Hotel Management System

A PROJECT REPORT

Submitted To:

Professor Tanya Ma'am

Submitted By:

Name: Manpreet Kaur

UID: 24MCA20186

Section: 24MCA-7A

in partial fulfilment for the award of the degree of

Master of Computer Application

Chandigarh University

Aug 2024 – Nov 2024

Declaration

I hereby declare that the project report entitled **“Hotel Management System”** submitted in partial fulfilment of the requirements for the degree of Master in Computer Application is my original work and has not been submitted previously to any other institution or university for any degree or diploma.

I have carried out the work presented in this project report under the guidance of **Tanya Ma’am.** All the work presented here is genuine to the best of my knowledge, and any reference used has been appropriately acknowledged.

I am fully responsible for the authenticity of the content and any inaccuracies in the report.

Acknowledgment

I would like to express my sincere gratitude to everyone who has contributed to the successful completion of this case study on the **Hotel Management System** in AIP.

First and foremost, I am grateful to my instructor **Tanya Ma'am**, whose continuous guidance, valuable insights, and unwavering support have been instrumental throughout this project. Her expert advice and suggestions have helped me in understanding the core concepts of web application development, Java Servlets, and database connectivity in real-world scenarios.

My heartfelt thanks to my friends, family, and colleagues for their encouragement and moral support during this journey. Their constant motivation kept me focused and dedicated to the successful completion of this study.

Lastly, I would like to acknowledge the contribution of various online resources, documentation, and textbooks on web technologies and relational databases that provided me with the theoretical foundation necessary to carry out this case study.

This project has been a valuable learning experience, allowing me to bridge the gap between academic concepts and practical application in the field of full-stack web development.

Manpreet kaur

Table of Content

Introduction

1.1 Background

With the increasing demand for seamless guest services and efficient internal operations, many hotels are transitioning to digital management systems. Manual record-keeping often leads to inefficiencies, such as double-booking or slow check-ins. This project attempts to address those challenges by developing a simple yet functional **Hotel Management System**.

1.2 Project Objective

The primary goal of this project is to create a fully functional hotel management system using only **Java technologies** without relying on external frameworks like Hibernate. It is designed to:

- Manage hotel rooms and bookings.
- Maintain guest and employee records.
- Implement login and logout functionality with session tracking.
- Offer a clean and responsive UI using Bootstrap.
- Use RESTful services for frontend-backend communication.
- Provide theme toggle support (dark/light mode) for better UX.

1.3 Scope

The project covers all basic hotel operations from room availability to customer management. It is not limited to administrators only but is built in a scalable way to support different roles in the future. While billing and payments are excluded for simplicity, the system supports:

- Room listing and availability status.
- Guest check-in/check-out processes.
- Secure login/logout.
- API integration through Java Servlets and Fetch API.

Technology Stack

Java Servlets

Java Servlets form the core of the backend functionality. They handle client requests, process data, interact with the database, and return responses. Servlets are responsible for controlling the flow of the application, managing sessions, and exposing RESTful endpoints that the frontend communicates with.

JSP (JavaServer Pages)

JSP is used for rendering dynamic web pages. It allows embedding Java code directly into HTML, which helps generate content based on user interactions or server-side data. In this project, JSP files serve as the view layer, displaying data such as room availability, bookings, and guest details retrieved from the backend.

HTML & CSS

HTML provides the structure of the web pages, defining elements like forms, tables, buttons, and headers. CSS is used to apply styles to these elements, ensuring that the layout, fonts, colors, and spacing remain consistent and visually appealing across the entire application.

Bootstrap

Bootstrap is a CSS framework included to make the UI responsive and mobile-friendly. It simplifies the design process by offering pre-built components and classes for layout, buttons, forms, and grid systems. Bootstrap is also used to implement the dark/light theme toggle, enhancing the visual experience.

MySQL

MySQL is the database management system used to store all essential data, including user credentials, room records, booking information, and check-in/check-out logs. The database is accessed using JDBC from the Java Servlets, allowing the application to perform operations like inserting, updating, and retrieving records.

Fetch API

The Fetch API enables communication between the frontend (JSP) and backend (Java Servlets) through asynchronous HTTP requests. It is used to send data to REST endpoints (e.g., making a room booking or fetching guest details) and receive JSON responses without reloading the page.

Jakarta EE

Jakarta EE (formerly Java EE) provides the platform for building the web application. It includes specifications for Servlets, JSP, and RESTful services, enabling structured development and deployment. The project leverages Jakarta EE's architecture to organize logic and manage web requests efficiently.

