Enterprise Application Development (EAD)

SE4040

Group Assignment

B.Sc. (Hons) in Information Technology

Field of specialization: Software Engineering

Sri Lanka Institute of Information Technology

October 2023

Table of Contents

Tabel of Figures

1 Individual Contribution

IT Number	Name	Individual Contribution
IT20233358	Diunugala M. W.	Traveler Management with Sign in, Sign up and user profile. • Create, update, view and delete traveler profiles. • View each traveler details. • Activate and Deactivate user profiles.
IT20076566	De Silva G.K.S	Train Management • Creating new train schedules. • Updating existing train schedules. • Canceling train schedules. • View train schedules • View each train schedules
IT20152864	Perera K.A.P.M	Ticket Booking Management • Creating new reservations • Updating reservations • Cancelling reservations • View all traveler reservations. • View own reservations. • View each reservation. • Manage traveler's reservations
IT20120634	Ariyarathna H.M.M.M	Mobile Application • Create sign in, sign up and user profile with deactivate option. • Create, update, view and delete your own reservations. • View reservation history. • View train schedules.

2 High Level Diagram

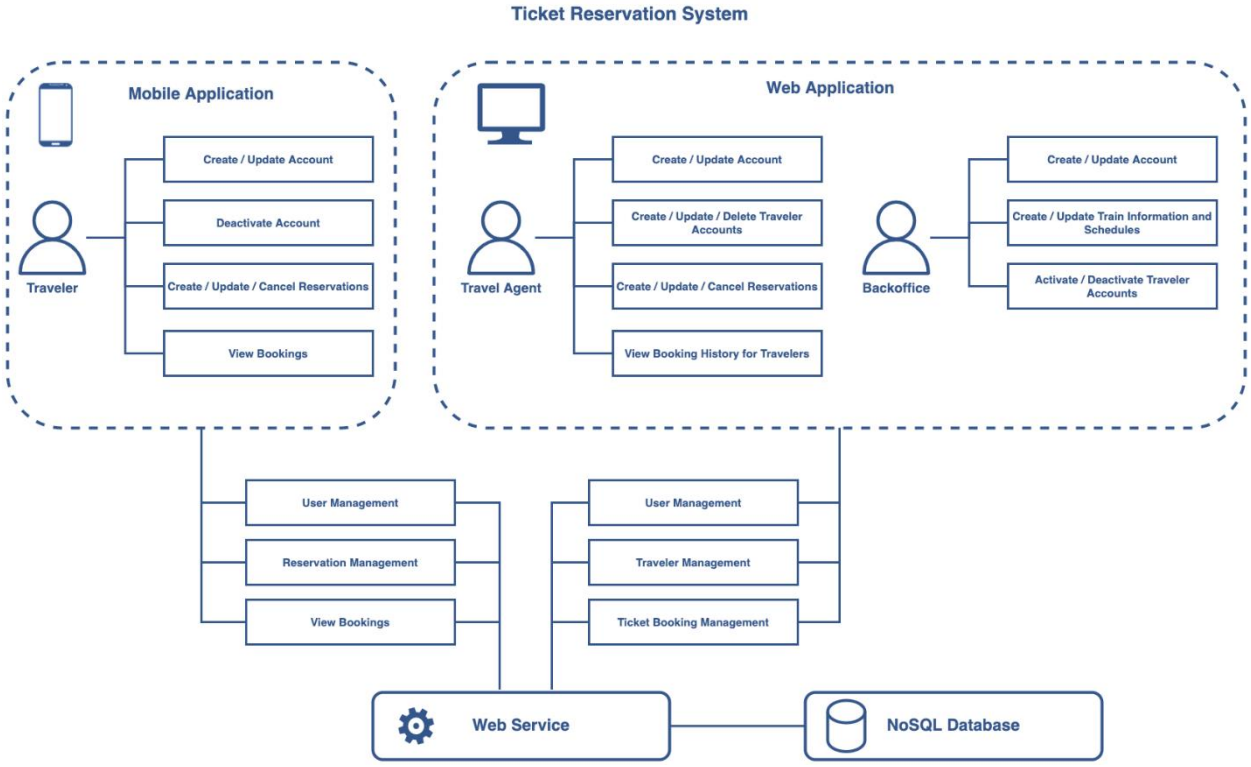

Figure 2.1: High level Diagram

3 Use Case Diagram

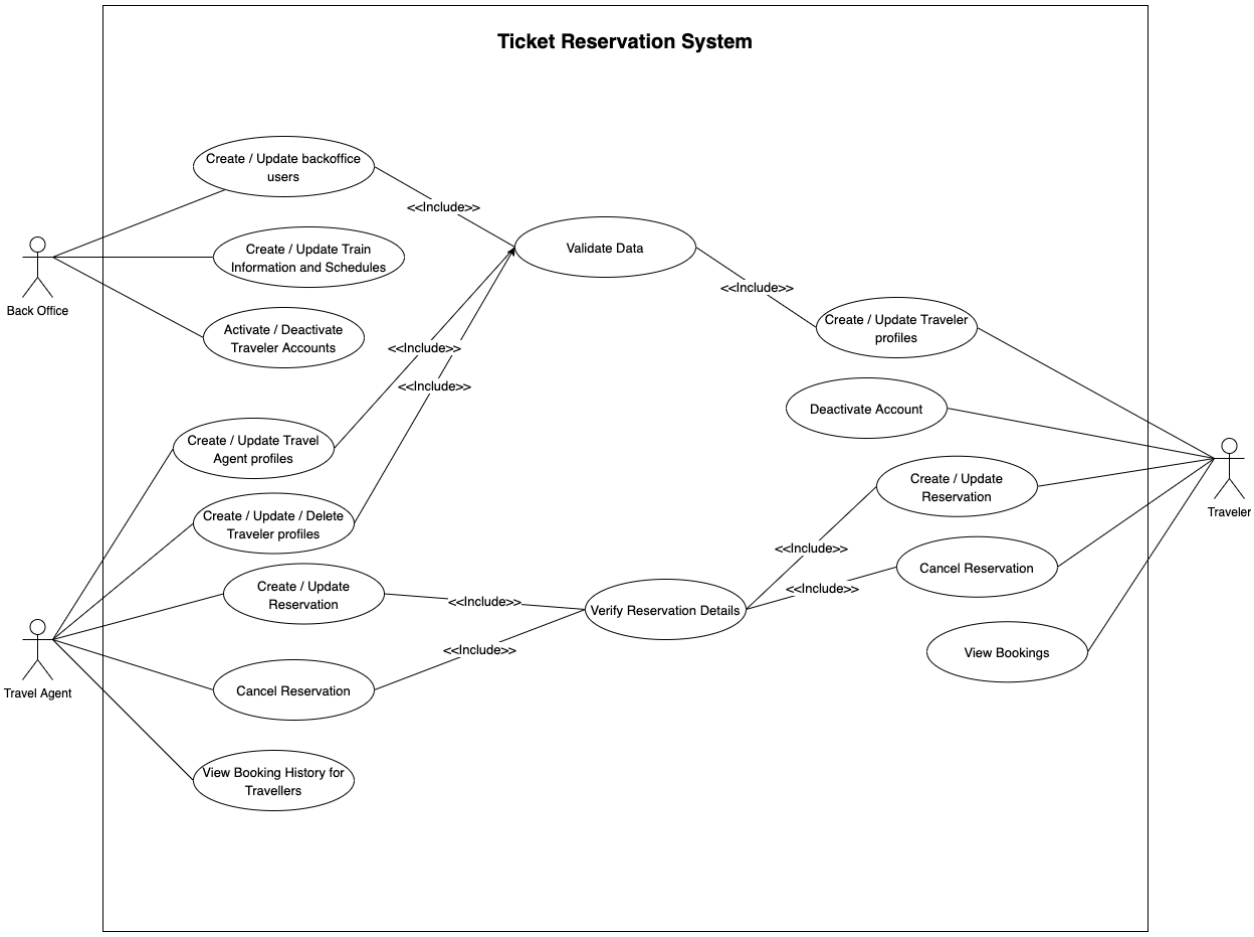

Figure 3.1: Use Case Diagram

4 DFD Diagram

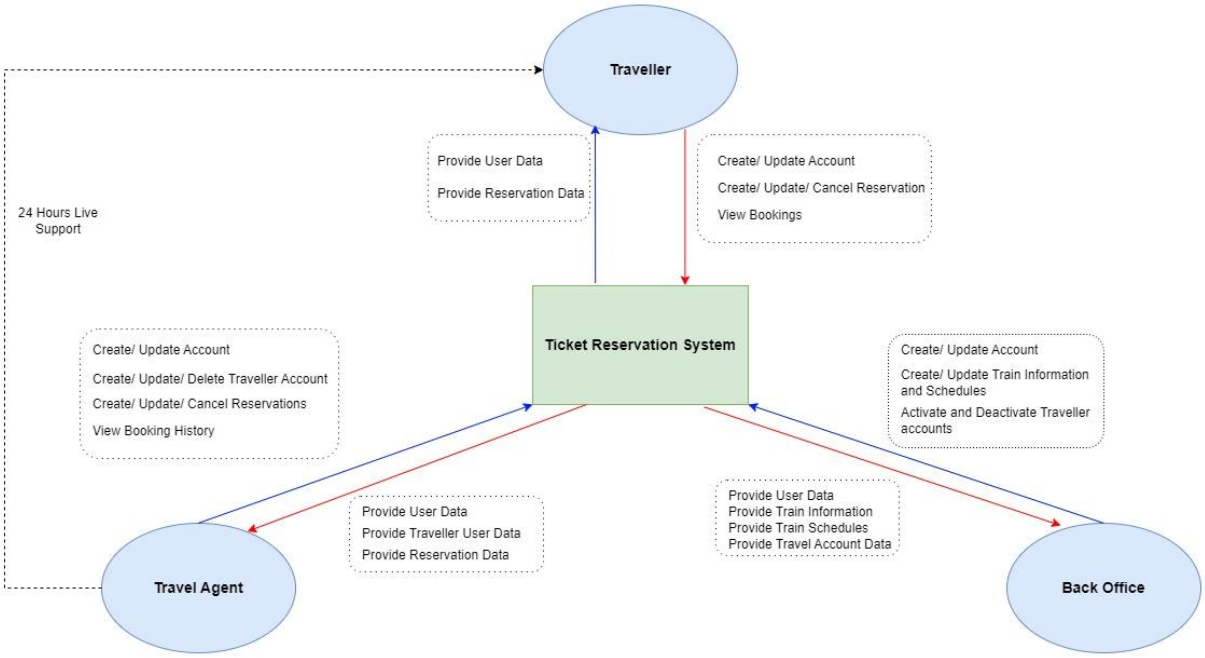

Figure 4.1: DFD Diagram

5 Database Design

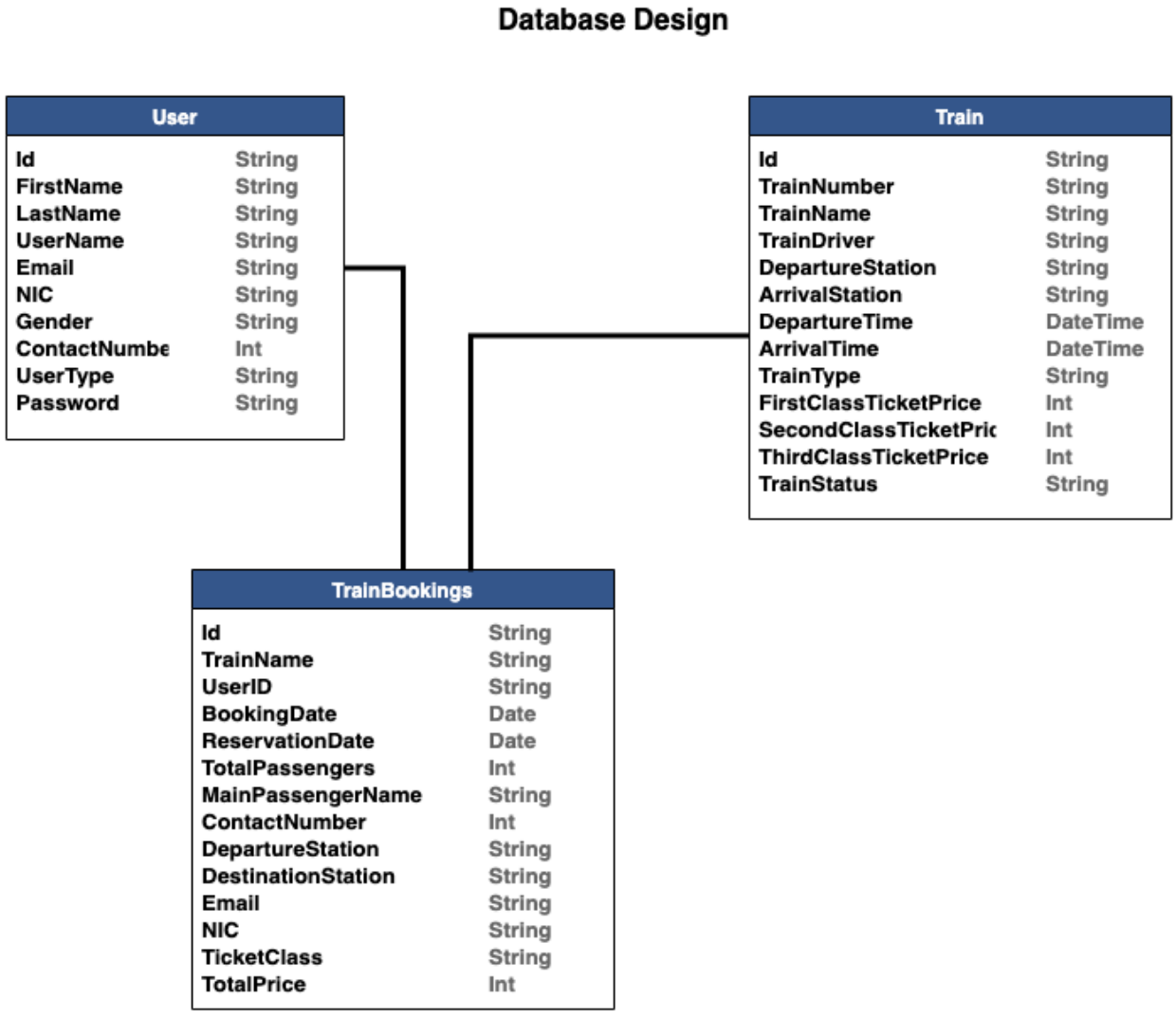

Figure 5.1: Database Design

Database Collections

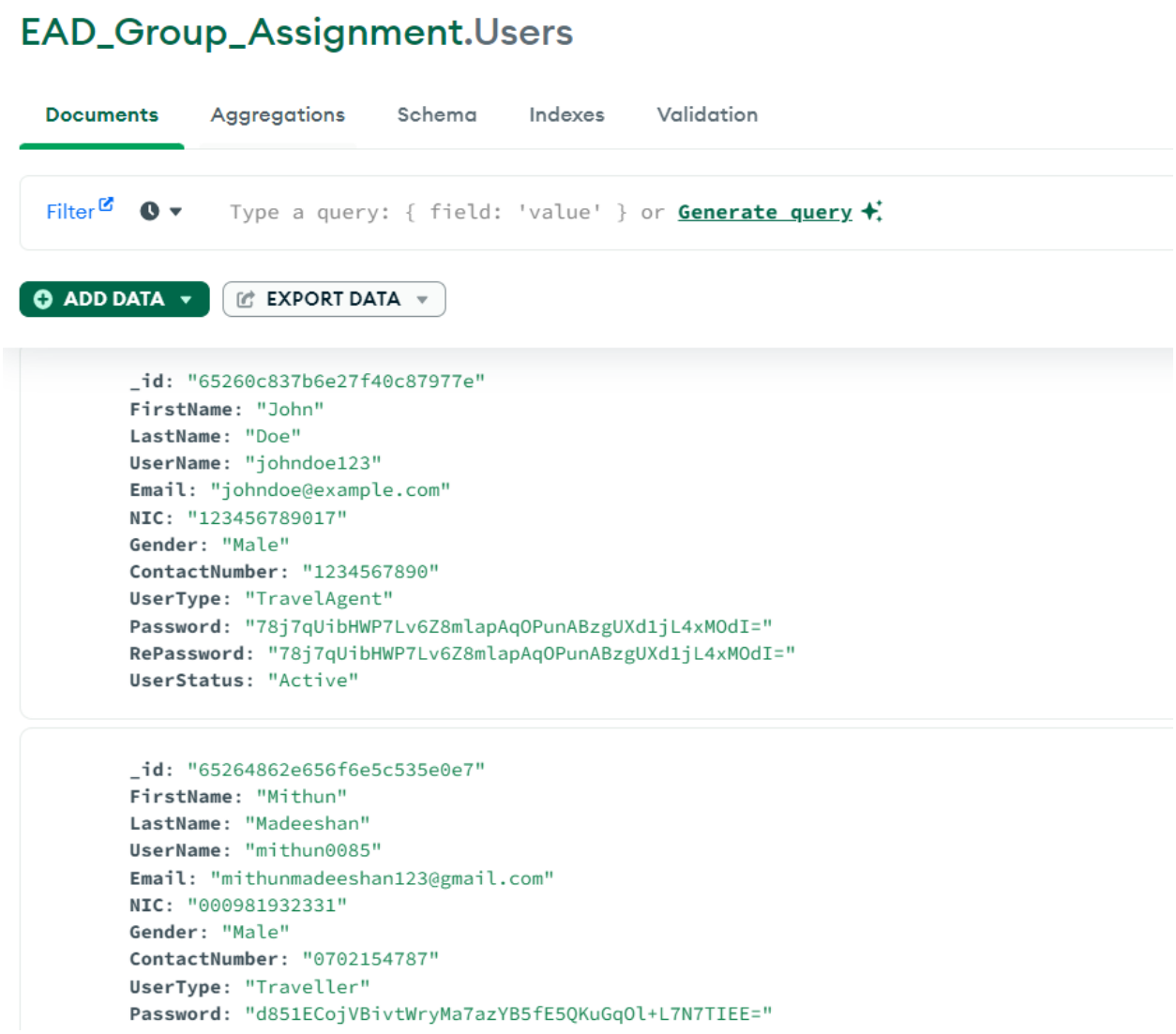

Figure 5.2: Users Collection

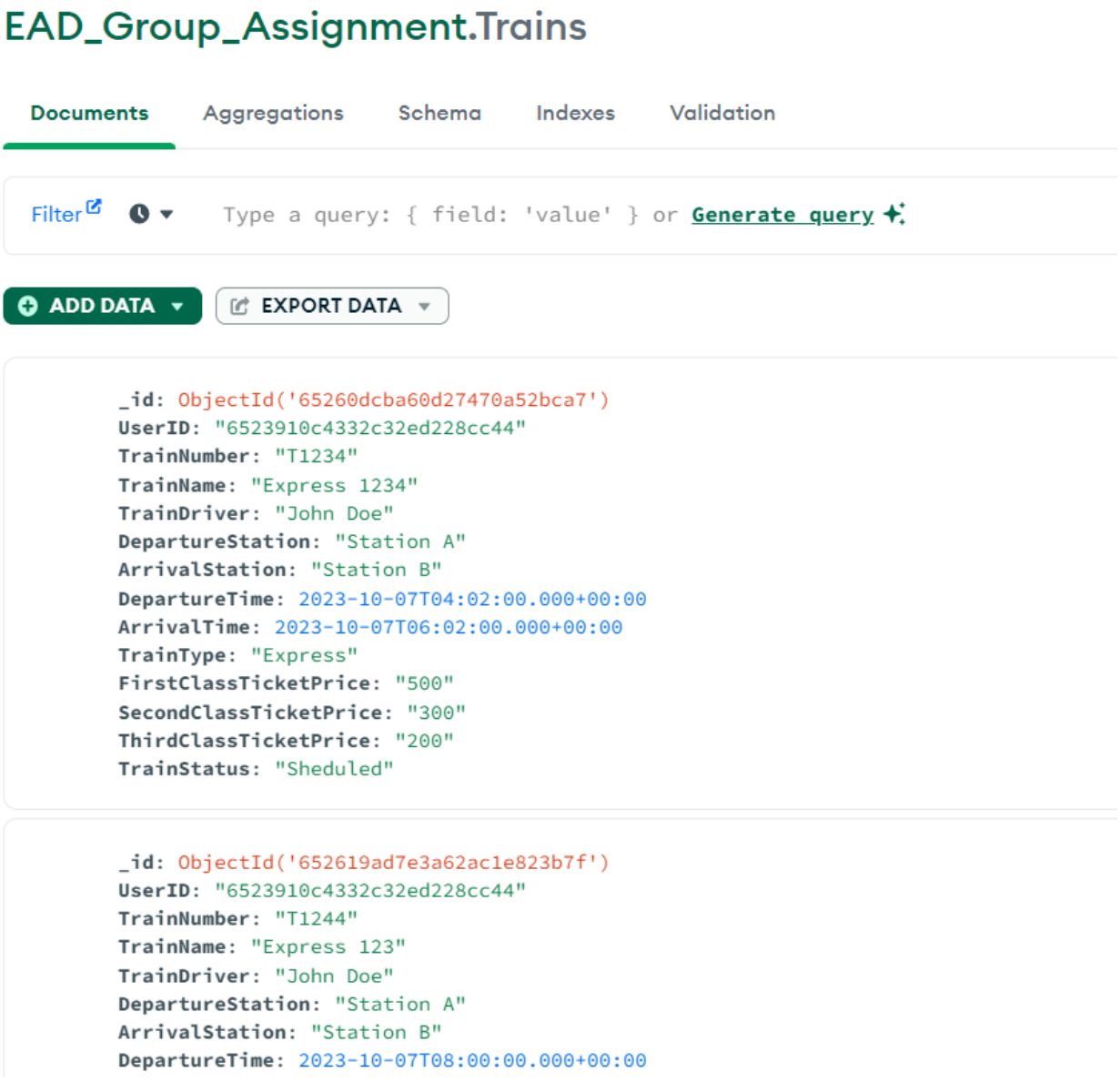

Figure 5.3: Trains Collections

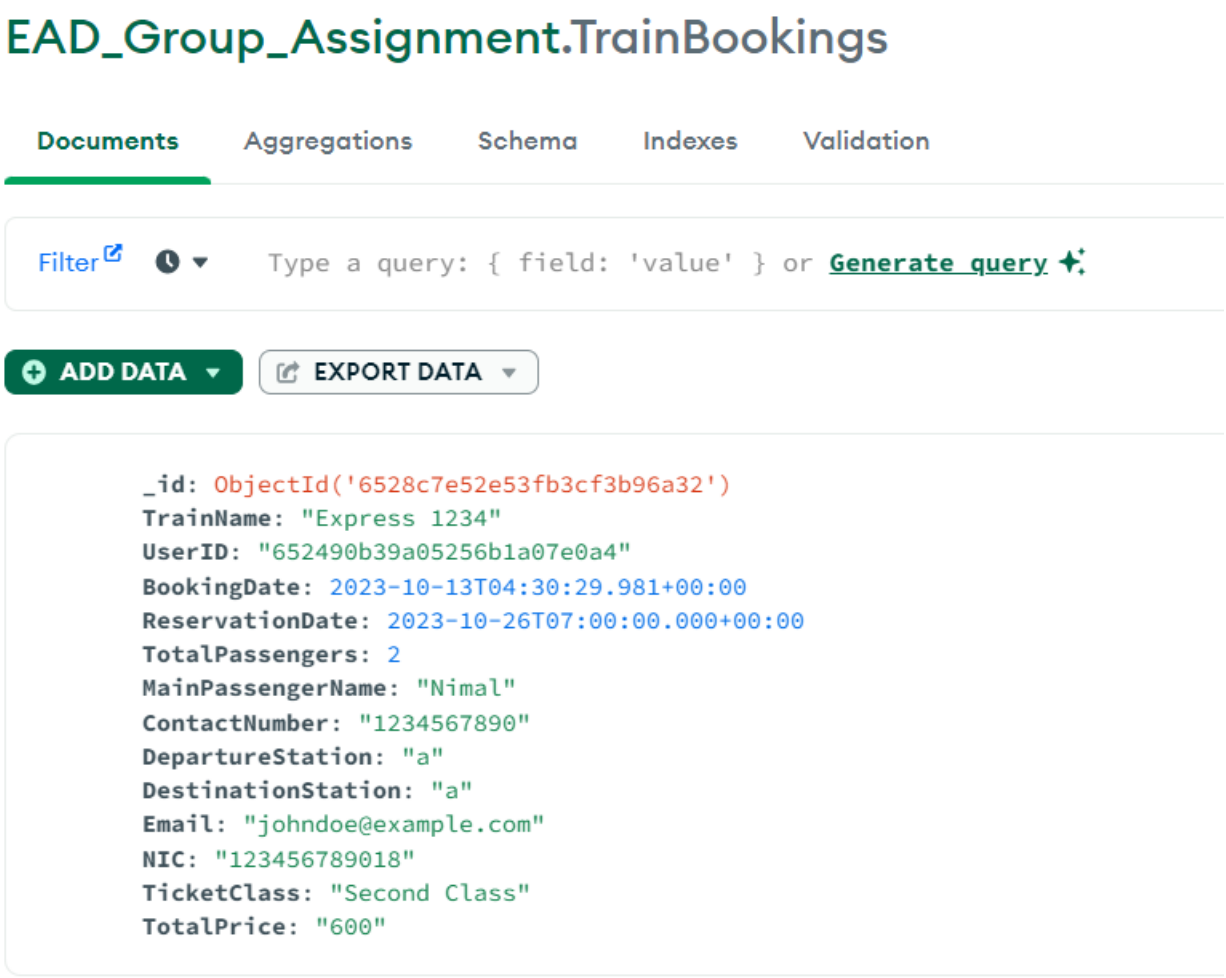

Figure 5.4: Train Bookings Collections

6 Screenshots of all UI's

6.1 Web Application

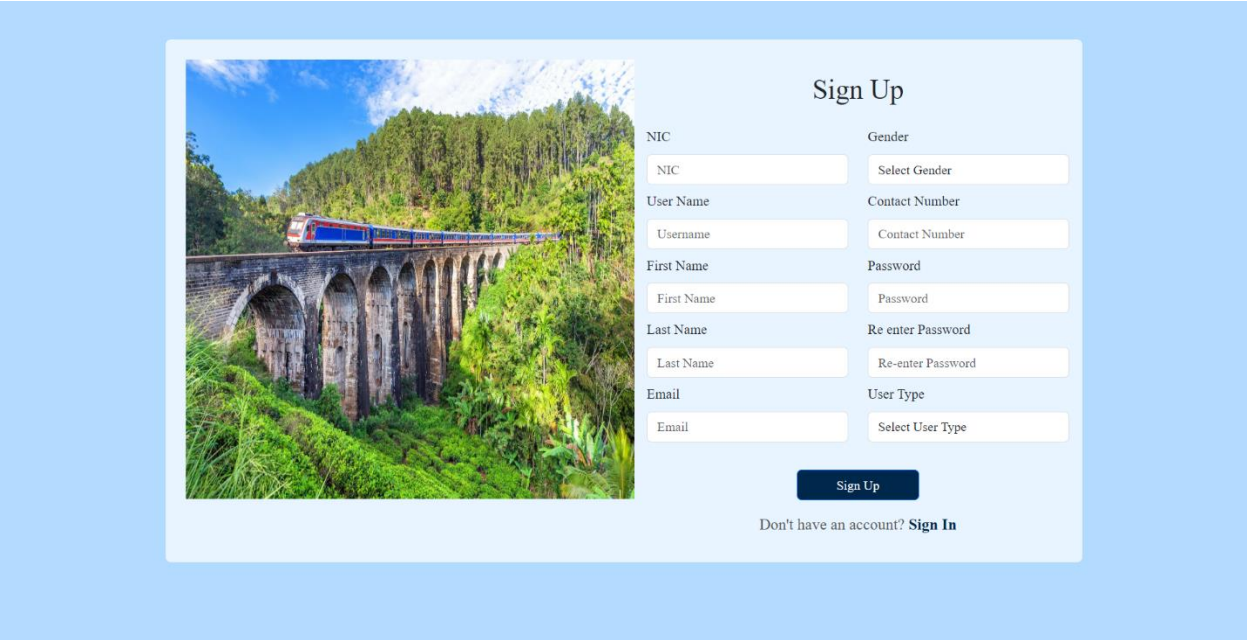

Figure 6.1: Sign Up UI

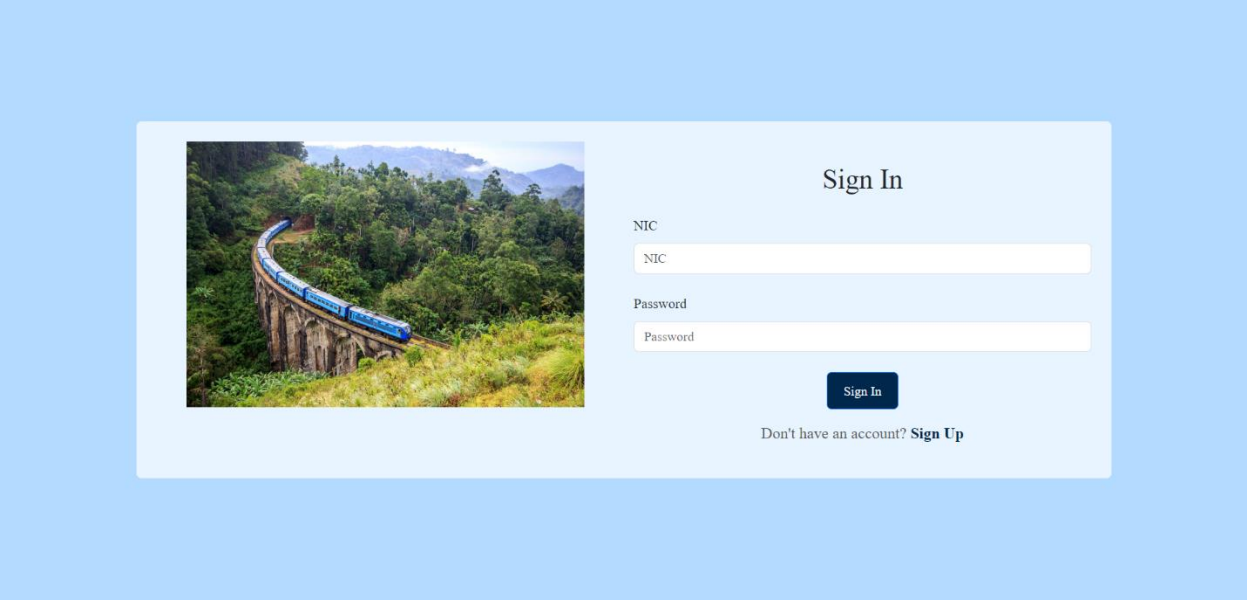

Figure 6.2: Sign In UI

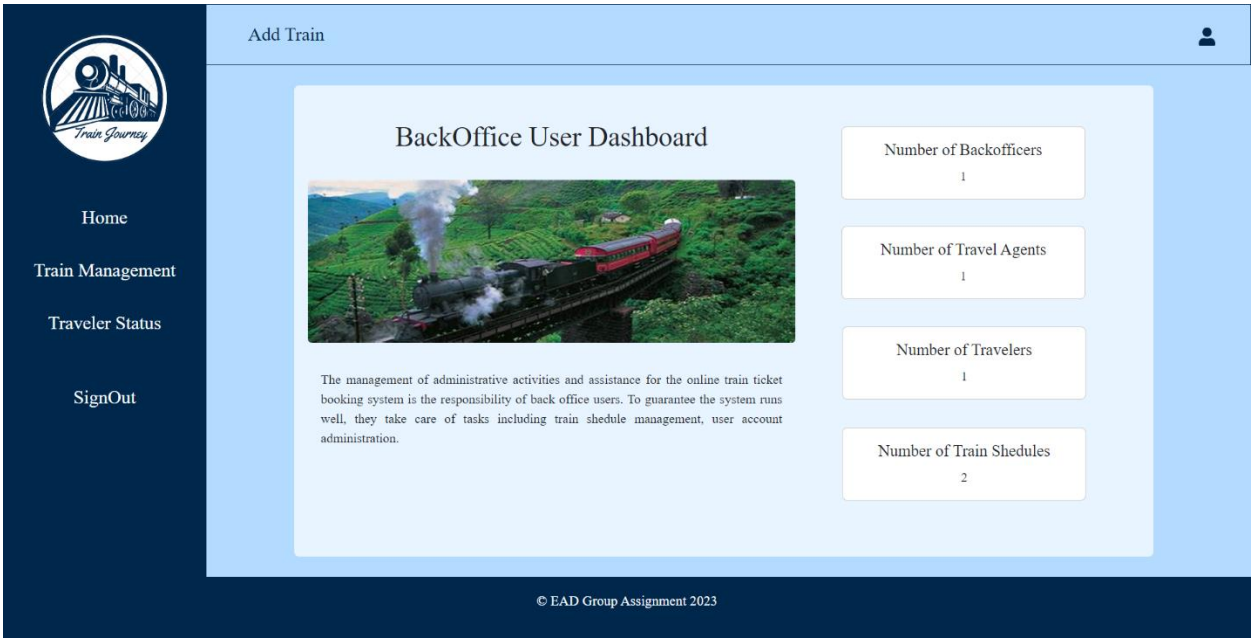

Figure 6.3: Backoffice User Dashboard UI

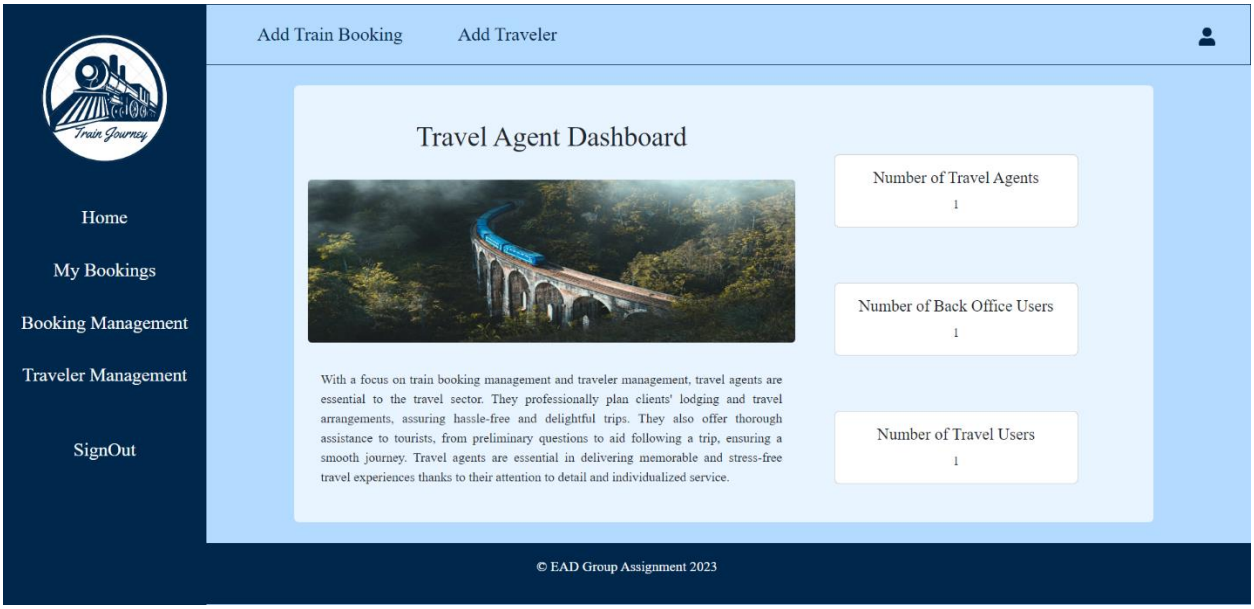

Figure 6.4: Travel Agent User Dashboard UI

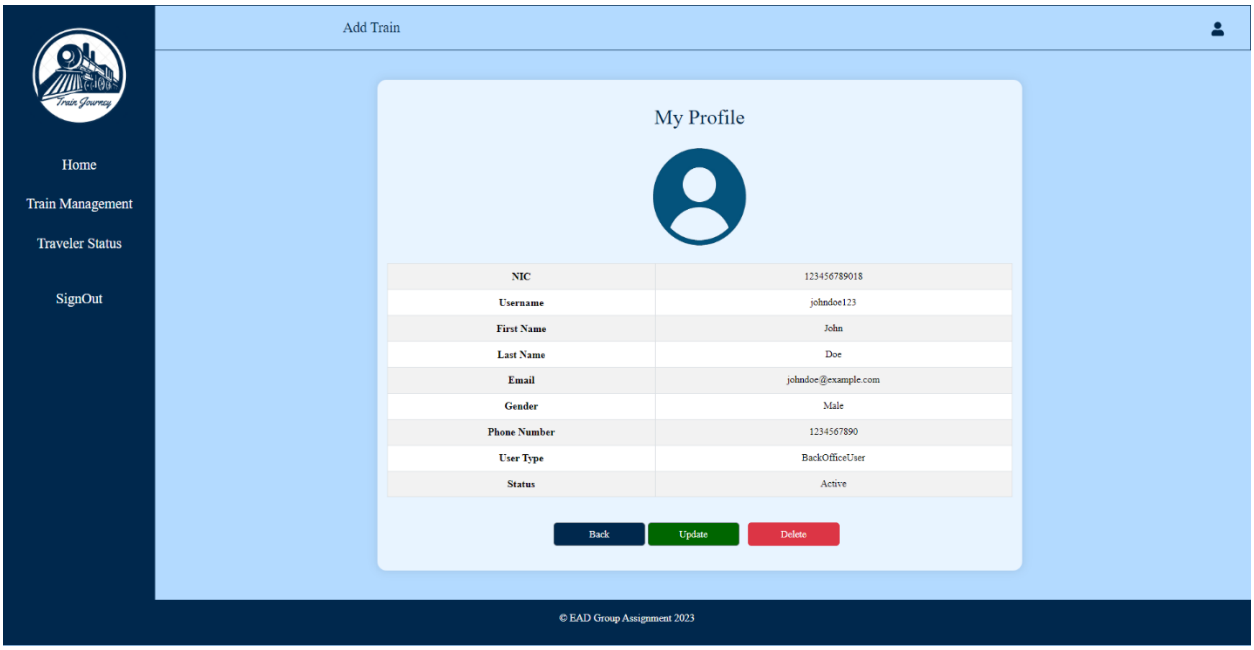

Figure 6.5: User profile UI

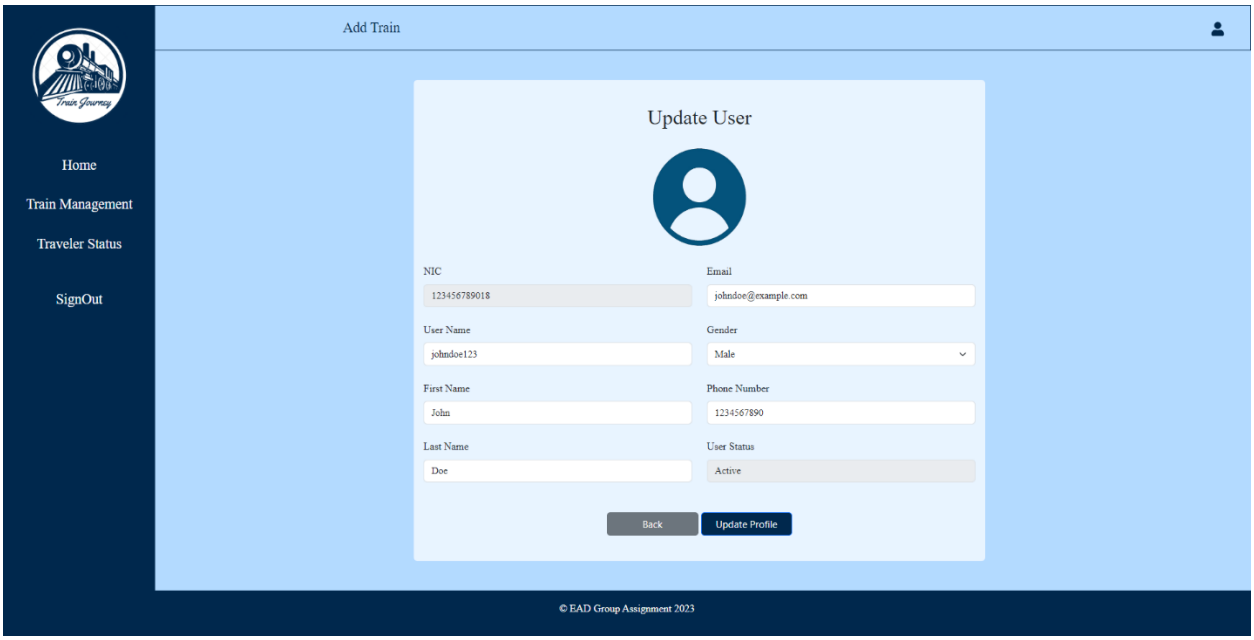

Figure 6.6: Update User profile UI

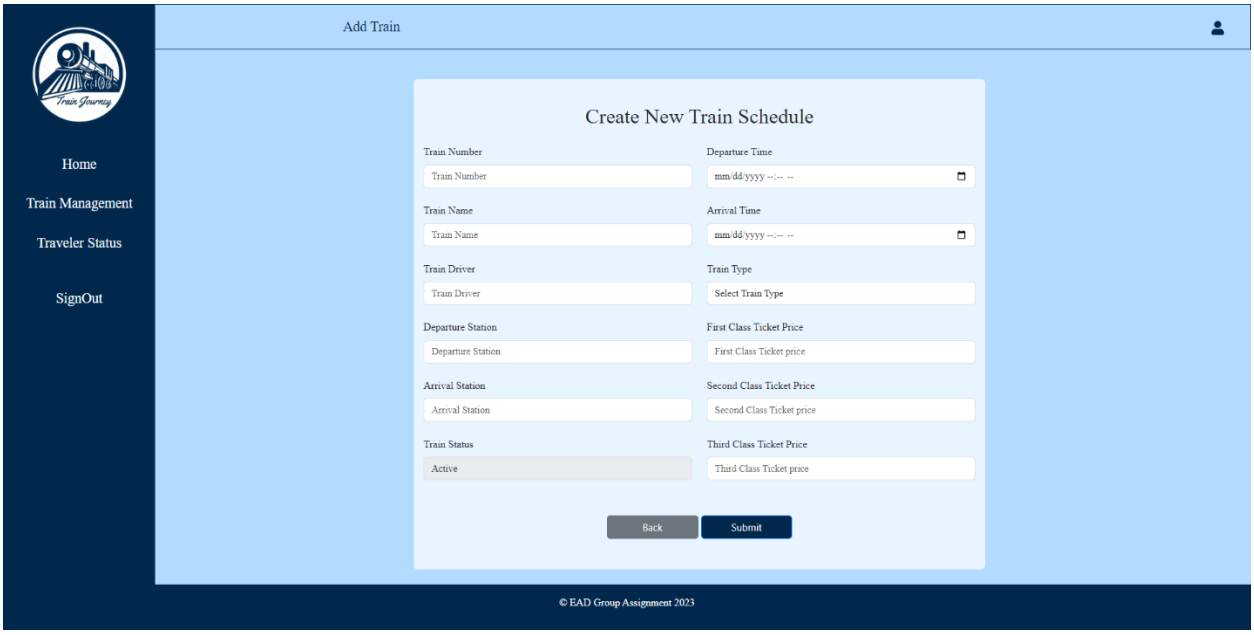

Figure 6.7: Create New Train Schedule UI

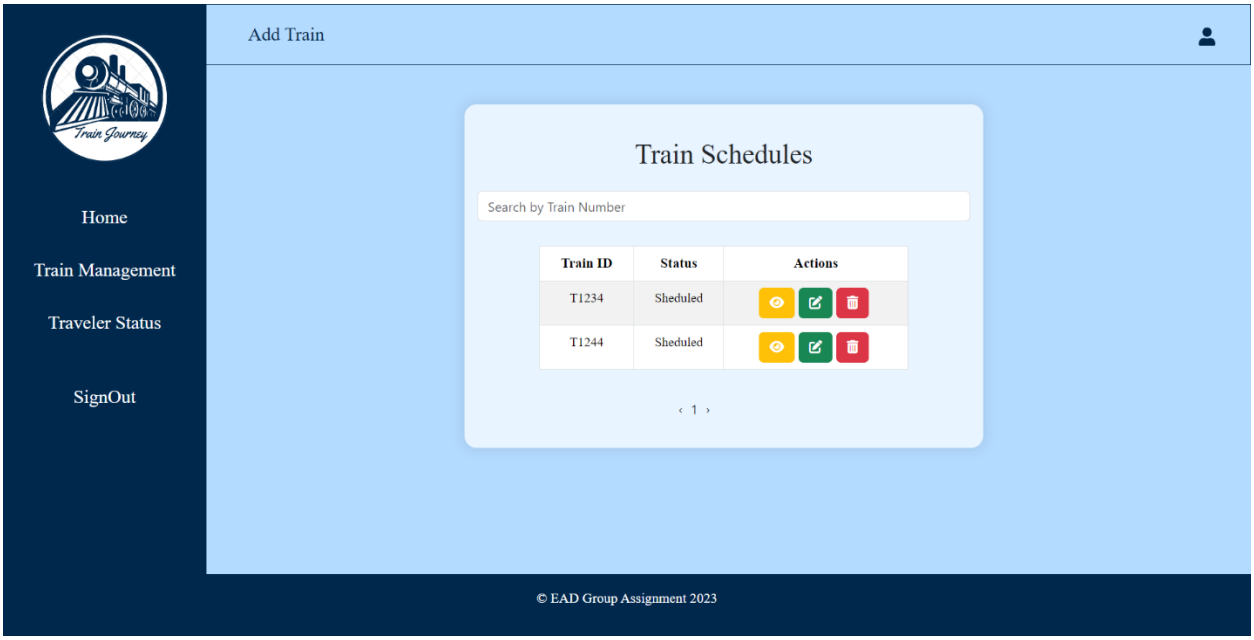

Figure 6.8: Train Schedule UI

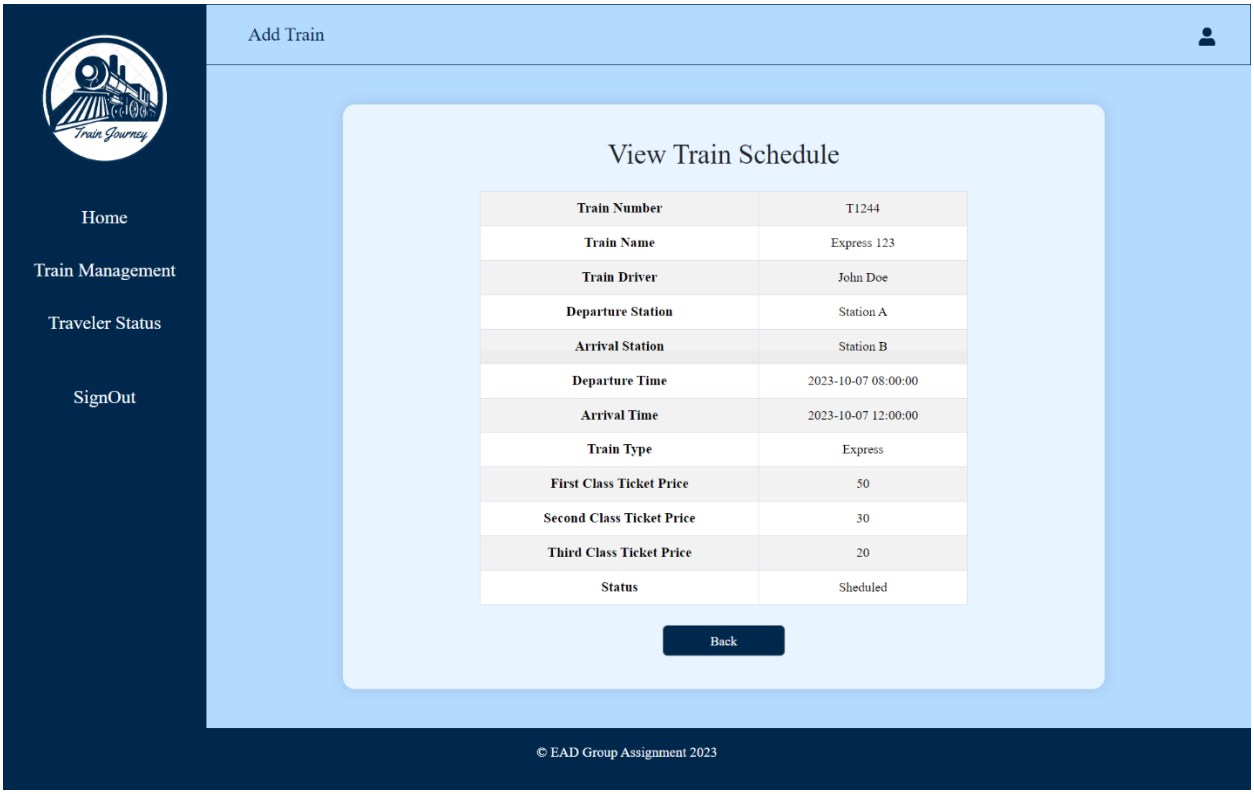

Figure 6.9: View Train Schedule UI

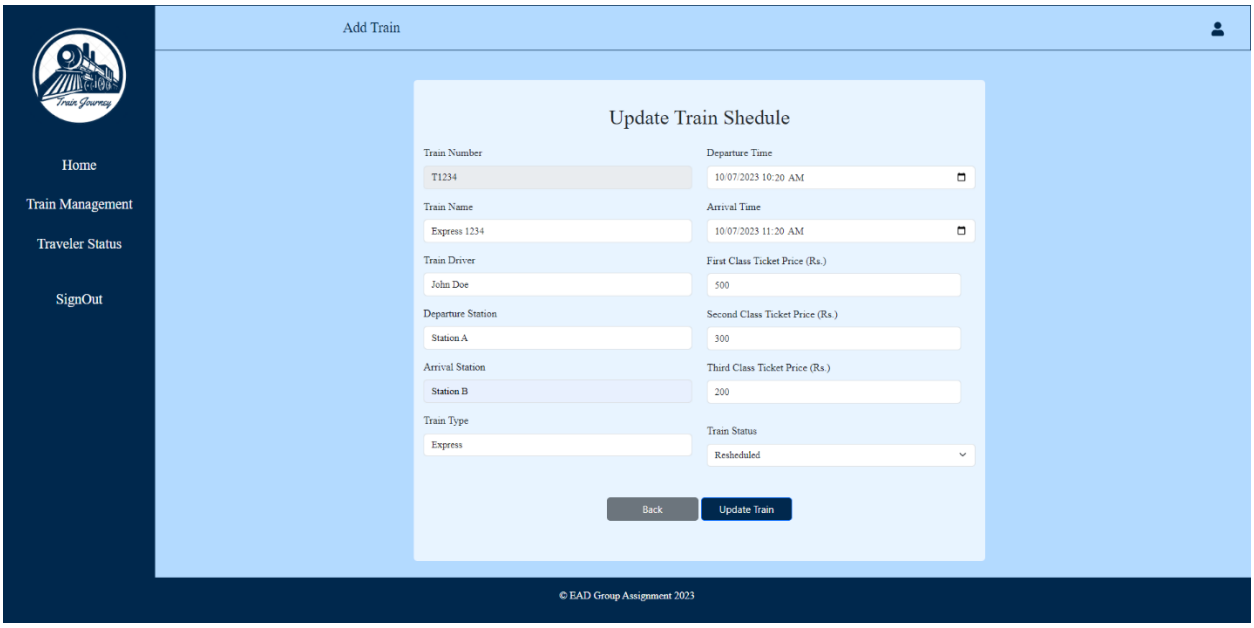

Figure 6.10: Update Train Schedule UI

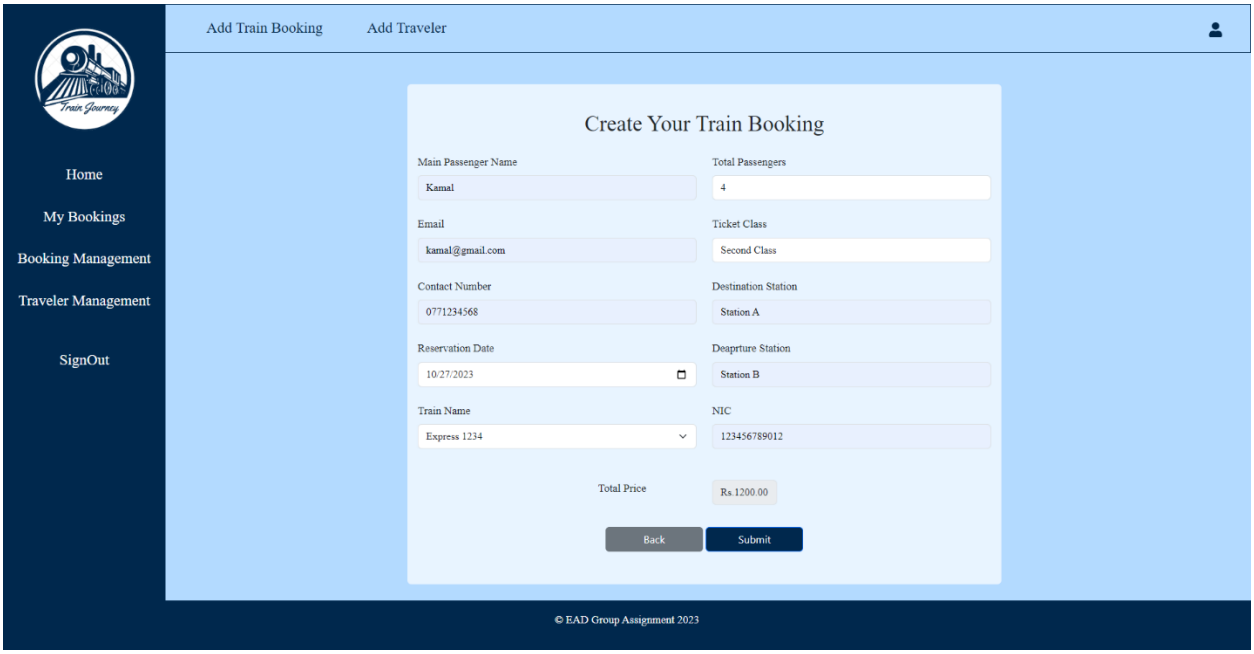

Figure 6.11: Create Your Train Booking UI

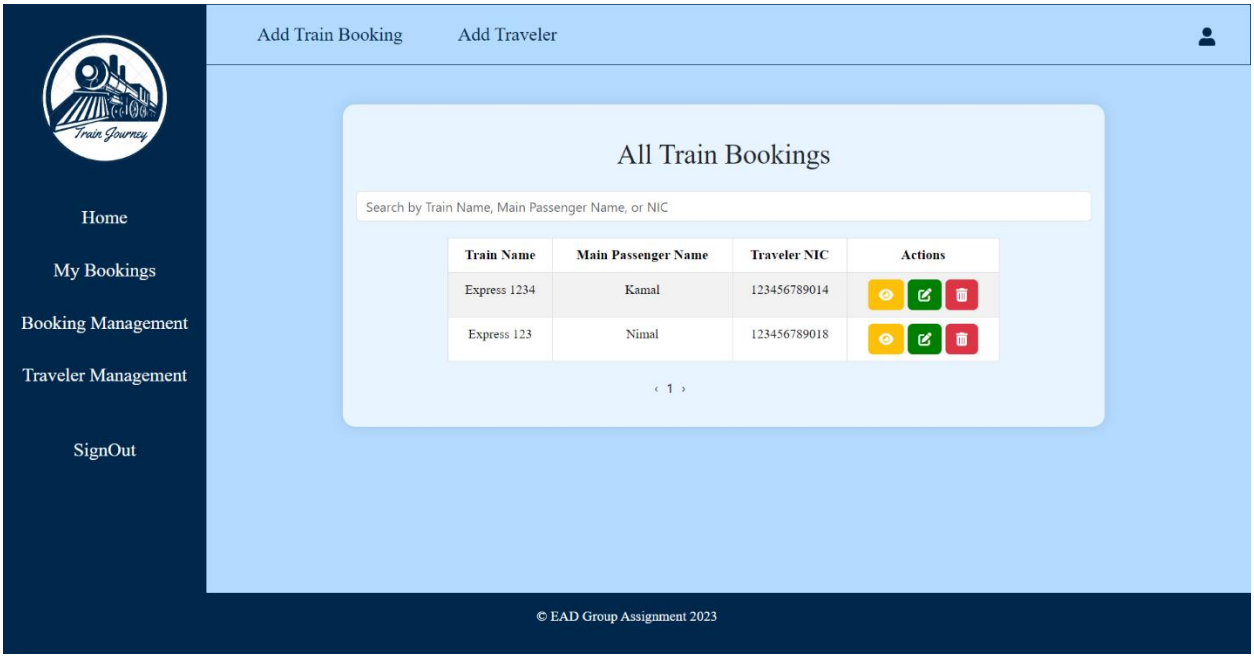

Figure 6.12: All Train Booking UI

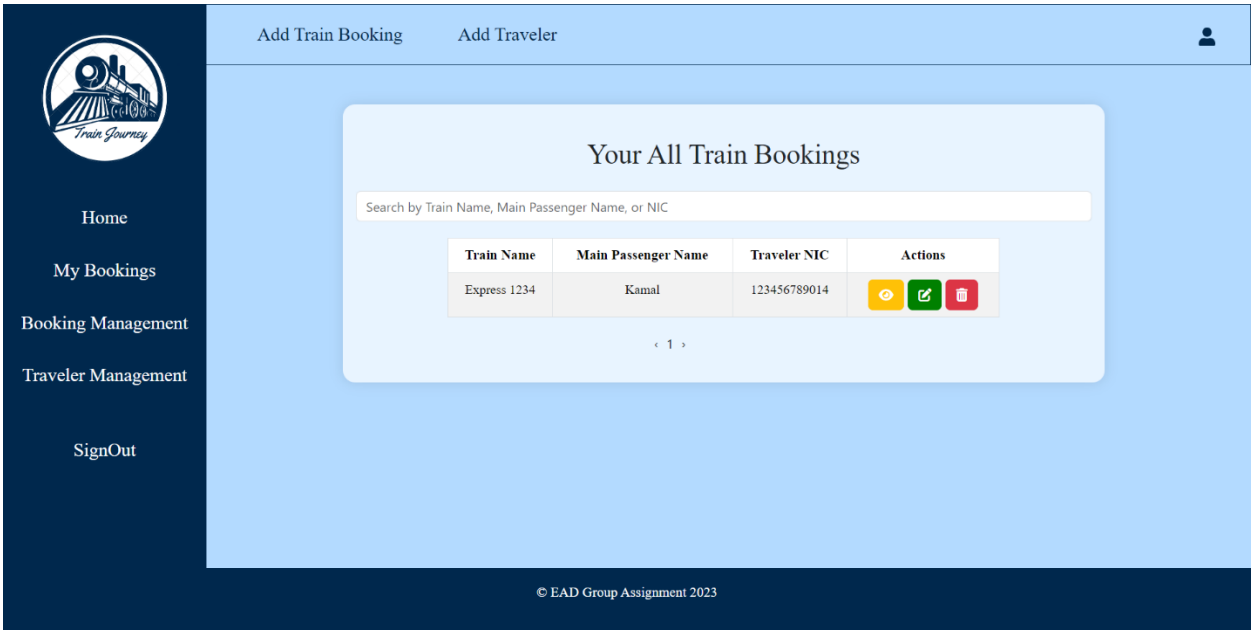

Figure 6.13: Your All Train Bookings UI

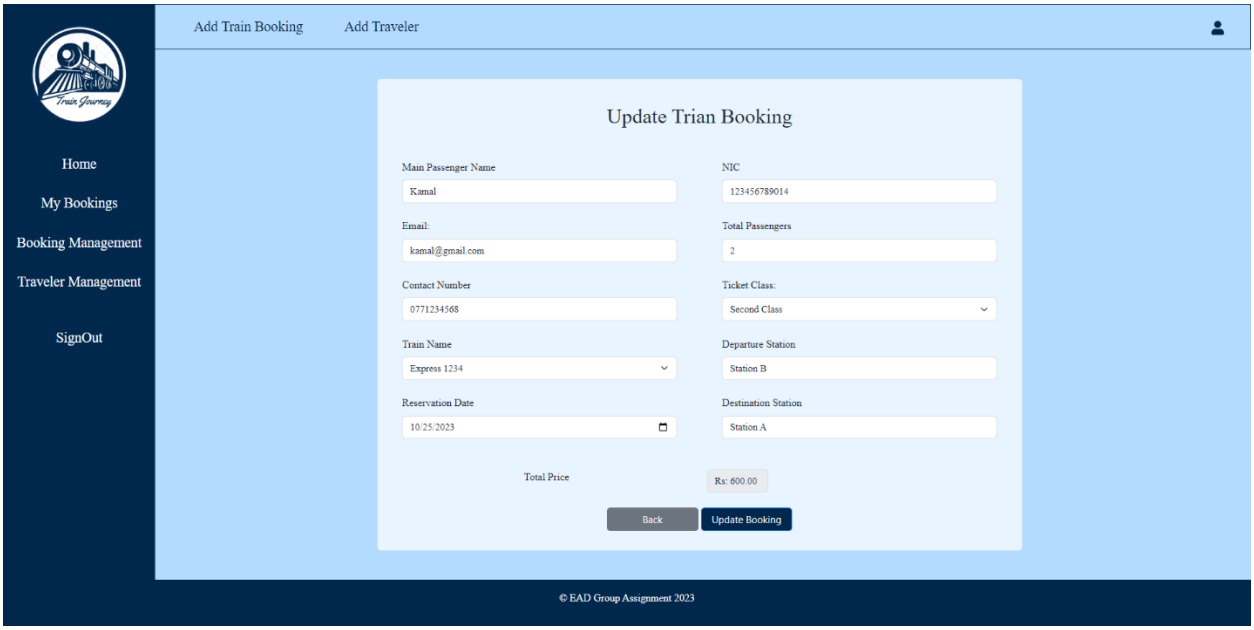

Figure 6.14: Update Train Booking UI

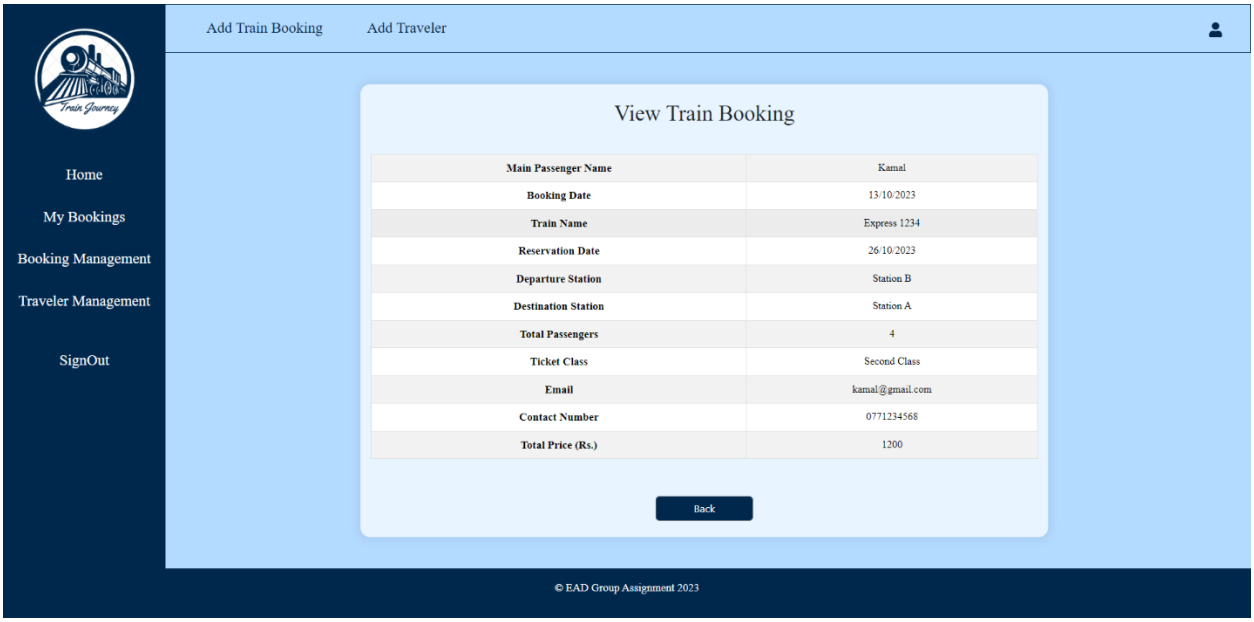

Figure 6.15: View Train Booking UI

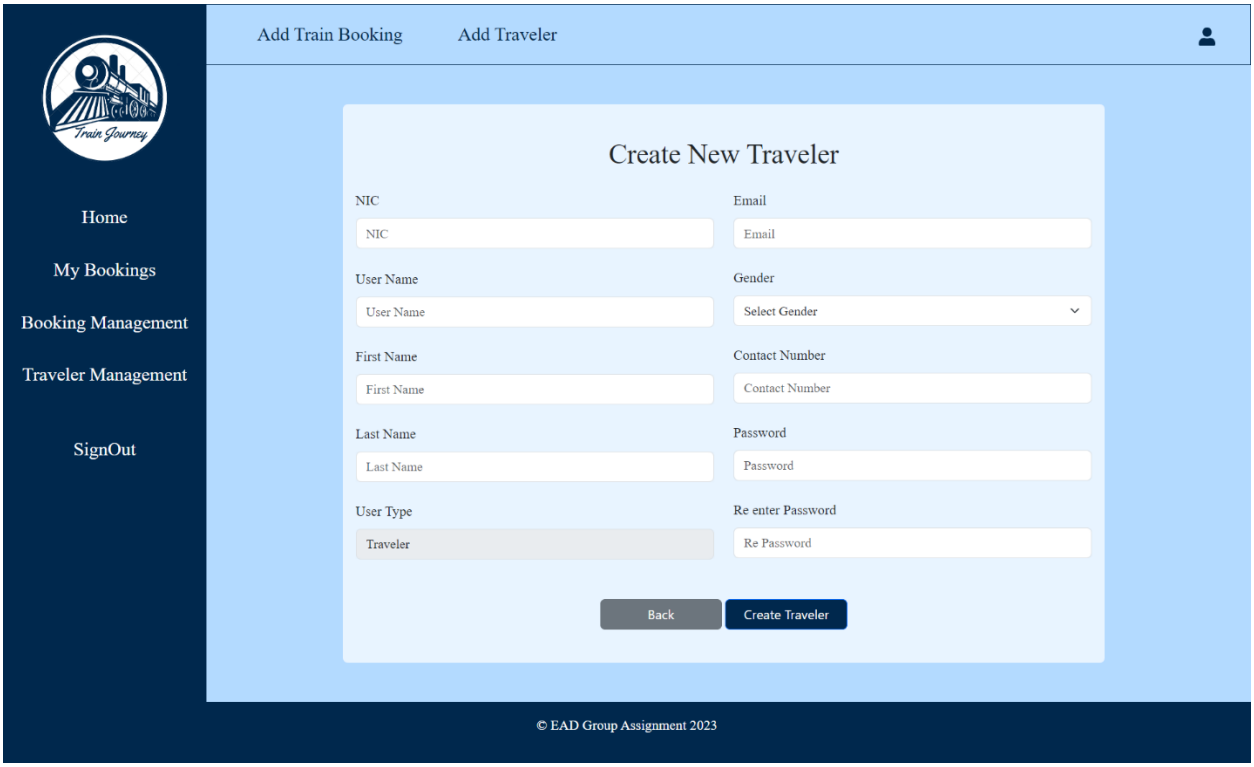

Figure 6.16: Create New Traveler UI

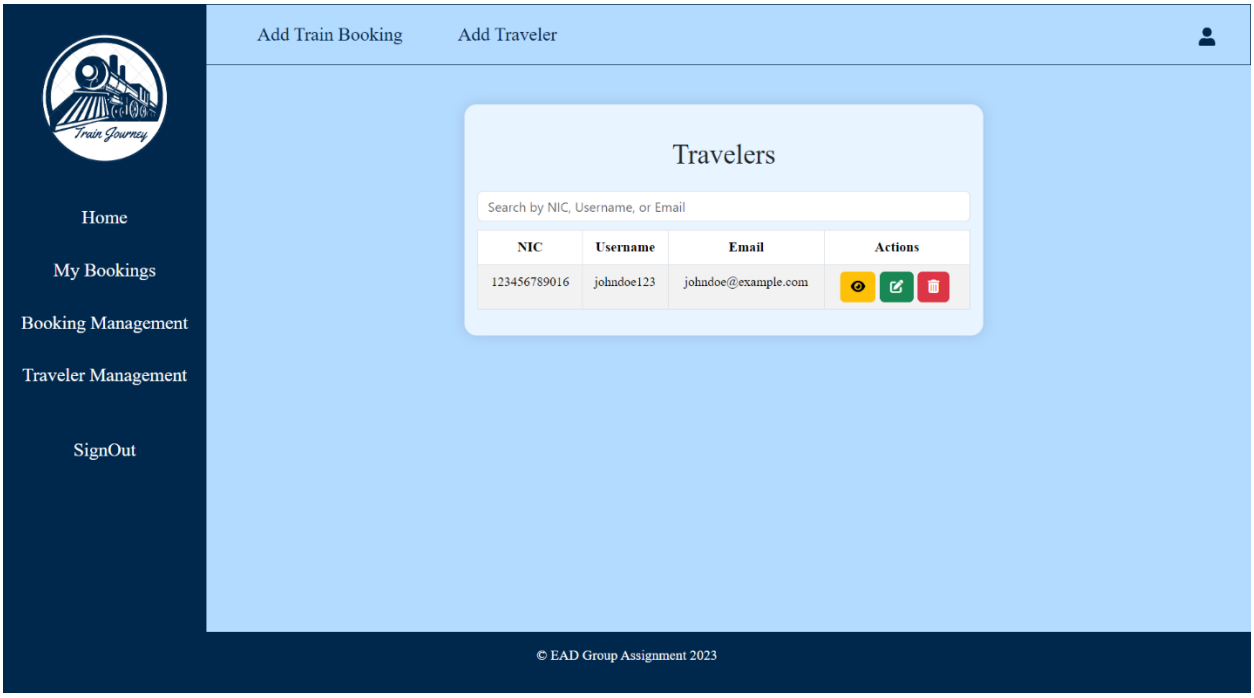

Figure 6.17: Travelers UI

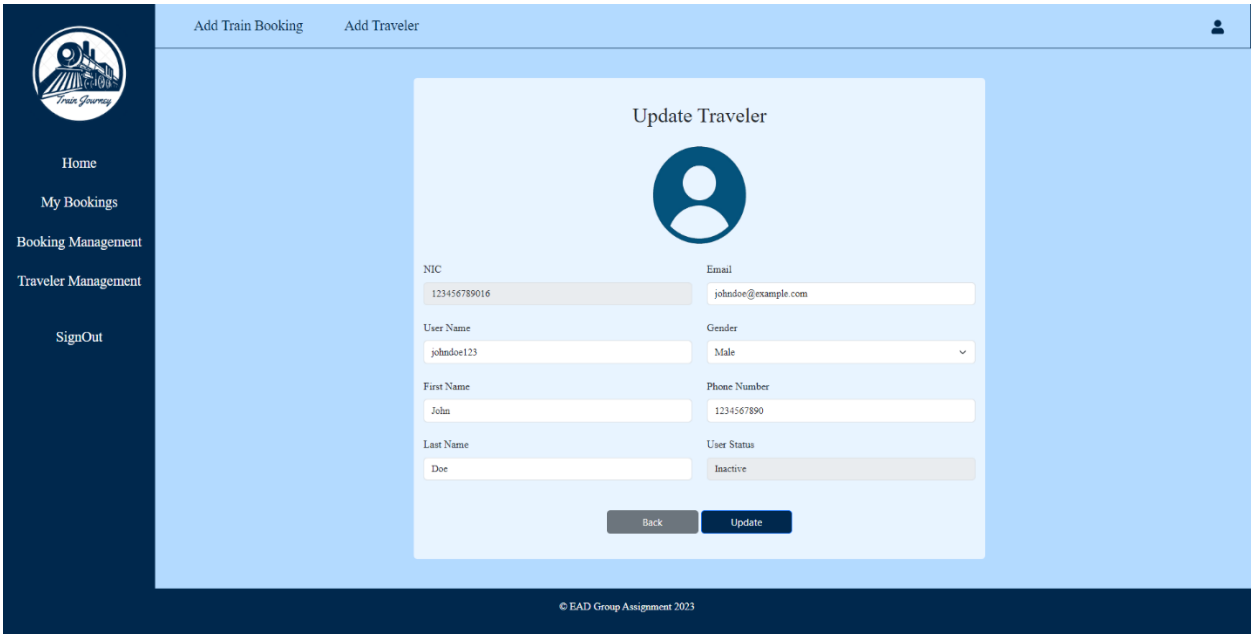

Figure 6.18: Update Traveler UI

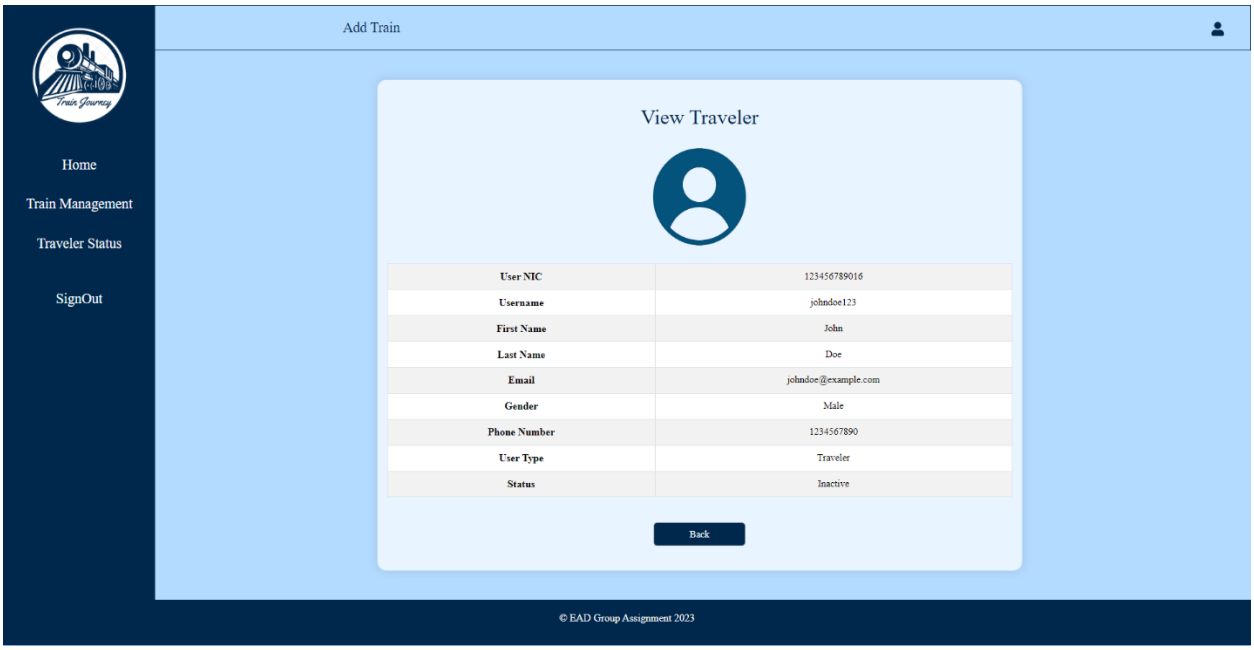

Figure 6.19: View Traveler UI

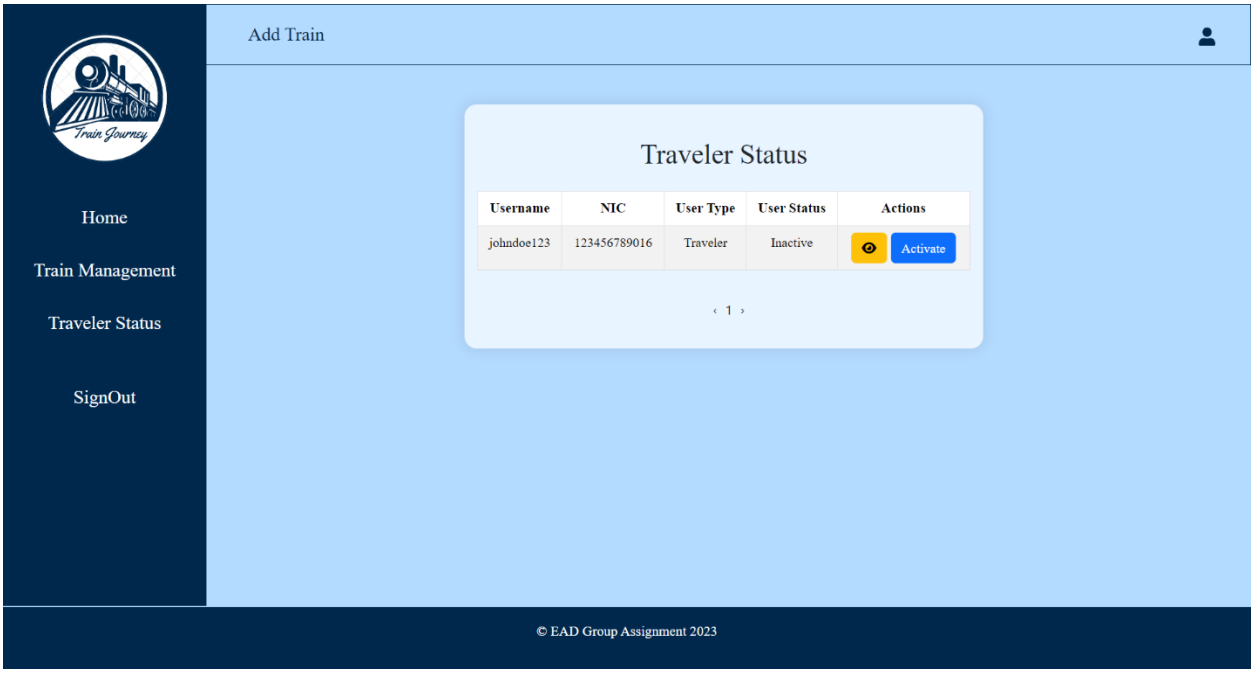

Figure 6.20: Change Traveler Status UI

6.2 Mobile Application

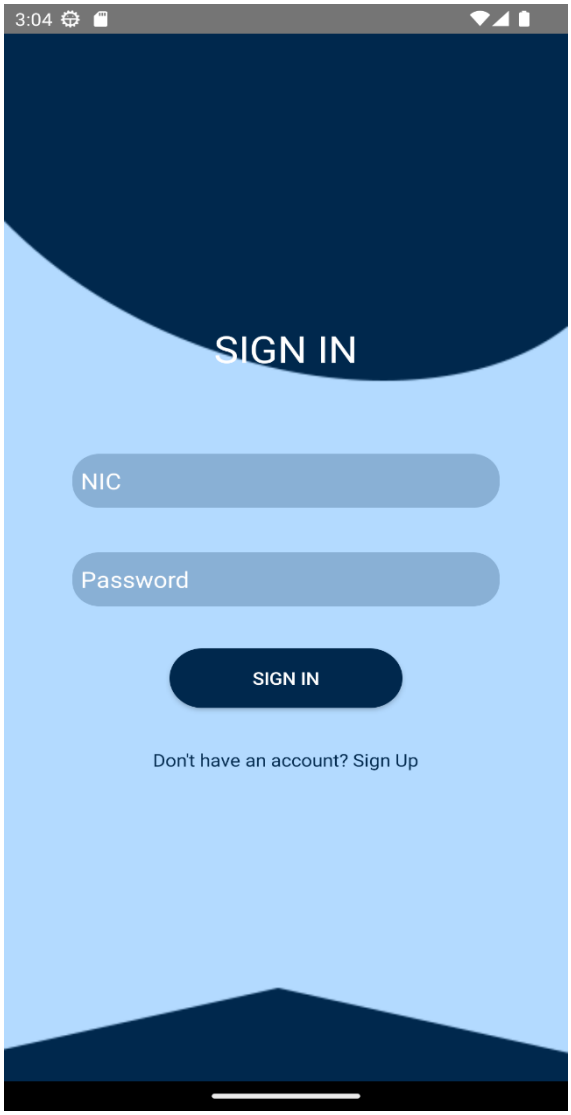

Figure 6.22: Sign In UI

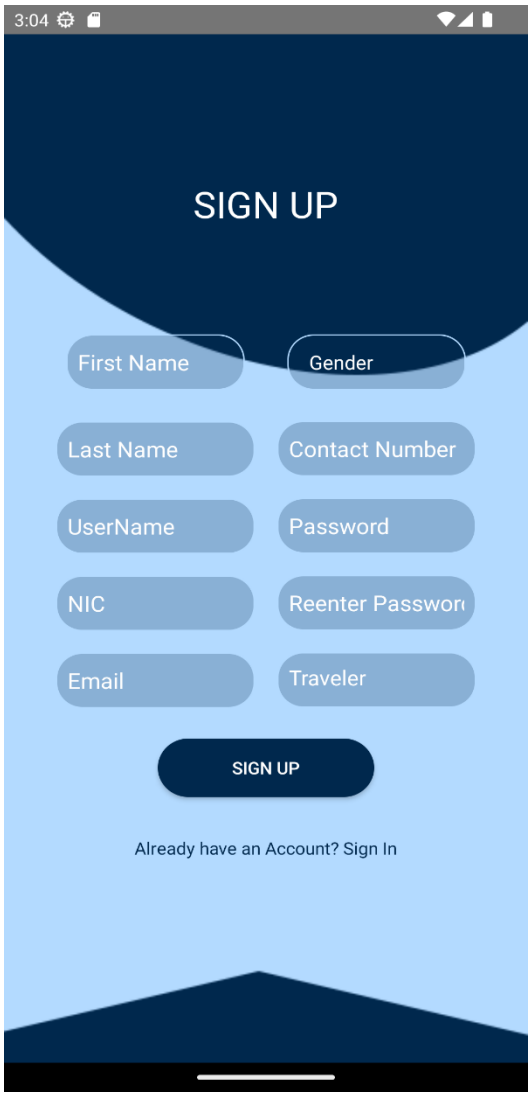

Figure 6.21: Sign Up UI

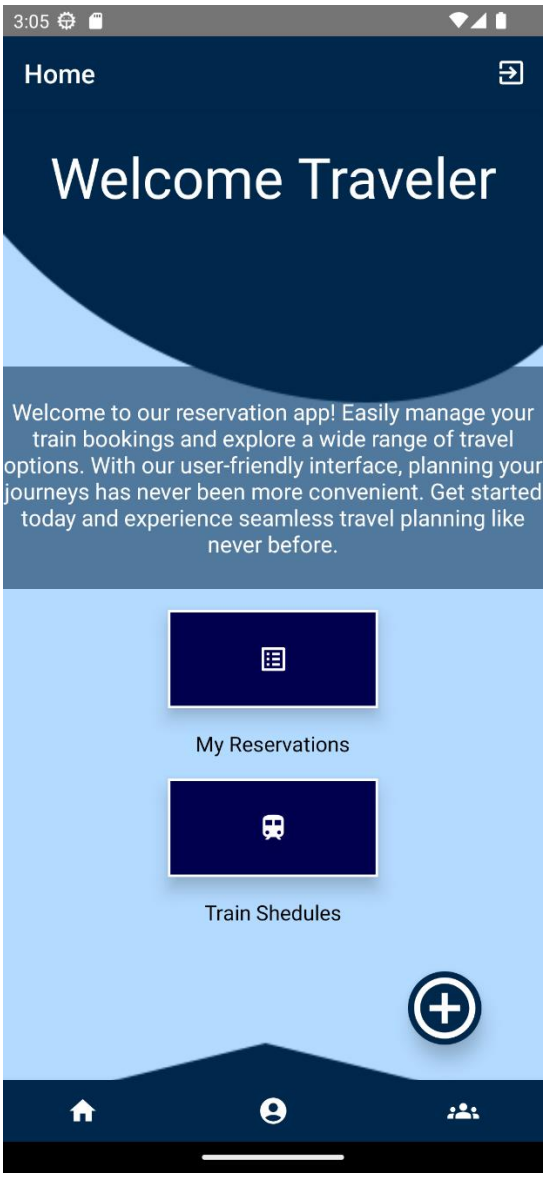

Figure 6.23: Home UI

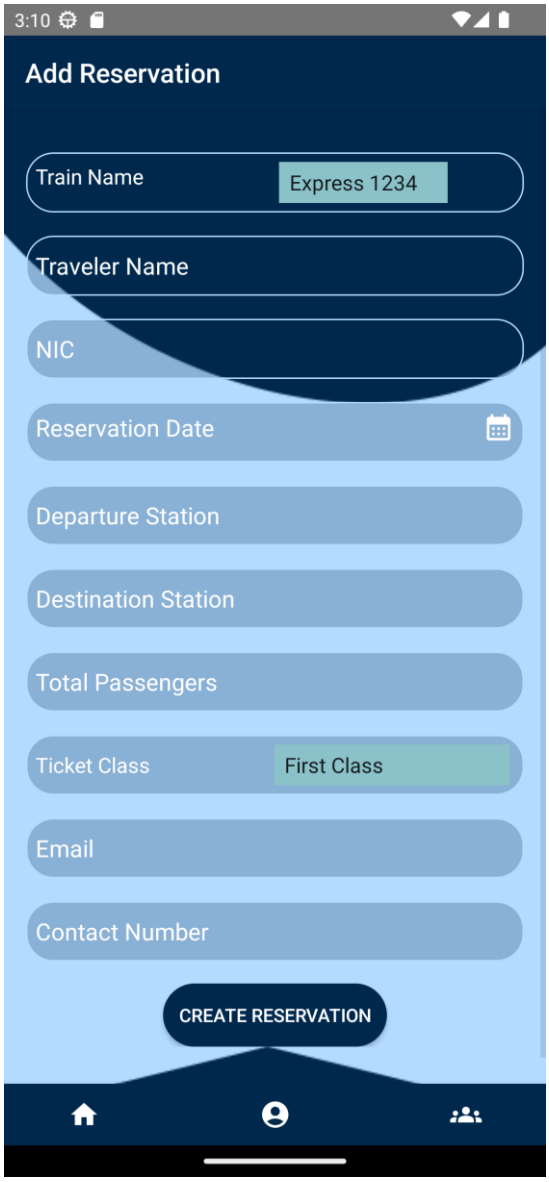

Figure 6.24: Add Reservation UI

Figure 6.25: Update Reservation UI

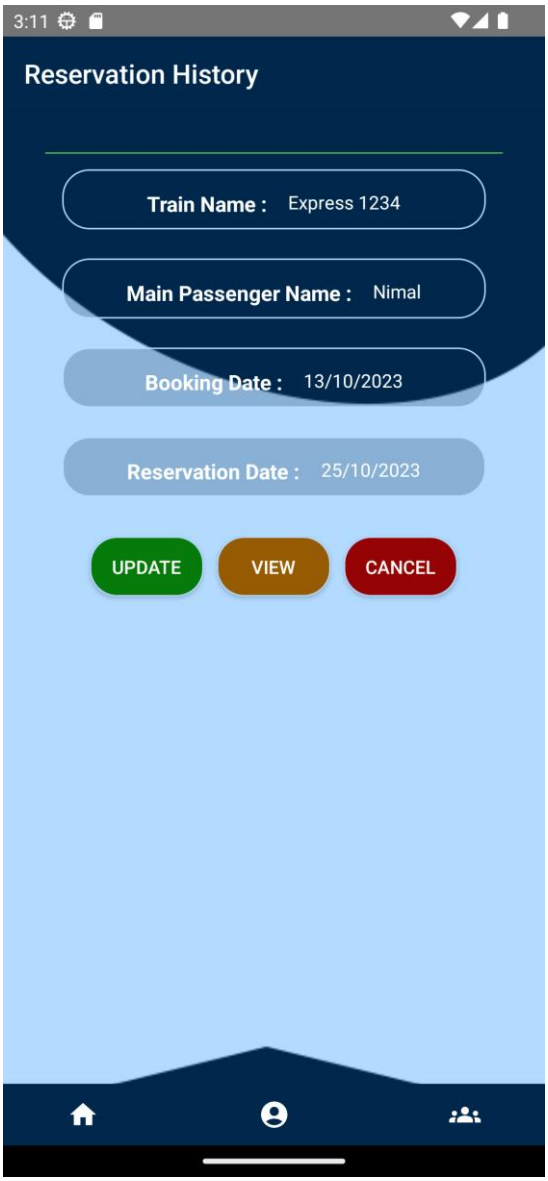

Figure 6.26: Reservation History UI

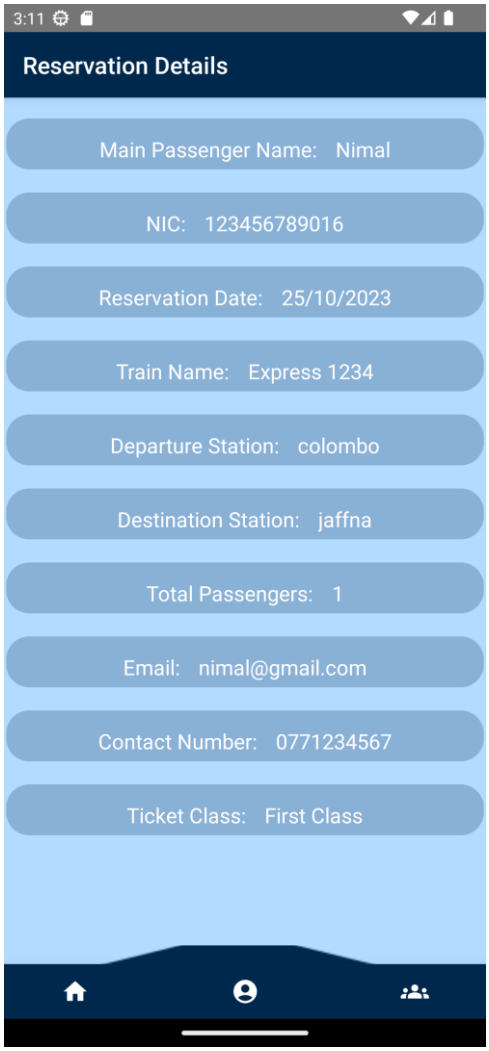

Figure 6.28: Reservation Details UI

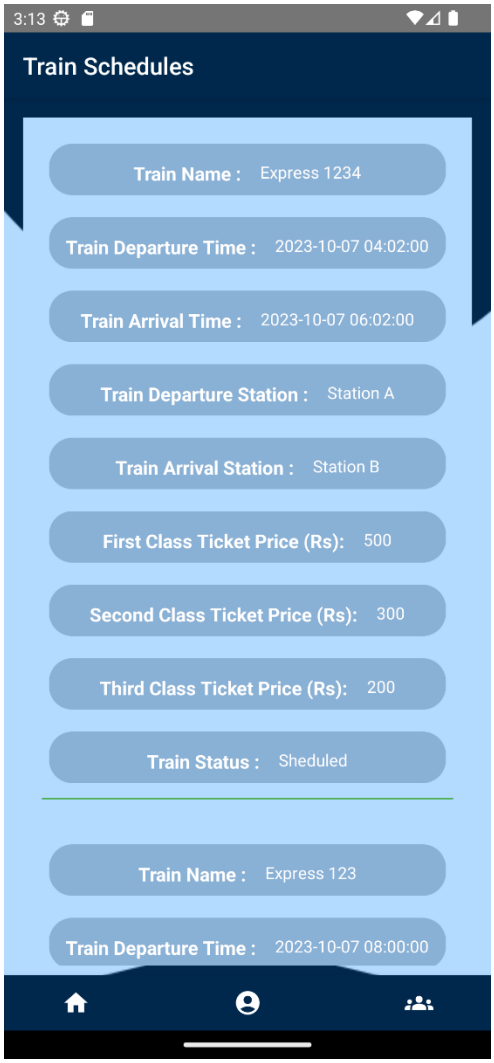

Figure 6.27: Train Schedules UI

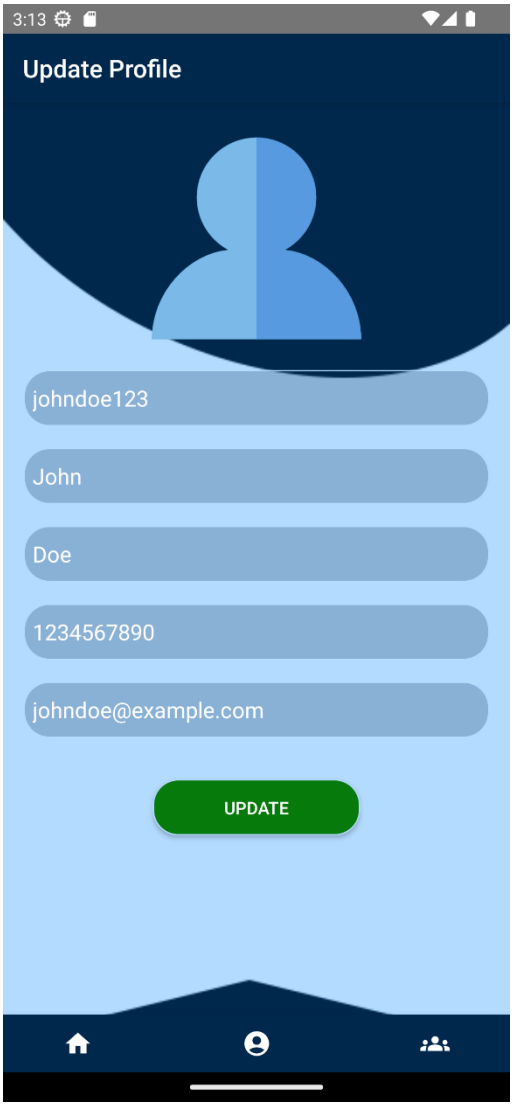

Figure 6.29: Update Profile UI

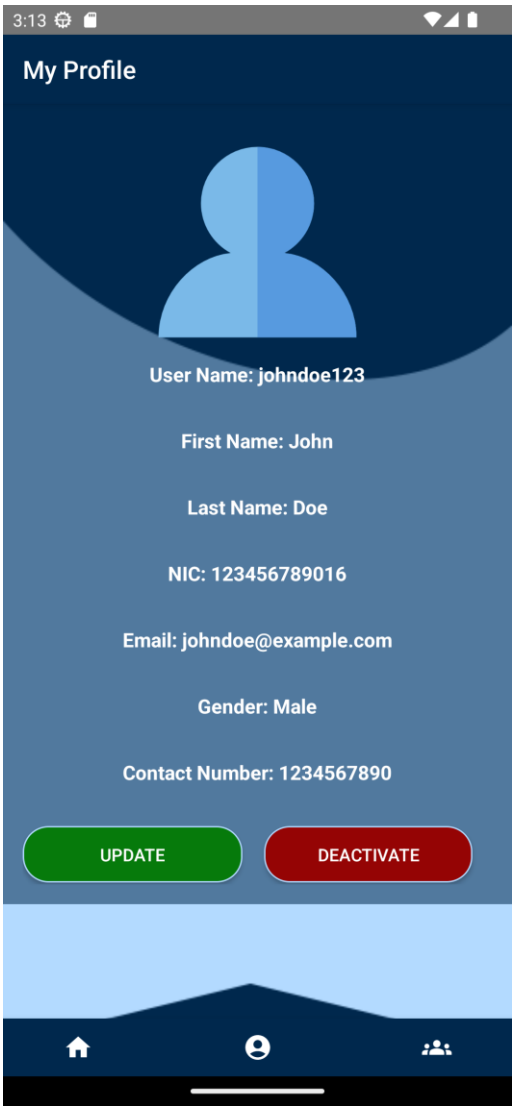

Figure 6.30: My Profile UI

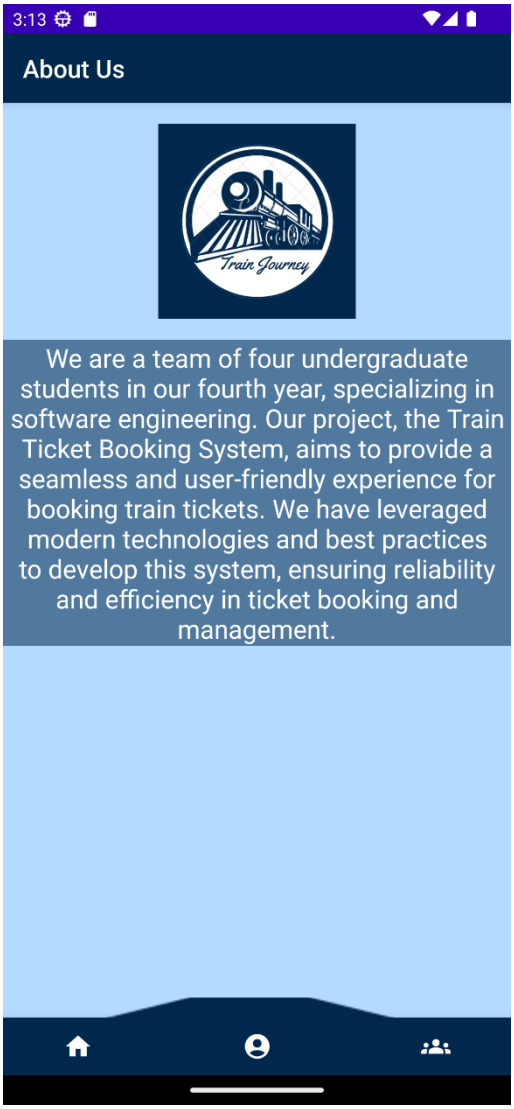

Figure 6.31: About Us UI

7 Source Code

7.1 Web Service

WebApiConfig.cs

```
/*
 * File: WebApiConfig.cs
 * Purpose: Registers Web API configuration and services.
 * Description: This file contains the configuration for the Web API, including
CORS settings and route mapping.
 */

using System.Web.Http;
using System.Web.Http.Cors; // Add this line for CORS

namespace Web_Service
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Enable CORS
            var cors = new EnableCorsAttribute("http://localhost:3000", "*",
"*");
            config.EnableCors(cors);

            // Web API configuration and services
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Models

User.cs

```
/*
    Filename: User.cs
    Description:
    This file contains the definition of the User class, which represents user
data in the application.
*/

using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

namespace Web_Service.Models
{
    public class User
    {
        // Unique identifier for the user
        [BsonId]
        [BsonRepresentation(BsonType.String)]
        public string UserID { get; set; }

        // First name of the user
        public string FirstName { get; set; }

        // Last name of the user
        public string LastName { get; set; }

        // Username chosen by the user
        public string UserName { get; set; }

        // Email address of the user
        public string Email { get; set; }

        // National Identity Card number of the user
        public string NIC { get; set; }

        // Gender of the user
        public string Gender { get; set; }

        // Contact number of the user
        public string ContactNumber { get; set; }

        // Type of user (e.g., admin, regular user, etc.)
        public string UserType { get; set; }
```

```
        // User's chosen password
        public string Password { get; set; }

        // Re-entered password for confirmation
        public string RePassword { get; set; }

        // Status of the user's account (e.g., active, inactive, etc.)
        public string UserStatus { get; set; }
    }
}
```

Train.cs

```
/*
    Filename: Train.cs
    Description:
    This file contains the definition of the Train class, which represents train
