TRAVELS AND TOURISM

**A MINI PROJECT REPORT**

**Submitted by**

**ALFRED SAM D 230701026**

**ARUN MC 230701036**

**In partial fulfillment for the award of the degree of**

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE**

**RAJALAKSHMI ENGINEERING COLLEGE**

**(AUTONOMOUS)  
Thandalam,**

**Chennai - 602105**

**2023-24**

**BONAFIDE CERTIFICATE**

**This is to certify that the project report titled “TRAVEL AND TOURISM MANAGEMENT SYSTEM” is the bonafide work of ARUN MC and ALFRED SAM D, who carried out the project work under my supervision.**

**Submitted for the Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_.**

**SIGNATURE**

**Dr.DHARNI DEVI,**

**ASSISTANT PROFESSOR ,**

**Computer Science and Engineering,**

**Rajalakshmi Engineering College,**

**(Autonomous),**

**Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**Dr.Dharni Devi**

**ABSTRACT**

The Travel and Tourism Management System is an interactive application built for managing bus services. Using this system, users can perform operations such as viewing, updating, inserting, and deleting bus information.

**Key functionalities include:**

* **Viewing all buses with detailed information.**
* **Finding a single bus based on user criteria.**
* **Finding multiple buses using specified conditions.**
* **Updating existing bus information.**
* **Adding new buses to the system.**
* **Deleting buses based on specified conditions.**

The backend is implemented using Spring Boot, and the application leverages MongoDB as the database for managing data efficiently. The frontend employs React to deliver a dynamic and user-friendly interface.

**TABLE OF CONTENTS**

1. **Introduction  
   1.1 Introduction  
   1.2 Objectives  
   1.3 Modules**
2. **Survey of Technologies  
   2.1 Software Description  
   2.2 Languages  
   - 2.2.1 Spring Boot  
   - 2.2.2 React  
   - 2.2.3 MongoDB**
3. **Requirements and Analysis  
   3.1 Requirement Specification  
   3.2 Hardware and Software Requirements  
   3.3 Architecture Diagram  
   3.4 ER Diagram  
   3.5 Data Flow Diagram**
4. **Program Code**
5. **Results and Discussion**
6. **Conclusion**

**CHAPTER 1: INTRODUCTION**

The Travel and Tourism Management System focuses on managing bus services efficiently. The application is designed to handle tasks such as issuing, updating, and deleting bus records with a simple and interactive interface.

**Key features include:**

* **A graphical user interface for easy navigation and operation.**
* **Operations to manage buses efficiently, including viewing, updating, and deleting records.**
* **Use of a robust backend for handling business logic and database operations.**

**1.2 Objectives**

**The primary objectives of this project are:**

1. To develop a user-friendly application for managing bus services effectively.
2. To provide efficient mechanisms for CRUD (Create, Read, Update, Delete) operations on bus records.
3. To implement robust backend logic using Spring Boot to handle complex queries and updates.
4. To ensure scalable and secure storage using MongoDB.
5. To deliver an intuitive and responsive front-end experience with React.

**1.3 Modules**

**The application consists of the following modules:**

1. View All Buses: Displays a comprehensive list of all buses in the system with their details.
2. Find Single Bus: Allows users to search for a single bus based on specific criteria, such as the type of bus.
3. Find Multiple Buses: Facilitates fetching multiple bus records that satisfy user-defined conditions.
4. Update Bus Information: Enables the modification of existing bus details, such as seat availability or bus type.
5. Insert New Bus: Provides functionality to add a new bus to the system with required attributes.
6. Delete Bus: Removes bus records from the system based on specified criteria.