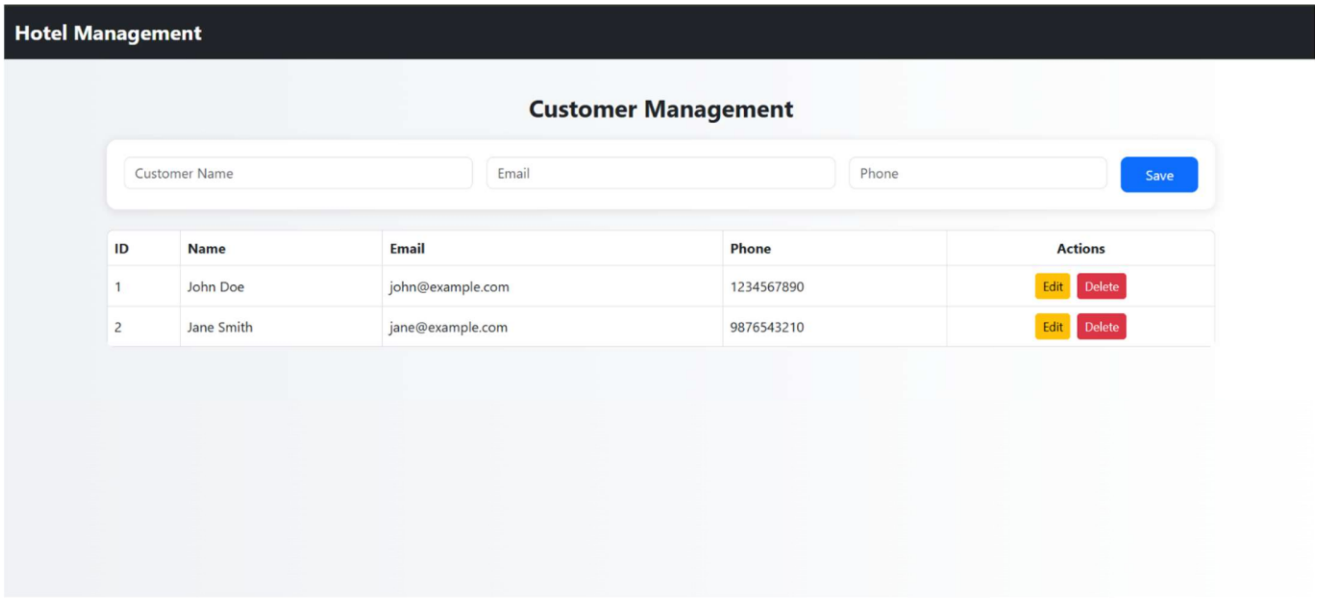
Apache Tomcat

Apache Tomcat is the server used to deploy and run the hotel management web application. It supports the execution of Servlets and JSP pages and acts as the middleware between the user interface and server-side logic. Tomcat is essential for hosting and testing the application during development.

Project Structure Overview

3.1 Page (index.jsp)

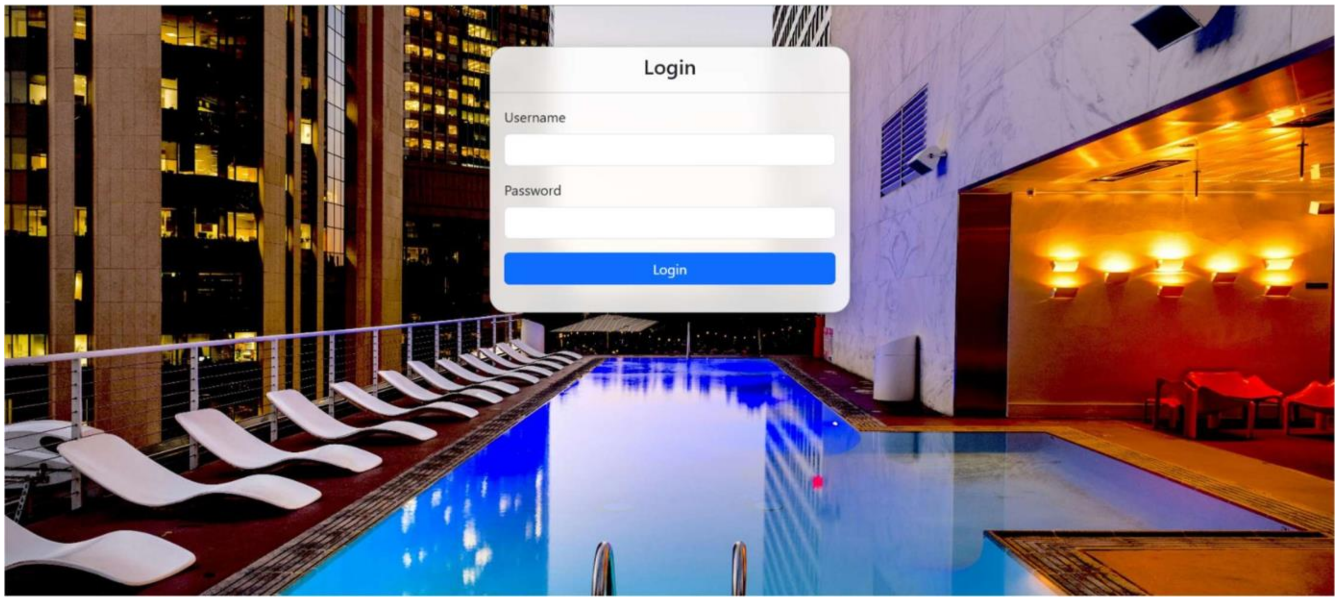
- Welcome message, navigation links to login and booking.
- UI toggle (light/dark mode) implemented with Bootstrap classes.



