

Namma Ooru Bus
(An Online Bus Ticket Booking
Application)
Documentation

Table of Contents

1. Introduction
2. Technologies Used
3. Project Structure
4. Backend and Frontend Setup
 - ❖ Development Environment
 - ❖ Dependencies
 - ❖ Configuration
 - ❖ Entity Class
 - ❖ Repository Layer
 - ❖ Service Layer
 - ❖ Controller Layer
 - ❖ DTO Layer
 - ❖ Exception Layer
 - ❖ Config Layer
 - ❖ HTML Pages for Bookings, User and Admin with Tailwind CSS
5. Table Schema
6. Web Endpoints
7. Running the Application
8. Conclusion

Introduction

In today's fast-paced world, public transportation plays a crucial role in ensuring that individuals can commute efficiently and affordably. With the increasing urbanization and population growth, the demand for reliable and user-friendly transportation solutions has never been higher. The Namma Ooru Bus project aims to address this need by providing an online bus ticket booking application that simplifies the process of purchasing tickets for public bus services. The name "Namma Ooru" translates to "My Town" in Tamil, reflecting the project's focus on serving local communities.

The primary purpose of the Namma Ooru Bus application is to streamline the bus ticket booking process for users. Traditionally, purchasing bus tickets can be a cumbersome experience, often requiring individuals to visit physical ticket counters, wait in long queues, and deal with cash transactions. This application seeks to eliminate these inconveniences by offering a digital platform where users can easily book tickets from the comfort of their homes or on the go.

The application is designed to cater to a diverse audience, including daily commuters, students, and tourists. By providing a user-friendly interface and a seamless booking experience, the Namma Ooru Bus application aims to encourage more people to utilize public transportation, thereby reducing traffic congestion and promoting sustainable travel options.

The target audience for the Namma Ooru Bus application includes daily commuters who rely on public transportation for their daily travel to work or school, students who depend on buses to reach educational institutions, tourists exploring the local area, and local residents looking to travel within their town or city.

The Namma Ooru Bus application is equipped with several key features designed to enhance user experience, including user registration and login, real-time bus route and schedule information, easy ticket booking,

cancellation and refund options, and notifications and alerts regarding bookings.

In summary, the Namma Ooru Bus application represents a significant step forward in modernizing public transportation services. By leveraging technology to simplify the ticket booking process, the application aims to enhance the overall travel experience for users while promoting the use of public transport. As urban areas continue to grow, solutions like Namma Ooru Bus will play a vital role in creating efficient and sustainable transportation systems that benefit both individuals and communities. Through continuous improvement and user feedback, the application aspires to evolve and meet the changing needs of its users, ultimately contributing to a more connected and accessible urban environment.

Technologies Used

Backend: Java, Spring Boot, Maven

Frontend: Thymeleaf, HTML, Tailwind CSS

Testing: Junit Jupiter, Mockito Core

Database: MongoDB

Development Tools:

- ❖ IntelliJ IDEA Community Edition 2024.3.5

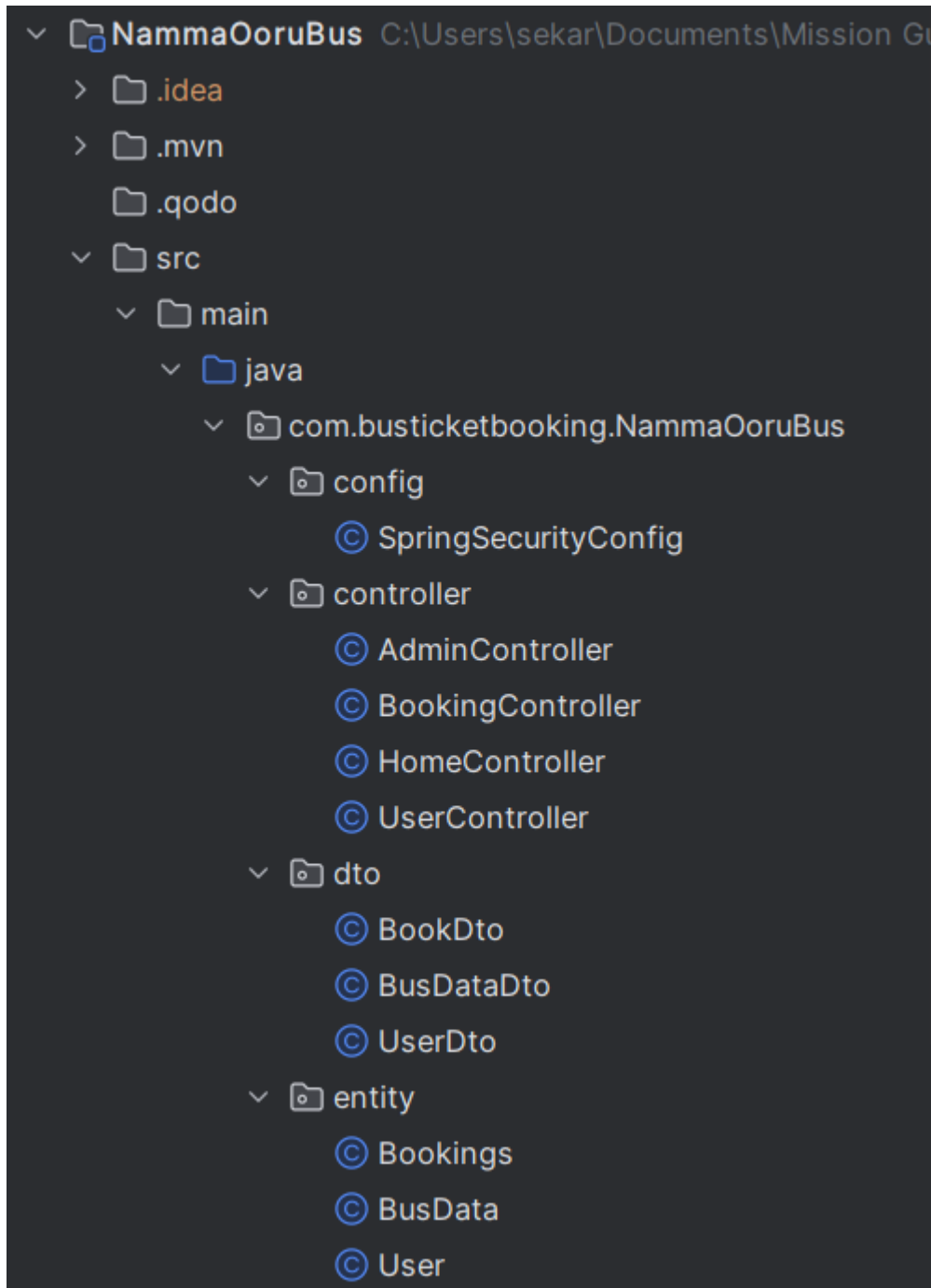
Note: Don't try the latest version (2025.1) and above. Because thymeleaf is not supportable for updated version. Kindly recommended to use 2024.3.5 and below version.




















Project Structure

Spring Boot Structure with Dependencies

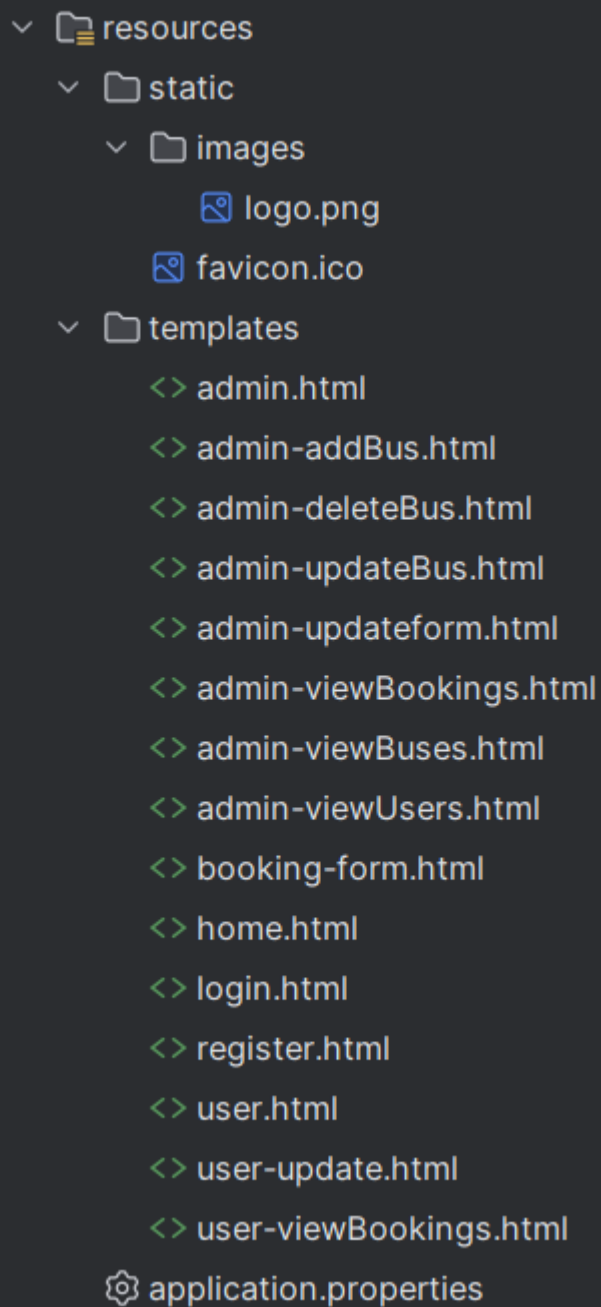
- ❖ Spring Boot Starter Web
- ❖ Thymeleaf
- ❖ Starter Test
- ❖ MongoDB Driver
- ❖ Lombok
- ❖ Spring Validation
- ❖ Spring Security

Backend Setup



- ▼  exception
 -  DuplicateBusNumberException
 -  EmailPresent
 -  ErrorDetails
 -  GlobalExceptionHandler
 -  PhoneNumberPresent
 -  ResourceNotFoundException
- ▼  repository
 -  BookingRepo
 -  BusDataRepo
 -  UserRepo
- ▼  service
 -  BookingService
 -  BookingServiceImpl
 -  BusDataService
 -  BusDataServiceImpl
 -  UserService
 -  UserServiceImpl
-  NammaOoruBusApplication

