CHAPTER 1

INTRODUCTION

The **Hotel Booking Management System** is a web-based platform designed to revolutionize the hotel reservation process by providing a seamless and efficient experience for both guests and administrators. The system enables users to search for available rooms, select check-in and check-out dates, and complete bookings effortlessly. By replacing traditional paper-based and manual reservation systems, it offers a more reliable and convenient digital solution. With real-time room availability updates, automated confirmations, and secure payment options, the system eliminates common booking errors and enhances the overall customer experience.

Designed with user convenience in mind, the system provides an intuitive interface that allows guests to filter hotel rooms based on factors like price, room type, and amenities. Users can register an account, manage their bookings, and receive instant notifications regarding their reservations. Additionally, the system includes a cancellation feature, allowing users to modify or cancel bookings within specified conditions. By integrating a structured database and modern security protocols, it ensures data accuracy and protects user information from unauthorized access.

The **Hotel Booking Management System** also benefits hotel administrators by offering tools to manage room availability, pricing, and customer transactions efficiently. The Admin Module enables hotel staff to monitor reservations, update room details, and oversee guest feedback. By automating these operations, hotels can reduce manual workloads, minimize double bookings, and optimize revenue management. Furthermore, built-in analytics and reporting tools provide valuable insights into occupancy trends, helping hotels make data-driven business decisions.

Built using **Java, Hibernate, and MySQL**, the system ensures efficient handling of reservations while maintaining data integrity. Hibernate ORM simplifies database interactions, while JDBC facilitates smooth connectivity with MySQL, ensuring high-speed transaction processing. The use of relational databases ensures that room availability is updated in real-time, preventing overbookings and enhancing customer satisfaction. Additionally, strong authentication mechanisms and encryption techniques protect sensitive customer data, making the system secure and reliable for both users and hotel operators.

CHAPTER 2

OBJECTIVES

[1] **Simplify Online Hotel Booking** – The primary objective of this system is to provide an easy-to-use platform for guests to book hotel rooms online. It allows users to search for available rooms, choose check-in and check-out dates, select room types, and make payments efficiently, reducing the dependency on manual booking processes.

[2] **Automate Room and Reservation Management** – The system automates the management of hotel rooms, reservations, and availability. By integrating an organized database, it prevents double bookings, ensures real-time room availability updates, and provides instant booking confirmations to guests.

[3] **Enhance User and Admin Experience** – The system is designed to offer a smooth and intuitive interface for both guests and hotel administrators. Users can easily view their booking history, cancel reservations, and provide feedback, while admins can manage room details, monitor reservations, and oversee guest interactions efficiently.

[4] **Ensure Secure Transactions and Data Management** – With the integration of authentication mechanisms and database security measures, the system ensures that user data, payment details, and booking records remain protected. Role-based access control allows only authorized hotel staff to manage critical system functions securely.

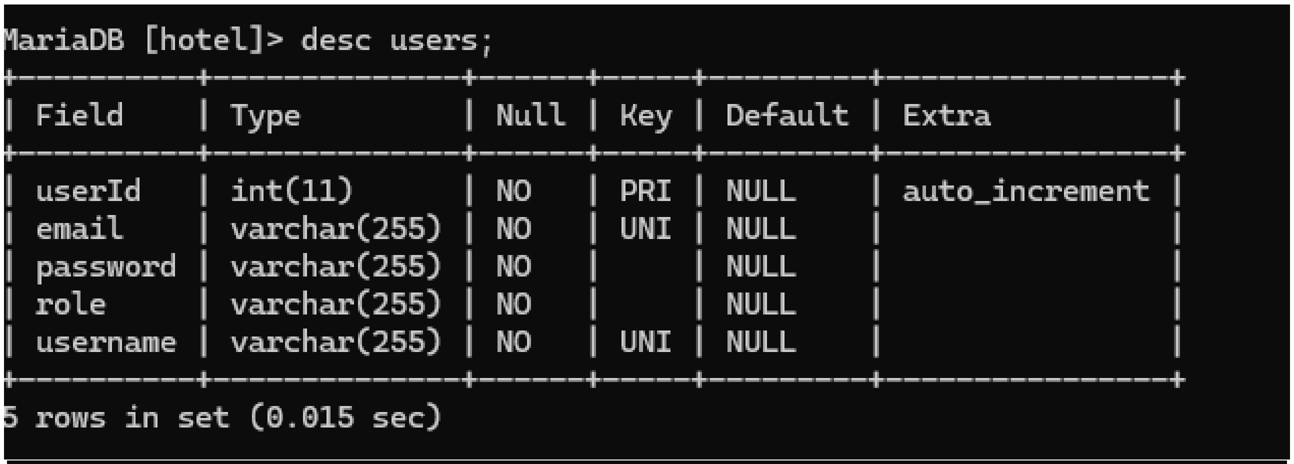
[5] **Improve Efficiency and Reduce** Errors – By digitizing the hotel booking process, the system minimizes human errors, eliminates paperwork, and speeds up the reservation process. This leads to better efficiency in hotel operations and enhances customer satisfaction by providing a seamless booking experience.

