Group-03

Online Bus Booking System

TEAM :

- ❖ Pudu Venkata Subba Reddy(**TL)**
- ❖ Somineni Lavanya
- ❖ Mayoor N K
- ❖ B R Rachitha Jain
- ❖ Pradeep Raj S
- ❖ Roshan Thomas
- ❖ Sreeramoju Sathwik
- ❖ Md Talib Alam
- ❖ Manike Anusha
- ❖ Jadhav Dinesh

CONTENTS :

1. Introduction

- In this project you can see the implementation of Microservice architecture.
- Here we identified the services from the UI and developed the relevant back-end logic.
- We implemented patterns and achieved load balancing.

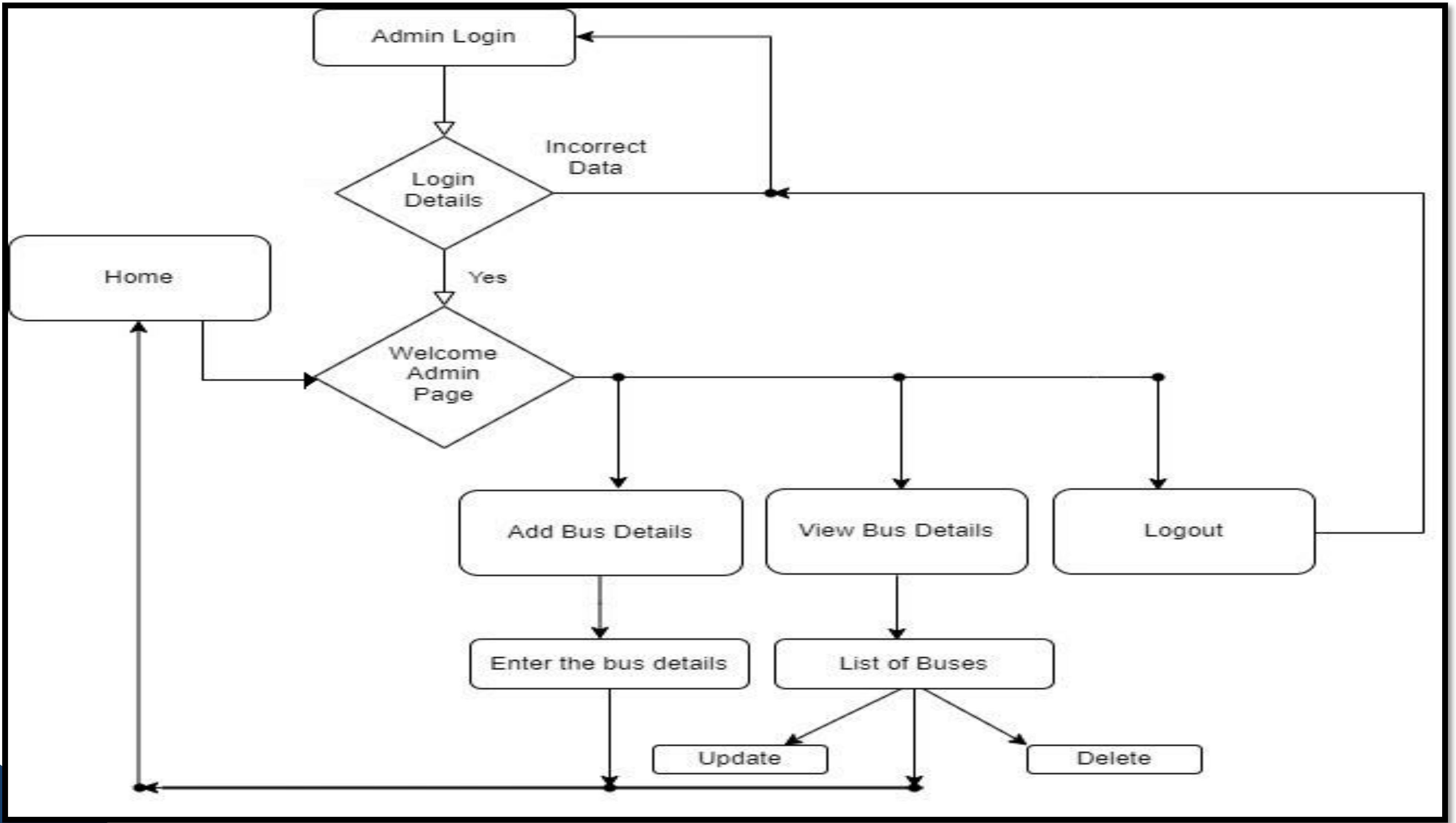
2. Objective

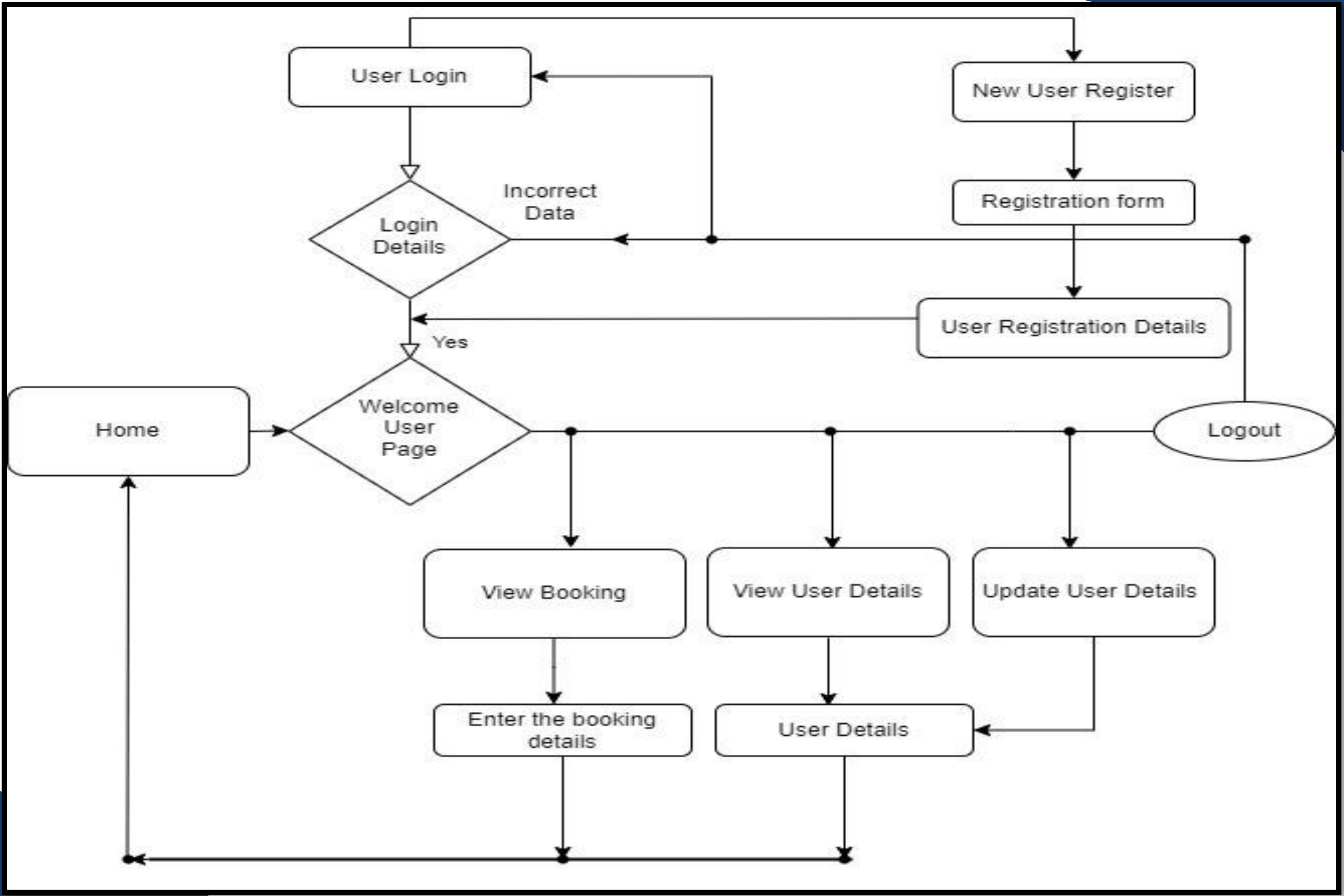
- To Create microservices to the bus booking system to handle the backend logic.
- To implement basic patterns of Microservice Architecture.

3. Technologies used :

- Spring Boot
- Spring Data Jpa
- MySQL
- Docker
- RabbitMQ
- GitHub

4. Flow-diagrams:




User Login
New User Register
Registration form
Incorrect Data
Login Details
User Registration Details
Yes
Home
Welcome User Page
Logout
View Booking
View User Details
Update User Details
Enter the booking details
User Details

5. Patterns:

Spring Cloud Config

- Spring Cloud Config Server is a central hub for managing configuration properties for distributed applications and microservices.
- Spring Cloud Config Server integrates with version control systems like Git and provides a versioned history of configuration changes.
- Spring Cloud Config Server offers security features to protect sensitive configuration data, including authentication and authorization mechanisms.
- Config Server can be deployed in a high availability configuration to ensure resilience and availability.

Spring Cloud-Netflix Eureka

- Eureka is the Netflix Service Discovery Server and Client.
- Eureka is a service registry and discovery server that allows microservices to register themselves and discover other services within the same system.
- The server can be configured and deployed to be highly functional, with each server copying the state of the registered services to the others.
- Eureka enables client-side load balancing, allowing services to locate and call other services without knowing their exact locations, improving system scalability.

