***B113 Daksh Rathi***

***Project Report: Hostel Management System (Roomify)***

**1. Project Overview**

**1.1 Project Title**

**Hostel Management System (Roomify)**

**1.2 Project Description**

The **Hostel Management System (Roomify)** is an application designed to efficiently manage the operations of a hostel. It aims to streamline the process of room allocation, fee management, student check-ins/outs, and maintenance requests. The system automates key processes, making it easier for hostel administrators to manage daily tasks while also providing a seamless experience for students.

**1.3 Problem Statement**

Hostel management involves various manual tasks such as room allocation, fee tracking, attendance monitoring, and maintenance requests, which are prone to errors and inefficiency. The lack of a centralized system often results in confusion, delays, and difficulty in maintaining records. This project aims to develop an integrated software solution that will automate and simplify hostel management processes, ensuring smoother operations and a better experience for both administrators and students.

**1.4 Project Objectives**

* **Automate Room Allocation**: Provide a system that allows for easy room allocation based on available rooms and student preferences.
* **Manage Hostel Fees**: Track hostel fees and generate reminders for payment.
* **Student Management**: Maintain student records, including their check-in/check-out dates, personal details, and room assignments.
* **Maintenance Requests**: Enable students to submit maintenance requests, which can be tracked by the hostel management team.
* **Reporting**: Generate reports on student occupancy, fee payments, and maintenance activities.

**2. System Design**

**2.1 Functional Requirements**

The system must include the following modules:

* **User Authentication**: Login functionality for students and hostel administrators.
* **Room Allocation**: A feature for room allocation based on student preferences and availability.
* **Fee Management**: Module to track hostel fee payments and send reminders for overdue payments.
* **Student Registration**: A form to register new students with their details such as name, course, contact, etc.
* **Maintenance Tracking**: Feature for students to request maintenance and for the hostel staff to track and resolve these requests.
* **Reporting**: Generate dynamic reports for administrators on occupancy, payments, and maintenance issues.

**2.2 Non-Functional Requirements**

* **Usability**: The system should be user-friendly, with intuitive navigation.
* **Scalability**: The system should support future expansions such as adding more rooms, students, or features.
* **Security**: All user data should be securely stored, with password protection and encrypted connections.
* **Performance**: The system should have quick response times for requests, even with large amounts of data.

**3. Technology Stack**

**3.1 Frontend Technologies**

* **HTML/CSS**: For structuring and styling the web pages.
* **JavaScript (React.js)**: To build a responsive and interactive user interface.
* **Bootstrap**: For designing a clean and mobile-friendly UI.

**3.2 Backend Technologies**

* **Node.js**: Server-side JavaScript runtime environment for handling requests.
* **Express.js**: A web framework for Node.js to build RESTful APIs.
* **MongoDB**: NoSQL database to store student and hostel information.
* **Mongoose**: ODM for MongoDB to interact with the database.

**3.3 Tools**

* **Git/GitHub**: For version control and collaborative development.
* **Postman**: For API testing and debugging.
* **Visual Studio Code**: Integrated Development Environment (IDE) for coding.

**4. System Architecture**

**4.1 Overview**

The system follows a **client-server architecture**, where the front-end communicates with the back-end through REST APIs. The client-side handles user interactions, while the server-side processes the requests and interacts with the database.

**4.2 Components**

* **Frontend (React.js)**: Handles the user interface and sends HTTP requests to the backend.
* **Backend (Node.js/Express.js)**: Handles business logic, processes data, and communicates with the database.
* **Database (MongoDB)**: Stores and retrieves data like student information, room details, and maintenance logs.

**5. Implementation**

**5.1 User Interface Design**

The user interface consists of:

* **Dashboard**: Displays a summary of hostel occupancy, fee status, and pending maintenance requests.
* **Room Allocation Page**: Displays available rooms and allows students to request a room.
* **Student Management**: Admins can view, add, or remove student records.
* **Maintenance Requests**: Allows students to submit maintenance requests, which are tracked by the management team.

**5.2 Database Design**

The database includes the following collections:

* **Students**: Stores student details such as name, course, contact information, room allocation, etc.
* **Rooms**: Stores details about available rooms such as room number, type, and status (occupied/vacant).
* **Fees**: Stores fee details, including the amount, due date, and payment status.
* **Maintenance Requests**: Stores maintenance issues, request dates, and status (open/closed).

