Project Title	**Online Bus Booking Application**
Technologies	MERN
Domain	Industry
Project Level	Medium

Table

Table of Contents

1. Problem Statement: 2
   1.1. What is a Bus Booking Application? 2
   1.2. Project Objective 3
   1.3. Scope of The Project 3
   1.4. Functional and Non-Functional Requirements: - 3
      1.4.1. Functional Requirements 3
      1.4.2. Non-Functional Requirements 4
   1.5. Modules List 4
2. Project Evaluation metrics: 4
   2.2. Database: 5
   2.3. API Details or User Interface: 5

2.4. Deployment:
..... 5 Deploy the application on your preferred service..... 5
2.5. Solutions Design:..... 5
2.6. System Architecture: 5
2.7. Optimization of solutions: 5
3. Submission requirements:
..... 5
3.1. High-level Document: 5
3.2. Low-level document: 5
3.3. Architecture: 6
3.4. Wireframe: 6
3.5. Project code: 6
3.6. Detail project report: 6
3.7. Project demo video: 6
3.8. The project LinkedIn a post: 6

1. Problem Statement:

Design a web application “Online Bus Booking Application” to allow users to check and book tickets online.

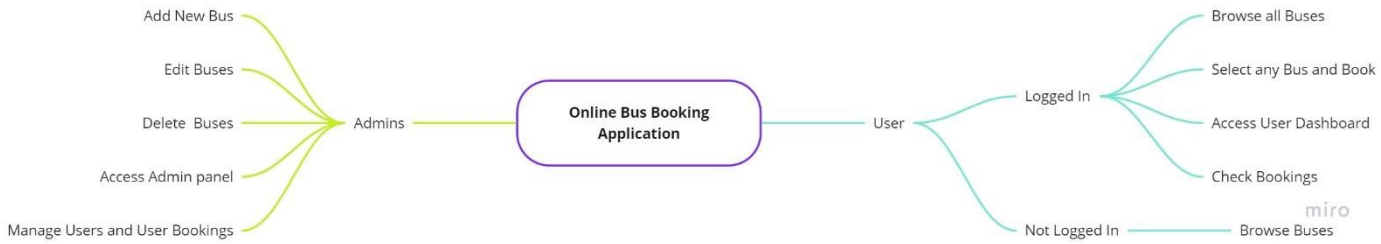
1.1. What is a Bus Booking Application?

A Bus Booking Application is an online tool that has all the bus data along with the bus departure time and bus arrival time. Any person can visit the site and check for the buses between any city and can book any available bus from the list of all the available buses. Once the booking is confirmed user will get a confirmation email with the trip details (Ex: from city, to city, departure and arrival time, date of journey, etc.,)

You might have used AbhiBus or RedBus to book Bus tickets.

Now that you understand what a Bus Booking Application is, let’s discuss some of the functionality of the “**Online Bus Booking Application**” that you will design.

1. To create a web-based system that will enable users to book any bus available from the list along with their requested date and time.
2. Users must be able to select the bus between their requested cities and only the buses going in that route should be visible.
3. Users need to register themselves first before making any successful bus booking.
4. There has to be a user dashboard which should also show users their booking details.
5. The admins can add/edit/delete any of the buses available for booking.
6. All non-logged-in users must be able to browse the buses available for booking and see the relevant info for any specific bus.
7. Admins should be able to either accept the booking or reject a booking (with reason) 8. The project should be very easy to use with a clean UI.



Low-Level system flow

1.2. Project Objective

- To build a system which minimizes the manual task of going to a bus station to enquire about the bus availability and as well as save time.
- To make the users life easy by giving them a platform where they can book tickets anytime and anywhere.
- The program functions as an always-open office.

1.3. Scope of The Project

1. The online bus booking application maintains the data of all the buses and the availability of the buses and the information of how many seats are booked and how many are remaining in a nice dashboard.

2. To give users the power to get the info on any bus from the convenience of their homes instead of them visiting a bus station to enquire about any bus.

3. To let users fill in their details themselves and book the bus and eliminate the process where the users fill a booking receipt and somebody then sits and re-enters the user data into the system.

1.4. Functional and Non-Functional Requirements: -

1.4.1. Functional Requirements

Admin:

- Can manage all the data.
- Can approve Buses.
- Can approve or reject bus booking and/or departure (Cancel any Bus).

User:

- Users can browse the available buses and can even book the bus after registration.
- Users can view the details of all the available buses.

UI:

- There should be a dashboard for admins, and registered users.
- Frontend should be clean and minimal which shows the available buses along with their info.
- Register/Login forms and a search box for easier navigation.

1.4.2. Non-Functional Requirements

1. **Security:** The system should have a certain level of security such that not anybody can have access to sensitive information and the passwords should be properly encrypted.
2. **Robustness:** If the user's system crashes, a backup of the user data must be stored on remote database servers to enable recovery.
3. **Performance:** The application must be lightweight and the UI should be fast and responsive.
4. **Authentication Requirements:** There should be an authentication system in place which checks the credentials each time a user logs in.

1.5. Modules List

Admin:

- Login
- Add Buses o From City o To City o Departure info
  - o Bus Name and other necessary info
- Update/Delete Buses
- Cancel buses (If required) **User:**
- Login/Register
- View Buses
- View Bookings o Book time
  - o View status of request (Approved/Rejected)
- Book a Bus o City (From – To), Contact details, Date and Time of departure
- Cancel their own Booking

- My Profile
- Logout

2. Project Evaluation metrics:

2.1. Code:

- You are supposed to write code in a modular fashion
- Safe: It can be used without causing harm.
- Testable: It can be tested at the code level.
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment (operating system).
- You have to maintain your code on GitHub.
- You have to keep your GitHub repo public so that anyone can check your code.
- Proper readme file you have to maintain for any project development.
- You should include the basic workflow and execution of the entire project in the readme file on GitHub.
- Follow the coding standards.

2.2. Database:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

2.3. API Details or User Interface:

You have to expose your complete solution as an API or try to create a user interface for your model testing. Anything will be fine for us.

2.4. Deployment:

Deploy the application on your preferred service.

2.5. Solutions Design:

You have to submit complete solution design strategies in High-level Document (HLD), Lowlevel Document (LLD), and Wireframe documents**.**

2.6. System Architecture:

You must submit a system architecture design in your wireframe and architecture documents.

2.7. Optimization of solutions:

Try to optimize your solution on the code level, and architecture level, and mention all of these things in your final submission.

Mention your test cases for your project.

3. Submission requirements:

3.1. High-level Document:

You have to create a high-level document design for your project. You can reference the HLD form below the link.

Sample link: HLD Document Link

3.2. Low-level document:

You have to create a Low-level document design for your project; you can refer to the LLD from the link below.

Sample link: LLD Document Link

3.3. Architecture:

You have to create an Architecture document design for your project; you can refer to the Architecture from the link below.

Sample link: Architecture sample link

3.4. Wireframe:

You have to create a Wireframe document design for your project; refer to the Wireframe from the link below.

Demo link: Wireframe Document Link

3.5. Project code:

You have to submit your code to the GitHub repo in your dashboard when the final submission of your project.

Demo link: Project code sample link

3.6. Detail project report:

You have to create a detailed project report and submit that document as per the given sample.

Demo link: DPR sample link

3.7. Project demo video:

You have to record a project demo video for at least 5 Minutes and submit that link as per the given demo.

3.8. The project LinkedIn a post:

You have to post your project details on LinkedIn and submit that post link in your dashboard in your respective field.