**CHAPTER 2: SURVEY OF TECHNOLOGIES**

**2.1 Software Description**

The Travel and Tourism Management System leverages the following technologies to ensure robust, scalable, and efficient performance:

1. **Spring Boot:**Spring Boot is a Java-based framework designed to simplify the development of backend applications. Key features include:
   * Dependency Injection: Automatically manages object dependencies to reduce boilerplate code.
   * REST API Support: Facilitates easy creation of RESTful web services for seamless interaction between the frontend and backend.
   * MongoDB Integration: Provides built-in support for NoSQL databases like MongoDB, enabling easy data persistence and querying.
   * Microservices Architecture: Supports modular design for scalable and maintainable backend development.
2. **React:**React is a widely-used JavaScript library for building interactive user interfaces. Key features include:
   * Component-Based Architecture: Breaks the UI into reusable components, ensuring easier management and maintenance.
   * Virtual DOM: Improves application performance by updating only the necessary parts of the DOM.
   * State Management: Provides hooks and libraries like Redux for maintaining application state effectively.
   * Responsive Design: Ensures compatibility across devices with varying screen sizes.
3. **MongoDB:**MongoDB is a NoSQL database that uses a JSON-like document structure. Key features include:
   * Schema Flexibility: Allows dynamic schema, making it easy to handle evolving data requirements.
   * High Performance: Supports efficient data storage and retrieval with indexing and sharding capabilities.
   * Integration with Spring Boot: Simplifies database operations like CRUD with Spring Data MongoDB.
   * Scalability: Easily accommodates growing application needs with horizontal scaling.

**2.2 Languages**

The system uses the following programming languages and formats:

**2.2.1 Java**

* Java serves as the primary language for backend development using Spring Boot.
* It offers object-oriented programming principles, ensuring clean, modular, and reusable code.
* Java supports multithreading, security features, and a rich ecosystem of libraries, making it ideal for enterprise applications.

**2.2.2 JavaScript**

* JavaScript is used to develop the frontend with React, enabling interactive and dynamic web pages.
* It supports asynchronous programming and integrates seamlessly with REST APIs for real-time data updates.
* As the backbone of the React framework, JavaScript facilitates creating responsive UI components.

**2.2.3 JSON**

* JSON (JavaScript Object Notation) is used for data exchange between the backend and frontend.
* It provides a lightweight, human-readable format to structure data, making it easy to transmit over APIs.
* MongoDB’s JSON-like document storage aligns perfectly with JSON, simplifying data handling and querying.

**CHAPTER 3: REQUIREMENTS AND ANALYSIS**

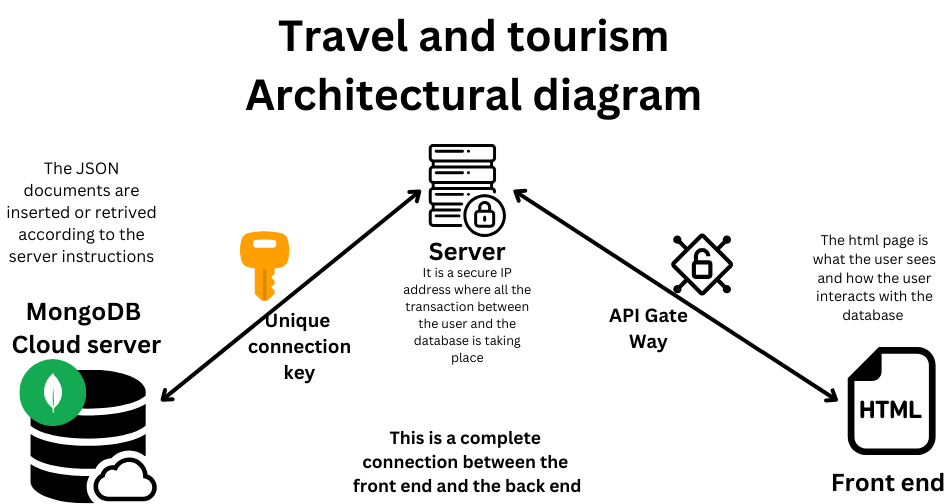
**3.1 Requirement Specification**

* **Functional Requirements:**
  + View all buses.
  + Search for a single or multiple buses based on user-specified criteria.
  + Insert new bus records.
  + Update existing bus details.
  + Delete bus records.
* **Non-Functional Requirements:**
  + The application must respond within 2 seconds for every operation.
  + Data integrity must be maintained in all database transactions.
  + Ensure user-friendly interaction with a graphical interface.