information in the application.
*/

using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
using System;

namespace Web_Service.Models.Train_Management
{
    public class Train
    {
        // Unique identifier for the train
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string TrainID { get; set; }

        // ID of the user associated with this train
        public string UserID { get; set; }

        // Number assigned to the train
        public string TrainNumber { get; set; }

        // Name of the train
        public string TrainName { get; set; }

        // Driver of the train
        public string TrainDriver { get; set; }
```

```
// Station where the train departs from
public string DepartureStation { get; set; }

// Station where the train arrives
public string ArrivalStation { get; set; }

// Time when the train departs
public DateTime DepartureTime { get; set; }

// Time when the train arrives
public DateTime ArrivalTime { get; set; }

// Formatted string representation of departure time
public string FormattedDepartureTime
{
    get
    {
        return DepartureTime.ToString("yyyy-MM-dd HH:mm:ss");
    }
}

// Formatted string representation of arrival time
public string FormattedArrivalTime
{
    get
    {
        return ArrivalTime.ToString("yyyy-MM-dd HH:mm:ss");
    }
}

// Type or category of the train
public string TrainType { get; set; }

// Price of a first class ticket
public string FirstClassTicketPrice { get; set; }

// Price of a second class ticket
public string SecondClassTicketPrice { get; set; }

// Price of a third class ticket
public string ThirdClassTicketPrice { get; set; }

// Status of the train (e.g., active, inactive, etc.)
public string TrainStatus { get; set; }
```

```
    }
}
```

TicketBooking.cs

```
/*
    Filename: TicketBooking.cs
    Description:
    This file contains the definition of the TicketBooking class, which
represents ticket booking information in the application.
*/

using System;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

namespace Web_Service.Models.Ticket_Booking_Management
{
    public class TicketBooking
    {
        // Unique identifier for the booking
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string BookingID { get; set; }

        // Name of the train associated with the booking
        public string TrainName { get; set; }

        // ID of the user who made the booking
        public string UserID { get; set; }

        // Date when the booking was made
        public DateTime BookingDate { get; set; }

        // Date when the reservation is for
        public DateTime ReservationDate { get; set; }

        // Formatted string representation of booking date
        public string FormattedBookingDate
        {
            get
            {
                return BookingDate.ToString("dd/MM/yyyy");
```

```
            }
        }

        // Formatted string representation of reservation date
        public string FormattedReservationDate
        {
            get
            {
                return ReservationDate.ToString("dd/MM/yyyy");
            }
        }

        // Total number of passengers in the booking
        public int TotalPassengers { get; set; }

        // Name of the main passenger
        public string MainPassengerName { get; set; }

        // Contact number for the booking
        public string ContactNumber { get; set; }

        // Departure station for the journey
        public string DepartureStation { get; set; }

        // Destination station for the journey
        public string DestinationStation { get; set; }

        // Email address associated with the booking
        public string Email { get; set; }

        // National Identity Card number of the passenger
        public string NIC { get; set; }

        // Class of the ticket (e.g., first class, second class, etc.)
        public string TicketClass { get; set; }

        // Total price of the booking
        public String TotalPrice { get; set; }
    }
}
```

Controllers

UserController.cs

```
/*
    Filename: UsersController.cs
    Description:
    This file contains the definition of the UsersController class, which handles
user-related API endpoints.
*/

using System.Linq;
using System.Web.Http;
using Web_Service.Models;
using MongoDB.Bson;
using MongoDB.Driver;
using System.Security.Cryptography;
using System;
using System.Text;
using System.Text.RegularExpressions;

namespace WebSevice.Controllers
{
    [RoutePrefix("api/users")]
    public class UsersController : ApiController
    {
        private readonly IMongoCollection<User> _usersCollection;

        public UsersController()
        {
            var connectionString =
"mongodb+srv://Pasindu:Pasindu@cluster0.4fhs7.mongodb.net";
            var collectionName = "Users";
            var client = new MongoClient(connectionString);
            var database = client.GetDatabase("EAD_Group_Assignment");
            _usersCollection = database.GetCollection<User>(collectionName);
        }

        // Sign Up endpoint
        [HttpPost]
        [Route("signup")]
        public IHttpActionResult signup(User user)
        {
            var existingUser = _usersCollection.Find(u => u.NIC ==
user.NIC).FirstOrDefault();
```

```
            if (existingUser != null)
            {
                return BadRequest("User with this NIC already exists.");
            }

            if (user.Password != user.RePassword)
            {
                return BadRequest("Passwords do not match.");
            }

            // Validate email format
            if (!IsValidEmail(user.Email))
            {
                return BadRequest("Invalid email format.");
            }

            // Validate NIC format (Assuming NIC is a 9-digit number)
            if (!IsValidNIC(user.NIC))
            {
                return BadRequest("Invalid NIC format.");
            }

            // Validate contact number format (Assuming 10-digit number)
            if (!IsValidContactNumber(user.ContactNumber))
            {
                return BadRequest("Invalid contact number format.");
            }

            // Validate password format (Assuming at least 8 characters, one
uppercase, one lowercase, one digit, and one special character)
            string passwordPattern = @"^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$";
