CHAPTER 1: COMPANY PROFILE

Softech Infoways, Sirsa:

Softech Infoways, located in Agrasain Colony, Sirsa, Haryana, has been a prominent institution in the region, specializing in computer education and skill development. Established in 2004, it gained recognition for offering a range of training programs aimed at enhancing technical proficiency among students and professionals.

Service Offered:

Softech Infoways provided various courses and training sessions, including:

- C++ Programming
- Multimedia
- Computer Science (CS) Exam Assistance
- PGDCA (Post Graduate Diploma in Computer Applications)
- JavaScript

These programs were designed to equip individuals with practical knowledge and skills essential for the evolving IT industry.

Infrastructure:

The center was known for its well-equipped facilities, providing a conducive learning environment with modern computers and software tools, facilitating hands-on training for students.

Reputation:

Softech Infoways earned a positive reputation among its clientele, reflected in its 4.5-star rating based on 56 reviews. Students appreciated the quality of education and the practical approach adopted in the training sessions.

Conclusion:

Softech Infoways played a significant role in imparting computer education in Sirsa, contributing to the skill development of many individuals. While it is no longer operational, its impact on the local community's technological advancement remains noteworthy.

CHAPTER 2: INTRODUCTION

2.1 Background and Motivation :

In the modern digital era, the hospitality industry increasingly relies on technology to enhance user convenience and improve operational efficiency. Online hotel booking platforms have revolutionized how guests search, view, and reserve accommodations. With the growing trend toward seamless and aesthetic web experiences, creating a responsive and visually appealing hotel booking system has become a fundamental requirement for premium brands such as Radisson.

The "**Radisson Hotel Booking System**" project aims to design and develop a **fully functional, interactive, and visually rich hotel website** using **React.js**, combining modern UI/UX design principles with efficient front-end architecture. React's component-based structure provides scalability, responsiveness, and high performance, making it ideal for building dynamic interfaces like room galleries, booking forms, and user authentication systems.

The motivation behind this project is to simulate a professional-grade hotel booking experience that reflects luxury, reliability, and smooth navigation. The project integrates **modern frontend technologies**, **glassmorphic UI design**, and **dynamic interactions**, giving users an elegant and immersive experience comparable to real-world hotel platforms.

2.2 Project Objectives :

The objective of this project is to develop a **React-based Hotel Booking Web Application** inspired by **Radisson's premium standards**. The major objectives include:

1. **Responsive User Interface:**
   Develop a visually appealing, mobile-friendly layout with a luxury-themed aesthetic and glassmorphism effects.
2. **Room Display and Carousel Integration:**
   Showcase different room types (Superior, Deluxe, Executive, and Presidential Suites) through interactive carousels supporting multiple images per room.
3. **Booking Functionality:**
   Implement booking forms that allow users to select check-in/check-out dates, room types, and guest details, integrated with backend APIs.
4. **User Authentication:**
   Provide login, sign-up, and admin modules with secure API communication for managing users and bookings.
5. **Dynamic Animations and Hover Effects:**
   Incorporate smooth transitions, 3D room card animations, and responsive hover effects to improve engagement.

6. **Admin Management Panel:**
   Create an admin dashboard for viewing and managing bookings, customer data, and room information.

2.3 Scope and Methodology :

2.3.1 React Environment :

React.js was chosen as the primary front-end framework for its **component reusability**, **virtual DOM efficiency**, and **scalability**. It allows modular development and seamless integration of visual effects and dynamic features.

2.3.2 React's Role and Core Characteristics :

A. **Component-Based Design:**
Each section (Navbar, Rooms, Booking Form, Login, Admin Dashboard) is implemented as an independent component for reusability and clarity.

B. **State and Props Management:**
React's state and props mechanism helps manage UI reactivity and real-time data flow across components.

C. **Integration with Backend APIs:**
Axios is used to handle HTTP requests for booking, authentication, and admin operations efficiently.

D. **Responsive Design:**
CSS Flexbox and media queries ensure smooth rendering across different devices and screen sizes.