Frontend Setup



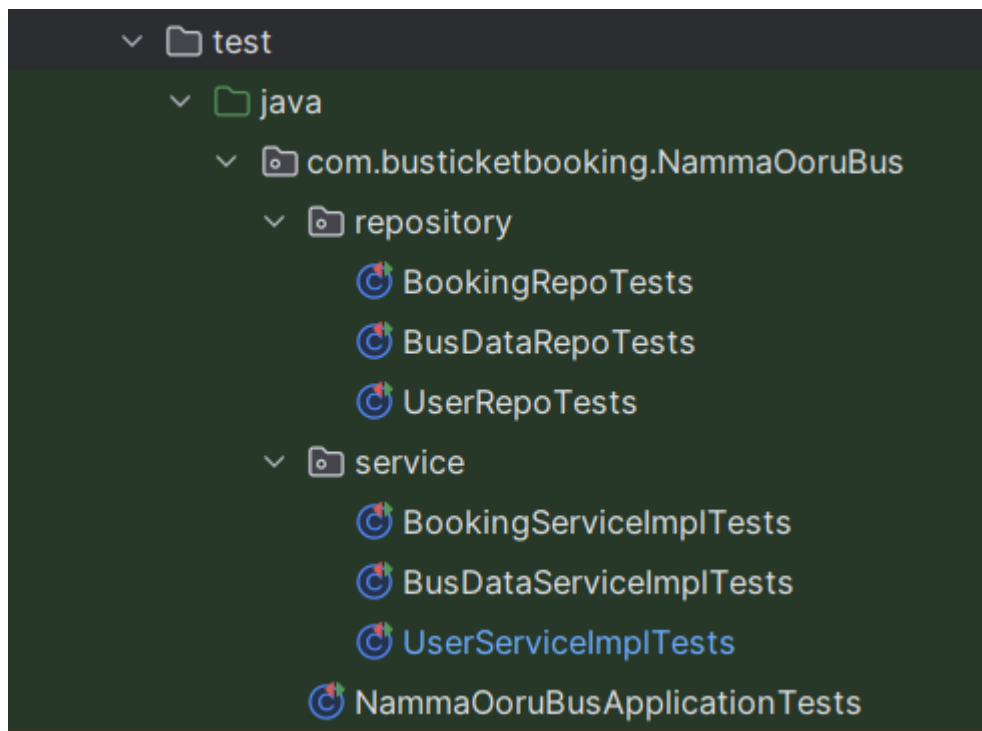
```

  resources
  static
    images
      logo.png
      favicon.ico
    templates
      admin.html
      admin-addBus.html
      admin-deleteBus.html
      admin-updateBus.html
      admin-updateform.html
      admin-viewBookings.html
      admin-viewBuses.html
      admin-viewUsers.html
      booking-form.html
      home.html
      login.html
      register.html
      user.html
      user-update.html
      user-viewBookings.html
  application.properties

```

The image shows a file explorer interface with a dark background. It displays a project structure for a frontend application. The root directory contains a 'resources' folder, a 'static' folder, and an 'application.properties' file. The 'static' folder is expanded, showing an 'images' subfolder with 'logo.png' and 'favicon.ico' files, and a 'templates' subfolder containing 15 HTML files. The HTML files are: 'admin.html', 'admin-addBus.html', 'admin-deleteBus.html', 'admin-updateBus.html', 'admin-updateform.html', 'admin-viewBookings.html', 'admin-viewBuses.html', 'admin-viewUsers.html', 'booking-form.html', 'home.html', 'login.html', 'register.html', 'user.html', 'user-update.html', and 'user-viewBookings.html'. Each file is preceded by a green double-angle bracket icon (<>). The 'application.properties' file is preceded by a gear icon.

Testing Setup



Core Components and Packages

Backend Contents

This section details the purpose of the key packages within `src/main/java/com/busticketbooking/NammaOoruBus/`.

1. config

* Purpose: Declares a class that provides **bean definitions** using `@Bean` methods, used to **centralize configuration** of dependencies and services and enables **Java-based configuration** for the Spring container

* Key File: `SpringSecurityConfig.java`

This configures security settings such as authentication, authorization, and access control for the application.

2.controller

* Purpose: This package handles incoming HTTP requests and maps them to appropriate service methods to process user actions like booking tickets or viewing schedules.

*Key Files:

1. AdminController.java

*A Spring MVC that manages admin-specific HTTP requests for tasks like adding buses, managing schedules, and overseeing bookings.

2. BookingController.java

*A Spring MVC that manages user requests related to ticket booking, viewing bookings, and handling cancellations.

3. HomeController.java

*A Spring MVC that handles general user-facing requests such as loading the home page, searching buses, or displaying basic site information.

4. UserController.java

*A Spring MVC that handles user-related operations such as registration, login, profile management, and viewing user-specific data.

3. dto

* Purpose: This package holds Data Transfer Objects (DTOs).

Encapsulation: Hide model/internal details and expose only required fields.

Security: Prevent sending sensitive or unwanted fields (e.g., passwords, internal IDs).

Efficiency: Avoid sending large or nested entities when only a subset is needed.

Validation: Enable client-side or request-based validation on input data.

Mapping Simplification: Customize how domain models are presented to clients or APIs.

* Key Files:

1.BookDto.java

*Serves as a data transfer object to carry booking-related data between the client and server.

2.BusDataDto.java

*A data transfer object used to encapsulate and transfer bus-related details such as route, timings, and seat availability.

3.UserDto.java

*A data transfer object used to carry user-related information such as name, email, and credentials between the client and server.

4. entity

* Purpose: This package contains the MongoDB model classes, which are designed to represent the data model of the application.

* Key Files:

1.Bookings.java

*Represents and manage booking details such as user, bus, seats, and travel date within the application's data model.

2.BusData.java

*Used to store and manage information about buses, such as bus number, route, timings, and seat availability.

3.User.java

*Used to define and manage user entity details such as name, email, password, and contact information.

5. exception

* Purpose: This package is used to handle errors gracefully and ensure smooth program flow by providing meaningful error messages and actions.

* Key Files:

1. DuplicateBusNumberException.java

*Used to handle errors when attempting to add a bus with a number that already exists.

2. EmailPresent.java

*Used to handle exceptions when a user tries to register with an email that is already in use.

3. ErrorDetails.java

*Used to encapsulate and provide structured error information, such as timestamp, message, and details, for exception handling responses.

4. GlobalExceptionHandler.java

*Used to centrally handle and respond to exceptions across the entire application with consistent error messages.

5. PhoneNumberPresent.java

*Used to handle exceptions when a user tries to register with a phone number that is already registered.

6. ResourceNotFoundException.java

*Used to handle cases where a requested resource, such as a user or bus, cannot be found in the system.

6. repository

Purpose: This package provides an interface for accessing, managing, and querying data from the database for various entities like users, buses, and bookings.

* Key Files:

1. BookingRepo.java

*Used to provide database access methods for performing CRUD operations on booking records.

2. BusDataRepo.java

*Used to handle database operations related to bus information, such as saving, updating, and retrieving bus details.

3.UserRepo.java

*Used to manage database operations for user entities, including saving, finding, and validating user data.

7.Service

* Purpose: This package contains the business logic and coordinate between the controller and repository layers.

* Key Files:

1.BookingService.java

*An interface that used to implement the business logic for creating, managing, and retrieving bookings.

2.BookingServiceImpl.java

*An implementation class that used to provide the concrete implementation of the booking-related business logic defined in the BookingService interface.

3.BusDataService.java

*An interface that used to define the business logic interface for managing bus-related operations such as adding, updating, and retrieving bus data.

4. BusDataServiceImpl.java

*An implementation class that used to provide the implementation of bus-related business logic defined in the BusDataService interface.

5.UserService.java

*An interface that used to define the business logic interface for user-related operations like registration, authentication, and profile management.

6. UserServiceImpl.java

*An implementation class that used to provide the concrete implementation of user-related business logic defined in the UserService interface.

NammaOoruBusApplication.java

* Purpose: This is the main class that bootstraps and launches the Spring Boot application. It contains the main method.

Frontend Contents

This section details the purpose of the key packages within `src/main/resources/`.

Purpose: This directory is used for storing configuration files (e.g., `application.properties` or `application.yml`), static resources, and templates.

A. Static

- * In this directory, I've added a `favicon.ico` which consists of 48x48 ICO file with 32-bit color that related to the application.

- * I've added a directory named as `images` and included a `logo.png` file that relates to the application.

B. Templates

In this directory, there are 15 html files available for accessing the application with a presentable view, user friendly, easy access and attractive icons, animations and effects.

Listing the html files that contains in `src/main/resources/templates`

- `admin.html` -> admin page to access
- `admin-addbus.html` -> adding a new bus by admin
- `admin-deletebus.html` -> deleting an existing bus by admin
- `admin-updatebus.html` -> update an existing bus by admin
- `admin-updateform.html` -> update the bus details by admin
- `admin-viewbookings.html` -> admin can view the bookings by user
- `admin-viewbuses.html` -> admin can view the list of buses
- `admin-viewUsers.html` -> admin can view the list of users
- `booking-form.html` -> user can fill the form to book the ticket
- `home.html` -> Homepage for both user and admin
- `login.html` -> Login page for existing user
- `register.html` -> Registration page for new user
- `user.html` -> User page to access
- `user-update.html` -> Update profile by the user
- `user-viewbookings.html` -> Past booking history of user

Testing Contents

This section details the purpose of the key packages within `src/test/java/com/busticketbooking/NammaOoruBus/`.

1. repository

Purpose: In tests, repositories are used to **prepare, validate, and clean up** real data in an **in-memory database** to ensure the application behaves correctly under different conditions.

*Key files:

1.BookingRepositoryTests.java

* A test class that used to test and verify the correctness of database operations related to bookings. (Test passed 1/1)

2.BusDataRepositoryTests.java

* A test class that used to test and validate the database operations for managing bus data. (Test passed 3/3)

3.UserRepositoryTests.java

* A test class that to test and ensure the correctness of database operations related to user management. (Test passed 3/3)

2. Service

Purpose: The service layer is tested to ensure that the application's business logic works correctly, independently of the database or web layer.

*Key files:

1.BookingServiceImplTests.java

* A test class that used to test the business logic and functionality implemented in the BookingServiceImpl class. (Test passed 2/2)

2.BusDataServiceImplTests.java

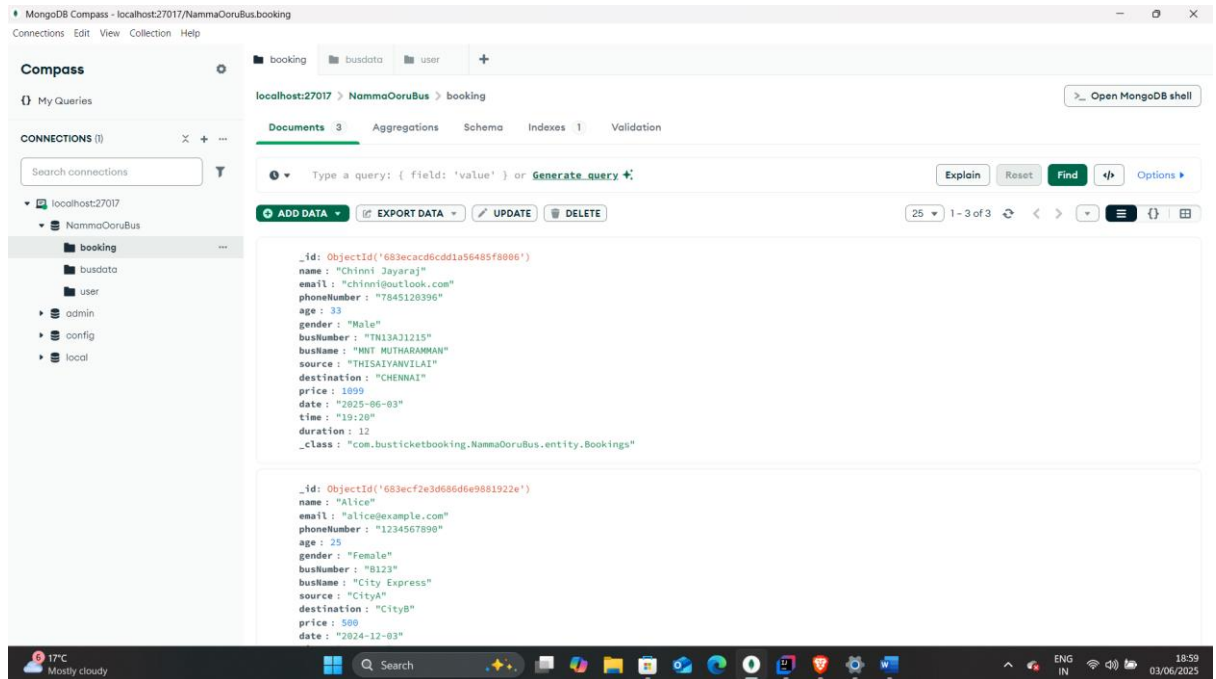
* A test class that used to test the business logic and functionality implemented in the BusDataServiceImpl class. (Test passed 6/6)