            if (!Regex.IsMatch(user.Password, passwordPattern))
            {
                return BadRequest("Invalid password format.");
            }

            if (!Regex.IsMatch(user.RePassword, passwordPattern))
            {
                return BadRequest("Invalid password format.");
            }

            // Hash the password
            string hashedPassword = HashPassword(user.Password);
```

```
            var newUser = new User
            {
                UserID = ObjectId.GenerateNewId().ToString(),
                FirstName = user.FirstName,
                LastName = user.LastName,
                UserName = user.UserName,
                Email = user.Email,
                ContactNumber = user.ContactNumber,
                NIC = user.NIC,
                Gender = user.Gender,
                UserType = user.UserType,
                Password = hashedPassword,
                RePassword = hashedPassword,
                UserStatus = "Active"
            };
            _usersCollection.InsertOne(newUser);
            return Ok(newUser);
        }



        // Hashing function
        private string HashPassword(string password)
        {
            using (SHA256 sha256 = SHA256.Create())
            {
                byte[] hashedBytes =
sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
                return Convert.ToBase64String(hashedBytes);
            }
        }

        // Validate email format
        private bool IsValidEmail(string email)
        {
            // Use a regular expression to validate email format
            string emailPattern = @"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-
.]+$";
            return Regex.IsMatch(email, emailPattern);
        }

        // Validate NIC format (Assuming NIC is a 9-digit number)
        private bool IsValidNIC(string nic)
        {
            // Use a regular expression to validate NIC format
```

```
            string nicPattern = @"^\d{12}$";
            return Regex.IsMatch(nic, nicPattern);
        }

        // Validate contact number format (Assuming 10-digit number)
        private bool IsValidContactNumber(string contactNumber)
        {
            // Use a regular expression to validate contact number format
            string contactNumberPattern = @"^\d{10}$";
            return Regex.IsMatch(contactNumber, contactNumberPattern);
        }

        // Sign In endpoint
        [HttpPost]
        [Route("signin")]
        public IHttpActionResult signin(User user)
        {
            var existingUser = _usersCollection.Find(u => u.NIC ==
user.NIC).FirstOrDefault();

            if (existingUser == null)
            {
                return NotFound();
            }

            if (existingUser.UserStatus == "Deactive")
            {
                return BadRequest("User is deactive and cannot sign in.");
            }

            // Check if the provided password matches the stored hashed password
            if (!VerifyPassword(user.Password, existingUser.Password))
            {
                return BadRequest("Incorrect password.");
            }

            return Ok(existingUser);
        }

        private bool VerifyPassword(string enteredPassword, string
storedHashedPassword)
        {
            using (SHA256 sha256 = SHA256.Create())
            {
```

```
                byte[] enteredBytes =
sha256.ComputeHash(Encoding.UTF8.GetBytes(enteredPassword));
                string enteredHashedPassword =
Convert.ToBase64String(enteredBytes);
                return enteredHashedPassword == storedHashedPassword;
            }
        }

        // Get User by ID endpoint
        [HttpGet]
        [Route("getuser/{id}")]
        public IHttpActionResult GetUser(string id)
        {
            var user = _usersCollection.Find(u => u.UserID ==
id).FirstOrDefault();

            if (user == null)
            {
                return NotFound();
            }
            return Ok(user);
        }

        // Update User endpoint
        [HttpPut]
        [Route("updateuser/{id}")]
        public IHttpActionResult updateuser(string id, User updatedUserData)
        {
            // Validate email format
            if (!IsValidEmail(updatedUserData.Email))
            {
                return BadRequest("Invalid email format.");
            }

            // Validate contact number format (Assuming 10-digit number)
            if (!IsValidContactNumber(updatedUserData.ContactNumber))
            {
                return BadRequest("Invalid contact number format.");
            }

            var filter = Builders<User>.Filter.Eq(u => u.UserID, id);
            var update = Builders<User>.Update
                .Set(u => u.UserName, updatedUserData.UserName)
                .Set(u => u.FirstName, updatedUserData.FirstName)
                .Set(u => u.LastName, updatedUserData.LastName)
```

```
                .Set(u => u.Email, updatedUserData.Email)
                .Set(u => u.Gender, updatedUserData.Gender)
                .Set(u => u.ContactNumber, updatedUserData.ContactNumber);

            var result = _usersCollection.UpdateOne(filter, update);

            if (result.ModifiedCount == 0)
            {
                return NotFound();
            }

            // Retrieve the updated user
            var updatedUser = _usersCollection.Find(u => u.UserID ==
id).FirstOrDefault();

            if (updatedUser == null)
            {
                return NotFound();
            }

            return Ok(updatedUser);
        }

        // Get all users with UserType 'Traveler'
        [HttpGet]
        [Route("getallusers")]
        public IHttpActionResult GetAllUsers()
        {
            var filter = Builders<User>.Filter.Eq(u => u.UserType, "Traveler");
            var travelusers = _usersCollection.Find(filter).ToList();
            return Ok(travelusers);
        }

        // Update User Status endpoint
        [HttpPut]
        [Route("updateuserstatus/{id}")]
        public IHttpActionResult updateuser(string id, [FromBody] UserStatusModel
updateModel)
        {
            var filter = Builders<User>.Filter.And(
                Builders<User>.Filter.Eq(u => u.UserID, id)
            );
            var update = Builders<User>.Update.Set(u => u.UserStatus,
updateModel.UserStatus);
```

