



UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

“Book On Rails”

Submitted by:

Shreeja Rajesh	PES1UG21CS564
Shreya Mishra	PES1UG21CS574
Sinchana	PES1UG21CS596
Siri M M	PES1UG21CS600

6th Semester J Section

Team Number : J6

Prof. Vinay Joshi
Assistant Professor

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Table of Contents

Table of Contents.....	2
Problem Statement.....	3
Major Features.....	3
Minor Features.....	3
Models.....	4
Use Case Diagram.....	4
Class Diagram.....	4
State Diagrams:.....	5
Activity Diagram.....	7
Architecture Patterns.....	9
Design Principles.....	13
Design Patterns.....	17
Github.....	21
Individual Contributions :	22
Screenshots.....	23
Database Before Changes.....	23
Frontend.....	27
Admin side.....	39
Database After Changes.....	56

Problem Statement

With the increasing demand for convenient transportation options, there is a pressing need for a comprehensive Train Ticket Booking Management System that caters to the diverse needs of both users and administrators. The primary objective of this project is to develop a robust and user-friendly **Book On Rails** using Java, focused on providing seamless booking experiences and efficient management functionalities. It includes both user-facing features for booking tickets and managing profiles, as well as administrative functionalities for overseeing system operations and ensuring data integrity.

Major Features

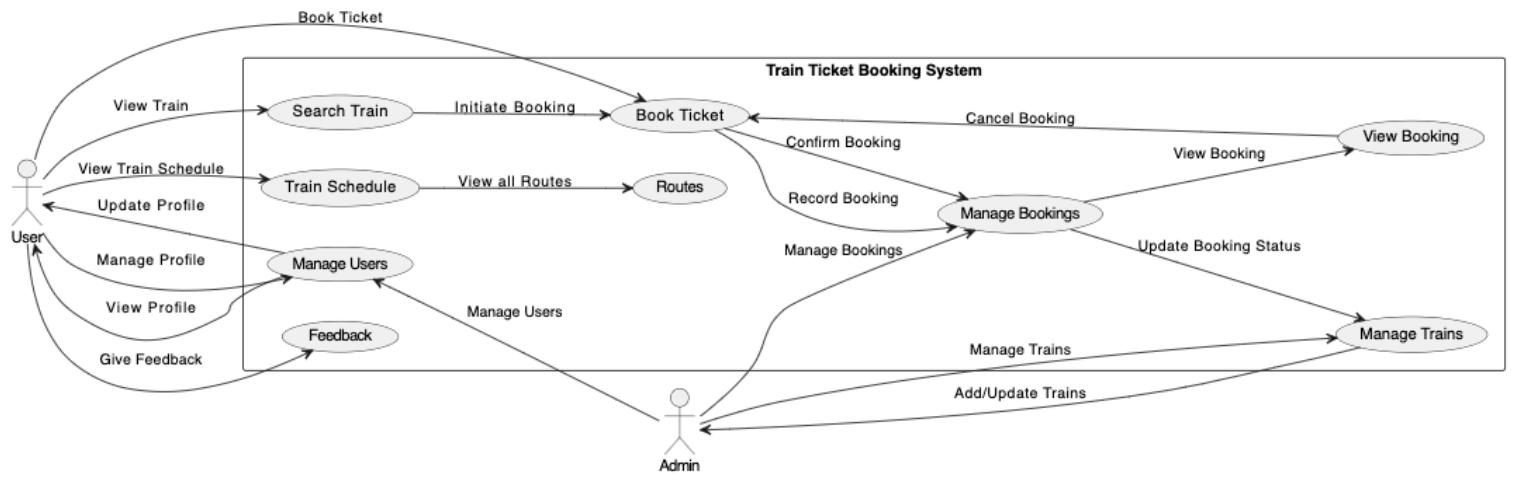
- **Search Trains:** Users can search for trains based on criteria such as class, station, and date, facilitating efficient journey planning.
- **Booking Ticket Online:** Seamless online booking experience featuring fare calculation, seat allocations, and instant booking confirmation.
- **User Management:** Allowing users to manage their profiles, view booking histories, and view their tickets as needed.
- **Admin Controls:** Empowers administrators to manage trains, user accounts, bookings, and seat availability, ensuring smooth system operations.

Minor Features

- **User Feedback and Ratings System:** Enables users to provide feedback and ratings for their travel experiences, enhancing overall user satisfaction and providing valuable insights for system improvement.
- **In-Train Food Menu:** Integrates an in-train food menu feature, allowing users to browse and select from the given food options (vegetarian and non-vegetarian) during their journey.
- **Viewing Train Schedules:** Allows users to easily access and view detailed schedules of trains, including information about all the stations they stop at, along with arrival and departure times, enhancing user convenience for trip planning.
- **Cancel Ticket:** Allows users to cancel their booked tickets, providing flexibility and convenience in adjusting travel plans.

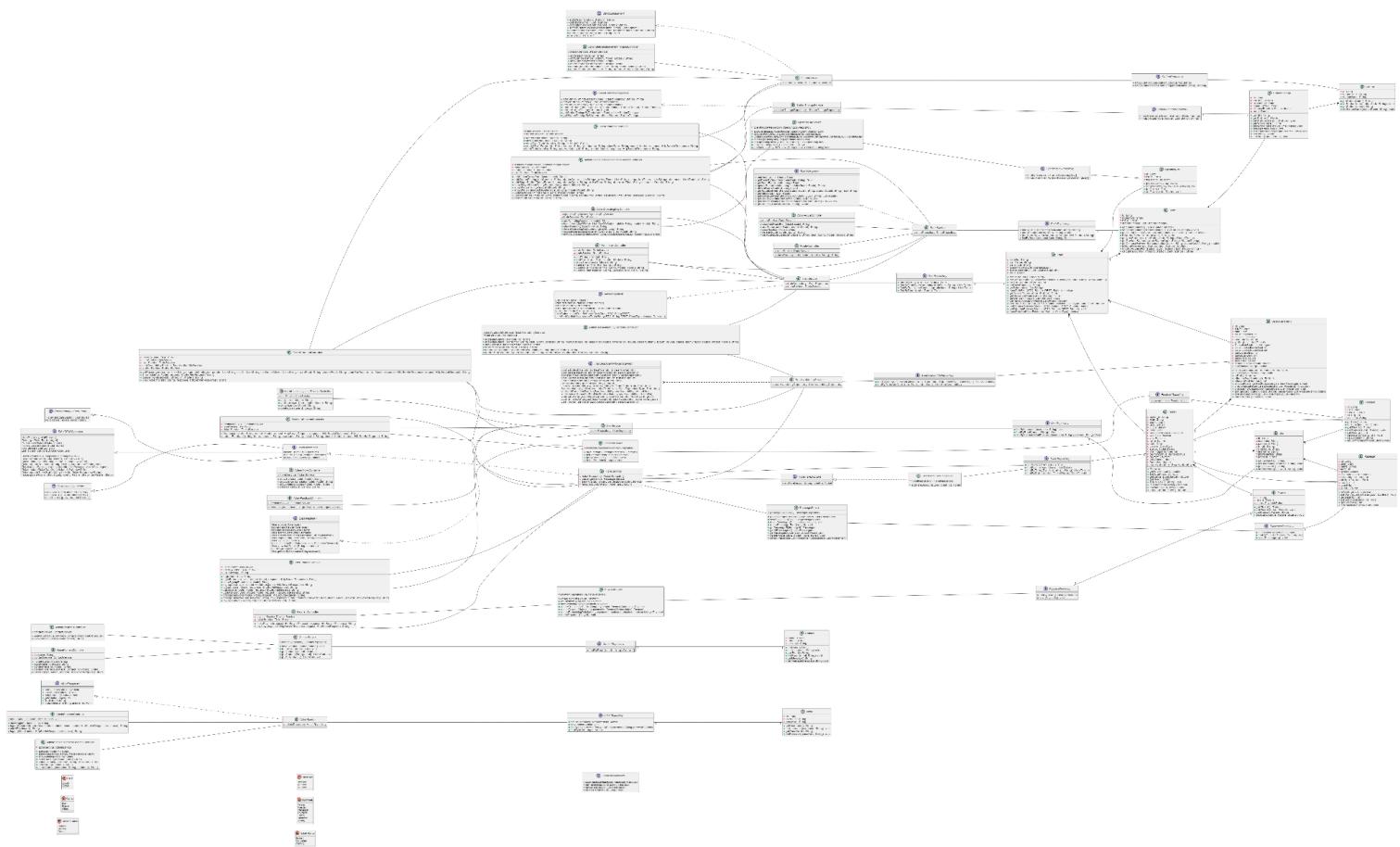
Models

Use Case Diagram



Class Diagram

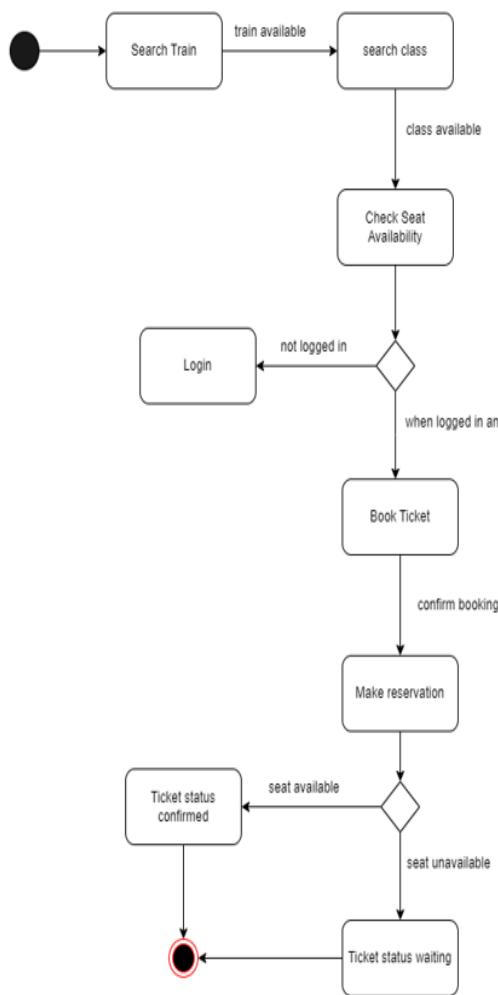
[ClassDiagram](#)



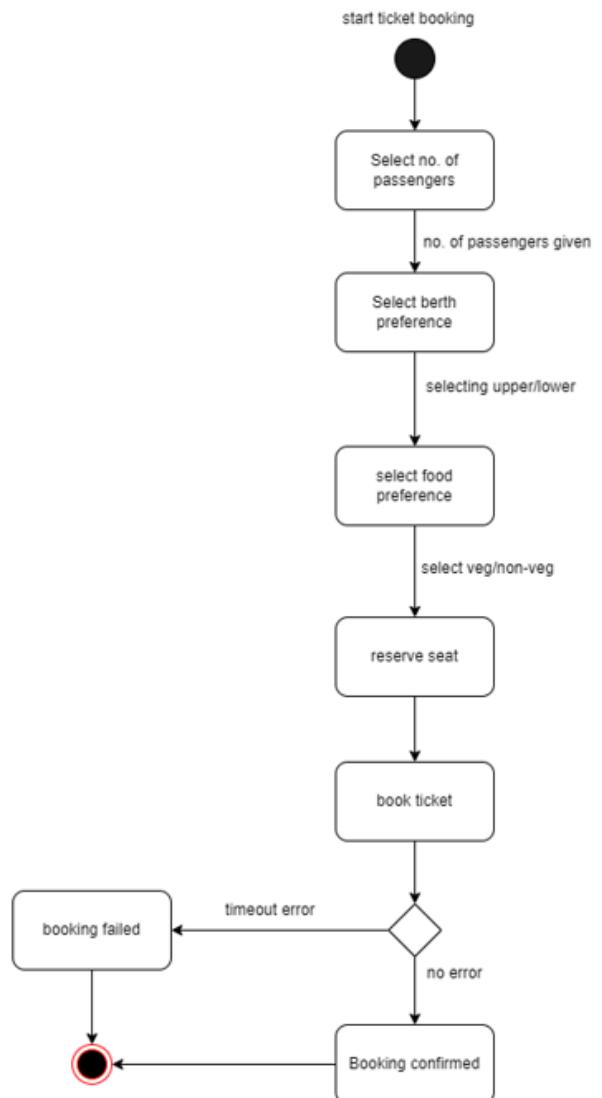
The class diagram illustrates a railway management system with various entities such as Admin, Train, Ticket, User, Feedback, and more, each encapsulating specific attributes and behaviors. These entities are interconnected through relationships like composition, aggregation, and association, facilitating data flow and system functionality. Additionally, the diagram outlines interfaces and services responsible for managing operations related to administration, feedback, ticketing, train management, and user interactions. Overall, it provides a comprehensive overview of the system's structure and functionalities.