3.UserServiceImplTests.java

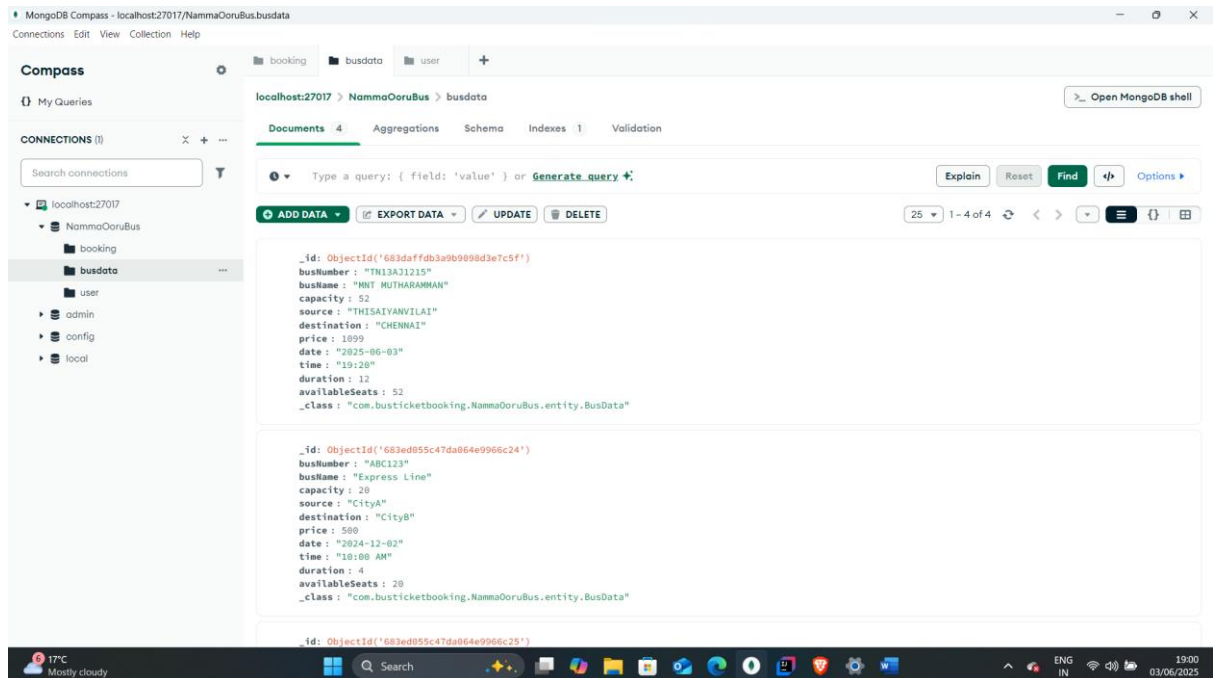
* A test class that used to test the business logic and functionality implemented in the UserServiceImpl class. (Test passed 6/6)

Table Schema

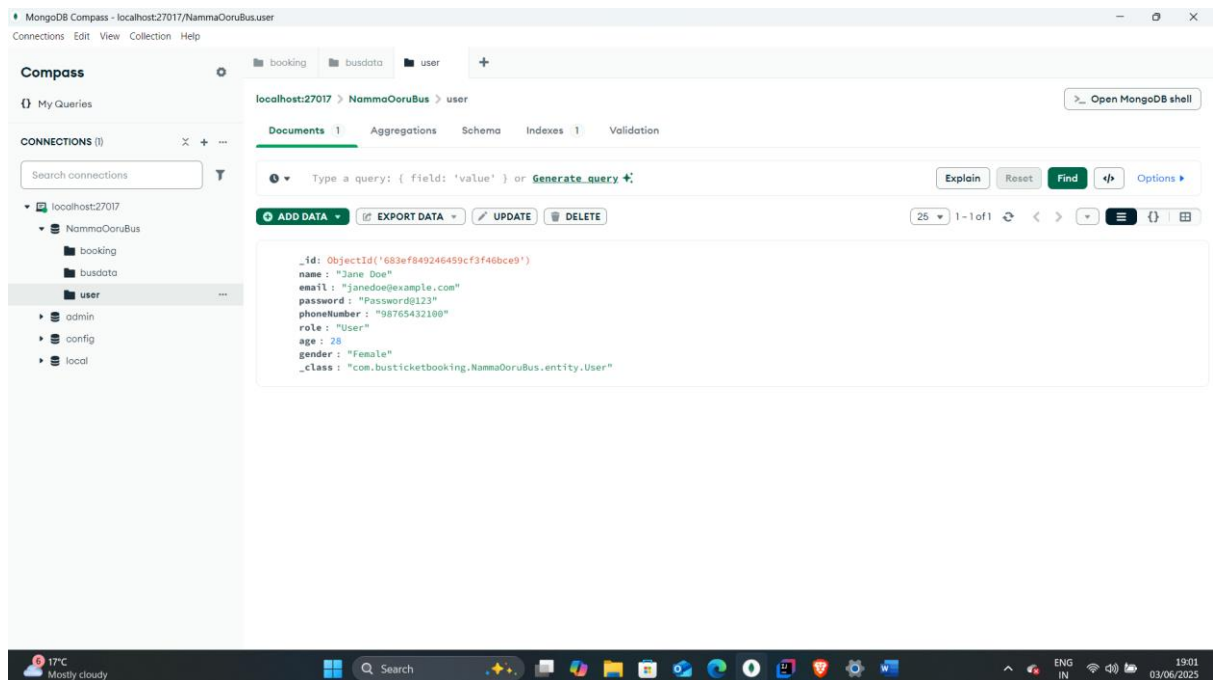
1) Booking Table



2) BusData Table



3) User Table

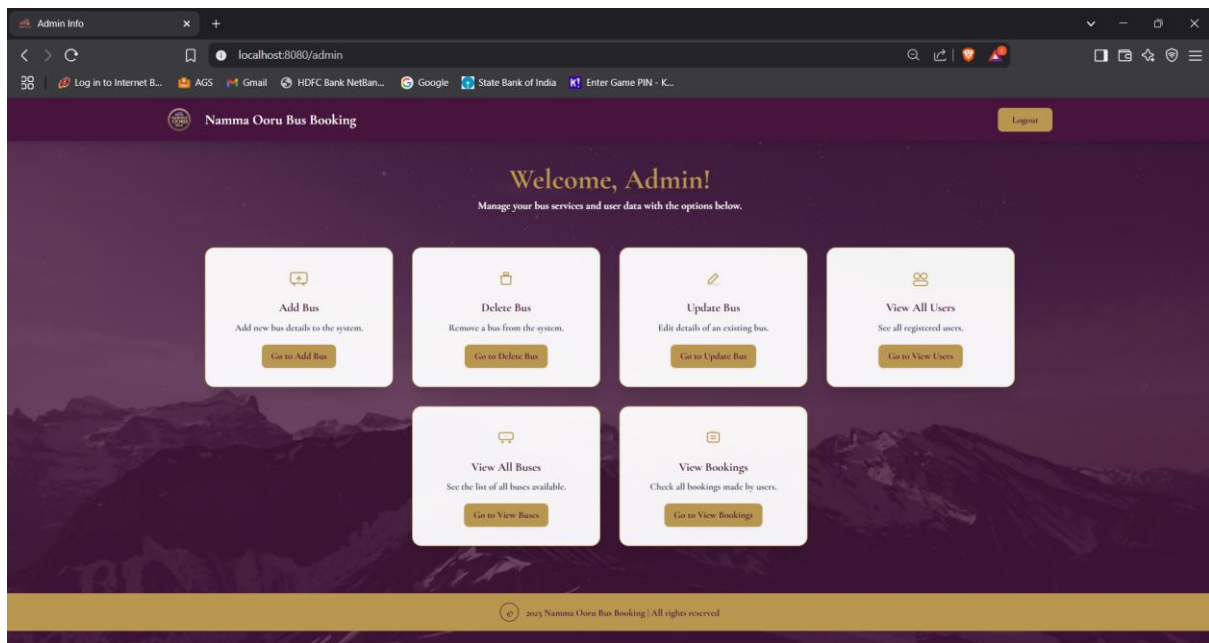


Web Endpoints

AdminController.java

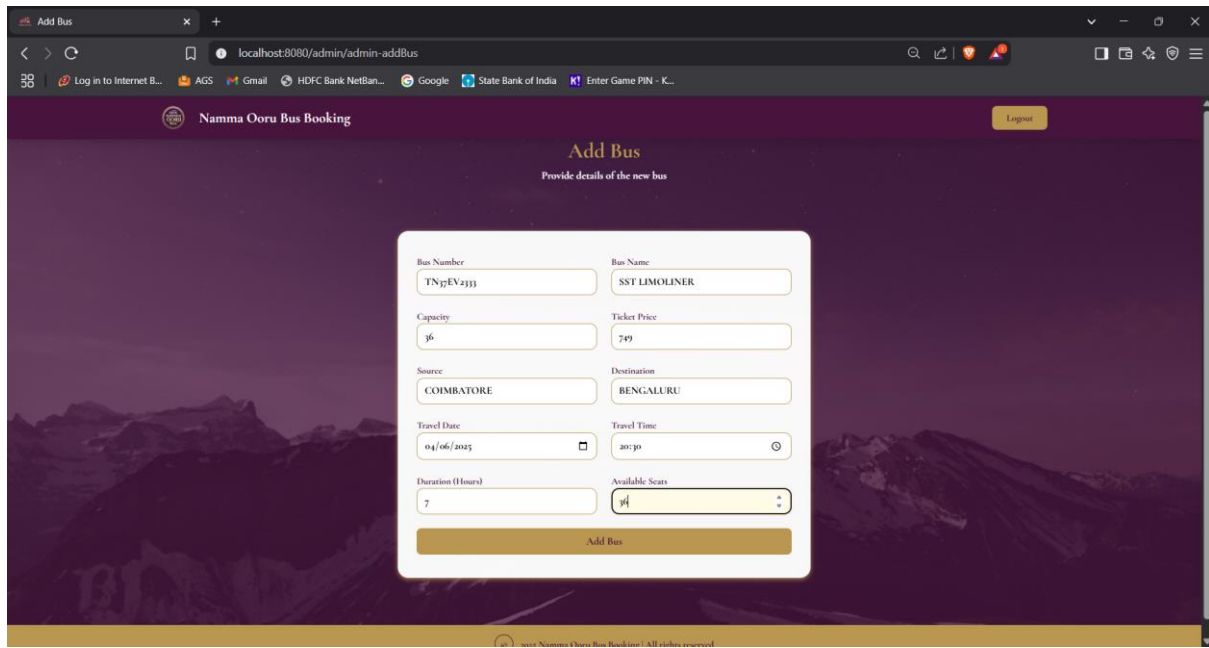
1. Admin Homepage

- **Endpoint:** /admin
- **Method:** GET
- **Description:** Shows the admin home page.



2. Adding a New Bus

- **Endpoint:** /admin/admin-addBus
- **Method:** GET
- **Description:** Displays the form to add a new bus.



