

**INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE**

“Albergo-Hotel Management System”

## PG-DAC Feb 2025

Submitted By:

Group No: 38

**Roll No. Name of Student**

252048 Madhumita Ashish Mahata

252099 Sunita Singh

**Mr. Harshal Waghchaure Mr. Prashant Deshpande**

Project Guide Centre Coordinator

# ABSTRACT

**Albero - Hotel Management System**  is a comprehensive full-stack web application built to streamline hotel operations through a responsive and dynamic Hotel Management System. Developed using Spring Boot for the backend and React.js for the frontend, the application ensures a seamless and efficient user experience. The backend integrates Spring Security and JWT (JSON Web Token) to enforce secure login and role-based access control, managing distinct privileges for Admin, Manager, Receptionist, and Customer roles. The MySQL database efficiently stores and manages hotel data, including customer profiles, room availability, bookings, and billing information.

The system supports real-time room booking, automated service allocation, and dynamic billing generation, reducing manual effort and errors. RESTful APIs facilitate smooth communication between the client and server, adhering to best practices for scalability and maintainability. JWT tokens ensure secure and stateless authentication, reducing the risk of session hijacking. Each user role features a tailored dashboard, enhancing productivity and usability through React-based interactive UIs.

This project showcases the development of a modern, scalable web application with a layered architecture. It demonstrates how to integrate frontend frameworks, backend technologies, security protocols, and relational databases to build a complete enterprise-grade solution that aligns with current industry standards.

# ACKNOWLEDGEMENT

I would like to take this opportunity to formally express my sincere gratitude and deep appreciation to all the individuals and institutions whose support and guidance have been instrumental in the successful completion of this project. Their encouragement and assistance have played a pivotal role throughout this challenging yet enriching journey.

First and foremost, I extend my heartfelt thanks to my esteemed project guide, Mr. Harshal, for his invaluable guidance, insightful advice, and unwavering support at every critical stage of this project. His mentorship was essential in shaping the practical aspects of this work, which are deeply rooted in the concepts he has imparted to us. His constant motivation and timely suggestions ensured that the project remained on the right path.

I am also profoundly grateful to our respected Centre Co-ordinator, Mr. Prashant Deshpandey, for granting us access to the necessary resources and facilities that greatly contributed to the project's execution. My appreciation also extends to all the faculty members for their continuous encouragement and support.

Lastly, I would like to thank my family and friends for their constant encouragement, understanding, and moral support throughout this endeavor. Their presence and belief in me have been a source of strength during every phase of this project.

Madhumita Ashish Mahata (250241220099)

Sunita Singh (250241220194)

IACSD Albergo

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Description** | **Page No.** |
| 1 | Introduction | 1 |
| 2 | SRS | 16 |
| 3 | Diagrams | 23 |
| 3.1 | ER Diagram | 23 |
| 3.2 | Use Case Diagram | 24 |
| 3.3 | Data Flow Diagram | 25 |
| 3.4 | Activity Diagram | 26 |
| 3.5 | Class Diagram | 29 |
| 3.6 | Sequence Diagram | 30 |
| 4 | Database Design | 31 |
| 5 | Snapshots | 34 |
| 6 | Conclusion | 38 |
| 7 | References | 39 |

1. **INTRODUCTION**

**Albero - Hotel Management System**  is a comprehensive full-stack web application built to streamline hotel operations through a responsive and dynamic Hotel Management System. Developed using Spring Boot for the backend and React.js for the frontend, the application ensures a seamless and efficient user experience. The backend integrates Spring Security and JWT (JSON Web Token) to enforce secure login and role-based access control, managing distinct privileges for Admin, Manager, Receptionist, and Customer roles. The MySQL database efficiently stores and manages hotel data, including customer profiles, room availability, bookings, and billing information.

In the **Hotel Management** system, both **User** and **Admin** roles play crucial roles in the management and operation of the platform. Each role comes with specific responsibilities, permissions, and access levels to ensure that the system operates smoothly and securely.

**1. Admin Use Case:**

**Task:** Add a new hotel branch and manage user accounts

**Example:**Upon logging in securely via JWT-based authentication, the Admin gains access to a centralized Admin Dashboard. From here, the Admin can register a new hotel branch, providing details such as branch name, location, number of rooms, available facilities, and service types. Once the branch is added, the Admin proceeds to create user accounts for the staff at this branch, including assigning roles (e.g., Manager, Receptionist) and defining access permissions. This ensures that users have access only to the features relevant to their roles. The Admin can also deactivate users, audit logs of user activities, and manage system-wide settings such as service tax rates or seasonal pricing rules.

**2. Manager Use Case:**

**Task:** Monitor room occupancy and oversee operations

**Example:**  
The Manager logs in using secure credentials and is presented with a Manager Dashboard tailored to operational needs. This dashboard displays real-time data on room occupancy, upcoming bookings, cancellations, and customer check-ins/check-outs. Managers can also generate analytical reports to evaluate revenue performance, booking trends, and service usage. Based on business needs or seasonal demands, they can update room prices, enable discounts or offers, and monitor customer feedback submitted through the system. The Manager also supervises staff performance, ensuring that bookings and billing are handled accurately by the Receptionists.

**3. Receptionist Use Case:**

**Task:** Handle bookings, check-ins, and billing

**Example:**

Receptionists use their credentials to access a simplified Receptionist Dashboard, focusing on daily operations. When a customer arrives without a prior booking, the receptionist checks real-time room availability, assigns a room based on customer preferences, and completes the booking through the system. All customer details and stay duration are captured securely. Upon checkout, the system calculates charges automatically based on the length of stay, room type, and any additional services (e.g., food, laundry). The application then generates an itemized bill, which can be printed or emailed to the customer. Receptionists can also handle booking modifications, cancellations, and special service requests.

**4. Customer Use Case (Web Portal):**

**Task:** Browse, book, and manage room reservations

**Example:**

Customers interact with the system through a responsive web portal developed in React.js. They can register an account or log in to an existing one. Once authenticated, customers can browse available rooms, view amenities, check room images, and see

pricing details. The booking interface allows selection of check-in/check-out dates and room types. Customers complete the booking process by providing personal details and selecting a payment method (if integrated). After booking, they can view their booking history, cancel or modify upcoming reservations, and even submit feedback. This portal ensures transparency, convenience, and a smooth user experience.

