### SRI KRISHNA COLLEGE OF TECHNOLOGY

**An Autonomous Institution | Accredited by NAAC with 'A' Grade Affiliated to Anna University | Approved by AICTE**

### KOVAIPUDUR, COIMBATORE 641042

**TRAVELPRISM**

### A PROJECT REPORT

***Submitted by***

### KISHORE S 727821TUAD027

### *in partial fulfilment for the award of the degree*

### *Of*

### BACHELOR OF TECHNOLOGY

**IN**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**JUNE 2023**

# CERTIFICATE

## BONAFIDE CERTIFICATE

Certified that this project report **“TRAVELPRISM”** is the Bonafide work of **KISHORE S** who carried out the project work under my supervision.

SIGNATURE SIGNATURE

**Mr.Mano Palani Raja A** **Dr. Maheswaran C P**

**SUPERVISOR HEAD OF THE DEPARTMENT**

Skill Development Engineer Associate Professor

Iamneo. Department of AI&DS Sri Krishna College of

Technology,

Kovaipudur,

Coimbatore-42.

Certified that the candidates were examined by us in the Project Work viva-voce examination held on ............................... at Sri Krishna College of Technology, Coimbatore -641 042.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

### ACKNOWLEDGEMENT

First and foremost, we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We are grateful to our beloved Dean-Academics **Dr. Manju P &** Dean-Administration **Dr R Ramesh Kumar** for their tireless and relentless support.

We extend our sincere thanks to our Head of the Department

**Dr. Maheswaran CP** for his encouragement and inspiration.

We are greatly indebted to our Industry Mentor **Mr.Mano Palani Raja A** & Project guide **Ms. Nivedhitha**, Assistant Professor, Department of CSE for their valuable guidance and suggestions in all aspects that aided us to ameliorate our skills.

We are thankful to all those who have directly and indirectly extended their help to us in completing this project work successfully.

# ABSTRACT

## ABSTRACT

TravelPrism is a comprehensive solution for efficient travel management. It provides seamless tracking and updating of travel details, empowering users with accurate and timely information. The application offers functionalities such as itinerary management, expense tracking, and travel information updates. Users can make informed travel decisions based on real-time data, ensuring smooth and well-organized trips. The user-friendly interface simplifies navigation and facilitates quick access to relevant travel information. By streamlining travel management processes, the application enhances productivity and improves overall travel experiences, resulting in efficient travel planning. The review Management Full Stack Application utilizes modern technologies and frameworks such as JavaScript, HTML, CSS for the front-end, and a robust backend framework like Node.js. Database management systems like MySQL is used to securely store and manage data. RESTful APIs ensure smooth communication between the front-end and back-end components, enabling seamless data synchronization and enhancing overall application performance.

# TABLE OF CONTENT

## TABLE OF CONTENTS

**CHAPTER TITLE PAG E NO**

[ACKNOWLEDGEMENT III](#_TOC_250014)

[ABSTRACT IV](#_TOC_250013)

[LIST OF FIGURES VII](#_TOC_250012)

LIST OF ABBREVATIONS VIII

1. INTRODUCTION 1
   1. [OVERVIEW 1](#_TOC_250011)
   2. OBJECTIVES 3
2. LITERATURE SURVEY 4
3. SYSTEM SPECIFICATION 6
   1. [VS CODE 6](#_TOC_250010)
   2. [REACT JS 7](#_TOC_250009)
   3. [ROUTERS IN REACT 7](#_TOC_250008)
   4. [HTML 7](#_TOC_250007)
   5. [CSS 8](#_TOC_250006)
   6. [REST API 8](#_TOC_250005)
4. PROPOSED SYSTEM AND ADVANTAGES 10
   1. [PROPOSED SYSTEM 10](#_TOC_250004)
   2. [ADVANTAGES 11](#_TOC_250003)
   3. [FLOW CHART 12](#_TOC_250002)
5. METHODOLOGIES 13
   1. METHODOLOGIES 13
   2. [TOOLS TO USE 14](#_TOC_250001)
6. IMPLEMENTATIONS 16
   1. [LOGIN PAGE 17](#_TOC_250000)
   2. DETAILS PAGE 17
   3. ADD PAGE 18
   4. UPDATE PAGE 19
7. CONCLUSION AND FUTURE WORKS 20

APPENDIX 1 SAMPLE CODE 21

REFERENCES 30

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 4.1 | Flow diagram | 12 |
| 6.1 | Login page | 17 |
| 6.2 | Traveller list | 18 |
| 6.3 | Add new details | 18 |
| 6.4 | Update Traveller details | 19 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATION** | **ACRONYMS** |
| CURD | Create, Update, Read, Delete |
| REST | Representational State Transfer |
| JWT | JSON Web Token |
| API | Application Programming Interface |
| UI | User Interface |
| UX | User Experience |

## CHAPTER 1 INTRODUCTION

TravelPrism leverages React.js and Java Spring Boot to create a powerful full-stack application. React.js is renowned for its ability to build dynamic and interactive user interfaces, while Java Spring Boot provides a robust framework for backend development. The application encompasses essential components such as sign-in, sign-up, and a landing page featuring a data table with comprehensive CRUD operations. Although the current iteration does not incorporate advanced features like validation, authorization, or JWT authentication, it sets a solid foundation for future enhancements. By integrating these additional functionalities, the application can further enhance security and user access control.