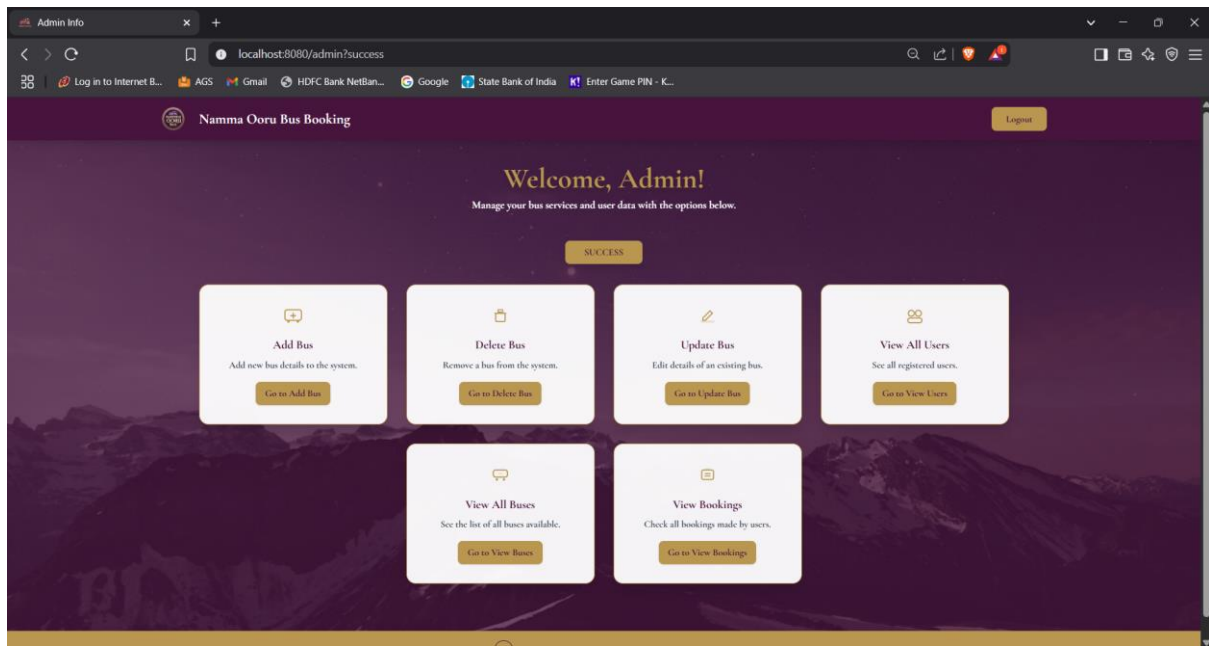
The screenshot shows a web browser window with the URL `localhost:8080/admin/admin-addBus`. The page title is "Add Bus" and the subtitle is "Provide details of the new bus". The form is titled "Add Bus" and contains the following fields:

| Field | Value |
|------------------|---------------|
| Bus Number | TN7EV2333 |
| Bus Name | SST LIMOLINER |
| Capacity | 36 |
| Ticket Price | 749 |
| Source | COIMBATORE |
| Destination | BENGALURU |
| Travel Date | 04/06/2025 |
| Travel Time | 20:30 |
| Duration (Hours) | 7 |
| Available Seats | 36 |

At the bottom of the form is a button labeled "Add Bus". The background of the page features a mountain landscape. The footer text reads: "© 2025 Namma Ooru Bus Booking. All rights reserved."

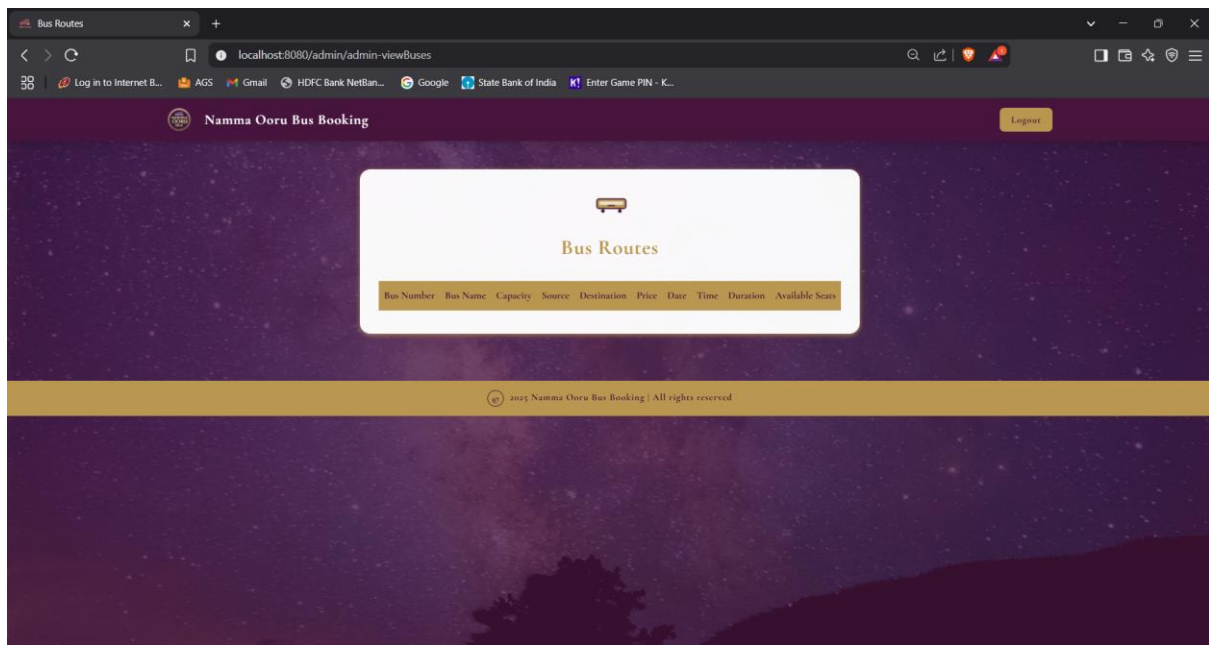
3. Saving a new bus

- **Endpoint:** /admin/admin-addBus/save
- **Method:** POST
- **Description:** Handles the submission of the add bus form and saves a new bus.



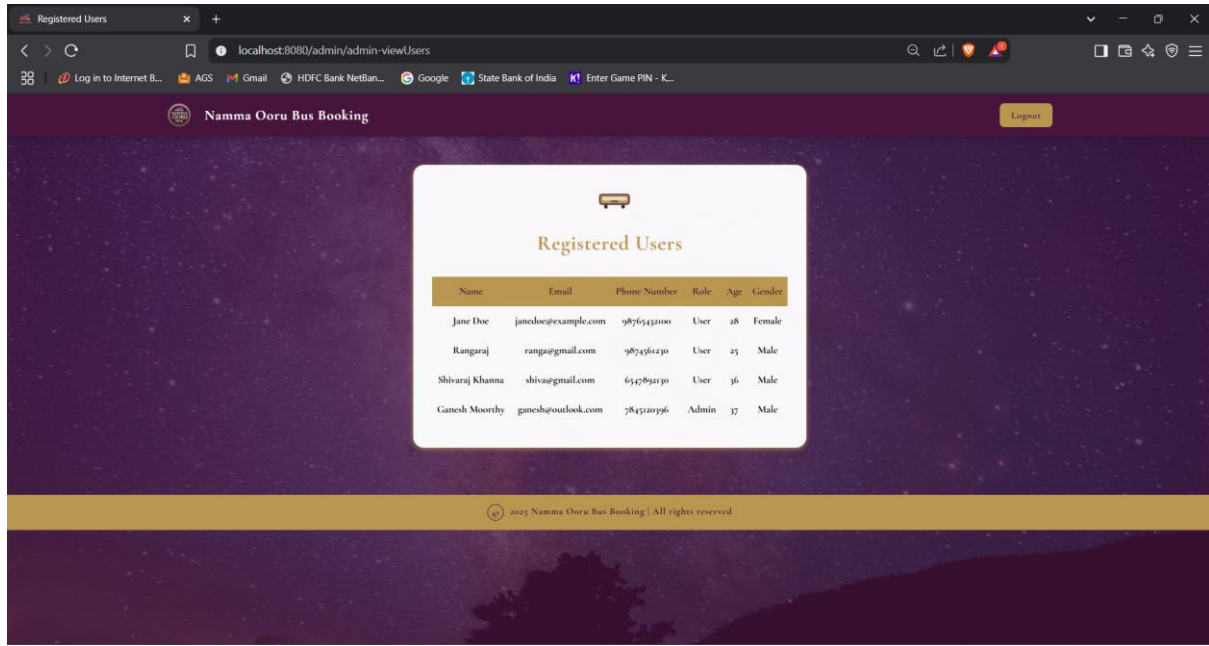
4. Viewing a list of buses

- **Endpoint:** /admin/admin-viewBuses
- **Method:** GET
- **Description:** Displays a list of all buses.



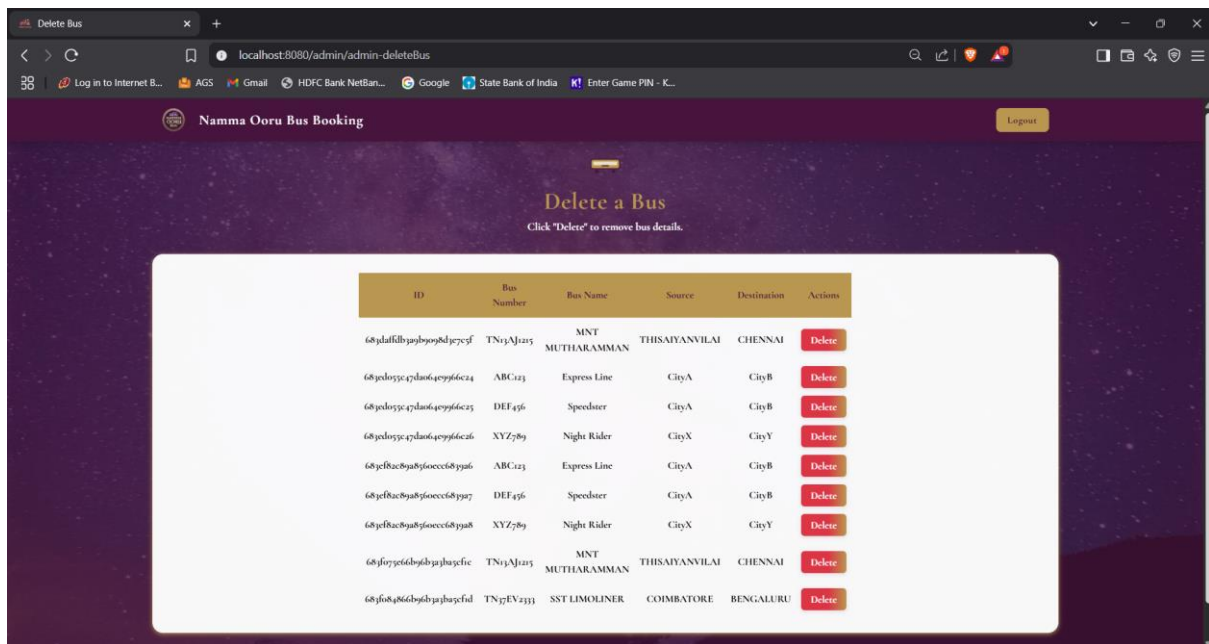
5. Viewing a list of users

- **Endpoint:** /admin/admin-viewUsers
- **Method:** GET
- **Description:** Displays a list of all users.