```
            var result = _usersCollection.UpdateOne(filter, update);

            if (result.ModifiedCount == 0)
            {
                return NotFound();
            }
            return Ok("User Status Updated");
        }

        public class UserStatusModel
        {
            public string UserStatus { get; set; }
        }

        // Delete User endpoint
        [HttpDelete]
        [Route("deleteuser/{id}")]
        public IHttpActionResult DeleteUser(string id)
        {
            var result = _usersCollection.DeleteOne(u => u.UserID == id);

            if (result.DeletedCount == 0)
            {
                return NotFound();
            }
            return Ok("user deleted");
        }

        // Get total number of users
        [HttpGet]
        [Route("getusercount")]
        public IHttpActionResult GetUserCount()
        {
            long userCount = _usersCollection.CountDocuments(new BsonDocument());

            return Ok(userCount);
        }

        // Get total number of 'BackOfficeUser'
        [HttpGet]
        [Route("getbackofficeusercount")]
        public IHttpActionResult GetTravelAgentCount()
        {
            var filter = Builders<User>.Filter.Eq(u => u.UserType,
"BackOfficeUser");
```

```
            long backofficeUserCount = _usersCollection.CountDocuments(filter);

            return Ok(backofficeUserCount);
        }

        // Get total number of 'TravelAgent'
        [HttpGet]
        [Route("gettravelagentcount")]
        public IHttpActionResult GetBackofficeUserCount()
        {
            var filter = Builders<User>.Filter.Eq(u => u.UserType,
"TravelAgent");
            long backofficeUserCount = _usersCollection.CountDocuments(filter);

            return Ok(backofficeUserCount);
        }

        // Get total number of 'Traveler'
        [HttpGet]
        [Route("gettravelusercount")]
        public IHttpActionResult GetTravelUserCount()
        {
            var filter = Builders<User>.Filter.Eq(u => u.UserType, "Traveler");
            long backofficeUserCount = _usersCollection.CountDocuments(filter);

            return Ok(backofficeUserCount);
        }

    }
}
```

TrainController.cs

```
/*
    Filename: TrainsController.cs
    Description:
    This file contains the definition of the TrainsController class, which
handles train-related API endpoints.
*/

using System.Web.Http;
using MongoDB.Bson;
using MongoDB.Driver;
using Web_Service.Models.Train_Management;
```

```
namespace Web_Service.Controllers
{
    [RoutePrefix("api/trains")]
    public class TrainsController : ApiController
    {
        private readonly IMongoCollection<Train> _trainsCollection;

        public TrainsController()
        {
            var connectionString =
"mongodb+srv://Pasindu:Pasindu@cluster0.4fhs7.mongodb.net";
            var collectionName = "Trains";
            var client = new MongoClient(connectionString);
            var database = client.GetDatabase("EAD_Group_Assignment");
            _trainsCollection = database.GetCollection<Train>(collectionName);
        }

        // Create Train endpoint
        [HttpPost]
        [Route("createtrain")]
        public IHttpActionResult CreateTrain(Train train)
        {
            // Validate Train Name
            if (string.IsNullOrWhiteSpace(train.TrainName))
            {
                return BadRequest("Train Name is required.");
            }

            // Validate DepartureStation and ArrivalStation
            if (string.IsNullOrWhiteSpace(train.DepartureStation) ||
string.IsNullOrWhiteSpace(train.ArrivalStation))
            {
                return BadRequest("Both Departure and Arrival Stations are
required.");
            }

            // Validate DepartureTime and ArrivalTime
            if (train.DepartureTime >= train.ArrivalTime)
            {
                return BadRequest("Departure Time must be before Arrival Time.");
            }

            // Validate Train Type
            if (string.IsNullOrWhiteSpace(train.TrainType))
            {
```

```
                return BadRequest("Train Type is required.");
            }

            // Validate Ticket Prices
            if (string.IsNullOrWhiteSpace(train.FirstClassTicketPrice) ||
                string.IsNullOrWhiteSpace(train.SecondClassTicketPrice) ||
                string.IsNullOrWhiteSpace(train.ThirdClassTicketPrice))
            {
                return BadRequest("All Ticket Prices are required.");
            }

            // Set Train ID and Status
            train.TrainID = ObjectId.GenerateNewId().ToString();
            train.TrainStatus = "Sheduled";

            // Insert the train into the database
            _trainsCollection.InsertOne(train);

            return Ok(train);
        }

        // Get all trains
        [HttpGet]
        [Route("getalltrains")]
        public IHttpActionResult GetAllTrains()
        {
            var trains = _trainsCollection.Find(new BsonDocument()).ToList();
            return Ok(trains);
        }

        // Get all scheduled trains
        [HttpGet]
        [Route("getallSheduledtrains")]
        public IHttpActionResult GetActiveTrains()
        {
            var activeTrains = _trainsCollection.Find(t => t.TrainStatus ==
"Sheduled").ToList();
            return Ok(activeTrains);
        }

        // Get train by Train Name
        [HttpGet]
        [Route("gettrain/{id}")]
        public IHttpActionResult GetTrain(string id)
        {
```

```
            var train = _trainsCollection.Find(t => t.TrainName ==
id).FirstOrDefault();
            if (train == null)
            {
                return NotFound();
            }
            return Ok(train);
        }

        // Get train by Train ID
        [HttpGet]
        [Route("gettrainbyId/{id}")]
        public IHttpActionResult GetTrainById(string id)
        {
            var train = _trainsCollection.Find(t => t.TrainID ==
id).FirstOrDefault();
            if (train == null)
            {
                return NotFound();
            }
            return Ok(train);
        }

        // Update Train endpoint
        [HttpPut]
        [Route("updatetrain/{id}")]
        public IHttpActionResult updatetrain(string id, Train updatedTrain)
        {
            if (string.IsNullOrWhiteSpace(updatedTrain.TrainName))
            {
                return BadRequest("Train Name is required.");
            }

            if (string.IsNullOrWhiteSpace(updatedTrain.DepartureStation) ||
string.IsNullOrWhiteSpace(updatedTrain.ArrivalStation))
            {
                return BadRequest("Both Departure and Arrival Stations are
required.");
            }

            if (updatedTrain.DepartureTime >= updatedTrain.ArrivalTime)
            {
                return BadRequest("Departure Time must be before Arrival Time.");
            }
```

```
            if (string.IsNullOrWhiteSpace(updatedTrain.TrainType))
            {
                return BadRequest("Train Type is required.");
            }

            if (string.IsNullOrWhiteSpace(updatedTrain.FirstClassTicketPrice) ||
                string.IsNullOrWhiteSpace(updatedTrain.SecondClassTicketPrice) ||
                string.IsNullOrWhiteSpace(updatedTrain.ThirdClassTicketPrice))
            {
                return BadRequest("All Ticket Prices are required.");
            }

            var filter = Builders<Train>.Filter.Eq(t => t.TrainID, id);
            var update = Builders<Train>.Update
                .Set(t => t.TrainNumber, updatedTrain.TrainNumber)
                .Set(t => t.TrainName, updatedTrain.TrainName)
                .Set(t => t.TrainDriver, updatedTrain.TrainDriver)
                .Set(t => t.DepartureStation, updatedTrain.DepartureStation)
                .Set(t => t.ArrivalStation, updatedTrain.ArrivalStation)
                .Set(t => t.DepartureTime, updatedTrain.DepartureTime)
                .Set(t => t.ArrivalTime, updatedTrain.ArrivalTime)
                .Set(t => t.TrainType, updatedTrain.TrainType)
                .Set(t => t.FirstClassTicketPrice,
updatedTrain.FirstClassTicketPrice)
                .Set(t => t.SecondClassTicketPrice,
updatedTrain.SecondClassTicketPrice)
                .Set(t => t.ThirdClassTicketPrice,
updatedTrain.ThirdClassTicketPrice)
                .Set(t => t.TrainStatus, updatedTrain.TrainStatus);

            var result = _trainsCollection.UpdateOne(filter, update);

            if (result.ModifiedCount == 0)
            {
                return NotFound();
            }

            return Ok("Train schedule updated");
        }

        // Delete Train endpoint
        [HttpDelete]
        [Route("deletetrain/{id}")]
        public IHttpActionResult DeleteTrain(string id)
        {
```

```
            var result = _trainsCollection.DeleteOne(t => t.TrainID == id);

            if (result.DeletedCount == 0)
            {
                return NotFound();
            }

            return Ok("Train shedule deleted");
        }

        // Get total number of trains
        [HttpGet]
        [Route("gettraincount")]
        public IHttpActionResult GetTrainCount()
        {
            long trainCount = _trainsCollection.CountDocuments(new
BsonDocument());
            return Ok(trainCount);
        }
    }
}
```

TicketBookingController.cs

```
/*
    Filename: TrainTicketBookingController.cs
    Description:
    This file contains the definition of the TrainTicketBookingController class,
which handles train ticket booking related API endpoints.
*/

using MongoDB.Bson;
using System;
using System.Linq;
using System.Web.Http;
using MongoDB.Driver;
using Web_Service.Models.Ticket_Booking_Management;
using System.Text.RegularExpressions;
using System.Collections.Generic;

namespace WebSevice.Controllers
{
    [RoutePrefix("api/trainbooking")]
    public class TrainTicketBookingController : ApiController
    {
```

```
        private readonly IMongoCollection<TicketBooking> _bookingsCollection;

        public TrainTicketBookingController()
        {
            var connectionString =
"mongodb+srv://Pasindu:Pasindu@cluster0.4fhs7.mongodb.net";
            var collectionName = "TrainBookings";
            var client = new MongoClient(connectionString);
            var database = client.GetDatabase("EAD_Group_Assignment");
            _bookingsCollection =
database.GetCollection<TicketBooking>(collectionName);
        }

        // Create Train Ticket Booking endpoint
        [HttpPost]
        [Route("createticketbooking")]
        public IHttpActionResult CreateBooking(TicketBooking booking)
        {
            booking.BookingID = ObjectId.GenerateNewId().ToString();

            // Calculate the difference in days between ReservationDate and
BookingDate
            int daysDifference = (booking.ReservationDate - DateTime.Now).Days;

            if (daysDifference < 30)
            {

                if (!IsValidEmail(booking.Email))
                {
                    return BadRequest("Invalid email address.");
                }

                if (!IsValidContactNumber(booking.ContactNumber))
                {
                    return BadRequest("Invalid contact number.");
                }

                if (!IsValidTicketClass(booking.TicketClass))
                {
                    return BadRequest("Invalid ticket class.");
                }

                if (!IsValidNIC(booking.NIC))
                {
                    return BadRequest("Invalid NIC.");
```

```
                }

                var NIC = booking.NIC;
                var maxReservationsCount = _bookingsCollection.CountDocuments(x
=> x.NIC == NIC);

                if (maxReservationsCount >= 4)
                {
                    return BadRequest("Maximum 4 reservations allowed per NIC.");
                }

                if (!IsValidNIC(booking.NIC))
                {
                    return BadRequest("Invalid NIC format.");
                }

                booking.BookingDate = DateTime.Now;

                _bookingsCollection.InsertOne(booking);

                return Ok(booking);
            }

            return BadRequest("Reservation date must be within 30 days from the
current date.");
        }

        // Validate NIC format (Assuming NIC is a 9-digit number)
        private bool IsValidNIC(string nic)
        {
            // Use a regular expression to validate NIC format
            string nicPattern = @"^\d{12}$";
            return Regex.IsMatch(nic, nicPattern);
        }

        private bool IsValidEmail(string email)
        {
            // Add your email validation logic here
            return Regex.IsMatch(email, @"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-
Z|a-z]{2,}");
        }

        private bool IsValidContactNumber(string contactNumber)
        {
            // Add your contact number validation logic here
```

```
            return Regex.IsMatch(contactNumber, @"^[0-9]{10}$");
        }

        private bool IsValidTicketClass(string ticketClass)
        {
            // Add your ticket class validation logic here
            return ticketClass == "First Class" || ticketClass == "Second Class"
|| ticketClass == "Third Class";
        }

        // Get booking by ID
        [HttpGet]
        [Route("getticketbooking/{id}")]
        public IHttpActionResult GetBookingById(string id)
        {
            var reservation = _bookingsCollection.Find(b => b.BookingID ==
id).FirstOrDefault();

            if (reservation == null)
            {
                return NotFound();
            }
            return Ok(reservation);
        }

        // Get all bookings of a user
        [HttpGet]
        [Route("getallticketbookings/{userId}")]
        public IHttpActionResult GetAllMyBookings(string userId)
        {
            var reservations = _bookingsCollection.Find(b => b.UserID ==
userId).ToList();
            return Ok(reservations);
        }

        // Get all bookings
        [HttpGet]
        [Route("getallticketbookings")]
        public IHttpActionResult GetAllBookings()
        {
            var reservations = _bookingsCollection.Find(_ => true).ToList();
            return Ok(reservations);
        }

        // Update Train Ticket Booking endpoint
```

```
        [HttpPut]
        [Route("updateticketbooking/{id}")]
        public IHttpActionResult UpdateTicketBooking(string id, TicketBooking
updatedBooking)
        {
            var filter = Builders<TicketBooking>.Filter.Eq(t => t.BookingID, id);
            var today = DateTime.Now;

            // Calculate the difference in days between ReservationDate and
BookingDate
            int daysDifference = (updatedBooking.ReservationDate - today).Days;

            if (daysDifference > 5)
            {

                if (!IsValidEmail(updatedBooking.Email))
                {
                    return BadRequest("Invalid email address.");
                }

                if (!IsValidContactNumber(updatedBooking.ContactNumber))
                {
                    return BadRequest("Invalid contact number.");
                }

                if (!IsValidTicketClass(updatedBooking.TicketClass))
                {
                    return BadRequest("Invalid ticket class.");
                }
                var update = Builders<TicketBooking>.Update
                    .Set(t => t.TrainName, updatedBooking.TrainName)
                    .Set(t => t.ReservationDate, updatedBooking.ReservationDate)
                    .Set(t => t.TotalPassengers, updatedBooking.TotalPassengers)
                    .Set(t => t.MainPassengerName,
updatedBooking.MainPassengerName)
                    .Set(t => t.ContactNumber, updatedBooking.ContactNumber)
                    .Set(t => t.DepartureStation,
updatedBooking.DepartureStation)
                    .Set(t => t.DestinationStation,
updatedBooking.DestinationStation)
                    .Set(t => t.Email, updatedBooking.Email)
                    .Set(t => t.NIC, updatedBooking.NIC)
                    .Set(t => t.TicketClass, updatedBooking.TicketClass)
                    .Set(t => t.TotalPrice, updatedBooking.TotalPrice);
```

```
                var result = _bookingsCollection.UpdateOne(filter, update);

                if (result.ModifiedCount == 0)
                {
                    return NotFound();
                }

                return Ok("Train ticket booking updated");
            }

            return BadRequest("Reservation can only be updated if the difference
between Reservation Date and Booking Date is greater than 5 days.");
        }

        // Cancel Train Ticket Booking endpoint
        [HttpPut]
        [Route("cancelticketbooking/{id}")]
        public IHttpActionResult CancelBooking(string id)
        {
            var filter = Builders<TicketBooking>.Filter.Eq(b => b.BookingID, id);
            var existingBooking =
_bookingsCollection.Find(filter).FirstOrDefault();

            if (existingBooking == null)
            {
                return NotFound();
            }

            var today = DateTime.Now;
            var reservationDate = existingBooking.ReservationDate;

            if (reservationDate > today && reservationDate.Subtract(today).Days
>= 5)
            {
                var result = _bookingsCollection.DeleteOne(filter);

                if (result.DeletedCount == 0)
                {
                    return NotFound();
                }

                return Ok("Reservation cancelled and booking deleted.");
            }
```

```
            return BadRequest("Reservation can only be canceled at least 5 days
before the reservation date.");
        }

        // Get booking count for a user
        [HttpGet]
        [Route("getbookingcount/{id}")]
        public IHttpActionResult GetBookingCount(string agentId)
        {
            var bookingCount = _bookingsCollection.CountDocuments(b => b.UserID
== agentId);
            return Ok(bookingCount);
        }
    }
}
```

7.2 Web Application

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';
import { UserProvider } from './Components/AuthContext';
import ARouter from './Components/ARouter';

ReactDOM.render(
  <Router>
    <UserProvider>
      <ARouter />
    </UserProvider>
  </Router>,
  document.getElementById('root')
);
```

SignIn.jsx

```
import React, { useState, useContext } from 'react';
import axios from 'axios';
import { useHistory, Link } from 'react-router-dom';
import { AuthContext } from '../AuthContext';
import { Form, Button, Container, Row, Col, Card } from 'react-bootstrap';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import {IsValidNIC, IsValidPassword} from '../Validations'
```

```
const SignIn = () => {

   // Access the setUser and UserType from the AuthContext
  const { setUser, UserType } = useContext(AuthContext);

  // State for form data
  const [formData, setFormData] = useState({
    NIC: '',
    Password: ''
  });

  // Access to the history object to navigate
  const history = useHistory();

  // Handle changes in form fields
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value
    });
  };

  // Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    if (!IsValidNIC(formData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    if (!IsValidPassword(formData.Password)) {
      toast.error('Invalid Password format.');
      return;
    }

    axios.post('http://pasinduperera-001-site1.atempurl.com/api/users/signin',
formData)
      .then(response => {
        console.log('User authenticated:', response.data);
        setUser(response.data.UserID, response.data.UserType);
        sessionStorage.setItem('userID', response.data.userID);
```

```
        if (response.data.UserType === 'BackOfficeUser') {
          history.push('/backofficeuserdashboard');
          window.location.href="/backofficeuserdashboard";
        } else if (response.data.UserType === 'TravelAgent') {
          history.push('/travelagentdashboard');
          window.location.href="/travelagentdashboard";
        } else {
          history.push('/');
        }
      })
      .catch(error => {
        if (error.response && error.response.status === 401) {
          toast.error('Password or NIC mismatch. Please try again.');
        } else if (error.response && error.response.status === 400) {
          toast.error('Invalid User credentials. Please check your inputs.');
        } else {
          console.error('Error authenticating user:', error);
          toast.error('Invalid User credentials. Please check your inputs.');
        }
      });
  };

  return (
    <Container className="d-flex justify-content-center align-items-center"
style={{ minHeight: '100vh' }}>
      <Card style={{ padding: '25px', width: "1200px", textAlign: "center",
background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
        <Row>
        <Col>
        <img src='https://lp-cms-production.imgix.net/2022-
02/shutterstockRF_376030297.jpg?auto=format&q=75&w=1920' style={{width: "87%"}}/>
        </Col>
          <Col>
        <ToastContainer position="top-center" autoClose={1000} hideProgressBar />
        <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Sign In</Card.Title>
        <Form onSubmit={handleSubmit} style={{textAlign: "left"}}>
          <Form.Group controlId="NIC" style={{ marginBottom: "25px", fontFamily:
"Montserrat" }}>
          <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
            <Form.Control
              type="text"
              name="NIC"
              value={formData.NIC}
```

```
              onChange={handleChange}
              placeholder='NIC'
              required
            />
          </Form.Group>
          <Form.Group controlId="Password" style={{ marginBottom: "25px",
fontFamily: "Montserrat" }}>
          <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Password</Form.Label>
            <Form.Control
              type="password"
              name="Password"
              value={formData.Password}
              onChange={handleChange}
              placeholder='Password'
              required
            />
          </Form.Group>
          <div style={{ textAlign: 'center' }}>
  <Button variant="primary" type="submit" style={{ padding: '10px 20px',
backgroundColor: '#00284d', fontFamily: "Montserrat" }}>
    Sign In
  </Button>
</div>
          </Form>
        <p style={{ marginTop: "15px", fontSize: "1.2em", color: "#555",
fontFamily: "Montserrat" }}>
          Don't have an account? <Link to="/signup" style={{ color: "#00284d",
textDecoration: "none", fontWeight: "bold", fontFamily: "Montserrat" }}>Sign
Up</Link>
        </p>
        </Col>
        </Row>
      </Card>
    </Container>
  );
};

export default SignIn;
```

SignUp.jsx

```
import React, { useState } from 'react';
import axios from 'axios';
```

```
import { Form, Button, Container, Card, Row, Col } from 'react-bootstrap';
import { useHistory, Link } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { IsValidEmail, IsValidPassword, IsValidNIC, IsValidContactNumber } from
'../Validations';

const Signup = () => {

  // State for form data
  const [formData, setFormData] = useState({
    NIC: '',
    UserName: '',
    FirstName: '',
    LastName: '',
    Gender: '',
    Email: '',
    Password: '',
    RePassword: '',
    ContactNumber: '',
    UserType: ''
  });

  // Access to the history object to navigate
  const history = useHistory();

  // Handle changes in form fields
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value
    });
  };

  // Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    if (formData.Password !== formData.RePassword) {
      toast.error('Passwords do not match.');
      return;
    }

    if (!IsValidEmail(formData.Email)) {
```

```
      toast.error('Invalid email format.');
      return;
    }

    if (!IsValidNIC(formData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    if (!IsValidPassword(formData.Password)) {
      toast.error('Invalid Password format.');
      return;
    }

    if (!IsValidContactNumber(formData.ContactNumber)) {
      toast.error('Invalid contact number format.');
      return;
    }

    axios.post('http://pasinduperera-001-site1.atempurl.com/api/users/signup',
formData, {
      headers: {
        'Content-Type': 'application/json',
      },
    })
    .then(response => {
      console.log('Success:', response.data);
      toast.success('User registered successfully!');
      history.push('/');
    })
    .catch(error => {
      console.error('Error:', error);
      toast.error('Error registering user. Please try again.');
    });
  };

  return (
    <div>
    <Container style={{width:"75%", marginTop: "47px", marginBottom: "48px"}}>
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
        <Row>
          <Col>
          <img
src='https://www.telegraph.co.uk/content/dam/travel/2023/03/10/TELEMMGLPICT000328
```

```
038771_trans_NvBQzQNjv4BqqVzuuqpFlyLIwiB6NTmJwSX5rhseiWKOo9p9OQ-ymek.jpeg'
style={{width: "100%", margin: "25px", height: "84%"}}/>
          </Col>
          <Col>
        <Card.Body>
        <ToastContainer position="top-center" autoClose={1000} hideProgressBar />
        <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px", textAlign: "center" }}>Sign Up</Card.Title>
          <Form onSubmit={handleSubmit}>
            <Row>
              <Col>
            <Form.Group controlId="NIC" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
              <Form.Control
                type="text"
                name="NIC"
                style={{fontFamily: "Montserrat"}}
                value={formData.NIC}
                onChange={handleChange}
                placeholder="NIC"
                required
              />
            </Form.Group>
            <Form.Group controlId="UserName" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Name</Form.Label>
              <Form.Control
                type="text"
                name="UserName"
                style={{fontFamily: "Montserrat"}}
                value={formData.UserName}
                onChange={handleChange}
                placeholder="Username"
                required
              />
            </Form.Group>
            <Form.Group controlId="FirstName" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>First Name</Form.Label>
              <Form.Control
                type="text"
                name="FirstName"
                style={{fontFamily: "Montserrat"}}
                value={formData.FirstName}
```

```
                onChange={handleChange}
                placeholder="First Name"
                required
              />
            </Form.Group>
            <Form.Group controlId="LastName" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Last
Name</Form.Label>
              <Form.Control
                type="text"
                name="LastName"
                style={{fontFamily: "Montserrat"}}
                value={formData.LastName}
                onChange={handleChange}
                placeholder="Last Name"
                required
              />
            </Form.Group>

            <Form.Group controlId="Email" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email</Form.Label>
              <Form.Control
                type="Email"
                name="Email"
                style={{fontFamily: "Montserrat"}}
                value={formData.Email}
                onChange={handleChange}
                placeholder="Email"
                required
              />

              </Form.Group>
              </Col>
            <Col>
            <Form.Group controlId="Gender" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Gender</Form.Label>
    <Form.Control
        as="select"
        name="Gender"
        style={{fontFamily: "Montserrat"}}
        value={formData.Gender}
        onChange={handleChange}
        required
```

```
    >
        <option value="">Select Gender</option>
        <option value="male">Male</option>
        <option value="female">Female</option>
    </Form.Control>
</Form.Group>
            <Form.Group controlId="ContactNumber" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Contact Number</Form.Label>
              <Form.Control
                type="text"
                name="ContactNumber"
                style={{fontFamily: "Montserrat"}}
                value={formData.ContactNumber}
                onChange={handleChange}
                placeholder="Contact Number"
                required
              />
            </Form.Group>
            <Form.Group controlId="Password" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Password</Form.Label>
              <Form.Control
                type="password"
                name="Password"
                style={{fontFamily: "Montserrat"}}
                value={formData.Password}
                onChange={handleChange}
                placeholder="Password"
                required
              />
            </Form.Group>
            <Form.Group controlId="RePassword" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Re
enter Password</Form.Label>
              <Form.Control
                type="password"
                name="RePassword"
                style={{fontFamily: "Montserrat"}}
                value={formData.RePassword}
                onChange={handleChange}
                placeholder="Re-enter Password"
                required
              />
            </Form.Group>
```

```
            <Form.Group controlId="UserType" style={{marginBottom:"7px"}}>
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Type</Form.Label>
              <Form.Control
                as="select"
                name="UserType"
                style={{fontFamily: "Montserrat"}}
                value={formData.UserType}
                onChange={handleChange}
                required
              >
                <option value="">Select User Type</option>
                <option value="BackOfficeUser">Backoffice User</option>
                <option value="travelAgent">Travel Agent</option>
              </Form.Control>
            </Form.Group>
            </Col>
          </Row>
          <Row className="justify-content-center">
              <Col xs="auto">
            <Button type="submit" variant="primary" style={{ width: '150px',
backgroundColor: '#00284d', fontFamily: "Montserrat", marginTop: "27px" }}>Sign
Up</Button>
              </Col>
            </Row>
          <div className="text-center mt-2">
            <p style={{ marginTop: "15px", fontSize: "1.2em", color: "#555",
fontFamily: "Montserrat" }}>
  Don't have an account? <Link to="/" style={{ color: "#00284d", textDecoration:
"none", fontWeight: "bold", fontFamily: "Montserrat" }}>Sign In</Link>
</p>
          </div>
          </Form>
        </Card.Body>
        </Col>
        </Row>
      </Card>
    </Container>
    </div>
  );
};

export default Signup;
```

UserProfile.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams } from 'react-router-dom';
import { useHistory } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Container, Table, Row, Col, Button, Card } from 'react-bootstrap';
import imageprofileavatar from '../Assests/profileavatar.png'

const UserProfile = () => {

  const history = useHistory();

    // Get the user ID from URL parameters
  const { userId } = useParams();

  // State to store user data
  const [user, setUser] = useState(null);

  useEffect(() => {
    // Function to fetch user data
    const fetchData = async () => {
      try {
        const response = await axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/users/getuser/${userId}`);
        setUser(response.data);
      } catch (error) {
        console.error('Error fetching user:', error);
      }
    };

    // Check if userId exists before fetching data
    if (userId) {
      fetchData();
    }
  }, [userId]);

  // Function to handle update button click
  const handleUpdate = () => {
    history.push(`/updateuserprofile/${userId}`);
  };

  // Function to handle delete button click
```

```
  const handleDelete = async () => {
    try {
      await axios.delete(`http://pasinduperera-001-
site1.atempurl.com/api/users/deletuser/${userId}`);
      toast.success('User deleted successfully!');
      window.location.href="/"
    } catch (error) {
      console.error('Error deleting user:', error);
      toast.error('Error deleting user. Please try again later.');
    }
  };

  if (!userId) {
    return <div>No user ID found</div>;
  }

  if (!user) {
    return <div>Loading...</div>;
  }

  return (
    <Container className="my-5 text-center" style={{ paddingLeft: "250px" }}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
        <Card.Body>
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px", color: "#00284d" }}>My Profile</Card.Title>
          <div className="text-center mb-4">
            <img src={imageprofileavatar} alt="Profile" style={{ width: '250px',
height: '170px', borderRadius: '50%' }} />
          </div>
          <Table striped bordered responsive style={{ fontFamily: "Onest" }}>
            <tbody>
            <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>NIC</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.NIC}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Username</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserName}</td>
      </tr>
      <tr>
```

```
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>First
Name</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.FirstName}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Last
Name</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.LastName}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Email</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.Email}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Gender</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.Gender}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Phone
Number</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.ContactNumber}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>User
Type</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserType}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Status</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserStatus}</td>
      </tr>
            </tbody>
          </Table>
          <Row className="justify-content-center" style={{ margin: "25px" }}>
        <Col xs="auto">
        <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px', backgroundColor: '#00284d', fontFamily: "Montserrat"
}}>Back</Button>{' '}
          <Button variant="secondary" onClick={handleUpdate} style={{ width:
'150px', backgroundColor: '#006600', fontFamily: "Montserrat", marginRight:
'10px' }}>Update</Button>{' '}
```

```
          <Button variant="danger" onClick={handleDelete} style={{ width:
'150px', fontFamily: "Montserrat", marginRight: '10px' }}>Delete</Button>{' '}
        </Col>
      </Row>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default UserProfile;
```

UpdateUserprofile.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams, useHistory } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Container, Form, Button, Row, Col, Card } from 'react-bootstrap';
import { IsValidEmail, IsValidPassword, IsValidNIC, IsValidContactNumber } from
'../Validations';
import imageprofileavatar from '../Assests/profileavatar.png'

const UpdateTraveller = () => {

  // Extracting UserID from the URL parameters
  const { UserID } = useParams();

  // Initializing history hook for navigation
  const history = useHistory();

  // Initializing state to hold user data
  const [userData, setUserData] = useState({
    UserName: '',
    FirstName: '',
    LastName: '',
    Email: '',
    Gender: '',
    ContactNumber: '',
    UserType: '',
  });

  // Fetching user data on component mount
```

```
  useEffect(() => {
    axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/users/getuser/${UserID}`)
      .then(response => {
        setUserData(response.data);
      })
      .catch(error => {
        console.error('Error fetching user data:', error);
      });
  }, [UserID]);

  // Handler for input changes
  const handleChange = (e) => {
    const { name, value } = e.target;
    setUserData({
      ...userData,
      [name]: value,
    });
  };

  // Handler for form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    // Validate email format
    if (!IsValidEmail(userData.Email)) {
      toast.error('Invalid email format.');
      return;
    }

    // Validate NIC format
    if (!IsValidNIC(userData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    // Validate contact number format
    if (!IsValidContactNumber(userData.ContactNumber)) {
      toast.error('Invalid contact number format.');
      return;
    }

    // Sending a PUT request to update user data
    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/users/updateuser/${UserID}`, userData)
```

```
      .then(response => {
        console.log('User updated:', response.data);
        setTimeout(() => {
          toast.success('user updated successfully!');
        }, 2000)
      })
      .catch(error => {
        console.error('Error updating user:', error);
      });
  };

  return (
    <Container className="my-5 text-center" style={{width: "1200px", paddingLeft:
"250px"}}>
      <ToastContainer position="top-center" autoClose={1000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
            <Card.Body>
  <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One", fontSize:
"34px" }}>Update User</Card.Title>
  <div className="text-center mb-4">
  <img src={imageprofileavatar} alt="Profile" style={{ width: '250px', height:
'170px', borderRadius: '50%' }} />
          </div>
  <Form onSubmit={handleSubmit}>
  <div className="row">
  <div className="col-md-6" style={{textAlign: "left"}}>
  <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
      <Form.Control
        type="text"
        id="NIC"
        name="NIC"
        placeholder='NIC'
        value={userData.NIC}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
        disabled
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Name</Form.Label>
```

```
      <Form.Control
        type="text"
        id="UserName"
        name="UserName"
        placeholder='User Name'
        value={userData.UserName}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>First
Name</Form.Label>
      <Form.Control
        type="text"
        id="FirstName"
        name="FirstName"
        placeholder='First Name'
        value={userData.FirstName}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Last
Name</Form.Label>
      <Form.Control
        type="text"
        id="LastName"
        name="LastName"
        placeholder='Last Name'
        value={userData.LastName}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    </div>
    <div className="col-md-6" style={{textAlign: "left"}}>
    <Form.Group>
```

```
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email</Form.Label>
      <Form.Control
        type="Email"
        id="Email"
        name="Email"
        placeholder='Email'
        value={userData.Email}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group controlId="Gender" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Gender</Form.Label>
  <Form.Select
    name="Gender"
    value={userData.Gender}
    style={{fontFamily: "Onest"}}
    onChange={handleChange}
    required
  >
    <option value="">Select Gender</option>
    <option value="Male">Male</option>
    <option value="Female">Female</option>
    <option value="Other">Other</option>
  </Form.Select>
</Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Phone
Number</Form.Label>
      <Form.Control
        type="text"
        id="ContactNumber"
        name="ContactNumber"
        placeholder='Contact Number'
        value={userData.ContactNumber}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
```

```
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Status</Form.Label>
      <Form.Control
        type="text"
        id="UserStatus"
        name="UserStatus"
        placeholder='User Status'
        value={userData.UserStatus}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
        disabled
      />
    </Form.Group>
    <br/>
    </div>
            </div>
            <Row className="justify-content-center" style={{margin: "25px"}}>
              <Col xs="auto">
              <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px' }}>Back</Button>{' '}
            <Button type="submit" variant="primary" style={{ width: '150px',
backgroundColor: "#00284d" }} onClick={() => window.history.back()} >Update
Profile</Button>
              </Col>
            </Row>
  </Form>
  </Card.Body>
  </Card>
</Container>
  );
};

export default UpdateTraveller;
```

BackOfficeUserDashboard.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Container, Row, Col, Card, CardBody } from 'react-bootstrap';
```

```
const BackOfficeUserDashboard = () => {
  // State variables to store user counts
  const [travelAgentCount, setTravelAgentCount] = useState(0);
  const [backofficeUserCount, setBackofficeUserCount] = useState(0);
  const [travelUserCount, setTravelUserCount] = useState(0);
  const [trainshedulerCount, setTrainSheduleCount] = useState(0);

  // Fetch counts from the API
  useEffect(() => {
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/getbackofficeusercount')
      .then(response => {
        setBackofficeUserCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });

    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/gettravelagentcount')
      .then(response => {
        setTravelAgentCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });

    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/gettravelusercount')
      .then(response => {
        setTravelUserCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });

  axios.get('http://pasinduperera-001-
site1.atempurl.com/api/trains/gettraincount')
      .then(response => {
        setTrainSheduleCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });
  }, []);
```

```
  return (
    <Container className="text-center" style={{ paddingLeft: "250px",
marginBottom: "25px", marginTop: "25px" }}>
      <Card style={{ padding: "1px", background: 'rgba(255, 255, 255, 0.7)',
border: 'none' }}>
        <Card.Body>
          <Row className="mb-4">
            <Col>
              {/* Dashboard title */}
              <Card.Title style={{ textAlign: "center", fontSize: "34px",
padding: "25px", width: "600px", fontFamily: "Dela Gothic One" }}>
                BackOffice User Dashboard
              </Card.Title>
              {/* Dashboard image */}
              <Card.Img
src='https://th.bing.com/th/id/OIP.qZl2uQ0RJQTxq2DMJsjsAQHaC5?pid=ImgDet&w=1280&h
=500&rs=1' style={{ height: "200px", marginBottom: "17px" }} />
              <CardBody>
                <Card.Text style={{fontFamily: "Onest", textAlign:
"justify"}}>The management of administrative activities and assistance for the
online train ticket booking system is the responsibility of back office users.
                  To guarantee the system runs well, they take care of tasks
including train shedule management, user account administration.</Card.Text>
              </CardBody>
            </Col>
            <Col>
              {/* Card displaying Back Office Users count */}
              <Card style={{width: "75%", margin: "33px", marginBottom: "7%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
                  <Card.Title>Number of Backofficers</Card.Title>
                  <Card.Text>{backofficeUserCount}</Card.Text>
                </Card.Body>
              </Card>
              {/* Card displaying Travel Agents count */}
              <Card style={{width: "75%", margin: "33px", marginBottom: "8%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
                  <Card.Title>Number of Travel Agents</Card.Title>
                  <Card.Text>{travelAgentCount}</Card.Text>
                </Card.Body>
              </Card>
              {/* Card displaying Travellers count */}
              <Card style={{width: "75%", margin: "34px", marginBottom: "8%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
```