State Diagrams:

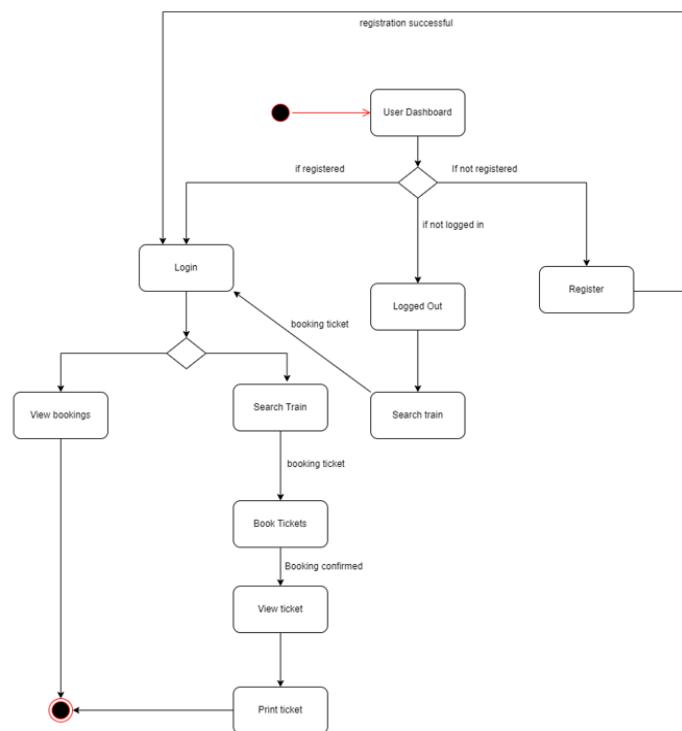
1. Search Train:



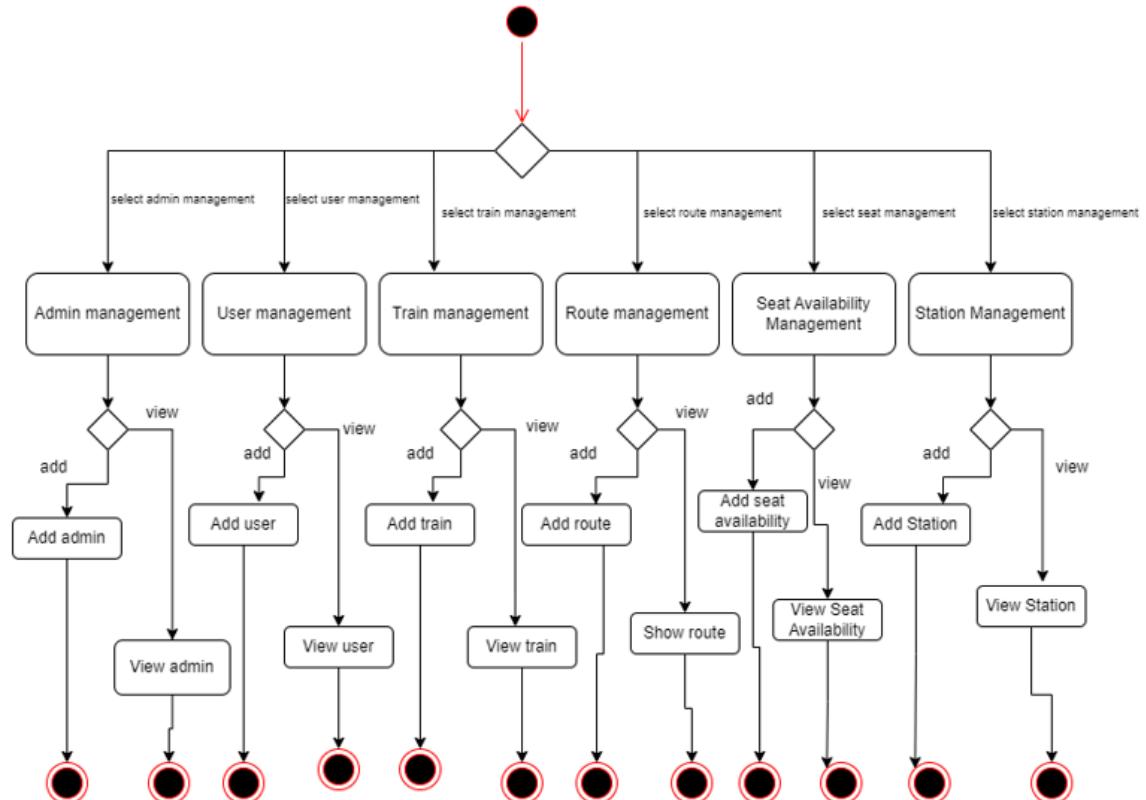
2. Booking :



3. User Management

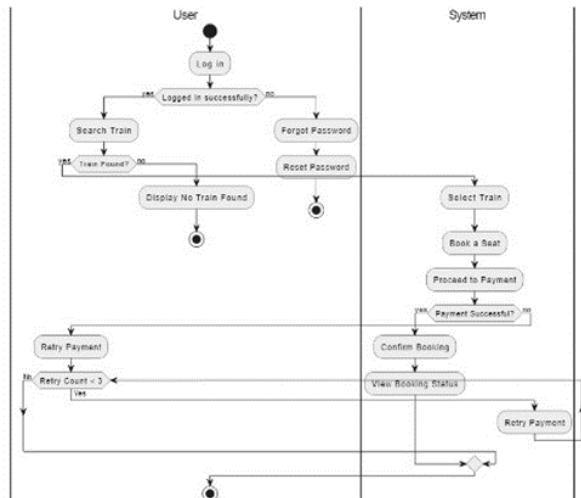


4. Admin Management

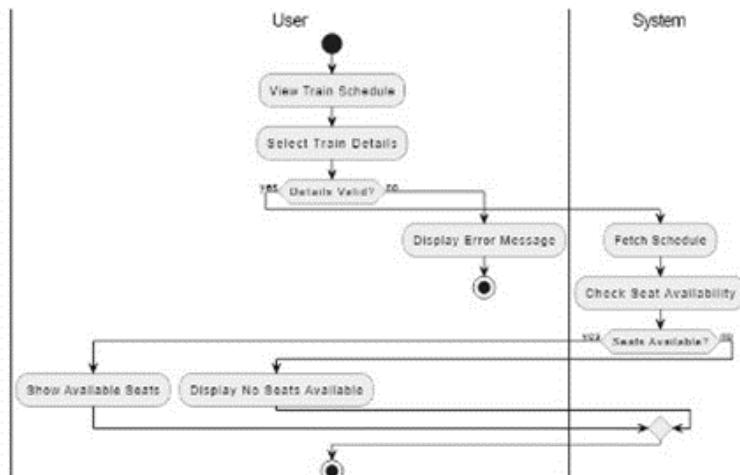


Activity Diagram

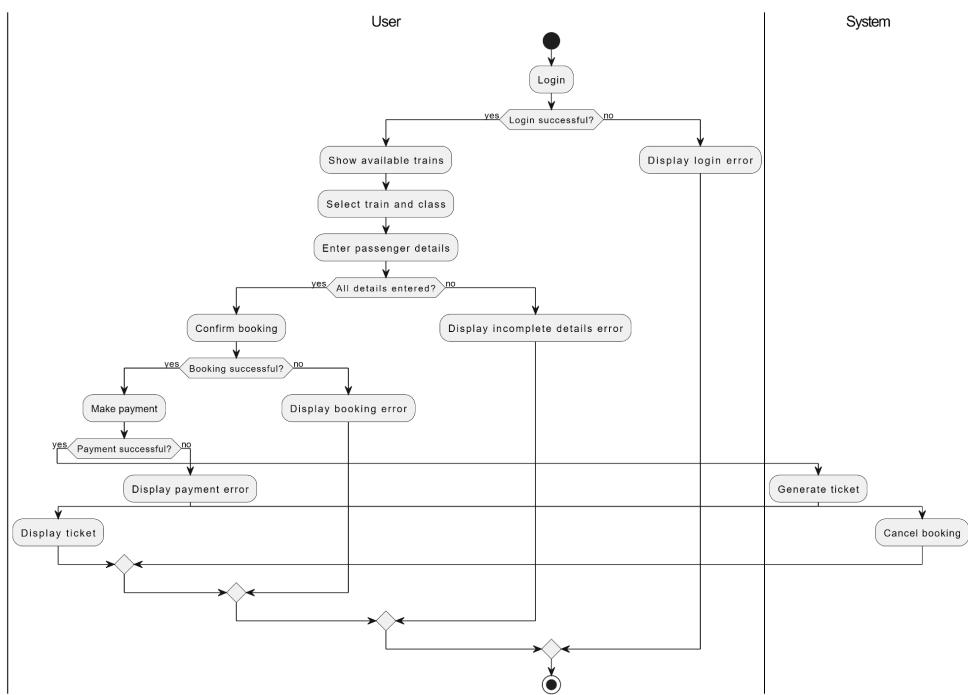
1. Search Train:



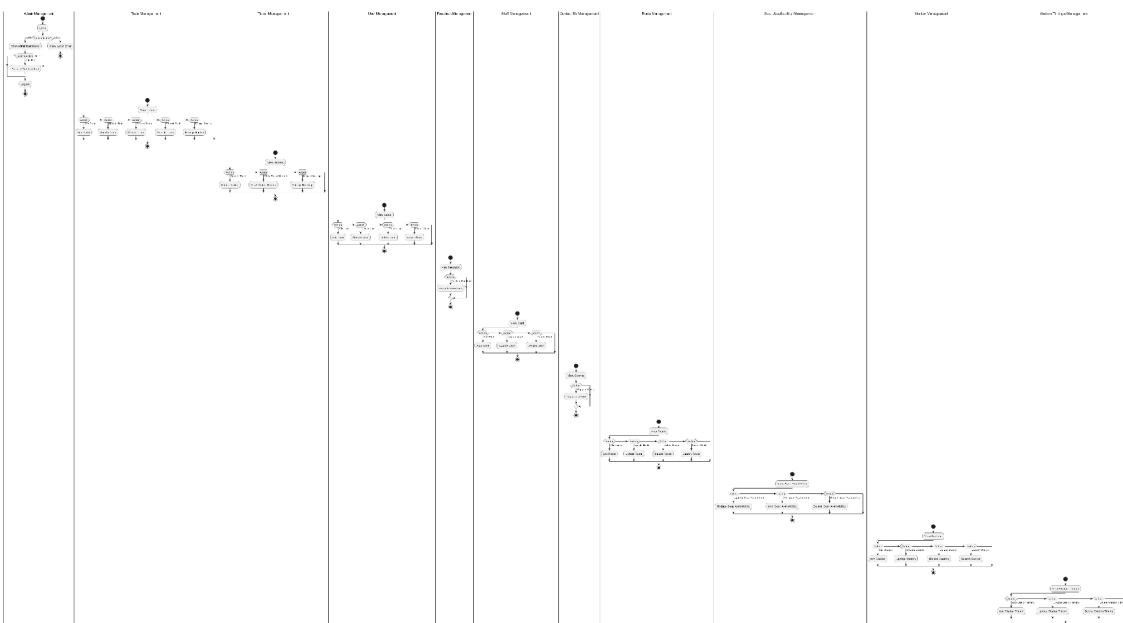
2. Train Schedule & Seat Availability:



3. User Management



4. Admin Management



Architecture Patterns

This project is a web application for managing train seat availability, we can map the MVC architecture as follows:

1. Model (Data):

- In this project, the Model represents the underlying data structure and business logic related to train seat availability. This includes entities like 'SeatAvailability', 'Train', and 'Ticket', along with their respective attributes and methods.
- The Model layer is responsible for managing data persistence, validation, and business rules related to seat availability, such as booking seats, managing waiting lists, and calculating fares.

2. View (User Interface):

- In this project, the View component utilizes JSP (JavaServer Pages) technology to create dynamic web pages that present data and interact with users. JSP pages contain a mix of HTML and Java code, allowing for the dynamic generation of content based on data retrieved from the Model layer.
- JSP pages are responsible for rendering the UI elements for viewing seat availability, booking tickets, and managing reservations. They integrate with server-side Java components, such as servlets and controllers, to retrieve data from the Model and process user requests.
- Views encapsulate the presentation logic and provide a user-friendly interface for interacting with the application.

3. Controller (Logic):

- The Controller component acts as an intermediary between the Model and View. It handles user requests, processes input data, interacts with the Model to retrieve or manipulate data, and determines which View to render in response to user actions.
- In this project, controllers like 'AdminSeatAvailabilityFrontendController' receive HTTP requests, extract parameters from request payloads or URLs, invoke appropriate methods in the Model layer (e.g., 'seatAvailabilityService'), and pass data to the View for rendering.

Example:

When a user accesses a JSP page to view seat availability (/admin/seat-availability/show.jsp), the server-side Controller (AdminSeatAvailabilityFrontendController) handles the request.

The Controller interacts with the Model layer (seatAvailabilityService) to retrieve seat availability data. The retrieved data is passed to the JSP page, where it is dynamically rendered using HTML and JSP tags, presenting the available seats and allowing users to perform actions like booking or managing reservations.

