Project Report: Hotel Management System (HMS)

1. Cover Page

Project Title	**Hotel Management System (HMS)**
Student Name	Abhijay Joshi
Registration No.	25BCE11029
Course	CSE 1021
Submission Date	25 November 2025

2. Introduction

This project, the Hotel Management System (HMS), is a console-based application developed in Python that interacts with a MySQL database. The primary goal of the system is to digitalize and automate the core transactional processes of a small-to-medium-sized hotel, specifically managing rooms, customer registration, and booking reservations.

By leveraging the mysql.connector library, the application establishes a robust connection between the Python application logic and the persistent data storage layer, ensuring data integrity and real-time operational updates.

3. Problem Statement

In many small-scale hospitality environments, the management of rooms, customer records, and reservations is often manual, utilizing ledgers or spreadsheets. This manual process is highly susceptible to human errors, such as double-booking rooms, inaccurate pricing, and lost customer information.

The problem is the need for a centralized, reliable, and automated system to manage these core functions efficiently, minimizing errors and providing an immediate, clear overview of room availability and current bookings.

4. Functional Requirements

The system provides the following five core functions, directly mapped to the application's

menu options:

1. **Room Registration:** Allows an administrator to add new rooms to the system by providing a unique room number, room type (e.g., single, double, suite), and daily price.
2. **Customer Registration:** Allows the registration of new customers by capturing their name, phone number, and email address.
3. **Room Booking:** Facilitates the reservation of an available room for a registered customer by recording the Room ID, Customer ID, and specific Check-in and Check-out dates. Upon booking, the room status is updated to 'Booked'.
4. **View Available Rooms:** Displays a list of all rooms currently marked as 'Available', allowing staff to quickly identify vacant inventory.
5. **View Bookings:** Retrieves and displays a detailed list of all current reservations, showing the booking ID, room number, customer name, check-in, and check-out dates, utilizing a relational join query.

5. Non-functional Requirements

Requirement Category	Description
Reliability	The application must maintain a persistent connection to the MySQL database to ensure that all data is committed and consistent.
Performance	Basic operations (CRUD) must execute within seconds, typical for a localized, low-traffic CLI application.
Security	Basic security is implemented by utilizing a specific database user with credentials (root/abc123) to limit direct user access to the raw database files. (Note: Security needs significant future enhancement).
Usability	The system features a simple, text-based command-line interface (CLI) that is easy to navigate for trained hotel staff.

6. System Architecture

The project follows a basic **Two-Tier Architecture**:

1. **Presentation/Application Tier (Client):** Implemented using Python. This tier handles the user interface (the main menu and input prompts) and the application logic (the

functions like add_room, book_room).
2. **Data Tier (Server):** Implemented using the MySQL Relational Database Management System (RDBMS). This tier stores all persistent data related to rooms, customers, and bookings.

The communication between the two tiers is managed by the Python mysql.connector library, which sends SQL queries from the application to the database and retrieves the results.

7. Design Diagrams

7.1 Entity-Relationship (ER) Diagram

The system relies on three core entities and their relationships.

- **Rooms:** Stores room details.
- **Customers:** Stores guest details.
- **Bookings:** Stores the many-to-many relationship between rooms and customers (a customer can have multiple bookings, and a room can be part of many sequential bookings).

7.2 Use Case Diagram

The primary user of the system is the **Hotel Staff/Admin**.

7.3 Sequence Diagram (Book Room Process)

This diagram illustrates the sequence of operations for the crucial book_room function.

8. Design Decisions & Rationale

Decision	Rationale
Python Language	Chosen for its simplicity, readability, and powerful standard libraries, enabling rapid development of the application logic.
MySQL Database	Selected as the RDBMS for its robustness, scalability, and transactional integrity, making it suitable for managing relational data like bookings and inventory.
mysql.connector	The official Oracle connector was used to ensure a reliable and supported connection method between Python and MySQL.

CLI Interface	A simple Command-Line Interface (CLI) was chosen to focus the project scope primarily on the backend logic and database interaction rather than complex front-end development.

9. Implementation Details

The implementation is structured around modular functions, each corresponding to a specific SQL operation.

Database Connection:
The application initializes the connection globally. A key design point is the use of parameterized queries (e.g., sql, (number, rtype, price)) to prevent SQL Injection vulnerabilities.
add_room and add_customer:
These functions use simple INSERT statements followed by db.commit() to persist the new record to the respective tables.
book_room (Transaction Logic):
This function demonstrates a simple transaction:

1. An UPDATE query changes the status of the specified room to 'Booked'.
2. An INSERT query creates the new record in the bookings table.
3. A single db.commit() ensures both operations succeed or fail together, maintaining data consistency.

view_bookings (Relational Query):
This function uses a multi-table JOIN query to retrieve meaningful data (Room Number and Customer Name) from the rooms, customers, and bookings tables, showcasing the relational capabilities of the system.

10. Screenshots / Results

The application provides a clean, menu-driven interface.

```
---- HOTEL MANAGEMENT SYSTEM ----
1. Add Room
2. Add Customer
3. Book Room
4. View Available Rooms
5. View Bookings
6. Exit
Enter choice: 1
Room Number: 2
Room Type: Normal
Price: 2000
Room added successfully!


---- HOTEL MANAGEMENT SYSTEM ----
1. Add Room
2. Add Customer
3. Book Room
4. View Available Rooms
5. View Bookings
6. Exit
Enter choice: 2
Customer Name: Akshit
Phone: 897656789
Email: akshit@vit.com
Customer registered successfully!


---- HOTEL MANAGEMENT SYSTEM ----
1. Add Room
2. Add Customer
3. Book Room
4. View Available Rooms
5. View Bookings
6. Exit
Enter choice: 3
Room ID: 2
Customer ID: 010
Check-In Date (YYYY-MM-DD): 2025-11-24
Check-Out Date (YYYY-MM-DD): 2025-11-29
Room booked successfully!
```

```
---- HOTEL MANAGEMENT SYSTEM ----
1. Add Room
2. Add Customer
3. Book Room
4. View Available Rooms
5. View Bookings
6. Exit
Enter choice: 4

Available Rooms:
(3, '2', 'Normal', Decimal('2000.00'), 'Available')


---- HOTEL MANAGEMENT SYSTEM ----
1. Add Room
2. Add Customer
3. Book Room
4. View Available Rooms
5. View Bookings
6. Exit
Enter choice: 5

Bookings:
```

11. Testing Approach

The system was tested using a functional black-box approach:

1. **Unit Testing (CRUD):** Each function (add_room, add_customer, book_room, etc.) was tested individually to ensure correct SQL execution and data commitment.
2. **Integration Testing:** The book_room function was tested to verify that a single execution correctly triggered both the UPDATE (room status change) and the INSERT (new booking record) operations.
3. **Relational Testing:** The view_bookings function was tested to ensure the JOIN operation correctly linked the Room Number and Customer Name from their respective tables.