[6] **Facilitate Feedback and Service Improvement** – The system includes a feedback module where guests can rate their stay experience. This helps hotel management analyze user feedback and make necessary improvements to enhance service quality and guest satisfaction.

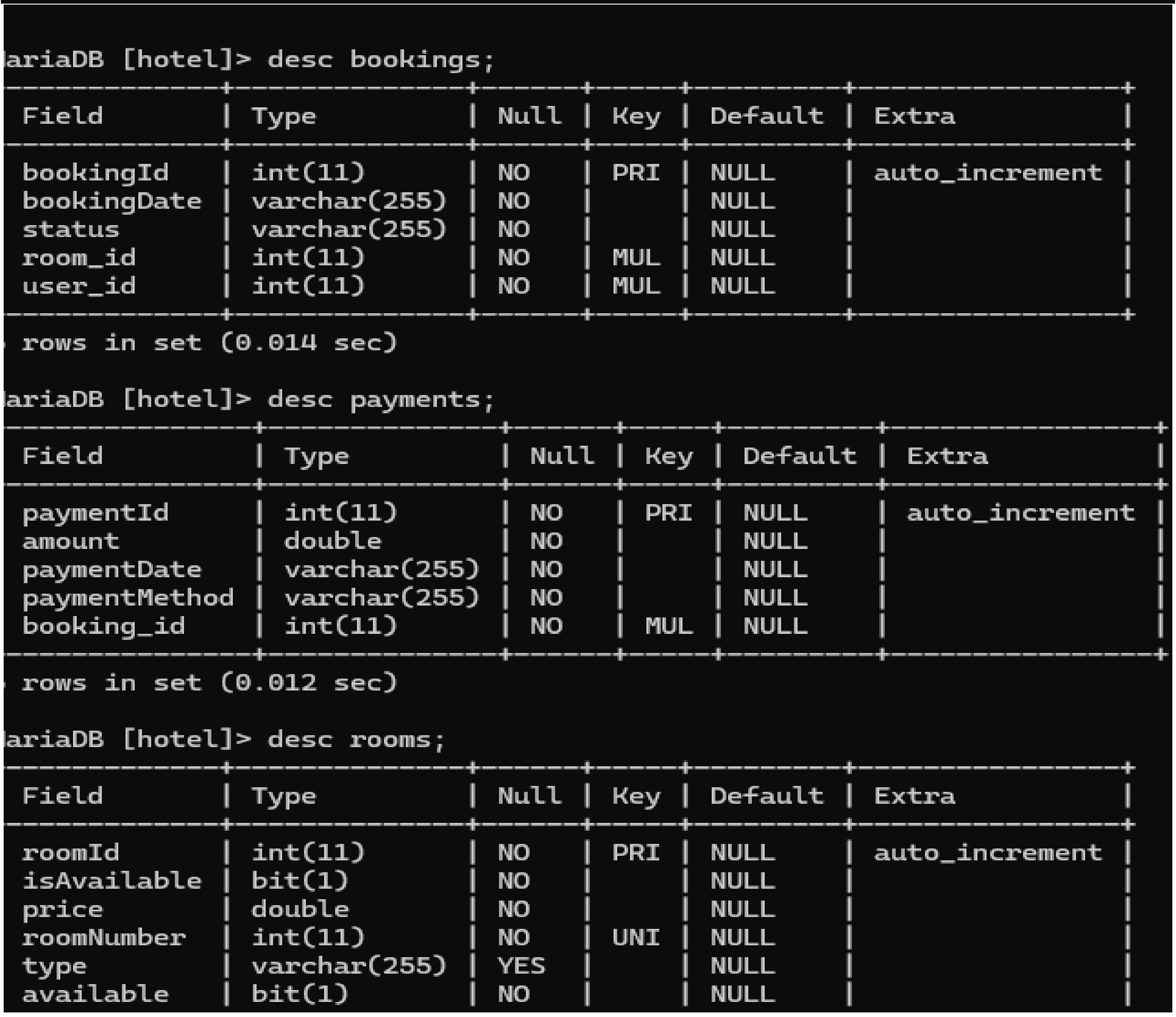
[7] **Support Scalability and Future Enhancements** – The system is designed to be scalable, allowing for the addition of new features like dynamic pricing, loyalty programs, and integration with third-party travel agencies. It ensures long-term usability and adaptability to future needs.