**5. Secure Authentication and Role-Based Access Control:**

**Task:** Restrict access to features based on user roles

**Example:**

The application uses Spring Security integrated with JWT for stateless, token-based authentication. Each user is assigned a role upon registration or account creation. These roles determine which parts of the system they can access. For instance, if a Receptionist attempts to access Admin functionalities like user creation or branch management, the system identifies their role and denies access automatically. This is enforced at both the backend level (API authorization) and frontend level (UI rendering logic), ensuring data security and preventing unauthorized operations. This design safeguards sensitive information and maintains system integrity.

### Responsibilities:

* + **User Management:**
* Admin is responsible for creating, updating, and deleting user accounts (Admin, Manager, Receptionist, Customer).
* Assigns appropriate roles and access levels to each user using a secure interface.
* Maintains user profile data and credentials.
* Tracks and audits user activity for security and compliance.
* Ensures data privacy by restricting user access to personal or sensitive information based on roles.
  + **Booking Management:**
* Add, update, or delete product information.
* Receptionist handles the booking process for walk-in and online customers.
* Customer can make, view, or cancel bookings via the web portal.
* Manager monitors room occupancy, booking trends, cancellations, and overbookings.
* The system automatically updates booking status (confirmed, checked-in, checked-out, or cancelled).
* Generates notifications or alerts for upcoming check-ins or overbooked scenarios.

### Room Management:

* Admin adds rooms when creating a new booking.
* Manager updates room pricing, availability, and seasonal rates.
* Receptionist assigns rooms during the booking or check-in process.
* System automatically blocks unavailable rooms (under maintenance or already booked).
* Manages room types (e.g., deluxe, suite), amenities, and room images.

### Payment Management:

* Receptionist initiates the billing process at checkout.
* The system automatically generates itemized invoices based on duration of stay, room rates, and additional services.
* Customer can view and download payment receipts.
* Integration with online payment gateways (optional) for digital transactions.
* Maintains a transaction history and ensures accurate payment tracking

### Security Management:

* Spring Security + JWT is implemented to ensure secure, stateless authentication.
* Only authenticated users can access the system functionalities.
* Prevents unauthorized access by validating roles for each API request.
* Encrypts sensitive data (e.g., passwords, tokens) and secures sessions against hijacking.
* Logs suspicious activities or login attempts for monitoring and audit.

### Access Level:

* Each user role (Admin, Manager, Receptionist, Customer) is assigned specific access privileges:
* Admin: Full access to all modules and settings.
* Manager: Operational control over bookings, rooms, reports; limited to assigned branch.
* Receptionist: Access to booking, billing, and check-in/check-out functionalities.
* Customer: Access to personal profile, booking history, and online reservation features.
* Role-based access ensures data confidentiality, integrity, and least privilege principle

## User Role

In Albergo, users primarily refer to customers who interact with the system via the hotel’s web portal. Their access is restricted to functionalities relevant to booking and personal account management, ensuring a secure and streamlined experience.

### Responsibilities:

### Room Booking:

* Search and filter available rooms based on type, price, and amenities.
* Select check-in/check-out dates and complete bookings in real time.
* Cancel or modify bookings as per hotel policy.

### Payment Management:

* Make payments for room bookings using integrated payment options.
* View and download payment history and invoices.

### Profile Management:

* Update personal details such as name, contact number, and address.
* Change account password and manage login credentials securely.

### Booking History:

* View previous and upcoming bookings.
* Download receipts and stay summaries for record-keeping.

### Support Requests:

* Submit queries or service requests related to bookings, rooms, or billing.
* Communicate with hotel staff or support through the system interface.

### Access Level:

* Customers have limited access based on their role.
* They can interact only with their personal account, room listings, and booking/payment features.
* They do not have access to administrative controls, system settings, or other users' data.

### Role-Based Access Control (RBAC)

Albergo enforces RBAC using Spring Security:

* Admins, Managers, and Receptionists have higher-level access to manage operations.
* Customers can access only the features necessary for booking and profile management.
* Access control is applied both on UI and backend API levels to prevent unauthorized usage.

### Security Considerations

* Authentication: Verifying the identity of users before granting access to the system.
* Authorization: Ensuring that users can only access the resources permitted by their role.
* Session Management: Secure handling of user sessions to prevent unauthorized access.

Albergo’s structured role-based system ensures that customers can enjoy a secure, seamless, and self-service hotel booking experience, while hotel staff focus on operational tasks without cross-role interference. This separation of duties enhances efficiency, security, and user satisfaction.

## Purpose

The primary purpose of the Albergo Hotel Management System is to modernize hotel operations by automating bookings, billing, and user management through a centralized platform. It offers a secure, scalable, and user-friendly interface for Admins, Managers, Receptionists, and Customers. By integrating real-time features and role-based access control, Albergo ensures efficient service delivery, data accuracy, and enhanced customer satisfaction, reducing manual effort and operational complexity across single or multi-branch hotel environments.

## Scope

Albergo is a full-stack hotel management solution that supports booking, billing, room management, and user access control for Admin, Manager, Receptionist, and Customer roles. It enables real-time reservations, secure authentication, and centralized multi-branch operations. Built with React.js, Spring Boot, and MySQL, it ensures scalability, data security, and automation. Albergo streamlines hotel operations, enhances customer experience, and supports role-based dashboards and reporting for informed decision-making and efficient management.

## Objective of Albergo-Hotel Management System

The purpose of Albergo is to modernize hotel operations, reduce manual effort, improve customer satisfaction, and ensure secure and efficient management of hotel resources and services - all from a single digital platform.

1. **Centralized Operations Management:**

To provide an all-in-one system that manages bookings, room availability, billing, users, and services across one or multiple hotel branches.

1. **Real-Time Room Booking and Service Allocation:**

To enable quick and error-free booking (both online and on-site) with real-time updates of room availability and automated service management.

1. **Role-Based Access Control (RBAC):**

To ensure that each user (Admin, Manager, Receptionist, Customer) has restricted and relevant access, enhancing both security and usability.

