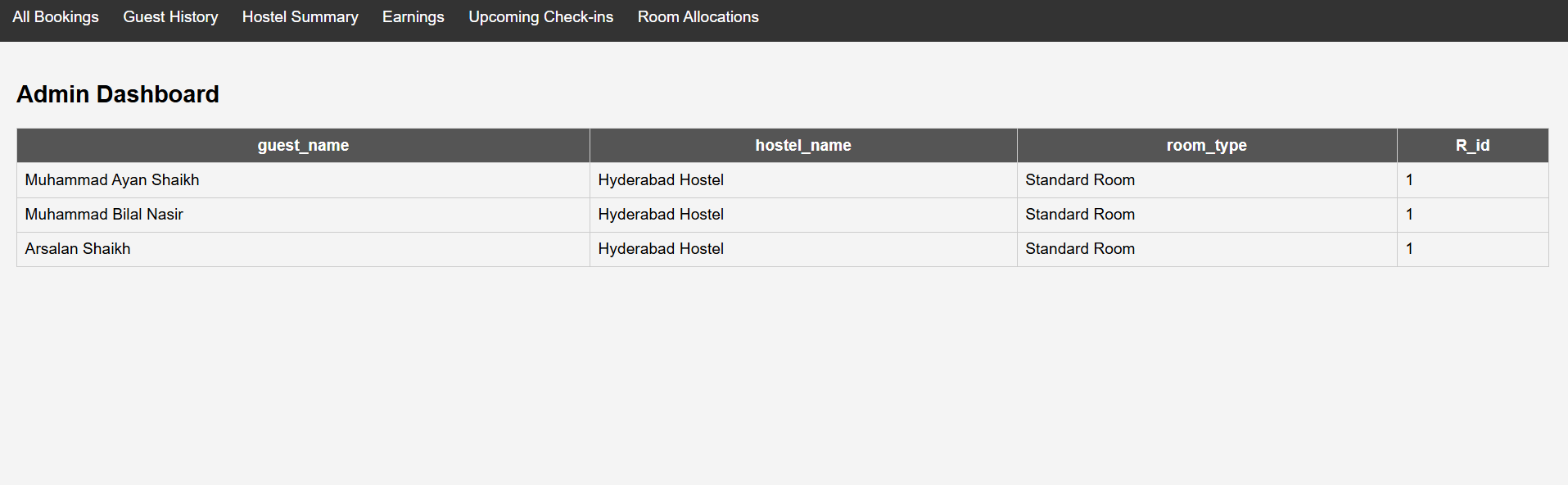
**Hostel Summary**

****

**Earnings**

****

**Room Allocations**

****

# **Detailed Documentations**

## **1. Introduction**

### **Overview of the Project**

### **FlateMate allows people to use the internet to look at, book and control their hostel rooms in several cities. The site brings together frontend UIs and backend PHP by using a MySQL database on a local XAMPP server.**

### **Objectives and Goals**

* Provide an intuitive interface for room bookings.
* Store and manage guest and hostel data efficiently.
* Offer an admin dashboard for real-time management and insights.

### **Significance of the Project**

The significance of the project is that it simplifies hostel management, reduces manual errors, and provides a modern, centralized platform for both users and administrators.

## **2. System Analysis**

### **Requirements Gathering**

System requirements were identified through a detailed examination of typical hostel workflows and user expectations regarding digital booking functionality. (Also We looked At Your Requirements For The Project)

### **Use Case Diagrams**

Use cases include:

* **User Booking Flow**: View hostels → Fill booking form → Submit
* **Admin Flow**: Login → View dashboard → Monitor guests, bookings, and earnings

### **Functional Requirements**

* Booking submission and storage
* Dynamic dashboard views for admin
* Automated room allocation and data updating

### **Non-Functional Requirements**

* Easy usability
* Fast performance
* Secure data handling
* Scalability for more hostels and rooms