CHAPTER 3

DATABASE DESIGN



```
MariaDB [hotel]> desc users;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| userId   | int(11)      | NO   | PRI | NULL    | auto_increment |
| email    | varchar(255) | NO   | UNI | NULL    |                |
| password | varchar(255) | NO   |     | NULL    |                |
| role     | varchar(255) | NO   |     | NULL    |                |
| username | varchar(255) | NO   | UNI | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.015 sec)
```

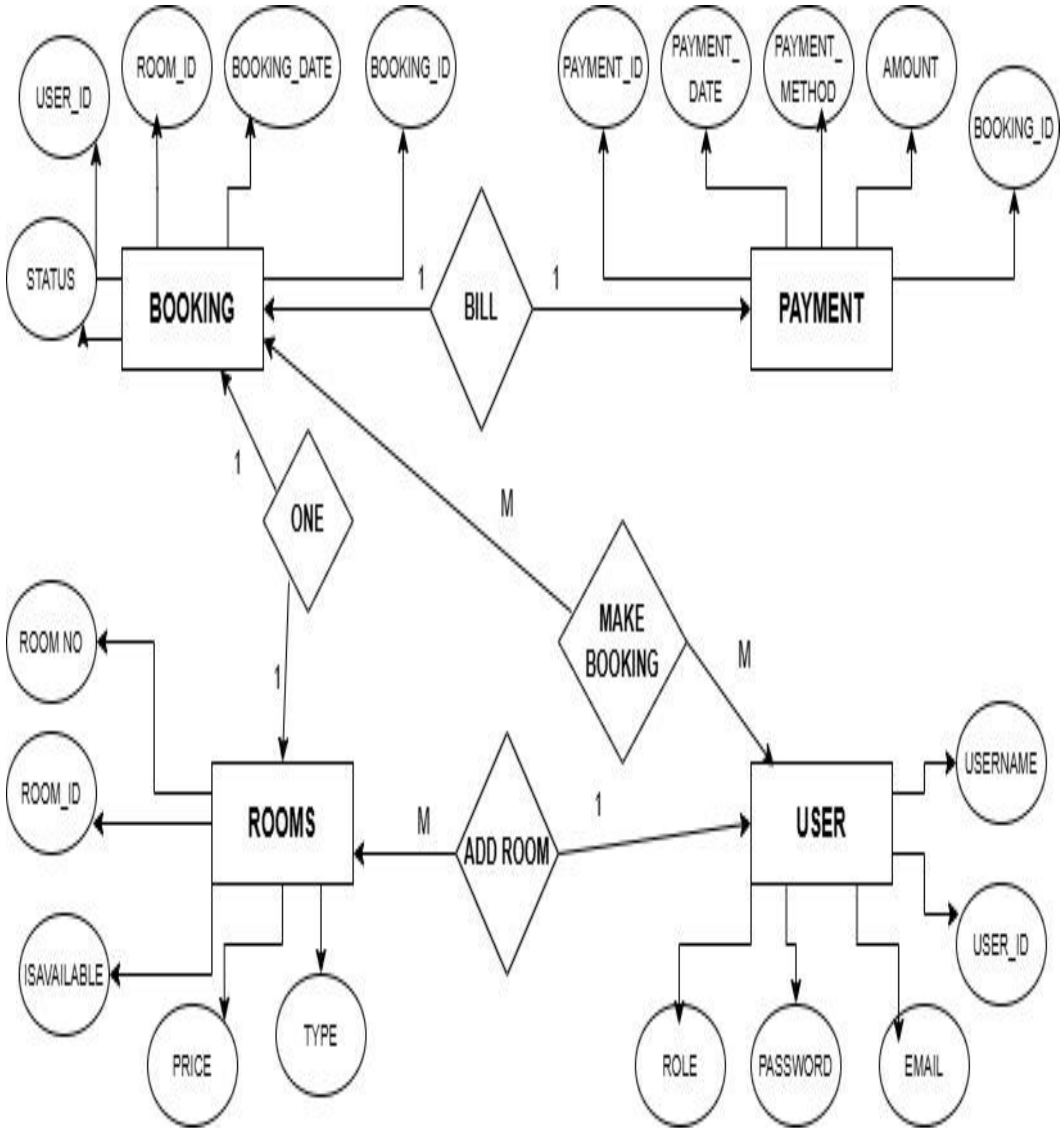


```
ariaDB [hotel]> desc bookings;
-------------+--------------+------+-----+---------+----------------+
 Field       | Type         | Null | Key | Default | Extra          |
-------------+--------------+------+-----+---------+----------------+
 bookingId   | int(11)      | NO   | PRI | NULL    | auto_increment |
 bookingDate | varchar(255) | NO   |     | NULL    |                |
 status      | varchar(255) | NO   |     | NULL    |                |
 room_id     | int(11)      | NO   | MUL | NULL    |                |
 user_id     | int(11)      | NO   | MUL | NULL    |                |
-------------+--------------+------+-----+---------+----------------+
 rows in set (0.014 sec)

ariaDB [hotel]> desc payments;
---------------+--------------+------+-----+---------+----------------+
 Field         | Type         | Null | Key | Default | Extra          |