1. **Secure Authentication and Data Protection:**

To integrate robust security mechanisms using Spring Security and JWT for safe login and data access, protecting sensitive user and hotel data.

1. **Customer Self-Service Portal:**

To allow customers to browse, book, manage, and cancel rooms through a responsive web interface without relying on manual support..

1. **Operational Efficiency & Reporting:**

To assist managers and admins in making data-driven decisions by generating reports on revenue, occupancy, cancellations, and customer feedback.

1. **Scalability and Maintainability:**

To build a system that can easily scale with new branches, features, or user types, following modern full-stack architecture principles.

1. **Cost Optimization**

Automation, resource planning, centralized control, paperless systems, energy savings

1. **Enhanced Customer Experience**

Faster service, self-service portal, personalized features, real-time updates

## Functionalities Provided by Albergo-Hotel Management System

### User Management

* + User Registration and Login
    - Stakeholders (Admin, Manager, Recep tionist, Customer) can register, log in, and manage profiles.
    - Secure authentication and authorization using
  + Role-Based Access Control:
    - Roles with specific permissions: Admin manages all operations, Manager oversees hoteperformance, Receptionist handles bookings, and Customers make reservations.
    - Admins can manage roles and assign them dynamically

Profile Management

* + - Users can update personal details (name, contact, email).
    - Features for password change and reset securely.

### Room and Property Management

* + Room Catalog Management
    - Add, update, or delete rooms with type, price, capacity, status, and amenities.
    - Manage room availability and condition (e.g., under maintenance).
  + Room Status Monitoring
    - Real-time room status: Available, Booked, Checked-In, Checked-Out.
    - Alerts for unavailable or overbooked rooms.
  + Room Adjustment and Maintenance
    - Manual updates to room data (for repairs, cleaning, etc.).
    - Assign rooms dynamically to customers.
  + **Booking and Billing Management**

Booking Operations:

* + - Customers and Receptionists can book rooms with live availability checking.
    - Bookings can be confirmed, modified, or cancelled.
  + Payment Integration:
    - Secure online payments (e.g., Razorpay) using cards, UPI, wallets, etc.
    - Auto-update booking status post-payment.
  + Booking Tracking and History
    - View status of bookings, payment confirmation, and check-in/out status.
    - Historical booking data available for customers and staff.
  + Invoicing and Automated Billing
    - Generate itemized bills at checkout.
    - View/download invoices in user dashboard.

### Customer Management

* + Customer Profiles
    - Maintain records with contact, booking history, preferences, and feedback.
  + Loyalty and Discount Management
    - Apply loyalty points, discount codes, or seasonal offers to returning guests.
  + Customer Support
    - Provide assistance via support forms or reception desk queries.
    - Admins manage tickets/complaints raised by customers.
  + Feedback and Reviews
    - Guests can rate rooms or services post-stay.
    - Admins can monitor and moderate reviews.

### Security and Compliance

* + Data Encryption
    - All sensitive data (payments, user data) is encrypted.
  + Audit Trails
    - Logs maintained for booking changes, user actions, and payment activity.
  + Legal Compliance
    - Adheres to data protection laws (e.g., GDPR equivalent).
    - Role-based access ensures least-privilege principle.

### ****Enhanced Customer Experience****

* + Search and Filtering
  + Customers can filter rooms by price, capacity, location, or amenities.
  + Favorites and Booking Cart
  + Users can save rooms for future booking or add to booking cart.
  + Mobile Responsive Design
  + Fully responsive interface across web, mobile, and tablet for seamless access.

### Integration and Extensibility

* + RESTful API Support
  + APIs available for third-party integrations (e.g., travel sites, analytics tools).
  + Third-Party Integrations
  + Future-ready for integrating payment gateways, CRM, or accounting tools.

# SOFTWARE REQUIREMENT SPECIFICATION

The functional requirements for Albergo-Hotel Management System outline the specific features and capabilities that the system must provide to meet the needs of its users. These requirements are essential for guiding the development process and ensuring that the final product aligns with the objectives of Hotel management. The system aims to streamline operations for user, and ensure efficient administrative control.

## Functional Requirements for Albergo-Hotel Management System

### User Management

* + User Registration:
  + The system shall allow new users to create/register an account by providing personal details, such as name, email, and password.
  + User Authentication:
  + The system shall authenticate users during login using their registered email and password.
  + Role-Based Access Control:
  + The system shall support role-based access, where different users (Admin, customer, Manager, Receptionist) have different permissions.
  + Profile Management:
  + Users shall be able to view and update their profiles, including personal details and passwords.

### Room Management

* + The system shall allow admins/managers to add, update, and delete room details such as type, description, price, amenities, images, and capacity.

### Booking Management

 Room Booking:

* + Customers shall be able to book rooms online by selecting dates, number of guests, and room type.
* Booking Processing:
  + The system shall process bookings, update room availability, and reserve necessary in-room inventory.
* Payment Processing:
  + The system shall integrate with Razorpay to handle secure payments via credit/debit cards, UPI, and wallets.
* Booking Tracking:
  + Customers shall be able to view booking status: Confirmed, Checked-in, Checked-out, or Cancelled.

 Invoicing:

* + The system shall generate downloadable invoices upon booking completion.

### Customer Management

* Customer Profiles:
  + Maintain customer details, booking history, and preferences.
* Booking History:
  + Customers shall be able to view their complete booking history.
* Feedback & Reviews:
  + Allow customers to leave reviews for rooms and services.

### Security

* Data Encryption:
  + Encrypt sensitive information like payment details and customer personal data.
* Authentication & Authorization:
  + Enforce strong authentication and role-based access controls to secure system access.

## Non-Functional Requirements for Albergo-Hotel Management System

### Performance

* Response Time:
  + The system shall respond to user actions (e.g., browsing available rooms, checking availability, making a booking, or processing payments) within 2 seconds under normal operating conditions.
* Scalability
  + The system shall efficiently handle an increasing number of hotels, rooms, customers, and booking transactions without performance degradation. It should support many concurrent users (e.g., during festive seasons, peak tourism periods, or holiday sales).