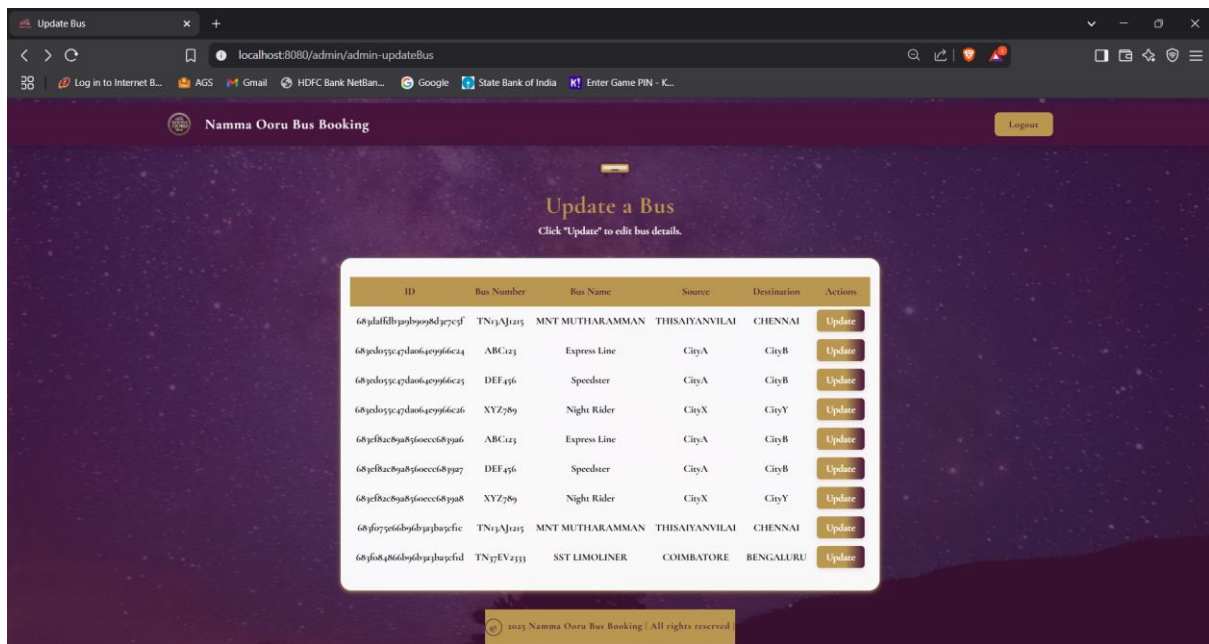
6. Deleting a bus

- **Endpoint:** /admin/admin-deleteBus
- **Method:** GET
- **Description:** Shows the page to select and delete a bus.



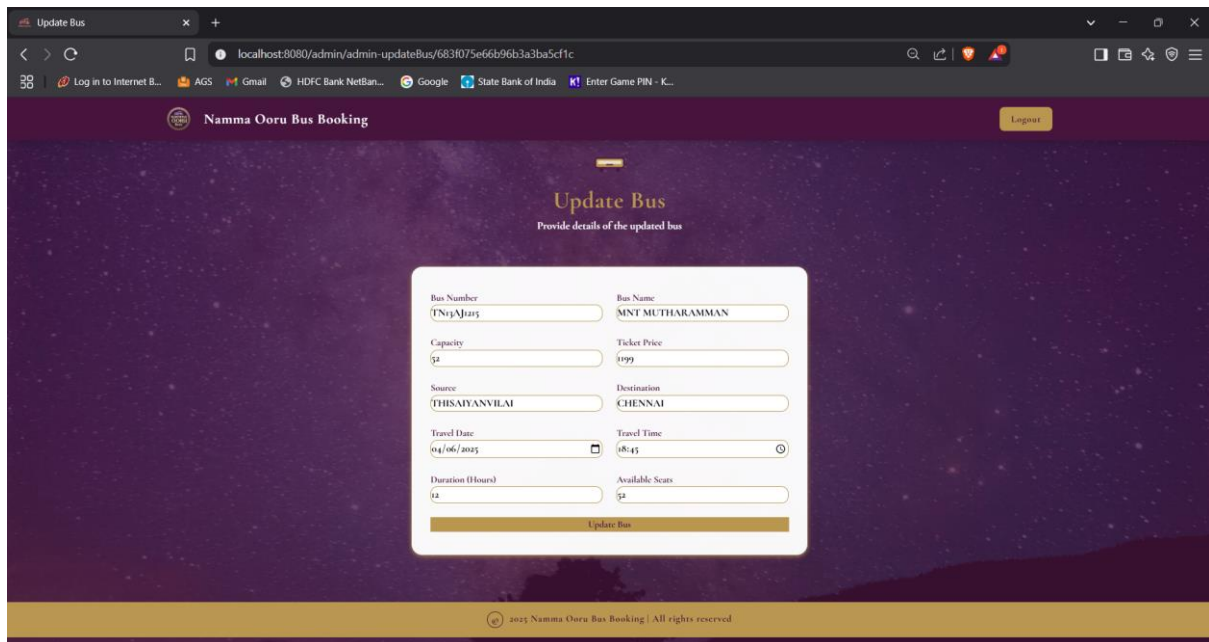
7. Updating an existing bus

- **Endpoint:** /admin/admin-updateBus
- **Method:** GET
- **Description:** Shows the page to select a bus for updating.



8. Updating a form

- **Endpoint:** /admin/admin-updateform/{id}
- **Method:** POST
- **Description:** Handles the submission of the update bus form and updates the bus with the given id.



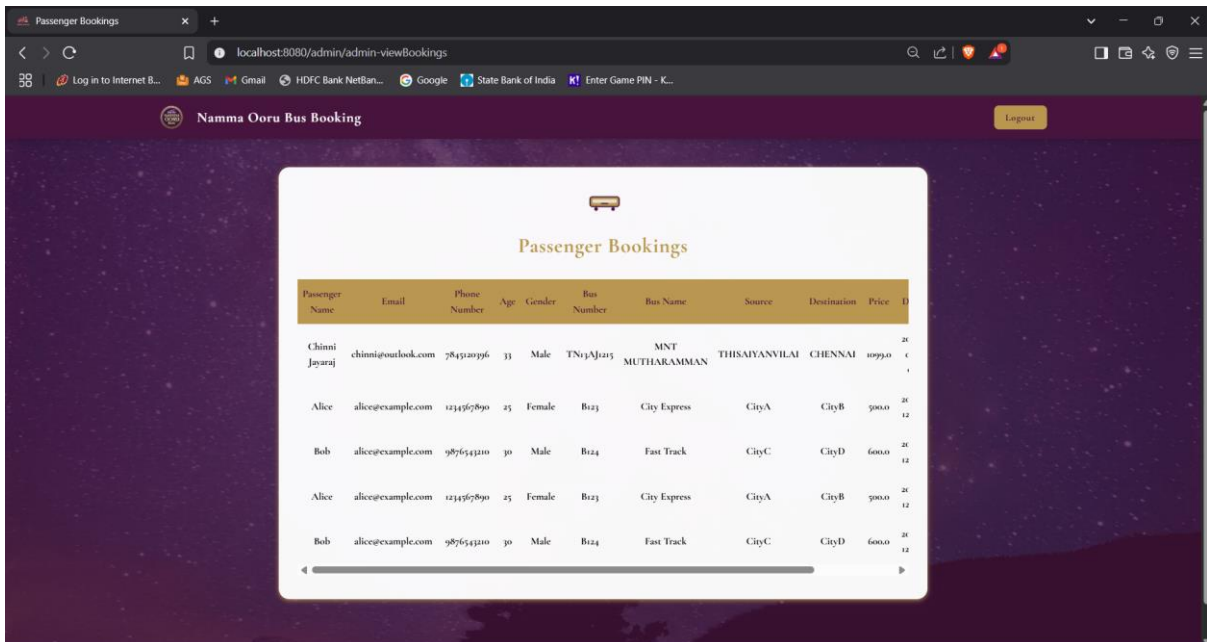
The screenshot shows a web browser window with the URL `localhost:8080/admin/admin-updateBus/683f075e66b96b3a3ba5cf1c`. The page title is "Update Bus" and the subtitle is "Provide details of the updated bus". The form is a white card with a light purple background. It contains the following fields:

| Field | Value |
|------------------|------------------|
| Bus Number | TN01AJ1015 |
| Bus Name | MINT MUTHARAMMAN |
| Capacity | 52 |
| Ticket Price | 1999 |
| Source | THESAIYANVELAI |
| Destination | CHENNAI |
| Travel Date | 04/06/2025 |
| Travel Time | 08:45 |
| Duration (Hours) | 12 |
| Available Seats | 52 |

At the bottom of the form is a button labeled "Update Bus". The footer of the page reads "© 2025 Namma Ooru Bus Booking | All rights reserved".

9. Viewing the details of bookings

- **Endpoint:** /admin/admin-viewBookings
- **Method:** GET
- **Description:** Displays a list of all bookings.



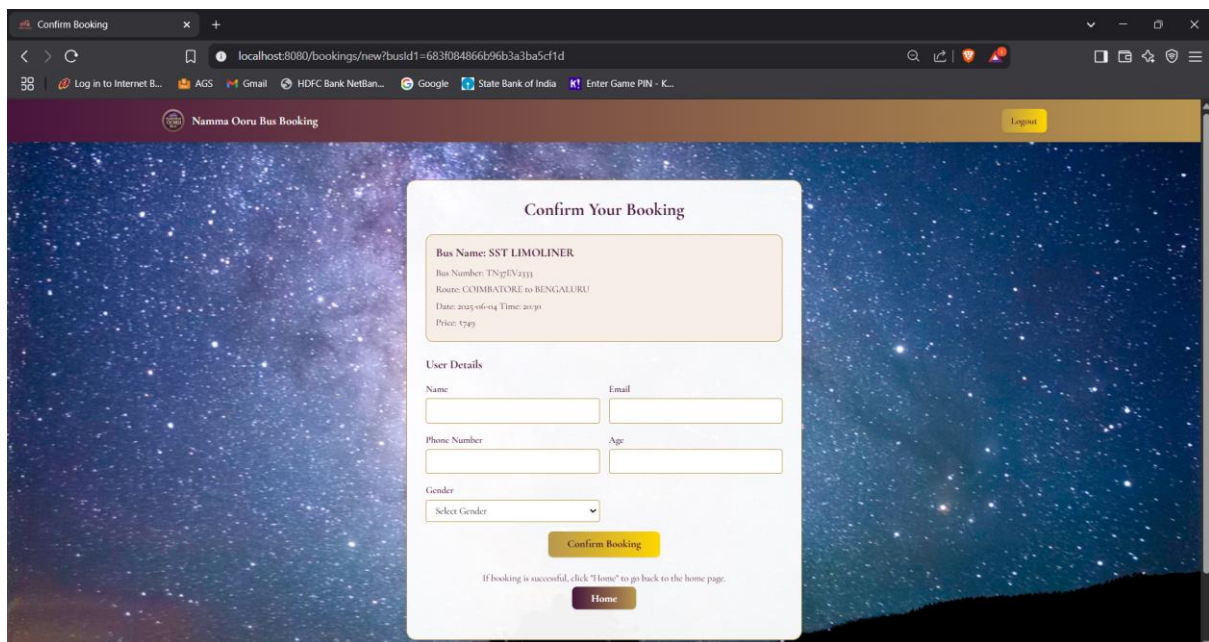
The screenshot shows a web browser window with the URL `localhost:8080/admin/admin-viewBookings`. The page header includes the logo and name 'Namma Ooru Bus Booking' and a 'Logout' button. The main content area features a table titled 'Passenger Bookings' with the following data:

| Passenger Name | Email | Phone Number | Age | Gender | Bus Number | Bus Name | Source | Destination | Price | Date |
|----------------|--------------------|--------------|-----|--------|------------|-----------------|--------------------|-------------|---------|------------|
| Chinni Jayaraj | chinni@outlook.com | 9849203996 | 33 | Male | TN13AJ2015 | MNT MUTHARAMMAN | THIRUVANANTHAPURAM | CHENNAI | 10999.0 | 20/11/2023 |
| Alice | alice@example.com | 1234567890 | 25 | Female | B123 | City Express | CityA | CityB | 5000.0 | 20/12/2023 |
| Bob | alice@example.com | 9876543210 | 30 | Male | B124 | Fast Track | CityC | CityD | 6000.0 | 20/12/2023 |
| Alice | alice@example.com | 1234567890 | 25 | Female | B123 | City Express | CityA | CityB | 5000.0 | 20/12/2023 |
| Bob | alice@example.com | 9876543210 | 30 | Male | B124 | Fast Track | CityC | CityD | 6000.0 | 20/12/2023 |