---------------+--------------+------+-----+---------+----------------+
 paymentId     | int(11)      | NO   | PRI | NULL    | auto_increment |
 amount        | double       | NO   |     | NULL    |                |
 paymentDate   | varchar(255) | NO   |     | NULL    |                |
 paymentMethod | varchar(255) | NO   |     | NULL    |                |
 booking_id    | int(11)      | NO   | MUL | NULL    |                |
---------------+--------------+------+-----+---------+----------------+
 rows in set (0.012 sec)

ariaDB [hotel]> desc rooms;
-------------+--------------+------+-----+---------+----------------+
 Field       | Type         | Null | Key | Default | Extra          |
-------------+--------------+------+-----+---------+----------------+
 roomId      | int(11)      | NO   | PRI | NULL    | auto_increment |
 isAvailable | bit(1)       | NO   |     | NULL    |                |
 price       | double       | NO   |     | NULL    |                |
 roomNumber  | int(11)      | NO   | UNI | NULL    |                |
 type        | varchar(255) | YES  |     | NULL    |                |
 available   | bit(1)       | NO   |     | NULL    |                |
```

E R DIAGRAM


USER_ID
ROOM_ID
BOOKING_DATE
BOOKING_ID
STATUS
BOOKING
1
BILL
1
PAYMENT_ID
PAYMENT_ DATE
PAYMENT_ METHOD
AMOUNT
BOOKING_ID
PAYMENT
1
ONE
M
MAKE BOOKING
M
1
ROOM NO
ROOM_ID
ROOMS
M
ADD ROOM
1
USER
USERNAME
USER_ID
ISAVAILABLE
PRICE
TYPE
ROLE
PASSWORD
EMAIL

