Resource Booking Application

MIDHUNRAJA V A 7376221SE126 P-ID 5 SEAT NO: 285

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the requirements for the Resource Booking Application.

1.2 Scope

The Resource Booking Application is intended to facilitate the booking of resources such as classrooms, seminar halls, auditoriums, and labs in the college by students and faculty.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **UI**: User Interface

2. Overall Description

2.1 Product Perspective

The Resource Booking Application will be a standalone web application that interfaces with a database to manage bookings.

2.2 User Classes and Characteristics

- **Students**: Users who are students at the college.
- **Faculty**: Users who are faculty members at the college.

2.3 Operating Environment

The application will be accessed through a web browser on desktop and mobile devices.

2.4 Design and Implementation Constraints

The application will be developed using the MERN stack (MongoDB, Express.js, React.js, Node.js).

3. Specific Requirements

3.1 Functional Requirements

1. **User Registration**: Users should be able to register for an account.
2. **User Login**: Registered users should be able to log in to the application.
3. **Resource Booking**: Users should be able to view available resources and book them for specific time slots.
4. **Admin Panel**: Admin users should be able to approve or reject booking requests.
5. **Notifications**: Users should receive notifications via email or text message once their booking is approved or rejected.
6. **Time Slot Selection**: Users should be able to select time slots for booking, with options for both bulk time booking and minimum time booking.
7. **Availability Display**: Available time slots should be displayed in a user-friendly format.

3.2 Non-functional Requirements

1. **Performance**: The application should be able to handle multiple concurrent users without significant slowdowns.
2. **Security**: User data should be stored securely and protected against unauthorized access.
3. **Usability**: The user interface should be intuitive and easy to use.
4. **Reliability**: The application should be reliable and available whenever users need to access it.

3.3 User Interfaces

- **Login Page**: Allows users to log in to their accounts.
- **Resource Booking Page**: Allows users to view available resources and book them for specific time slots.
- **Admin Panel**: Allows admin users to approve or reject booking requests.

3.4 System Interfaces

- **Database**: The application will interface with a MongoDB database to store user and booking information.

4 Application Features

4.1 User-side Features

- **Sort** : The application will interface with a MongoDB database to store user and booking information.

- **Bulk Booking**: Allow users to book a venue for a duration of up to 30 days, selecting a specific start date and end date for the booking.

- **Multi-Venue Booking**: Enable users to book multiple venues simultaneously, selecting different venues for different time slots.

- **Editing and Modification**: Provide users with the ability to edit or modify their booking details, such as changing the date, time, or venue, if they made a mistake or need to make changes.

- **Booking Approval Status**: Display the status of the booking request, indicating whether it is pending, approved, or rejected. Users can track the progress of their booking requests.

- **Notification on Approval**: Send a notification to the user when their booking request is approved by the admin, confirming the successful reservation.

- **Booking Details and Reason**: Allow users to view their booked venues, along with the reason for the booking. This information helps both the user and the admin understand the purpose of the reservation.

4.2 Admin-Side Features

- **Bulk Booking Management**: Enable admins to handle bulk booking requests, reviewing and approving them for the requested duration and time slots.