Open-feign

- Declarative API Requests: Feign simplifies HTTP requests by defining them as Java interfaces with annotated methods.
- Integration with Service Discovery: Feign can seamlessly integrate with service discovery tools like Eureka or Consul.
- Load Balancing: It works with load balancing solutions (e.g., Ribbon) to evenly distribute requests among service instances.
- Fallback and Error Handling: Feign provides mechanisms for handling errors and defining fallback methods for graceful degradation in case of service failures.

Resilience 4J

- ➢ Resilience4j is a lightweight fault tolerance library that provides a variety of fault tolerance and stability patterns to a web application.
- ➢ Resilience4j provides higher-order functions (decorators) to enhance any functional interface, lambda expression or method reference with a Circuit Breaker, Rate Limiter, Retry or Bulkhead.
- ➢ With Resilience4j you don't have to go all-in, you can pick what you need.
- ➢ Resilience4j provides several core modules and add-on modules

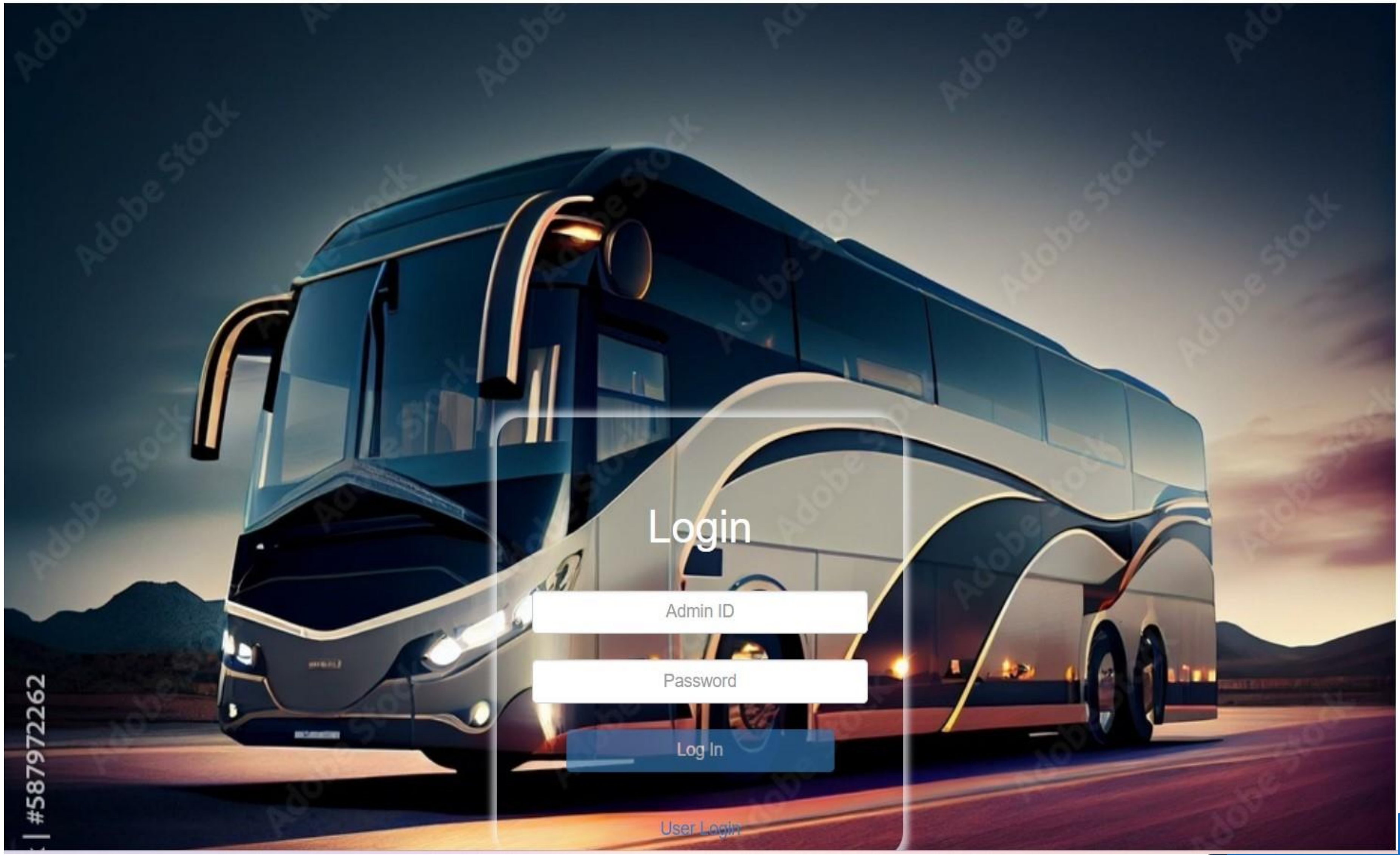
Retry

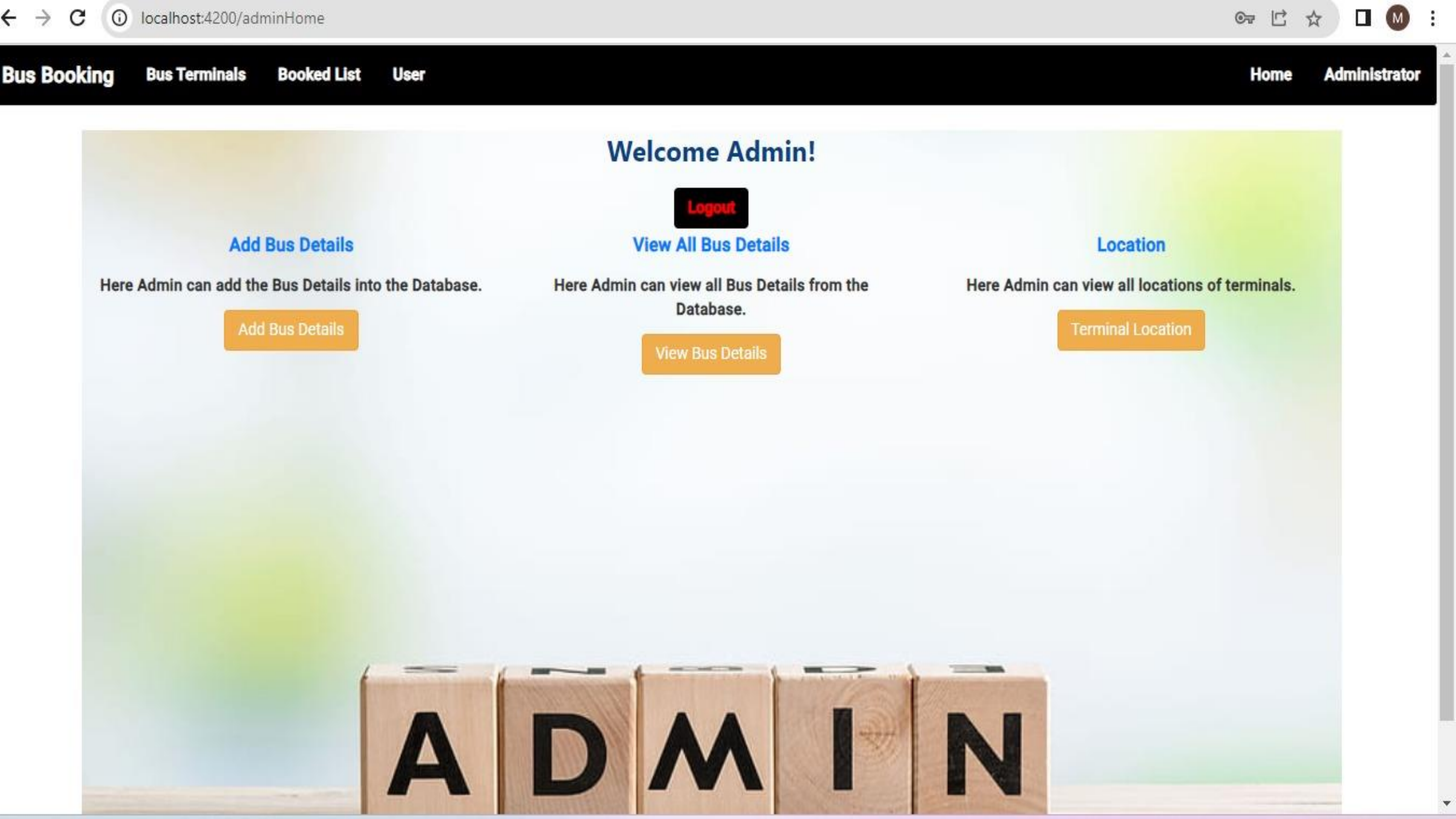
- Automated Retry: Resilience4j's Retry module automates retries for handling transient failures in microservices.
- Declarative or Programmatic: Retry behavior can be configured declaratively with annotations or programmatically in code.
- Exponential Backoff: It supports strategies like exponential backoff to reduce the load on failing services.
- Customizable Control: Highly customizable with options for defining retry count, intervals, and conditions for precise control over retry logic in microservices.

Sleuth and Zipkin

- Distributed Tracing: Sleuth is a Java library for distributed tracing that helps monitor and analyse requests as they traverse through microservices.
- Unique Trace IDs: Sleuth generates and propagates unique trace and span IDs, allowing you to trace requests across multiple microservices and gather insights into performance bottlenecks.
- Distributed Tracing Tool: Zipkin is an open-source distributed tracing system that helps monitor and troubleshoot microservices-based applications.
- Integration with Other Tools: Zipkin can integrate with other observability and monitoring tools, allowing you to correlate trace data with metrics and logs for comprehensive insights into your microservices ecosystem.