2.3.3 Application Flow and Logic :

The application follows a clear flow from **homepage navigation to booking confirmation**. Conceptual flow:

User Visits Website →

Views Rooms & Details →

Selects Room →

Fills Booking Form →

Submits Data to Backend →

Receives Confirmation

2.3.4 Admin and User Interaction Logic :

- **User Side:** Handles room browsing, form inputs, login/signup, and booking confirmations.
- **Admin Side:** Manages all booking entries, customer details, and room categories through secure APIs

CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATION

3.1 Functional Requirements (FRs)

ID	Requirement Title	Description	Priority	Component
FR01	Homepage & Navigation	Display the main homepage with a luxury-themed layout, including navigation to Rooms, Booking, Login, and Admin pages.	High	React Router, Navbar Component
FR02	Room Display Carousel	Show multiple room categories (Superior, Deluxe, Executive Suite, Presidential Suite) using image carousels with smooth transitions.	High	RoomCard Component, Carousel
FR03	Booking Form Functionality	Allow users to enter details such as name, email, dates, and room type for booking confirmation.	High	BookingForm Component, Input Fields
FR04	User Authentication	Enable users to log in, sign up, or reset passwords through secure API endpoints.	High	Login/Signup Component, Axios API
FR05	Admin Module	Allow administrators to manage room details, bookings, and customer data through an interactive dashboard.	High	AdminModule Component, Backend APIs
FR06	Responsive Design	Ensure that the website adapts to various screen sizes including mobile, tablet, and desktop views.	High	CSS Flexbox, Media Queries
FR07	Dynamic UI Effects	Implement hover animations, 3D card rotation effects, and smooth page transitions for visual appeal.	Medium	CSS Animations, Framer Motion

FR08	Booking Confirmation	After submitting a booking form, display a confirmation message or modal popup with user details.	High	Modal Component, API Response Handling
FR09	Search and Filter (Optional)	Provide an option to search or filter room types by price or capacity.	Medium	Filter Component, State Management
FR10	Error Handling & Validation	Display error messages for missing inputs, invalid data, or failed API calls.	High	Form Validation Logic, Toast Notifications

---

3.2 Non-Functional Requirements (NFRs)

ID	Requirement Title	Description	Metric/Constraint	Component
NFR01	Performance	Website should load under 3 seconds and handle API requests efficiently.	Page Load ≤ 3s	React Optimization, Axios
NFR02	Responsiveness	Layout and content should render correctly across all screen sizes.	Supports ≥ 95% screen sizes	CSS Flexbox, Media Queries
NFR03	Portability	Website should run smoothly across browsers (Chrome, Edge, Firefox, Safari).	Browser Compatibility ≥ 95%	React Frontend
NFR04	Maintainability	Code should be modular and well-commented for future enhancements.	Modular Components	React Component Architecture

NFR05	Security	Protect user credentials and booking data using secure API communication.	HTTPS, Encrypted Data	Backend API, JWT/Auth Logic
NFR06	Scalability	The system should easily support additional rooms, features, or new modules.	Handle ≥ 10,000 users	React Component-Based Design
NFR07	Visual Consistency	Maintain a consistent glassmorphic and luxury UI throughout the application.	Uniform Design System	CSS Variables, Global Styles
NFR08	Reliability	Ensure stable functionality even with heavy traffic or simultaneous bookings.	99% Uptime Target	Backend API & Hosting
NFR09	Usability	The interface should be intuitive for both customers and admin users.	95% Task Success Rate	UI/UX Design
NFR10	Accessibility	Follow web accessibility guidelines for inclusive design (contrast, labels, alt-text).	WCAG 2.1 Compliance	UI Components

CHAPTER 4: SYSTEM ANALYSIS

4.1 System Context and Boundary :

The system is a **web-based hotel booking application** where users interact through a **graphical user interface built using React.js**. It allows customers to view rooms, book stays, and manage their bookings, while administrators can manage hotel data through a secure dashboard.

- **System Input:**
  User inputs through web forms (e.g., room selection, dates, login credentials, booking details).
- **System Output:**
  Dynamic web pages showing room details, booking confirmation modals, and admin dashboard updates.
- **Boundary:**
  The system operation begins when the user accesses the website and ends when the user closes the browser session or completes the booking confirmation process.