```
                <Card.Title>Number of Travelers</Card.Title>
                <Card.Text>{travelUserCount}</Card.Text>
              </Card.Body>
            </Card>
            {/* Card displaying Train Shedule count */}
            <Card style={{width: "75%", margin: "34px", marginBottom: "7%"}}>
              <Card.Body style={{fontFamily: "Montserrat"}}>
                <Card.Title>Number of Train Shedules</Card.Title>
                <Card.Text>{trainshedulerCount}</Card.Text>
              </Card.Body>
            </Card>
          </Col>
        </Row>
      </Card.Body>
    </Card>
  </Container>
 );
};

export default BackOfficeUserDashboard;
```

TravelAgentDashboard.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Container, Row, Col, Card, CardBody } from 'react-bootstrap';

const TravelAgentDashboard = () => {
  // State variables to store user counts
  const [travelAgentCount, setTravelAgentCount] = useState(0);
  const [backofficeUserCount, setBackofficeUserCount] = useState(0);
  const [travelUserCount, setTravelUserCount] = useState(0);

  useEffect(() => {
    // Fetch user counts from the API for back office users, travel agents, and
travel users
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/getbackofficeusercount')
      .then(response => {
        setBackofficeUserCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });
```

```
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/gettravelagentcount')
      .then(response => {
        setTravelAgentCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });

    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/gettravelusercount')
      .then(response => {
        setTravelUserCount(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });
  }, []);

  return (
    <Container className="text-center" style={{ paddingLeft: "250px",
marginBottom: "25px", marginTop: "25px" }}>
      <Card style={{ padding: "1px", height: "537px", background: 'rgba(255, 255,
255, 0.7)', border: 'none' }}>
        <Card.Body>
          <Row className="mb-4">
            <Col>
              {/* Dashboard title */}
              <Card.Title style={{ textAlign: "center", fontSize: "34px",
padding: "25px", width: "600px", fontFamily: "Dela Gothic One" }}>
                Travel Agent Dashboard
              </Card.Title>
              {/* Dashboard image */}
              <Card.Img
src='https://t4.ftcdn.net/jpg/02/90/79/53/360_F_290795351_i0kJX32HPEdjTgK1iSjaDjT
b9MN3qk4O.jpg' style={{ height: "200px", marginBottom: "17px" }} />
              <CardBody>
                <Card.Text style={{fontFamily: "Onest", textAlign: "justify"}}>
                  With a focus on train booking management and traveler
management, travel agents are essential to the travel sector. They professionally
plan clients' lodging and travel arrangements, assuring hassle-free and
delightful trips. They also offer thorough assistance to tourists, from
preliminary questions to aid following a trip, ensuring a smooth journey. Travel
```

agents are essential in delivering memorable and stress-free travel experiences thanks to their attention to detail and individualized service.

```
                </Card.Text>
              </CardBody>
            </Col>
            <Col>
              {/* Card displaying Travel Agents count */}
              <Card style={{width: "75%", margin: "24px", marginTop: "17%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
                  <Card.Title>Number of Travel Agents</Card.Title>
                  <Card.Text>{travelAgentCount}</Card.Text>
                </Card.Body>
              </Card>
              {/* Card displaying Back Office Users count */}
              <Card style={{width: "75%", margin: "24px", marginTop: "17%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
                  <Card.Title>Number of Back Office Users</Card.Title>
                  <Card.Text>{backofficeUserCount}</Card.Text>
                </Card.Body>
              </Card>
              {/* Card displaying Travel Users count */}
              <Card style={{width: "75%", margin: "24px", marginTop: "17%"}}>
                <Card.Body style={{fontFamily: "Montserrat"}}>
                  <Card.Title>Number of Travel Users</Card.Title>
                  <Card.Text>{travelUserCount}</Card.Text>
                </Card.Body>
              </Card>
            </Col>
          </Row>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default TravelAgentDashboard;
```

AddTrainShedule.jsx

```
import React, { useState, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from '../AuthContext';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
```

```
import { Form, Button, Container, Row, Col, Card } from 'react-bootstrap';
import { useHistory } from 'react-router-dom';
import { IsValidTrainNumber } from '../Validations';

const AddTrainShedule = () => {
  // Getting userID from AuthContext
  const { userID } = useContext(AuthContext);

  // State for submission status
  const [ setSubmissionSuccessful] = useState(false);

  // State for storing train data
  const [trainData, setTrainData] = useState({
    TrainNumber: '',
    userID: userID,
    TrainName: '',
    TrainDriver: '',
    DepartureStation: '',
    ArrivalStation: '',
    DepartureTime: '',
    ArrivalTime: '',
    TrainType: '',
    FirstClassTicketPrice: '',
    SecondClassTicketPrice: '',
    ThirdClassTicketPrice: '',
    TrainStatus: 'Sheduled',
  });

  // Getting history object for navigation
  const history = useHistory();

  // Function to handle input changes
  const handleChange = (e) => {
    const { name, value } = e.target;
    setTrainData({
      ...trainData,
      [name]: value,
    });
  };

  // Function to handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    // Validating Train Number format
```

```
  if (!IsValidTrainNumber(trainData.TrainNumber)) {
    toast.error('Invalid Train Number. Please enter a valid Train Number format
(TXXXX).');
    return;
}

    const TrainIDPattern = /^[A-Z]\d{4}$/;

    if (!TrainIDPattern.test(trainData.TrainNumber)) {
      toast.error('Invalid Train Number. Please enter a valid Train Number format
(TXXXX).');
      return;
    }
    axios.post('http://pasinduperera-001-
site1.atempurl.com/api/trains/createtrain', trainData)
      .then(response => {
        toast.success("Train Added");
        console.log('Train added:', response.data);
      setSubmissionSuccessful(true);
      setTimeout(() => {
        history.push('/backofficeuserdashboard');
      }, 2000);
      })
      .catch(error => {
        toast.error('Departure Time must be before Arrival Time.');
      });
  };

  return (
    <Container className="text-center mt-5" style={{width: "1200px", paddingLeft:
"250px", marginBottom: "25px"}}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Row className="justify-content-center">
        <Col>
          <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none'
}}>
            <Card.Body>
              <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Create New Train Schedule</Card.Title>
              <Form onSubmit={handleSubmit}>
              <div className="row">
  <div className="col-md-6" style={{textAlign: "left"}}>
                <Form.Group controlId="TrainID" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
```

```
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Number</Form.Label>
                  <Form.Control
                    type="text"
                    name="TrainNumber"
                    placeholder='Train Number'
                    style={{fontFamily: "Onest"}}
                    value={trainData.TrainNumber}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="trainName" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Name</Form.Label>
                  <Form.Control
                    type="text"
                    name="TrainName"
                    placeholder='Train Name'
                    style={{fontFamily: "Onest"}}
                    value={trainData.TrainName}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="trainDriver" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Driver</Form.Label>
                  <Form.Control
                    type="text"
                    name="TrainDriver"
                    placeholder='Train Driver'
                    style={{fontFamily: "Onest"}}
                    value={trainData.TrainDriver}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="departureStation" style={{fontSize:
"17px", fontFamily: "Montserrat"}}>
```

```
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Departure Station</Form.Label>
                  <Form.Control
                    type="text"
                    name="DepartureStation"
                    placeholder='Departure Station'
                    style={{fontFamily: "Onest"}}
                    value={trainData.DepartureStation}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="arrivalStation" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Arrival Station</Form.Label>
                  <Form.Control
                    type="text"
                    name="ArrivalStation"
                    placeholder='Arrival Station'
                    style={{fontFamily: "Onest"}}
                    value={trainData.ArrivalStation}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="trainStatus" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Status</Form.Label>
                  <Form.Control
                    type="text"
                    name="trainStatus"
                    placeholder='Train Status'
                    style={{fontFamily: "Onest"}}
                    value="Active"
                    onChange={handleChange}
                    required
                    disabled
                  />
                </Form.Group>
                <br/>
                </div>
```

```
                <div className="col-md-6" style={{textAlign: "left"}}>
                <Form.Group controlId="departureTime" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Departure Time</Form.Label>
                  <Form.Control
                    type="datetime-local"
                    name="DepartureTime"
                    placeholder='Departure Time'
                    style={{fontFamily: "Onest"}}
                    value={trainData.DepartureTime}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="arrivalTime" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Arrival Time</Form.Label>
                  <Form.Control
                    type="datetime-local"
                    name="ArrivalTime"
                    placeholder='Arrival Time'
                    style={{fontFamily: "Onest"}}
                    value={trainData.ArrivalTime}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>

                <Form.Group controlId="trainType" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
  <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Train
Type</Form.Label>
  <Form.Control
    as="select"
    name="TrainType"
    placeholder='Train Type'
    style={{fontFamily: "Onest"}}
    value={trainData.TrainType}
    onChange={handleChange}
    required
  >
```

```
    <option value="">Select Train Type</option>
    <option value="Express">Express</option>
    <option value="Intercity">Intercity</option>
    <option value="Local">Local</option>
  </Form.Control>
</Form.Group>
                <br/>
                <Form.Group controlId="firstClassTicketPrice" style={{fontSize:
"17px", fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>First Class Ticket Price</Form.Label>
                  <Form.Control
                    type="text"
                    name="FirstClassTicketPrice"
                    placeholder='First Class Ticket price'
                    style={{fontFamily: "Onest"}}
                    value={trainData.FirstClassTicketPrice}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="secondClassTicketPrice" style={{fontSize:
"17px", fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Second Class Ticket Price</Form.Label>
                  <Form.Control
                    type="text"
                    name="SecondClassTicketPrice"
                    placeholder='Second Class Ticket price'
                    value={trainData.SecondClassTicketPrice}
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                <Form.Group controlId="thirdClassTicketPrice" style={{fontSize:
"17px", fontFamily: "Montserrat"}}>
                  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Third Class Ticket Price</Form.Label>
                  <Form.Control
                    type="text"
                    name="ThirdClassTicketPrice"
                    placeholder='Third Class Ticket price'
                    value={trainData.ThirdClassTicketPrice}
```

```
                    onChange={handleChange}
                    required
                  />
                </Form.Group>
                <br/>
                </div>
                </div>
                <Row className="justify-content-center">
                  <Col xs="auto" style={{ margin: "34px" }}>
                    <Button variant="secondary" onClick={() =>
window.history.back()} style={{ width: '150px' }}>Back</Button>{' '}
                    <Button type="submit" variant="primary" style={{ width:
'150px', backgroundColor: "#00284d" }}>Submit</Button>
                  </Col>
                </Row>
              </Form>
            </Card.Body>
          </Card>
        </Col>
      </Row>
    </Container>
  );
};

export default AddTrainShedule;
```

GetAllTrainShedules.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Table, Button, Card, Container, Form } from 'react-bootstrap';

const GetAllTrainShedules = () => {
  // State for storing train data
  const [trains, setTrains] = useState([]);

  // State for current page in pagination
  const [currentPage, setCurrentPage] = useState(1);

  // State for search query
  const [searchQuery, setSearchQuery] = useState('');
```

```
  // Number of items per page
  const itemsPerPage = 5;

  // Calculate index of last and first item on current page
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;

  // Function to handle pagination
  const handlePagination = (pageNumber) => {
    setCurrentPage(pageNumber);
  };

  // Function to handle search input change
  const handleSearchChange = (e) => {
    setSearchQuery(e.target.value);
  };

  // Fetching train data on component mount
  useEffect(() => {
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/trains/getalltrains')
      .then(response => {
        setTrains(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });
  }, []);

  // Function to handle train deletion
  const handleDelete = (TrainID) => {
    axios.delete(`http://pasinduperera-001-
site1.atempurl.com/api/trains/deletetrain/${TrainID}`)
      .then(response => {
        toast.success('Train successfully deleted!');
        setTimeout(() => {
          window.location.reload();
        }, 2000)
      })
      .catch(error => {
        console.error('Error:', error);
        toast.error('Error deleting train. Please try again later.');
      });
  };
```

```
  // Filtered trains based on search query
  const filteredTrains = trains.filter(train =>
    train.TrainNumber.toLowerCase().includes(searchQuery.toLowerCase())
  );

  // Get current items to display on the page
  const currentItems = filteredTrains.slice(indexOfFirstItem, indexOfLastItem);

  return (
    <Container className="my-5 text-center" style={{ paddingLeft: "250px",
maxWidth: "900px", height: "530px" }}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
        <Card.Body>
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Train Schedules</Card.Title>
          <Form.Control
  type="text"
  placeholder="Search by Train Number"
  value={searchQuery}
  onChange={handleSearchChange}
  style={{ marginBottom: '10px' }}
/>
          <Table striped bordered hover style={{ marginTop: '20px', width: '75%'
}} className="mx-auto" responsive>
            <thead>
              <tr style={{ fontSize: "17px", fontFamily: "Montserrat" }}>
                <th>Train ID</th>
                <th>Status</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {currentItems.map(train => (
                <tr key={train.ID} style={{ fontFamily: "Onest" }}>
                  <td>{train.TrainNumber}</td>
                  <td>{train.TrainStatus}</td>
                  <td>
                    <Link to={`/viewtrainshedule/${train.TrainID}`}>
                      <Button variant="warning" style={{ marginRight: '5px',
color: 'white' }}><i className="fas fa-eye"></i></Button>
                    </Link>
                    <Link to={`/updatetrainshedule/${train.TrainID}`}>
```

```
                    <Button variant="success" style={{ marginRight: '5px' }}><i
className="fas fa-edit"></i></Button>
                  </Link>
                  <Button variant="danger" onClick={() =>
handleDelete(train.TrainID)} style={{ marginRight: '5px' }}>
                    <i className="fas fa-trash-alt"></i>
                  </Button>
                </td>
              </tr>
            ))}
          </tbody>
        </Table>

        <div className="pagination" style={{ textAlign: 'Right', margin:
"20px", marginLeft: "40%" }}>
          <span
            onClick={() => currentPage > 1 && handlePagination(currentPage -
1)}
            className={currentPage === 1 ? 'disabled' : ''}
            style={{ margin: "0 5px", cursor: "pointer" }}
          >
            &#8249;
          </span>

          {Array.from({ length: Math.ceil(filteredTrains.length / itemsPerPage)
}).map((_, index) => (
            <span
              key={index}
              onClick={() => handlePagination(index + 1)}
              className={currentPage === index + 1 ? 'active' : ''}
              style={{ margin: "0 5px", cursor: "pointer" }}
            >
              {index + 1}
            </span>
          ))}

          <span
            onClick={() => currentPage < Math.ceil(filteredTrains.length /
itemsPerPage) && handlePagination(currentPage + 1)}
            className={currentPage === Math.ceil(filteredTrains.length /
itemsPerPage) ? 'disabled' : ''}
            style={{ margin: "0 5px", cursor: "pointer" }}
          >
            &#8250;
          </span>
```

```
          </div>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default GetAllTrainShedules;
```

GetTrainShedule.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Table, Row, Col, Button, Card, Container } from 'react-bootstrap';

const GetTrainShedule = () => {
  // State for storing train data
  const [train, setTrain] = useState({});

  // Get TrainID from route parameters
  const { TrainID } = useParams();

  // Fetch train data based on TrainID
  useEffect(() => {
    axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trains/gettrainbyId/${TrainID}`)
      .then(response => {
        setTrain(response.data);
      })
      .catch(error => {
        console.error('Error:', error);
      });
  }, [TrainID]);

  return (
    <Container className="my-5 text-center" style={{ width: "1200px",
paddingLeft: "250px" }}>
  <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
  <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
    <Card.Body>
```

```
      <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>View Train Schedule</Card.Title>
      <div className="mx-auto" style={{ maxWidth: '600px' }}>
        <Table striped bordered hover style={{ fontFamily: "Onest" }}>
            <tbody>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Train
Number</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.TrainNumber}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Train
Name</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.TrainName}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Train
Driver</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.TrainDriver}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Departure
Station</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.DepartureStation}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Arrival
Station</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.ArrivalStation}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Departure
Time</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.FormattedDepartureTime}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Arrival
Time</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.FormattedArrivalTime}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Train
Type</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.TrainType}</td>
  </tr>
```

```
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>First Class
Ticket Price</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.FirstClassTicketPrice}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Second Class
Ticket Price</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.SecondClassTicketPrice}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Third Class
Ticket Price</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.ThirdClassTicketPrice}</td>
  </tr>
  <tr>
    <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Status</strong></td>
    <td style={{fontFamily: "Onest"}}>{train.TrainStatus}</td>
  </tr>
</tbody>
</Table>
        <Row className="justify-content-center" style={{ margin: '25px' }}>
          <Col xs="auto">
            <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px', backgroundColor: '#00284d', fontFamily: "Montserrat"
}}>Back</Button>{' '}
          </Col>
        </Row>
      </div>
    </Card.Body>
  </Card>
</Container>
  );
};

export default GetTrainShedule;
```

UpdateTrainShedule.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Container, Form, Button, Row, Col, Card } from 'react-bootstrap';
import { ToastContainer, toast } from 'react-toastify';
```

```
import 'react-toastify/dist/ReactToastify.css';
import { useParams } from 'react-router-dom';
import { useHistory } from 'react-router-dom';
import { IsValidTrainNumber } from '../Validations';

const UpdateTrainShedule = () => {
  // Get TrainID from route parameters
  const { TrainID } = useParams();

  // State for storing train ID (not sure if you intended to use it, if not, you
can remove it)
  const [ setTrainID] = useState('');

  // State for storing updated train data
  const [updatedTrainData, setUpdatedTrainData] = useState({
    TrainNumber: '',
    TrainName: '',
    TrainDriver: '',
    DepartureStation: '',
    ArrivalStation: '',
    DepartureTime: '',
    ArrivalTime: '',
    TrainType: '',
    FirstClassTicketPrice: '',
    SecondClassTicketPrice: '',
    ThirdClassTicketPrice: '',
    TrainStatus: ''
  });

  const history = useHistory();

  const handleChange = (e) => {
    const { name, value } = e.target;
    setUpdatedTrainData({
      ...updatedTrainData,
      [name]: value,
    });
  };

  const handleTrainIDChange = (e) => {
    setTrainID(e.target.value);
};

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
    // Validate Train Number format
    if (!IsValidTrainNumber(updatedTrainData.TrainNumber)) {
      toast.error('Invalid Train Number. Please enter a valid Train Number format
(TXXXX).');
      return;
  }

  // Send PUT request to update train data
    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/trains/updatetrain/${TrainID}`, updatedTrainData)
      .then(response => {
        console.log('Train updated:', response.data);
        toast.success('Train updated successfully!');
        setTimeout(() => {
        history.push('/trainshedulelist');
        }, 2000)
      })
      .catch(error => {
        console.error('Error:', error);
        toast.error('Departure Time must be before Arrival Time.');
      });
  };

  // Fetch train data based on TrainID
  useEffect(() => {
    if (TrainID) {
      axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trains/gettrainbyId/${TrainID}`)
        .then(response => {
          setUpdatedTrainData(response.data);
        })
        .catch(error => {
          console.error('Error fetching train data:', error);
        });
    }
  }, [TrainID]);

  return (
    <Container className="text-center mt-5" style={{width: "1200px", paddingLeft:
"250px", marginBottom: "27px"}}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
        <Card.Body>
```

```
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Update Train Shedule</Card.Title>
      <Form onSubmit={handleSubmit}>
      <div className="row">
  <div className="col-md-6" style={{textAlign: "left"}}>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Number</Form.Label>
            <Form.Control
              type="text"
              placeholder="Train Number"
              style={{fontFamily: "Onest"}}
              value={updatedTrainData.TrainNumber}
              onChange={handleTrainIDChange}
              disabled
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Name</Form.Label>
            <Form.Control
              type="text"
              name="TrainName"
              style={{fontFamily: "Onest"}}
              placeholder="Train Name"
              value={updatedTrainData.TrainName}
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Driver</Form.Label>
            <Form.Control
              type="text"
              name="TrainDriver"
              style={{fontFamily: "Onest"}}
              placeholder="Train Driver"
              value={updatedTrainData.TrainDriver}
              onChange={handleChange}
            />
```

```
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Departure Station</Form.Label>
            <Form.Control
              type="text"
              name="DepartureStation"
              style={{fontFamily: "Onest"}}
              placeholder="Departure Station"
              value={updatedTrainData.DepartureStation}
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Arrival Station</Form.Label>
            <Form.Control
              type="text"
              name="ArrivalStation"
              style={{fontFamily: "Onest"}}
              placeholder="Arrival Station"
              value={updatedTrainData.A}
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
          <Form.Group controlId="trainType" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
  <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Train
Type</Form.Label>
  <Form.Control
    as="select"
    name="TrainType"
    placeholder='Train Type'
    style={{fontFamily: "Onest"}}
    value={updatedTrainData.TrainType}
    onChange={handleChange}
    required
  >
```

```
    <option value="">Select Train Type</option>
    <option value="Express">Express</option>
    <option value="Intercity">Intercity</option>
    <option value="Local">Local</option>
  </Form.Control>
</Form.Group>
          </Col>
        </Row>

        </div>
  <div className="col-md-6" style={{textAlign: "left"}}>
  <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Departure Time</Form.Label>
            <Form.Control
              type="datetime-local"
              style={{fontFamily: "Onest"}}
              name="DepartureTime"
              placeholder="Departure Time"
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Arrival Time</Form.Label>
            <Form.Control
              type="datetime-local"
              name="ArrivalTime"
              placeholder="Arrival Time"
              style={{fontFamily: "Onest"}}
              onChange={handleChange}
            />
            </Col>


        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat",
marginTop: "17px"}}>First Class Ticket Price (Rs.)</Form.Label>
            <Form.Control
              type="text"
              name="FirstClassTicketPrice"
```

```
              style={{fontFamily: "Onest"}}
              placeholder="First Class Ticket Price"
              value={updatedTrainData.FirstClassTicketPrice}
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Second Class Ticket Price (Rs.)</Form.Label>
            <Form.Control
              type="text"
              name="SecondClassTicketPrice"
              style={{fontFamily: "Onest"}}
              placeholder="Second Class Ticket Price"
              value={updatedTrainData.SecondClassTicketPrice}
              onChange={handleChange}
            />
          </Col>
        </Row>
        <Row className="mb-3">
          <Col className="mx-auto">
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Third Class Ticket Price (Rs.)</Form.Label>
            <Form.Control
              type="text"
              name="ThirdClassTicketPrice"
              style={{fontFamily: "Onest"}}
              placeholder="Third Class Ticket Price"
              value={updatedTrainData.ThirdClassTicketPrice}
              onChange={handleChange}
            />
          </Col>
        </Row>

        </Row>
        <Row className="mb-3">
  <Col className="mx-auto">
    <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Train
Status</Form.Label>
    <Form.Select
      name="TrainStatus"
      placeholder='Train Status'
      style={{fontFamily: "Onest"}}
```

```
      value={updatedTrainData.TrainStatus}
      onChange={handleChange}
    >
      <option value="Resheduled">Resheduled</option>
      <option value="cancelled">Cancelled</option>
    </Form.Select>
  </Col>
</Row>
</div>
        </div>
        <Row className="mb-3">
          <Col className="mx-auto" style={{margin: "34px"}}>
            <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px' }}>Back</Button>{' '}
            <Button variant="primary" type="submit" style={{ width: '150px',
backgroundColor: "#00284d" }}>Update Train</Button>
          </Col>
        </Row>
      </Form>
      </Card.Body>
      </Card>
    </Container>
  );
};

export default UpdateTrainShedule;
```

AddTrainTicketBooking.jsx

```
import React, { useState, useContext, useEffect } from 'react';
import axios from 'axios';
import { AuthContext } from '../AuthContext';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Form, Button, Card, Container, Row, Col } from 'react-bootstrap';
import { useHistory } from 'react-router-dom';
import { IsValidNIC, IsValidContactNumber, IsValidTicketClass } from
'../Validations';

const AddTrainTicketBooking = () => {
  // Context for authentication
  const { userId } = useContext(AuthContext);

  // State variables
```

```
  const [trainData, setTrainData] = useState([]);
  const [inputsDisabled, setInputsDisabled] = useState(true);
  const [formData, setFormData] = useState({
    MainPassengerName: '',
    UserID: userId,
    ReservationDate: '',
    BookingDate: '',
    DepartureStation: '',
    DestinationStation: '',
    TrainName: '',
    TotalPassengers: "",
    TicketClass: '',
    Email: '',
    NIC: '',
    ContactNumber: '',
    TotalPrice: ''
  });

  const history = useHistory();

  // Function to calculate total price based on ticket class and number of
passengers
  const calculateTotalPrice = () => {
    const ticketClass = formData.TicketClass;
    const totalPassengers = parseInt(formData.TotalPassengers);

    let TotalPrice = 0;
    if (ticketClass === 'First Class') {
      TotalPrice = totalPassengers * formData.ticketPrice1;
      console.log("Total price"+ TotalPrice);
    } else if (ticketClass === 'Second Class') {
      TotalPrice = totalPassengers * formData.ticketPrice2;
    } else if (ticketClass === 'Third Class') {
      TotalPrice = totalPassengers * formData.ticketPrice3;
    }

    setFormData({
      ...formData,
      TotalPrice: TotalPrice.toString()
    });
  };

  // Function to get the current date in the required format
  const getCurrentDate = () => {
    const today = new Date();
```

```
    const year = today.getFullYear();
    let month = today.getMonth() + 1;
    let day = today.getDate();

    if (month < 10) {
      month = `0${month}`;
    }

    if (day < 10) {
      day = `0${day}`;
    }

    return `${year}-${month}-${day}`;
  };

  // Function to handle form input changes
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });

    if (name === 'TrainName') {
      fetchTicketPrice(value);
      setInputsDisabled(false);
    }
  };

  // Function to fetch ticket price for a selected train
  const fetchTicketPrice = (id) => {
    console.log(`Fetching ticket price for train ID: ${id}`);
    axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trains/gettrain/${id}`)
      .then(response => {
        const ticketPrice1 = response.data.FirstClassTicketPrice;
        const ticketPrice2 = response.data.SecondClassTicketPrice;
        const ticketPrice3 = response.data.ThirdClassTicketPrice;

        setFormData(prevState => ({
          ...prevState,
          ticketPrice1,
          ticketPrice2,
          ticketPrice3
        }));
```

```
      })
      .catch(error => {
        console.error('Error fetching ticket price:', error);
      });
  };

// Fetches the list of trains and sets ticket prices based on total passengers
and ticket class
  useEffect(() => {
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/trains/getalltrains')
      .then(response => {
        setTrainData(response.data);
        calculateTotalPrice();
      })
      .catch(error => {
        console.error('Error fetching train data:', error);
      });
  }, [formData.TotalPassengers, formData.TicketClass]);

  console.log('Form Data:', formData);
  console.log('UserID:', formData.UserID);
  console.log('TrainName:', formData.TrainName);

  // Handles form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    // Handle NIC Validation
    if (!IsValidNIC(formData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    // Handle Contact Number Validation
    if (!IsValidContactNumber(formData.ContactNumber)) {
      toast.error('Invalid Contact Number format.');
      return;
    }

    // Handle Ticket Class Validation
    if (!IsValidTicketClass(formData.TicketClass)) {
      toast.error('Invalid ticket Class format.');
      return;
    }
```

```
    setFormData({
      ...formData,
      BookingDate: getCurrentDate()
    });

    axios.post('http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/createticketbooking', formData)
      .then(response => {
        console.log('Reservation created:', response.data);
        toast.success("Reservation Added");
        setTimeout(() => {
        history.push('/travelagentdashboard');
        }, 2000)
      })
      .catch(error => {
        // toast.error('Reservation date must be within 30 days from the current
date.', error);
        toast.error(error.response.data.Message);
      });
  };

  return (
    <Container className="text-center mt-5" style={{width: "1200px", paddingLeft:
"250px", marginBottom: "25px"}}>
      <ToastContainer position="top-center" autoClose={1000} hideProgressBar />
      <div className="container">
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
            <Card.Body>
              <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Create Your Train Booking</Card.Title>
        <Form onSubmit={handleSubmit}>
          <div className="row">
            {/* Left Column */}
            <div className="col-md-6" style={{textAlign: "left"}}>
            <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Main
Passenger Name</Form.Label>
              <Form.Control
                type="text"
                name="MainPassengerName"
                placeholder='main Passenger Name'
                style={{fontFamily: "Onest"}}
                value={formData.MainPassengerName}
                onChange={handleChange}
                required
```

```
              /><br />

<Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email</Form.Label>
              <Form.Control
                type="Email"
                name="Email"
                value={formData.Email}
                style={{fontFamily: "Onest"}}
                onChange={handleChange}
                placeholder="Email"
                required
              /><br />

              <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Contact Number</Form.Label>
              <Form.Control
                type="text"
                name="ContactNumber"
                style={{fontFamily: "Onest"}}
                value={formData.ContactNumber}
                onChange={handleChange}
                placeholder="Contact Number"
                required
              /><br />

              <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Reservation Date</Form.Label>
              <Form.Control
                type="date"
                name="ReservationDate"
                placeholder='Reservation Date'
                style={{fontFamily: "Onest"}}
                value={formData.ReservationDate}
                onChange={handleChange}
                required
              /><br />
              <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Train Name</Form.Label>
              <Form.Group controlId="TrainName">
          <Form.Select
            name="TrainName"
            value={formData.TrainName}
            onChange={handleChange}
            style={{fontFamily: "Onest"}}
```

```
            required
          >
            <option value="">Select Train Name</option>
            {trainData.map(train => (
              <option key={train._id} value={train._id}>
                {train.TrainName}
              </option>
            ))}
          </Form.Select>

        </Form.Group>
              <br />
            </div>
            <div className="col-md-6" style={{textAlign: "left"}}>
            <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Total Passengers</Form.Label>
              <Form.Control
                type="number"
                name="TotalPassengers"
                value={formData.TotalPassengers}
                style={{fontFamily: "Onest"}}
                onChange={handleChange}
                placeholder='Total Passengers'
                required
              /><br />
              <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Ticket Class</Form.Label>
              <Form.Group controlId="TicketClass">
  <Form.Control
    as="select"
    name="TicketClass"
    value={formData.TicketClass}
    style={{fontFamily: "Onest"}}
    disabled={inputsDisabled}
    onChange={handleChange}
    required
  >
    <option value="" disabled>Select Ticket Class</option>
    <option value="First Class">First Class</option>
    <option value="Second Class">Second Class</option>
    <option value="Third Class">Third Class</option>
  </Form.Control>
  </Form.Group>
              <br />
```

```
<Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Destination
Station</Form.Label>
              <Form.Control
                type="text"
                name="DestinationStation"
                value={formData.DestinationStation}
                style={{fontFamily: "Onest"}}
                onChange={handleChange}
                placeholder='Destination Station'
                required
              /><br />

<Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Deaprture
Station</Form.Label>
              <Form.Control
                type="text"
                name="DepartureStation"
                value={formData.DepartureStation}
                style={{fontFamily: "Onest"}}
                onChange={handleChange}
                placeholder='Departure Station'
                required
              /><br />

<Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>NIC</Form.Label>
              <Form.Control
                type="text"
                name="NIC"
                value={formData.NIC}
                style={{fontFamily: "Onest"}}
                onChange={handleChange}
                placeholder='NIC'
                required
              /><br />
            </div>
          </div>
          <br/>
          <Row>
            <Col>
          <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat",
marginLeft: "200px"}}>Total Price</Form.Label>
          </Col>
          <Col>
              <Form.Control
                type="text"
```

```
                name="TotalPrice"
                style={{marginRight: "100px", width: "100px", fontFamily:
"Onest"}}
                value={"Rs." + formData.TotalPrice + ".00"}
                onChange={handleChange}
                placeholder='Total Price'
                required
                disabled
              />
              </Col>
              </Row>


          <div className="text-center" style={{margin: "34px"}}>
            <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px' }}>Back</Button>{' '}
            <Button type="submit" variant="primary" style={{ width: '150px',
backgroundColor: "#00284d" }}>Submit</Button>
          </div>
        </Form>
        </Card.Body>
        </Card>
      </div>
    </Container>
  );
};

export default AddTrainTicketBooking;
```

GetAllTrainTicketBookings.jsx

```
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from '../AuthContext';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Link } from 'react-router-dom';
import Cookies from 'js-cookie';
import { Table, Button, Card, Container, Form } from 'react-bootstrap';

const GetAllTrainTicketBookings = () => {
  // Context for authentication
  const { userId, setUser } = useContext(AuthContext);
```

```
  // State variables
  const [reservations, setReservations] = useState([]);
  const [cancellationLoading, setCancellationLoading] = useState(false);
  const [currentPage, setCurrentPage] = useState(1);
  const [searchQuery, setSearchQuery] = useState('');
  const itemsPerPage = 5;
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;

  // Handle pagination
  const handlePagination = (pageNumber) => {
    setCurrentPage(pageNumber);
  };

  // Handle search input change
  const handleSearchChange = (e) => {
    setSearchQuery(e.target.value);
  };

  // Handle cancellation of a booking
  const handleCancel = (id, ReservationDate, BookingDate) => {
    const ReservationDateObj = new Date(ReservationDate);
    const BookingDateObj = new Date(BookingDate);

    const differenceInMilliseconds = ReservationDateObj - BookingDateObj;

    const differenceInDays = differenceInMilliseconds / (1000 * 60 * 60 * 24);

    if (differenceInDays < 5) {
      toast.error("Cancellation is allowed only if reservation date is within 5
days of booking date.");
      return;
    }

    setCancellationLoading(true);
    // Function to cancel booking data
    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/cancelticketbooking/${id}`)
      .then(response => {
        console.log(`Booking with ID ${id} has been cancelled.`);
        setReservations(prevReservations => prevReservations.filter(res => res.ID
!== id));
        toast.success("Reservation has been cancelled.");
        window.location.href="#"
      })
```

```
      .catch(error => {
        toast.error("Cancellation is allowed only if reservation date is within 5
days of booking date.");
      })
      .finally(() => {
        setCancellationLoading(false);
      });
  };

  // Fetch reservations and handle user authentication
  useEffect(() => {
    const saveduserID = Cookies.get('userID');

    if (!userId && saveduserID) {
      setUser(saveduserID);
    }
    if (userId) {
      // Function to fetch booking data
      axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/getallticketbookings`)
        .then(response => {
          setReservations(response.data);
          localStorage.setItem('reservations', JSON.stringify(response.data));
        })
        .catch(error => {
          console.error('Error fetching reservations:', error);
        });
    } else {
      const savedReservations = JSON.parse(localStorage.getItem('reservations'));
      if (savedReservations) {
        setReservations(savedReservations);
      }
    }
  }, [userId, setUser]);

  // Filter reservations based on search query and calculate current items for
pagination
  const filteredReservations = reservations.filter(reservation =>
    reservation.MainPassengerName.toLowerCase().includes(searchQuery.toLowerCase(
)) ||
    reservation.TrainName.toLowerCase().includes(searchQuery.toLowerCase()) ||
    reservation.NIC.toLowerCase().includes(searchQuery.toLowerCase())
  );
```

```
  const currentItems = filteredReservations.slice(indexOfFirstItem,
indexOfLastItem);

  return (
    <Container className="text-center mt-5" style={{ height: "600px",
paddingLeft: "250px", maxWidth: "1200px" }}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
        <Card.Body>
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>All Train Bookings</Card.Title>
          <Form.Control
            type="text"
            placeholder="Search by Train Name, Main Passenger Name, or NIC"
            value={searchQuery}
            onChange={handleSearchChange}
            style={{ marginBottom: '10px' }}
          />
          <Table striped bordered hover style={{ marginTop: '20px', width: "75%"
}} className="mx-auto">
            <thead>
              <tr style={{ fontSize: "17px", fontFamily: "Montserrat" }}>
                <th>Train Name</th>
                <th>Main Passenger Name</th>
                <th>Traveler NIC</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {currentItems.map(reservation => (
                <tr key={reservation.ID} style={{ fontFamily: "Onest" }}>
                  <td>{reservation.TrainName}</td>
                  <td>{reservation.MainPassengerName}</td>
                  <td>{reservation.NIC}</td>
                  <td>
                    <Button
                      variant="warning"
                      as={Link}
                      to={`/getticketbooking/${reservation.BookingID}`}
                      style={{ color: 'white', marginRight: '5px',
textDecoration: 'none' }}
                    >
                      <i className="fas fa-eye"></i>
                    </Button>
```

```
                    <Button
                      variant="link"
                      as={Link}
                      to={`/updateticketbooking/${reservation.BookingID}`}
                      style={{ background: 'green', color: 'white',
textDecoration: 'none', marginRight: '5px' }}
                    >
                      <i className="fas fa-edit"></i>
                    </Button>

                    <Button
                      variant="danger"
                      onClick={() => handleCancel(reservation.BookingID,
reservation.ReservationDate, reservation.BookingDate)}
                    >
                      <i className="fas fa-trash-alt"></i>
                    </Button>
                  </td>
                </tr>
              ))}
            </tbody>
          </Table>
          <div className="pagination" style={{ textAlign: 'Right', margin:
"20px", marginLeft: "40%" }}>
            <span
              onClick={() => currentPage > 1 && handlePagination(currentPage -
1)}
              className={currentPage === 1 ? 'disabled' : ''}
              style={{ margin: "0 5px", cursor: "pointer" }}
            >
              &#8249;
            </span>

            {Array.from({ length: Math.ceil(filteredReservations.length /
itemsPerPage) }).map((_, index) => (
              <span
                key={index}
                onClick={() => handlePagination(index + 1)}
                className={currentPage === index + 1 ? 'active' : ''}
                style={{ margin: "0 5px", cursor: "pointer" }}
              >
                {index + 1}
              </span>
            ))}
```