6. Project Requirement




localhost:4200/adminHome

Bus Booking Bus Terminals Booked List User Home Administrator

Welcome Admin!

Logout

Add Bus Details

Here Admin can add the Bus Details into the Database.

Add Bus Details

View All Bus Details

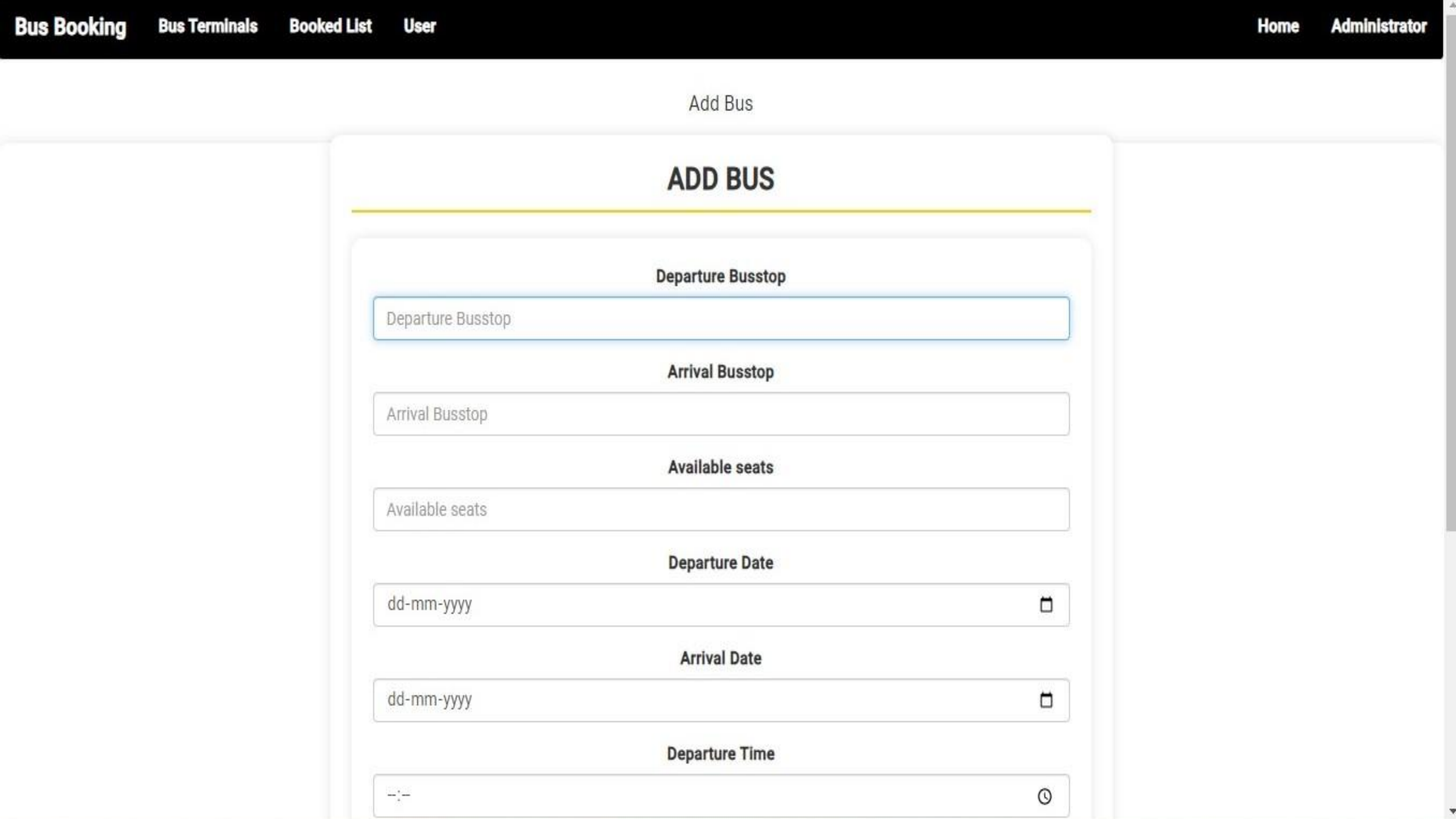
Here Admin can view all Bus Details from the Database.

View Bus Details

Location

Here Admin can view all locations of terminals.