Controller

```
1 package com.bookonrails.ood.FrontendController;
2
3 import java.sql.Date;
4 import java.util.List;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.web.bind.annotation.*;
10
11 import com.bookonrails.ood.Model.*;
12 import com.bookonrails.ood.Service.SeatAvailabilityService;
13 import com.bookonrails.ood.Service.TrainService;
14
15 @Controller
16 @RequestMapping(path = "/admin/seat-availability")
17 @CrossOrigin(origins = "*")
18 public class AdminSeatAvailabilityFrontendController {
19     @Autowired
20     private SeatAvailabilityService seatAvailabilityService;
21
22     @Autowired
23     private TrainService trainService;
24
25     @GetMapping("/add")
26     public String addSeatAvailability(Model m) {
27         m.addAttribute(attributeName:"SeatAvailability", new SeatAvailability());
28         return "admin/seat-availability/add";
29     }
30
31     @PostMapping("/add")
32     public String addSeatAvailability(
33             @RequestParam("trainNo") String trainNo,
34             @RequestParam("date") String date,
35             @RequestParam("classes") String classes,
36             @RequestParam("no_of_coaches") int noOfCoaches,
37             @RequestParam("basePrice") double basePrice,
38             @RequestParam("farePerKM") double farePerKM,
39             @RequestParam("seniorCitizenDiscount") double seniorCitizenDiscount,
40             @RequestParam("cancellationCharge") double cancellationCharge,
41             Model model) {
42         SeatAvailability seatAvailability = new SeatAvailability(noOfCoaches);
43         // ...
```

model

```
package com.bookonrails.ood.Model;

import java.util.ArrayList;
import java.sql.Date;
import java.util.List;

import jakarta.persistence.*;

@Entity
public class SeatAvailability {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "trainNo", nullable = false)
    private Train train;

    @Temporal(TemporalType.DATE)
    private Date date;

    private int no_of_coaches;
    @Enumerated(EnumType.STRING) // to store the type of coach as a string
    private ClassType classes;
    private int availableSeats;

    @OneToMany(mappedBy = "seatAvailability", cascade = CascadeType.ALL)
    private List<Ticket> waitingList;

    @ElementCollection
    @CollectionTable(name = "CancelledSeats", joinColumns = @JoinColumn(name = "seatAvailability_id"))
    @Column(name = "seat_number")
    private List<Integer> CancelledSeats; // check n%2 = 1 -> Lower, n%2 = 0 -> Upper

    private int lastUnbookedLowerSeat;
    private int lastUnbookedUpperSeat;
    private int lastLowerSeat;
    private int lastUpperSeat;

    private double basePrice;
    private double farePerKM.
```

view

```
<!DOCTYPE html>
<!-- Base for all html pages --&gt;
&lt;html xmlns:th="http://www.thymeleaf.org" lang="en"&gt;

&lt;head&gt;
    &lt;meta charset="UTF-8"&gt;
    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
    &lt;link rel="stylesheet" th:href="@{/main.css}"&gt;
    &lt;link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous"&gt;
    |   &lt;!-- &lt;script src="https://cdn.tailwindcss.com"&gt;&lt;/script&gt; --&gt;

        &lt;title&gt;Book On Rails&lt;/title&gt;
&lt;/head&gt;

&lt;body&gt;
    &lt;header th:insert="~{admin-base :: header}"&gt;&lt;/header&gt;

    &lt;div class="content-body container-fluid"&gt;
        &lt;div class="container mt-5"&gt;

            &lt;!-- Block Content will be here --&gt;
            &lt;h2 class="mb-4"&gt;Seat Availability Form&lt;/h2&gt;
            &lt;div id="info" class="text-bold text-danger"&gt;

            &lt;/div&gt;
            &lt;form action="/admin/seat-availability/add" method="post"&gt;
                &lt;!-- Train Number Input --&gt;
                &lt;div class="form-group"&gt;
                    &lt;label for="trainNo"&gt;Train Number:&lt;/label&gt;
                    &lt;input type="text" class="form-control" id="trainNo" name="trainNo" required /&gt;
                &lt;/div&gt;

                &lt;!-- Date --&gt;
                &lt;div class="form-group"&gt;
                    &lt;label for="date"&gt;Date:&lt;/label&gt;
                    &lt;input type="date" class="form-control" id="date" name="date" min="" max="" required /&gt;
                &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/body&gt;</pre>
```

Design Principles

The SOLID principles are followed throughout the project, guaranteeing a solid and flexible software architecture:

1. Single Responsibility Principle(SRP): Classes and modules display distinct and well-defined roles across the project. As an example, service classes like ‘TicketCRUDOperations’, ‘TicketAllocationOperation’ and ‘TicketQueryOperation’ follow the SRP and are devoted to particular functionality like ticket booking operations. Repository classes, on the other hand, also emphasise maintainability and simplicity of understanding by concentrating only on data access and management using single responsibility principle.

```
package com.bookonrails.oad.Interface;

import java.sql.Date;
import java.util.List;

import com.bookonrails.oad.Model.Passenger;
import com.bookonrails.oad.Model.PaymentStatus;
import com.bookonrails.oad.Model.Station;
import com.bookonrails.oad.Model.Ticket;
import com.bookonrails.oad.Model.TicketStatus;
import com.bookonrails.oad.Model.Train;
import com.bookonrails.oad.Model.User;

public interface TicketCRUDOperations {
    List<Ticket> getAllTickets();
    Ticket getTicketById(Long id);
    Ticket saveTicket(Ticket ticket);
    Ticket updateTicket(Ticket ticket);
    void deleteTicket(Long id);
    List<Ticket> getTicketByUser(User user);

    // update
    Ticket updatePNR(Long ticketId, String newPNR);
    Ticket updateTrain(Long ticketId, Train newTrain);
    Ticket updateSource(Long ticketId, Station newSource);
    Ticket updateDestination(Long ticketId, Station newDestination);
    Ticket updatePassengers(Long ticketId, List<Passenger> newPassengers);
    Ticket updateTicketDate(Long ticketId, Date newDate);
    Ticket updateTicketStatus(Long ticketId, TicketStatus newStatus);
    Ticket updatePaymentStatus(Long ticketId, PaymentStatus newPaymentStatus);
}

package com.bookonrails.oad.Interface;

import com.bookonrails.oad.Model.Ticket;
import java.util.List;

public interface TicketAllocationOperations {
    Ticket allocateSeats(Ticket ticket);
    void cancelTicket(Ticket ticket);
}

public interface TicketQueryOperations {
    List<Ticket> getCancelledTickets();
    List<Ticket> getConfirmedTickets();
    List<Ticket> getWaitingListTickets();
}
```

2. Open/Closed Principle (OCP): The architecture of the project is intended to be closed to modification yet open to extension. It is possible to add new features or improvements without changing the current codebase. For example, by extending current interfaces or introducing new components, further capabilities like discount on fare can be smoothly added while preserving the stability and integrity of the Fare management system.

```

T CLICK HERE TO ASK DIDDUOX TO HELP YOU CODE FASTER
package com.bookonrails.oodl.Interface;

import com.bookonrails.oodl.Model.Ticket;

// for ocp
public interface TicketFareDiscount {
    Ticket addFareDiscount(Long ticketId);
}

5
6 import com.bookonrails.oodl.Interface.TicketFareDiscount;
7 import com.bookonrails.oodl.Model.Ticket;
8 import com.bookonrails.oodl.Repository.TicketRepository;
9
10 @Service
11 public class TicketFareDiscountService implements TicketFareDiscount {
12
13     @Autowired
14     private TicketRepository ticketRepository;
15
16     // @Override
17     public Ticket addFareDiscount(Long ticketId) {
18         Ticket ticket = ticketRepository.findById(ticketId).orElse(null);
19         if (ticket != null) {
20             double currentFare = ticket.getTotalAmount();
21             double discountedFare = currentFare - 50.0;
22             if (discountedFare < 0) {
23                 discountedFare = 0; // Ensure fare doesn't go negative
24             }
25             ticket.setFare(discountedFare);
26             return ticketRepository.save(ticket);
27         }
28         return null;
29     }
30 }
31

```

3. Interface Segregation Principle (ISP): Interfaces are created to be consistent and narrowly focused, following the ISP, throughout the project. To ensure that clients only rely on the methods they need, each interface defines a defined set of methods related to a single component of the system. Our system has used a lot of interfaces to ensure this.

```

2
3 import java.util.List;
4
5 import com.bookonrails.oodl.Model.Admin;
6
7 public interface AdminManagement {
8     public Admin authenticate(Admin a);
9     public void saveAdmin(Admin a);
10    public void deleteAdmin(Long id);
11    public Admin getAdmin(Long id);
12    public List<Admin> findAll();
13    public Admin findByUsername(String username);
14
15
16 }
17

```

```

@Service
public class AdminService implements AdminManagement {

    @Autowired
    private AdminRepository adminRepository;

    public Admin authenticate(Admin a) {
        // Implement authentication logic by checking credentials in the database
        Admin admin = adminRepository.findByUsernameAndPassword(a.getUsername(), a.getPassword());
        return admin;
    }

    public void saveAdmin(Admin admin) {
        adminRepository.save(admin);
    }

    public void deleteAdmin(Long id) {
        adminRepository.deleteById(id);
    }

    public Admin getAdmin(Long id) {
        return adminRepository.findById(id).get();
    }

    public List<Admin> findAll() {
        return adminRepository.findAll();
    }

    @Override
    public Admin findByUsername(String username){
        return adminRepository.findByUsername(username);
    }
}

```

4. Dependency Inversion Principle (DIP): DIP encourages the decoupling of high-level modules from low-level components by applying dependency injection and abstraction techniques. Interfaces are used to inject dependencies into classes, making it simple to switch out implementations. This improves the system's scalability, encourages code reuse, and makes testing easier.

The high-level module, represented by components like the 'RouteService', relies on access to seat availability data.

The low-level component corresponds to the data access layer, responsible for interactions with the database to manage seat availability information. This layer can be implemented using various technologies such as JDBC or JPA.

Dependency Injection is utilized by the 'RouteService'. Rather than directly instantiating a specific implementation of the 'RouteRepository', it declares a dependency on the 'RouteRepository' interface. Through Spring's '@Autowired' annotation, an appropriate implementation of the repository interface is injected at runtime. This approach decouples the service from the underlying data access mechanism, promoting flexibility and ease of maintenance.

```

1 package com.bookonrails.oodl.Service;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8 import org.springframework.transaction.annotation.Transactional;
9
10 import com.bookonrails.oodl.Interface.RouteManagement;
11 import com.bookonrails.oodl.Model.Route;
12 import com.bookonrails.oodl.Repository.RouteRepository;
13
14 @Service
15 public class RouteService implements RouteManagement {
16     @Autowired
17     private RouteRepository routeRepository; // dependency injection
18
19     public Route addRoute(Route route) {
20         return routeRepository.save(route);
21     }
22
23     public Route getRouteByRouteCode(String routeCode) {
24         return routeRepository.findByRouteCode(routeCode);
25     }
26
27     public Route getRouteById(Long routeId) {
28         return routeRepository.findById(routeId).orElse(null);
29     }
30
31 package com.bookonrails.oodl.Repository;
32
33 import java.util.List;
34
35 // import org.springframework.data.jdbc.repository.Query;
36 import org.springframework.data.jpa.repository.JpaRepository;
37 import org.springframework.data.jpa.repository.Query;
38 import org.springframework.data.repository.query.Param;
39 import org.springframework.stereotype.Repository;
40
41 import com.bookonrails.oodl.Model.Route;
42
43 @Repository
44 public interface RouteRepository extends JpaRepository<Route, Long> {
45     Route findByRouteCode(String routeCode);
46
47     @Procedure(name = "FindRouteBetweenStations") // returns route codes
48     public List<String> findRouteBetweenStation(@Param("stationCode1") String stationCode1, @Param("stationCode2") String stationCode2);
49
50     List<Route> findByTrainTrainName(String trainName);
51     Route findByTrainTrainNo(String trainNo);
52
53     List<Route> findByStationTimingsStationStationCode(@Param("stationCode") String stationCode);
54 }
```

Design Patterns

Facade Pattern

The Facade design pattern is a structural design pattern that provides a unified interface to a set of interfaces in a subsystem. It hides the complexities of the subsystem and provides a simpler interface to the client or user.