4.2 High-Level System Architecture

1. **Frontend (React.js):**
   Provides an interactive user interface for customers and admins. Components like Navbar, RoomCard, BookingForm, and AdminModule handle user interactions, animations, and data presentation.
2. **Backend API (Node.js/Express):**
   Manages authentication, booking storage, and data retrieval through RESTful APIs.
3. **Database (MongoDB/MySQL):**
   Stores user credentials, room details, and booking information persistently.
4. **Axios Middleware:**
   Acts as the communication layer between frontend components and backend APIs for data transmission.
5. **Admin Dashboard:**
   Enables the admin to view, add, update, and delete bookings and room information securely.

Component Interaction Overview:

User → React Frontend → Axios → Node.js API → Database → Response to Frontend

---

4.3 Data Flow Analysis

Step	Process Description	Responsible Component	Key Data Transformation
1	Capture User Input	BookingForm, LoginForm	User enters booking info, dates, or login credentials.
2	Validate Input	Frontend Validation Logic	Data is checked for completeness and correctness.
3	Send Data to Backend	Axios (Frontend Middleware)	Form data converted to JSON and sent via POST request.
4	Process Request	Node.js/Express Server	Backend verifies data, processes bookings, and interacts with database.
5	Update Database	MongoDB/MySQL	New booking, user, or room entry stored or updated.
6	Return Response	Backend API → Frontend	Confirmation message or error returned as JSON.
7	Display Output	React Components (Modal, Dashboard)	Booking confirmation, room updates, or error messages shown dynamically.

CHAPTER 5: HARDWARE REQUIREMENTS

5.1 Minimum Hardware Requirements

Component	Minimum Specification	Impact on Performance
Processor (CPU)	Dual-Core 2.0 GHz or higher	Handles React build processes, API calls, and local server execution efficiently.
Memory (RAM)	4 GB	Sufficient for running React development server and lightweight backend simultaneously.
Storage	500 MB available space	Required for Node.js modules, project files, and browser cache.
Graphics (GPU)	Integrated Graphics	Supports rendering animations, transitions, and UI effects smoothly in browsers.
Display Resolution	1366 × 768 pixels	Minimum screen resolution for viewing responsive layouts correctly.
Network Connectivity	Stable Internet Connection	Needed for accessing APIs, rendering hosted images, and testing backend endpoints.

CHAPTER 6: SOFTWARE REQUIREMENTS

6.1 Minimum Software Requirements

Component	Software / Tool	Version / Specification	Purpose / Description
Operating System	Windows 10 / Linux / macOS	Any modern version	Provides the environment for running the development tools and local servers.
Frontend Framework	React.js	18.x or later	Used for building the interactive and responsive user interface components.
Backend Environment	Node.js with Express.js	Node 18.x / Express 4.x	Handles server-side logic, API creation, and database communication.
Database	MongoDB / MySQL	Latest stable version	Stores user details, booking information, and room data persistently.
Code Editor	Visual Studio Code (VS Code)	Latest version	Primary IDE for coding, debugging, and running the development environment.
Package Manager	npm (Node Package Manager)	9.x or later	Installs and manages project dependencies for both frontend and backend.
Browser	Google Chrome / Microsoft Edge / Mozilla Firefox	Latest version	Used to view, test, and debug the web application interface.
Version Control	Git & GitHub	Latest release	Manages source code versions and project collaboration.
Design & Styling Tools	CSS3, TailwindCSS / Bootstrap	Any recent version	Used for creating the glassmorphic and responsive user interface design.
API Testing Tool	Postman	Latest version	For testing backend API endpoints and verifying data responses.

CHAPTER 7: IMPLEMENTATION

7.1 Overview

The **Radisson Hotel Booking Web Application** has been implemented using **React.js** for the frontend, **Node.js with Express.js** for the backend, and **MongoDB** as the database. The system provides both customer and admin modules, featuring functionalities like room browsing, interactive booking, and secure authentication.

The implementation follows a **component-based approach**, ensuring modularity, scalability, and maintainability across the system. Each feature—such as room display, booking form, or admin dashboard—is encapsulated in a dedicated component with its own logic, state, and styling.