**6. Testing and Evaluation**

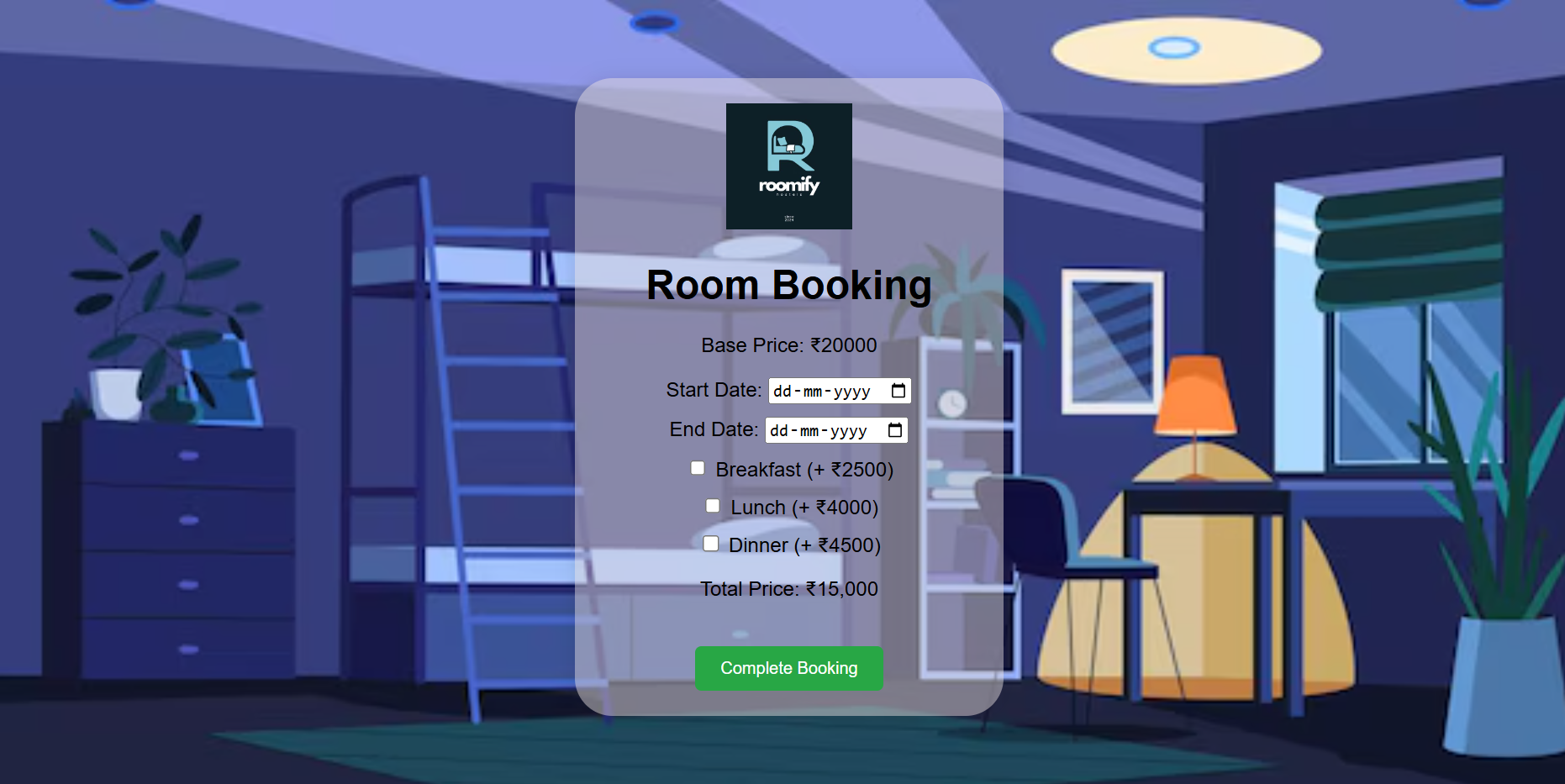
**6.1 Testing Methodologies**

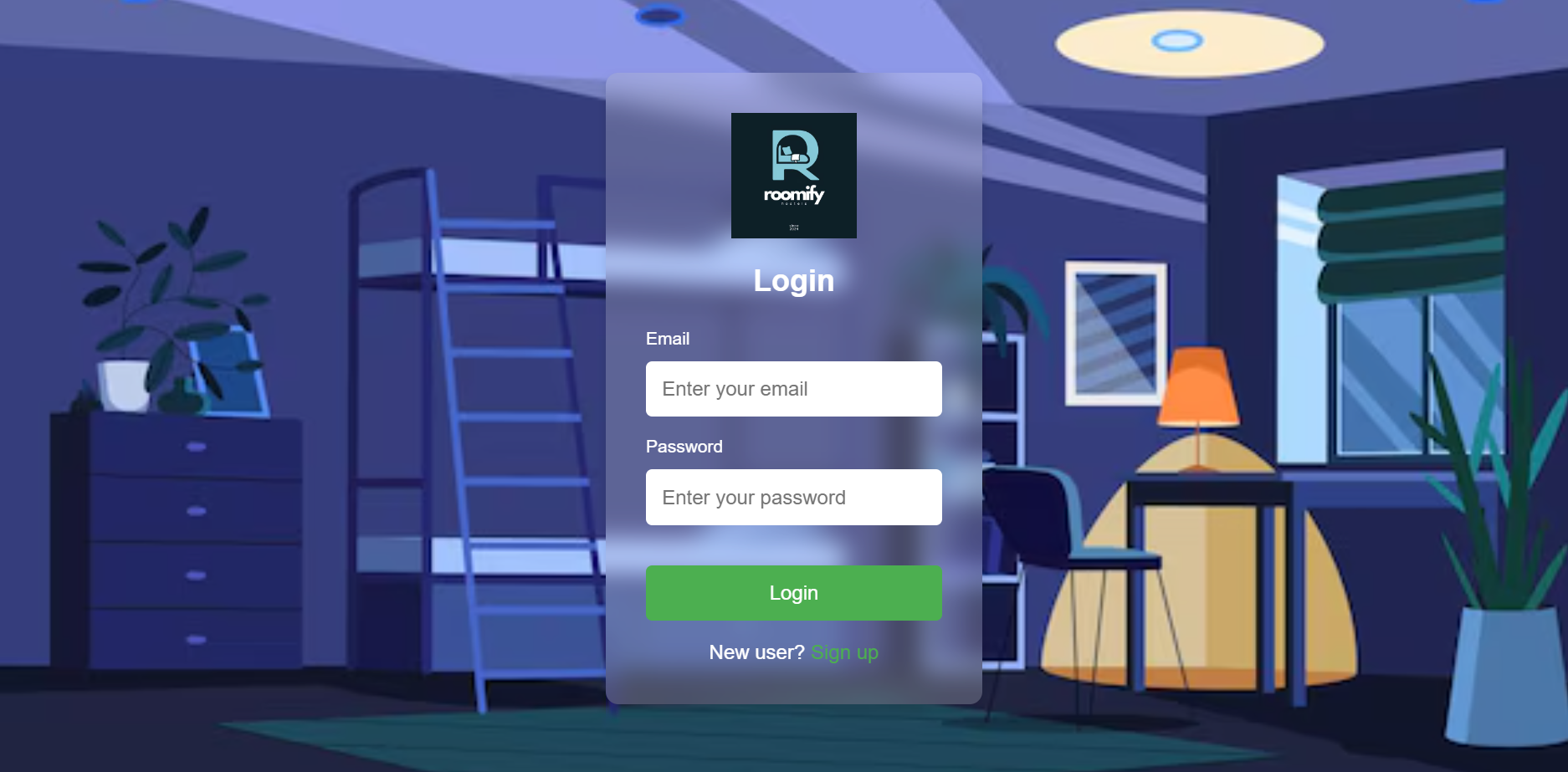
* **Unit Testing**: Individual modules (such as room allocation or fee management) were tested to ensure they perform as expected.
* **Integration Testing**: The interaction between front-end, back-end, and database was thoroughly tested.
* **User Acceptance Testing (UAT)**: Feedback was gathered from hostel administrators to ensure the system meets user expectations.