### Throughput:

* + The system shall process at least 100 simultaneous user actions per second during high-traffic periods such as mass check-ins/check-outs or peak online booking hours.

### Reliability

* Availability:
  + The system shall maintain 99.9% uptime over a 12-month period, ensuring continuous service availability for customers, managers, receptionists, and administrators.
* Fault Tolerance:
  + Albergo shall continue functioning in the event of hardware or software failures, with minimal impact on ongoing operations.
* Error Handling:
  + The system shall handle errors gracefully, providing clear and meaningful messages to guide users in case of failures (e.g., failed bookings, payment errors, or network interruptions).

### Usability

* User Interface:
  + Albergo shall provide an intuitive and responsive interface that is easy to navigate for all user roles, with a minimal learning curve and clear instructions for all core actions.

### Maintainability

* Modularity:
  + The system shall be developed using a modular architecture, allowing easy updates, scalability, and integration of new features (e.g., AI-based room recommendations, advanced analytics) without impacting existing functionality.
* Code Quality:
  + The system’s codebase shall follow industry best practices, ensuring it is clean, well-structured, well-documented, and maintainable for future enhancements.
* Testing:
  + The system shall undergo comprehensive testing, including unit testing, integration testing, and user acceptance testing, to ensure system stability and quality before production deployment.

**Other Requirements:**

Hardware and Network Interfaces:

Back-end Server Configuration:

* + Intel Pentium-IV Processor
  + 8 GB RAM

Front-end Client Configuration:

* + AMD RYZEN 5 Processor
  + 128 MB SDRAM
  + 10 GB Hard Disk Drive
  + 104 Keys Keyboard
  + PS2 Mouse with pad

### Software Interfaces:

Software configuration for back-end Services:

* + Java EE 21
  + Spring Boot, JPA, Razor Pay, Spring Authentication
  + MySQL 8