7.2 Frontend Implementation (React.js)

The frontend was developed using **React.js (v18)** with **functional components** and **React Hooks** for managing state and side effects. The application is fully responsive and includes interactive UI effects inspired by Radisson's elegant design standards.

Key Frontend Components:

1. **App.js** – Root component controlling routing and layout.
2. **Navbar.js** – Navigation bar for quick access to Home, Rooms, Booking, and Login pages.
3. **Rooms.js** – Displays various room types (Superior, Deluxe, Executive Suite, Presidential Suite) using a carousel layout.
4. **BookingForm.js** – Allows users to enter guest details, check-in/check-out dates, and room preferences.
5. **Login.js / Signup.js** – Manages customer authentication and form validation.
6. **AdminModule.js** – Provides administrative features such as viewing bookings, managing room data, and handling user details.
7. **Modal.js** – Displays booking confirmation messages.

Frontend Logic Example:

```
// Example: BookingForm.js
import React, { useState } from "react";
import axios from "axios";

const BookingForm = () => {
  const [formData, setFormData] = useState({
    name: "", email: "", roomType: "", checkIn: "", checkOut: ""
  });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    const res = await axios.post("https://hotel-backend.onrender.com/book", formData);
    alert(res.data.message || "Booking Successful!");
  };

  return (
    <form className="booking-form" onSubmit={handleSubmit}>
      <input name="name" placeholder="Full Name" onChange={handleChange} required />
      <input name="email" placeholder="Email" type="email" onChange={handleChange}
required />
      <select name="roomType" onChange={handleChange} required>
        <option>Select Room Type</option>
        <option>Deluxe Room</option>
        <option>Executive Suite</option>
```

```
    </select>
    <input name="checkIn" type="date" onChange={handleChange} required />
    <input name="checkOut" type="date" onChange={handleChange} required />
    <button type="submit">Book Now</button>
  </form>
 );
};
export default BookingForm;
```

7.3 Backend Implementation (Node.js and Express.js)

The backend was implemented using **Node.js** with **Express.js** framework to handle RESTful APIs. It manages user authentication, booking operations, and data retrieval from the database.

Key Backend Features:

1. **User Authentication APIs** – For user signup, login, and admin authentication.
2. **Booking API** – Handles booking creation, retrieval, and cancellation.
3. **Admin API** – Allows admin to manage all user and booking records.

Backend Logic Example:

```
// Example: bookingController.js
const express = require("express");
const router = express.Router();
const Booking = require("../models/Booking");

router.post("/book", async (req, res) => {
 try {
  const { name, email, roomType, checkIn, checkOut } = req.body;
  if (!name || !email || !roomType || !checkIn || !checkOut) {
   return res.status(400).json({ message: "All fields are required" });
  }
```

```
    const newBooking = new Booking({ name, email, roomType, checkIn, checkOut });
    await newBooking.save();
    res.json({ message: "Booking successful!" });
  } catch (err) {
    res.status(500).json({ message: "Server Error" });
  }
});

module.exports = router;
```

7.4 Database Implementation (MongoDB)

MongoDB was used for data persistence. It stores user credentials, booking information, and room details in JSON-like collections. The **Mongoose ORM** simplifies schema creation and database operations.

Booking Schema Example:

```
// models/Booking.js
const mongoose = require("mongoose");

const bookingSchema = new mongoose.Schema({
  name: String,
  email: String,
  roomType: String,
  checkIn: String,
  checkOut: String,
  createdAt: { type: Date, default: Date.now }
});

module.exports = mongoose.model("Booking", bookingSchema);
```

7.5 Data Flow Summary

Step	Action	Component	Data Handled
1	User fills booking form	React (BookingForm.js)	Form Data
2	Data sent to backend API	Axios POST request	JSON Request
3	Backend validates and stores booking	Node.js + MongoDB	Booking Record
4	Server sends confirmation response	Express API	JSON Response
5	Frontend displays confirmation modal	React Modal Component	Confirmation Message

CHAPTER 8: TESTING

8.1 Overview

Testing ensures that the Radisson Hotel Booking website functions correctly and meets user requirements. The system was tested for functionality, performance, and responsiveness across various modules such as Login, Signup, Room Booking, and Admin Management.