The combination of React.js and Java Spring Boot showcases their potential in delivering efficient, scalable, and feature-rich full-stack applications. With its seamless frontend-backend integration, the project demonstrates the significance of these frameworks in meeting the evolving demands of the technology landscape. Developers can leverage React.js and Java Spring Boot to create intuitive, high-performance applications that streamline data management and elevate user experiences.

## OVERVIEW

The full-stack application developed in this project combines the power of React.js and Java Spring Boot to deliver a comprehensive and efficient user experience. The application consists of a sign-in page, sign-up page, and a landing page displaying a table of records from a database. Users can perform CRUD operations on the table data directly through the frontend.

React.js, a popular JavaScript library for building user interfaces, is utilized

for the frontend development. Its component-based architecture and efficient state management allow for the creation of dynamic and interactive UI components. React Router facilitates seamless client-side routing and navigation within the application.

On the backend, Java Spring Boot serves as the framework for building RESTful APIs and handling backend logic. Spring Boot's extensive features, including dependency injection and database integration, streamline the development process and ensure scalability and reliability.

The sign-in page enables users to authenticate by providing their credentials, while the sign-up page allows new users to register by providing their information. The frontend captures the user input, which is then securely processed by the backend for storage in the database.

The landing page is the central interface, displaying a table populated with records fetched from the database. React.js components efficiently render the data, allowing users to perform CRUD operations seamlessly. Users can create new records, update existing ones, and delete records directly from the table interface.

Through RESTful API endpoints, the frontend communicates with the backend to perform the necessary operations on the database. The application prioritizes a user-centric approach, aiming to provide an intuitive and efficient data management experience.

In conclusion, the full-stack application developed using React.js and Java Spring Boot offers a seamless integration of frontend and backend functionalities. With its sign-in, sign-up, and landing pages, the application enables users to perform CRUD operations on a table of records. Future iterations can expand on its capabilities by incorporating additional features such as validation and authorization, further enhancing the application's security and functionality.

## OBJECTIVE

The objective of the application is to provide efficient management of traveller details for planned tours. The objective of developing the TravelPrism with React.js and Java Spring Boot is to address the challenges and shortcomings of existing travel management systems and create a comprehensive, user-friendly, and efficient solution.User-Friendly Interface is a design and develop a visually appealing and intuitive user interface using React.js that enhances the overall user experience. Focus on creating a seamless and responsive interface that allows users to easily navigate, search for destinations, view accommodation details, and manage their travel itineraries.

## CHAPTER 2

### LITERATURE SURVEY

1. Book: "Spring Boot in Action" by Craig Walls

This book provides a comprehensive guide to developing Java applications with Spring Boot. It covers the fundamentals of Spring Boot, including configuration, data access, testing, and deployment. The book also explores the integration of Spring Boot with RESTful APIs, showcasing best practices and design principles.

2. Book: "RESTful Java with JAX-RS 2.0" by Bill Burke

This book focuses on building RESTful web services using the Java API for RESTful Web Services (JAX-RS) specification. It covers the basics of REST, JAX-RS concepts and annotations, handling data, security, and advanced topics such as hypermedia and asynchronous programming. The book provides practical examples and insights into building REST APIs with Java.

3. Documentation: Spring Boot Reference Guide

The official Spring Boot Reference Guide is a valuable resource for understanding the features and capabilities of Spring Boot. It covers topics such as dependency management, auto-configuration, testing, security, and deployment options. The guide also provides detailed information on building RESTful APIs with Spring Boot and integrating with other frameworks.

4."Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications" by Felipe Gutierrez

This book provides a comprehensive guide to building

enterprise applications and microservices using Spring Boot. It covers topics

such as RESTful APIs, security, database integration, testing, and deployment.

It also includes practical examples and best practices for developing full-stack applications.

## CHAPTER 3 SYSTEM SPECIFICATION

In this chapter, we are going to see about system specification of the full- stack application.

The application is designed to provide efficient for the traveller details. It enables users to effectively view, update, and monitor traveller details, ensuring accurate inventory records and travel optimizing decisions. By leveraging user registration and authentication, users can securely access the system and manage their details with confidence. The app facilitates stock management through features such as product listing, search, and real-time stock updates. Users can perform essential CRUD operations on product records Users can register and create secure accounts to access the TravelPrism. User authentication ensures that only authorized individuals can manage travel arrangements and associated data. The user-friendly interface enhances the overall user experience.

The following list includes the various modules that were used in this topic:

* VS Code
* React js
* Router in react
* HTML
* CSS
* REST API

## VS CODE

Visual Studio Code is a source code editor developed by Microsoft for

Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customisable, so users can change the editor's theme, keyboard shortcuts, and preferences.

## REACT JS

React is a JavaScript library created by Facebook for building user interfaces. It is a component-based, declarative, and highly efficient library that is used to develop interactive UIs (user interfaces) for single page web applications. React uses a virtual DOM (Document Object Model) that makes it faster and easier to manipulate the DOM elements. It also provides declarative components that allow developers to write code that is easy to read and maintain. React also offers an extensive library of tools and components that make it easier to develop complex user interfaces.