BookingController.java

1. Creating a new booking

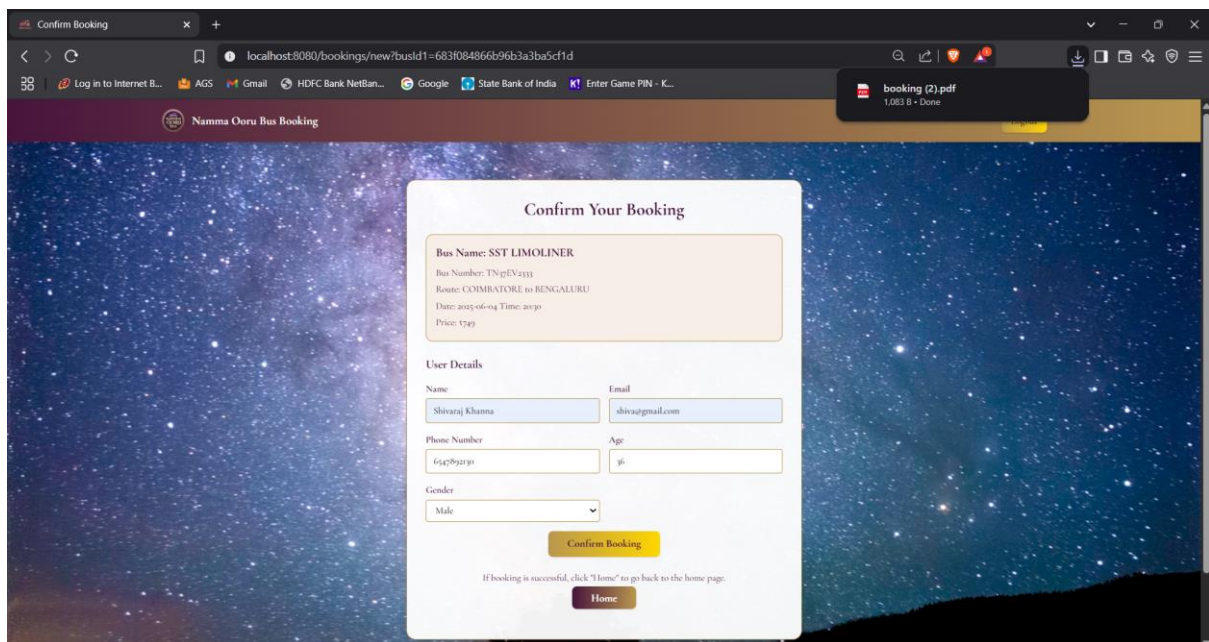
- **Endpoint:** /bookings/new
- **Method:** GET
- **Description:** Displays the booking form for a specific bus, using the busId1 request parameter.



The screenshot shows a web browser window with the address bar displaying `localhost:8080/bookings/new?busId1=683f084866b96b3a3ba5cf1d`. The page title is "Confirm Booking". The main content area has a dark blue starry background. A white modal box titled "Confirm Your Booking" is centered on the page. Inside the modal, the booking details are listed: "Bus Name: SST LIMOLINER", "Bus Number: TN97EV3111", "Route: COIMBATORE to BENGALURU", "Date: 2023-06-04 Time: 20:30", and "Price: 1749". Below this, the "User Details" section contains input fields for "Name", "Email", "Phone Number", and "Age", and a dropdown menu for "Gender" with the option "Select Gender". A yellow "Confirm Booking" button is positioned below the form fields. At the bottom of the modal, a message states "If booking is successful, click 'Home' to go back to the home page." with a purple "Home" button.

2. Viewing a list of booking

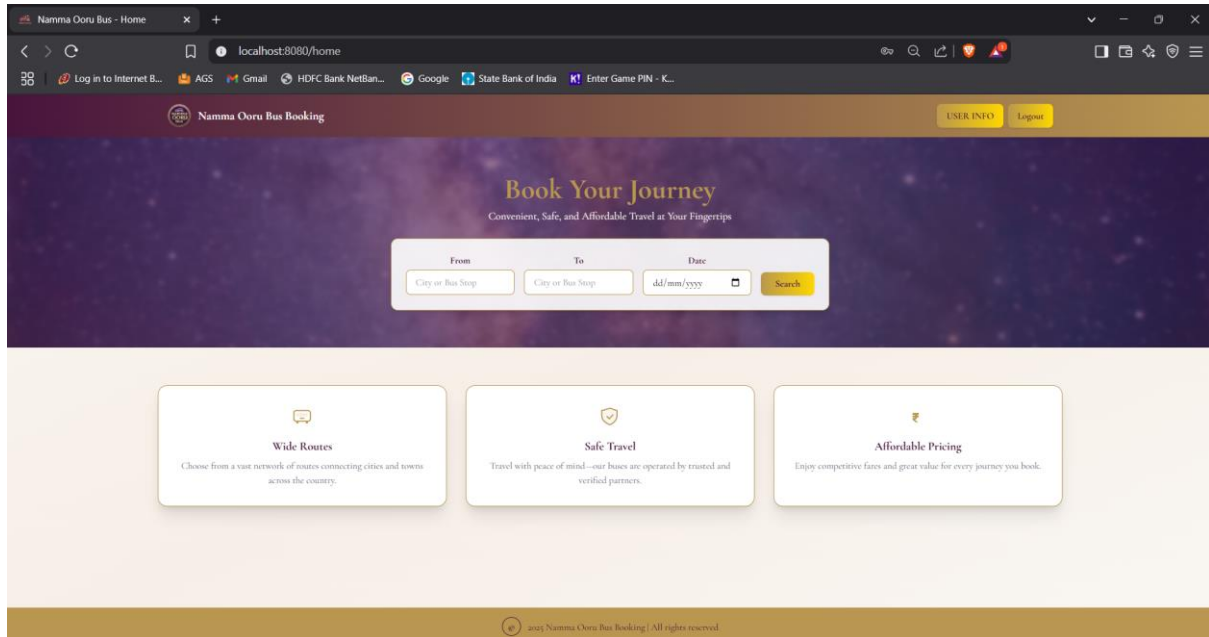
- **Endpoint:** /bookings/confirm
- **Method:** POST
- **Description:** Handles booking form submission, saves the booking, and generates a PDF ticket in the response.



HomeController.java

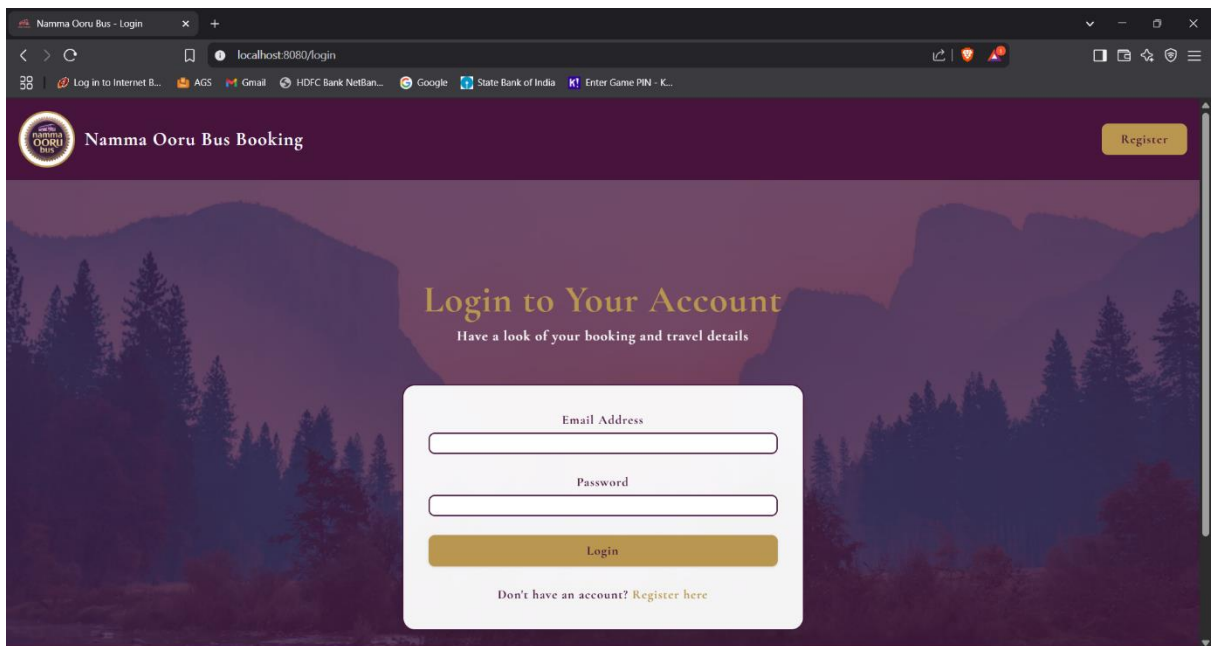
1. Home Page

- **Endpoint:** /
- **Method:** GET
- **Description:** Displays the home page.