8.2 Types of Testing

1. **Unit Testing:**
   Each component and backend API was tested individually. Modules like login, signup, and booking were verified for input validation and correct data flow.
2. **Integration Testing:**
   Verified smooth interaction between frontend (React), backend (Express), and database (MongoDB). Booking data was correctly saved and retrieved.
3. **System Testing:**
   End-to-end testing ensured that all modules worked together as a complete web application.
4. **UI and Responsiveness Testing:**
   Checked for proper layout, animations, and usability on desktop and mobile screens.
5. **Security Testing:**
   Ensured data validation, restricted admin access, and protection against invalid or malicious inputs.

8.3 Test Results

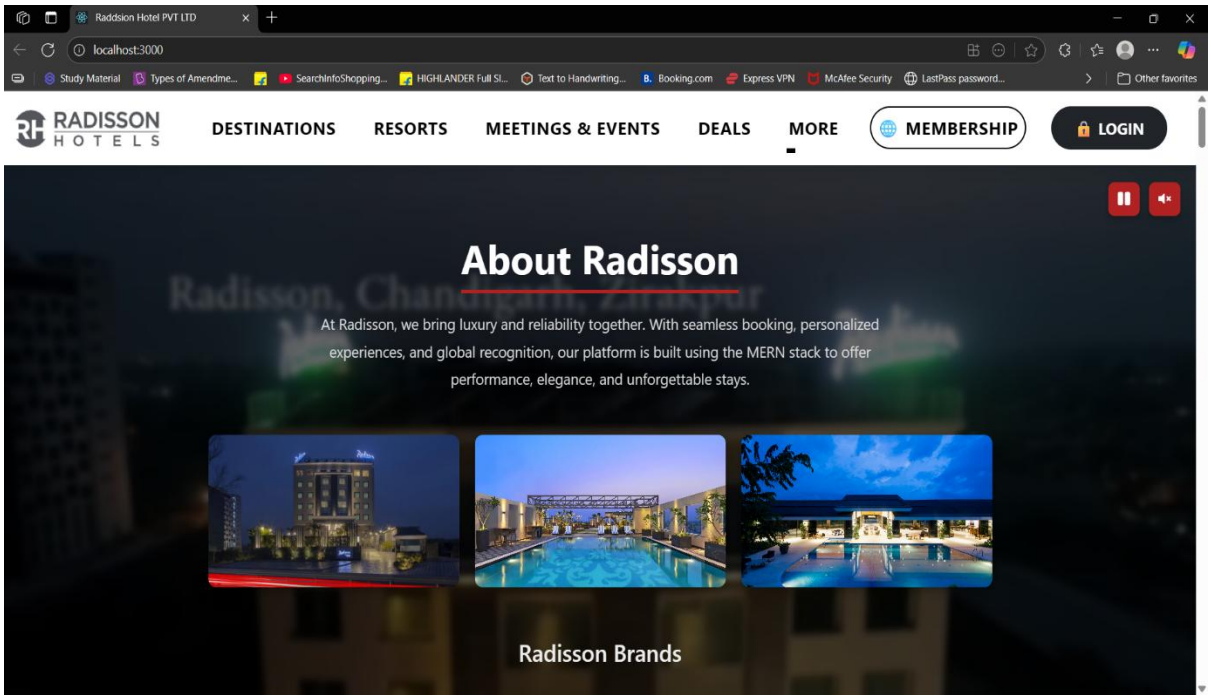
All modules were successfully tested.
The website was responsive, secure, and functioned smoothly across browsers.