## ROUTERS IN REACT

Routers are important components in React applications. They provide the ability to navigate between different views or components of the application. React Router is the most popular library to handle routing in React applications. It provides the ability to define routes, set up links, and render components based on the current route. It also provides features like data fetching, code-splitting, and server-side rendering

## HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting

languages such as JavaScript.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

## CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of content and presentation, including layout, colours and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

## REST API

A REST API (Representational State Transfer Application Programming Interface) is an architectural style for designing networked applications. It provides a set of rules and constraints for building web services that can be accessed over the internet. REST APIs have become widely adopted due to their simplicity, scalability, and compatibility with various platforms and programming languages. They are commonly used in web and mobile applications to enable communication between client-side applications and server-side resources. Developers can interact with REST APIs by sending HTTP

requests (GET, POST, PUT, DELETE) to specific endpoints, providing necessary parameters, and handling the responses returned by the server.

REST APIs use the HTTP protocol as the underlying communication protocol, making use of its methods, headers, and status codes. The API endpoints are typically represented by URLs, and data is transferred in a structured format, often JSON or XML, in the request and response bodies.

Developers can interact with REST APIs by sending HTTP requests to the API endpoints using libraries or tools that support HTTP communication, such as cURL, Postman, or programming language-specific libraries.

REST APIs have become a popular choice for building web services due to their simplicity, scalability, and compatibility with various programming languages and platforms. They are widely used in web and mobile applications to retrieve and manipulate data from servers.

## CHAPTER 4

**PROPOSED SYSTEM AND ADVANTAGES**

In this chapter, we will learn about the proposed system and advantages of the full-stack application.

The development of any project comes with its fair share of challenges. When building a project, it is crucial to anticipate potential problems that may arise and identify effective solutions to overcome them. In the context of this particular project, several key areas require attention to ensure successful implementation and operation. This includes addressing scalability issues to accommodate growing user demands, implementing robust security measures to protect sensitive data, and ensuring cross-platform compatibility for a seamless user experience. By proactively identifying these potential challenges and implementing the suggested solutions, the project can be developed with a strong foundation, enhancing its scalability, security, and usability. This introduction sets the stage for exploring the specific problems and solutions within the three identified headings, highlighting the importance of addressing these critical areas during the development process.

## PROPOSED SYSTEM:

The proposed method for the project encompasses several key elements to achieve its objectives. It involves utilizing React.js for frontend development, creating interactive user interfaces, and ensuring a seamless user experience. For the backend, Java Spring Boot is employed to handle server-side functionalities, data storage, and API development. The integration of a suitable database management system enables efficient storage and retrieval of traveller records. Emphasis is placed on thorough testing and quality assurance processes to ensure reliability and functionality. Additionally, deployment on a suitable hosting

platform, configuration of server environments, and implementation of monitoring and maintenance procedures are considered for seamless application operation. The proposed method provides a structured approach to the project's development, ensuring the successful implementation of frontend and backend components, reliable data management, and adherence to quality standards. By following this method, the project can achieve its desired outcomes effectively and deliver a robust TravelPrism application.

## ADVANTAGES:

Simplicity: The project provides a straightforward and easy-to-understand example of a full-stack application using React.js and Java Spring Boot. Its simplicity makes it accessible to developers who are new to these technologies or looking for a basic implementation.

Integration Demonstration: By showcasing the integration between React.js and Java Spring Boot, the project helps developers understand how the frontend and backend components work together to create a cohesive application. It provides insights into handling data flow, API communication, and rendering components.

Quick Setup and Development: The project's simplicity allows for a quick setup and development process. Developers can focus on learning and implementing the core concepts without getting overwhelmed by complex features or extensive configurations.

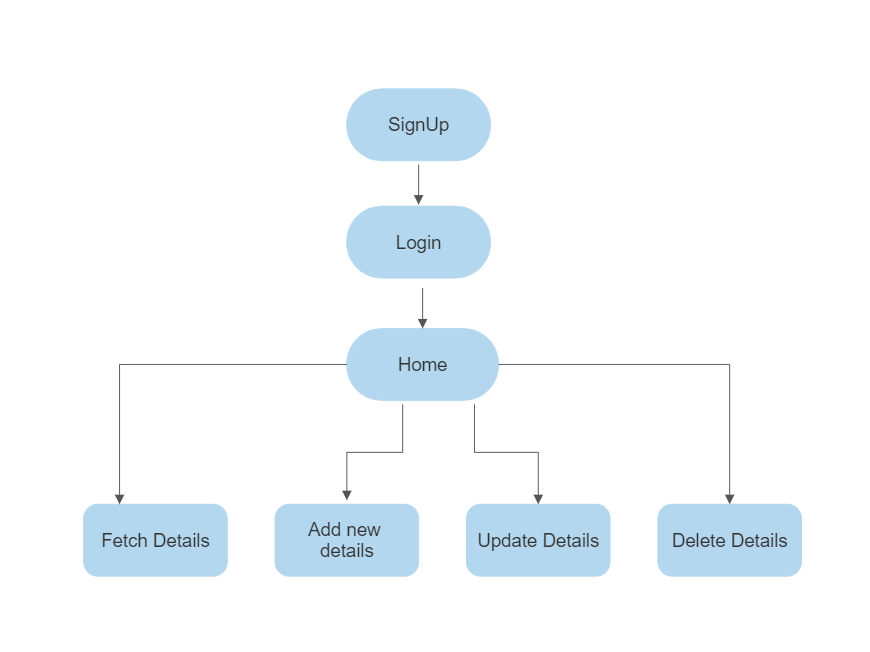
Learning Resource: The project serves as a valuable learning resource for developers interested in React.js and Java Spring Boot. By exploring the codebase, developers can gain insights into best practices, project structure, and common patterns used in full-stack development.