```
            <span
              onClick={() => currentPage < Math.ceil(filteredReservations.length
/ itemsPerPage) && handlePagination(currentPage + 1)}
              className={currentPage === Math.ceil(filteredReservations.length /
itemsPerPage) ? 'disabled' : ''}
              style={{ margin: "0 5px", cursor: "pointer" }}
            >
              &#8250;
            </span>
          </div>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default GetAllTrainTicketBookings;
```

GetMyTrainTicketBookings.jsx

```
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from '../AuthContext';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Link } from 'react-router-dom';
import Cookies from 'js-cookie';
import { Table, Button, Card, Container, Form } from 'react-bootstrap';

const GetMyTrainTicketBooking = () => {
  // Context for authentication
  const { userId, setUser } = useContext(AuthContext);

  // State variables
  const [reservations, setReservations] = useState([]);
  const [cancellationLoading, setCancellationLoading] = useState(false);
  const [currentPage, setCurrentPage] = useState(1);
  const [searchQuery, setSearchQuery] = useState('');
  const itemsPerPage = 5;
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;

  // Handle pagination
```

```
  const handlePagination = (pageNumber) => {
    setCurrentPage(pageNumber);
  };

  // Handle search input change
  const handleSearchChange = (e) => {
    setSearchQuery(e.target.value);
  };

  // Handle cancellation of a booking
  const handleCancel = (id, ReservationDate, BookingDate) => {
    const ReservationDateObj = new Date(ReservationDate);
    const BookingDateObj = new Date(BookingDate);

    const differenceInMilliseconds = ReservationDateObj - BookingDateObj;

    const differenceInDays = differenceInMilliseconds / (1000 * 60 * 60 * 24);

    if (differenceInDays < 5) {
      toast.error("Cancellation is allowed only if reservation date is within 5
days of booking date.");
      return;
    }

    setCancellationLoading(true);
    // Function to cancel booking data.
    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/cancelticketbooking/${id}`)
      .then(response => {
        console.log(`Booking with ID ${id} has been cancelled.`);
        setReservations(prevReservations => prevReservations.filter(res => res.ID
!== id));
        toast.success("Reservation has been cancelled.");
        window.location.href="#"
      })
      .catch(error => {
        toast.error("Cancellation is allowed only if reservation date is within 5
days of booking date.");
      })
      .finally(() => {
        setCancellationLoading(false);
      });
  };

  // Fetch reservations and handle user authentication
```

```
  useEffect(() => {
    const saveduserID = Cookies.get('userID');

    if (!userId && saveduserID) {
      setUser(saveduserID);
    }
    if (userId) {
      // Function to fetch booking data.
      axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/getallticketbookings/${userId}`)
        .then(response => {
          setReservations(response.data);
          localStorage.setItem('reservations', JSON.stringify(response.data));
        })
        .catch(error => {
          console.error('Error fetching reservations:', error);
        });
    } else {
      const savedReservations = JSON.parse(localStorage.getItem('reservations'));
      if (savedReservations) {
        setReservations(savedReservations);
      }
    }
  }, [userId, setUser]);

  // Filter reservations based on search query and calculate current items for
pagination
  const filteredReservations = reservations.filter(reservation =>
    reservation.MainPassengerName.toLowerCase().includes(searchQuery.toLowerCase(
)) ||
    reservation.TrainName.toLowerCase().includes(searchQuery.toLowerCase()) ||
    reservation.NIC.toLowerCase().includes(searchQuery.toLowerCase())
  );

  const currentItems = filteredReservations.slice(indexOfFirstItem,
indexOfLastItem);

  return (
    <Container className="text-center mt-5" style={{ height: "570px",
paddingLeft: "250px", maxWidth: "1200px" }}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
        <Card.Body>
```

```
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Your All Train Bookings</Card.Title>
          <Form.Control
            type="text"
            placeholder="Search by Train Name, Main Passenger Name, or NIC"
            value={searchQuery}
            onChange={handleSearchChange}
            style={{ marginBottom: '10px' }}
          />
          <Table striped bordered hover style={{ marginTop: '20px', width:"75%"
}} className="mx-auto">
            <thead>
              <tr style={{ fontSize: "17px", fontFamily: "Montserrat" }}>
                <th>Train Name</th>
                <th>Main Passenger Name</th>
                <th>Traveler NIC</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {currentItems.map(reservation => (
                <tr key={reservation.ID} style={{fontFamily: "Onest"}}>
                  <td>{reservation.TrainName}</td>
                  <td>{reservation.MainPassengerName}</td>
                  <td>{reservation.NIC}</td>
                  <td>
                    <Button
                      variant="warning"
                      as={Link}
                      to={`/getticketbooking/${reservation.BookingID}`}
                      style={{ color: 'white', marginRight: '5px',
textDecoration: 'none' }}
                    >
                      <i className="fas fa-eye"></i>
                    </Button>

                    <Button
                      variant="link"
                      as={Link}
                      to={`/updateticketbooking/${reservation.BookingID}`}
                      style={{ background: 'green', color: 'white',
textDecoration: 'none' }}
                    >
                      <i className="fas fa-edit"></i>
                    </Button>
```

```
                    <Button
                      variant="danger"
                      onClick={() => handleCancel(reservation.BookingID,
reservation.ReservationDate, reservation.BookingDate)}
                      disabled={cancellationLoading}
                      style={{ marginLeft: '5px' }}
                    >
                      <i className="fas fa-trash-alt"></i>
                    </Button>
                  </td>
                </tr>
              ))}
            </tbody>
          </Table>
          <div className="pagination" style={{ textAlign: 'Right', margin:
"20px", marginLeft: "40%" }}>
            <span
              onClick={() => currentPage > 1 && handlePagination(currentPage -
1)}
              className={currentPage === 1 ? 'disabled' : ''}
              style={{margin: "0 5px", cursor: "pointer"}}
            >
              &#8249;
            </span>

            {Array.from({ length: Math.ceil(filteredReservations.length /
itemsPerPage) }).map((_, index) => (
              <span
                key={index}
                onClick={() => handlePagination(index + 1)}
                className={currentPage === index + 1 ? 'active' : ''}
                style={{margin: "0 5px", cursor: "pointer"}}
              >
                {index + 1}
              </span>
            ))}

            <span
              onClick={() => currentPage < Math.ceil(filteredReservations.length
/ itemsPerPage) && handlePagination(currentPage + 1)}
              className={currentPage === Math.ceil(filteredReservations.length /
itemsPerPage) ? 'disabled' : ''}
              style={{margin: "0 5px", cursor: "pointer"}}
            >
```

```
              &#8250;
            </span>
          </div>
        </Card.Body>
      </Card>
    </Container>
  );
};

export default GetMyTrainTicketBooking;
```

GetTrainTicketBooking.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Card, Table, Row, Col, Button, Container } from 'react-bootstrap';

const GetTrainTicketBooking = () => {
  // Fetching the reservation details based on BookingID from the API
  const [reservation, setReservation] = useState(null);
  const { BookingID } = useParams();

  useEffect(() => {
    if (BookingID) {
      axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/getticketbooking/${BookingID}`)
        .then(response => {
          setReservation(response.data);
          console.log(response.data);
        })
        .catch(error => {
          console.error('Error fetching reservation:', error);
        });
    }
  }, [BookingID]);

  // Rendering a loading message while waiting for reservation data
  if (!reservation) {
    return <div>Loading...</div>;
  }
```

```
  return (
    <Container className="my-5 text-center" style={{ paddingLeft: "250px" }}>
  <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
  <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
    <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
    <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One", fontSize:
"34px" }}>View Train Booking</Card.Title>
    <Card.Body>
      <Table striped bordered hover>
        <tbody>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Main Passenger Name</strong></td>
            <td style={{fontFamily:
"Onest"}}>{reservation.MainPassengerName}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Booking Date</strong></td>
            <td style={{fontFamily:
"Onest"}}>{reservation.FormattedBookingDate}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Train Name</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.TrainName}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Reservation Date</strong></td>
            <td style={{fontFamily:
"Onest"}}>{reservation.FormattedReservationDate}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Departure Station</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.DepartureStation}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Destination Station</strong></td>
            <td style={{fontFamily:
"Onest"}}>{reservation.DestinationStation}</td>
          </tr>
```

```
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Total Passengers</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.TotalPassengers}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Ticket Class</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.TicketClass}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Email</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.Email}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Contact Number</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.ContactNumber}</td>
          </tr>
          <tr>
            <td className="text" style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Total Price (Rs.)</strong></td>
            <td style={{fontFamily: "Onest"}}>{reservation.TotalPrice}</td>
          </tr>
        </tbody>
      </Table>
    </Card.Body>
    <Row className="justify-content-center" style={{ margin: "25px" }}>
              <Col xs="auto">
              <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px', backgroundColor: "#00284d" }}>Back</Button>{' '}
              </Col>
            </Row>
  </Card>
  </Container>
  );
};

export default GetTrainTicketBooking;
```

UpdateTrainTicketBooking.jsx

```
import React, { useState, useContext, useEffect } from 'react';
```

```
import axios from 'axios';
import { Container, Form, Button, Row, Col, Card } from 'react-bootstrap';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { AuthContext } from '../AuthContext';
import { useHistory } from 'react-router-dom';
import { useParams } from 'react-router-dom';
import { IsValidNIC, IsValidContactNumber, IsValidTicketClass } from
'../Validations';

const UpdateTrainTicketBooking = () => {
  // Getting the BookingID from URL parameters
  const { BookingID } = useParams();

  // Getting user ID from AuthContext
  const { userId } = useContext(AuthContext);

  // State for storing train data
  const [trainData, setTrainData] = useState([]);

  // State for disabling form inputs
  const [inputsDisabled, setInputsDisabled] = useState(true);

  // State for storing updated reservation data
  const [updatedReservationData, setUpdatedReservationData] = useState({
    MainPassengerName: '',
    UserID: userId,
    ReservationDate: '',
    BookingDate: '',
    DepartureStation: '',
    DestinationStation: '',
    TrainName: '',
    TotalPassengers: "",
    TicketClass: '',
    Email: '',
    ContactNumber: '',
    TotalPrice: ''
  });

  // Getting history object for navigation
  const history = useHistory();


  // Function to calculate total price based on ticket class and total passengers
  const calculateTotalPrice = () => {
```

```
  const ticketClass = updatedReservationData.TicketClass;
  const totalPassengers = parseInt(updatedReservationData.TotalPassengers);

  let TotalPrice = 0;
  if (ticketClass === 'First Class') {
    TotalPrice = totalPassengers * updatedReservationData.uticketPrice1;
  } else if (ticketClass === 'Second Class') {
    TotalPrice = totalPassengers * updatedReservationData.uticketPrice2;
  } else if (ticketClass === 'Third Class') {
    TotalPrice = totalPassengers * updatedReservationData.uticketPrice3;
  }

  console.log(`Ticket Class: ${ticketClass}`);
  console.log(`Total Passengers: ${totalPassengers}`);
  console.log(`Total Price: ${TotalPrice}`);

  setUpdatedReservationData({
    ...updatedReservationData,
    TotalPrice: TotalPrice.toString()
  });
};

// Function to handle form input changes
const handleChange = (e) => {
  const { name, value } = e.target;

  if (name === 'TrainName') {
    console.log('Selected Train Name:', value);
    setUpdatedReservationData({
      ...updatedReservationData,
      [name]: value,
    });

    fetchTicketPrice(value);
    setInputsDisabled(false);
  } else {
    console.log('Other Field Name:', name);
    console.log('Other Field Value:', value);
    setUpdatedReservationData({
      ...updatedReservationData,
      [name]: value,
    });

    if (value === 'TicketClass' || value === 'TotalPassengers') {
      calculateTotalPrice();
```

```
      }
    }
  };

  // Function to fetch ticket price based on selected train
  const fetchTicketPrice = (id) => {
    axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trains/gettrain/${id}`)
      .then(response => {
        const uticketPrice1 = response.data.FirstClassTicketPrice;
        const uticketPrice2 = response.data.SecondClassTicketPrice;
        const uticketPrice3 = response.data.ThirdClassTicketPrice;

        setUpdatedReservationData(prevState => ({
          ...prevState,
          uticketPrice1,
          uticketPrice2,
          uticketPrice3
        }));
      })
      .catch(error => {
        console.error('Error fetching ticket price:', error);
      });
  };

  // Function to handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    // Handle NIC Validation
    if (!IsValidNIC(updatedReservationData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    // Handle Contact Number Validation
    if (!IsValidContactNumber(updatedReservationData.ContactNumber)) {
      toast.error('Invalid Contact Number format.');
      return;
    }

    // Handle Ticket Class Validation
    if (!IsValidTicketClass(updatedReservationData.TicketClass)) {
      toast.error('Invalid ticket Class format.');
      return;
```

```
    }

    // Handle update booking data
    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/updateticketbooking/${BookingID}`,
updatedReservationData)
      .then(response => {
        console.log('Reservation updated:', response.data);
        toast.success('Reservation updated successfully!');
        setTimeout(() => {
        history.push('/travelagentdashboard');
        }, 2000)
      })
      .catch(error => {
        console.error('Error:', error);
        toast.error('Reservation can only be updated if reservation date is more
than 5 days after booking date.');
      });
  };

  // Effect to fetch and set train data
  useEffect(() => {
    if (BookingID) {
      axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/getticketbooking/${BookingID}`)
        .then(response => {
          setUpdatedReservationData(response.data);
        })
        .catch(error => {
          console.error('Error fetching reservation data:', error);
        });
    }
  }, [BookingID]);

  // Effect to fetch all shedule trains
  useEffect(() => {
    let isMounted = true;

    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/trains/getallsheduledtrains')
      .then(response => {
        if (isMounted) {
          setTrainData(response.data);
          calculateTotalPrice();
        }
```

```
      })
      .catch(error => {
        console.error('Error fetching train data:', error);
      });

    return () => {
      isMounted = false;
    };
  }, [updatedReservationData.TotalPassengers,
updatedReservationData.TicketClass]);

  return (
    <Container className="my-5 text-center" style={{width: "75%", paddingLeft:
"250px"}}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
  <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
            <Card.Body>
              <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Update Trian Booking</Card.Title>


  <Form onSubmit={handleSubmit}>
  <div className="row">
  <div className="col-md-6" style={{textAlign: "left"}}>
    <Form.Group style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Main
Passenger Name</Form.Label>
      <Form.Control
        type="text"
        name="TravelerName"
        value={updatedReservationData.MainPassengerName}
        onChange={handleChange}
        placeholder="Main Passenger Name"
        style={{fontFamily: "Onest"}}
        required
      />
    </Form.Group>
    <Form.Group style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email:</Form.Label>
      <Form.Control
        type="Email"
        name="Email"
        value={updatedReservationData.Email}
        style={{fontFamily: "Onest"}}
```

```
        onChange={handleChange}
        placeholder="Email"
        required
      />
    </Form.Group>
    <Form.Group style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Contact
Number</Form.Label>
      <Form.Control
        type="tel"
        name="Phone"
        value={updatedReservationData.ContactNumber}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        placeholder="Contact Number"
        required
      />
    </Form.Group>

    <Form.Group controlId="TrainName" style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Train
Name</Form.Label>
      <Form.Select
        name="TrainName"
        style={{fontFamily: "Onest"}}
        value={updatedReservationData.TrainName}
        onChange={handleChange}
        required
      >
        <option value="">Select Train Name</option>
        {trainData.map(train => (
  <option key={train._id} value={train._id}>
    {train.TrainName}
  </option>
))}
      </Form.Select>
    </Form.Group>
    <Form.Group style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Reservation Date</Form.Label>
      <Form.Control
  type="date"
  name="ReservationDate"
  placeholder='Reservation Date'
  style={{fontFamily: "Onest"}}
```

```
  value={updatedReservationData.ReservationDate}
  onChange={handleChange}
  required
/>
    </Form.Group>

    </div>
    <div className="col-md-6" style={{textAlign: "left"}}>

    <Form.Group style={{textAlign:"left", margin: "25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
      <Form.Control
        type="text"
        name="NIC"
        style={{fontFamily: "Onest"}}
        value={updatedReservationData.NIC}
        onChange={handleChange}
        placeholder="NIC"
        required
      />
    </Form.Group>

    <Form.Group controlId="TotalPassengers" style={{textAlign:"left", margin:
"25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Total
Passengers</Form.Label>
      <Form.Control
        type="number"
        name="TotalPassengers"
        style={{fontFamily: "Onest"}}
        value={updatedReservationData.TotalPassengers}
        onChange={handleChange}
        placeholder="Total Passengers"
        required
      />
    </Form.Group>

    <Form.Group controlId="TicketClass" style={{textAlign:"left", margin:
"25px"}}>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Ticket
Class:</Form.Label>
      <Form.Select
        name="TicketClass"
        style={{fontFamily: "Onest"}}
```

```
          value={updatedReservationData.TicketClass}
          disabled={inputsDisabled}
          onChange={handleChange}
          required
        >
          <option value="">Select Ticket Class</option>
          <option value="First Class">First Class</option>
          <option value="Second Class">Second Class</option>
          <option value="Third Class">Third Class</option>
        </Form.Select>
      </Form.Group>

      <Form.Group style={{textAlign:"left", margin: "25px"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Departure
Station</Form.Label>
        <Form.Control
          type="text"
          name="DepartureStation"
          style={{fontFamily: "Onest"}}
          value={updatedReservationData.DepartureStation}
          onChange={handleChange}
          placeholder="Departure Station"
          required
        />
      </Form.Group>

      <Form.Group style={{textAlign:"left", margin: "25px"}}>
        <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Destination Station</Form.Label>
        <Form.Control
          type="text"
          style={{fontFamily: "Onest"}}
          name="DestinationStation"
          value={updatedReservationData.DestinationStation}
          onChange={handleChange}
          placeholder="Destination Station"
          required
        />
      </Form.Group>

          </div>
          </div>
          <Form.Group style={{textAlign:"left", margin: "25px"}}>
          <Row>
              <Col>
```

```
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat", marginLeft:
"200px"}}>Total Price</Form.Label>
      </Col>
      <Col>
      <Form.Control
        type="text"
        style={{fontFamily: "Onest", marginRight: "100px", width: "100px"}}
        name="TotalPrice"
        value={"Rs: "+updatedReservationData.TotalPrice+ ".00"}
        onChange={handleChange}
        placeholder="Total Price"
        required
        disabled
      />
      </Col>
      </Row>
    </Form.Group>
        <Row className="mb-3" style={{margin: "25px"}}>
          <Col md={0} className="mx-auto">
            <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px' }}>Back</Button>{' '}
            <Button variant="primary" type="submit" style={{ width: '150px',
backgroundColor: "#00284d" }}>Update Booking</Button>
          </Col>
        </Row>
  </Form>


  </Card.Body>
  </Card>

</Container>
  );
};

export default UpdateTrainTicketBooking;
```

Traveler.jsx

```
import React, { useState, useContext, useEffect } from 'react';
import axios from 'axios';
import { AuthContext } from '../../AuthContext';
import { Link } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
```

```
import 'react-toastify/dist/ReactToastify.css';
import { Container, Table, Button, Card } from 'react-bootstrap';

const TravellerUser = () => {
  // Retrieve UserID from the AuthContext
  const { UserID } = useContext(AuthContext);

  // State to hold the list of travelers
  const [traveler, setTravellers] = useState([]);

  // Pagination state
  const [currentPage, setCurrentPage] = useState(1);
  const itemsPerPage = 5;
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;
  const currentItems = traveler.slice(indexOfFirstItem, indexOfLastItem);

  // Function to handle pagination
  const handlePagination = (pageNumber) => {
    setCurrentPage(pageNumber);
  };

  // Fetch traveler data on component mount
  useEffect(() => {
    axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/getallusers')
      .then(response => {
        setTravellers(response.data);
      })
      .catch(error => {
        console.error('Error fetching traveler:', error);
      });
  }, []);

  // Function to handle status change (Activate/Deactivate)
  const handleStatusChange = (UserID, currentStatus) => {
    const newStatus = currentStatus === 'Active' ? 'active' : 'Active';

    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/users/updateuserstatus/${UserID}`, { UserStatus: newStatus
})
      .then(response => {
        if (response.status === 200) {
          setTravellers(traveler.map(traveler =>
```

```
            traveler.UserID === UserID ? { ...traveler, UserStatus: newStatus } :
traveler
          ));
          toast.success('Status updated successfully');
        } else {
          console.error('Error updating status:', response.data);
          toast.error('Failed to update status');
        }
      })
      .catch(error => {
        console.error('Error updating status:', error);
        toast.error('Failed to update status');
      });
  };

  return (
    <Container className="my-5 text-center" style={{ height: "570px",
paddingLeft: "250px", maxWidth: "900px" }}>
  <ToastContainer position="top-center" autoClose={1000} hideProgressBar />
  <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
    <Card.Body>
      <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Traveler Status</Card.Title>
      <Table striped bordered hover responsive>
        <thead>
          <tr style={{ fontSize: "17px", fontFamily: "Montserrat" }}>
            <th>Username</th>
            <th>NIC</th>
            <th>User Type</th>
            <th>User Status</th>
            <th>Actions</th>
          </tr>
    </thead>
    <tbody>
      {traveler.map(traveler => (
        <tr key={traveler.UserID} style={{fontFamily: "Onest"}}>
          <td>{traveler.UserName}</td>
          <td>{traveler.NIC}</td>
          <td>{traveler.UserType}</td>
          <td>{traveler.UserStatus}</td>
          <td>
          <Link to={`/viewtraveller/${traveler.UserID}`} className="mr-2">
              <Button variant="warning" style={{marginRight: "5px"}}><i
className="fas fa-eye"></i></Button>
```

```
            </Link>
            <Button
              variant="primary"
              onClick={() => handleStatusChange(traveler.UserID,
traveler.UserStatus)}
            >
              {traveler.UserStatus === 'Active' ? 'Deactivate' : 'Activate'}
            </Button>
          </td>
        </tr>
      ))}
    </tbody>
  </Table>
  <div className="pagination" style={{ textAlign: 'Right', margin: "20px",
marginLeft: "47%" }}>
  <span
    onClick={() => currentPage > 1 && handlePagination(currentPage - 1)}
    className={currentPage === 1 ? 'disabled' : ''}
    style={{margin: "0 5px", cursor: "pointer"}}
  >
    &#8249;  {/* Left arrow */}
  </span>

  {Array.from({ length: Math.ceil(traveler.length / itemsPerPage) }).map((_,
index) => (
    <span
      key={index}
      onClick={() => handlePagination(index + 1)}
      className={currentPage === index + 1 ? 'active' : ''}
      style={{margin: "0 5px", cursor: "pointer"}}
    >
      {index + 1}
    </span>
  ))}

  <span
    onClick={() => currentPage < Math.ceil(traveler.length / itemsPerPage) &&
handlePagination(currentPage + 1)}
    className={currentPage === Math.ceil(traveler.length / itemsPerPage) ?
'disabled' : ''}
    style={{margin: "0 5px", cursor: "pointer"}}
  >
    &#8250;  {/* Right arrow */}
  </span>
</div>
```

```
  </Card.Body>
  </Card>
</Container>
  );
};

export default TravellerUser;
```

AddTraveler.jsx

```
import React, { useState } from 'react';
import axios from 'axios';
import { Form, Button, Container, Row, Col, Card } from 'react-bootstrap';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { useHistory } from 'react-router-dom';
import { IsValidEmail, IsValidPassword, IsValidNIC, IsValidContactNumber } from
'../../Validations';

const AddTraveller = () => {

  // Initialize state with form data
  const [formData, setFormData] = useState({
    NIC: '',
    UserName: '',
    FirstName: '',
    LastName: '',
    Email: '',
    Gender: '',
    Password: '',
    RePassword: '',
    ContactNumber: '',
    UserType: "Traveler"
  });

  // Get the history object for programmatic navigation
  const history = useHistory();

  // Handle input changes
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value
```

```
    });
  };

  // Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    // Check if passwords match
    if (formData.Password !== formData.RePassword) {
      toast.error('Passwords do not match.');
      return;
    }

    // Validate email
    if (!IsValidEmail(formData.Email)) {
      toast.error('Invalid email format.');
      return;
    }

    // Validate NIC
    if (!IsValidNIC(formData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    // Validate password
    if (!IsValidPassword(formData.Password)) {
      toast.error('Invalid Password format.');
      return;
    }

    // Validate contact number
    if (!IsValidContactNumber(formData.ContactNumber)) {
      toast.error('Invalid contact number format.');
      return;
    }

    // Send POST request to create a new traveler
    axios.post('http://pasinduperera-001-site1.atempurl.com/api/users/signup',
formData, {
      headers: {
        'Content-Type': 'application/json',
      },
    })
    .then(response => {
```

```
      console.log('Success:', response.data);
      toast.success('User Created successfully!');
      setTimeout(() => {
      history.push('/travelagentdashboard');
      }, 2000)
    })
    .catch(error => {
      toast.error('user already exist');
    });
  };

  return (
    <Container className="my-5 text-center" style={{width: "1200px", paddingLeft:
"250px"}}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
  <Row className="justify-content-center">
    <Col>
    <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
            <Card.Body>
              <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Create New Traveler</Card.Title>
      <Form onSubmit={handleSubmit}>
      <div className="row">
      <div className="col-md-6" style={{textAlign: "left"}}>
        <Form.Group controlId="NIC" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
          <Form.Control
            type="text"
            name="NIC"
            value={formData.NIC}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="NIC"
            required
          />
          <br/>
        </Form.Group>
        <Form.Group controlId="UserName" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Name</Form.Label>
          <Form.Control
            type="text"
```

```
            name="UserName"
            value={formData.UserName}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="User Name"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="FirstName" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>First
Name</Form.Label>
          <Form.Control
            type="text"
            name="FirstName"
            value={formData.FirstName}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="First Name"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="LastName" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Last
Name</Form.Label>
          <Form.Control
            type="text"
            name="LastName"
            value={formData.LastName}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Last Name"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="userType" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Type</Form.Label>
          <Form.Control
            type="text"
```

```
            name="userType"
            value={formData.UserType}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Traveler"
            required
            disabled
          />
        </Form.Group>
        <br/>
        </div>
        <div className="col-md-6" style={{textAlign: "left"}}>
        <Form.Group controlId="Email" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email</Form.Label>
          <Form.Control
            type="Email"
            name="Email"
            value={formData.Email}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Email"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="Gender" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Gender</Form.Label>
  <Form.Select
    name="Gender"
    value={formData.Gender}
    style={{fontFamily: "Onest"}}
    onChange={handleChange}
    required
  >
    <option value="">Select Gender</option>
    <option value="Male">Male</option>
    <option value="Female">Female</option>
  </Form.Select>
</Form.Group>
        <br/>
```

```
        <Form.Group controlId="ContactNumber" style={{fontSize: "17px",
fontFamily: "Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Contact
Number</Form.Label>
          <Form.Control
            type="text"
            name="ContactNumber"
            value={formData.ContactNumber}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Contact Number"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="Password" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Password</Form.Label>
          <Form.Control
            type="password"
            name="Password"
            value={formData.Password}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Password"
            required
          />
        </Form.Group>
        <br/>
        <Form.Group controlId="RePassword" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
        <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Re enter
Password</Form.Label>
          <Form.Control
            type="password"
            name="RePassword"
            value={formData.RePassword}
            style={{fontFamily: "Onest"}}
            onChange={handleChange}
            placeholder="Re Password"
            required
          />
        </Form.Group>
        <br/>
```

```
        </div>
        </div>
        <Row className="justify-content-center" style={{margin: "25px"}}>
              <Col xs="auto">
              <Button variant="secondary" onClick={() =>
window.history.back('/travelerlist')} style={{ width: '150px' }}>Back</Button>{'
'}
            <Button type="submit" variant="primary" style={{ width: '150px',
backgroundColor: "#00284d" }}>Create Traveler</Button>
              </Col>
            </Row>
      </Form>
      </Card.Body>
  </Card>
    </Col>
  </Row>
</Container>
  );
};

export default AddTraveller;
```

GetAllTravelers.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Table, Button, Container, Card, Form } from 'react-bootstrap';

const GetAllTravelers = () => {

  // State for storing the list of users
  const [users, setUsers] = useState([]);

  // State for storing the search query
  const [searchQuery, setSearchQuery] = useState('');

  // Function to fetch users from the API
  useEffect(() => {
    const fetchUsers = async () => {
      try {
```

```
        const response = await axios.get('http://pasinduperera-001-
site1.atempurl.com/api/users/getallusers');
        setUsers(response.data);
      } catch (error) {
        console.error('Error fetching users:', error);
      }
    };

    fetchUsers();
  }, []);

  // Function to handle user deletion
  const handleDeleteUser = async (userID) => {
    try {
      await axios.delete(`http://pasinduperera-001-
site1.atempurl.com/api/users/deleteuser/${userID}`);

      const updatedUsers = users.filter(user => user.UserID !== userID);
      setUsers(updatedUsers);
      toast.success('Travel user deleted successfully!');
      setTimeout(() => {
        window.location.href="#";
      }, 2000)
    } catch (error) {
      console.error('Error deleting user:', error);
    }
  };

  // Function to filter users based on search query
  const filteredUsers = users.filter(user => {
    const { NIC, UserName, Email } = user;
    const query = searchQuery.toLowerCase();
    return NIC.toLowerCase().includes(query) ||
           UserName.toLowerCase().includes(query) ||
           Email.toLowerCase().includes(query);
  });

  return (
    <Container className="my-5 text-center" style={{ height: "600px",
paddingLeft: "250px", maxWidth: "900px" }}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
        <Card.Body>
```

```
          <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px" }}>Travelers</Card.Title>
          <Form.Control
            type="text"
            placeholder="Search by NIC, Username, or Email"
            value={searchQuery}
            onChange={(e) => setSearchQuery(e.target.value)}
            style={{ marginBottom: '10px' }}
          />
          <Table striped bordered hover responsive>
            <thead>
              <tr style={{fontSize: "17px", fontFamily: "Montserrat"}}>
                <th>NIC</th>
                <th>Username</th>
                <th>Email</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {filteredUsers.map(user => (
                <tr key={user.ID} style={{fontFamily: "Onest"}}>
                  <td>{user.NIC}</td>
                  <td>{user.UserName}</td>
                  <td>{user.Email}</td>
                  <td>
                    <Link to={`/viewtraveller/${user.UserID}`} className="mr-2">
                      <Button variant="warning" style={{marginRight: "5px"}}><i
className="fas fa-eye"></i></Button>
                    </Link>
                    <Link to={`/updatetraveller/${user.UserID}`} className="mr-
2">
                      <Button variant="success" style={{marginRight: "5px"}}><i
className="fas fa-edit"></i></Button>
                    </Link>
                    <Button variant="danger" onClick={() =>
handleDeleteUser(user.UserID)} style={{marginRight: "5px"}}><i className="fas fa-
trash-alt"></i></Button>
                  </td>
                </tr>
              ))}
            </tbody>
          </Table>
        </Card.Body>
      </Card>
    </Container>
```

```
  );
};

export default GetAllTravelers;
```

GetTraveler.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Container, Table, Row, Col, Button, Card } from 'react-bootstrap';
import imageprofileavatar from '../../Assests/profileavatar.png'

const GetTraveler = () => {

  // Retrieve UserID from URL parameters
  const { UserID } = useParams();

  // State for storing the user details
  const [user, setUser] = useState(null);

  // Function to fetch user details from the API
  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/users/getuser/${UserID}`);
        setUser(response.data);
      } catch (error) {
        console.error('Error fetching user:', error);
      }
    };

    if (UserID) {
      fetchData();
    }
  }, [UserID]);

  // If no UserID is found, display a message
  if (!UserID) {
    return <div>No user ID found</div>;
  }
```

```
  // If user details are still loading, display a loading message
  if (!user) {
    return <div>Loading...</div>;
  }

  return (
    <Container className="my-5 text-center" style={{ paddingLeft: "250px" }}>
  <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
  <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none',
borderRadius: '15px', boxShadow: '0px 0px 15px rgba(0, 0, 0, 0.1)' }}>
    <Card.Body>
      <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One",
fontSize: "34px", color: "#00284d" }}>View Traveler</Card.Title>
      <div className="text-center mb-4">
      <img src={imageprofileavatar} alt="Profile" style={{ width: '250px',
height: '170px', borderRadius: '50%' }} />
      </div>
      <Table striped bordered responsive style={{ fontFamily: "Onest" }}>
    <tbody>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>User
NIC</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.NIC}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Username</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserName}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>First
Name</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.FirstName}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Last
Name</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.LastName}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Email</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.Email}</td>
      </tr>
```

```
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Gender</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.Gender}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>Phone
Number</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.ContactNumber}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily: "Montserrat"}}><strong>User
Type</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserType}</td>
      </tr>
      <tr>
        <td style={{fontSize: "17px", fontFamily:
"Montserrat"}}><strong>Status</strong></td>
        <td style={{fontFamily: "Onest"}}>{user.UserStatus}</td>
      </tr>
    </tbody>
    </Table>
      <Row className="justify-content-center" style={{ margin: "25px" }}>
        <Col xs="auto">
          <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px', backgroundColor: '#00284d', fontFamily: "Montserrat"
}}>Back</Button>{' '}
        </Col>
      </Row>
    </Card.Body>
  </Card>
</Container>
  );
};

export default GetTraveler;
```

UpdateTraveler.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useParams, useHistory } from 'react-router-dom';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
```

```
import { Container, Form, Button, Row, Col, Card } from 'react-bootstrap';
import { IsValidEmail, IsValidPassword, IsValidNIC, IsValidContactNumber } from
'../../Validations';
import imageprofileavatar from '../../Assests/profileavatar.png'

const UpdateTraveller = () => {

  // Retrieve UserID from URL parameters
  const { UserID } = useParams();

  // Access to the history object to navigate
  const history = useHistory();

  // State for storing the user data
  const [userData, setUserData] = useState({
    UserName: '',
    FirstName: '',
    LastName: '',
    Email: '',
    Gender: '',
    ContactNumber: '',
    UserType: '',
  });

  // Fetch user data from the API
  useEffect(() => {
    axios.get(`http://pasinduperera-001-
site1.atempurl.com/api/users/getuser/${UserID}`)
      .then(response => {
        setUserData(response.data);
      })
      .catch(error => {
        console.error('Error fetching user data:', error);
      });
  }, [UserID]);

  // Handle changes in form fields
  const handleChange = (e) => {
    const { name, value } = e.target;
    setUserData({
      ...userData,
      [name]: value,
    });
  };
```

```
  // Handle form submission
  const handleSubmit = (e) => {
    e.preventDefault();

    if (!IsValidEmail(userData.Email)) {
      toast.error('Invalid email format.');
      return;
    }

    if (!IsValidNIC(userData.NIC)) {
      toast.error('Invalid NIC format.');
      return;
    }

    if (!IsValidContactNumber(userData.ContactNumber)) {
      toast.error('Invalid contact number format.');
      return;
    }

    axios.put(`http://pasinduperera-001-
site1.atempurl.com/api/users/updateuser/${UserID}`, userData)
      .then(response => {
        console.log('User updated:', response.data);
        toast.success('Travel user updated successfully!');
        setTimeout(() => {
        history.push(`/viewtraveller/${UserID}`);
        window.location.href = `/travelerlist`;
        }, 2000)
      })
      .catch(error => {
        console.error('Error updating user:', error);
      });
  };

  return (
    <Container className="my-5 text-center" style={{width: "1200px", paddingLeft:
"250px"}}>
      <ToastContainer position="top-center" autoClose={3000} hideProgressBar />
      <Card style={{ background: 'rgba(255, 255, 255, 0.7)', border: 'none' }}>
            <Card.Body>
  <Card.Title style={{ margin: "25px", fontFamily: "Dela Gothic One", fontSize:
"34px" }}>Update Traveler</Card.Title>
  <div className="text-center mb-4">
  <img src={imageprofileavatar} alt="Profile" style={{ width: '250px', height:
'170px', borderRadius: '50%' }} />
```

```
          </div>
  <Form onSubmit={handleSubmit}>
  <div className="row">
  <div className="col-md-6" style={{textAlign: "left"}}>
  <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>NIC</Form.Label>
      <Form.Control
        type="text"
        id="NIC"
        name="NIC"
        placeholder='NIC'
        value={userData.NIC}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
        disabled
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Name</Form.Label>
      <Form.Control
        type="text"
        id="UserName"
        name="UserName"
        placeholder='User Name'
        value={userData.UserName}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>First
Name</Form.Label>
      <Form.Control
        type="text"
        id="FirstName"
        name="FirstName"
        placeholder='First Name'
        value={userData.FirstName}
        style={{fontFamily: "Onest"}}
```