**3.2 Hardware and Software Requirements**

* **Hardware:**
  + Processor: Intel Core i5 or equivalent.
  + RAM: 8GB minimum.
  + Storage: 250GB minimum.
* **Software:**
  + Operating System: Windows 10 or Linux.
  + Backend Framework: Spring Boot.
  + Frontend Library: React.
  + Database: MongoDB.
  + IDE: IntelliJ IDEA or Eclipse for backend, VS Code for frontend.

### **3.3 Architecture Diagram**



### **3.4 Collections (Entities)**

1. **Train Booking**
   * Attributes: \_id, trainName, userId, bookingDate, source, destination, seatNumber, status
   * Relationships: Linked to Train Receipt by bookingId and to Users by userId
2. **Train Receipt**
   * Attributes: \_id, bookingId, amountPaid, paymentDate, paymentMethod
   * Relationships: bookingId references Train Booking.\_id
3. **Hotel Booking**
   * Attributes: \_id, hotelName, userId, bookingDate, checkInDate, checkOutDate, roomNumber, status
   * Relationships: Linked to Hotel Receipt by bookingId and to Users by userId
4. **Hotel Receipt**
   * Attributes: \_id, bookingId, amountPaid, paymentDate, paymentMethod
   * Relationships: bookingId references Hotel Booking.\_id
5. **Tourist Spot Booking**
   * Attributes: \_id, spotName, userId, bookingDate, visitDate, numberOfTickets, status
   * Relationships: Linked to Tourist Spot Receipt by bookingId and to Users by userId
6. **Tourist Spot Receipt**
   * Attributes: \_id, bookingId, amountPaid, paymentDate, paymentMethod
   * Relationships: bookingId references Tourist Spot Booking.\_id
7. **Bus Booking**
   * Attributes: \_id, busName, userId, bookingDate, source, destination, seatNumber, status
   * Relationships: Linked to Bus Receipt by bookingId and to Users by userId
8. **Bus Receipt**
   * Attributes: \_id, bookingId, amountPaid, paymentDate, paymentMethod
   * Relationships: bookingId references Bus Booking.\_id

### **Central Relationships**

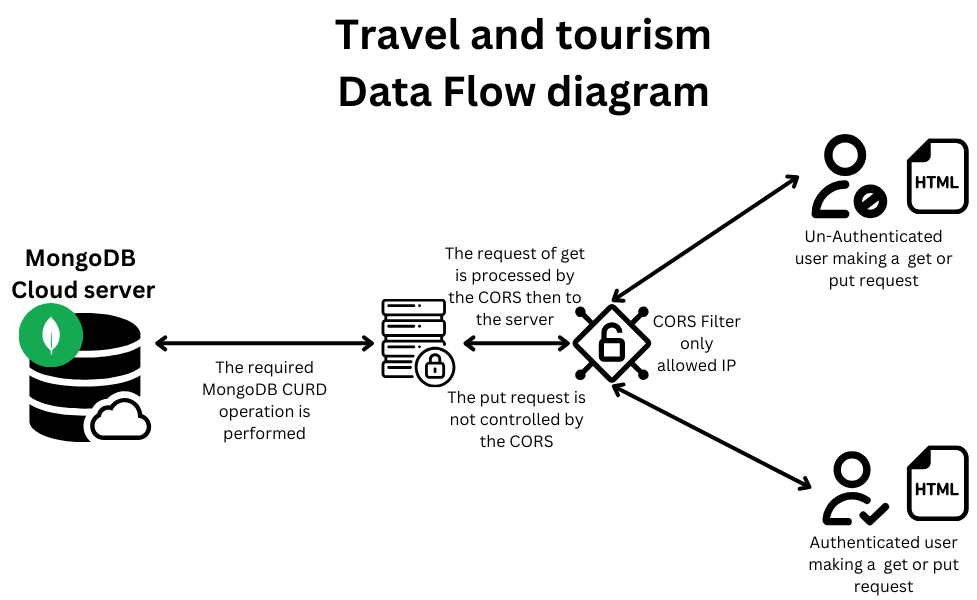
* All **Booking Collections** (Train, Hotel, Tourist Spot, Bus) are linked to a common Users collection via userId.