- STS 4.30.0

### Software configuration for front-end Services:

* + ReactJS 19
  + HTML, CSS, JS
  + TailWindCSS 4
  + VS Code 9

# 3 .DIAGRAMS

## 3.1 Entity Relationship Diagram:

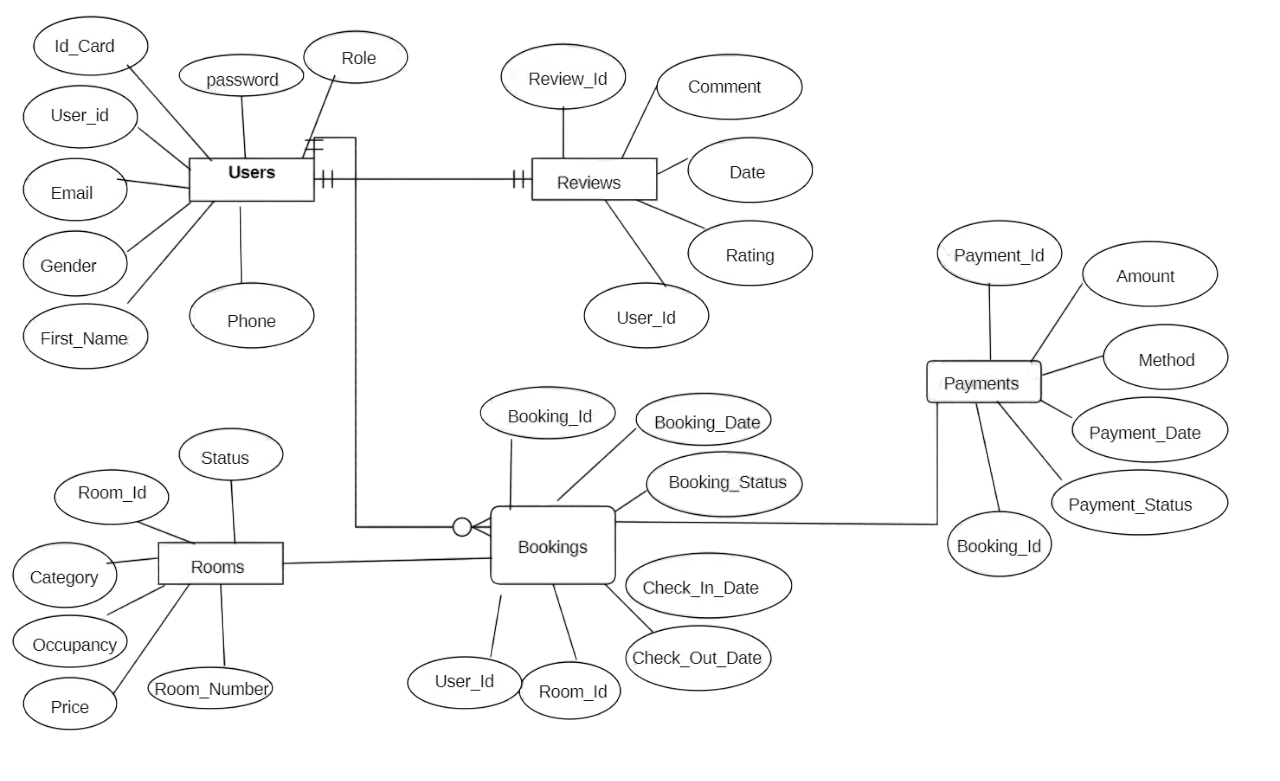


Fig. ER Diagram for Albergo- Hotel Management System

## Use Case Diagram:

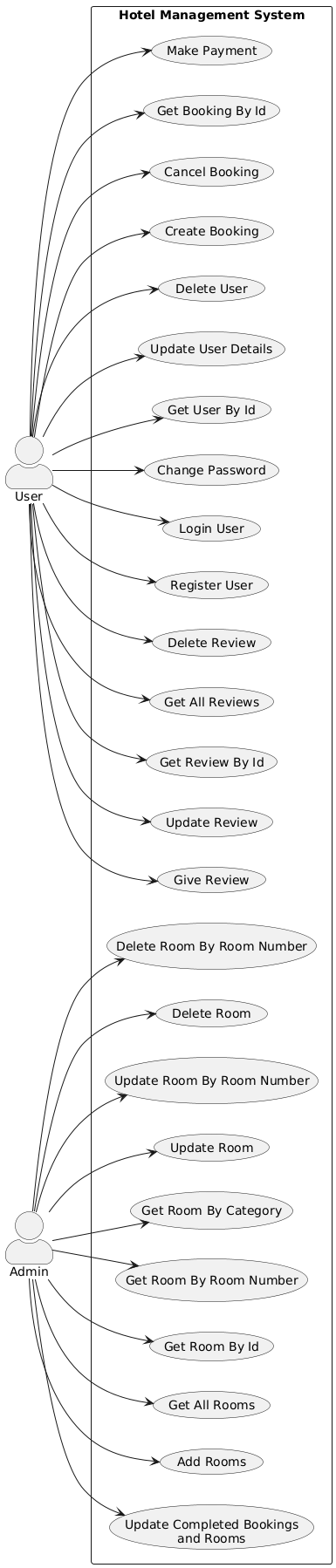
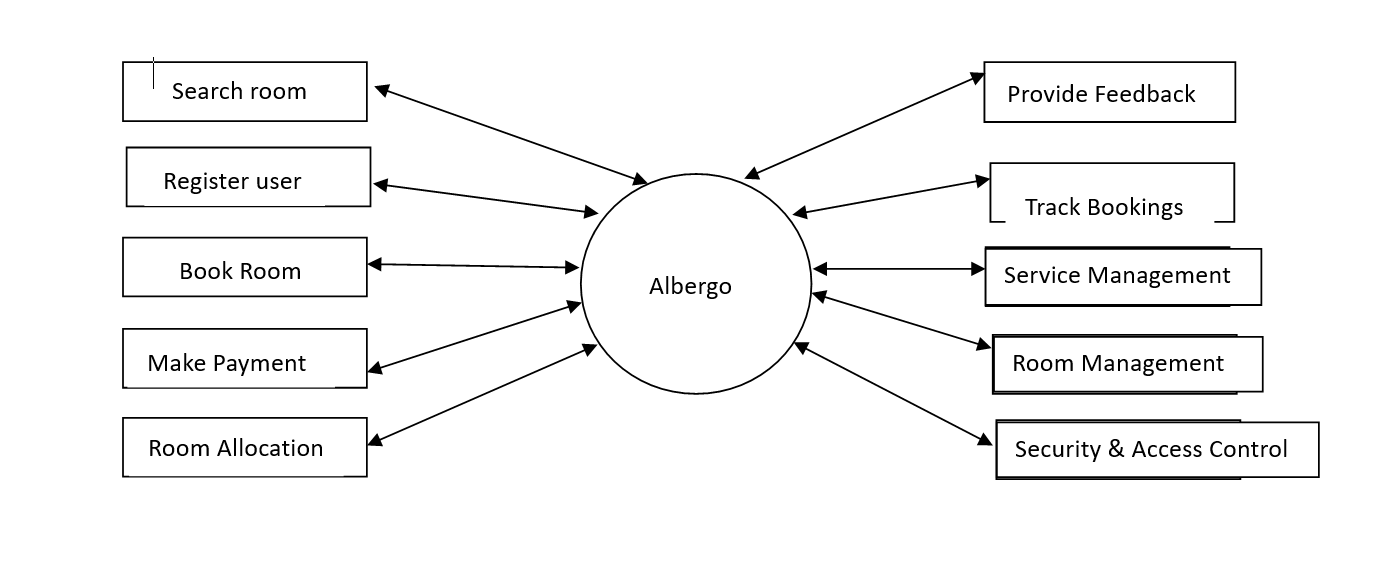


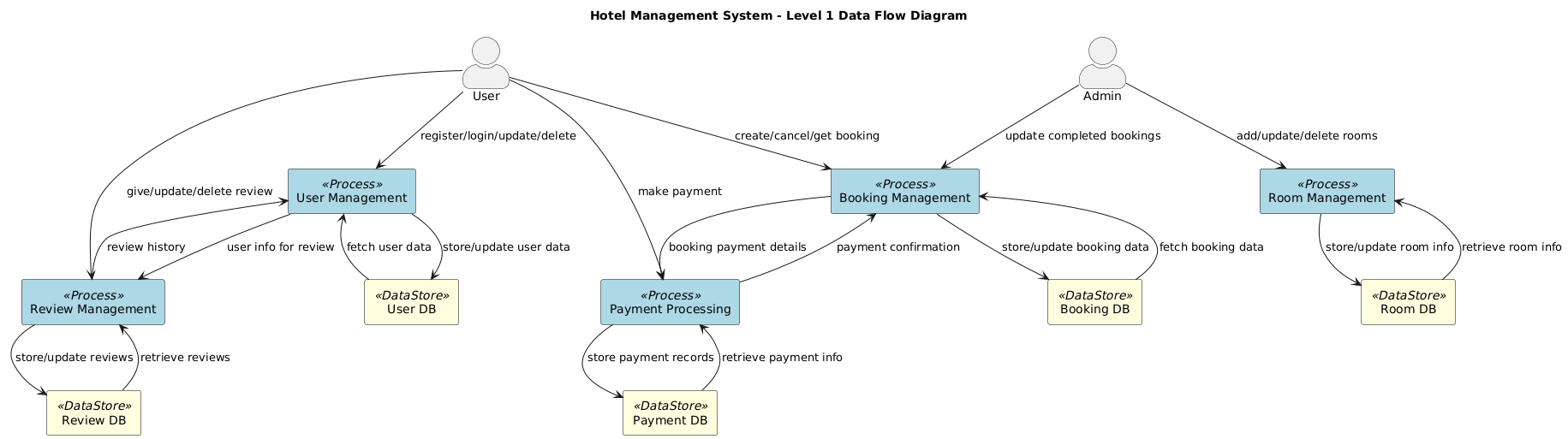
Fig. Use Case Diagram for Albergo-Hotel Management