```
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Last
Name</Form.Label>
      <Form.Control
        type="text"
        id="LastName"
        name="LastName"
        placeholder='Last Name'
        value={userData.LastName}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    </div>
    <div className="col-md-6" style={{textAlign: "left"}}>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Email</Form.Label>
      <Form.Control
        type="Email"
        id="Email"
        name="Email"
        placeholder='Email'
        value={userData.Email}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group controlId="Gender" style={{fontSize: "17px", fontFamily:
"Montserrat"}}>
  <Form.Label style={{fontSize: "17px", fontFamily:
"Montserrat"}}>Gender</Form.Label>
  <Form.Select
    name="Gender"
    value={userData.Gender}
    style={{fontFamily: "Onest"}}
```

```
    onChange={handleChange}
    required
  >
    <option value="">Select Gender</option>
    <option value="Male">Male</option>
    <option value="Female">Female</option>
    <option value="Other">Other</option>
  </Form.Select>
</Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>Phone
Number</Form.Label>
      <Form.Control
        type="text"
        id="ContactNumber"
        name="ContactNumber"
        placeholder='Contact Number'
        value={userData.ContactNumber}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <br/>
    <Form.Group>
      <Form.Label style={{fontSize: "17px", fontFamily: "Montserrat"}}>User
Status</Form.Label>
      <Form.Control
        type="text"
        id="UserStatus"
        name="UserStatus"
        placeholder='User Status'
        value={userData.UserStatus}
        style={{fontFamily: "Onest"}}
        onChange={handleChange}
        required
        disabled
      />
    </Form.Group>
    <br/>
    </div>
            </div>
            <Row className="justify-content-center" style={{margin: "25px"}}>
              <Col xs="auto">
```

```
              <Button variant="secondary" onClick={() => window.history.back()}
style={{ width: '150px' }}>Back</Button>{' '}
            <Button type="submit" variant="primary" style={{ width: '150px',
backgroundColor: "#00284d" }}>Update</Button>
              </Col>
            </Row>
  </Form>
  </Card.Body>
  </Card>
</Container>
  );
};

export default UpdateTraveller;
```

ARouter.jsx

```
import React from 'react';
import { BrowserRouter as Router, Switch, Route, useLocation } from 'react-
router-dom';
import { UserProvider } from './AuthContext';

 // Navbar file path
import NavBar from '../Components/Dashboard/NavBar';

// Footer file path
import Footer from './Dashboard/Footer';

// SignIn file path
import SignIn from './User_Management/SignIn';
// Signup file path
import Signup from './User_Management/SignUp';
// UserProfile file path
import UserProfile from '../Components/User_Management/UserProfile';
// UpdateUserProfile file path
import UpdateUserProfile from './User_Management/UpdateUserProfile';

// AddTrainShedule file path
import AddTrainShedule from './Train_Management/AddTrainShedule';
// UpdateTrainShedule file path
import UpdateTrainShedule from './Train_Management/UpdateTrainShedule';
// GetTrainShedule file path
import GetTrainShedule from './Train_Management/GetTrainShedule';
// GetAllTrainShedules file path
```

```
import GetAllTrainShedules from './Train_Management/GetAllTrainShedules';

// AddTrainTicketBooking file path
import AddTrainTicketBooking from
'./Ticket_Booking_Management/AddTrainTicketBooking';
// UpdateTrainTicketBooking file path
import UpdateTrainTicketBooking from
'./Ticket_Booking_Management/UpdateTrainTicketBooking';
// GetTrainTicketBooking file path
import GetTrainTicketBooking from
'./Ticket_Booking_Management/GetTrainTicketBooking';
// GetMyTrainTicketBooking file path
import GetMyTrainTicketBooking from
'./Ticket_Booking_Management/GetMyTrainTicketBooking';
// GetAllTrainTicketBookings file path
import GetAllTrainTicketBookings from
'./Ticket_Booking_Management/GetAllTrainTicketBookings';

// Traveler file path
import Traveler from './Traveler_Management/BackOfficeUser/Traveler';

// AddTraveler file path
import AddTraveler from './Traveler_Management/TravelAgent/AddTraveler';
// UpdateTraveler file path
import UpdateTraveler from './Traveler_Management/TravelAgent/UpdateTraveler';
// GetTraveler file path
import GetTraveler from './Traveler_Management/TravelAgent/GetTraveler';
// GetAllTravelers file path
import GetAllTravelers from './Traveler_Management/TravelAgent/GetAllTravelers';

// BackOfficeUserDashboard file path
import BackOfficeUserDashboard from './Dashboard/BackOfficeUserDashboard';
// TravelAgentDashboard file path
import TravelAgentDashboard from './Dashboard/TravelAgentDashboard';

// SideBar file path
import SideBar from './Dashboard/SideBar';

const ARouter = () => {
  // Get the current location
  const location = useLocation();

  // Define routes to hide Navbar, Sidebar, and Footer components
  const hideNavbarRoutes = ['/', '/signup'];
  const hideSideBarRoutes = ['/', '/signup'];
```

```
  const hideFooterRoutes = ['/', '/signup'];

  // Determine if Navbar, Sidebar, and Footer should be hidden
  const shouldHideNavbar = hideNavbarRoutes.includes(location.pathname);
  const shouldHideSideBar = hideSideBarRoutes.includes(location.pathname);
  const shouldHideFooter = hideFooterRoutes.includes(location.pathname);

  return (
    <Router>
      <UserProvider>
        {/* Render NavBar unless shouldHideNavbar is true */}
        {!shouldHideNavbar && <NavBar />}

        {/* Render SideBar unless shouldHideSideBar is true */}
        {!shouldHideSideBar && <SideBar />}
        <Switch>
          <Route path="/signup" exact component={Signup} />
          <Route path="/" exact component={SignIn} />
          <Route path="/updateuserprofile/:UserID" component={UpdateUserProfile}
/>
          <Route path="/userprofile/:userId" component={UserProfile} />

          <Route path="/addtrainshedule" component={AddTrainShedule}/>
          <Route path="/updatetrainshedule/:TrainID"
component={UpdateTrainShedule} />
          <Route path="/viewtrainshedule/:TrainID" component={GetTrainShedule} />
          <Route path="/trainshedulelist" exact component={GetAllTrainShedules}
/>

          <Route path="/addticketbooking" component={AddTrainTicketBooking}/>
          <Route path="/updateticketbooking/:BookingID"
component={UpdateTrainTicketBooking} />
          <Route path="/getticketbooking/:BookingID"
component={GetTrainTicketBooking} />
          <Route path="/getmyticketbookings"
component={GetMyTrainTicketBooking}/>
          <Route path="/getallticketbookings"
component={GetAllTrainTicketBookings}/>

          <Route path="/travelerstatus" component={Traveler} />

          <Route path="/addtraveler" component={AddTraveler} />
          <Route path="/updatetraveller/:UserID" component={UpdateTraveler} />
          <Route path="/viewtraveller/:UserID" component={GetTraveler} />
          <Route path="/travelerlist" component={GetAllTravelers} />
```

```
          <Route path="/backofficeuserdashboard"
component={BackOfficeUserDashboard} />
          <Route path="/travelagentdashboard" component={TravelAgentDashboard} />

          <Route path="/footer" component={Footer} />
        </Switch>
        {/* Render Footer unless shouldHideFooter is true */}
        {!shouldHideFooter && <Footer />}
      </UserProvider>
    </Router>
  );
};

export default ARouter;
```

AuthContext.jsx

```
import React, { createContext, useState } from 'react';
import Cookies from 'js-cookie';

// Create a context to hold authentication information
const AuthContext = createContext();

// Create a provider component to manage authentication state
const UserProvider = ({ children }) => {
  // Initialize state for userId using a function that reads from cookies
  const [userId, setUserId] = useState(() => {
    const savedUserId = Cookies.get('userId');
    return savedUserId || null;
  });

  // Initialize state for UserType using a function that reads from cookies
  const [UserType, setUserType] = useState(() => {
    const savedUserType = Cookies.get('UserType');
    return savedUserType || null;
  });

  // Function to set user information and store it in cookies
  const setUser = (id, type) => {
    console.log(`Setting UserType to: ${type}`);
    setUserId(id);
    setUserType(type);
    Cookies.set('userId', id, { expires: 7 });
```

```
    Cookies.set('UserType', type, { expires: 7 });
    console.log(`Set UserType to: ${type}`);
  };

  // Provide the authentication context to its children
  return (
    <AuthContext.Provider value={{ userId, UserType, setUser }}>
      {children}
    </AuthContext.Provider>
  );
};

export { UserProvider, AuthContext };
```

Validations.jsx

```
export const IsValidEmail = (email) => {
    // Checks if the provided string is a valid email address.
    const emailPattern = /^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/;
    return emailPattern.test(email);
};

export const IsValidPassword = (password) => {
    // Checks if the provided string meets password requirements.
    const passwordPattern = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-
z\d@$!%*?&]{8,}$/;
    return passwordPattern.test(password);
};

export const IsValidNIC = (nic) => {
    // Checks if the provided string is a valid National Identity Card (NIC)
number. Assumes a 12-digit format.
    const nicPattern = /^\d{12}$/;
    return nicPattern.test(nic);
};

export const IsValidContactNumber = (contactNumber) => {
    // Checks if the provided string is a valid contact number. Assumes a 10-
digit format.
    const contactNumberPattern = /^\d{10}$/;
    return contactNumberPattern.test(contactNumber);
};

export const IsValidTicketClass = (ticketClass) => {
```

```
    // Checks if the provided string is a valid ticket class. It validates
against an array of valid classes: "First Class", "Second Class", and "Third
Class".
    return ["First Class", "Second Class", "Third Class"].includes(ticketClass);
};

export const IsValidTimeRange = (departureTime, arrivalTime) => {
    // Checks if the departure time is earlier than the arrival time. Assumes
time format in HH:mm.
    const departureTimeArray = departureTime.split(':').map(Number);
    const arrivalTimeArray = arrivalTime.split(':').map(Number);

    return (
        departureTimeArray[0] < arrivalTimeArray[0] ||
        (departureTimeArray[0] === arrivalTimeArray[0] && departureTimeArray[1] <
arrivalTimeArray[1])
    );
};

export const IsValidTrainNumber = (trainNumber) => {
    // Checks if the provided string is a valid train number. Assumes a format
where it starts with a capital letter followed by four digits.
    const trainIDPattern = /^[A-Z]\d{4}$/;
    return trainIDPattern.test(trainNumber);
};
```

NavBar.jsx

```
import React, { useContext } from 'react';
import { Link } from 'react-router-dom';
import Navbar from 'react-bootstrap/Navbar';
import Nav from 'react-bootstrap/Nav';
import 'bootstrap/dist/css/bootstrap.min.css';
import { AuthContext } from '../AuthContext';
import { FaUser } from 'react-icons/fa';

const NavBar = () => {
  // Get userId and UserType from AuthContext
  const { userId, UserType } = useContext(AuthContext);

  return (
    <Navbar style={{ backgroundColor: '#b3daff', marginBottom: "25px", height:
"75px", fontFamily: "Dela Gothic One", fontSize: "23px", border: "1px solid
#00284d", color: "#00284d" }} className="justify-content-between">
```

```
      {/* Section for Travel Agents */}
      <Nav className="m3-auto">
        {UserType === 'TravelAgent' && (
          <>
            <Nav.Link as={Link} to="/addticketbooking" style={{padding: "34px",
color: "#00284d", marginLeft: "278px"}}>Add Train Booking</Nav.Link>
            <Nav.Link as={Link} to="/addtraveler" style={{padding: "34px", color:
"#00284d"}}>Add Traveler</Nav.Link>
          </>
        )}
      </Nav>

      {/* Section for Back Office Users */}
      <Nav className="m3-auto">
        {UserType === 'BackOfficeUser' && (
          <>
            <Nav.Link as={Link} to="/addtrainshedule" style={{padding: "34px",
color: "#00284d", marginRight: "740px"}}>Add Train</Nav.Link>
          </>
        )}
      </Nav>

      {/* Section for User Profile */}
      <Nav>
        <Nav.Link as={Link} to={`/userprofile/${userId}`} style={{padding:
"34px", color: "#00284d"}}>
          <FaUser style={{ marginRight: '8px' }} />
        </Nav.Link>
      </Nav>
    </Navbar>
  );
};

export default NavBar;
```

SideBar.jsx

```
import React, { useContext } from 'react';
import { Link, useHistory } from 'react-router-dom';
import Nav from 'react-bootstrap/Nav';
import 'bootstrap/dist/css/bootstrap.min.css';
import { AuthContext } from '../AuthContext';
import Cookies from 'js-cookie';
import imglogo from '../Assests/logo.png'
```

```
const SideBar = () => {
  const { setUser, UserType } = useContext(AuthContext);
  const history = useHistory();

  const handleLogout = () => {
    // Remove userId cookie and set user to null
    Cookies.remove('userId');
    setUser(null);
    history.push('/');
    window.location.href = "/"; // Redirect to home
  };

  return (
    <div style={{ width: '250px', height: '100%', backgroundColor: '#00284d',
position: 'fixed', top: '0', left: '0', fontFamily: "Dela Gothic One", fontSize:
"23px" }}>
      {/* Logo */}
      <div style={{ padding: '15px', textAlign: 'center' }}>
        <img
          src={imglogo}
          alt="Logo"
          style={{ width: '200px', height: '200px', borderRadius: '50%' }}
        />
      </div>

      <Nav className="flex-column">
        {UserType === 'TravelAgent' && (
          <>
            <Nav.Link as={Link} to="/travelagentdashboard" style={{ color:
"white", padding: "15px", textAlign: "center" }}>Home</Nav.Link>
            <Nav.Link as={Link} to="/getmyticketbookings" style={{ color:
"white", padding: "15px", textAlign: "center" }}>My Bookings</Nav.Link>
            <Nav.Link as={Link} to="/getallticketbookings" style={{ color:
"white", padding: "15px", textAlign: "center" }}>Booking Management</Nav.Link>
            <Nav.Link as={Link} to="/travelerlist" style={{ color: "white",
padding: "15px", textAlign: "center" }}>Traveler Management</Nav.Link>
          </>
        )}
        {UserType === 'BackOfficeUser' && (
          <>
            <Nav.Link as={Link} to="/backofficeuserdashboard" style={{ color:
"white", padding: "15px", textAlign: "center" }}>Home</Nav.Link>
            <Nav.Link as={Link} to="/trainshedulelist" style={{ color: "white",
padding: "15px", textAlign: "center" }}>Train Management</Nav.Link>
```

```
            <Nav.Link as={Link} to="/travelerstatus" style={{ color: "white",
padding: "15px", textAlign: "center" }}>Traveler Status</Nav.Link>
          </>
        )}
        <Nav.Link onClick={handleLogout} style={{ color: "white", padding:
"15px", textAlign: "center", marginTop: "27px" }}>SignOut</Nav.Link>
      </Nav>
    </div>
  );
};

export default SideBar;
```

Footer.jsx

```
import React from 'react';

const Footer = () => {
  return (
    <footer style={{ backgroundColor: '#00284d', color: "white", padding: '17px',
textAlign: 'center', fontFamily: "Dela Gothic One", fontSize: "17px" }}>
      <p>&copy; EAD Group Assignment 2023</p>
    </footer>
  );
};

export default Footer;
```

7.3 Mobile Application

ManActivity.java

```
package com.example.trainbooking_mobileapp;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import
com.example.trainbooking_mobileapp.ReservationManagement.CreateReservationActivit
y;
import
com.example.trainbooking_mobileapp.ReservationManagement.ReservationDetailsActivi
ty;
import com.example.trainbooking_mobileapp.TrainManagement.TrainDetailsActivity;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

public class MainActivity extends AppCompatActivity {

    private Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Set up the toolbar
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        setTitle("Home");

        getSupportActionBar().setDisplayHomeAsUpEnabled(false);

        // Get the user ID from the intent
        String userID = getIntent().getStringExtra("userID");

        // Initialize buttons
```

```
        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button2 = findViewById(R.id.button2);
        ImageButton Button3 = findViewById(R.id.button3);
        ImageButton Button4 = findViewById(R.id.button4);
        ImageButton Button5 = findViewById(R.id.button5);
        ImageButton Button6 = findViewById(R.id.button6);

        // Set up click listeners for the buttons
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
CreateReservationActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
ReservationDetailsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
TrainDetailsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
```

```
            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_sign_out:
                // Perform sign out logic here
                // For example, you can start the sign-in activity
                Intent intent = new Intent(MainActivity.this,
SignInActivity.class);
                startActivity(intent);
                finish(); // Finish the current activity after signing out
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

```
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".MainActivity">

    <include layout="@layout/top_bar" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Welcome Traveler"
            android:layout_gravity="center"
            android:layout_marginTop="78dp"
            android:gravity="center"
            android:textSize="43sp"
            android:textColor="@color/white"
            android:layout_marginBottom="16dp"/>

        <TextView
            android:id="@+id/appDescription"
            android:layout_width="wrap_content"
            android:layout_height="170dp"
            android:text="Welcome to our reservation app! Easily manage your
train bookings and explore a wide range of travel options. With our user-friendly
interface, planning your journeys has never been more convenient. Get started
today and experience seamless travel planning like never before."
            android:textSize="17sp"
            android:textColor="#FFFFFF"
            android:background="#8B00284D"
            android:layout_centerHorizontal="true"
```

```
            android:layout_marginTop="100dp"
            android:gravity="center" />


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center">

            <ImageButton
                android:id="@+id/button3"
                android:layout_width="160dp"
                android:layout_height="75dp"
                android:layout_margin="16dp"
                android:background="@drawable/button_border"
                android:elevation="8dp"
                android:src="@drawable/ic_baseline_library_books_24"
                android:tint="#FFFFFF"
                app:layout_behavior="com.google.android.material.behavior.HideBot
tomViewOnScrollBehavior" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="My Reservations"
                android:textSize="16dp"
                android:textColor="#000000" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center">

            <ImageButton
                android:id="@+id/button4"
                android:layout_width="160dp"
                android:layout_height="75dp"
                android:layout_margin="16dp"
                android:background="@drawable/button_border"
                android:elevation="8dp"
                android:src="@drawable/ic_baseline_train_24"
                android:tint="#FFFFFF"
```

```
                app:layout_behavior="com.google.android.material.behavior.HideBot
tomViewOnScrollBehavior" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Train Shedules"
                android:textSize="16dp"
                android:textColor="#000000" />
        </LinearLayout>



        <ScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginBottom="27dp"
            tools:context=".AboutUsActivity">

            <ImageButton
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="bottom|end"
                android:layout_margin="47dp"
                android:background="@drawable/rounded_button_bg"
                android:elevation="8dp"
                android:src="@drawable/ic_baseline_add_circle_outline_24"
                android:tint="#FFFFFF"
                app:layout_behavior="com.google.android.material.behavior.HideBot
tomViewOnScrollBehavior" />

        </ScrollView>
    </LinearLayout>

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
```

```
        android:background="#00284d"
        android:elevation="8dp">

        <ImageButton
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_home_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button5"
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>
</RelativeLayout>
```

User.java

```
package com.example.trainbooking_mobileapp.UserManagement;

import java.io.Serializable;

public class User implements Serializable {

    // Member variables
```

```
    private String ID;
    private String NIC;
    private String FirstName;
    private String LastName;
    private String UserName;
    private String Email;
    private String Gender;
    private String ContactNumber;
    private String UserType;
    private String Password;
    private String RePassword;
    private String UserStatus;

    // Constructor to initialize user data
    public User(String ID, String FirstName, String LastName, String UserName,
String Email, String NIC, String Gender, String ContactNumber, String userType,
String Password, String RePassword, String userStatus) {
        this.ID = ID;
        this.FirstName = FirstName;
        this.LastName = LastName;
        this.UserName = UserName;
        this.Email = Email;
        this.NIC = NIC;
        this.Gender = Gender;
        this.Password = Password;
        this.RePassword = RePassword;
        this.ContactNumber = ContactNumber;
        this.UserType = "Traveller";
        this.UserStatus = "Active";
    }

    // Getter and setter methods for user properties

    public String getID() {
        return ID;
    }

    public void setID(String UserID) {
        this.ID = UserID;
    }

    public String getNIC() {
        return NIC;
    }
```

```java
public void setNIC(String NIC) {
    this.NIC = NIC;
}

public String getGender() {
    return Gender;
}

public void setGender(String Gender) {
    this.Gender = Gender;
}

public String getContactNumber() {
    return ContactNumber;
}

public void setContactNumber(String ContactNumber) {
    this.ContactNumber = ContactNumber;
}

public String getFirstName() {
    return FirstName;
}

public void setFirstName(String firstName) {
    this.FirstName = firstName;
}

public String getLastName() {
    return LastName;
}

public void setLastName(String lastName) {
    this.LastName = lastName;
}

public String getUserName() {
    return UserName;
}

public void setUserName(String username) {
    this.UserName = username;
}

public String getEmail() {
```

```
        return Email;
    }

    public void setEmail(String email) {
        this.Email = email;
    }

    public String getPassword() {
        return Password;
    }

    public void setPassword(String password) {
        this.Password = password;
    }

    public String getRePassword() {
        return RePassword;
    }

    public void setRePassword(String reenteredPassword) {
        this.RePassword = reenteredPassword;
    }

    public String getUserType() {
        return UserType;
    }

    public void setUserType(String userType) {
        this.UserType = userType;
    }

    public String getUserStatus() {
        return UserStatus;
    }

    public void setUserStatus(String userStatus) {
        this.UserStatus = userStatus;
    }
}
```

UserApiClient.java

```
package com.example.trainbooking_mobileapp.UserManagement;
```

```
import android.os.AsyncTask;
import android.util.Log;
import org.json.JSONObject;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;

public class UserApiClient {

    // API endpoint URL
    private static final String API_URL = "http://pasinduperera-001-
site1.atempurl.com/api/users/getallusers";

    // Interface for receiving user data
    public interface OnUserDataReceivedListener {
        void onUserDataReceived(List<User> userList);
        void onError(String errorMessage);
    }

    // Method for updating user data in API
    public static void updateUserInAPI(final User user, final
OnUserUpdatedListener listener) {
        AsyncTask<Void, Void, String> task = new AsyncTask<Void, Void, String>()
{
            @Override
            protected String doInBackground(Void... voids) {
                try {
                    // Create a URL object for the update user endpoint
                    URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/users/updateuser/" + user.getID());
                    Log.d("UpdateUserActivity", "Url: " + url);
                    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                    connection.setRequestMethod("PUT");
                    connection.setRequestProperty("Content-Type",
"application/json");

                    // Create a JSON object with updated user data
                    JSONObject requestBody = new JSONObject();
                    requestBody.put("FirstName", user.getFirstName());
                    requestBody.put("LastName", user.getLastName());
                    requestBody.put("UserName", user.getUserName());
                    requestBody.put("email", user.getEmail());
                    requestBody.put("Gender", user.getGender());
```

```
                requestBody.put("ContactNumber", user.getContactNumber());

                // Enable output for sending data
                connection.setDoOutput(true);
                OutputStream os = connection.getOutputStream();
                os.write(requestBody.toString().getBytes("UTF-8"));
                os.close();

                int responseCode = connection.getResponseCode();

                if (responseCode == HttpURLConnection.HTTP_OK) {
                    return "Success";
                } else {
                    return "Error";
                }
            } catch (Exception e) {
                e.printStackTrace();
                return "Error: " + e.getMessage();
            }
        }

        @Override
        protected void onPostExecute(String response) {
            if (response != null && response.equals("Success")) {
                listener.onUserUpdated();
            } else {
                listener.onError("Error updating user");
            }
        }
    };

    task.execute();
}

// Interface for user update callbacks
public interface OnUserUpdatedListener {
    void onUserUpdated();
    void onError(String errorMessage);
}
}
```

SignInActivity.java

```
package com.example.trainbooking_mobileapp.UserManagement;
```

```java
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class SignInActivity extends AppCompatActivity {

    // EditText fields for NIC and password
    private EditText nicEditText, passwordEditText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_in);

        // Initialize EditText fields
        nicEditText = findViewById(R.id.nicEditText);
        passwordEditText = findViewById(R.id.passwordEditText);

        // Sign In Button Click Listener
        Button signInButton = findViewById(R.id.signInButton);
        signInButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get NIC and password from EditText fields
                String nic = nicEditText.getText().toString();
                String password = passwordEditText.getText().toString();
```

```
                // Validate NIC and password
                if (nic.isEmpty() || password.isEmpty()) {
                    Toast.makeText(SignInActivity.this, "NIC and password cannot
be empty", Toast.LENGTH_SHORT).show();
                    return;
                }

                if (!isValidNIC(nic)) {
                    Toast.makeText(SignInActivity.this, "Invalid NIC format.",
Toast.LENGTH_SHORT).show();
                    return;
                }

                if (!isValidPassword(password)) {
                    Toast.makeText(SignInActivity.this, "Invalid password
format.", Toast.LENGTH_SHORT).show();
                    return;
                }

                // Create JSON object for NIC and password
                JSONObject json = new JSONObject();
                try {
                    json.put("NIC", nic);
                    json.put("Password", password);
                } catch (JSONException e) {
                    e.printStackTrace();
                }

                // Execute SignInTask
                new SignInTask().execute(json.toString());
            }
        });

        // Sign Up TextView Click Listener
        TextView txtSignUp = findViewById(R.id.txtSignUp);
        txtSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(SignInActivity.this,
SignUpActivity.class);
                startActivity(intent);
            }
        });
    }
```

```
    // Validate NIC format
    private boolean isValidNIC(String nic) {
        String nicPattern = "^\\d{12}$";
        return nic.matches(nicPattern);
    }

    // Validate password format
    private boolean isValidPassword(String password) {
        String passwordPattern = "^(?=.*[a-z])(?=.*[A-
Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]{8,}$";
        return password.matches(passwordPattern);
    }

    // AsyncTask to handle sign in process
    private class SignInTask extends AsyncTask<String, Void, String> {
        // Background task to perform sign in request
        @Override
        protected String doInBackground(String... params) {
            MediaType JSON = MediaType.parse("application/json; charset=utf-8");
            OkHttpClient client = new OkHttpClient();
            String url = "http://pasinduperera-001-
site1.atempurl.com/api/users/signin";

            RequestBody body = RequestBody.create(JSON, params[0]);
            Request request = new Request.Builder()
                    .url(url)
                    .post(body)
                    .build();

            try {
                Response response = client.newCall(request).execute();
                String responseData = response.body().string();
                Log.d("SignInTask", "Response Data: " + responseData);

                if (response.isSuccessful()) {
                    JSONObject json = new JSONObject(responseData);
                    String userID = json.getString("UserID");
                    return userID;
                } else {
                    return null;
                }
            } catch (IOException | JSONException e) {
                e.printStackTrace();
                return null;
            }
```

```
        }

        // Handle response after sign in request
        @Override
        protected void onPostExecute(String userID) {
            if (userID != null) {
                Toast.makeText(SignInActivity.this, "Sign-in successful!",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(SignInActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            } else {
                Toast.makeText(SignInActivity.this, "Incorrect NIC or password or
User account is deactivated", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

activity_sign_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="32dp"
    android:gravity="center"
    android:background="@drawable/background"
    tools:context=".UserManagement.SignInActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="47dp"
        android:text="SIGN IN"
        android:textColor="@color/white"
        android:textSize="30sp" />

    <EditText
```

```
        android:id="@+id/nicEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:background="@drawable/rounded_border"
        android:padding="8dp"
        android:textColor="@color/white"
        android:hint="NIC"
        android:textColorHint="@color/white"
        android:layout_margin="16dp" />

    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:background="@drawable/rounded_border"
        android:padding="8dp"
        android:textColor="@color/white"
        android:hint="Password"
        android:textColorHint="@color/white"
        android:layout_margin="16dp" />

    <Button
        android:id="@+id/signInButton"
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:text="Sign In"
        android:textColor="#FFFFFF"
        android:background="@drawable/rounded_button_bg"
        android:layout_gravity="center"
        android:layout_margin="16dp"/>

    <TextView
        android:id="@+id/txtSignUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Don't have an account? Sign Up"
        android:textColor="#00284d"
        android:layout_margin="16dp" />
</LinearLayout>
```

SignUpActivity.java

```java
package com.example.trainbooking_mobileapp.UserManagement;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.example.trainbooking_mobileapp.R;
import com.google.gson.Gson;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class SignUpActivity extends AppCompatActivity {

    // EditText fields
    private EditText etFirstName, etLastName, etEmail, etNIC, etUserName,
etPassword, etRePassword, etContactNumber;
    // Button for sign up
    private Button btnSignUp;
    // Spinner for gender selection
    Spinner spinnerGender;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);

        // Initialize EditText fields and Button
        etFirstName = findViewById(R.id.etFirstName);
        etLastName = findViewById(R.id.etLastName);
        etEmail = findViewById(R.id.etEmail);
```

```
        etNIC = findViewById(R.id.etNIC);
        spinnerGender = findViewById(R.id.spinnerGender);
        etUserName = findViewById(R.id.etUserName);
        etPassword = findViewById(R.id.etPassword);
        etRePassword = findViewById(R.id.etRePassword);
        etContactNumber = findViewById(R.id.etContactNumber);
        btnSignUp = findViewById(R.id.btnSignUp);

        // Set click listener for sign up button
        btnSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                signUp();
            }
        });

        // Set click listener for "Sign In" TextView
        TextView txtSignIn = findViewById(R.id.txtSignIn);
        txtSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(SignUpActivity.this,
SignInActivity.class);
                startActivity(intent);
                finish();
            }
        });

        // Get the Spinner
        Spinner spinnerGender = findViewById(R.id.spinnerGender);

        // Create an ArrayAdapter for the spinner
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.gender_array,
R.layout.custom_spinner_dropdown_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        spinnerGender.setAdapter(adapter);
    }

    // Method to handle sign up process
    private void signUp() {
        // Get values from EditText fields
        String firstName = etFirstName.getText().toString().trim();
        String lastName = etLastName.getText().toString().trim();
```

```
        String email = etEmail.getText().toString().trim();
        String nic = etNIC.getText().toString().trim();
        String gender = spinnerGender.getSelectedItem().toString();
        String username = etUserName.getText().toString().trim();
        String password = etPassword.getText().toString().trim();
        String reenteredPassword = etRePassword.getText().toString().trim();
        String contactNumber = etContactNumber.getText().toString().trim();

        // Validate email, password, NIC, and contact number
        if (!isValidEmail(email)) {
            Toast.makeText(SignUpActivity.this, "Invalid email format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidPassword(password)) {
            Toast.makeText(SignUpActivity.this, "Invalid password format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!password.equals(reenteredPassword)) {
            Toast.makeText(SignUpActivity.this, "Passwords do not match.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidNIC(nic)) {
            Toast.makeText(SignUpActivity.this, "Invalid NIC format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidContactNumber(contactNumber)) {
            Toast.makeText(SignUpActivity.this, "Invalid contact number format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        // Create a User object
        User user = new User("", firstName, lastName, username, email, nic,
gender, contactNumber, "", password, reenteredPassword, "");

        // Use OkHttp to send a POST request to the server
        OkHttpClient client = new OkHttpClient();
```

```
        MediaType JSON = MediaType.parse("application/json; charset=utf-8");
        RequestBody requestBody = RequestBody.create(JSON, new
Gson().toJson(user));

        Request request = new Request.Builder()
                .url("http://pasinduperera-001-
site1.atempurl.com/api/users/signup")
                .post(requestBody)
                .build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                e.printStackTrace();
                Log.e("SignUpActivity", "Error during sign up: " +
e.getMessage());
            }

            @Override
            public void onResponse(Call call, Response response) throws
IOException {
                if (response.isSuccessful()) {
                    Log.d("SignUpActivity", "Sign up successful!");

                    final String responseData = response.body().string();
                    SignUpActivity.this.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {

                            Toast.makeText(SignUpActivity.this, "Sign up
successful!", Toast.LENGTH_SHORT).show();

                            Intent intent = new Intent(SignUpActivity.this,
SignInActivity.class);
                            startActivity(intent);
                            finish();
                        }
                    });

                } else {

                    final String errorResponse = response.body().string();
                    SignUpActivity.this.runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
```

```
                            Toast.makeText(SignUpActivity.this, "Sign up failed:
" + errorResponse, Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            }
        });
    }

    // Method to hash the password
    private String hashPassword(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hashBytes = md.digest(password.getBytes());
            StringBuilder hashString = new StringBuilder();

            for (byte hashByte : hashBytes) {
                String hex = Integer.toHexString(0xff & hashByte);
                if (hex.length() == 1) hashString.append('0');
                hashString.append(hex);
            }

            return hashString.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        return null;
    }

    // Method to validate email format
    private boolean isValidEmail(String email) {
        String emailPattern = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-
.]+$";
        return email.matches(emailPattern);
    }

    // Method to validate password format
    private boolean isValidPassword(String password) {
        String passwordPattern = "^(?=.*[a-z])(?=.*[A-
Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]{8,}$";
        return password.matches(passwordPattern);
    }
```

```
    // Method to validate NIC format
    private boolean isValidNIC(String nic) {
        String nicPattern = "^\\d{12}$";
        return nic.matches(nicPattern);
    }

    // Method to validate contact number format
    private boolean isValidContactNumber(String contactNumber) {
        String contactNumberPattern = "^\\d{10}$";
        return contactNumber.matches(contactNumberPattern);
    }
}
```

Activity_sign_up.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="@drawable/background"
    tools:context=".UserManagement.SignUpActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp"
        tools:context=".MainActivity">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SIGN UP"
            android:textSize="30sp"
            android:layout_gravity="center"
            android:textColor="@color/white"
            android:layout_marginTop="87dp" />

        <LinearLayout
            android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:orientation="horizontal"
android:layout_marginTop="70dp">

<!-- Column 1 -->
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical">



    <EditText
        android:id="@+id/etFirstName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="First Name"
        android:textColor="@color/white"
        android:background="@drawable/rounded_border"
        android:padding="10dp"
        android:textColorHint="@color/white"
        android:layout_margin="16dp" />

    <EditText
        android:id="@+id/etLastName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Last Name"
        android:textColor="@color/white"
        android:background="@drawable/rounded_border"
        android:padding="10dp"
        android:textColorHint="@color/white"
        android:layout_margin="8dp" />

    <EditText
        android:id="@+id/etUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="UserName"
        android:textColor="@color/white"
        android:background="@drawable/rounded_border"
        android:padding="10dp"
        android:textColorHint="@color/white"
        android:layout_margin="8dp" />
```

```
    <EditText
        android:id="@+id/etNIC"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="NIC"
        android:textColor="@color/white"
        android:background="@drawable/rounded_border"
        android:padding="10dp"
        android:textColorHint="@color/white"
        android:layout_margin="8dp" />

    <EditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:hint="Email"
        android:textColor="@color/white"
        android:background="@drawable/rounded_border"
        android:padding="10dp"
        android:textColorHint="@color/white"
        android:layout_margin="8dp" />

</LinearLayout>

<!-- Column 2 -->
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical">

    <Spinner
        android:id="@+id/spinnerGender"
        android:layout_width="match_parent"
        android:layout_height="47dp"
        android:layout_margin="16dp"
        android:background="@drawable/rounded_border"
        android:hint="Gender"
        android:minHeight="16dp"
        android:padding="10dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />
```

```
<EditText
    android:id="@+id/etContactNumber"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="phone"
    android:hint="Contact Number"
    android:textColor="@color/white"
    android:background="@drawable/rounded_border"
    android:padding="10dp"
    android:textColorHint="@color/white"
    android:layout_margin="8dp" />

<EditText
    android:id="@+id/etPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Password"
    android:textColor="@color/white"
    android:background="@drawable/rounded_border"
    android:padding="10dp"
    android:textColorHint="@color/white"
    android:layout_margin="8dp" />

<EditText
    android:id="@+id/etRePassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Reenter Password"
    android:textColor="@color/white"
    android:background="@drawable/rounded_border"
    android:padding="10dp"
    android:textColorHint="@color/white"
    android:layout_margin="8dp" />

<TextView
    android:id="@+id/etUserType"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="text"
    android:textSize="17sp"
    android:text="Traveler"
    android:textColor="@color/white"
    android:background="@drawable/rounded_border"
```

```
                    android:padding="10dp"
                    android:textColorHint="@color/white"
                    android:layout_margin="8dp" />

            </LinearLayout>

        </LinearLayout>

        <Button
            android:id="@+id/btnSignUp"
            android:layout_width="170dp"
            android:layout_height="wrap_content"
            android:text="Sign Up"
            android:textColor="#FFFFFF"
            android:background="@drawable/rounded_button_bg"
            android:padding="12dp"
            android:layout_gravity="center"
            android:layout_margin="16dp" />

        <TextView
            android:id="@+id/txtSignIn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Already have an Account? Sign In"
            android:textColor="#00284d"
            android:layout_gravity="center"
            android:layout_margin="16dp" />