ID	Name	Email	Phone	Actions
1	John Doe	john@example.com	1234567890	Edit Delete
2	Jane Smith	jane@example.com	9876543210	Edit Delete

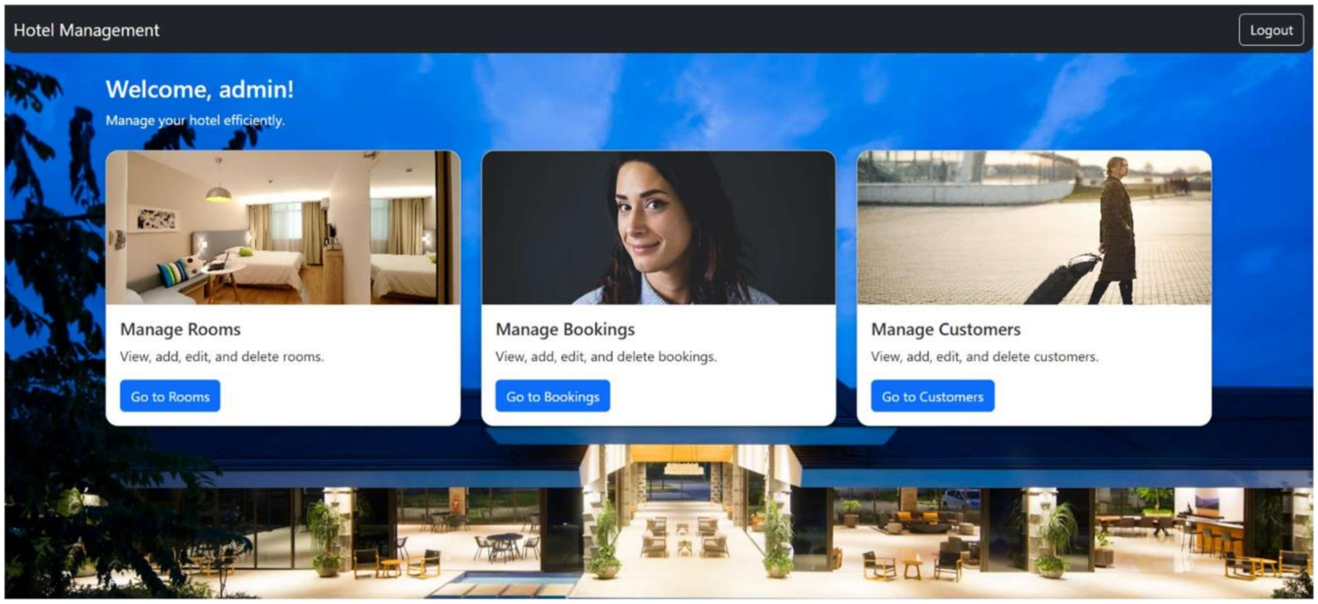
3.2 Admin Login (login.jsp)

- Admin registration with basic validation.
- Login functionality with session tracking.