## Data Flow Diagram:

**DFD Level 0:**

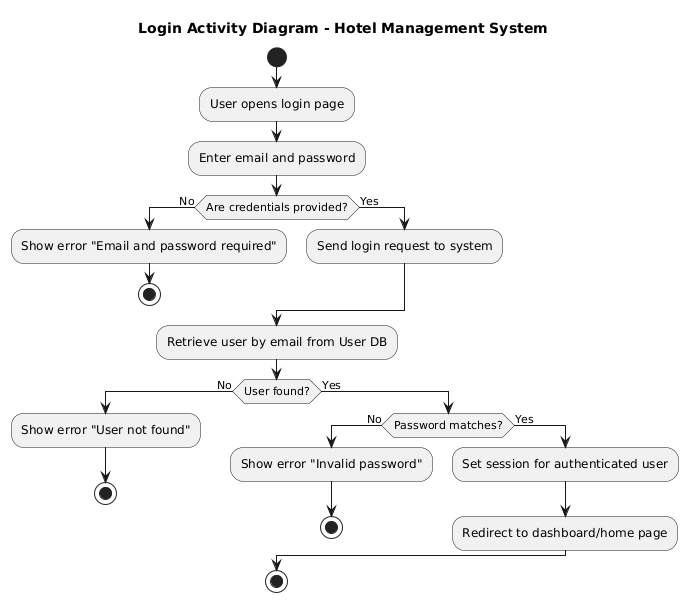
****

**DFD level 1 :**

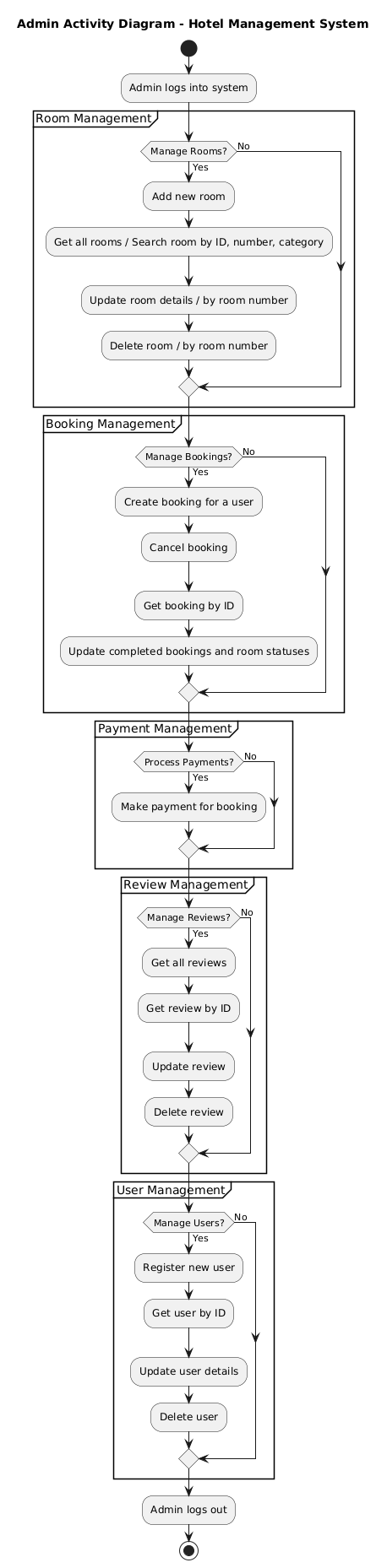


## Activity Diagram :

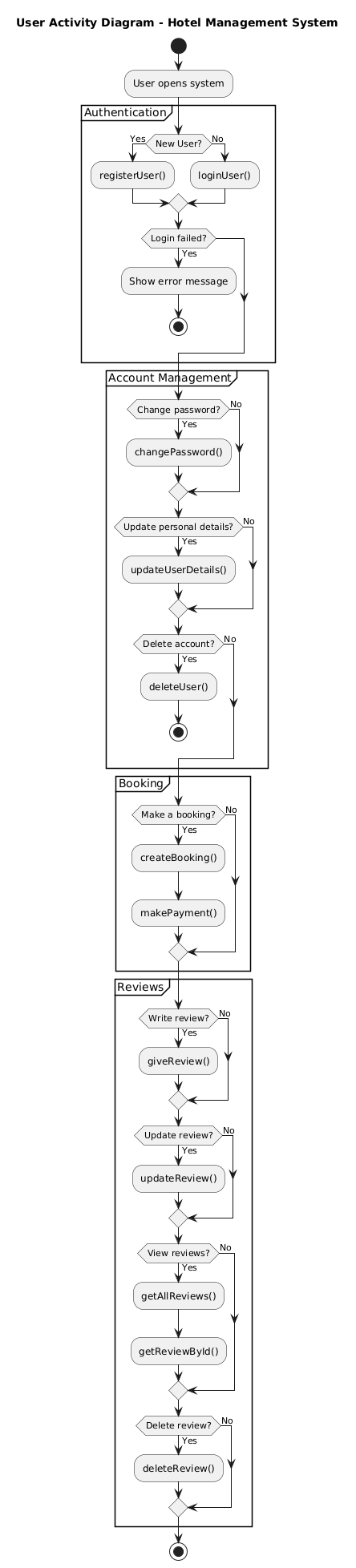
1. Login Activity Diagram



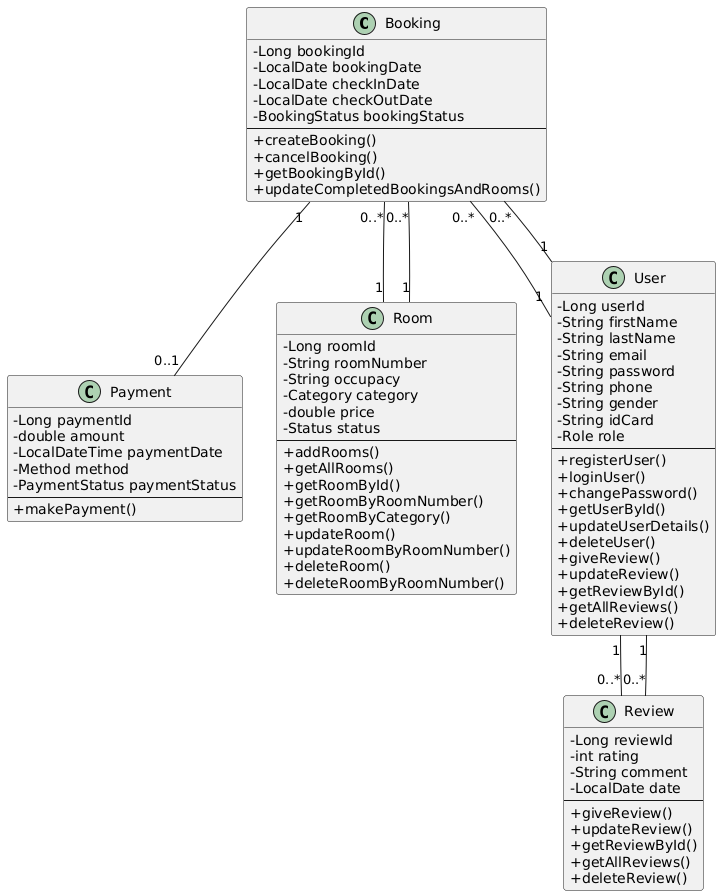