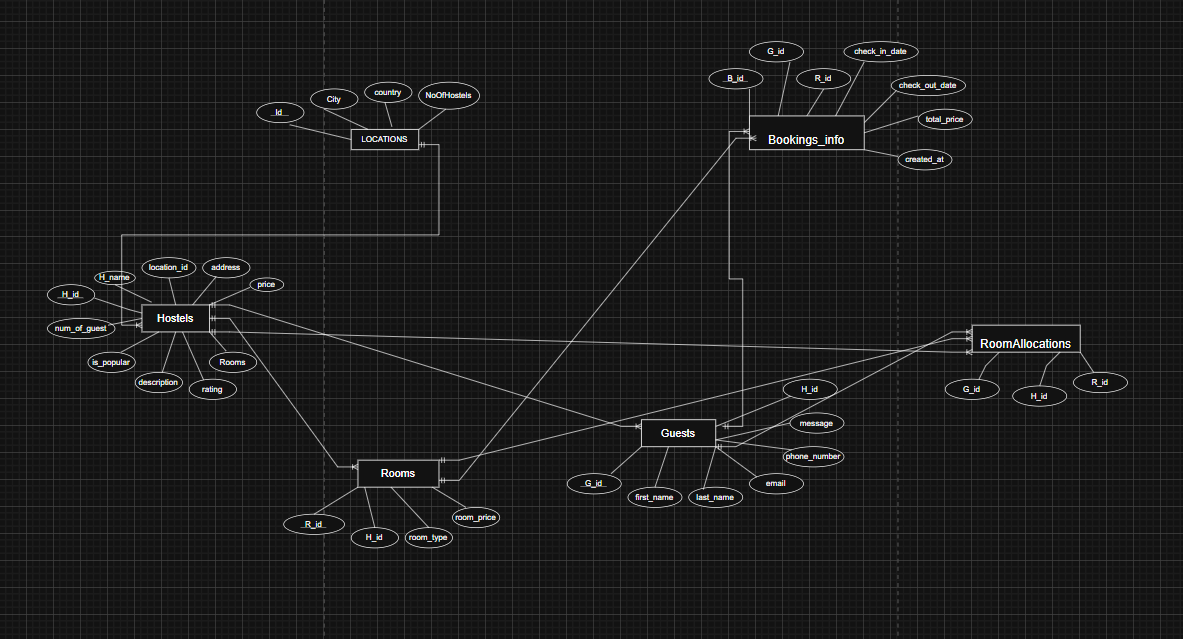
## **3. System Design**

### **Architectural Design**

The system follows a client-server architecture which has:

* **Frontend**: HTML, CSS, JS
* **Backend**: PHP
* **Database**: MySQL
* **Connection**: XAMPP