Terminal Location



Add Bus

ADD BUS

Departure Busstop

Departure Busstop

Arrival Busstop

Arrival Busstop

Available seats

Available seats

Departure Date

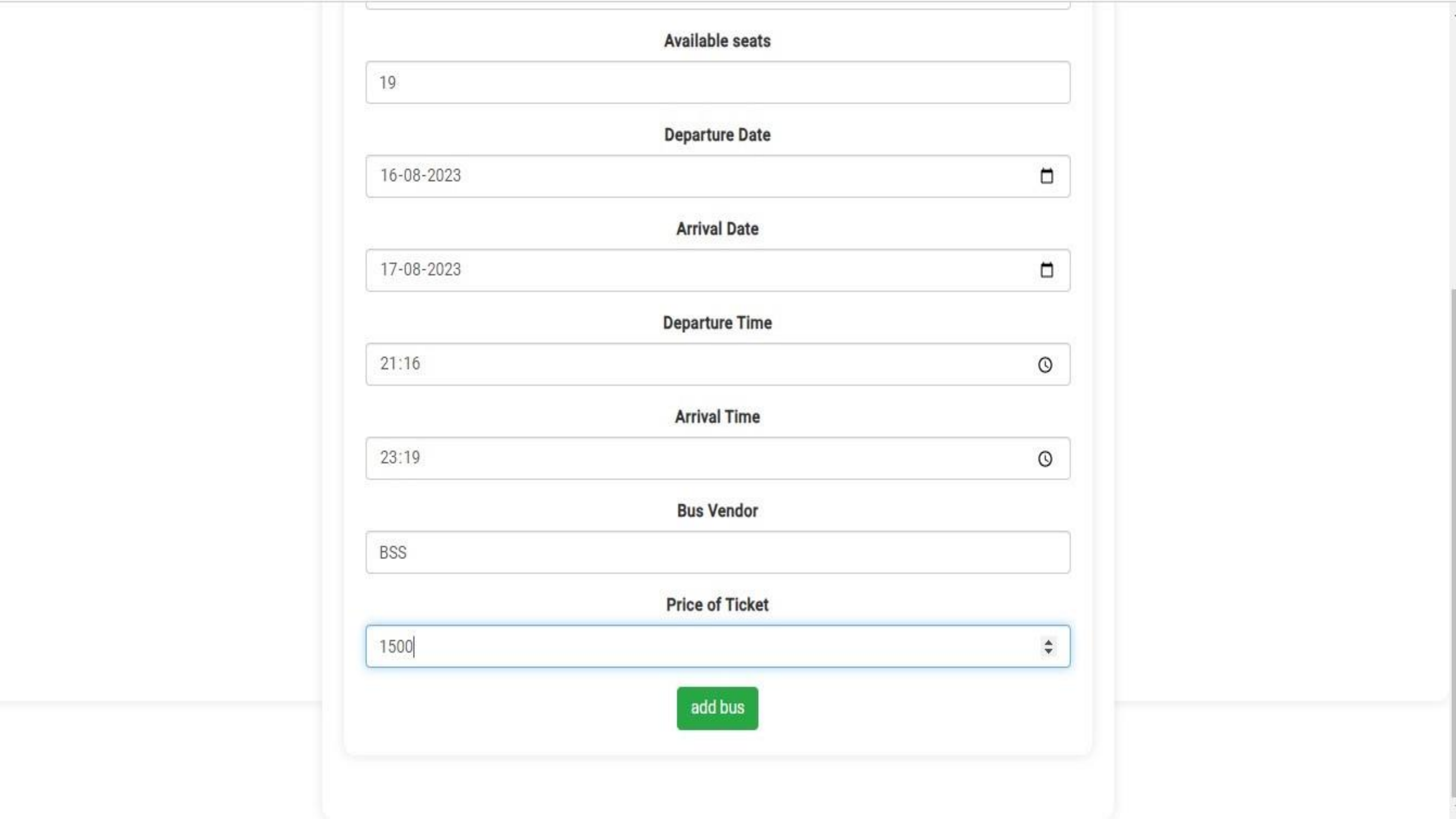
dd-mm-yyyy

Arrival Date

dd-mm-yyyy

Departure Time

--:--


Available seats

19

Departure Date

16-08-2023

Arrival Date

17-08-2023

Departure Time

21:16

Arrival Time

23:19

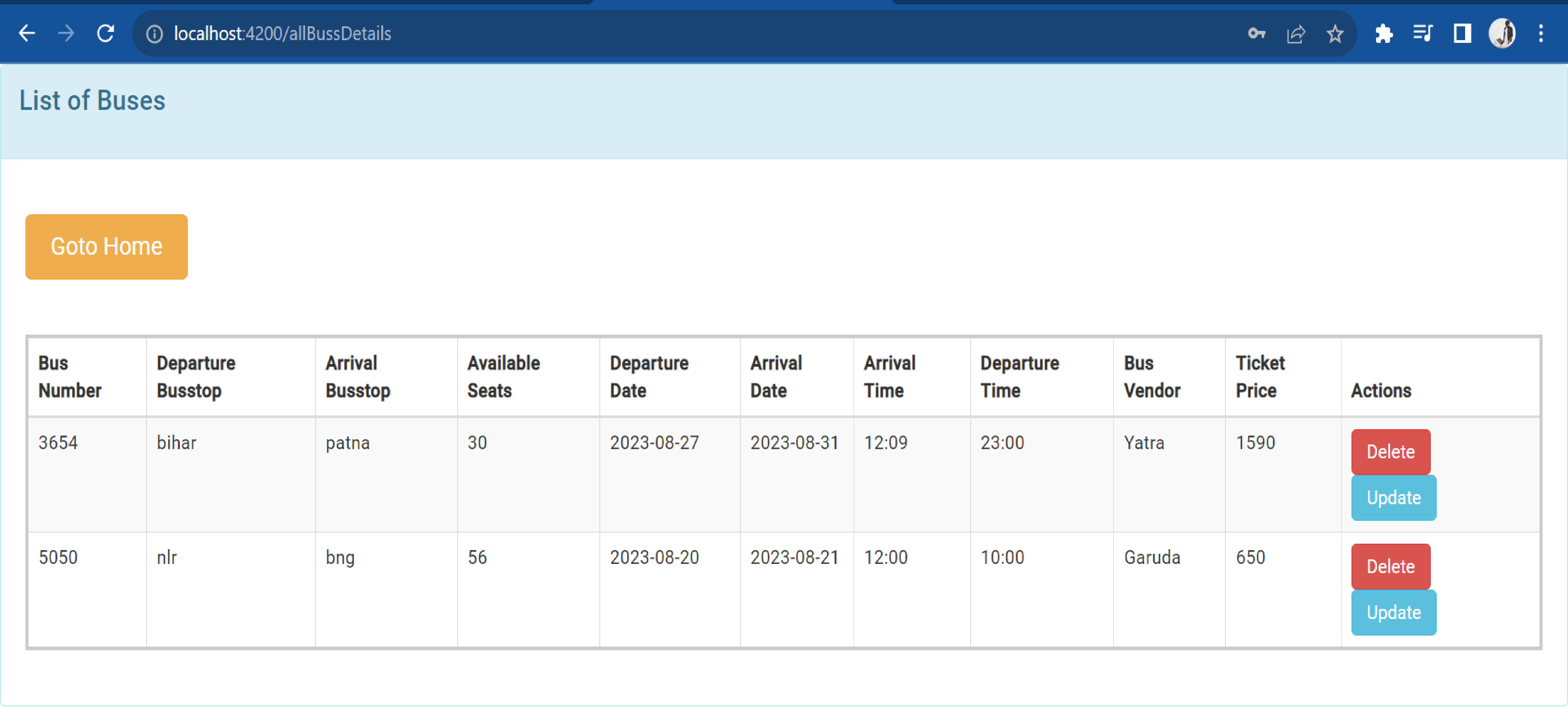
Bus Vendor

BSS

Price of Ticket

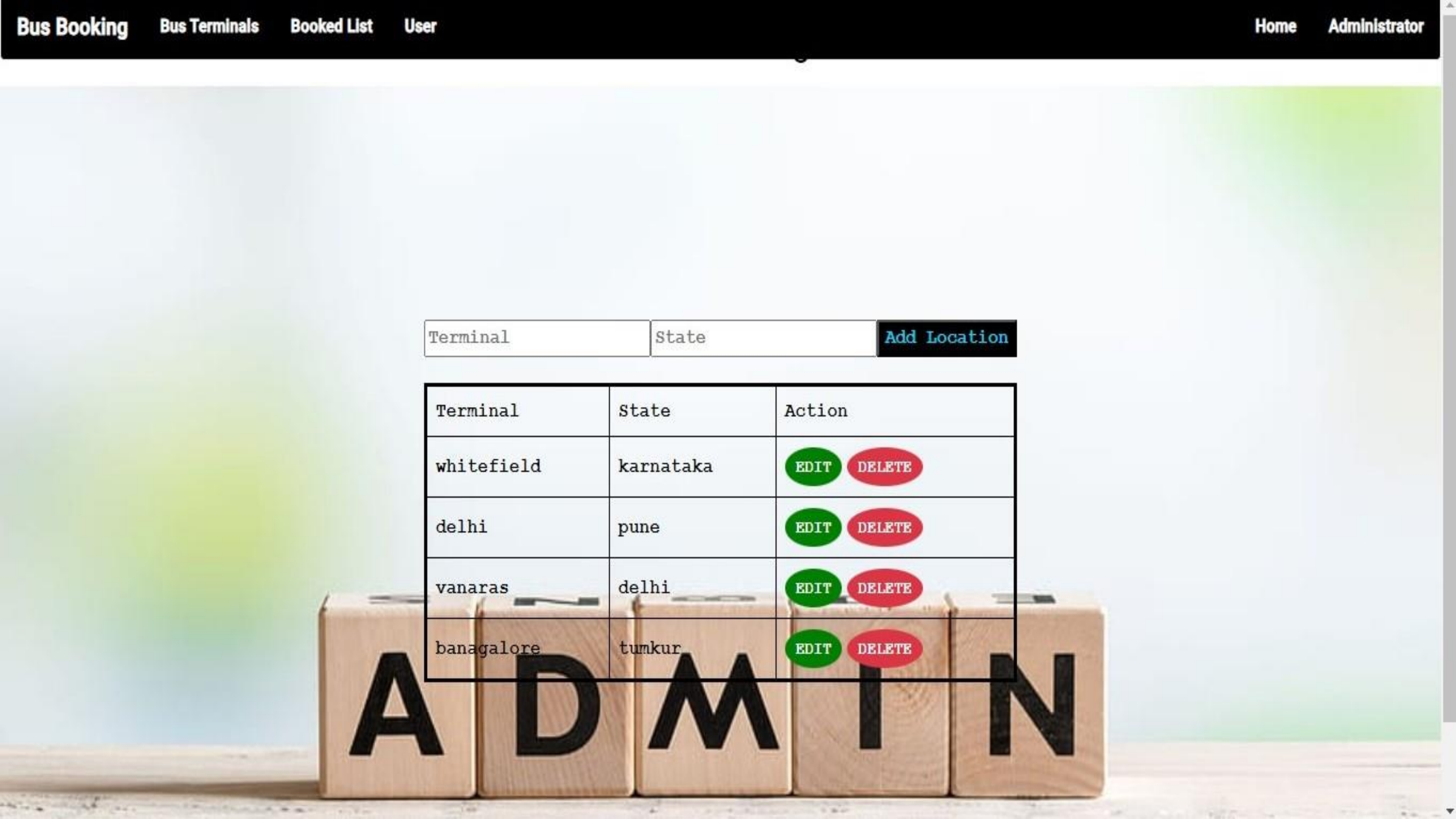
1500

add bus

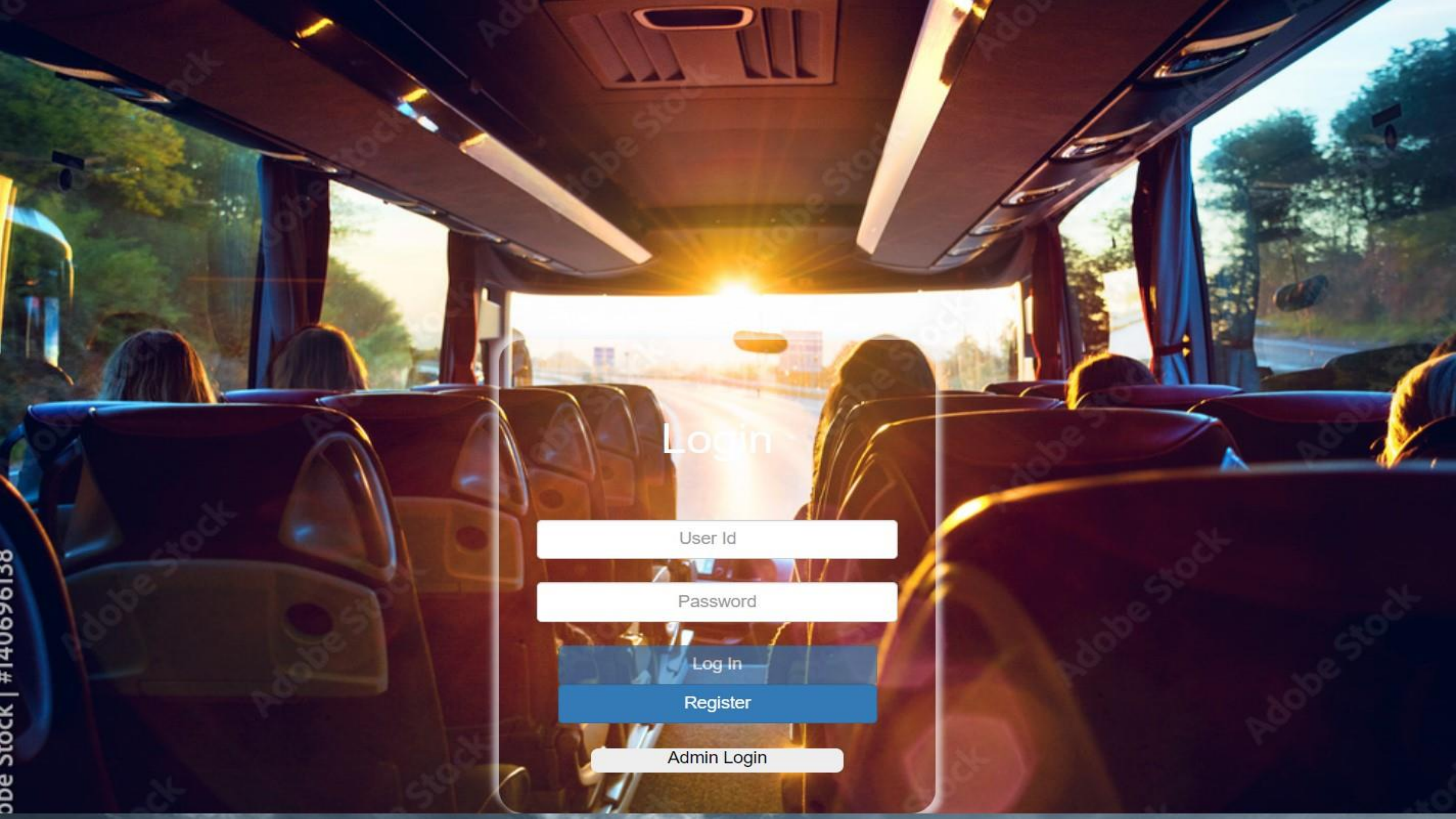


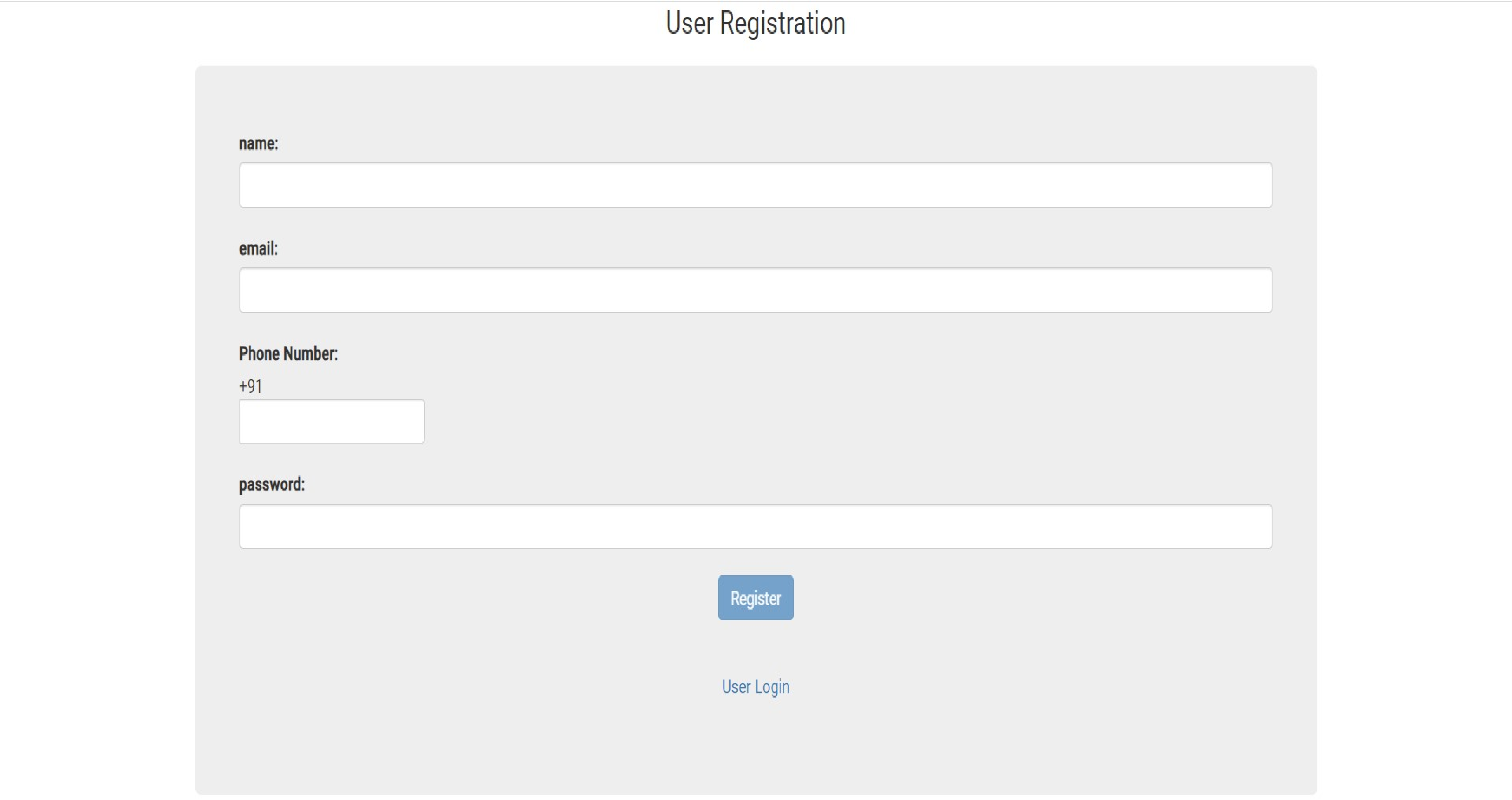