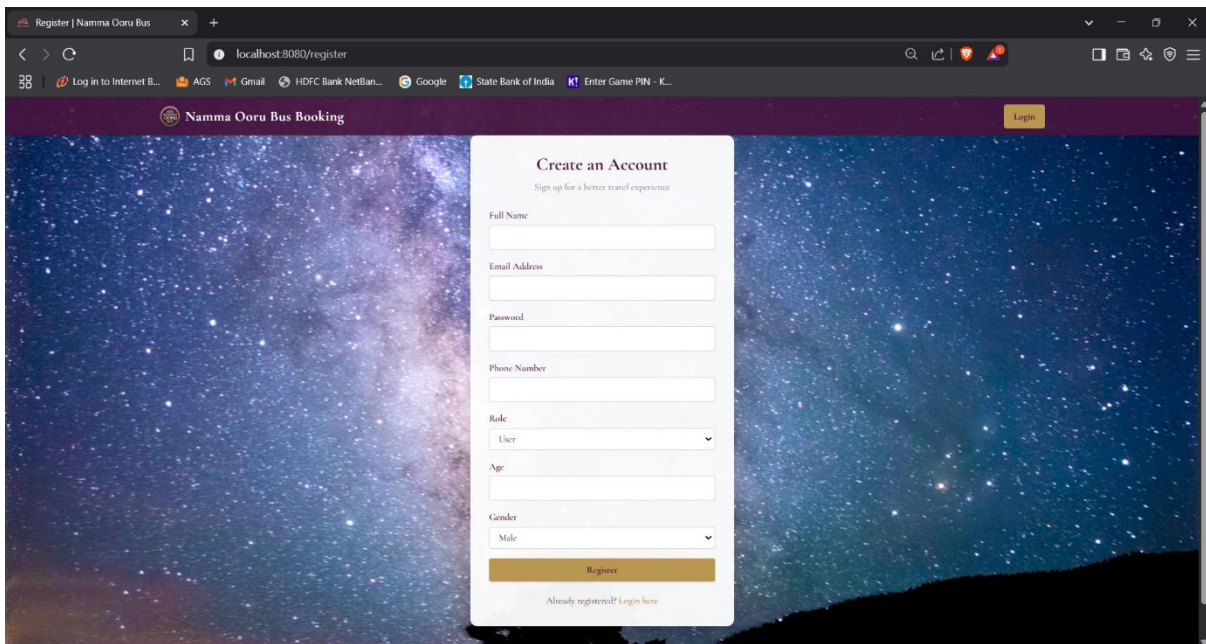
2. Login Page

- **Endpoint:** /login
- **Method:** GET
- **Description:** Shows the login page



3. Register Page

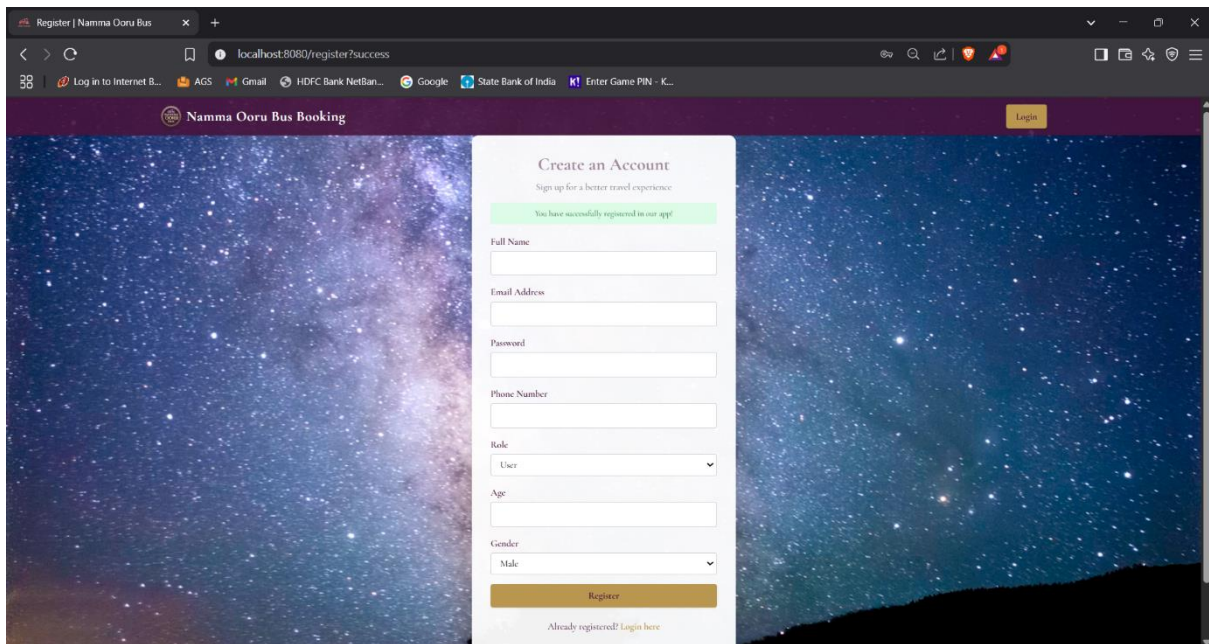
- **Endpoint:** /register
- **Method:** GET
- **Description:** Shows the user registration form.



The screenshot shows a web browser window with the URL `localhost:8080/register`. The page features a dark purple header with the text "Namma Ooru Bus Booking" and a "Login" button. The background is a vibrant image of the Milky Way galaxy. Centered on the page is a white "Create an Account" form with the subtitle "Sign up for a better travel experience". The form contains the following fields: "Full Name", "Email Address", "Password", "Phone Number", "Role" (a dropdown menu with "User" selected), "Age", and "Gender" (a dropdown menu with "Male" selected). At the bottom of the form is a "Register" button and a link that says "Already registered? Login here".

4. Saving a user or admin

- **Endpoint:** /register/save
- **Method:** POST
- **Description:** Handles user registration form submission.



The screenshot shows a web browser window with the URL `localhost:8080/register?success`. The page title is "Register | Namma Ooru Bus". The browser's address bar shows the URL, and the page has a dark purple header with the "Namma Ooru Bus Booking" logo and a "Login" button. The main content area features a "Create an Account" form with a green success message: "You have successfully registered in our app!". The form fields include "Full Name", "Email Address", "Password", "Phone Number", "Role" (a dropdown menu with "User" selected), "Age", and "Gender" (a dropdown menu with "Male" selected). A "Register" button is at the bottom of the form, and a link "Already registered? Login here" is below it. The background of the page is a starry night sky.

Register | Namma Ooru Bus

localhost:8080/register?success

Namma Ooru Bus Booking

Login

Create an Account

Sign up for a better travel experience

You have successfully registered in our app!

Full Name

Email Address

Password

Phone Number

Role

User

Age

Gender

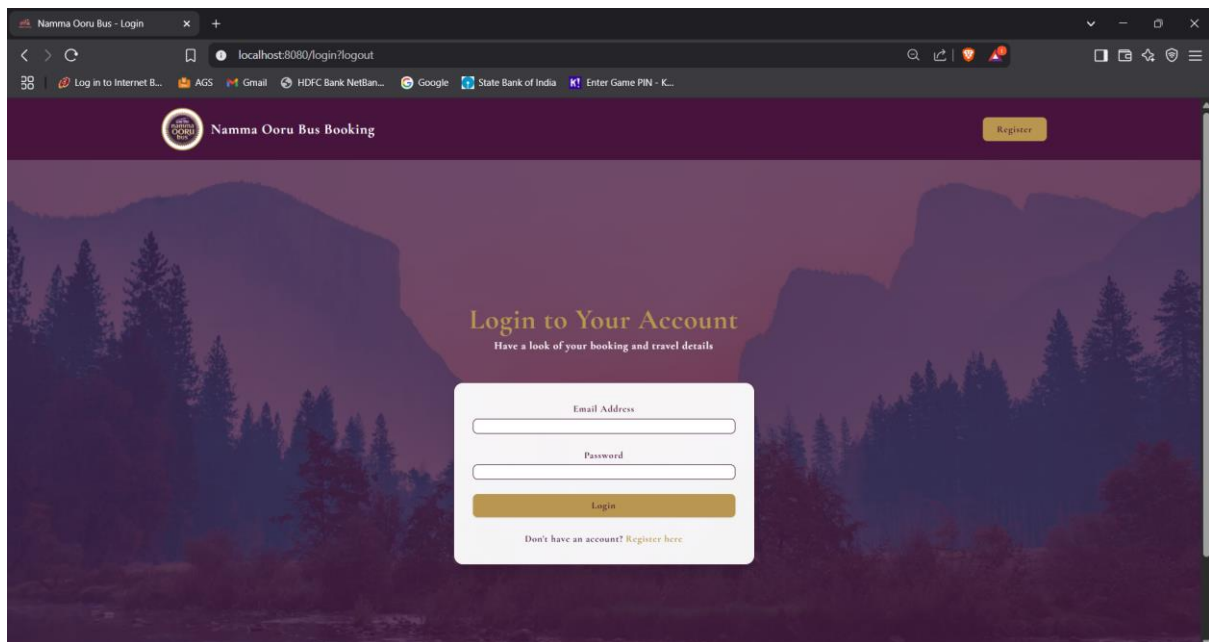
Male

Register

Already registered? Login here

5. Logging out

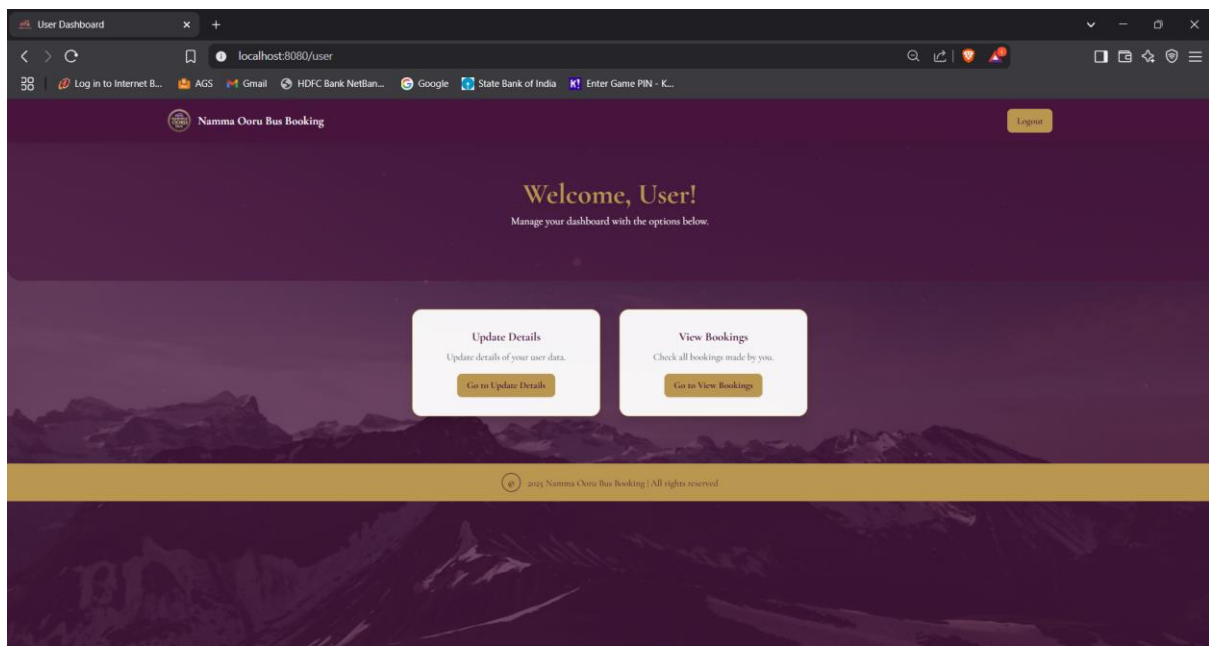
- **Endpoint:** /logout
- **Method:** GET
- **Description:** Logs out the current user.



UserController.java

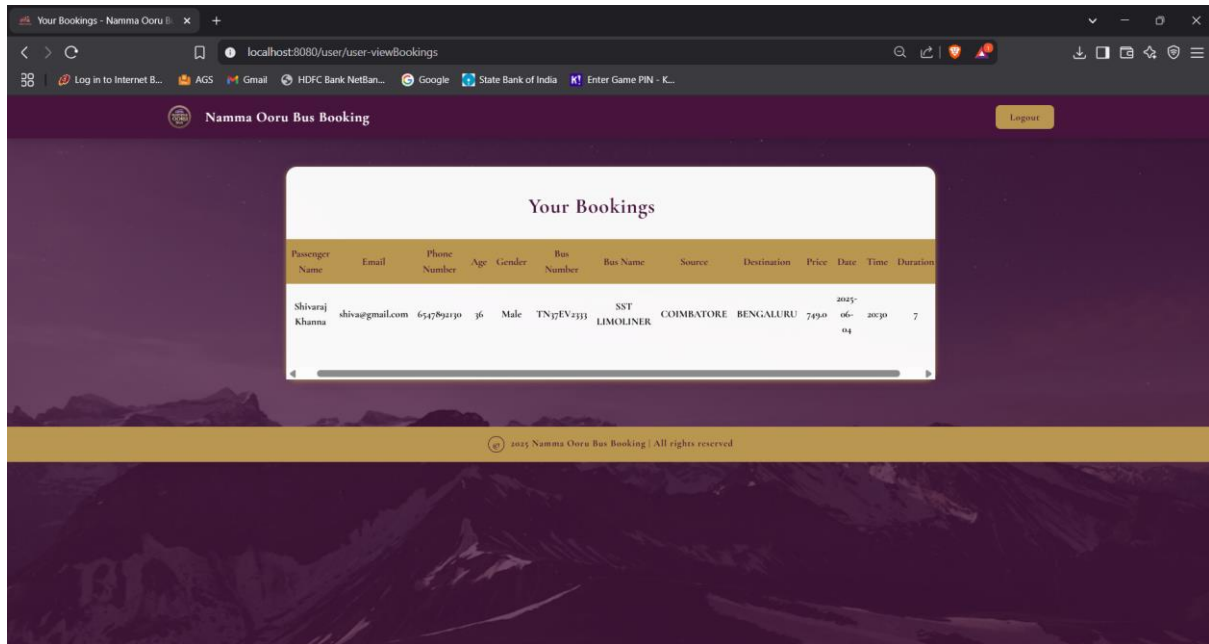
1. User Profile

- **Endpoint:** /user
- **Method:** GET
- **Description:** Displays the user dashboard with profile information.