1. Admin Activity Diagram:



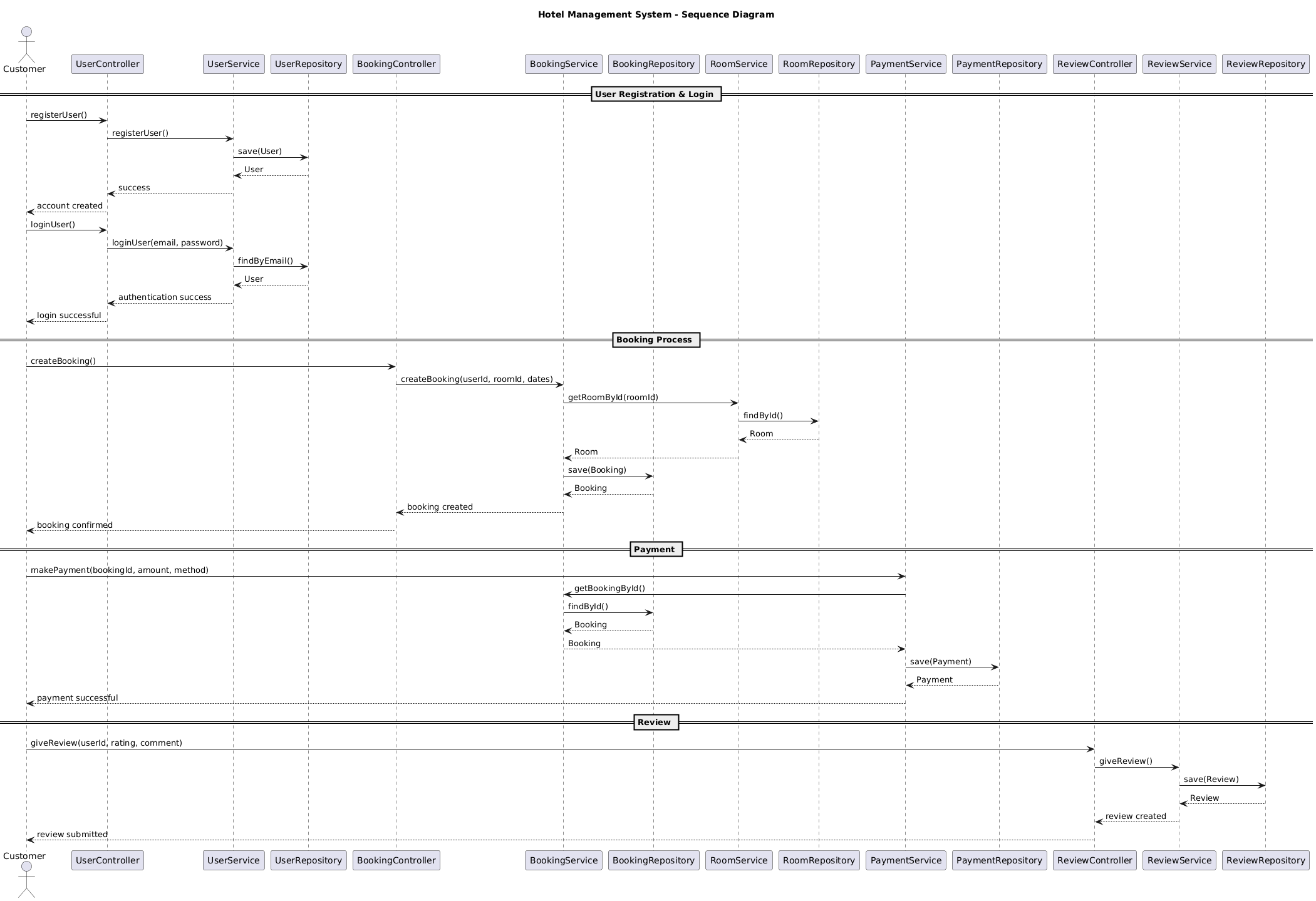
1. User Activity Diagram



* 1. **Class Diagram:**



* 1. **Sequence Diagram**



# DATABASE DESIGN

## Design:

## 

* 1. **Tables:**

The following table structures depict the database design.