Facade: This is the main interface that the client interacts with. It provides a simplified interface to the more complex subsystem. The facade delegates client requests to appropriate objects within the subsystem.

```
@Controller
@RequestMapping(path = "/ticket")
@CrossOrigin(origins = "*")
public class TicketFrontendController {
    @Autowired
    private TicketService ticketService;

    @Autowired
    private UserService userService;

    @Autowired
    private TrainService trainService;

    @Autowired
    private SeatAvailabilityService seatAvailabilityService;

    @Autowired
    private StationService stationService;

    @Autowired
    private GeneralTicket generalTicket;

    @Autowired
    private TatkalTicket tatkalTicket;

    private boolean CheckTatkal (Date todayDate, Date ticketDate){
        // check if ticket is tatkal
        // ticket is tatkal if booked one day before
        // if ticket is booked today, it is also tatkal
        if(todayDate.compareTo(ticketDate)<=1){
            return true;
        }
        return false;
    }
}
```

Subsystems: These are the individual components or classes that make up the subsystem. Each subsystem class has its own responsibilities and methods. The facade does not encapsulate these subsystems but rather provides a simplified interface to work with them.

```

@Service
public class TrainService implements TrainManagement {

    @Autowired
    private TrainRepository trainRepository;

    @Autowired
    private RouteRepository routeRepository;

    // Method to retrieve all trains
    public List<Train> getAllTrains() {
        return trainRepository.findAll();
    }

    // Method to retrieve a train by train number
    public Train getTrainByTrainNo(String trainNo) {
        return trainRepository.findByTrainNo(trainNo);
    }

    // Method to save a train
    public Train saveTrain(Train train) {
        return trainRepository.save(train);
    }

@Service
public class SeatAvailabilityService implements TrainSeatAvailabilityManagement
{
    @Autowired
    private SeatAvailabilityRepository seatAvailabilityRepository;

    public SeatAvailability findById(Long sa){
        return seatAvailabilityRepository.findById(sa).orElse(null);
    }
    public void addSeatAvailability(SeatAvailability seatAvailability){
        seatAvailabilityRepository.save(seatAvailability);
    }

    public void deleteSeatAvailability(SeatAvailability seatAvailability){
        if(seatAvailability==null){
            throw new IllegalArgumentException(s:"Seat availability object is null or has no ID.");
        }
        seatAvailabilityRepository.delete(seatAvailability);
    }

    public boolean checkSeatAvailability(SeatAvailability seatAvailability){
        return seatAvailability.isFull();
    }

    public void updateSeatAvailability(SeatAvailability seatAvailability){
        if (seatAvailability == null || seatAvailability.getId() == null) {
            throw new IllegalArgumentException(s:"Seat availability object is null or has no ID.");
        }
        seatAvailabilityRepository.save(seatAvailability);
    }

    public List<SeatAvailability> getAllSeatAvailabilities(){
        return seatAvailabilityRepository.findAll();
    }
}

```

Template Pattern:

The Template Method pattern is a behavioral design pattern that defines the skeleton of an algorithm or operations in a superclass (often abstract) and leaves the details to be implemented by the child classes. It allows subclasses to customize specific parts of the algorithm without altering its overall structure. In our code, we create TicketAbstractService (abstract class) which has two ticket types, General Ticket and Tatkal Ticket (child classes), which implement the fare discount in detail.

```
24  @Service
25  public abstract class TicketAbstractService implements TicketManagement {
26
27      @Autowired
28      private TicketRepository ticketRepository;
29
30      @Autowired
31      private PassengerService passengerService;
32
33      @Autowired
34      private SeatAvailabilityService seatAvailabilityService;
35
36      @Autowired
37      private TicketFareDiscount fareDiscountStrategy; // Inject the FareDiscountStrategy interface
38
39
40  public List<Ticket> getAllTickets() {
41      return ticketRepository.findAll();
42  }
43
44  public String generateUniquePNR() {
45
46      String uniqueID = UUID.randomUUID().toString();
47      String pnr = uniqueID.substring(beginIndex:0, endIndex:6);
48      pnr = "PNR"+ pnr;
49      return pnr;
50  }
51
52
53  public Ticket addFareDiscount(Long ticketId,double discount) {
54      return fareDiscountStrategy.addFareDiscount(ticketId,discount);
55  }
56
57
58  package com.bookonrails.oodl.Service;
59
60  import org.springframework.beans.factory.annotation.Autowired;
61
62  import com.bookonrails.oodl.Model.Ticket;
63  import com.bookonrails.oodl.Repository.TicketRepository;
64
65  public class TatkalTicket extends TicketAbstractService {
66
67      @Autowired
68      private TicketRepository ticketRepository;
69
70      TatkalTicket(){
71          super();
72      }
73
74
75      @Override
76      public Ticket addFareDiscount(Long ticketId,double discount) {
77          // If ticket is Tatkal, we give discount for the ticket price and add tax and a double the base fare
78          Ticket t= ticketRepository.findById(ticketId).orElse(other:null);
79          if(t!=null) {
80              double currentFare = t.getTotalAmount();
81              double doubleFare = 2* currentFare;
82              double discountedFare = doubleFare - discount;
83              if (discountedFare < 0) {
84                  discountedFare = 0; // Ensure fare doesn't go negative
85              }
86              // t.setFare(discountedFare);
87              t.setFare(discountedFare + (discountedFare * 0.18)); // Add 18% tax
88              return ticketRepository.save(t);
89          }
90          return null;
91      }
92
93
94  }
```

```

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

import com.bookonrails.oodl.Model.Ticket;
import com.bookonrails.oodl.Repository.TicketRepository;

public class GeneralTicket extends TicketAbstractService {

    @Autowired
    private TicketRepository ticketRepository;

    GeneralTicket(){
        super();
    }

    @Override
    public Ticket addFareDiscount(Long ticketId,double discount) {
        // If ticket is General, we give discount for the ticket price and add tax
        Ticket t= ticketRepository.findById(ticketId).orElse(null);
        if(t!=null) {
            double currentFare = t.getTotalAmount();
            double discountedFare = currentFare - discount;
            if (discountedFare < 0) {
                discountedFare = 0; // Ensure fare doesn't go negative
            }
            // t.setFare(discountedFare);
            t.setFare(discountedFare + (discountedFare * 0.18)); // Add 18% tax
            return ticketRepository.save(t);
        }
        return null;
    }
}

```

Proxy Pattern:

The proxy pattern is a technique that allows one object — the proxy — to control access to another object — the subject or service. In Spring, beans are proxied to control access to the underlying bean. We see this approach when using transactions in TrainService:

```

64
65     @Transactional
66     public List<Train> searchTrainBySrcAndDest(String SRC,String DEST){
67         List<Train> t = new ArrayList<>();
68         List<String> r= routeRepository.findRouteBetweenStation(SRC,DEST );
69         for(String s:r){
70             Route route= routeRepository.findByRouteCode(s);
71             t.add(trainRepository.findByRoute(route));
72             System.out.println(route.getRouteCode());
73
74         }
75         return t;
76     }
77
78     @Transactional
79     public List<SeatAvailability> searchTrain(String SRC,String DEST, ClassType classes, Date date){
80         List<Train> t=searchTrainBySrcAndDest(SRC,DEST);
81         // now we need to filter trains based on if they run on the given date
82         // and then filter them based on class type
83         List<SeatAvailability> newList= new ArrayList<>();
84         for(int i=0;i<t.size();i++){
85             if(t.get(i).doesDateAndClassExist(date,classes)){
86                 newList.add(t.get(i).getSeatAvailabilityBasedOnClassAndDate(date, classes));
87             }
88         }
89         return newList;
90     }
91

```

Singleton Pattern:

The singleton pattern is a mechanism that ensures only one instance of an object exists per application. This pattern can be useful when managing shared resources or providing cross-cutting services, such as logging. The Spring @Autowired annotation demonstrates autowired singleton pattern, which creates only one object for a repository and achieves loose coupling between classes.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

// import com.bookonrails.ood.Interface.TrainSeatAvailabilityManagement;
import com.bookonrails.ood.Model.ClassType;
import com.bookonrails.ood.Model.SeatAvailability;
import com.bookonrails.ood.Model.Ticket;
import com.bookonrails.ood.Model.Train;
import com.bookonrails.ood.Repository.SeatAvailabilityRepository;

@Service
public class SeatAvailabilityService // extends TrainSeatAvailabilityManagement
{
    @Autowired
    private SeatAvailabilityRepository seatAvailabilityRepository;

    public SeatAvailability findById(Long sa){
        return seatAvailabilityRepository.findById(sa).orElse(null);
    }
    public void addSeatAvailability(SeatAvailability seatAvailability){
        seatAvailabilityRepository.save(seatAvailability);
    }
}
```

```
@Service
public abstract class TicketAbstractService implements TicketManagement {

    @Autowired
    private TicketRepository ticketRepository;

    @Autowired
    private PassengerService passengerService;

    @Autowired
    private SeatAvailabilityRepository seatAvailabilityRepository;

    @Autowired
    private TicketFareDiscount fareDiscountStrategy; // Inject the FareDiscountStrategy interface

    public List<Ticket> getAllTickets() {
        return ticketRepository.findAll();
    }

    public String generateUniquePNR() {
        String uniqueID = UUID.randomUUID().toString();
        String pnr = uniqueID.substring(beginIndex:0, endIndex:6);
        pnr = "PNR"+ pnr;
        return pnr;
    }

    public List<Ticket> getTicketByUser(User user) {
```

Here TicketAbstractService and SeatAvailabilityService are using SeatAvailabilityRepository, but only one object of SeatAvailabilityRepository is created, following Singleton Pattern.

Github

<https://github.com/Mystery510/Book-On-Rails>

Individual Contributions :

Shreya Mishra

Shreya Mishra made significant contributions across both backend and frontend aspects of the project. She took a lead role in integrating the backend services with the database, ensuring seamless data storage and retrieval. Shreya also worked extensively on the frontend, where she leveraged her skills in HTML, CSS, and Thymeleaf to create dynamic web pages that displayed data fetched from the backend. Her holistic approach to development bridged the gap between the frontend and backend components, resulting in a cohesive and functional application.

Siri

Siri's contributions were predominantly in the backend development of the application. She took the lead in designing and implementing the core backend logic of the system. Siri's responsibilities included developing RESTful APIs, managing database interactions, and handling server-side functionalities. Her expertise in Java and Spring Boot framework enabled the creation of robust backend services supporting features such as user authentication, ticket booking, and data retrieval.

Shreeja

Shreeja's primary focus was on backend development, working closely with Siri. Together, they collaborated on designing and implementing the core logic of the system. Shreeja's responsibilities included developing RESTful APIs, managing database interactions, and implementing server-side functionalities. Her expertise in Java and Spring Boot framework was instrumental in creating the backend services supporting various application features. Shreeja's contributions ensured the backend was reliable, efficient, and scalable.

Sinchana

Sinchana played a crucial role in the frontend development of the application. Her primary focus was on creating intuitive and visually appealing user interfaces. She worked extensively on crafting responsive and dynamic web pages using HTML, CSS, and Bootstrap. Sinchana's attention to detail ensured that the frontend components were not only functional but also aesthetically pleasing. Additionally, she contributed to the implementation of client-side JavaScript functionality to enhance user interactivity.

Overall Collaboration

While each team member had their primary areas of focus, we all worked collaboratively on various aspects of the project. Regular meetings and discussions facilitated a smooth integration of frontend and backend components. Sinchana's frontend designs seamlessly integrated with the APIs developed by Siri and Shreeja, thanks to Shreya's efforts in database integration. This collaborative approach ensured that the final product met the project requirements and delivered a user-friendly experience.

Screenshots

Database Before Changes

User Table:

```
1 • use bookonrails;
2 • select * from user;
```

The screenshot shows the 'User' table in MySQL Workbench. The table has columns: id, address, email, first_name, last_name, password, phone_number, and username. There are 4 rows of data.

	id	address	email	first_name	last_name	password	phone_number	username
▶	1	Karnataka	dws414@gmail.com	Shreya	Mishra	shreya	343876721	ishreya09
	2	Bangalore	shreya.pes@gmail.com	SHREYA	MISHRA PES1UG21CS574	shreya	3727361221	shreya
	3	Bangalore	cc@gmail.com	Cloud Computing	Google Backup	cc	121135464	cc
*	4	Bangalore	abc@gmail.com	abc	abc	abc	123454535454	abc
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Admin Table:

```
4 • select * from admin;
```

The screenshot shows the 'admin' table in MySQL Workbench. The table has columns: id, password, and username. There are 5 rows of data.

	id	password	username
▶	1	shreya	shreya
	2	shreya123	shreya123
	4	shreya	ishreya09
*	5	adrija	adrija
	NULL	NULL	NULL

Contact Table:

```
5 • select * from contact;
```

The screenshot shows the 'contact' table in MySQL Workbench. The table has columns: id, email, message, and name. There are 3 rows of data.

	id	email	message	name
▶	1	shreya.pes@outlook.com	hey	PES1UG21CS574 Shreya Mishra
	2	dsb@gxd.com	bshfbewhb	sgd
*	NULL	NULL	NULL	NULL

Train :

	train_no	train_name	train_type	route_id	does
▶	12334	Ranchi Muri Express	Express	1	0
	123455	Hyderabad Vishakhapatnam Shatabdi	Shatabdi	10	0
	12844	Ranchi Yashwantpur Shatabdi	Shatabdi	2	0
	21112	Hatia Pune Garib Rath	Garib Rath	3	0
	63722	Hyderabad Bangalore Express	Express	8	0
	67732	Hatia Hyderabad Express	Express	4	0
*	NULL	NULL	NULL	NULL	NULL

Route:

	id	route_code
▶	1	R01: HATIA MURI ROUTE
	2	R02: HATIA YASHWANTPUR ROUTE
	3	R03
	4	R04
	8	R05
	10	R06
*	NULL	NULL

Operating day:

	id	day_of_week	train_no
▶	1	Monday	67732
	2	Thursday	67732
	3	Tuesday	12334
	5	Saturday	12334
	6	Tuesday	12844
	7	Friday	12844
	8	Thursday	12334
*	NULL	NULL	NULL

Seat Availability:

	id	available_seats	base_price	cancellation_charge	classes	date	fare_perkm	last_lower_seat	last_unbooked_lower_seat	last_
	11	30	100	50	Sleeper	2024-04-19	20	29	1	2
	12	30	100	50	AC3Tier	2024-04-19	20	29	1	2
	13	0	100	50	AC2Tier	2024-04-19	20	9	11	12
	14	28	100	50	AC3Tier	2024-04-23	20	29	3	4
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Station:

	station_code	station_name
▶	BLR	Bangalore
	HTE	Hatia (Ranchi)
	HYD	Hyderabad
	KGI	Kengiri
	KRP	Krishnarajapuram
	MURI	Muri
	PUNE	Pune
	RKL	Rourkela
station 9		x

Station Timings:

	id	arrival_time	departure_time	distance_from_next_station	route_id	station_code
▶	1	10:30:00.000000	10:45:00.000000	45	1	HTE
	2	01:30:00.000000	01:45:00.000000	120	1	RNC
	4	07:20:00.000000	07:45:00.000000	0	1	MURI
	5	10:30:00.000000	10:45:00.000000	45	2	HTE
	6	01:30:00.000000	01:45:00.000000	120	2	RNC
	7	07:20:00.000000	07:45:00.000000	0	2	MURI
	8	11:30:00.000000	11:45:00.000000	120	2	RKL
	9	22:30:00.000000	22:45:00.000000	200	2	YPR

Ticket:

	ticket_id	pnr	waiting_list_number	classes	date	foodprice	payment_status	quantity	status	total_amount	veg	wa
▶	19	PNR40c41b	-1	AC2Tier	2024-04-19	800	Success	2	Confirmed	24500	0	1
	20	PNRdb499d	-1	AC2Tier	2024-04-19	1200	Success	4	Confirmed	24500	1	1
	21	PNR2eb927	-1	AC3Tier	2024-04-23	800	Success	2	Confirmed	11550	0	1
*	HULL	NULL	NULL	HULL	NULL	NULL	NULL	HULL	HULL	NULL	NULL	NULL

Ticket_passenger:

```
13 • select * from ticket_passengers;
```

	ticket_ticket_id	passenger_id
▶	19	32
	19	33
	19	34
	19	35
	19	36
	20	37
	20	38
	20	39

Passenger:

```
14 • select * from passenger;
```

	id	age	birthpreference	gender	is_senior_citizen	is_waiting_list	name	seat_no	ticket_id	user_id	coach_no
37	11	Lower		Female	0	0	P1	9	20	2	10
38	34	Lower		Male	0	0	p2	11	20	2	12
39	34	Lower		Male	0	0	p3	11	20	2	12
40	32	Lower		Female	0	0	p4	11	20	2	12
41	5	Lower		Female	0	0	p5	11	20	2	12
42	12	Lower		Female	0	0	Shre...	1	21	2	2
43	21	Upper		Female	0	0	adrija	2	21	2	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Payment:

	id	status	ticket_id
▶	12	Success	19
	13	Success	20
	14	Success	21
*	NULL	NULL	NULL

payment 14 ×

Feedback:

	id	comment	rating	ticket_id	user_id
▶	1	good	4	20	2
	2	okayish	2	19	2
*	NULL	NULL	NULL	NULL	NULL

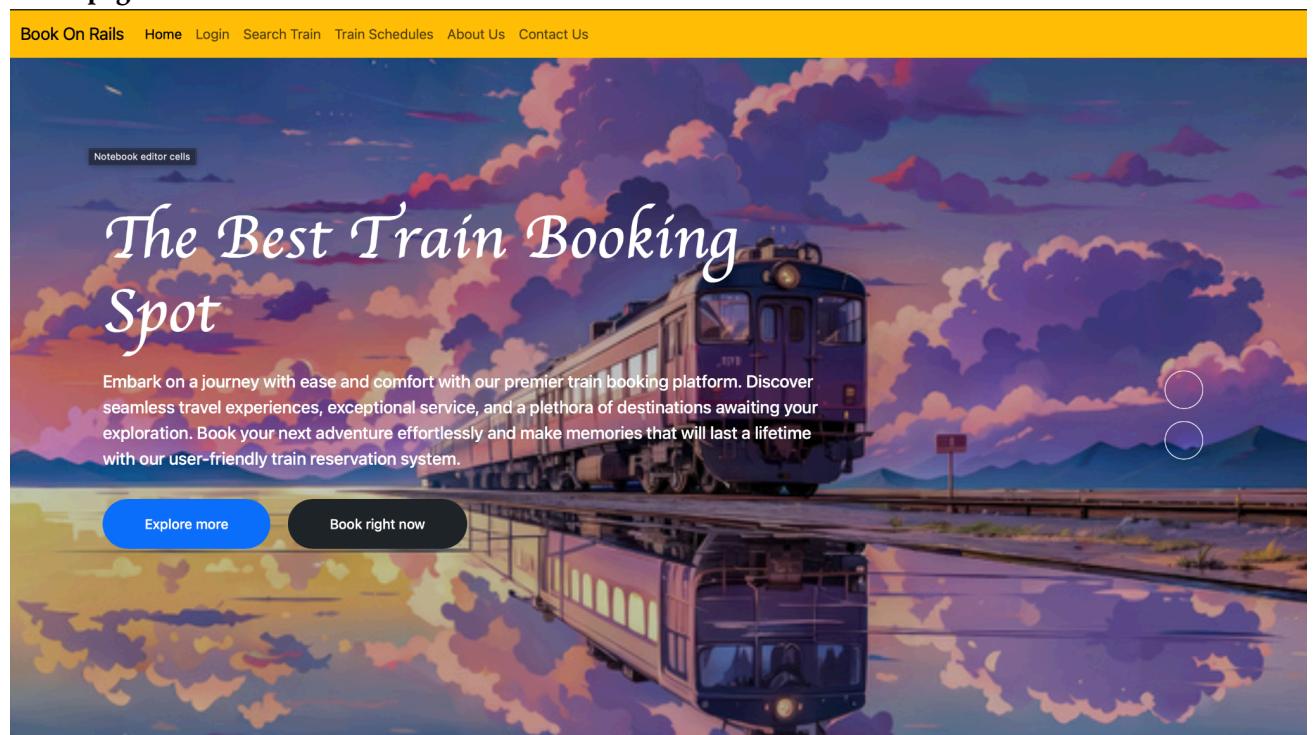
cancelled_seats:

```
17 •   select * from cancelled_seats;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	seat_availability_id	seat_number		
▶	5	2		

Frontend

Home page :



Contact Us :

Contact us

Name:

Email:

Message:

Contact Us

Adding Data :

Contact us

Name:

Email:

Message:

Hello, this is regarding train no 12844

Contact Us

Your message has been sent successfully

Added to Database :

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
		id	email	message	name
▶	1	shreya.pes@outlook.com	hey	PES1UG21CS574 Shreya Mishra	
	2	dsb@gxd.com	bshfbewhb	sgd	
*	3	shreyamishra414@gmail.com	Hello, this is regarding train no 12844	Shreya Mishra	NULL
		NULL	NULL	NULL	NULL

Login Page :

Book On Rails Home Login Search Train Train Schedules About Us Contact Us

Login Here...

Username:

Password:

Login

Don't have an account with us? [Register here](#)

Logging in using shreya username:

Book On Rails Home Login Search Train Train Schedules About Us Contact Us

Login Here...

Username:

Password:

Login

Don't have an account with us? [Register here](#)

User Dashboard:

Book On Rails Home Dashboard Logout Search Train Train Schedules About Us Contact Us

SHREYA's Dashboard

First Name:

Last Name:

Email:

Phone No.:

Address:

Other Options

[Change Password](#)

[My Trips](#)

[Search Train](#)

User's Booked Trips:

The screenshot displays a list of three booked trips under the heading "SHREYA's Trips". Each trip entry includes the train name, status, PNR, class, date, number of passengers, total fare, departure time, arrival time, and route details. Buttons for "View Ticket" and "Cancel Ticket" are shown for each trip.

Train Details	Departure Time	Arrival Time	Route
Ranchi Yashwantpur Shatabdi Confirmed Train No: #12844 PNR: PNR2eb927 Class: AC3Tier Date of Journey: 2024-04-23 Number Of Passengers: 2 Total Fare: 12350.0	10:45:00 RTE: Hatia (Ranchi)	22:30:00 YPR: Yashwantpur Junction	----- 2 days 06 hours 55 minutes ----->
Ranchi Yashwantpur Shatabdi Confirmed Train No: #12844 PNR: PNRdb499d Class: AC2Tier Date of Journey: 2024-04-19 Number Of Passengers: 5 Total Fare: 25700.0	01:45:00 RNC: Ranchi	22:30:00 YPR: Yashwantpur Junction	----- 0 days 22 hours 10 minutes ----->
Ranchi Yashwantpur Shatabdi Confirmed Train No: #12844 PNR: PNR40c41b Class: AC2Tier Date of Journey: 2024-04-19 Number Of Passengers: 5 Total Fare: 25300.0	01:45:00 RNC: Ranchi	22:30:00 YPR: Yashwantpur Junction	----- 0 days 22 hours 10 minutes ----->

View Ticket page:

The screenshot shows the "View Ticket" page with the URL "localhost:4000/ticket/show-ticket". It displays the "Electronic Reservation Slip" with journey details, train information, and payment status. Below this is the "Passenger Details" section, which lists five passengers with their names, ages, genders, seat numbers, coach numbers, and waiting list status. At the bottom are buttons for "Print Ticket" and "Book Another Ticket".

Name	Age	Gender	Seat Number	Coach Number	Waiting List
P1	11	Female	9	10	false
p2	34	Male	11	12	false
p3	34	Male	11	12	false
p4	32	Female	11	12	false
p5	5	Female	11	12	false

Feedback :

Book On Rails Home Dashboard Logout Search Train Train Schedules About Us Contact Us

Feedback Form

Rating:

★★★★★

Comments:

Food was awesome

Submit Feedback

Book On Rails Home Dashboard Logout Search Train Train Schedules About Us Contact Us

Thank you for your valuable feedback

Feedback added:

Result Grid		Filter Rows:		Edit:	
	ID	Comment	Rating	Ticket ID	User ID
▶	1	good	4	20	2
	2	okayish	2	19	2
*	3	Food was awesome	4	20	2
	NULL	NULL	NULL	NULL	NULL

Search Train:

Book On Rails Home Dashboard Logout Search Train Train Schedules About Us Contact Us

Search Trains

Date: dd-mm-2024

From Station: BLR: Bangalore

To Station: BLR: Bangalore

Class: 3rd AC

Search

Search Trains

Date:

From Station:

To Station:

Class:

Rendering trains available for the query:

Search Results

Train No	Train Name	Days Running	No. of Available Seats	Book Now
12844	Ranchi Yashwantpur Shatabdi	Tuesday Monday	28	<input type="button" value="Book Now"/>

Clicking on Book now, Train Reservation form opens up

Book On Rails Home Login Dashboard Logout Search Train Train Schedules About Us Contact Us

Train Reservation Form

Number of Passengers:

Passenger 1 Details

Name:

Age:

Gender:

Birth Preference:

Are you a Senior Citizen?

Passenger 2 Details

Name:

Age:

Gender:

Birth Preference:

Are you a Senior Citizen?

Do you want Food?

Make Reservation

Book On Rails Home Login Dashboard Logout Search Train Train Schedules About Us Contact Us

Passenger 2 Details

Name:

Age:

Gender:

Birth Preference:

Are you a Senior Citizen?

Yes

Veg

For how many people?

Make Reservation

Ticket Details before Payment

Ticket Details

PNR	: PNR579010
Ticket Id	: 22
Train No	: 12844
Train Name	: Ranchi Yashwantpur Shatabdi
SRC	: MURI: Muri
DEST	: YPR: Yashwantpur Junction
Date	: 2024-04-23
Class	: AC3Tier
Food Price	: 600.0
Ticket Price	: 4950.0
Amount to be paid	: 5550.0

Reservation Details

Passenger Name	Age	Gender	Birth Preference	Senior Citizen	
p1	20	Female	Lower	false	<button>Confirm payment</button>
p2	30	Female	Upper	false	<button>Confirm payment</button>

Pay

PAY NOWWW !!

Ticket booked

Electronic Reservation Slip

Journey Details

Source: MURI: Muri	Train No: 12844
Destination: YPR: Yashwantpur Junction	Train Name: Ranchi Yashwantpur Shatabdi
Date of Journey: 2024-04-23	Total No of Passengers: 2
Class: AC3Tier	Booked by: SHREYA MISHRA PES1UG21CS574
Departure Time: 07:45:00	Ticket Price: 4950.0
Arrival Time: 22:30:00	Food Price: 600.0
Time of Journey: 0 days 15 hours 55 minutes	Total Amount: 5550.0
Ticket Status: Confirmed	Payment Status: Success

Passenger Details

Name	Age	Gender	Seat Number	Coach Number	Waiting List
p1	20	Female	3	4	false
p2	30	Female	4	5	false

[Print Ticket](#) [Book Another Ticket](#)

The screenshot shows two overlapping windows. On the left is a 'Print' dialog box with settings for 'Printer' (Save as PDF), 'Layout' (Portrait), 'Pages' (All), and buttons for 'Save' and 'Cancel'. On the right is an 'Electronic Reservation Slip' titled 'Journey Details'. It lists travel information: Source: MURI: Muri, Destination: YPR: Yashwantpur Shatabdi, Date of Journey: 2024-04-23, Class: AC3Tier, Departure Time: 07:45:00, Arrival Time: 22:30:00, Time of Journey: 0 days 15 hours 55 minutes, and Ticket Status: Confirmed. To the right of this is a summary of the trip: Train No: 12844, Train Name: Ranchi Yashwantpur Shatabdi, Total No of Passengers: 2, Booked by: SHREYA MISHRA PES1UG21CS574, Ticket Price: 4950.0, Food Price: 600.0, Total Amount: 5550.0, and Payment Status: Success. Below this is a 'Passenger Details' table:

Name	Age	Gender	Seat Number	Coach Number	Waiting List
p1	20	Female	3	4	false
p2	30	Female	4	5	false

At the bottom of the reservation slip are 'Print Ticket' and 'Book Another Ticket' buttons.