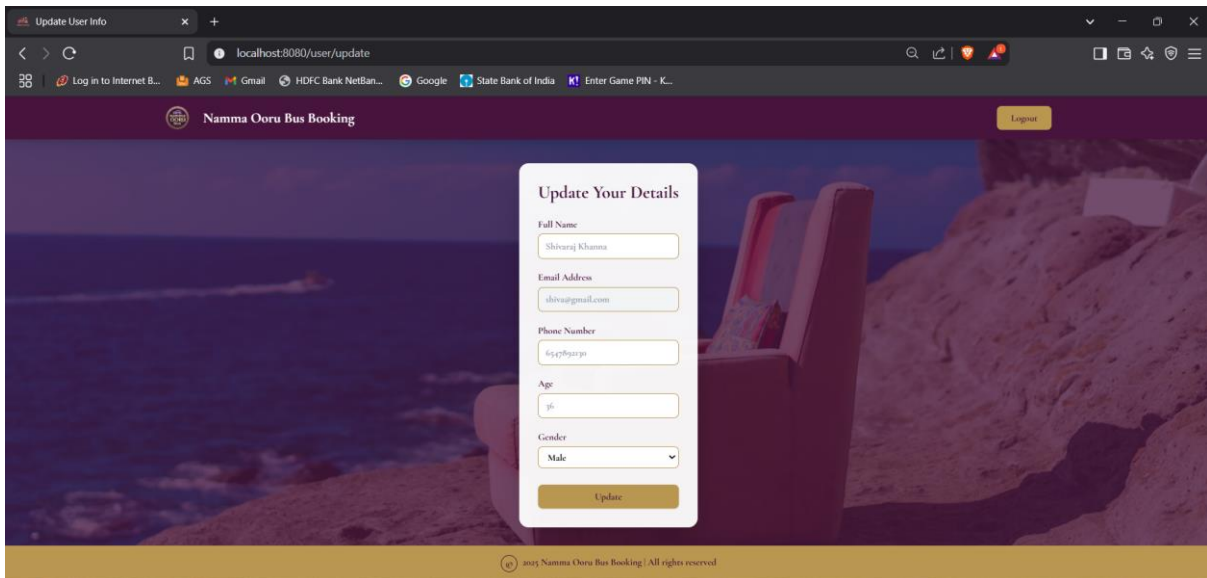
2. User's Booking List

- **Endpoint:** /user/user-viewBookings
- **Method:** GET
- **Description:** Shows all bookings for the currently logged-in user.



3. Updating the user profile

- **Endpoint:** /user/update
- **Method:** GET
- **Description:** Displays the form to update user details.



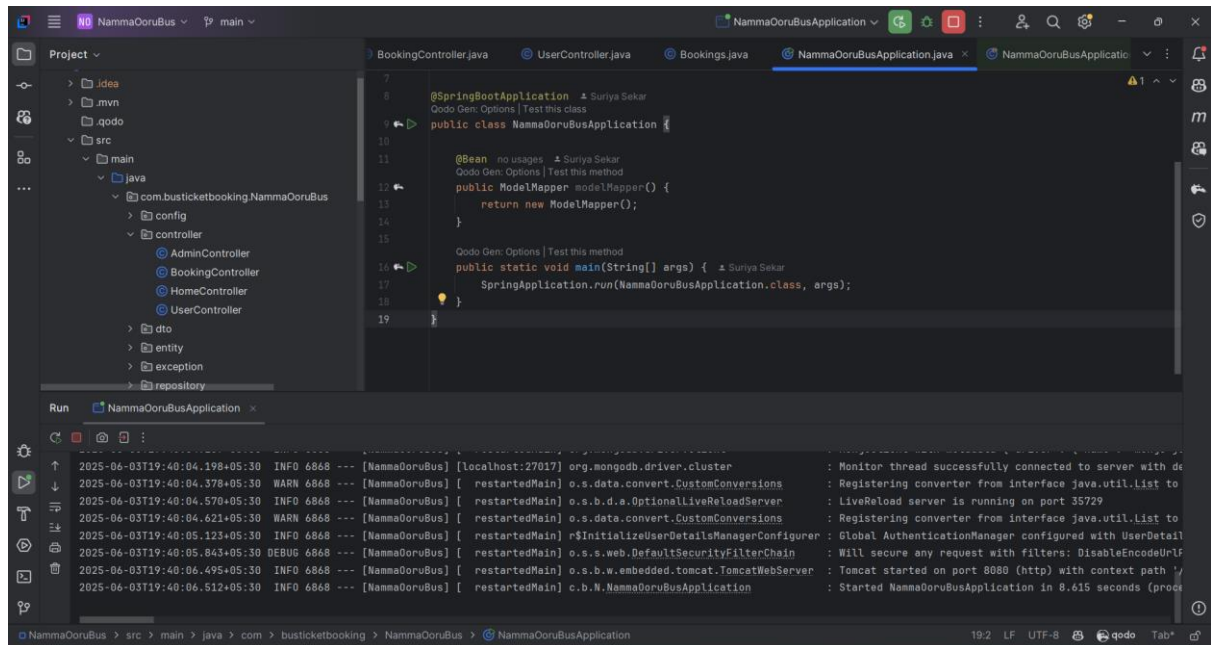
The screenshot shows a web browser window with the URL `localhost:8080/user/update`. The page features a dark purple header with the "Namma Ooru Bus Booking" logo and a "Logout" button. The main content area has a background image of a bus on a road. A white modal form titled "Update Your Details" is centered on the screen. The form contains the following fields:

- Full Name:** A text input field with the value "Shivraj Khanna".
- Email Address:** A text input field with the value "shivaj@gmail.com".
- Phone Number:** A text input field with the value "6457843210".
- Age:** A text input field with the value "36".
- Gender:** A dropdown menu with "Male" selected.
- Update:** A yellow button at the bottom of the form.

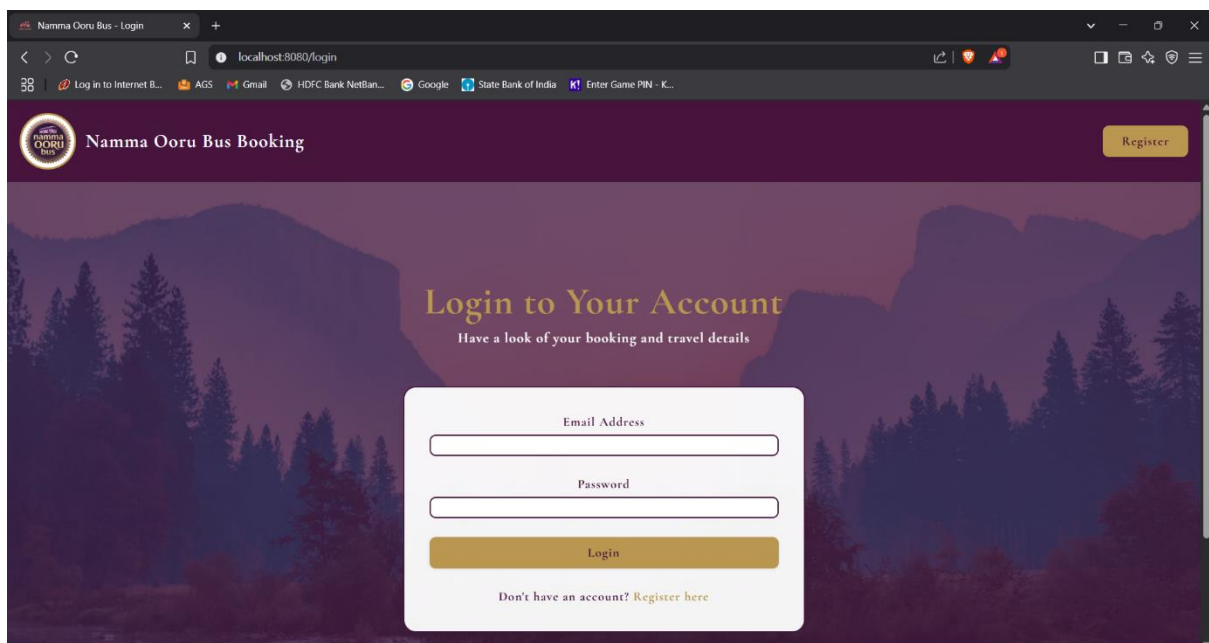
The footer of the page is dark purple and contains the text "© 2023 Namma Ooru Bus Booking | All rights reserved".

Running the Application

Initialization from IntelliJ IDEA



Login Page



Conclusion

In conclusion, the development of the Online Bus Ticket Booking Application, Namma Ooru Bus, represents a significant achievement in the realm of web-based transportation solutions. This project not only showcases the integration of modern technologies but also addresses the growing need for efficient and user-friendly travel booking systems.

Project Overview

Namma Ooru Bus is built using a robust technology stack that includes Spring Boot for the backend, Thymeleaf for the frontend, MongoDB for database management, and Tailwind CSS for styling. This combination ensures a seamless user experience, allowing customers to easily navigate the application and book their bus tickets with minimal effort. The choice of these technologies reflects a commitment to leveraging industry standards and best practices in software development.

User -Centric Design

The application is designed with the user in mind, emphasizing simplicity and accessibility. The intuitive interface allows users to search for buses, view schedules, and complete bookings in just a few clicks. By prioritizing user experience, Namma Ooru Bus aims to reduce the friction often associated with online ticket purchasing, thereby encouraging more travelers to utilize public transportation.

Technical Achievements

Throughout the development process, several technical challenges were encountered and successfully addressed. The integration of MongoDB as a NoSQL database allows for flexible data storage and retrieval, which is essential for handling the dynamic nature of bus schedules and user bookings. Additionally, the use of Spring Boot facilitates rapid development and deployment, ensuring that the application can be updated and maintained efficiently.

Future Enhancements

Looking ahead, there are numerous opportunities for enhancing the Namma Ooru Bus application. Future iterations could include features such as real-time bus tracking, user account management, and personalized travel recommendations based on user preferences. Implementing these features would not only improve the functionality of the application but also enhance user engagement and satisfaction.

Community and Feedback

The project encourages community involvement and feedback, which are vital for continuous improvement. By sharing the documentation and inviting input from users and developers alike, Namma Ooru Bus aims to foster a collaborative environment that can lead to innovative solutions and enhancements. This open approach not only benefits the application but also contributes to the broader developer community.

Final Thoughts

In summary, the Namma Ooru Bus Online Bus Ticket Booking Application stands as a testament to the potential of technology in transforming everyday tasks into streamlined processes. As public transportation continues to evolve, applications like Namma Ooru Bus will play a crucial role in making travel more accessible and efficient for everyone. The journey does not end here; it is merely the beginning of a larger vision to revolutionize the way we think about and utilize public transport. With ongoing development and community support, Namma Ooru Bus is poised to make a lasting impact in the transportation sector.