List of Buses

Bus Number	Departure Busstop	Arrival Busstop	Available Seats	Departure Date	Arrival Date	Arrival Time	Departure Time	Bus Vendor	Ticket Price	Actions
3654	bihar	patna	30	2023-08-27	2023-08-31	12:09	23:00	Yatra	1590	Delete Update
5050	nlr	bng	56	2023-08-20	2023-08-21	12:00	10:00	Garuda	650	Delete Update



Terminal	State	Action
whitefield	karnataka	EDIT DELETE
delhi	pune	EDIT DELETE
vanaras	delhi	EDIT DELETE
banagalore	tumkur	EDIT DELETE


Login
User Id
Password
Log In
Register
Admin Login



User Registration

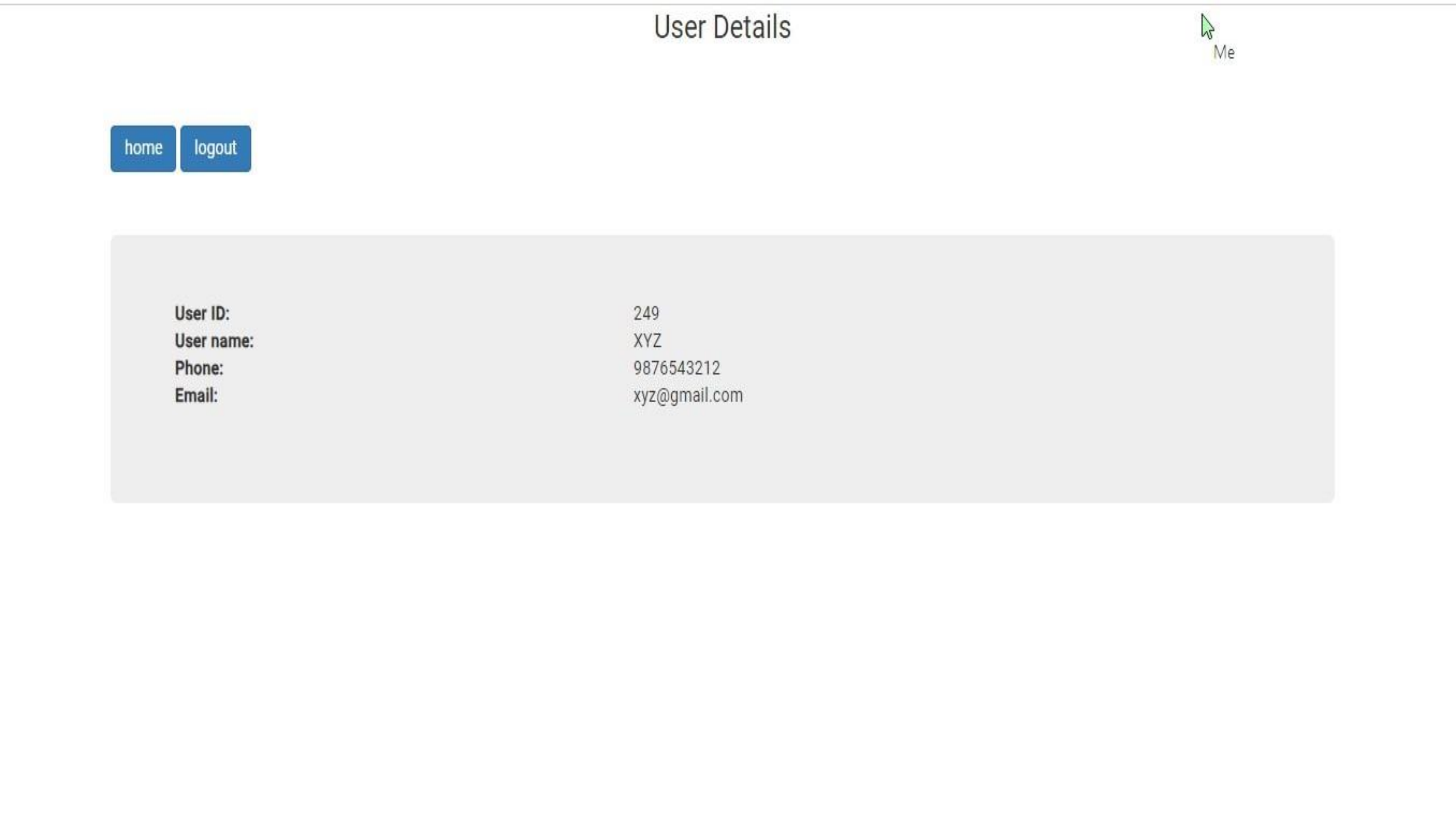
name:

email:

Phone Number:

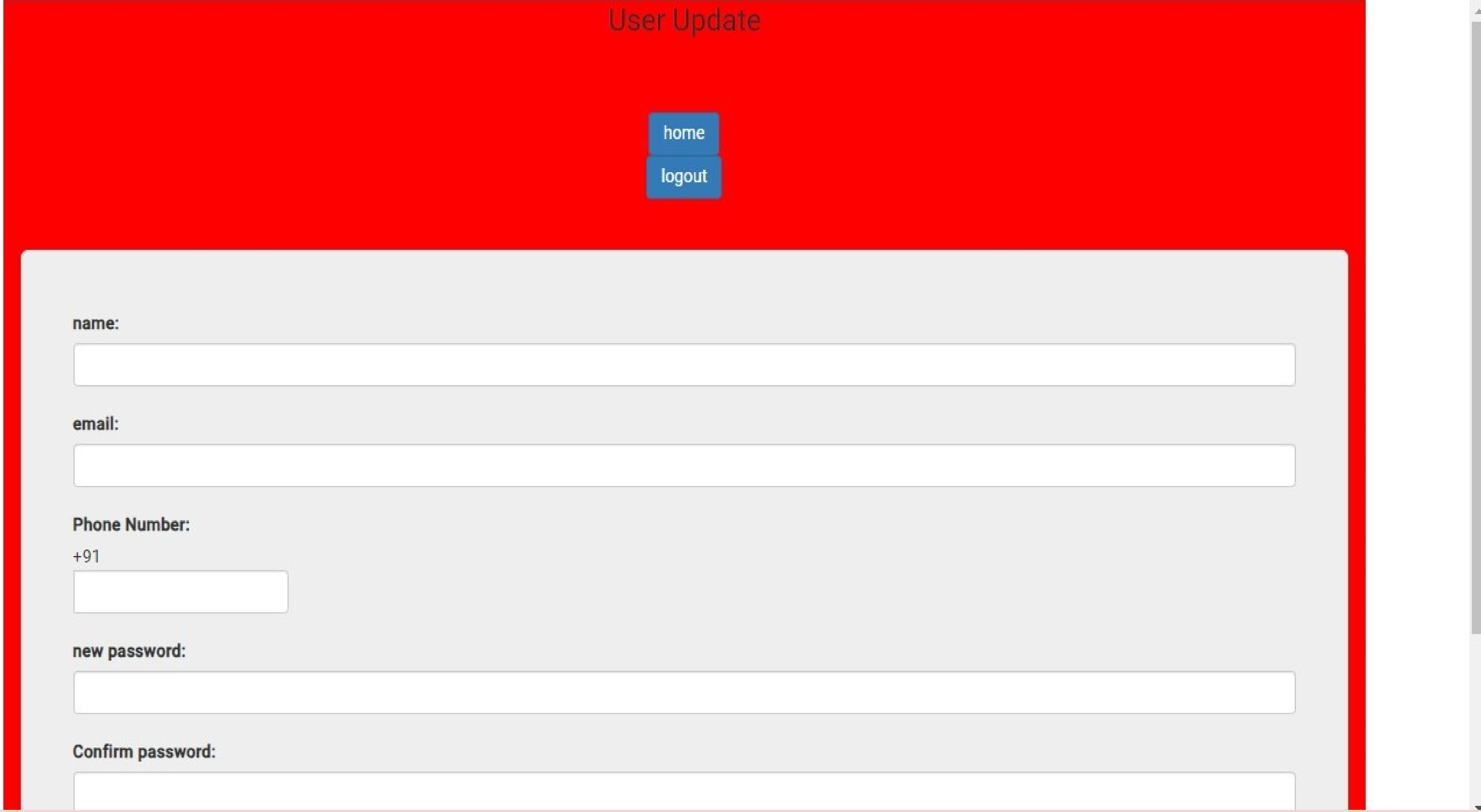
+91

password:



User Details

User ID:	249
User name:	XYZ
Phone:	9876543212
Email:	xyz@gmail.com


User Update

name:

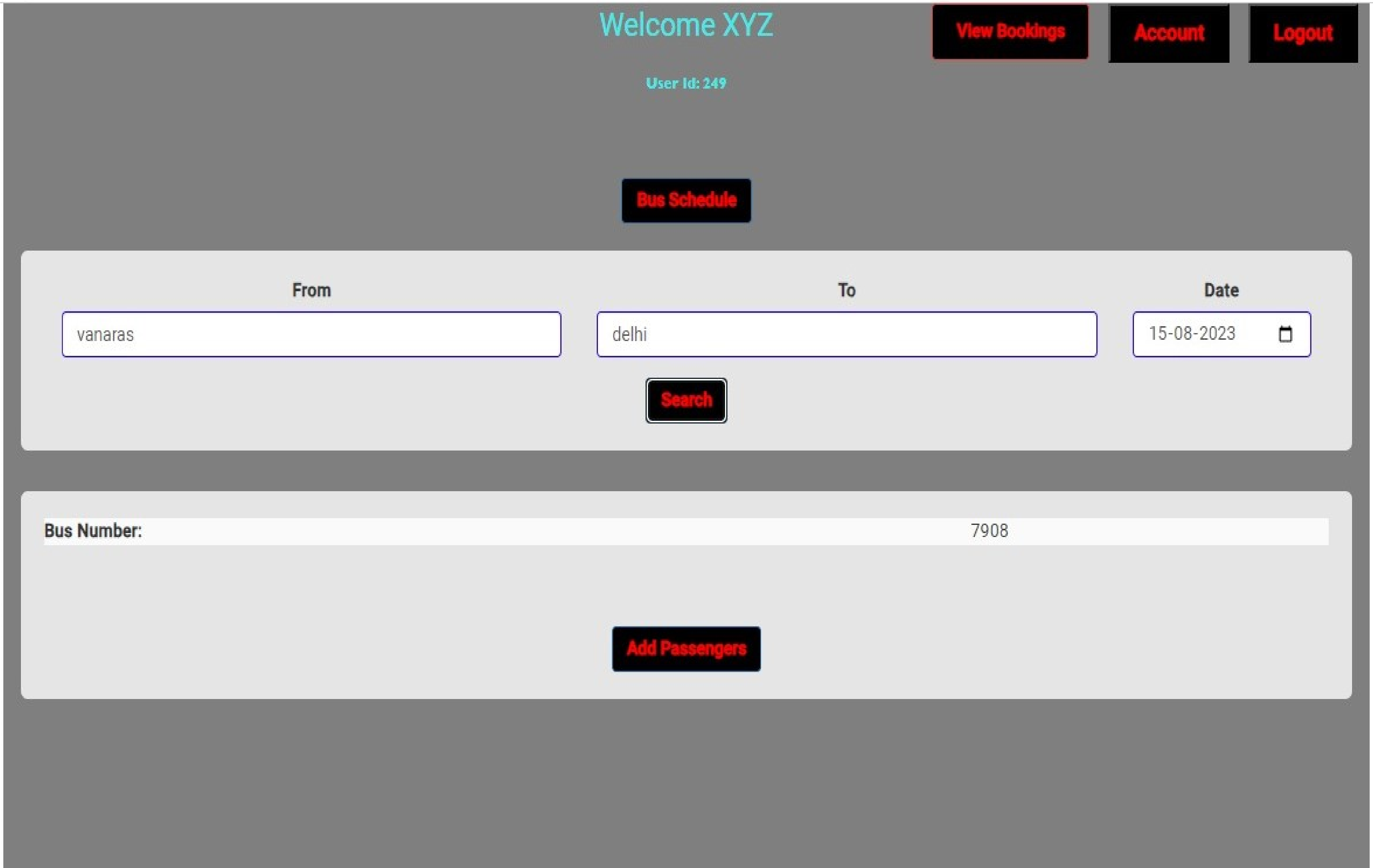
email:

Phone Number:

+91

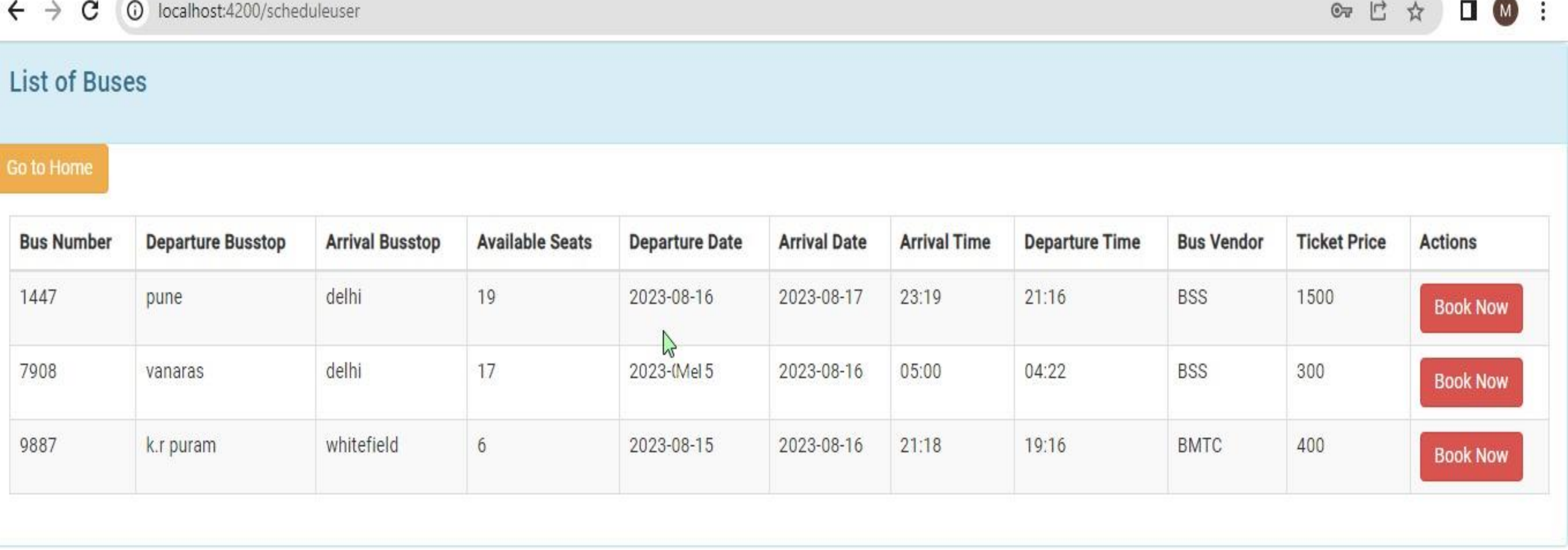
new password:

Confirm password:



From	To	Date
vanaras	delhi	15-08-2023

Bus Number:	7908

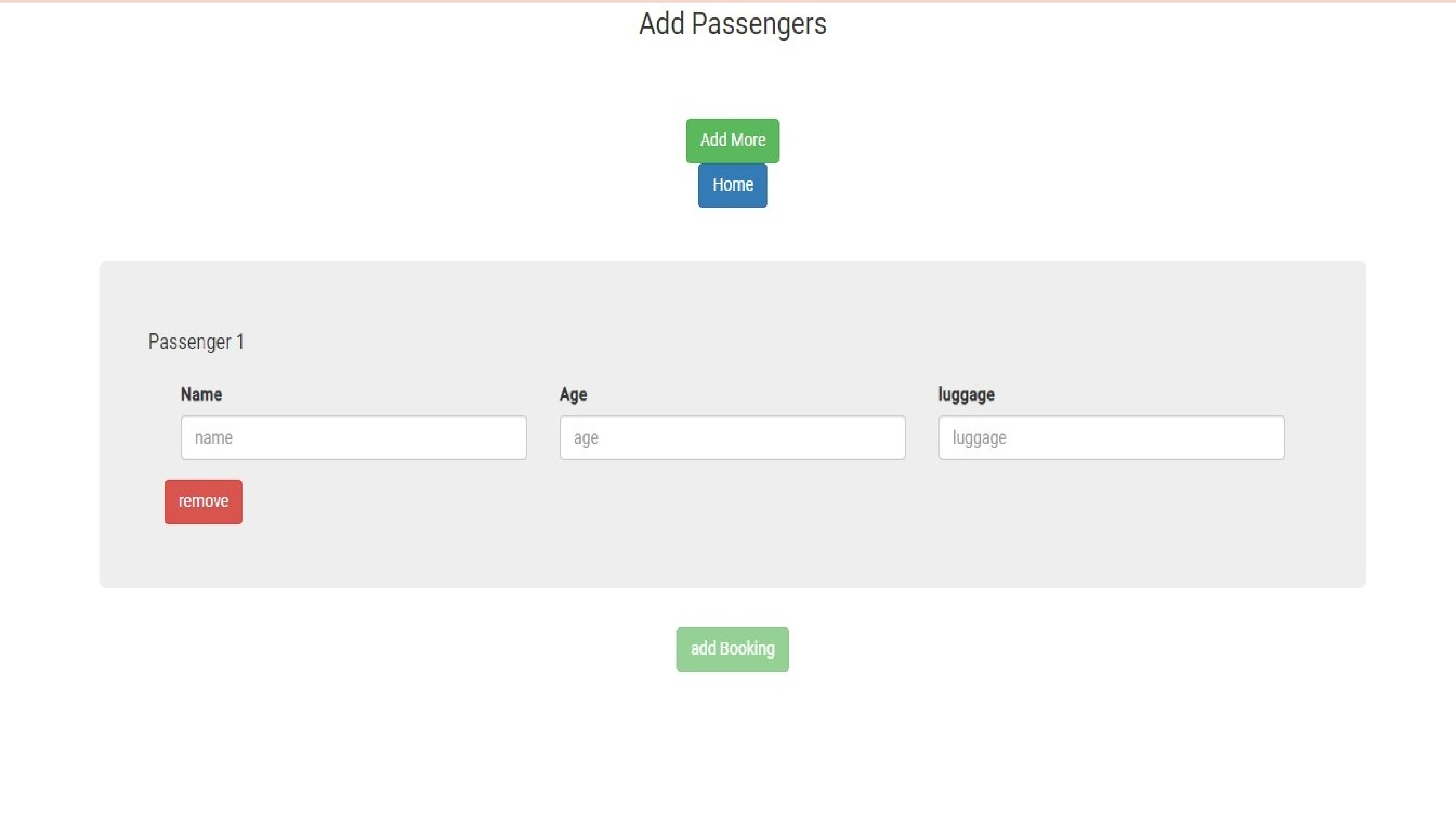


localhost:4200/scheduleuser

List of Buses

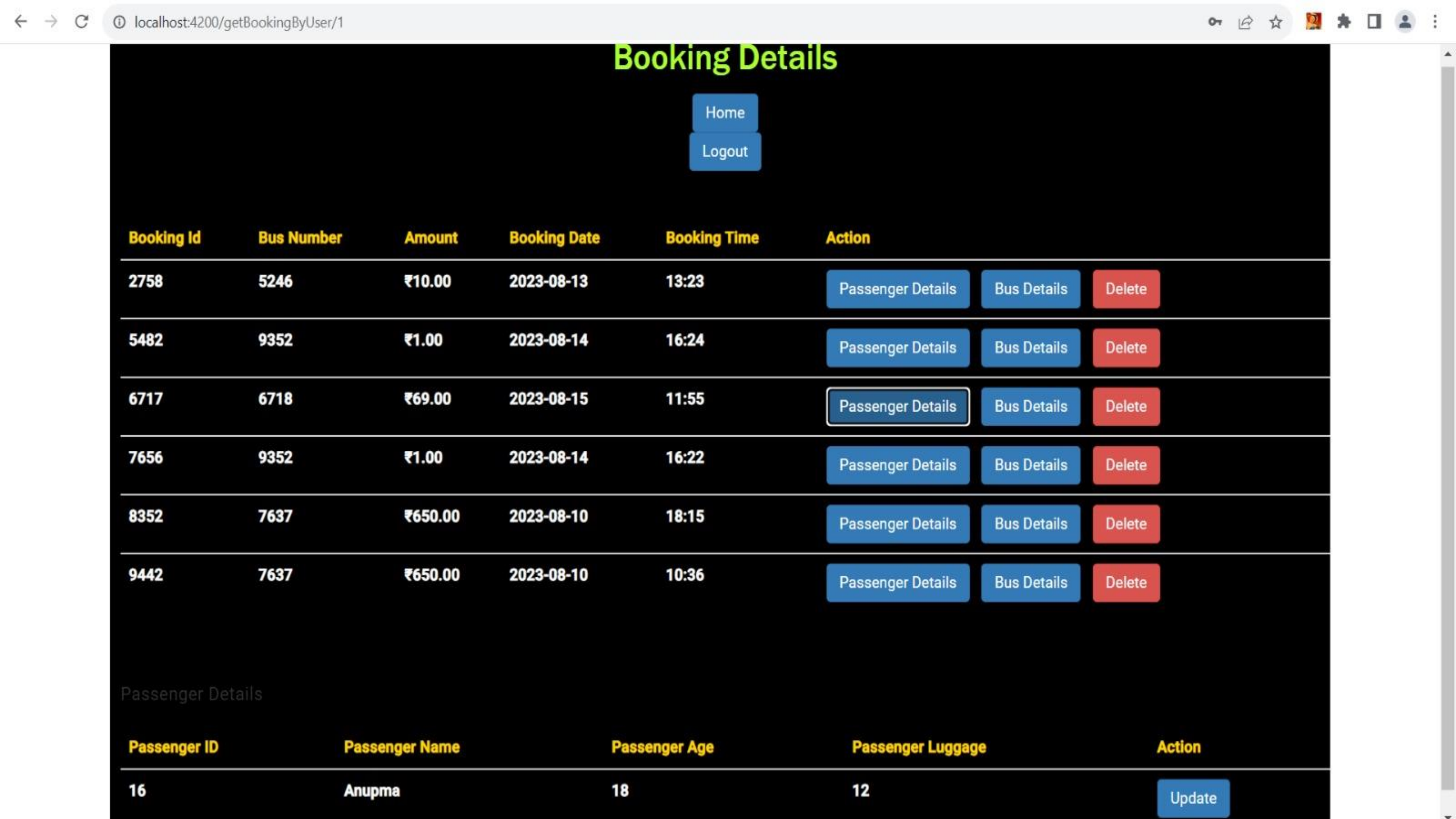
Go to Home

Bus Number	Departure Busstop	Arrival Busstop	Available Seats	Departure Date	Arrival Date	Arrival Time	Departure Time	Bus Vendor	Ticket Price	Actions
1447	pune	delhi	19	2023-08-16	2023-08-17	23:19	21:16	BSS	1500	Book Now
7908	vanaras	delhi	17	2023-08-15	2023-08-16	05:00	04:22	BSS	300	Book Now
9887	k.r puram	whitefield	6	2023-08-15	2023-08-16	21:18	19:16	BMTC	400	Book Now