Collaboration: The project's simplicity facilitates collaboration among developers. It provides a common ground for team members to discuss, share knowledge, and contribute to the application's development, fostering a collaborative and learning-oriented environment.

Overall, our project's advantages lie in its simplicity, integration demonstration, quick setup, customization potential, learning resource value, and facilitation of collaboration. It serves as a practical starting point for developers to gain hands-on experience and explore the capabilities of React.js and Java Spring Boot in a full-stack context.

## FLOW CHART

The flow of the website depicted below:



**Figure.No4.1 Flow diagram**

## CHAPTER 5 METHODOLOGIES

In this chapter, we are going to see about methodologies used to build the full-stack application.

Methodologies and implementation play a crucial role in the project's success. The chosen methodologies provide a systematic framework for project management and development, ensuring adherence to best practices. Implementation involves executing the development tasks, integrating components, and deploying the application. This phase includes frontend and backend development, database integration, API creation, testing, deployment, and maintenance. By following the established methodologies and effectively implementing the project plan, the development team can ensure a structured and organized approach, leading to efficient collaboration, timely delivery, and the achievement of project objectives.

## PROJECT APPROACH:

The Agile methodology offers an ideal approach for your project, combining flexibility and efficient progress. With a focus on iterative development and regular feedback, Agile enables you to tailor the methodology to your specific needs. By identifying user stories and creating a backlog, you can prioritize features and functionalities that align with your project's objectives. Breaking down the backlog into manageable tasks within sprints allows you to maintain a clear roadmap and work in a structured manner.

As you proceed with development, you have the freedom to manage your time and resources effectively. Regular testing during and after each sprint ensures the quality and functionality of your application. Seeking feedback from users or stakeholders at the end of each sprint provides valuable insights for

adaptation and refinement. This iterative process allows you to continually improve and adapt your project based on real-world input.

Embracing the practice of continuous integration and automating deployment streamlines the delivery of new features and bug fixes. This ensures a seamless and efficient development workflow, enabling you to focus on enhancing your application.

By applying Agile methodologies tailored to your individual development process, you can maintain a structured and adaptable approach. This empowers you to incorporate user feedback, stay focused on your project's objectives, and deliver a high-quality application. The flexibility of Agile allows you to strike a balance between efficient progress and accommodating changes, ultimately leading to the successful completion of your project as a single developer.

## TOOLS TO USE:

### Frontend tools:

* 1. Code Editor:
     1. Visual Studio Code
  2. Package Manager:
     1. npm (Node Package Manager)
  3. Frontend Framework/Library:
     1. React.js
     2. React Router
     3. Material-UI

### Backend Tools:

* 1. Code Editor:
     1. Visual Studio Code
  2. Version Control System:
     1. Git
     2. GitHub
  3. Package Manager:
     1. Maven
  4. Backend Framework:
     1. Java Spring Boot
     2. Spring Data JPA
     3. Database Tools:

### Database Management System:

* 1. MySQL

### API Development and Testing Tools:

* 1. Postman

## CHAPTER 6 IMPLEMENTATIONS

In this chapter, we are going to see about implementation of the full-stack application.

The implementation of your project, as a single developer, involves leveraging the power of React.js and Java Spring Boot to create a functional and user-friendly application. With React.js, you can design and develop an intuitive user interface, allowing users to easily navigate through the sign-in, sign-up, and landing pages. The landing page, featuring a table displaying records from the database, provides a comprehensive view of the data and empowers users to perform CRUD operations seamlessly through the frontend.

As you handle both frontend and backend development, you have the flexibility to ensure a seamless integration between the two components. With Java Spring Boot, you can efficiently handle database operations and expose RESTful APIs that enable smooth communication between the frontend and backend.

Throughout the implementation process, you can maintain a systematic approach by following established best practices. This includes adhering to coding standards, conducting thorough testing at each stage, and incorporating error handling mechanisms to enhance the robustness and reliability of your application.