Feature	Status
Login/Signup	Pass
Room Display	Pass
Booking Form	Pass
Admin Panel	Pass
Responsiveness	Pass

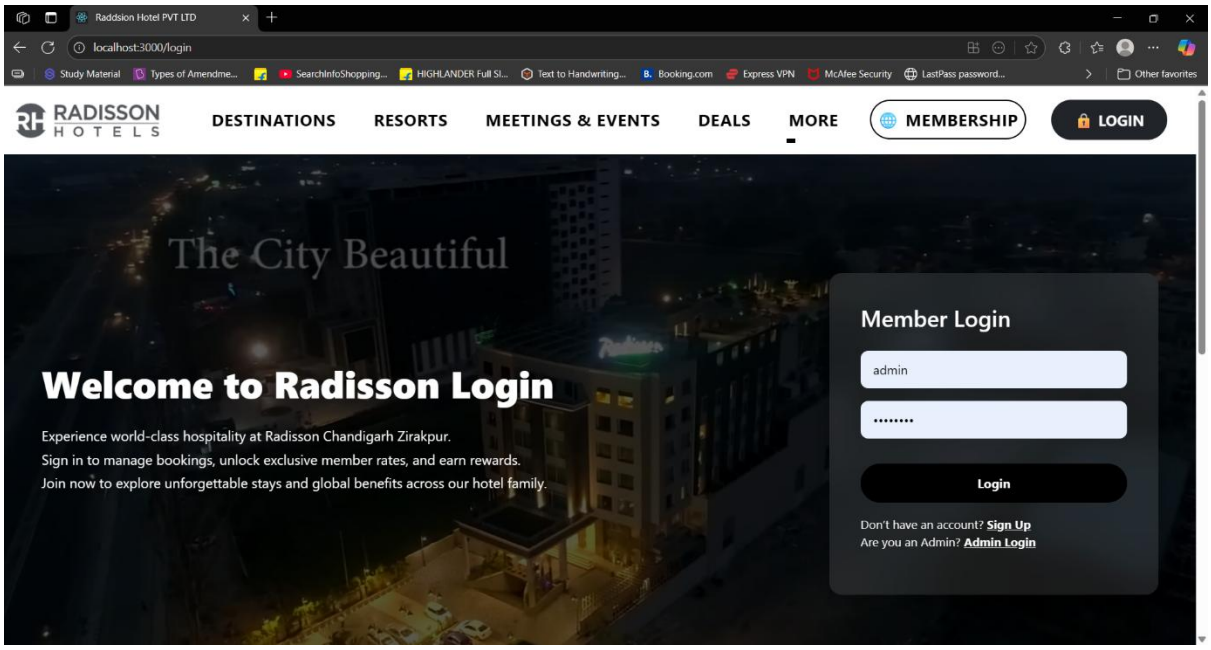
8.4 Conclusion

Testing confirmed that the system meets its functional and non-functional requirements.
The Radisson Hotel Booking Website performs efficiently, provides accurate results, and delivers a seamless user experience.

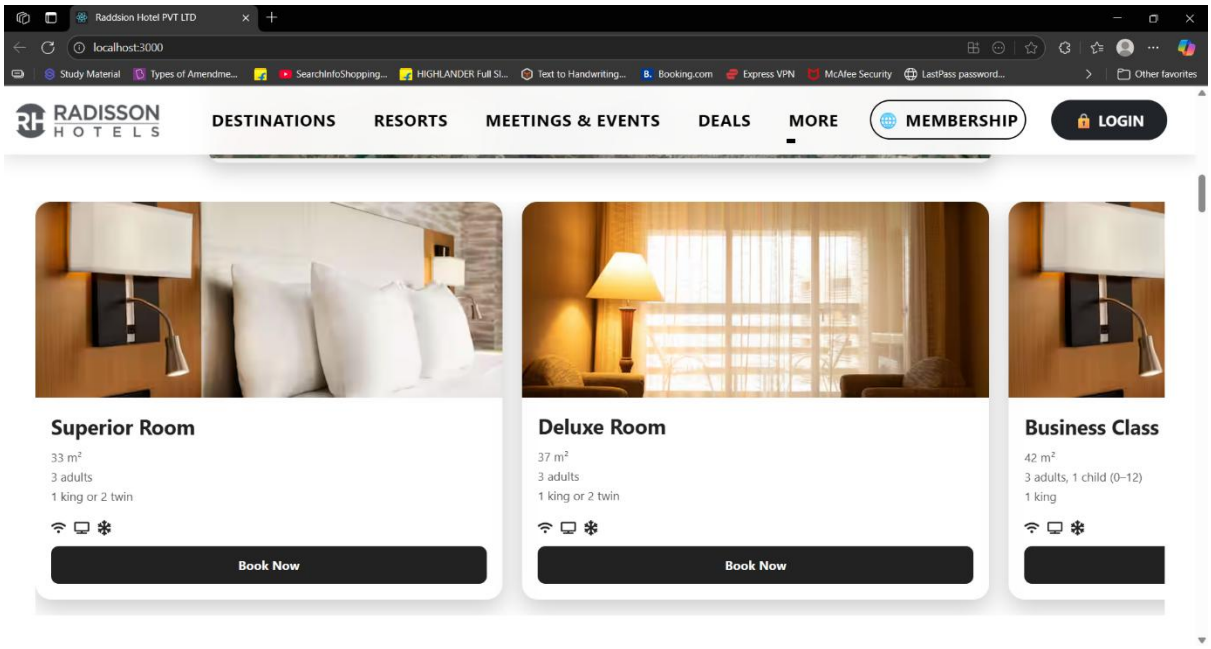
8.5 HOME:



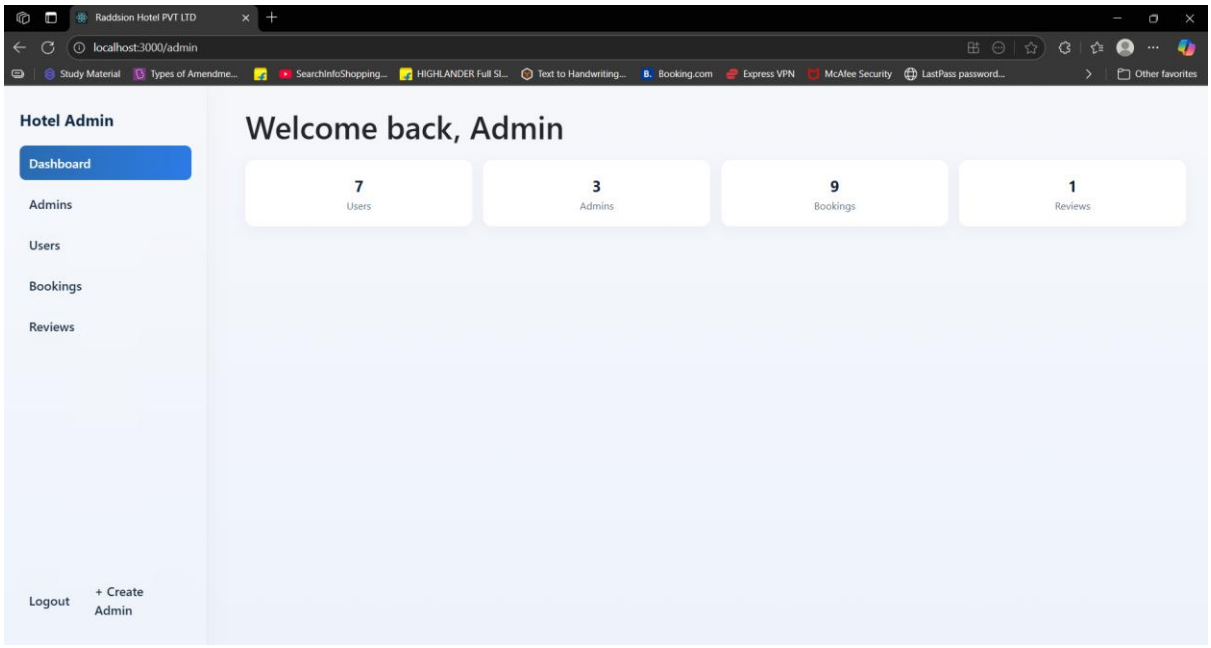
8.6 LOGIN:



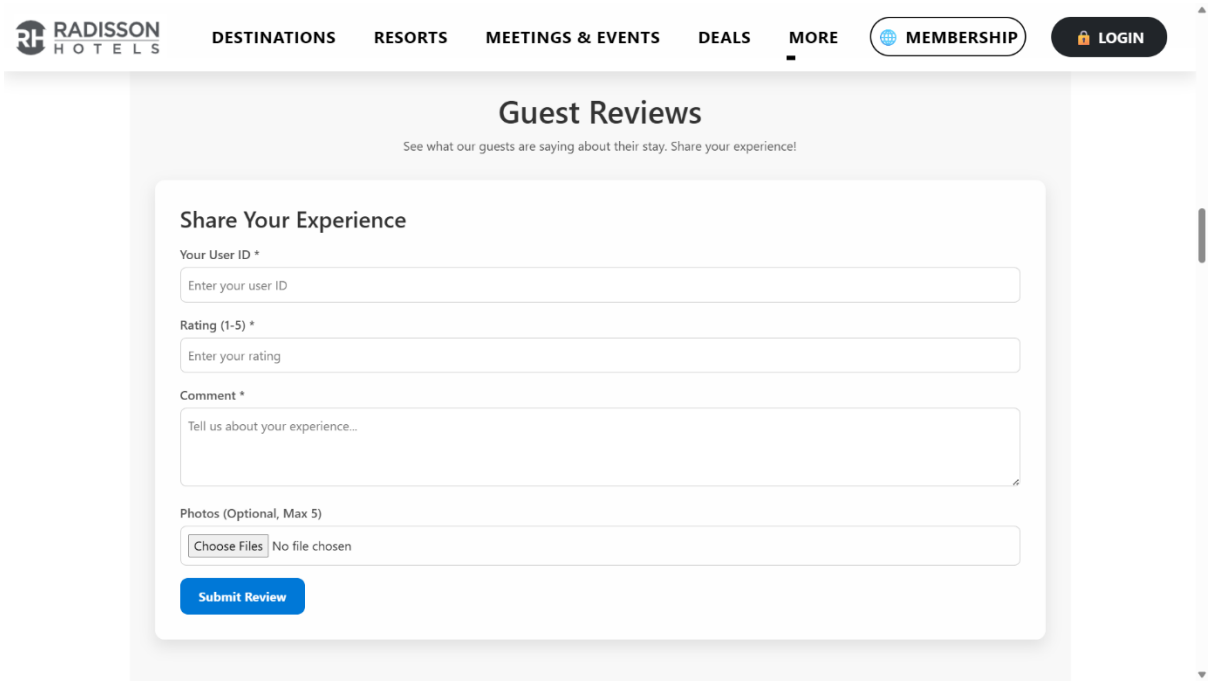
8.7 BOOKING:



8.8 ADMIN:



8.9 REVIEW:



CONCLUSION AND FUTURE SCOPE

Conclusion:

The Radisson Hotel Booking System successfully provides a smooth and user-friendly platform for customers to explore rooms, check availability, and make bookings online. It streamlines hotel management by connecting customers and administrators in a single digital interface.
All modules, including login, signup, booking, and admin management, have been implemented and tested effectively.
The project meets its primary goal of delivering a responsive, modern, and efficient hotel booking experience.

Future Scope:

1. **Online Payment Integration:**
   Adding secure online payment gateways like UPI, credit/debit cards, or PayPal.
2. **Advanced Admin Analytics:**
   Integration of dashboards for revenue tracking, booking trends, and customer insights.

3. **AI-based Room Suggestions:**
   Personalized room recommendations based on user history and preferences.
4. **Mobile Application:**
   Developing Android and iOS apps for better accessibility.
5. **Multi-language Support:**
   Expanding usability for international customers.

BIBLIOGRAPHY

1. *React Official Documentation* – https://react.dev
2. *Node.js Official Documentation* – https://nodejs.org/en/docs
3. *Express.js Guide* – https://expressjs.com
4. *MongoDB Documentation* – https://www.mongodb.com/docs
5. *Mongoose Documentation* – https://mongoosejs.com/docs
6. *Axios Documentation* – https://axios-http.com/docs/intro
7. *TailwindCSS Documentation* – https://tailwindcss.com/docs
8. *Visual Studio Code* – https://code.visualstudio.com/docs
9. *Postman Documentation* – https://learning.postman.com/docs/
10. *Radisson Hotels Official Website* – https://www.radissonhotels.com
11. *MDN Web Docs: HTML, CSS, JavaScript* – https://developer.mozilla.org