CHAPTER 4

IMPLEMENTATION

ENTITY CLASS

This layer represents database tables using JPA entities, including:

1. **User Entity**
    - **Attributes:** userId, email, password, role, username
    - Represents registered users, differentiating between guests and administrators.
2. **Room Entity**
    - **Attributes:** roomId, roomNumber, price, type, isAvailable
    - Maintains room details, including pricing, status, and type.
3. **Booking Entity**
    - **Attributes:** bookingId, bookingDate, status, user_id, room_id
    - Stores reservation details, linking users to booked rooms.
4. **Payment Entity**
    - **Attributes:** paymentId, amount, paymentMethod, paymentDate, booking_id
    - Records payment transactions linked to booking.

DAO Layer (Data Access Objects)

This layer manages database interactions using DAO classes:

1. **UserDAO** – Handles user-related queries (**register, login, fetch user details**).
2. **RoomDAO** – Manages **room details**, updating availability when bookings are made.

3. **BookingDAO** – Responsible for **inserting, updating, and retrieving reservations**.

4. **PaymentDAO** – Processes **payment transactions and maintains records**.

Service Layer (Business Logic)

This layer implements business logic, ensuring data validation, security, and processing:

1. **UserService**
   - Validates **user credentials** during login.
   - Enforces **authentication and authorization**.
2. **RoomService**
   - Checks **room availability** before confirming a booking.
   - Updates room status in case of cancellations.
3. **BookingService**
   - Manages **reservation workflows**.
   - Ensures no overbooking occurs.
4. **PaymentService**
   - Handles **payment processing**, verifies transactions, and generates receipts.

MAIN CLASS

The Main class contains the main() method that initializes and runs the program, calling DAO methods to perform operations.

How It Works

This class is the starting point of the application.

Calls the UserDAO to save a user into the database.

Calls getAllUsers() to retrieve and print all users.

HibernateUtil Class

The HibernateUtil class is a helper class that creates and manages the Hibernate SessionFactory

How It Works

Loads Hibernate configurations from hibernate.cfg.xml.

Creates a SessionFactory, which is used to create Session instances for database interactions.

Provides a getSessionFactory() method to get a session for executing queries.

Hibernate.cfg.xml File

The hibernate.cfg.xml file is the Hibernate configuration file that defines database connection settings and Hibernate properties. It includes:

1. Database Connection – Specifies the database URL, username, password, and driver class.

2. Hibernate Properties – Defines dialect, SQL logging, and transaction handling.

3. Mapping Configuration – Lists entity classes mapped to database tables.

This file is essential for Hibernate to connect with the database and manage ORM (Object-Relational Mapping) efficiently.

pom.xml file

The pom.xml file contains dependencies for Hibernate, MySQL, and other required libraries.

Adds Hibernate, MySQL driver, and Jakarta Persistence dependencies.

- Handles **payment processing**, verifies transactions, and generates receipts.

CONCLUSION

The **Hotel Management System** streamlines hotel operations by automating **room booking, user management, payment processing, and reservation tracking**, ensuring efficiency, accuracy, and security. Utilizing **Spring Boot, Hibernate, and MySQL**, the system enables seamless database interactions through the **DAO (Data Access Object) Layer**, ensuring real-time updates and data integrity. The structured architecture enhances maintainability, scalability, and reliability by reducing redundancy and improving security with **role-based access control**. From an administrative perspective, it provides **real-time insights into bookings, room availability, and financial transactions**, while for users, it offers a **seamless digital experience**, enabling easy reservations, secure payments, and booking management. By eliminating manual errors and paperwork, the system significantly enhances **operational efficiency and customer satisfaction**, contributing to the **digital transformation of the hospitality industry**. Future enhancements, such as **AI-driven recommendations, dynamic pricing, and multi-property support**, will further improve its capabilities, making it an **innovative and scalable solution for modern hotel management**.