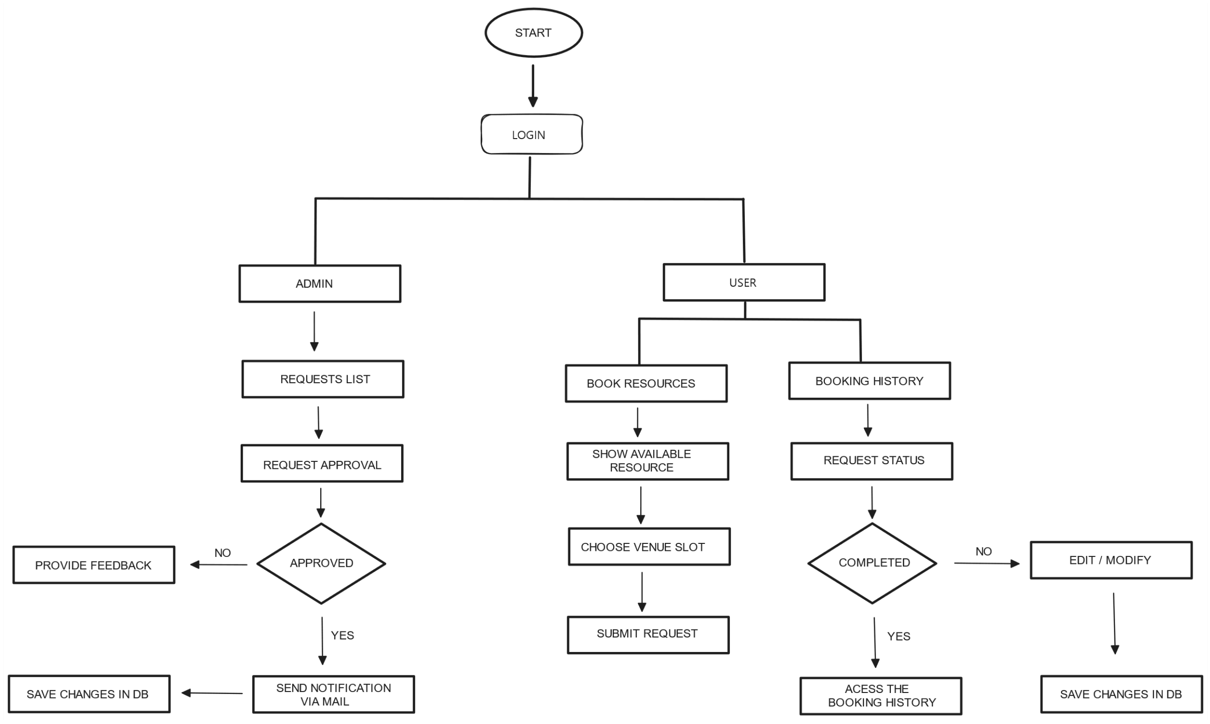
- **Multi-Venue Booking Management**: Provide admins with the ability to manage multiple venue bookings made by a user, ensuring there are no conflicts or scheduling issues.

- **Booking Modification Review**: Allow admins to review and approve or reject modification requests from users who want to edit or modify their booked venues.

- **Notification on Booking Approval**: Notify the user who made the booking when their request is approved, confirming the successful reservation.
- **Venue Availability Management**: After approval, automatically mark the booked venues as unavailable for other users during the booked time slots, preventing double bookings.

- **Booking Details and Reason Review**: Admins can view the booking details and the reason provided by the user, ensuring the booking aligns with college policies and regulations.

5.FLOW CHART



6. Technology Stack

6.1 Backend

- **Node.js**: Used for server-side logic and handling HTTP requests.
- **Express.js**: Framework for building the backend API and handling routing.

6.2 Frontend

- **React.js**: Used for building the user interface, providing a dynamic and responsive experience for users.

6.3 Database

- **MongoDB**: NoSQL database used for storing resource information, booking history, and user data.

6.4 Additional Libraries and Components

- **React Router**: For client-side routing in the React.js application, enabling navigation between different views.
- **Material-UI**: React component library for building a modern and visually appealing user interface.
- **Full Calendar**: JavaScript calendar library for displaying resource availability and booking dates.

7. Role of Each Component

7.1 Node.js and Express.js

- **Backend Logic**: Handle user authentication, booking requests, and interactions with the database.
- **API Development**: Create RESTful APIs for communication between the frontend and backend.

7.2 React.js

- **User Interface**: Build a dynamic and interactive user interface for booking resources and managing bookings.
- **State Management**: Manage application state and data flow using React's state and context APIs.

7.3 MongoDB

- **Database Storage**: Store resource information, booking history, and user data in a flexible and scalable manner.

7.4 React Router

- **Client-Side Routing**: Enable navigation between different views in the application without a page reload.

7.5 Material-UI

- **UI Components**: Use pre-built components for designing a modern and visually appealing user interface.

7.6 Full Calendar

- **Resource Availability**: Display resource availability and booking dates in a calendar format for easy visualization.