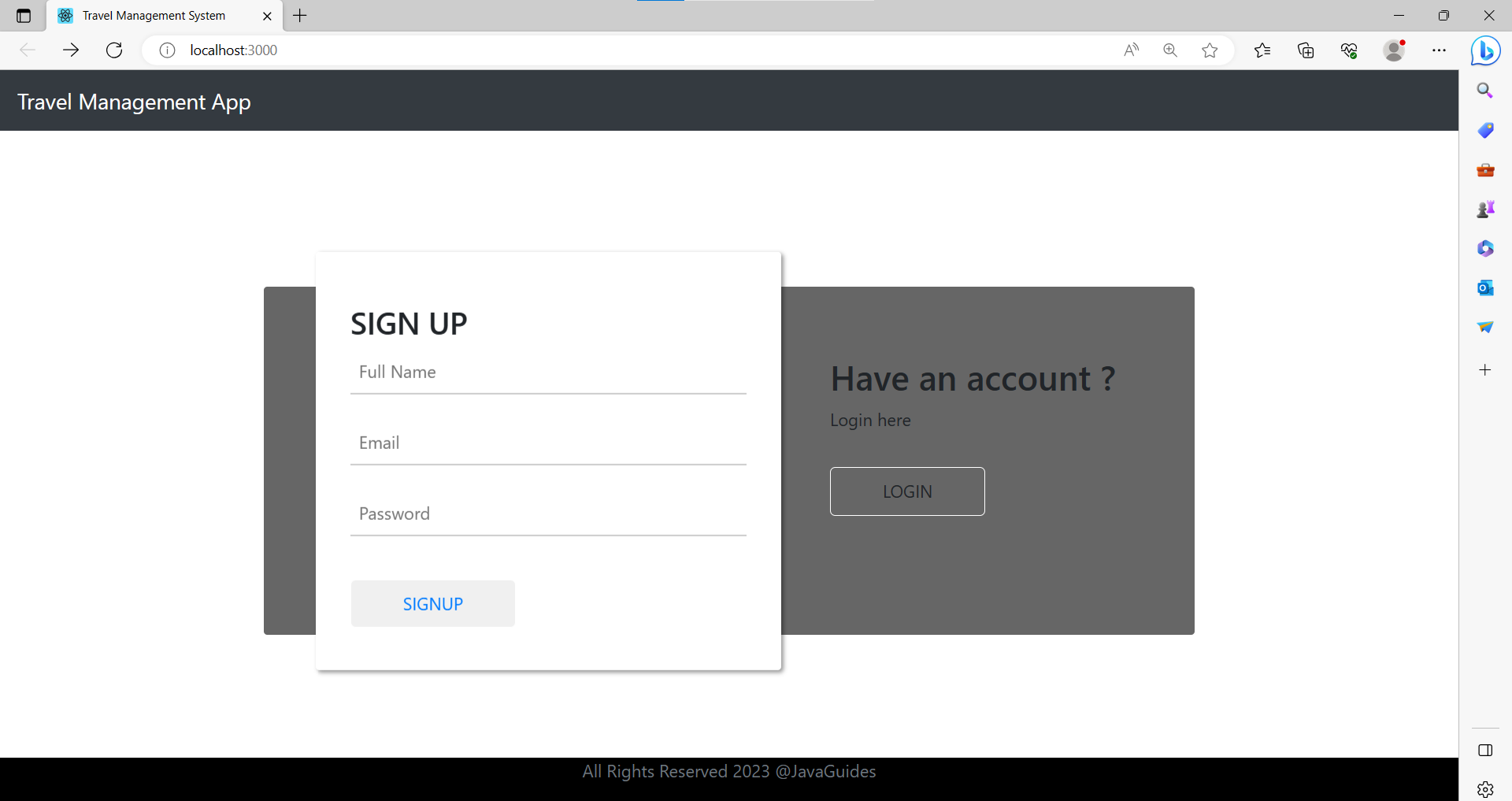
Moreover, as a single developer, you have the advantage of autonomy and control over the entire implementation process. You can fine-tune and optimize the performance of your application, as well as customize features according to your specific requirements. This allows for a streamlined development workflow

and the ability to quickly respond to any challenges or changes that may arise during the implementation phase.

By efficiently implementing React.js and Java Spring Boot, you can bring your project to life, creating a fully functional full-stack application. The implementation process, infused with your expertise and individual style, ensures that the application meets your vision while delivering a seamless user experience and robust functionality.

## 6.1 SIGNUP PAGE

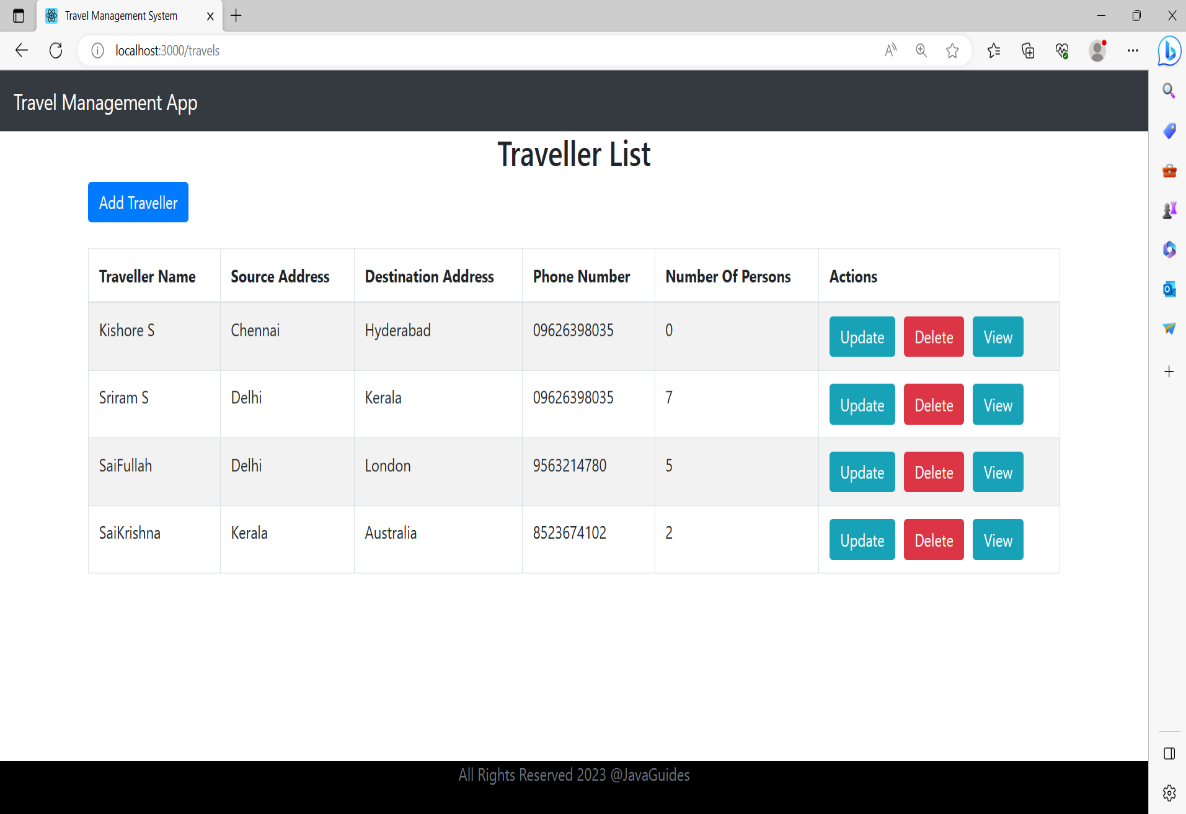
Signup page of the TravelPrism:



**Figure.No.6.1 Login Page**

## DETAILS PAGE

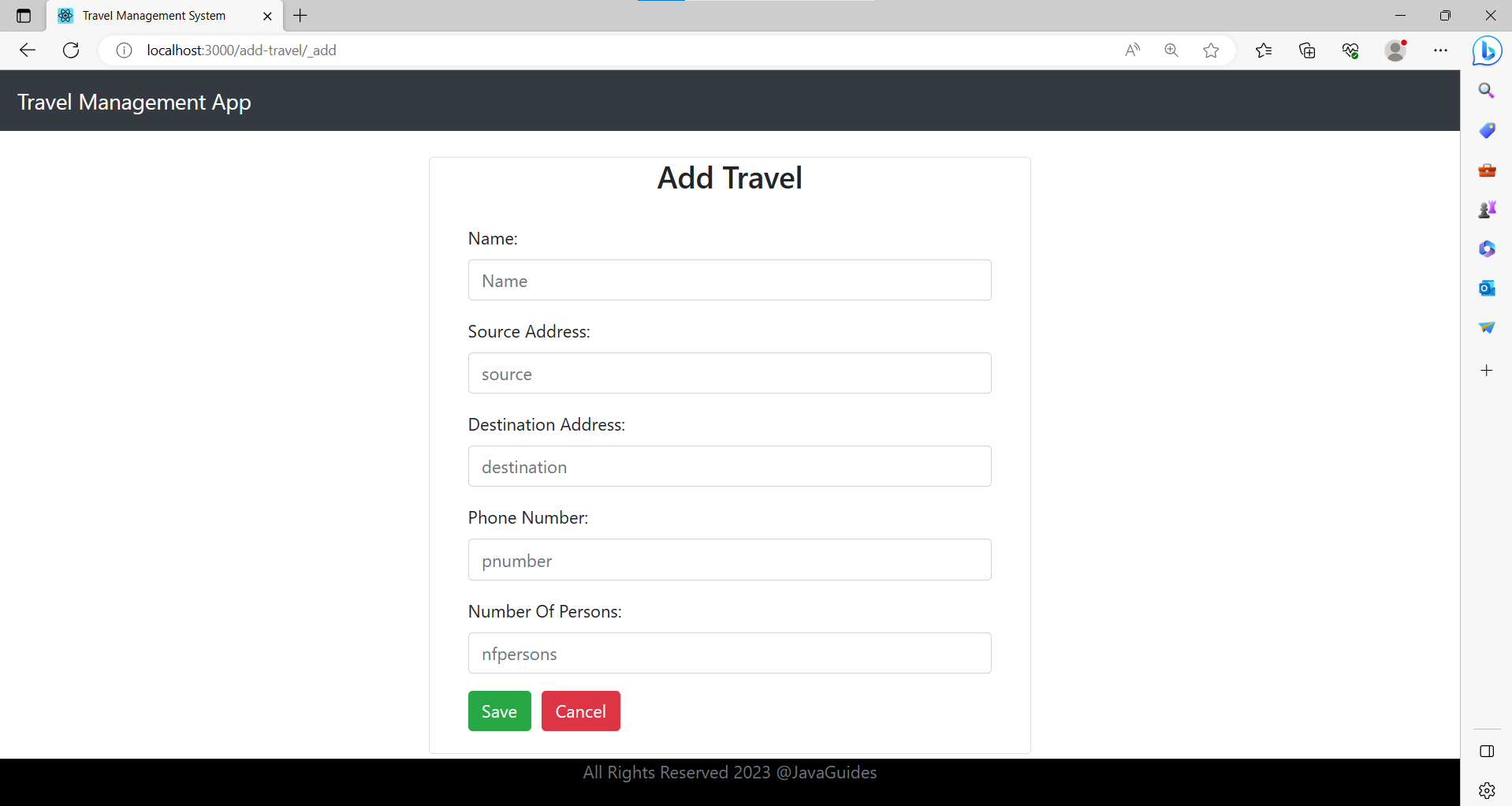
These pages show the entire details of Traveller persons. This page includes update, add, delete details options which navigate to their respective pages.