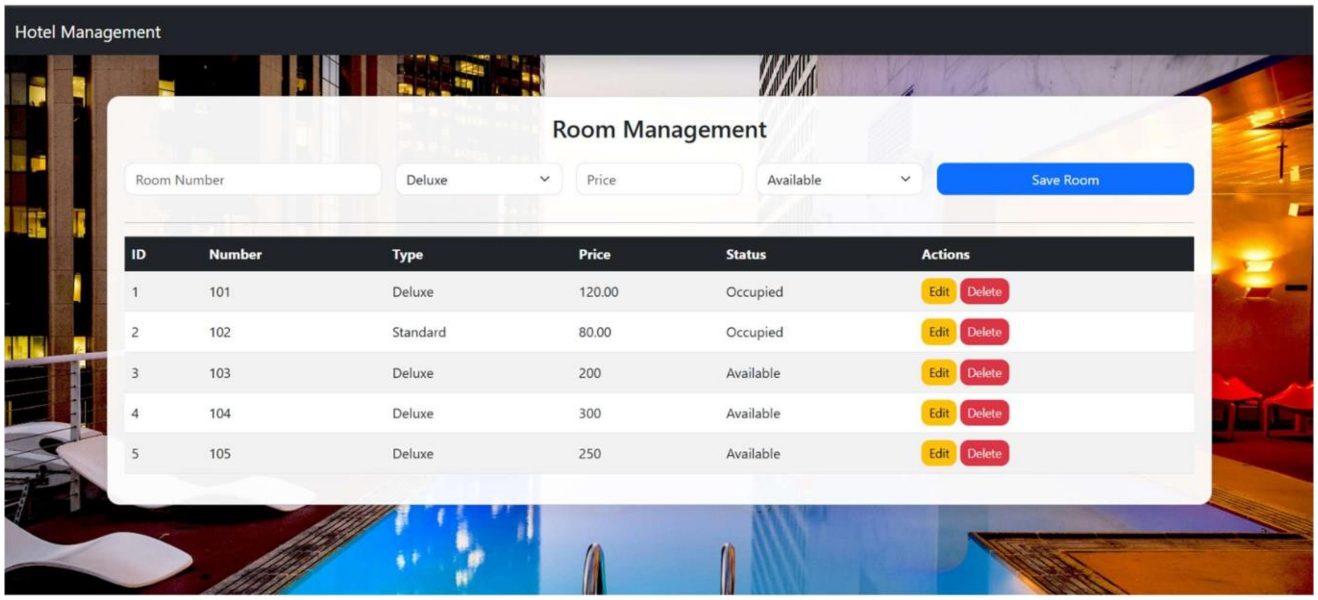
3.3 Admin Dashboard (adminDashboard.jsp)

- Restricted access with admin authentication.
- View and manage customers and room data.

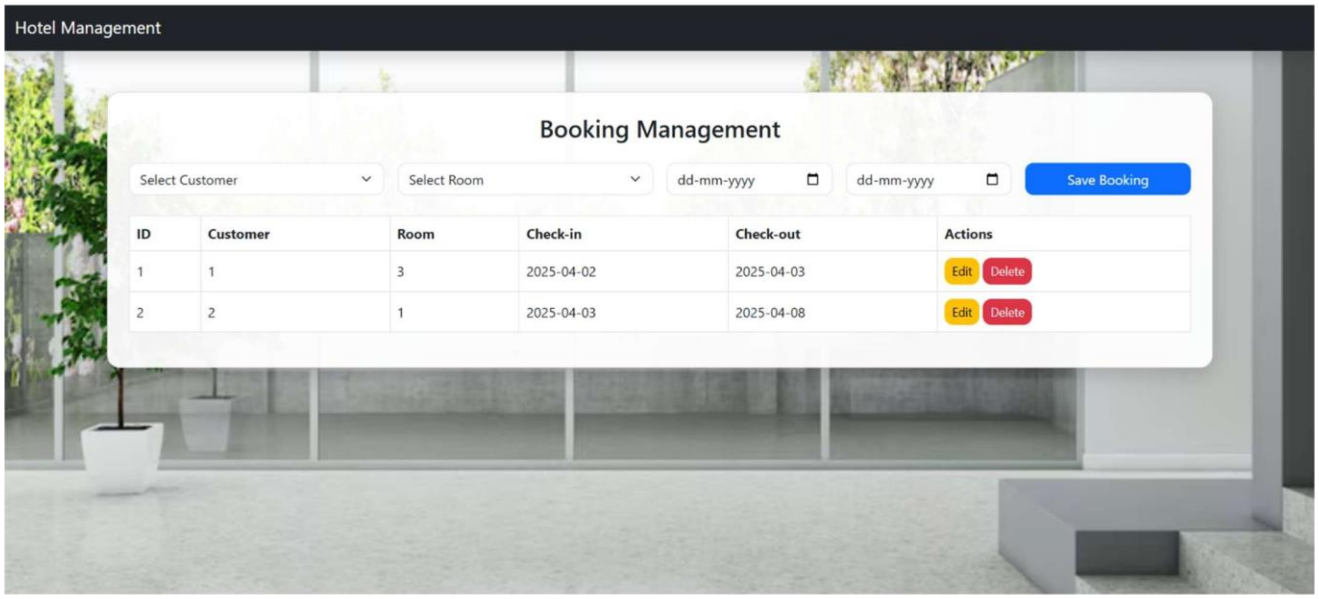

Hotel Management
Logout
Welcome, admin!
Manage your hotel efficiently.
Manage Rooms
View, add, edit, and delete rooms.
Go to Rooms
Manage Bookings
View, add, edit, and delete bookings.
Go to Bookings
Manage Customers
View, add, edit, and delete customers.
Go to Customers