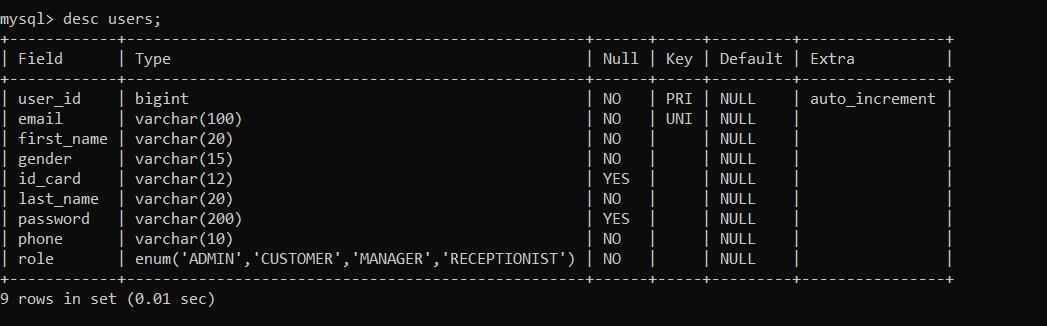


Table 1: users

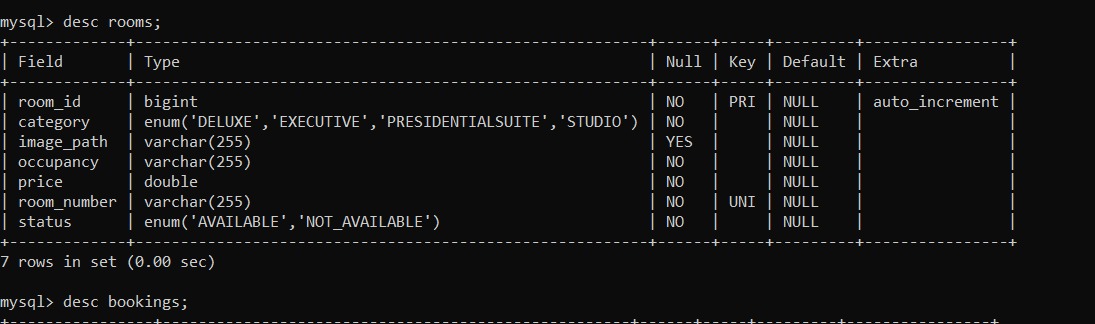


Table 2: rooms

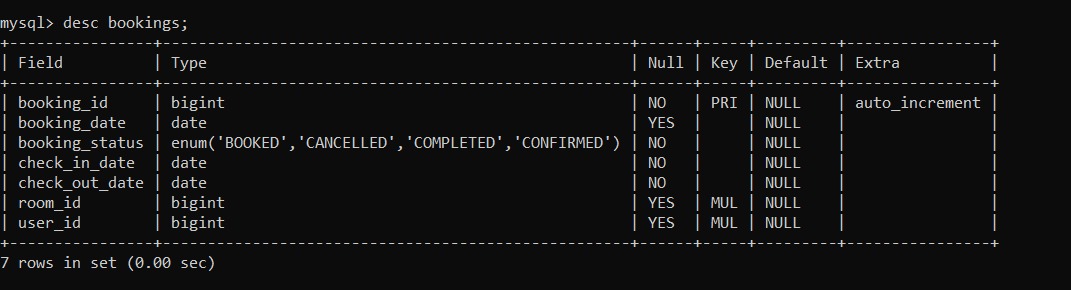


Table 3: bookings

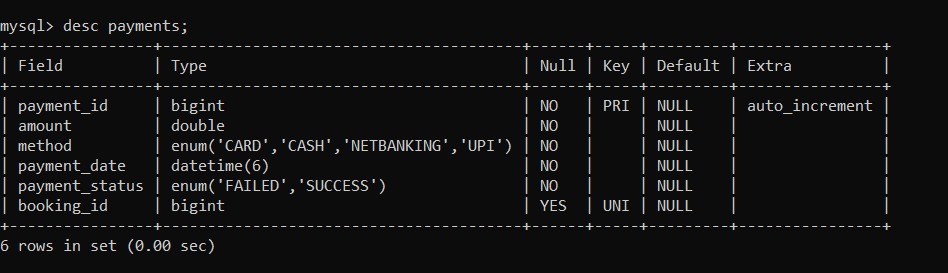


Table 4: payments

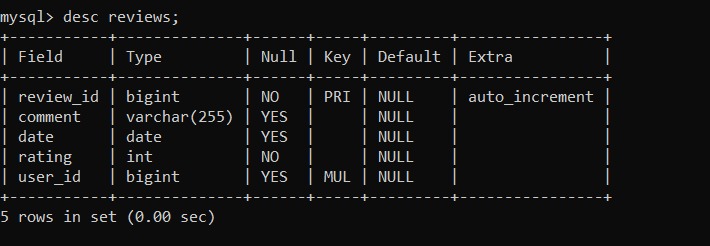


Table 5: reviews

**SNAPSHOTS**

### Home Page:

### 

### Login Page:

### 

### Registration Page:

### 

### Manager Dashboard:

### 

### Admin Dashboard:

### 

### Customer Dashboard:

### 

### Rooms Page:

### 

### Payment Page:

### 

# CONCLUSION

The Albergo – Hotel Management System is designed to streamline hotel operations by integrating reservation management, customer records, room allocation, billing, payment processing, and reporting into a single, unified platform. By leveraging modern technologies such as Spring Boot, Hibernate ORM, and secure payment APIs, Albergo aims to provide a robust, scalable, and user-friendly solution for hotel administrators, staff, and guests.

This system not only improves operational efficiency but also enhances the customer experience through faster service, accurate booking information, and secure transactions. The use of well-established software engineering principles, enterprise patterns, and domain-driven design ensures maintainability, flexibility, and long-term adaptability to changing business needs.

In conclusion, Albergo serves as a comprehensive hotel management solution capable of meeting the demands of small to large-scale hospitality businesses, while remaining extensible for integration with third-party services such as online booking platforms and payment gateways. Its design prioritizes reliability, performance, and ease of use, making it a valuable asset in the modern hospitality industry.

# REFERENCES

1. <https://docs.spring.io/spring-framework/reference/>
2. <https://docs.spring.io/spring-boot/docs/current/reference/html/>
3. <https://hibernate.org/orm/documentation/>
4. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
5. Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387.
6. Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison- Wesley.
7. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
8. **Gamma, E., Helm, R., Johnson, R., & Vlissides, J.** (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
9. **Pressman, R. S., & Maxim, B. R.** (2020). *Software Engineering: A Practitioner’s Approach*. McGraw-Hill.
10. **Booking.com API Documentation** – https://developers.booking.com/ (for third-party booking integration concepts).
11. **Stripe Payment API Documentation** – <https://stripe.com/docs/api> (for secure online payments in hotel systems).
12. **Soni, D., Nord, R. L., & Hofmeister, C.** (1995). *Software Architecture in Industrial Applications*. *Proceedings of the 17th International Conference on Software Engineering*, 196–207.

29