Add Passengers

Passenger 1

Name	Age	luggage
name	age	luggage

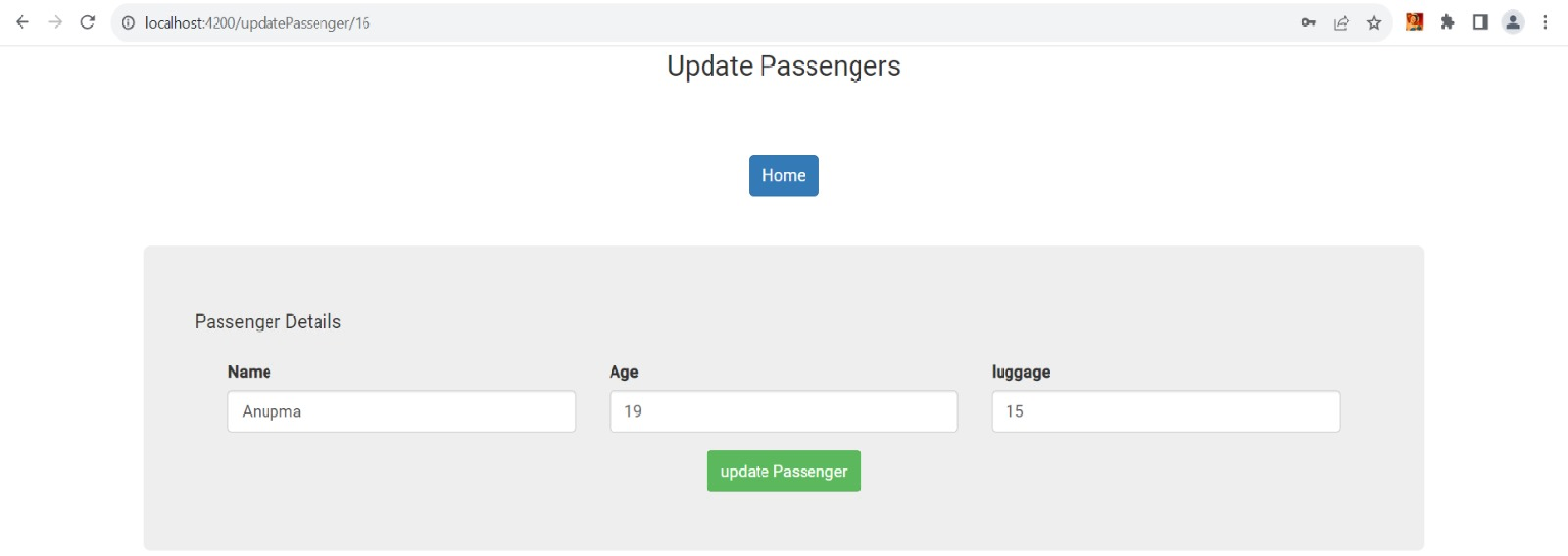


Booking Details

Booking Id	Bus Number	Amount	Booking Date	Booking Time	Action
2758	5246	₹10.00	2023-08-13	13:23	Passenger Details Bus Details Delete
5482	9352	₹1.00	2023-08-14	16:24	Passenger Details Bus Details Delete
6717	6718	₹69.00	2023-08-15	11:55	Passenger Details Bus Details Delete
7656	9352	₹1.00	2023-08-14	16:22	Passenger Details Bus Details Delete
8352	7637	₹650.00	2023-08-10	18:15	Passenger Details Bus Details Delete
9442	7637	₹650.00	2023-08-10	10:36	Passenger Details Bus Details Delete

Passenger Details

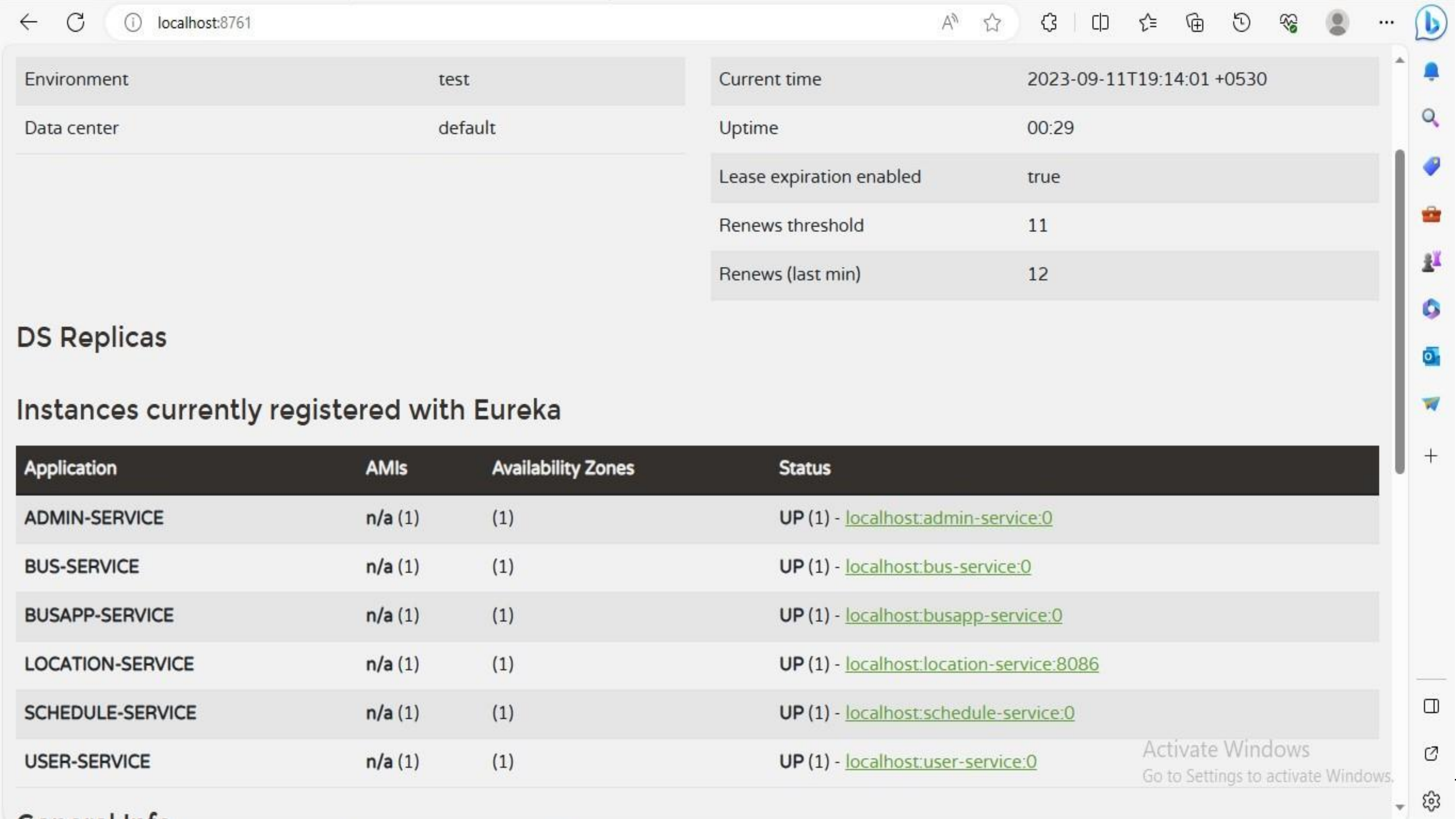
Passenger ID	Passenger Name	Passenger Age	Passenger Luggage	Action
16	Anupma	18	12	Update


localhost:4200/updatePassenger/16

Update Passengers

Passenger Details

Name	Age	luggage
Anupma	19	15



Environment	test
Data center	default

Current time	2023-09-11T19:14:01 +0530
Uptime	00:29
Lease expiration enabled	true
Renews threshold	11
Renews (last min)	12

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ADMIN-SERVICE	n/a (1)	(1)	UP (1) - localhost:admin-service:0
BUS-SERVICE	n/a (1)	(1)	UP (1) - localhost:bus-service:0
BUSAPP-SERVICE	n/a (1)	(1)	UP (1) - localhost:busapp-service:0
LOCATION-SERVICE	n/a (1)	(1)	UP (1) - localhost:location-service:8086
SCHEDULE-SERVICE	n/a (1)	(1)	UP (1) - localhost:schedule-service:0
USER-SERVICE	n/a (1)	(1)	UP (1) - localhost:user-service:0

7. Advantages of Microservice architecture:

- Scalability
- Flexibility
- Easy Integration
- Fault Isolation
- Resilience
- Faster Development
- Maintainability
- Faster to market

8. Conclusion:

•**Scalable and Adaptable**: Microservice architecture enables the online bus booking system to easily scale its services to accommodate growing user demand. It allows for the addition of new services and functionalities without disrupting the entire system.

•**Real-Time Responsiveness**: Microservices facilitate real-time information updates, ensuring that travelers always have access to the latest route details, seat availability, and booking confirmations. This enhances user experience and decision-making.

•**Efficient Operations**: By breaking down the application into smaller, specialized microservices, operational efficiency is improved. Each microservice can focus on specific tasks like scheduling, seat allocation, and payment processing, leading to streamlined administrative processes.

•**Adaptation to Future Technologies**: Microservice architecture positions the online bus booking system for future growth and technological integration. It can readily adopt new technologies and trends in the digital landscape, staying competitive and user-friendly.

THANK YOU