### **Entity-Relationship Diagram (ERD)**



## **4. Database Design**

### **Schema Design**

### CREATE DATABASE FlateMate\_DB;

### USE FlateMate\_DB;

### CREATE TABLE Locations (

### id INT PRIMARY KEY AUTO\_INCREMENT,

### city VARCHAR(50),

### country VARCHAR(50) NOT NULL,

### NoOfHostels INT NOT NULL

### );

### CREATE TABLE Hostels (

### H\_id INT PRIMARY KEY AUTO\_INCREMENT,

### H\_name VARCHAR(50),

### location\_id INT NOT NULL,

### address VARCHAR(255),

### price DECIMAL(8,2) NOT NULL,

### num\_of\_guests INT,

### description TEXT,

### rating DECIMAL(2,1) CHECK (rating BETWEEN 0 AND 5),

### num\_of\_Rooms INT NOT NULL,

### FOREIGN KEY (location\_id) REFERENCES Locations(id)

### ON DELETE RESTRICT ON UPDATE CASCADE

### );

### CREATE TABLE Rooms (

### R\_id INT PRIMARY KEY AUTO\_INCREMENT,

### H\_id INT NOT NULL,

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

### room\_price DECIMAL(8,2) NOT NULL,

### FOREIGN KEY (H\_id) REFERENCES Hostels(H\_id)

### ON DELETE CASCADE ON UPDATE CASCADE

### );

### CREATE TABLE Guests (

### G\_id INT PRIMARY KEY AUTO\_INCREMENT,

### first\_name VARCHAR(50) NOT NULL,

### last\_name VARCHAR(50),

### email VARCHAR(100) UNIQUE NOT NULL,

### phone\_number VARCHAR(20),

### message TEXT,

### H\_id INT,

### FOREIGN KEY (H\_id) REFERENCES Hostels(H\_id)

### ON DELETE SET NULL ON UPDATE CASCADE

### );

### CREATE TABLE RoomAllocations (

### G\_id INT NOT NULL,

### H\_id INT NOT NULL,

### R\_id INT NOT NULL,

### PRIMARY KEY (G\_id, H\_id, R\_id),

### FOREIGN KEY (G\_id) REFERENCES Guests(G\_id)

### ON DELETE CASCADE ON UPDATE CASCADE,

### FOREIGN KEY (H\_id) REFERENCES Hostels(H\_id)

### ON DELETE CASCADE ON UPDATE CASCADE,

### FOREIGN KEY (R\_id) REFERENCES Rooms(R\_id)

### ON DELETE CASCADE ON UPDATE CASCADE

### );

### CREATE TABLE Bookings\_info (

### B\_id INT PRIMARY KEY AUTO\_INCREMENT,

### G\_id INT NOT NULL,

### R\_id INT NOT NULL,

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

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

### total\_price DECIMAL(10,2) NOT NULL,

### created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

### number\_of\_guests INT NOT NULL DEFAULT 1,

### FOREIGN KEY (G\_id) REFERENCES Guests(G\_id)

### ON DELETE CASCADE ON UPDATE CASCADE,

### FOREIGN KEY (R\_id) REFERENCES Rooms(R\_id)

### ON DELETE CASCADE ON UPDATE CASCADE

### );

### INSERT INTO Locations (city, country, NoOfHostels) VALUES

### ('Hyderabad', 'Pakistan', 1),

### ('Karachi', 'Pakistan', 1),

### ('Islamabad', 'Pakistan', 1),

### ('Lahore', 'Pakistan', 1);

### INSERT INTO Hostels (H\_name, location\_id, address, price, num\_of\_guests, description, rating, num\_of\_Rooms) VALUES

### ('Hyderabad Hostel', 1, 'Hyderabad City Center', 200000.00, 0, 'Affordable stay in Hyderabad', 4.0, 5),

### ('Karachi Hostel', 2, 'Clifton Block 5, Karachi', 200000.00, 0, 'Comfortable and clean rooms in Karachi', 4.3, 6),

### ('Islamabad Hostel', 3, 'F-6 Sector, Islamabad', 200000.00, 0, 'Peaceful environment in Islamabad', 4.5, 4),

### ('Lahore Hostel', 4, 'Gulberg III, Lahore', 200000.00, 0, 'Centrally located in Lahore', 4.2, 7);

### INSERT INTO Rooms (H\_id, room\_type, room\_price) VALUES

### (1, 'Standard Room', 200000.00),

### (2, 'Standard Room', 200000.00),

### (3, 'Standard Room', 200000.00),

### (4, 'Standard Room', 200000.00);

### **Triggers**

USE FlateMate\_DB;

DELIMITER //

CREATE TRIGGER trg\_increment\_guest\_count

AFTER INSERT ON Bookings\_info

FOR EACH ROW

BEGIN

UPDATE Hostels

SET num\_of\_guests = num\_of\_guests + NEW.number\_of\_guests

WHERE H\_id = (

SELECT H\_id FROM Rooms WHERE R\_id = NEW.R\_id LIMIT 1

);

END;

//

DELIMITER ;

DELIMITER //

CREATE TRIGGER trg\_auto\_allocate\_room

AFTER INSERT ON Bookings\_info

FOR EACH ROW

BEGIN

DECLARE hostelId INT;

SELECT H\_id INTO hostelId FROM Rooms WHERE R\_id = NEW.R\_id;

INSERT INTO RoomAllocations (G\_id, H\_id, R\_id)

VALUES (NEW.G\_id, hostelId, NEW.R\_id);

END;

//

DELIMITER ;

### **Views**

USE FlateMate\_DB;

CREATE VIEW view\_all\_bookings AS

SELECT

b.B\_id,

CONCAT(g.first\_name, ' ', g.last\_name) AS guest\_name,

g.email,

h.H\_name AS hostel\_name,

r.room\_type,

b.check\_in\_date,

b.check\_out\_date,

b.number\_of\_guests,

b.total\_price,

b.created\_at

FROM Bookings\_info b

JOIN Guests g ON b.G\_id = g.G\_id

JOIN Rooms r ON b.R\_id = r.R\_id

JOIN Hostels h ON r.H\_id = h.H\_id;

CREATE VIEW view\_guest\_history AS

SELECT

g.G\_id,

CONCAT(g.first\_name, ' ', g.last\_name) AS guest\_name,

g.email,

h.H\_name AS hostel\_name,

r.room\_type,

b.check\_in\_date,

b.check\_out\_date,

b.total\_price

FROM Guests g

JOIN Bookings\_info b ON g.G\_id = b.G\_id

JOIN Rooms r ON b.R\_id = r.R\_id

JOIN Hostels h ON r.H\_id = h.H\_id;