### **Diagram Description**

The ER diagram would show:

1. **Users** as the central entity with 1-to-many relationships to each booking collection.
2. Each **Booking Collection** (Train, Hotel, Tourist Spot, Bus) connected 1-to-1 with its respective receipt collection.

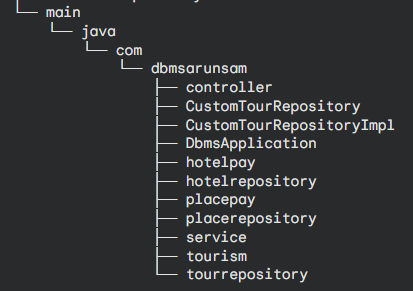
**3.5 Data Flow Diagram**



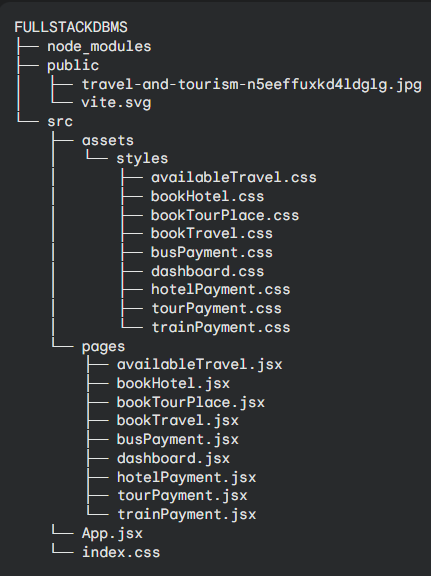
**CHAPTER 4: PROGRAM CODE**

The program code is very large and segregated into various java files

The simple representation for the structure is:



Other than this there is front end code in react with the following structure



**Thus it is recommended to verify this git hub link for the code implementation  
for backend:**

**https://github.com/D-AlfredSam/DataBase-Management-System/tree/main/DBMS**

**for frontend:**

**https://github.com/D-AlfredSam/DataBase-Management-System/tree/main/src**

**CHAPTER 5: RESULTS AND DISCUSSION**

**Results**

The Travel and Tourism Management System was tested successfully with the following results:

1. **View All Buses:** Displays a list of all buses in the system with their details.
2. **Find Single Bus:** Fetches the details of a bus based on specified criteria.
3. **Find Multiple Buses:** Retrieves multiple buses satisfying a set of conditions.
4. **Update Bus Information:** Updates existing records in the database.
5. **Insert New Bus:** Adds a new bus to the system with specified attributes.
6. **Delete Bus:** Deletes bus records from the database.

**Discussion**

The system leverages Spring Boot for a reliable and scalable backend while MongoDB ensures efficient data storage and retrieval. The graphical interface, created using Swing, enhances user interaction and usability. Challenges encountered during development, such as optimizing database queries and managing UI responsiveness, were resolved effectively by leveraging modern tools and frameworks.

**CHAPTER 6: CONCLUSION**

The Travel and Tourism Management System successfully fulfills its objective of providing a robust and user-friendly solution for managing bus services. By leveraging advanced technologies such as Spring Boot, React, and MongoDB, the system ensures efficient CRUD operations, scalability, and an intuitive user experience.

The application enables seamless management of bus records, including viewing, updating, inserting, and deleting information, all within a responsive and interactive interface. Its modular architecture and integration of modern frameworks make it adaptable to future enhancements, such as additional features or extended functionalities.

Overall, the project demonstrates a practical implementation of a full-stack development approach and provides a scalable foundation for addressing the needs of the travel and tourism industry.