**Figure.No.6.2 Persons List**

## ADD PAGE

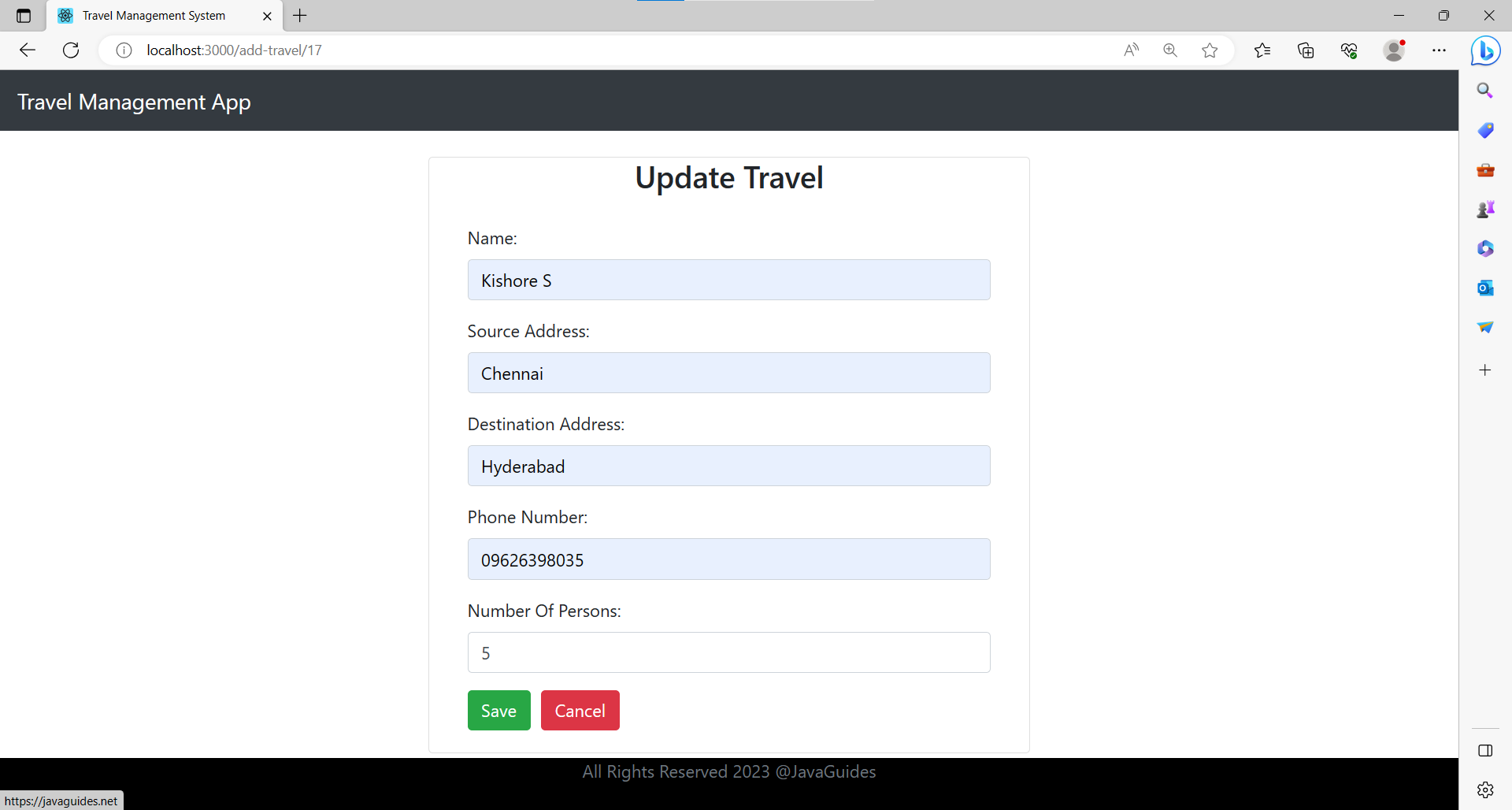
This page is used to add details to the database.



**Figure.No.6.3 Add New Details**

## UPDATE PAGE

This page is used to update the details that have been already saved in the database



**Figure.No.6.4 Update Traveller detail**

## CHAPTER 7 CONCLUSION AND FUTURE WORKS

In conclusion, our project successfully demonstrates the development of a basic full-stack application using React.js and Java Spring Boot. It showcases the integration between frontend and backend components, providing a foundation for further exploration and customization. Despite its limitations, the project serves as a valuable learning resource, allowing developers to understand the core concepts of full-stack development and gain hands-on experience.

## Future Works:

Enhanced Functionality: Adding advanced features such as validation, authorization, and JWT authentication would enhance the security and usability of the application. Implementing more sophisticated CRUD operations and incorporating search or filtering capabilities could further improve the user experience.

UI/UX Enhancements: Investing in a more visually appealing and user- friendly user interface (UI) design can elevate the overall look and feel of the application. Implementing responsive design principles would ensure optimal usability across different devices and screen sizes.

Performance Optimization: Conducting performance optimizations, such as implementing caching mechanisms, optimizing database queries, or employing server-side rendering, can improve the application's speed and responsiveness.

Scalability and Deployment: Considering the deployment of the application to a production environment, implementing scalability measures such as load balancing and horizontal scaling can ensure the application can handle increasing user traffic efficiently.