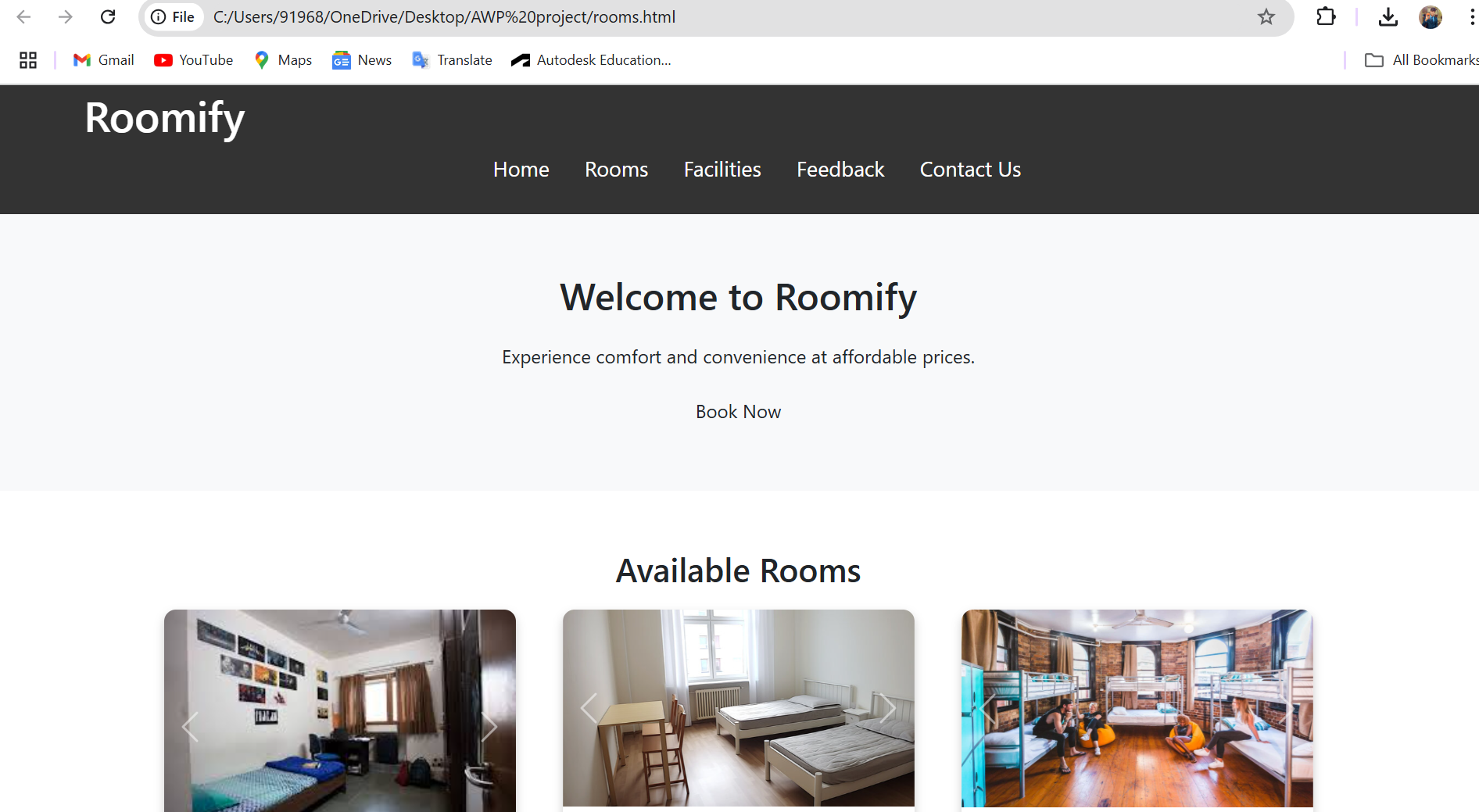
**6.2 Results**

* **Functional Testing**: All modules, including room allocation, fee management, and maintenance requests, functioned without any major issues.
* **Performance**: The system performed well under typical loads and provided responses within acceptable time frames.
* **Security**: Proper encryption was implemented for sensitive data, and access control was enforced.

Images:







**7. Challenges and Solutions**

**7.1 Challenges**

* **Room Allocation**: Ensuring that the room allocation was efficient and fair, especially when rooms were limited.
  + **Solution**: Implemented a priority-based allocation system based on student preferences and availability.
* **Security Concerns**: Securing sensitive student information such as contact details and fee payments.
  + **Solution**: Used hashing for password storage and encrypted sensitive data in transit.

**8. Conclusion**

The **Hostel Management System (Roomify)** has successfully automated key hostel management processes, reducing administrative workload and enhancing the student experience. The system provides a centralized platform for managing student records, room allocations, fee payments, and maintenance requests. By using modern technologies and efficient design, the system offers a scalable, secure, and user-friendly solution that can be further enhanced with additional features.

**9. Future Enhancements**

* **Mobile App Integration**: A mobile app version for students and admins to access the system on the go.
* **AI-Based Room Allocation**: Implementing an AI-based system that predicts the best room allocation based on student preferences and historical data.
* **Advanced Reporting**: Adding more customizable reporting features for deeper insights into hostel operations.

**10. References**

1. **React.js Documentation** – React Official Site
2. **Node.js Documentation** – Node.js Official Site
3. **MongoDB Documentation** – [MongoDB Official Site](https://www.mongodb.com/docs/)
4. **Express.js Documentation** – [Express Official Site](https://expressjs.com/)