CREATE VIEW view\_hostel\_summary AS

SELECT

h.H\_id,

h.H\_name,

l.city,

l.country,

h.address,

h.num\_of\_Rooms,

h.num\_of\_guests,

h.rating,

COUNT(b.B\_id) AS total\_bookings,

SUM(b.total\_price) AS total\_earnings

FROM Hostels h

JOIN Locations l ON h.location\_id = l.id

LEFT JOIN Rooms r ON h.H\_id = r.H\_id

LEFT JOIN Bookings\_info b ON r.R\_id = b.R\_id

GROUP BY h.H\_id;

CREATE VIEW view\_location\_stats AS

SELECT

l.city,

l.country,

l.NoOfHostels,

COUNT(DISTINCT h.H\_id) AS actual\_hostels,

SUM(h.num\_of\_guests) AS total\_guests

FROM Locations l

LEFT JOIN Hostels h ON l.id = h.location\_id

GROUP BY l.id;

CREATE VIEW view\_upcoming\_checkins AS

SELECT

b.B\_id,

CONCAT(g.first\_name, ' ', g.last\_name) AS guest\_name,

h.H\_name AS hostel\_name,

b.check\_in\_date,

b.check\_out\_date

FROM Bookings\_info b

JOIN Guests g ON b.G\_id = g.G\_id

JOIN Rooms r ON b.R\_id = r.R\_id

JOIN Hostels h ON r.H\_id = h.H\_id

WHERE b.check\_in\_date >= CURDATE()

ORDER BY b.check\_in\_date ASC;

CREATE VIEW view\_earnings\_per\_hostel AS

SELECT

h.H\_name,

DATE\_FORMAT(b.created\_at, '%Y-%m') AS month,

SUM(b.total\_price) AS monthly\_earnings

FROM Bookings\_info b

JOIN Rooms r ON b.R\_id = r.R\_id

JOIN Hostels h ON r.H\_id = h.H\_id

GROUP BY h.H\_id, month

ORDER BY month DESC;

CREATE VIEW view\_room\_allocation\_summary AS

SELECT

CONCAT(g.first\_name, ' ', g.last\_name) AS guest\_name,

h.H\_name AS hostel\_name,

r.room\_type,

ra.R\_id

FROM RoomAllocations ra

JOIN Guests g ON ra.G\_id = g.G\_id

JOIN Hostels h ON ra.H\_id = h.H\_id

JOIN Rooms r ON ra.R\_id = r.R\_id;

### **Table Definitions and Relationships**

* Primary keys for all major entities
* Foreign key constraints ensure referential integrity
* Triggers used for automation

### **Normalization Process**

Tables are normalized to 3NF having:

* No redundant data
* Each table has a single subject
* Foreign keys used for relationships

### **SQL Queries for Data Manipulation**

Insert Value Queries Were Provided Above In **4. Database Design** Section Also Triggers And Views Were given There.

## **5. Implementation**

### **Code Structure**

* **index.html** is used for landing page with booking form.
* **book.php** is used for processing booking and insert it into database.
* **dashboard.php** is used for loading dynamic views via PHP
* **db\_con.php** is used for database connection file.

### **Modules and Functions**

* The Booking handler will capture and store data.
* The Admin view will dynamically load data from SQL views.

### **Integration of Components**

* Frontend form → PHP handler → MySQL insert.
* Admin Dashboard → PHP routing → View queries.

## **6. User Manual**

### **Step-by-Step Instructions**

### 1. You Should Have A Running XAMPP Server. Run Apache And MySQL On Your XAMPP Control Panel

### 2. Copy Folders "FlateMate" And "FlateMate\_Admin" In C:\xampp\htdocs

### 3. Now Open MySQL Files And Run All Three SQL Files In Correct Order To Create And Initialize Database

### 4. Now Open "Localhost/Flatemate/" on your Web Browser To View Initial Front End. (This Will Be user View)

### 5. Book Any Of The Hostel From there By Filling Form.

### 6. Now Go To "Localhost/Flatemate\_admin/Dashboard.php" To View Admin View Of The Bookings

### **Screenshots**

Already provided Above

### **Common Troubleshooting Tips**

* Ensure XAMPP Apache & MySQL are running and working.
* Places project folder inside C:\xampp\htdocs.
* Checks db\_con.php for correct DB credentials.

## **7. Conclusion and Future Work**

### **Summary**

### This system creates a smooth user interface and operational system for both users and hotel staff by using real-time administration tools.

### **Limitations**

Its limitations are that:

* There will be no login system for users or admins.
* It lacks payment integration.

### **Potential Enhancements**

The potential enhancements that we can make to greatly enhance it are:

* Add user authentication.
* Integrate SMS/email confirmation.
* Add Online Payment Option