Ticket reflected in my trips

This section displays three confirmed train journeys for 'SHREYA'. Each journey is shown in a separate box with a yellow header bar containing 'View Ticket' and 'Cancel Ticket' buttons.

- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNR579010
 - Class: AC3Tier
 - Date of Journey: 2024-04-23
 - Number Of Passengers: 2
 - Total Fare: 5550.0
 Departure: 07:45:00 MURI: Muri, Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 0 days 15 hours 55 minutes
- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNR2eb927
 - Class: AC3Tier
 - Date of Journey: 2024-04-23
 - Number Of Passengers: 2
 - Total Fare: 12350.0
 Departure: 10:45:00 HTE: Hatia (Ranchi), Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 2 days 06 hours 55 minutes
- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNRdb499d
 - Class: AC2Tier
 - Date of Journey: 2024-04-19
 - Number Of Passengers: 5
 - Total Fare: 12350.0
 Departure: 01:45:00 RNC: Ranchi, Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 0 days 22 hours 10 minutes

Cancel 2nd Ticket

A confirmation dialog box from 'localhost:4000 says' asks 'Are you sure you want to cancel this ticket?' with 'OK' and 'Cancel' buttons.

The 'SHREYA's Trips' section shows the same three journeys as before. The second journey is highlighted with a red background.

- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNR579010
 - Class: AC3Tier
 - Date of Journey: 2024-04-23
 - Number Of Passengers: 2
 - Total Fare: 5550.0
 Departure: 07:45:00 MURI: Muri, Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 0 days 15 hours 55 minutes
- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNR2eb927
 - Class: AC3Tier
 - Date of Journey: 2024-04-23
 - Number Of Passengers: 2
 - Total Fare: 12350.0
 Departure: 10:45:00 HTE: Hatia (Ranchi), Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 2 days 06 hours 55 minutes
- Ranchi Yashwantpur Shatabdi** (Confirmed):
 - Train No: #12844
 - PNR: PNRdb499d
 - Class: AC2Tier
 - Date of Journey: 2024-04-19
 - Number Of Passengers: 5
 - Total Fare: 12350.0
 Departure: 01:45:00 RNC: Ranchi, Arrival: 22:30:00 YPR: Yashwantpur Junction, Duration: 0 days 22 hours 10 minutes

Ticket Cancelled (Amount that will be refunded is after cancellation charge is subtracted)

Book On Rails | Home | Dashboard | Logout | Search Train | Train Schedules | About Us | Contact Us

The amount of Rs. 12300.0 will be refunded in 10-15 working days

Ticket Cancelled

SHREYA's Trips

Ranchi Yashwantpur Shatabdi

Confirmed

Train No: #12844
PNR: PNR579010
Class: AC3Tier
Date of Journey: 2024-04-23
Number Of Passengers: 2
Total Fare: 5550.0

[View Ticket](#)

[Cancel Ticket](#)

07:45:00 MURI: Muri ----- 0 days 15 hours 55 minutes -----> 22:30:00 YPR: Yashwantpur Junction

Ranchi Yashwantpur Shatabdi

Cancelled

Train No: #12844
PNR: PNR2eb927
Class: AC3Tier
Date of Journey: 2024-04-23
Number Of Passengers: 2
Total Fare: 12350.0

[View Ticket](#)

10:45:00 HTE: Hatia (Ranchi) ----- 2 days 06 hours 55 minutes -----> 22:30:00 YPR: Yashwantpur Junction

Ranchi Yashwantpur Shatabdi

Confirmed

Train No: #12844
PNR: PNRdb499d
Class: AC2Tier
Date of Journey: 2024-04-19
Number Of Passengers: 2

[View Ticket](#)

[Give Feedback](#)

01:45:00 RNC: Ranchi ----- 0 days 22 hours 10 minutes -----> 22:30:00 YPR: Yashwantpur Junction

Sign Up Page

Book On Rails | Home | Login | Search Train | Train Schedules | About Us | Contact Us

Register Here...

First Name:

Last Name:

Email:

Phone Number:

Address:

Username:

Password:

[Sign Up](#)

Adding User

Book On Rails | Home | Login | Search Train | Train Schedules | About Us | Contact Us

Register Here...

First Name:

Last Name:

Email:

Phone Number:

Address:

Username:

Password:

User Dashboard for ooad user

Book On Rails | Home | Dashboard | Logout | Search Train | Train Schedules | About Us | Contact Us

ooad's Dashboard

First Name:

Last Name:

Email:

Phone No.:

Address:

Other Options

[Change Password](#)

[My Trips](#)

[Search Train](#)

[Update Profile](#)

User added in database

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:	
▶	1	Karnataka	dws414@gmail.com	Shreya	Mishra	shreya	343876721	ishreya09	
	2	Bangalore	shreya.pes@gmail.com	SHREYA	MISHRA PES1UG21CS574	shreya	3727361221	shreya	
	3	Bangalore	cc@gmail.com	Cloud Computing	Google Backup	cc	121135464	cc	
	4	Bangalore	abc@gmail.com	abc	abc	abc	123454535454	abc	
*	5	Bangalore	ooad@gmail.com	ooad	in java	ooad	9999923421	ooad	
*		NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Updating user

Book On Rails | Home | Dashboard | Logout | Search Train | Train Schedules | About Us | Contact Us

ooad's Dashboard

First Name:
Shreya

Last Name:
Mishra

Email:
ood@gmail.com

Phone No.:
9999923421

Address:
Bangalore

Other Options

[Change Password](#)

[My Trips](#)

[Search Train](#)

[Update Profile](#)

First and Last Name changed in database

Result Grid								
<input type="button" value="Edit"/> <input type="button" value="New"/> <input type="button" value="Delete"/> <input type="button" value="Import"/> <input type="button" value="Export"/> <input type="button" value="Wrap Cell Content"/>								
	id	address	email	first_name	last_name	password	phone_number	username
▶	1	Karnataka	dws414@gmail.com	Shreya	Mishra	shreya	343876721	ishreya09
	2	Bangalore	shreya.pes@gmail.com	SHREYA	MISHRA PES1UG21CS574	shreya	3727361221	shreya
	3	Bangalore	cc@gmail.com	Cloud Computing	Google Backup	cc	121135464	cc
	4	Bangalore	abc@gmail.com	abc	abc	abc	123454535454	abc
*	5	Bangalore	ood@gmail.com	Shreya	Mishra	ood	9999923421	ood
	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Viewing Train Schedules

Book On Rails | Home | Dashboard | Logout | Search Train | Train Schedules | About Us | Contact Us

Select Train

Choose a train...

[Show Schedule](#)

Viewing Schedule for a selected train

Book On Rails | Home | Dashboard | Logout | Search Train | Train Schedules | About Us | Contact Us

12844 : Ranchi Yashwantpur Shatabdi

Train No: 12844

Train Name: Ranchi Yashwantpur Shatabdi

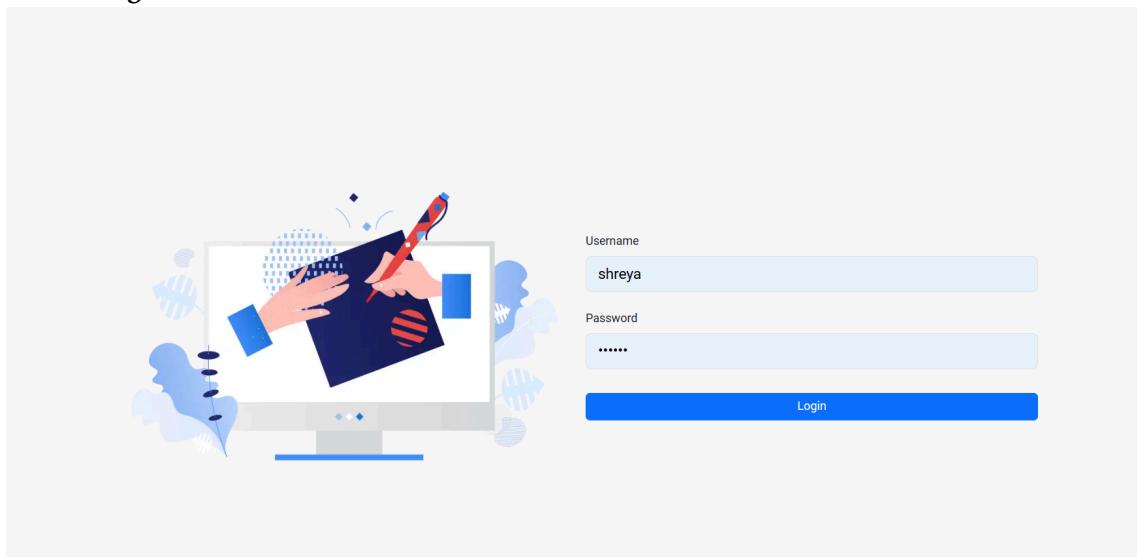
Train Type: Shatabdi

Operating Days: Tuesday Monday

Station Code	Station Name	Arrival Time	Departure Time	Distance to travel for next Station
HTE	Hatia (Ranchi)	10:30:00	10:45:00	45.0
RNC	Ranchi	01:30:00	01:45:00	120.0
MURI	Muri	07:20:00	07:45:00	0.0
RKL	Rourkela	11:30:00	11:45:00	120.0
YPR	Yashwantpur Junction	22:30:00	22:45:00	200.0

Admin side

Admin Login



Admin management

The image shows the Admin Dashboard. At the top, there's a yellow header bar with 'Book On Rails' and 'Dashboard'. Below it is a light gray sidebar with various management options. A blue button labeled 'Admin Management' is highlighted. Other options include 'User Management', 'Train Management', 'Route/Schedule Management', 'Ticket Management', 'Seat Availability Management', 'Station Management', 'Stops and Time Management for a Route', 'Train's Operating Day Management', and 'Contact Us Query'. To the right of the sidebar are two green buttons: 'Add Admins' and 'View Admins'.

View Admin

Add Admin		
Search for usernames...		
Admin Id	Username	Action
1	shreya	<button>Delete</button>
2	shreya123	<button>Delete</button>
4	ishreya09	<button>Delete</button>
5	adrija	<button>Delete</button>

Add Admin

Add Admin

Username
ooad

Password

Add Admin

ooad admin user added

Add Admin		
Search for usernames...		
Admin Id	Username	Action
1	shreya	<button>Delete</button>
2	shreya123	<button>Delete</button>
4	ishreya09	<button>Delete</button>
5	adrija	<button>Delete</button>
6	ooad	<button>Delete</button>

User management

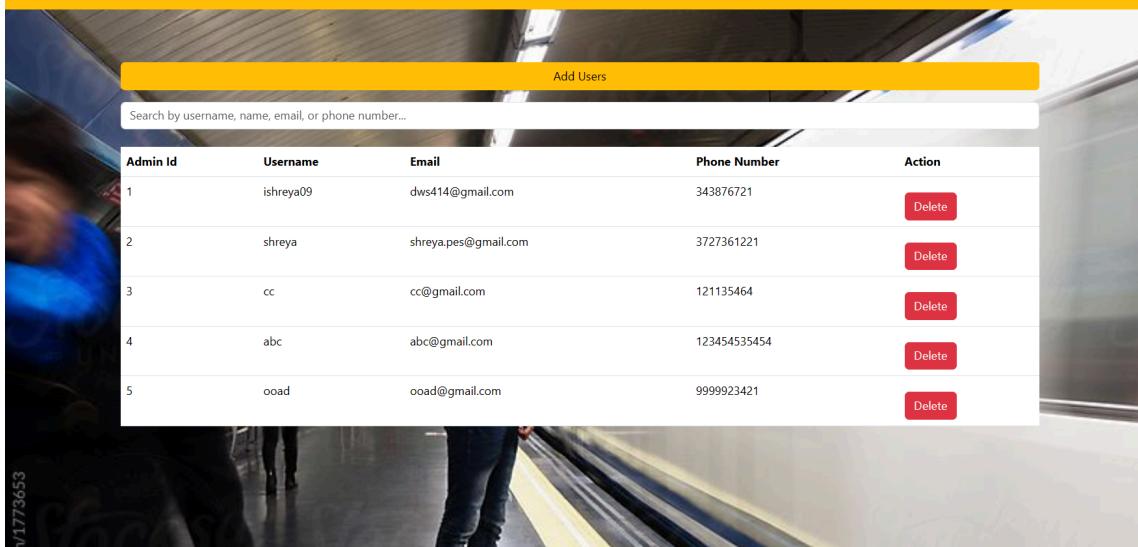
Book On Rails Dashboard

Admin Dashboard

- [Admin Management](#)
- [User Management](#)
- [Train Management](#)
- [Route/Schedule Management](#)
- [Ticket Management](#)
- [Seat Availability Management](#)
- [Station Management](#)
- [Stops and Time Management for a Route](#)
- [Train's Operating Day Management](#)
- [Contact Us Query](#)

View User

Book On Rails Dashboard Logout

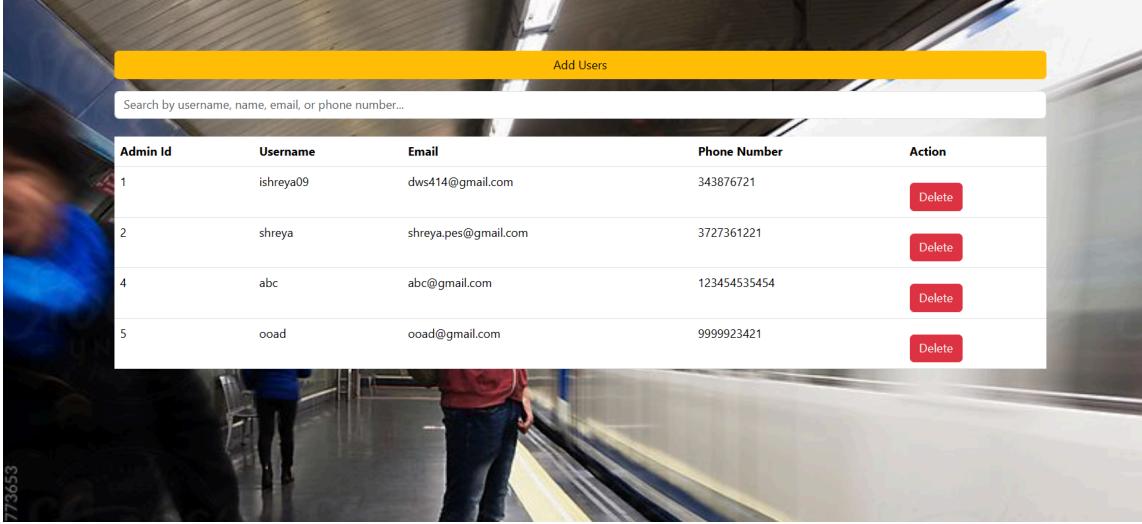


Add Users

Search by username, name, email, or phone number...