3.4 Booking Page (booking.jsp)

- Shows available rooms fetched via REST API.
- Allows customers to make a booking and view status.



ID	Number	Type	Price	Status	Actions
1	101	Deluxe	120.00	Occupied	Edit Delete
2	102	Standard	80.00	Occupied	Edit Delete
3	103	Deluxe	200	Available	Edit Delete
4	104	Deluxe	300	Available	Edit Delete
5	105	Deluxe	250	Available	Edit Delete



ID	Customer	Room	Check-in	Check-out	Actions
1	1	3	2025-04-02	2025-04-03	Edit Delete
2	2	1	2025-04-03	2025-04-08	Edit Delete

3.6 Servlets

- LoginServlet.java: Handles session login.
- LogoutServlet.java: Handles session destruction.
- BookingServlet.java: Handles room booking.
- RoomAPI.java: REST endpoint to retrieve available rooms (returns JSON).

Key Functionalities

4.1 Room Booking

- Check availability using REST API.
- Confirm booking and update database via servlet.

4.2 Admin Control Panel

- Add/view/delete customer and room entries.
- Simple table UI for CRUD operations via JSP and Java logic.

4.3 Theme Toggle

- Light/dark theme using Bootstrap switch.
- State retained using browser local storage.

Sample Code Snippets

5.1 Room Booking Logic in Servlet

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    String customerName = request.getParameter("name");
    int roomId = Integer.parseInt(request.getParameter("roomId"));

    Connection con = DriverManager.getConnection(DB_URL, USER, PASS);
    PreparedStatement ps = con.prepareStatement("INSERT INTO bookings (customer_name, room_id)
VALUES (?, ?)");
    ps.setString(1, customerName);
    ps.setInt(2, roomId);
    ps.executeUpdate();
    response.sendRedirect("bookingSuccess.jsp");
}
```

5.2 REST API for Room Availability

```
@Path("/rooms")
public class RoomAPI {
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Room> getAvailableRooms() {
        List<Room> roomList = new ArrayList<>();
        // Fetch from DB and return as JSON
        return roomList;
    }
}
```

Testing and Debugging

1. **Login Test:** Valid/invalid user credentials tested.
2. **Room API Test:** REST endpoints tested via Fetch + Chrome Dev Tools.
3. **Mobile Compatibility:** Bootstrap ensures responsiveness on phones.
4. **Session Expiry:** Accessing protected pages after logout redirects to login.

Limitations and Challenges

Limitations

- No payment integration (could be added later).
- Basic input validation only.

Challenges

- Integrating REST APIs into JSP without JavaScript logic.
- Managing state across JSP pages without using DAO or Hibernate.

Future Improvements

1. **Payment Gateway Integration:** Add support for online payments.
2. **Analytics Dashboard:** Graphical view of occupancy rates.
3. **Email Notifications:** Confirmation emails for bookings.
4. **Database Optimization:** Add indexing and query tuning.

Conclusion

This Hotel Management System showcases how a full-featured Java web application can streamline hotel operations. With a modular architecture using Servlets, REST APIs, and JSP, the system is both scalable and extendable. It demonstrates the effective use of core Java EE technologies, Bootstrap for UI, and SQL for robust data storage.

References

- **HTML Documentation**: https://developer.mozilla.org/en-US/docs/Web/HTML
- **CSS Documentation**: https://developer.mozilla.org/en-US/docs/Web/CSS
- **JavaScript (Fetch API) Documentation**: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- **Bootstrap Documentation**: https://getbootstrap.com/docs/5.3/getting-started/introduction/
- **Java Servlets (Jakarta EE) Documentation**: https://jakarta.ee/specifications/servlet/
- **JSP (JavaServer Pages) Documentation**: https://jakarta.ee/specifications/pages/
- **Apache Tomcat Documentation**: https://tomcat.apache.org/tomcat-10.1-doc/index.html
- **MySQL Documentation**: https://dev.mysql.com/doc/
- **JDBC (Java Database Connectivity)**: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/