## APPENDIX 1

1. **SAMPLE CODE**

### TravellerControler.java

package com.travel.springboot.controller;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.travel.springboot.exception.ResourceNotFoundException;

import com.travel.springboot.model.Travel;

import com.travel.springboot.repository.TravelRepository;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/api/v1")

public class TravelController {

@Autowired

private TravelRepository travelRepository;

// get all travels

@GetMapping("/travels")

public List<Travel> getAllTravels(){

return travelRepository.findAll();

}

// create travel rest api

@PostMapping("/travels")

public Travel createTravel(@RequestBody Travel travel) {

return travelRepository.save(travel);

}

// get travel by id rest api

@GetMapping("/travels/{id}")

public ResponseEntity<Travel> getTravelById(@PathVariable Long id) {

Travel travel = travelRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Travel not exist with id :" + id));

return ResponseEntity.ok(travel);

}

// update travel rest api

@PutMapping("/travels/{id}")

public ResponseEntity<Travel> updateTravel(@PathVariable Long id, @RequestBody Travel travelDetails){

Travel travel = travelRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Travel not exist with id :" + id));

travel.setName(travelDetails.getName());

travel.setsource(travelDetails.getsource());

travel.setdestination(travelDetails.getdestination());

Travel updatedTravel = travelRepository.save(travel);

return ResponseEntity.ok(updatedTravel);

}

// delete travel rest api

@DeleteMapping("/travels/{id}")

public ResponseEntity<Map<String, Boolean>> deleteTravel(@PathVariable Long id){

Travel travel = travelRepository.findById(id)

.orElseThrow(() -> new ResourceNotFoundException("Travel not exist with id :" + id));

travelRepository.delete(travel);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(response);

}

}

### App.js

import React from 'react';

import logo from './logo.svg';

import './App.css';

import {BrowserRouter as Router, Route, Switch} from 'react-router-dom'

import ListTravelComponent from './components/ListTravelComponent';

import HeaderComponent from './components/HeaderComponent';

import FooterComponent from './components/FooterComponent';

import CreateTravelComponent from './components/CreateTravelComponent';

import UpdateTravelComponent from './components/UpdateTravelComponent';

import ViewTravelComponent from './components/ViewTravelComponent';

import TravelLogin from './components/TravelLogin';

function App() {

return (

<div>

<Router>

<HeaderComponent />

<div className="container">

<Switch>

<Route path = "/" exact component = {ListTravelComponent}></Route>

{/\*<Route path = "/" exact component = {TravelLogin}></Route>

<Route exact path='/createtravel'><CreateTravelComponent /></Route>\*/}

<Route path="/listtravelcomponent" component = {ListTravelComponent}></Route>

{/\* <Route exact path="/listtravelcomponent"><ListTravelComponent /></Route> \*/}

<Route path = "/travels" component = {ListTravelComponent}></Route>

<Route path = "/add-travel/:id" component = {CreateTravelComponent}></Route>

<Route path = "/view-travel/:id" component = {ViewTravelComponent}></Route>

{/\* <Route path = "/update-travel/:id" component = {UpdateTravelComponent}></Route> \*/}

</Switch>

</div>

<FooterComponent />

</Router>

</div>

);

}

export default App;

## Travel.java

## package com.travel.springboot.model;

## import jakarta.persistence.Entity;

## import jakarta.persistence.GeneratedValue;

## import jakarta.persistence.GenerationType;

## import jakarta.persistence.Id;

## import jakarta.persistence.Table;

## @Entity

## @Table(name = "heloooo")

## public class Travel {

## @Id

## @GeneratedValue(strategy = GenerationType.IDENTITY)

## private int id;

## private String name;

## private String source;

## private String destination;

## private String pnumber;

## private int nfpersons;

## 

## public int getId() {

## return id;

## }

## public void setId(int id) {

## this.id = id;

## }

## public String getName() {

## return name;

## }

## public void setName(String name) {

## this.name = name;

## }

## public String getsource() {

## return source;

## }

## public void setsource(String source) {

## this.source = source;

## }

## public String getdestination() {

## return destination;

## }

## public void setdestination(String destination) {

## this.destination = destination;

## }

## public String getPnumber() {

## return pnumber;

## }

## public void setPnumber(String pnumber) {

## this.pnumber = pnumber;

## }

## public int getNfpersons() {

## return nfpersons;

## }

## public void setNfpersons(int nfpersons) {

## this.nfpersons = nfpersons;

## }

## public Travel(int id, String name, String source, String destination, String pnumber, int nfpersons) {

## super();

## this.id = id;

## this.name = name;

## this.source = source;

## this.destination = destination;

## this.pnumber = pnumber;

## this.nfpersons = nfpersons;

## }

## public Travel()

## {

## 

## }

## }

## TravelRepository.java

## package com.travel.springboot.repository;

## import org.springframework.data.jpa.repository.JpaRepository;

## import org.springframework.stereotype.Repository;

## import com.travel.springboot.model.Travel;

## @Repository

## public interface TravelRepository extends JpaRepository<Travel, Long>{

## }

## REFERENECE:

1. Spring Boot Official Documentation
2. Baeldung
3. Spring Guides
4. RESTful Web Services with Spring Framework
5. "Pro Spring Boot 2" by Felipe Gutierrez
6. "Spring Boot in Action" by Craig Walls
7. "Full Stack Development with JHipster" by Deepu K Sasidharan and Sendil Kumar N
8. "Mastering Full-Stack React Web Development" by Tomasz Dyl
9. "Spring Boot 2 Recipes: A Problem-Solution Approach" by Marten Deinum, Daniel Rubio, Josh Long, and Iuliana Cosmina
10. "Hands-On Full Stack Development with Spring Boot 2 and React: Build modern and scalable full stack applications using Spring Framework 5 and React with Hooks" by Juha Hinkula and Peter Frost