Admin Id	Username	Email	Phone Number	Action
1	ishreya09	dws414@gmail.com	343876721	Delete
2	shreya	shreya.pes@gmail.com	3727361221	Delete
3	cc	cc@gmail.com	121135464	Delete
4	abc	abc@gmail.com	123454535454	Delete
5	oad	oad@gmail.com	9999923421	Delete

Admin can delete a user in case of suspicious activity (We will delete cc)



Book On Rails Dashboard Logout

Add Users

Search by username, name, email, or phone number...

Admin Id	Username	Email	Phone Number	Action
1	ishreya09	dws414@gmail.com	343876721	<button>Delete</button>
2	shreya	shreya.pes@gmail.com	3727361221	<button>Delete</button>
4	abc	abc@gmail.com	123454535454	<button>Delete</button>
5	ooad	ooad@gmail.com	9999923421	<button>Delete</button>

Add user

Book On Rails Dashboard Logout

Add User

First Name: Shreja

Last Name: Rajesh

Email: shreeja@gmail.com

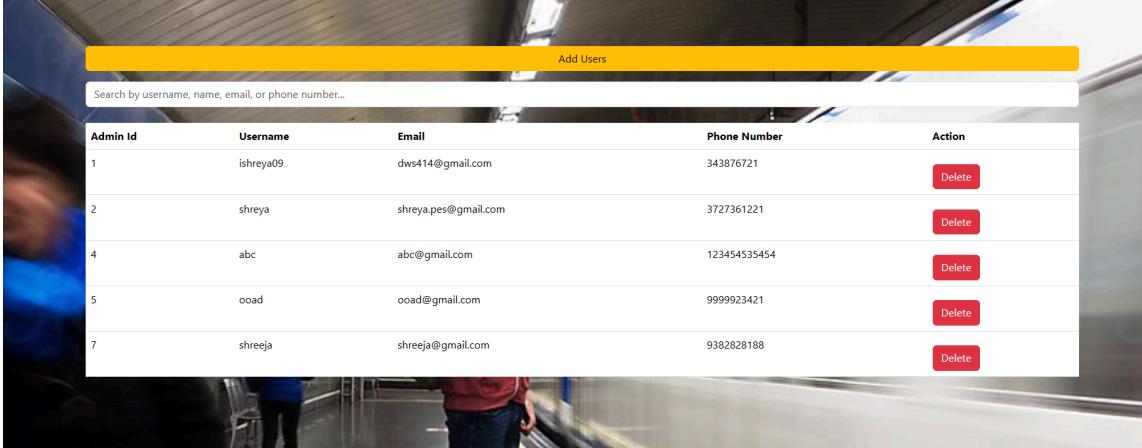
Phone Number: 9382828188

Address: Karnataka

Username: shreeja

Password:

User added



Book On Rails Dashboard Logout

Add Users

Search by username, name, email, or phone number...

Admin Id	Username	Email	Phone Number	Action
1	ishreya09	dws414@gmail.com	343876721	<button>Delete</button>
2	shreya	shreya.pes@gmail.com	3727361221	<button>Delete</button>
4	abc	abc@gmail.com	123454535454	<button>Delete</button>
5	ooad	ooad@gmail.com	9999923421	<button>Delete</button>
7	shreeja	shreeja@gmail.com	9382828188	<button>Delete</button>

Train management

Book On Rails Dashboard

Admin Dashboard

Admin Management

User Management

Train Management

Route/Schedule Management

Ticket Management

Seat Availability Management

Station Management

Stops and Time Management for a Route

Train's Operating Day Management

Contact Us Query

Add Train

View Train

View Train

Book On Rails Dashboard Logout

Search trains

Train No	Train Name	Train Type	Action
12334	Ranchi Muri Express	Express	<button>Update</button> <button>Delete</button>
123455	Hyderabad Vishakhapatnam Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
12844	Ranchi Yashwantpur Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
21112	Hatia Pune Garib Rath	Garib Rath	<button>Update</button> <button>Delete</button>
63722	Hyderabad Bangalore Express	Express	<button>Update</button> <button>Delete</button>
67732	Hatia Hyderabad Express	Express	<button>Update</button> <button>Delete</button>

Update a train

Train No
67732

Train Name
Hatia Hyderabad Shatabdi

Train Type:
Shatabdi

Update Train

Last Train updated

Train No	Train Name	Train Type	Action
12334	Ranchi Muri Express	Express	<button>Update</button> <button>Delete</button>
123455	Hyderabad Vishakhapatnam Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
12844	Ranchi Yashwantpur Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
21112	Hatia Pune Garib Rath	Garib Rath	<button>Update</button> <button>Delete</button>
63722	Hyderabad Bangalore Express	Express	<button>Update</button> <button>Delete</button>
67732	Hatia Hyderabad Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>

Add Train

Using R11 route to add train now

[Book On Rails](#) [Dashboard](#) [Logout](#)

Add Train

Train Number	12345
Train Name	Mumbai Delli Rajdhani
Train Type	Rajdhani
Route	R11
<button>Add Train</button>	

Train Added (2nd row- List sorted by train no)

[Book On Rails](#) [Dashboard](#) [Logout](#)

Search trains

Train No	Train Name	Train Type	Action
12334	Ranchi Muri Express	Express	<button>Update</button> <button>Delete</button>
12345	Mumbai Dilli Rajdhani	Rajdhani	<button>Update</button> <button>Delete</button>
123455	Hyderabad Vishakhapatnam Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
12844	Ranchi Yashwantpur Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>
21112	Hatia Pune Garib Rath	Garib Rath	<button>Update</button> <button>Delete</button>
63722	Hyderabad Bangalore Express	Express	<button>Update</button> <button>Delete</button>
67732	Hatia Hyderabad Shatabdi	Shatabdi	<button>Update</button> <button>Delete</button>

Route/ Schedule Management

Book On Rails Dashboard

Admin Dashboard

Admin Management

User Management

Train Management

Route/Schedule Management

Ticket Management

Seat Availability Management

Station Management

Stops and Time Management
for a Route

Train's Operating Day
Management

Contact Us Query

Add Route

Show Routes

View Route

Book On Rails Dashboard Logout

Add Route

Search for Route Code...

Route ID	Route Code	Action
1	R01: HATIA MURI ROUTE	<button>Delete</button>
2	R02: HATIA YASHWANTPUR ROUTE	<button>Delete</button>
3	R03	<button>Delete</button>
4	R04	<button>Delete</button>
8	R05	<button>Delete</button>
10	R06	<button>Delete</button>

Add Route

Add Route

Route Code

R11

Add Route

Route Added

We will use the R11 route now to create a train.

Add Route

Search for Route Code...

Route ID	Route Code	Action
1	R01: HATIA MURI ROUTE	<button>Delete</button>
2	R02: HATIA YASHWANTPUR ROUTE	<button>Delete</button>
3	R03	<button>Delete</button>
4	R04	<button>Delete</button>
8	R05	<button>Delete</button>
10	R06	<button>Delete</button>
12	R11	<button>Delete</button>

Train's Operating Day Management :

Book On Rails Dashboard

Admin Dashboard

- [Admin Management](#)
- [User Management](#)
- [Train Management](#)
- [Route/Schedule Management](#)
- [Ticket Management](#)
- [Seat Availability Management](#)
- [Station Management](#)
- [Stops and Time Management for a Route](#)
- [Train's Operating Day Management](#)
- [Contact Us Query](#)

View Operating Days

Book On Rails Dashboard Logout

Add operating day for a train

Search for Train No...

Train No.	Train Name	Day of week	Action
67732	Hatia Hyderabad Shatabdi	Monday	Delete
67732	Hatia Hyderabad Shatabdi	Thursday	Delete
12334	Ranchi Muri Express	Tuesday	Delete
12334	Ranchi Muri Express	Saturday	Delete
12844	Ranchi Yashwantpur Shatabdi	Tuesday	Delete
12844	Ranchi Yashwantpur Shatabdi	Friday	Delete
12334	Ranchi Muri Express	Thursday	Delete

Can search by train no

Add operating day for a train

12844

Train No.	Train Name	Day of week	Action
12844	Ranchi Yashwantpur Shatabdi	Tuesday	Delete
12844	Ranchi Yashwantpur Shatabdi	Friday	Delete

Add operating day

Book On Rails · Dashboard · Logout

Enter train number:

12345

Select which days of the week the train will run:

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday

[Add operating days](#)

Operating days added

12844	Ranchi Yashwantpur Shatabdi	Tuesday	Delete
12844	Ranchi Yashwantpur Shatabdi	Friday	Delete
12334	Ranchi Muri Express	Thursday	Delete
12345	Mumbai Dilli Rajdhani	Monday	Delete
12345	Mumbai Dilli Rajdhani	Wednesday	Delete
12345	Mumbai Dilli Rajdhani	Friday	Delete

Station Management

Book On Rails · Dashboard

Admin Dashboard

- [Admin Management](#)
- [User Management](#)
- [Train Management](#)

[Add Station](#)

Route/Schedule Management

- [Ticket Management](#)
- [Seat Availability Management](#)
- [Station Management](#)
- [Stops and Time Management for a Route](#)
- [Train's Operating Day Management](#)
- [Contact Us Query](#)

[View Station](#)

View Stations

Book On Rails Dashboard Logout

Add Station		
Stations		
Station Code	Station Name	Actions
BLR	Bangalore	<button>Update</button> <button>Delete</button>
HTE	Hatia (Ranchi)	<button>Update</button> <button>Delete</button>
HYD	Hyderabad	<button>Update</button> <button>Delete</button>
KGI	Kengiri	<button>Update</button> <button>Delete</button>
KRP	Krishnarajapuram	<button>Update</button> <button>Delete</button>
MURI	Muri	<button>Update</button> <button>Delete</button>
PUNE	Pune	<button>Update</button> <button>Delete</button>
RKL	Rourkela	<button>Update</button> <button>Delete</button>

Add Station

Add Station

Station Code

CST

Station Name

Mumbai-CST

Add Station

Add Station

Station Code

NDLS

Station Name

New Delhi Railway Station

Add Station

Book On Rails Dashboard Logout

Add Station

Station Code

ST

Station Name

Surat

Add Station

Add Station

Station Code	BR
Station Name	Varodara
Add Station	

Add Station

Station Code	KOTA
Station Name	Kota Jn
Add Station	

All stations Added

The screenshot shows a dashboard titled "Book On Rails" with "Dashboard" and "Logout" links at the top. A search bar is present. Below it is a table titled "Stations" with columns for "Station Code" and "Station Name". Each row has "Actions" buttons for "Update" and "Delete". The data in the table is as follows:

Station Code	Station Name	Actions
BLR	Bangalore	Update Delete
BR	Varodara	Update Delete
CST	Mumbai-CST	Update Delete
HTE	Hatia (Ranchi)	Update Delete
HYD	Hyderabad	Update Delete
KGI	Kengiri	Update Delete
KOTA	Kota Jn	Update Delete
KRP	Krishnarajapuram	Update Delete
MURI	Muri	Update Delete
NDLS	New Delhi Railway Station	Update Delete
PUNE	Pune	Update Delete
RKL	Rourkela	Update Delete
RNC	Ranchi	Update Delete
ST	Surat	Update Delete
VJW	Vijayawada	Update Delete
VSK	Vishakhapatnam	Update Delete
YPR	Yashwantpur Junction	Update Delete

Stops and Time Management for a Route

The screenshot shows a web-based administrative interface. At the top, there is a yellow header bar with the text "Book On Rails" and "Dashboard". Below this is a white sidebar on the left containing a list of management categories: "Admin Dashboard", "Admin Management", "User Management", "Train Management", "Route/Schedule Management", "Ticket Management", "Seat Availability Management", "Station Management", "Stops and Time Management for a Route" (which is highlighted with a blue background), "Train's Operating Day Management", and "Contact Us Query". To the right of the sidebar, there are two green rectangular buttons: "Add Station and Timing for a Route" and "View Station and Timings Available". The main area of the page is currently empty.

Add Stops for a train

Book On Rails · Dashboard · Logout

Train Station Details

Train Number:

12345

Add Stop Station:

CST: Mumbai-CST

Arrival Time:

09:05



Departure Time:

09:20



200

Add Stop Station:

ST: Surat

Arrival Time:

13:50



Departure Time:

14:20



100

Add Stop Station:

BRC: Vadodara

Arrival Time:

17:20



Departure Time:

17:30



200

Add Stop Station:

KOTA: Kota Jn

Arrival Time:

22:50



Departure Time:

23:00



100

Add Stop Station:

NLDS: New Delhi Railway Station

Arrival Time:

00:50



Departure Time:

01:20



0

[Add Stop](#) [Submit](#)

Stops for our train added

Add Stops and Timings for a train

Train Number	Train Name	Route Code	Action
12334	Ranchi Muri Express	R01: HATIA MURI ROUTE	View Complete Route
12345	Mumbai Dilli Rajdhani	R11	View Complete Route
123455	Hyderabad Vishakhapatnam Shatabdi	R06	View Complete Route
12844	Ranchi Yashwantpur Shatabdi	R02: HATIA YASHWANTPUR ROUTE	View Complete Route
21112	Hatia Pune Garib Rath	R03	View Complete Route
63722	Hyderabad Bangalore Express	R05	View Complete Route
67732	Hatia Hyderabad Shatabdi	R04	View Complete Route

View Route for the added train

[Book On Rails](#) | [Dashboard](#) | [Logout](#)

12345 : Mumbai Dilli Rajdhani

Station Code	Station Name	Arrival Time	Departure Time	Distance to travel for next Station	Action
CST	Mumbai-CST	09:05:00	09:20:00	200.0	Update Delete
ST	Surat	13:50:00	14:20:00	100.0	Update Delete
BRC	Vadodara	17:20:00	17:30:00	200.0	Update Delete
KOTA	Kota Jn	22:50:00	23:00:00	100.0	Update Delete
NDLS	New Delhi Railway Station	00:50:00	01:20:00	0.0	Update Delete

[Add New Stop](#)

Seat Availability Management

[Book On Rails](#) | [Dashboard](#)

Admin Dashboard

[Admin Management](#)
[User Management](#)
[Train Management](#)

[Route/Schedule Management](#)

[Ticket Management](#)
[Seat Availability Management](#)
[Station Management](#)
[Stops and Time Management
for a Route](#)
[Train's Operating Day
Management](#)
[Contact Us Query](#)

[Add Seat Availability](#)

[View Seat Availability](#)

View Seat Availability (Shows only for days after today)

[Book On Rails](#) | [Dashboard](#) | [Logout](#)

[Add Seat Availability](#)

Search by Train No...

Search by Class Type...

Seat Availability ID	Date	Train No	Train Name	Class Type	No of Coaches	Available Seats	Waiting List count	Base Price	Fare per Km Price	Senior Citizen Discount	Cancellation Charge	Action
14	2024-04-23	12844	Ranchi Yashwantpur Shatabdi	AC3Tier	1	24	2	100.0	20.0	50.0	50.0	Delete

Add Seat Availability

Book On Rails | Dashboard | Logout

Seat Availability Form

Operating Days: Monday,Wednesday,Friday

Train Number:

12345

Date:

24-04-2024

Coach Type:

Sleeper

Number of coaches:

5

Base Price:

100

Fare Per KM:

10

Senior Citizen Discount:

50

Cancellation Charge:

100

[Submit](#)

Seat Availability added

Add Seat Availability

Search by Train No...

Search by Class Type...

Seat Availability ID	Date	Train No	Train Name	Class Type	No of Coaches	Available Seats	Waiting List count	Base Price	Fare per Km Price	Senior Citizen Discount	Cancellation Charge	Action
14	2024-04-23	12844	Ranchi Yashwantpur Shatabdi	AC3Tier	1	24	2	100.0	20.0	50.0	50.0	Delete
15	2024-04-24	12345	Mumbai Dilli Rajdhani	Sleeper	5	150	0	100.0	10.0	50.0	100.0	Delete

Ticket Management

Book On Rails | Dashboard

Admin Dashboard

Admin Management

[Show Tickets](#)

User Management

[View Cancelled Tickets](#)

Train Management

[View Waiting List Tickets](#)

Route/Schedule Management

Ticket Management

Seat Availability Management

Station Management

Stops and Time Management
for a Route

Train's Operating Day
Management

Contact Us Query

Show Tickets

Ticket Details

PNR	Username	Train No	Date	Class Type	Payment Status	Ticket Status	Total Amount Paid	Action
PNR40c41b	shreya	12844	2024-04-19	AC2Tier	Success	Confirmed	25300.0	<button>Delete</button>
PNRdb499d	shreya	12844	2024-04-19	AC2Tier	Success	Confirmed	25700.0	<button>Delete</button>
PNR2eb927	shreya	12844	2024-04-23	AC3Tier	Success	Cancelled	12350.0	<button>Delete</button>
PNR579010	shreya	12844	2024-04-23	AC3Tier	Success	Confirmed	5550.0	<button>Delete</button>

Show Cancelled Tickets

Book On Rails Dashboard Logout

Ticket Details

PNR	Username	Train No	Date	Class Type	Payment Status	Ticket Status	Total Amount Paid	Action
PNR2eb927	shreya	12844	2024-04-23	AC3Tier	Success	Cancelled	12350.0	<button>Delete</button>

Show Waiting Tickets (in this case : None)

Book On Rails Dashboard Logout

Ticket Details

PNR	Username	Train No	Date	Class Type	Payment Status	Ticket Status	Total Amount Paid	Action

Contact Us Query :

Book On Rails Dashboard

Admin Dashboard

[Admin Management](#)

[User Management](#)

[Train Management](#)

[Route/Schedule Management](#)

[Ticket Management](#)

[Seat Availability Management](#)

[Station Management](#)

[Stops and Time Management
for a Route](#)

[Train's Operating Day
Management](#)

[View Contact us Query](#)

[Contact Us Query](#)

View Contact us

[Book On Rails](#) | [Dashboard](#) | [Logout](#)

Name	Email	Query	Action
PESTUG21CS574 Shreya Mishra	shreya.pes@outlook.com	hey	Send Email
sgd	dsb@gxd.com	bshfbewhb	Send Email
Shreya Mishra	shreyamishra414@gmail.com	Hello, this is regarding train no 12844	Send Email

View Feedbacks

[Book On Rails](#) | [Dashboard](#) | [Logout](#)

Username	PNR	Train No	Rating	Feedback	Action
shreya	PNRdb499d	12844	4	good	Send Email
shreya	PNR40c41b	12844	2	okayish	Send Email
shreya	PNRdb499d	12844	4	Food was awesome	Send Email

Database After Changes

User

Result Grid									Filter Rows:		Edit:	Export/Import:	Wrap Cell Content:
	id	address	email	first_name	last_name	password	phone_number	username					
▶	1	Karnat	Resets all sorted columns	Shreya	Mishra	shreya	343876721	ishreya09					
	2	Bangalore	shreya.pes@gmail.com	SHREYA	MISHRA PES1UG21CS574	shreya	3727361221	shreya					
	4	Bangalore	abc@gmail.com	abc	abc	abc	123454535454	abc					
	5	Bangalore	ooad@gmail.com	Shreya	Mishra	ooad	9999923421	ooad					
*	7	Karnataka	shreeja@gmail.com	Shreeja	Rajesh	shreeja	9382828188	shreeja					
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL					

Admin

Result Grid			Filter Rows:	Edit:	Exp
	id	password	username		
▶	1	shreya	shreya		
	2	shreya123	shreya123		
	4	shreya	ishreya09		
	5	adrija	adrija		
	6	ooad	ooad		
*	HULL	HULL	HULL		

Contact

Result Grid				Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	id	email	message	name			
▶	1	shreya.pes@outlook.com	hey	PES1UG21CS574	Shreya Mishra		
	2	dsb@gxd.com	bshfbewhb	sgd			
*	3	shreyamishra414@gmail.com	Hello, this is regarding train no 12844	Shreya Mishra			
	NULL	NULL	NULL	NULL	NULL		

Train

Result Grid				Filter Rows:	Edit:	Export/Import:
	train_no	train_name	train_type	route_id		
▶	12334	Ranchi Muri Express	Express	1		
	12345	Mumbai Dilli Rajdhani	Rajdhani	12		
	123455	Hyderabad Vishakhapatnam Shatabdi	Shatabdi	10		
	12844	Ranchi Yashwantpur Shatabdi	Shatabdi	2		
	21112	Hatia Pune Garib Rath	Garib Rath	3		
	63722	Hyderabad Bangalore Express	Express	8		
*	67732	Hatia Hyderabad Shatabdi	Shatabdi	4		
	NULL	NULL	NULL	NULL		

Route

Result Grid		Filter Rows:	Edit
	id	route_code	
▶	1	R01: HATIA MURI ROUTE	
	2	R02: HATIA YASHWANTPUR ROUTE	
	3	R03	
	4	R04	
	8	R05	
	10	R06	
*	12	R11	
	NULL	NULL	

Operating Day

Result Grid				Filter Rows:	Edit:	Export/Import:
	id	day_of_week	train_no			
	6	Tuesday	12844			
	7	Friday	12844			
	8	Thursday	12334			
	12	Monday	12345			
	13	Wednesday	12345			
*	14	Friday	12345			
	NULL	NULL	NULL			

Station Timings

	id	arrival_time	departure_time	distance_from_next_station	route_id	station_code
*	23	10:00:00.000000	10:30:00.000000	0	8	BLR
	25	09:05:00.000000	09:20:00.000000	200	12	CST
	26	13:50:00.000000	14:20:00.000000	100	12	ST
	27	17:20:00.000000	17:30:00.000000	200	12	BRC
	28	22:50:00.000000	23:00:00.000000	100	12	KOTA
	29	00:50:00.000000	01:20:00.000000	0	12	NDLS
*	HULL	HULL	HULL	HULL	HULL	HULL

Station

	station_code	station_name
	BRC	Varodara
	CST	Mumbai-CST
	HTE	Hatia (Ranchi)
	HYD	Hyderabad
	KGI	Kengiri
	KOTA	Kota Jn
	KRP	Krishnarajapuram

Seat Availability

	id	available_seats	base_price	cancellation_charge	classes	date	fare_perkm	last_lower_seat	last_unbooked_lower_seat	last_
*	11	30	100	50	Sleeper	2024-04-19	20	29	1	2
	12	30	100	50	AC3Tier	2024-04-19	20	29	1	2
	13	0	100	50	AC2Tier	2024-04-19	20	9	11	12
	14	22	100	50	AC3Tier	2024-04-23	20	29	9	10
*	15	150	100	100	Sleeper	2024-04-24	10	149	1	2
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

	cm	last_lower_seat	last_unbooked_lower_seat	last_unbooked_upper_seat	last_upper_seat	no_of_coaches	senior_citizen_discount	train_no
*	29	1	2	30	1	50	12844	
	29	1	2	30	1	50	12844	
	9	11	12	10	1	50	12844	
	29	9	10	30	1	50	12844	
*	149	1	2	150	5	50	12345	
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Ticket

	ticket_id	pnr	waiting_list_number	classes	date	foodprice	payment_status	quantity	status	total_amount	veg	war
*	19	PNR40c41b	-1	AC2Tier	2024-04-19	800	Success	2	Confirmed	24500	0	1
	20	PNRdb499d	-1	AC2Tier	2024-04-19	1200	Success	4	Confirmed	24500	1	1
	21	PNR2eb927	-1	AC3Tier	2024-04-23	800	Success	2	Cancelled	11550	0	1
*	22	PNR579010	-1	AC3Tier	2024-04-23	600	Success	2	Confirmed	4950	1	1
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Ticket_Passengers

	ticket_ticket_id	passenger_id
	20	40
	20	41
	21	42
	21	22
	22	43
	22	44
	22	45

Passenger

	id	age	birthpreference	gender	is_senior_citizen	is_waiting_list	name	seat_no	ticket_id	user_id	coach_no
	40	32	Lower	Female	0	0	p4	11	20	2	12
	41	5	Lower	Female	0	0	p5	11	20	2	12
	42	12	Lower	Female	0	0	Shre...	5	21	2	6
	43	21	Upper	Female	0	0	adrija	6	21	2	7
	44	20	Lower	Female	0	0	p1	7	22	2	8
	45	30	Upper	Female	0	0	p2	8	22	2	9
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Payment

	id	status	ticket_id
	12	Success	19
	13	Success	20
	14	Success	21
	15	Success	22
*	NULL	NULL	NULL

Feedback

	id	comment	rating	ticket_id	user_id
	1	good	4	20	2
	2	okayish	2	19	2
	3	Food was awesome	4	20	2

Cancelled_seats

	seat_availability_id	seat_number
	5	2
	14	1
	14	2