    </LinearLayout>

</LinearLayout>
```

UserProfileActivity.java

```
package com.example.trainbooking_mobileapp.UserManagement;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
```

```java
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class UserProfileActivity extends AppCompatActivity {

    // Declare TextViews
    private TextView nicTextView;
    private TextView firstNameTextView;
    private TextView lastNameTextView;
    private TextView emailTextView;
    private TextView genderTextView;
    private TextView usernameTextView;
    private TextView contactNumberTextView;

    // Declare User and Toolbar
    private User user;
    private Toolbar toolbar;

    // Declare userID
    private String userID;

    // Method for setting up the activity
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_profile);

        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
        setTitle("My Profile");

        getSupportActionBar().setDisplayHomeAsUpEnabled(false);

        userID = getIntent().getStringExtra("userID");

        Button updateButton = findViewById(R.id.updateButton);
        Button deactivateButton = findViewById(R.id.deactivateButton);

// Initialize and set up TextViews and Buttons
        nicTextView = findViewById(R.id.nicTextView);
        firstNameTextView = findViewById(R.id.firstNameTextView);
        lastNameTextView = findViewById(R.id.lastNameTextView);
        usernameTextView = findViewById(R.id.usernameTextView);
        emailTextView = findViewById(R.id.emailTextView);
        genderTextView = findViewById(R.id.genderTextView);
        contactNumberTextView = findViewById(R.id.contactNumberTextView);

        new FetchUserDataTask().execute(userID);

        // Set click listeners for buttons
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (user != null) {
                    Intent intent = new Intent(UserProfileActivity.this,
UpdateUserProfileActivity.class);
                    intent.putExtra("user", user);
                    startActivityForResult(intent, 1001);
                } else {
                    Toast.makeText(UserProfileActivity.this, "User data not
available", Toast.LENGTH_SHORT).show();
                }
            }
        });

        deactivateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (user != null) {
                    new DeactivateUserTask().execute(userID);
                } else {
                    Toast.makeText(UserProfileActivity.this, "User data not
available", Toast.LENGTH_SHORT).show();
                }
```

```
            }
        });

        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button5 = findViewById(R.id.button5);
        ImageButton Button6 = findViewById(R.id.button6);
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UserProfileActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);

            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UserProfileActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UserProfileActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    // Method for handling options menu items
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
```

```
        } else if (id == android.R.id.home) {
            onBackPressed();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // Method for signing out
    private void signOut() {

        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }

    // AsyncTask for deactivating a user
    private class DeactivateUserTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... params) {
            String userID = params[0];
            Log.d("DeactivateUserTask", "Userid: " + userID);
            return deactivateUser(userID);
        }

        @Override
        protected void onPostExecute(String response) {
            if (response != null && response.equals("Success")) {
                Toast.makeText(UserProfileActivity.this, "User deactivated
successfully", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(UserProfileActivity.this,
SignInActivity.class);
                startActivity(intent);
            } else {
                Toast.makeText(UserProfileActivity.this, "Error deactivating
user", Toast.LENGTH_SHORT).show();
            }
        }
    }

    // Method for deactivating a user
    private String deactivateUser(String userID) {
        try {
            URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/users/updateuserstatus/" + userID);
```

```
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("PUT");
            connection.setRequestProperty("Content-Type", "application/json");

            JSONObject jsonStatusData = new JSONObject();
            jsonStatusData.put("UserStatus", "Deactive");

            OutputStream outputStream = connection.getOutputStream();
            outputStream.write(jsonStatusData.toString().getBytes("UTF-8"));
            outputStream.close();

            int responseCode = connection.getResponseCode();

            if (responseCode == HttpURLConnection.HTTP_OK) {
                return "Success";
            } else {
                return "Error. Response code: " + responseCode;
            }
        } catch (Exception e) {
            e.printStackTrace();
            return "Error. Exception: " + e.getMessage();
        }
    }

    // AsyncTask for fetching user data
    private class FetchUserDataTask extends AsyncTask<String, Void, JSONObject> {
        @Override
        protected JSONObject doInBackground(String... params) {
            OkHttpClient client = new OkHttpClient();
            String userID = params[0];
            String apiUrl = "http://pasinduperera-001-
site1.atempurl.com/api/users/getuser/" + userID;

            Request request = new Request.Builder()
                    .url(apiUrl)
                    .get()
                    .build();
            try {
                Response response = client.newCall(request).execute();
                if (response.isSuccessful()) {
                    String responseData = response.body().string();
                    return new JSONObject(responseData);
                }
            } catch (IOException | JSONException e) {
```

```
                e.printStackTrace();
            }
            return null;
        }

        // Handling the result of an activity
        @Override
        protected void onPostExecute(JSONObject userData) {
            if (userData != null) {
                try {
                    String userID = userData.getString("UserID");
                    String firstName = userData.getString("FirstName");
                    String lastName = userData.getString("LastName");
                    String username = userData.getString("UserName");
                    String email = userData.getString("Email");
                    String NIC = userData.getString("NIC");
                    String gender = userData.getString("Gender");
                    String contactNumber = userData.getString("ContactNumber");

                    user = new User( userID, firstName, lastName, username,
email, NIC, gender, contactNumber, "", "", "", "");

                    firstNameTextView.setText("First Name: " + firstName);
                    lastNameTextView.setText("Last Name: " + lastName);
                    usernameTextView.setText("User Name: " + username);
                    emailTextView.setText("Email: " + email);
                    nicTextView.setText("NIC: " + NIC);
                    genderTextView.setText("Gender: " + gender);
                    contactNumberTextView.setText("Contact Number: " +
contactNumber);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Toast.makeText(UserProfileActivity.this, "Error fetching user
data", Toast.LENGTH_SHORT).show();
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == 1001 && resultCode == RESULT_OK) {
            String userID = getIntent().getStringExtra("userID");
            new FetchUserDataTask().execute(userID);
        }
    }
}
```

Activity_user_profile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"

    android:background="@drawable/background"
    tools:context=".UserManagement.UserProfileActivity">

    <include layout="@layout/top_bar" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginBottom="16dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:background="#8B00284D"
            android:padding="16dp"
            android:gravity="center_horizontal">

    <ImageView
        android:id="@+id/profileImage"
        android:layout_width="170dp"
        android:layout_height="160dp"
        android:src="@drawable/profileimage"
    android:layout_gravity="center"
    android:layout_marginTop="16dp"
```

```
/>

        <TextView
            android:id="@+id/usernameTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="User Name"
            android:textSize="16sp"
            android:textStyle="bold"
            android:textColor="@color/white"
            android:layout_margin="16dp"
            />


<TextView
    android:id="@+id/firstNameTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="First Name"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/white"
    android:layout_margin="16dp"
    />

<TextView
    android:id="@+id/lastNameTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Last Name"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/white"
    android:layout_margin="16dp"
    />

<TextView
    android:id="@+id/nicTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="NIC"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/white"
    android:layout_margin="16dp"
```

```
    />

<TextView
    android:id="@+id/emailTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Email"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/white"
    android:layout_margin="16dp"
    />

        <TextView
            android:id="@+id/genderTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="gender"
            android:textSize="16sp"
            android:textStyle="bold"
            android:textColor="@color/white"
            android:layout_margin="16dp"
            />

<TextView
    android:id="@+id/contactNumberTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Contact Number"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="@color/white"
    android:layout_margin="16dp"
    />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center_horizontal"
    android:layout_marginTop="16dp">

<Button
    android:id="@+id/updateButton"
    android:layout_width="179dp"
```

```
    android:layout_height="wrap_content"
    android:text="Update"
    android:textColor="#FFFFFF"
    android:background="@drawable/rounded_border_update"
    android:layout_marginRight="16dp"
/>

<Button
    android:id="@+id/deactivateButton"
    android:layout_width="170dp"
    android:layout_height="wrap_content"
    android:text="Deactivate"
    android:textColor="#FFFFFF"
android:background="@drawable/rounded_border_delete"
    android:layout_marginRight="16dp"
    />

</LinearLayout>

    </LinearLayout>
</ScrollView>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bottom_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_alignParentBottom="true"
    android:background="#00284d"
    android:elevation="8dp">

    <ImageButton
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_home_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button5"
```

```
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</LinearLayout>
```

UpdateUserProfileActivity.java

```
package com.example.trainbooking_mobileapp.UserManagement;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;

public class UpdateUserProfileActivity extends AppCompatActivity {

    // Declare EditText, Button, User, Toolbar, and String variables
```

```
    private EditText updatedUserNameEditText, updatedFirstNameEditText,
updatedLastNameEditText, updatedPhoneNumberEditText, updatedEmailEditText,
updatedGenderEditText;
    private Button updateButton;
    private User user;
    private Toolbar toolbar;
    private String userID;
    private  Spinner genderSpinner;

    // Validate email format
    private boolean isValidEmail(String email) {
        String emailPattern = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-
.]+$";
        return email.matches(emailPattern);
    }

    // Validate contact number format
    private boolean isValidContactNumber(String contactNumber) {
        String contactNumberPattern = "^\\d{10}$";
        return contactNumber.matches(contactNumberPattern);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_user_profile);

        // Initialize UI elements and retrieve user information
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        setTitle("Update Profile");
        user = (User) getIntent().getSerializableExtra("user");
        userID = getIntent().getStringExtra("userID");

        updatedFirstNameEditText = findViewById(R.id.updatedFirstNameEditText);
        updatedLastNameEditText = findViewById(R.id.updatedLastNameEditText);
        updatedUserNameEditText = findViewById(R.id.updatedUserNameEditText);
        updatedEmailEditText = findViewById(R.id.updatedEmailEditText);
        updatedPhoneNumberEditText =
findViewById(R.id.updatedPhoneNumberEditText);
        updateButton = findViewById(R.id.updateButton);

        // Populate fields with user information
        populateFields();
```

```
        // Set click listeners for buttons and image buttons
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateUser();
            }
        });

        String userID = getIntent().getStringExtra("userID");

        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button5 = findViewById(R.id.button5);
        ImageButton Button6 = findViewById(R.id.button6);
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateUserProfileActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateUserProfileActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateUserProfileActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    // Handle menu item selection
    @Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_sign_out) {
        signOut();
        return true;
    } else if (id == android.R.id.home) {
        onBackPressed();
        return true;
    }

    return super.onOptionsItemSelected(item);
}

// Sign out and navigate to sign in activity
private void signOut() {
    Intent intent = new Intent(this, SignInActivity.class);
    startActivity(intent);
    finish();
}

// Populate fields with user information
private void populateFields() {
    String ID = user.getID();
    String firstName = user.getFirstName();
    String lastName = user.getLastName();
    String userName = user.getUserName();
    String email = user.getEmail();
    String contactNumber = user.getContactNumber();

    updatedFirstNameEditText.setText(firstName);
    updatedLastNameEditText.setText(lastName);
    updatedUserNameEditText.setText(userName);
    updatedEmailEditText.setText(email);
    updatedPhoneNumberEditText.setText(contactNumber);
}

// Update user information
private void updateUser() {
    String firstName = updatedFirstNameEditText.getText().toString();
    String lastName = updatedLastNameEditText.getText().toString();
    String userName = updatedUserNameEditText.getText().toString();
    String email = updatedEmailEditText.getText().toString();
    String contactNumber = updatedPhoneNumberEditText.getText().toString();
```

```
        if (!isValidEmail(email)) {
            Toast.makeText(UpdateUserProfileActivity.this, "Invalid email
format.", Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidContactNumber(contactNumber)) {
            Toast.makeText(UpdateUserProfileActivity.this, "Invalid contact
number format.", Toast.LENGTH_SHORT).show();
            return;
        }

        User updatedUser = new User(user.getID(), firstName, lastName, userName,
email, user.getNIC(), user.getGender(), contactNumber, user.getUserType(),
user.getPassword(), user.getRePassword(), user.getUserStatus());

        UserApiClient.updateUserInAPI(updatedUser, new
UserApiClient.OnUserUpdatedListener() {
            @Override
            public void onUserUpdated() {
                setResult(RESULT_OK);
                finish();
                Toast.makeText(UpdateUserProfileActivity.this, "User updated
successfully", Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onError(String errorMessage) {
                Toast.makeText(UpdateUserProfileActivity.this, "Error updating
user", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Activity_update_user_profile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
android:background="@drawable/background"
tools:context=".UserManagement.UpdateUserProfileActivity">

<include layout="@layout/top_bar" />

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:padding="16dp"
    android:layout_marginBottom="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

<ImageView
    android:id="@+id/profileImage"
    android:layout_width="170dp"
    android:layout_height="180dp"
    android:src="@drawable/profileimage"
    android:layout_gravity="center"
    android:layout_marginTop="4dp"
    android:layout_marginBottom="16dp"
    />

<EditText
    android:id="@+id/updatedUserNameEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="UserName"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:background="@drawable/rounded_border"
    android:layout_marginBottom="16dp" />

<EditText
    android:id="@+id/updatedFirstNameEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="First Name"
    android:padding="8dp"
    android:textColor="@color/white"
```

```
    android:textColorHint="@color/white"
    android:background="@drawable/rounded_border"
    android:layout_marginBottom="16dp" />

<EditText
    android:id="@+id/updatedLastNameEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Last Name"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:background="@drawable/rounded_border"
    android:layout_marginBottom="16dp" />

<EditText
    android:id="@+id/updatedPhoneNumberEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="phone"
    android:hint="Contact Number"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:background="@drawable/rounded_border"
    android:layout_marginBottom="16dp" />

        <EditText
    android:id="@+id/updatedEmailEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress"
    android:hint="Email"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:background="@drawable/rounded_border"
    android:layout_marginBottom="16dp" />

<Button
    android:id="@+id/updateButton"
    android:layout_width="170dp"
    android:layout_height="wrap_content"
    android:text="Update"
    android:padding="12dp"
```

```
    android:layout_gravity="center"
    android:layout_marginTop="16dp"
    android:background="@drawable/rounded_border_update"
    android:layout_marginBottom="16dp"
    android:textColor="#FFFFFF"/>

    </LinearLayout>
</ScrollView>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bottom_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_alignParentBottom="true"
    android:background="#00284d"
    android:elevation="8dp">

    <ImageButton
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_home_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button5"
        android:layout_width="47dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_account_circle_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button6"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
        android:background="#00284d"
```

```
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</LinearLayout>
```

Train.java

```
package com.example.trainbooking_mobileapp.TrainManagement;

public class Train {
    private String TrainID;
    private String trainName;
    private String departureTime;
    private String arrivalTime;
    private String departureStation;
    private String arrivalStation;
    private String firstClassTicketPrice;
    private String secondClassTicketPrice;
    private String thirdClassTicketPrice;
    private String status;

    // Constructor to initialize Train object
    public Train(String TrainID, String trainName, String departureTime, String
arrivalTime, String departureStation, String arrivalStation, String
firstClassTicketPrice, String secondClassTicketPrice, String
thirdClassTicketPrice, String status) {
        this.TrainID = TrainID;
        this.trainName = trainName;
        this.departureTime = departureTime;
        this.arrivalTime = arrivalTime;
        this.departureStation = departureStation;
        this.arrivalStation = arrivalStation;
        this.firstClassTicketPrice = firstClassTicketPrice;
        this.secondClassTicketPrice = secondClassTicketPrice;
        this.thirdClassTicketPrice = thirdClassTicketPrice;
        this.status = status;
    }
```

```java
// Getter for TrainID
public String getTrainID() {
    return TrainID;
}

// Getter for trainName
public String getTrainName() {
    return trainName;
}

// Getter for departureTime
public String getDepartureTime() {
    return departureTime;
}

// Getter for arrivalTime
public String getArrivalTime() {
    return arrivalTime;
}

// Getter for departureStation
public String getDepartureStation() {
    return departureStation;
}

// Getter for arrivalStation
public String getArrivalStation() {
    return arrivalStation;
}

// Getter for firstClassTicketPrice
public String getFirstClassTicketPrice() {
    return firstClassTicketPrice;
}

// Getter for secondClassTicketPrice
public String getSecondClassTicketPrice() {
    return secondClassTicketPrice;
}

// Getter for thirdClassTicketPrice
public String getThirdClassTicketPrice() {
    return thirdClassTicketPrice;
}
```

```
    // Getter for status
    public String getStatus() {
        return status;
    }
}
```

TrainApiClient.java

```
package com.example.trainbooking_mobileapp.TrainManagement;

import android.annotation.SuppressLint;
import android.os.AsyncTask;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

public class TrainApiClient {

    // URL for fetching train data from the API
    private static final String API_URL = "http://pasinduperera-001-
site1.atempurl.com/api/trains/getalltrains";

    // Listener interface for receiving train data or error messages
    public interface OnTrainDataReceivedListener {
        void onTrainDataReceived(List<Train> trainList);
        void onError(String errorMessage);
    }

    // Method to fetch active trains from the API
    public void getActiveTrainsFromAPI(final OnTrainDataReceivedListener
listener) {
        // AsyncTask to perform network operation in the background
        @SuppressLint("StaticFieldLeak") AsyncTask<Void, Void, String> task = new
AsyncTask<Void, Void, String>() {
            @Override
            protected String doInBackground(Void... voids) {
                try {
                    // Establish connection and read response
```

```
                    URL url = new URL(API_URL);
                    HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                    BufferedReader reader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new StringBuilder();

                    String line;
                    while ((line = reader.readLine()) != null) {
                        stringBuilder.append(line);
                    }

                    return stringBuilder.toString();
                } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                }
            }

            @Override
            protected void onPostExecute(String response) {
                if (response != null) {
                    try {
                        // Parse JSON and notify listener with train data
                        List<Train> trainList = parseJson(response);
                        listener.onTrainDataReceived(trainList);
                    } catch (JSONException e) {
                        e.printStackTrace();
                        listener.onError("Error parsing JSON");
                    }
                } else {
                    listener.onError("Error fetching data from API");
                }
            }
        };

        task.execute(); // Execute the AsyncTask
    }

    // Method to parse JSON response into a list of Train objects
    private List<Train> parseJson(String json) throws JSONException {
        List<Train> trainList = new ArrayList<>();

        JSONArray jsonArray = new JSONArray(json);
        for (int i = 0; i < jsonArray.length(); i++) {
```

```
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            // Extract train information from JSON
            String TrainID = jsonObject.getString("TrainID");
            String trainName = jsonObject.getString("TrainName");
            String departureTime =
jsonObject.getString("FormattedDepartureTime");
            String arrivalTime = jsonObject.getString("FormattedArrivalTime");
            String departureStation = jsonObject.getString("DepartureStation");
            String arrivalStation = jsonObject.getString("ArrivalStation");
            String firstClassTicketPrice =
jsonObject.getString("FirstClassTicketPrice");
            String secondClassTicketPrice =
jsonObject.getString("SecondClassTicketPrice");
            String thirdClassTicketPrice =
jsonObject.getString("ThirdClassTicketPrice");
            String status = jsonObject.getString("TrainStatus");
            // Create Train object and add to the list
            Train train = new Train(TrainID, trainName, departureTime,
arrivalTime, departureStation, arrivalStation, firstClassTicketPrice,
secondClassTicketPrice, thirdClassTicketPrice, status);
            trainList.add(train);
        }
        return trainList;
    }
}
```

TrainListAdapter.java

```
package com.example.trainbooking_mobileapp.TrainManagement;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.trainbooking_mobileapp.R;
import java.util.List;

public class TrainListAdapter extends
RecyclerView.Adapter<TrainListAdapter.ViewHolder> {

    // List of trains to display
```

```
    private List<Train> trainList;

    // Inflater for inflating the layout
    private LayoutInflater inflater;

    // Constructor to initialize the adapter
    public TrainListAdapter(Context context, List<Train> trainList) {
        this.inflater = LayoutInflater.from(context);
        this.trainList = trainList;
    }

    // Create new views (invoked by the layout manager)
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view = inflater.inflate(R.layout.train_list_item, parent, false);
        return new ViewHolder(view);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Train train = trainList.get(position);

        // Set train details to the respective TextViews

        // Train Name
        holder.trainNameTextView.setText(train.getTrainName());

        // Departure Time
        if (train.getDepartureTime() != null) {
            holder.departureTimeTextView.setText(train.getDepartureTime());
        } else {
            holder.departureTimeTextView.setText("N/A");
        }

        // Arrival Time
        if (train.getArrivalTime() != null) {
            holder.arrivalTimeTextView.setText(train.getArrivalTime());
        } else {
            holder.arrivalTimeTextView.setText("N/A");
        }

        // Departure Station
```

```
        holder.departureStationTextView.setText(train.getDepartureStation());

        // Arrival Station
        holder.arrivalStationTextView.setText(train.getArrivalStation());

        // First Class Ticket Price
        holder.firstClassTicketPriceTextView.setText(train.getFirstClassTicketPri
ce());

        // Second Class Ticket Price
        holder.secondClassTicketPriceTextView.setText(train.getSecondClassTicketP
rice());

        // Third Class Ticket Price
        holder.thirdClassTicketPriceTextView.setText(train.getThirdClassTicketPri
ce());

        // Train Status
        holder.statusTextView.setText(train.getStatus());
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return trainList.size();
    }

    // Provide a reference to the views for each data item
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView trainNameTextView, departureTimeTextView, arrivalTimeTextView,
departureStationTextView, arrivalStationTextView, firstClassTicketPriceTextView,
secondClassTicketPriceTextView, thirdClassTicketPriceTextView, statusTextView;

        // Constructor to initialize the views
        public ViewHolder(@NonNull View itemView) {
            super(itemView);

            // Initialize TextViews

            trainNameTextView = itemView.findViewById(R.id.trainNameTextView);
            departureTimeTextView =
itemView.findViewById(R.id.departureTimeTextView);
            arrivalTimeTextView =
itemView.findViewById(R.id.arrivalTimeTextView);
```

```
            departureStationTextView =
itemView.findViewById(R.id.departureStationTextView);
            arrivalStationTextView =
itemView.findViewById(R.id.arrivalStationTextView);
            firstClassTicketPriceTextView =
itemView.findViewById(R.id.firstClassTicketPriceTextView);
            secondClassTicketPriceTextView =
itemView.findViewById(R.id.secondClassTicketPriceTextView);
            thirdClassTicketPriceTextView =
itemView.findViewById(R.id.thirdClassTicketPriceTextView);
            statusTextView = itemView.findViewById(R.id.statusTextView);
        }
    }
}
```

TrainDetailsActivity.java

```
package com.example.trainbooking_mobileapp.TrainManagement;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

import java.util.List;

public class TrainDetailsActivity extends AppCompatActivity implements
TrainApiClient.OnTrainDataReceivedListener {

    // RecyclerView to display train list
    private RecyclerView recyclerView;

    // Adapter for the RecyclerView
```

```
private TrainListAdapter trainListAdapter;

// API client for fetching train data
private TrainApiClient trainApiClient;

// Toolbar for the activity
private Toolbar toolbar;

// User ID associated with the activity
private String userID;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Set the content view to the activity_train_list.xml layout
    setContentView(R.layout.activity_train_list);

    // Initialize the toolbar
    toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    // Set the title of the activity
    setTitle("Train Details");

    // Enable the back button in the toolbar
    getSupportActionBar().setDisplayHomeAsUpEnabled(false);

    // Initialize the RecyclerView and set its layout manager
    recyclerView = findViewById(R.id.trainRecyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    // Initialize the TrainApiClient
    trainApiClient = new TrainApiClient();

    // Fetch active trains from the API
    trainApiClient.getActiveTrainsFromAPI(this);

    // Get the user ID from the intent
    userID = getIntent().getStringExtra("userID");

    // Initialize buttons and set click listeners
    ImageButton Button1 = findViewById(R.id.button1);
    ImageButton Button5 = findViewById(R.id.button5);
    ImageButton Button6 = findViewById(R.id.button6);
```

```
        // OnClickListener for Button1
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(TrainDetailsActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        // OnClickListener for Button5
        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(TrainDetailsActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        // OnClickListener for Button6
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(TrainDetailsActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    // Handle toolbar item selection
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
        } else if (id == android.R.id.home) {
            onBackPressed();
```

```
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // Method to sign out and redirect to SignInActivity
    private void signOut() {
        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }

    // Callback method when train data is received from the API
    @Override
    public void onTrainDataReceived(List<Train> trainList) {
        // Initialize the adapter and set it to the RecyclerView
        trainListAdapter = new TrainListAdapter(this, trainList);
        recyclerView.setAdapter(trainListAdapter);
    }

    // Callback method when an error occurs while fetching train data
    @Override
    public void onError(String errorMessage) {
        Log.e("TrainListActivity", errorMessage);
    }
}
```

Activity_train_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background"
    tools:context=".TrainManagement.TrainDetailsActivity">

    <include layout="@layout/top_bar" />

    <androidx.recyclerview.widget.RecyclerView
```

```
    android:id="@+id/trainRecyclerView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:padding="16dp"
    android:clipToPadding="false"
    android:scrollbars="vertical"
    tools:listitem="@layout/train_list_item"
    android:layout_marginBottom="8dp"/>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bottom_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_alignParentBottom="true"
    android:background="#00284d"
    android:elevation="8dp">

    <ImageButton
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_home_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button5"
        android:layout_width="47dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_account_circle_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button6"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
```

```
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</androidx.appcompat.widget.LinearLayoutCompat>
```

Activity_train_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="#B3DAFF"
    android:padding="16dp">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_border"
        android:orientation="horizontal"
        android:padding="8dp"
        android:gravity="center_horizontal"
        android:layout_margin="4dp">

        <TextView
            android:id="@+id/trainNameLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Train Name :"
            android:textSize="16sp"
            android:textColor="@color/white"
            android:padding="8dp"
            android:gravity="center_vertical"
            android:textStyle="bold"
            android:layout_gravity="center_horizontal" />

        <TextView
            android:id="@+id/trainNameTextView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/departureTimeLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Train Departure Time :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/departureTimeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/arrivalTimeLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="Train Arrival Time :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/arrivalTimeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
```

```
        android:id="@+id/departureStationLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Train Departure Station :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/departureStationTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/arrivalStationLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Train Arrival Station :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />
```

```
    <TextView
        android:id="@+id/arrivalStationTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/firstClassTicketPriceLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="First Class Ticket Price (Rs):"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/firstClassTicketPriceTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>
```

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/secondClassTicketPriceLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Class Ticket Price (Rs):"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/secondClassTicketPriceTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">
```

```
    <TextView
        android:id="@+id/thirdClassTicketPriceLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Third Class Ticket Price (Rs):"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:id="@+id/thirdClassTicketPriceTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal" />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/statusLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Train Status :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal" />
```

```
        <TextView
            android:id="@+id/statusTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="@color/white"
            android:padding="8dp"
            android:gravity="center_vertical"
            android:layout_gravity="center_horizontal" />

    </LinearLayout>

    <View
        android:layout_width="wrap_content"
        android:layout_height="1dp"
        android:background="#4CAF50" />

    </LinearLayout>
```

Reservation.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import java.io.Serializable;

public class Reservation implements Serializable {
    private String BookingID;
    private String TrainNumber;
    private String TrainName;
    private String userId;
    private String BookingDate;
    private String ReservationDate;
    private int TotalPassengers;
    private String MainPassengerName;
    private String ContactNumber;
    private String DepartureStation;
    private String DestinationStation;
    private String Email;
    private String NIC;
    private String TicketClass;

    public Reservation(String bookingID, String trainNumber, String trainName,
String userID, String bookingDate,
```

```
                        String reservationDate, int totalPassengers, String
mainPassengerName, String contactNumber,
                        String departureStation, String destinationStation, String
email, String nic,
                        String ticketClass) {
        BookingID = bookingID;
        TrainNumber = trainNumber;
        TrainName = trainName;
        userId = userID;
        BookingDate = bookingDate;
        ReservationDate = reservationDate;
        TotalPassengers = totalPassengers;
        MainPassengerName = mainPassengerName;
        ContactNumber = contactNumber;
        DepartureStation = departureStation;
        DestinationStation = destinationStation;
        Email = email;
        NIC = nic;
        TicketClass = ticketClass;
    }

    /*
     * Get the booking ID.
     */
    public String getBookingID() {
        return BookingID;
    }

    /*
     * Set the booking ID.
     */
    public void setBookingID(String bookingID) {
        BookingID = bookingID;
    }

    /*
     * Get the train number.
     */
    public String getTrainNumber() {
        return TrainNumber;
    }

    /*
     * Set the train number.
     */
```

```java
public void setTrainNumber(String trainNumber) {
    TrainNumber = trainNumber;
}

/*
 * Get the train name.
 */
public String getTrainName() {
    return TrainName;
}

/*
 * Set the train name.
 */
public void setTrainName(String trainName) {
    TrainName = trainName;
}

/*
 * Get the user ID.
 */
public String getUserID() {
    return userId;
}

/*
 * Set the user ID.
 */
public void setUserID(String userID) {
    userId = userId;
}

/*
 * Get the booking date.
 */
public String getBookingDate() {
    return BookingDate;
}

/*
 * Set the booking date.
 */
public void setBookingDate(String bookingDate) {
    BookingDate = bookingDate;
}
```

```java
/*
 * Get the reservation date.
 */
public String getReservationDate() {
    return ReservationDate;
}

/*
 * Set the reservation date.
 */
public void setReservationDate(String reservationDate) {
    ReservationDate = reservationDate;
}

/*
 * Get the total number of passengers.
 */
public int getTotalPassengers() {
    return TotalPassengers;
}

/*
 * Set the total number of passengers.
 */
public void setTotalPassengers(int totalPassengers) {
    TotalPassengers = totalPassengers;
}

/*
 * Get the main passenger's name.
 */
public String getMainPassengerName() {
    return MainPassengerName;
}

/*
 * Set the main passenger's name.
 */
public void setMainPassengerName(String mainPassengerName) {
    MainPassengerName = mainPassengerName;
}

/*
 * Get the contact number.
```

```java
 */
public String getContactNumber() {
    return ContactNumber;
}

/*
 * Set the contact number.
 */
public void setContactNumber(String contactNumber) {
    ContactNumber = contactNumber;
}

/*
 * Get the departure station.
 */
public String getDepartureStation() {
    return DepartureStation;
}

/*
 * Set the departure station.
 */
public void setDepartureStation(String departureStation) {
    DepartureStation = departureStation;
}

/*
 * Get the destination station.
 */
public String getDestinationStation() {
    return DestinationStation;
}

/*
 * Set the destination station.
 */
public void setDestinationStation(String destinationStation) {
    DestinationStation = destinationStation;
}

/*
 * Get the email address.
 */
public String getEmail() {
    return Email;
```

```
    }

    /*
     * Set the email address.
     */
    public void setEmail(String email) {
        Email = email;
    }

    /*
     * Get the NIC (National Identification Card) number.
     */
    public String getNIC() {
        return NIC;
    }

    /*
     * Set the NIC (National Identification Card) number.
     */
    public void setNIC(String NIC) {
        this.NIC = NIC;
    }

    /*
     * Get the ticket class.
     */
    public String getTicketClass() {
        return TicketClass;
    }

    /*
     * Set the ticket class.
     */
    public void setTicketClass(String ticketClass) {
        TicketClass = ticketClass;
    }
}
```

ReservatinApiClient.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.annotation.SuppressLint;
import android.os.AsyncTask;
```

```
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

public class ReservationApiClient {

    public interface OnTrainNamesReceivedListener {
        void onTrainNamesReceived(List<String> trainNames);
        void onError(String errorMessage);
    }

    // Method to get train names from the API
    public void getTrains(final OnTrainNamesReceivedListener listener) {
        @SuppressLint("StaticFieldLeak") AsyncTask<Void, Void, String> task = new
AsyncTask<Void, Void, String>() {
            @Override
            protected String doInBackground(Void... voids) {
                try {
                    URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/trains/getalltrains");
                    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                    connection.setRequestMethod("GET");
                    connection.setRequestProperty("Content-Type",
"application/json");

                    BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                    StringBuilder stringBuilder = new StringBuilder();

                    String line;
                    while ((line = reader.readLine()) != null) {
                        stringBuilder.append(line);
                    }

                    return stringBuilder.toString();
                } catch (Exception e) {
```

```
                    e.printStackTrace();
                    return null;
                }
            }

            @Override
            protected void onPostExecute(String response) {
                if (response != null) {
                    try {
                        List<String> trainNames = parseTrainNamesJson(response);
                        listener.onTrainNamesReceived(trainNames);
                    } catch (JSONException e) {
                        e.printStackTrace();
                        listener.onError("Error parsing JSON");
                    }
                } else {
                    listener.onError("Error fetching data from API");
                }
            }
        };

        task.execute();
    }

    // Method to parse JSON response and extract train names
    private List<String> parseTrainNamesJson(String json) throws JSONException {
        List<String> trainNames = new ArrayList<>();
        JSONArray jsonArray = new JSONArray(json);

        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            String trainName = jsonObject.optString("TrainName");
            trainNames.add(trainName);
        }
        return trainNames;
    }

    public interface OnReservationDataReceivedListener {
        void onReservationDataReceived(List<Reservation> reservationList);
        void onError(String errorMessage);
    }

    // Method to get reservations for a user from the API
    public void getReservationsForUserFromAPI(String userId, final
OnReservationDataReceivedListener listener) {
```

```
        @SuppressLint("StaticFieldLeak") AsyncTask<String, Void, String> task =
new AsyncTask<String, Void, String>() {
            @Override
            protected String doInBackground(String... userId) {
                try {
                    URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/getallticketbookings" + "/" + userId[0]);
                    Log.d("ReservationApiClient", "API call made for userID: " +
userId[0]);
                    HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
                    BufferedReader reader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new StringBuilder();

                    String line;
                    while ((line = reader.readLine()) != null) {
                        stringBuilder.append(line);
                    }

                    return stringBuilder.toString();
                } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                }
            }

            @Override
            protected void onPostExecute(String response) {
                if (response != null) {
                    try {
                        List<Reservation> reservationList = parseJson(response);
                        Log.d("ReservationClient", "API Response: " +
response);  // Log the API response
                        listener.onReservationDataReceived(reservationList);
                    } catch (JSONException e) {
                        e.printStackTrace();
                        listener.onError("Error parsing JSON");
                    }
                } else {
                    listener.onError("Error fetching data from API");
                }
            }
        };
```

```
        task.execute(userId);
    }

    // Method to cancel a reservation via API
    public static void cancelReservationFromAPI(final Reservation reservation,
final OnReservationCanceledListener listener) {
        AsyncTask<Void, Void, String> task = new AsyncTask<Void, Void, String>()
{
            @Override
            protected String doInBackground(Void... voids) {
                try {
                    URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/cancelticketbooking/" +
reservation.getBookingID());
                    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                    connection.setRequestMethod("PUT");
                    connection.setRequestProperty("Content-Type",
"application/json");

                    JSONObject requestBody = new JSONObject();
                    requestBody.put("MainPassengerName",
reservation.getMainPassengerName());
                    requestBody.put("NIC", reservation.getNIC());
                    requestBody.put("TrainNumber", reservation.getTrainNumber());
                    requestBody.put("TrainName", reservation.getTrainName());
                    requestBody.put("DepartureStation",
reservation.getDepartureStation());
                    requestBody.put("DestinationStation",
reservation.getDestinationStation());
                    requestBody.put("TotalPassengers",
reservation.getTotalPassengers());
                    requestBody.put("TicketClass", reservation.getTicketClass());
                    requestBody.put("ContactNumber",
reservation.getContactNumber());
                    requestBody.put("Email", reservation.getEmail());
                    requestBody.put("bookingDate", reservation.getBookingDate());
                    requestBody.put("ReservationDate",
reservation.getReservationDate());

                    connection.setDoOutput(true);
                    OutputStream os = connection.getOutputStream();
                    os.write(requestBody.toString().getBytes("UTF-8"));
                    os.close();
```

```
                    int responseCode = connection.getResponseCode();

                    if (responseCode == HttpURLConnection.HTTP_OK) {
                        return "Success";
                    } else {
                        return "Error";
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                }
            }

            @Override
            protected void onPostExecute(String response) {
                if (response != null && response.equals("Success")) {
                    listener.onReservationCanceled();
                } else {
                    listener.onError("Error canceling reservation");
                }
            }
        };
        task.execute();
    }

    // Method to handle the response after canceling a reservation
    public interface OnReservationCanceledListener {
        void onReservationCanceled();
        void onError(String errorMessage);
    }

    private List<Reservation> parseJson(String json) throws JSONException {
        List<Reservation> reservationList = new ArrayList<>();

        try {
            JSONArray jsonArray = new JSONArray(json);
            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = jsonArray.getJSONObject(i);
                String ID = jsonObject.optString("BookingID");
                String mainPassengerName =
jsonObject.optString("MainPassengerName");
                String nic = jsonObject.optString("NIC");
                String userID = jsonObject.optString("userID");
                Log.d("Updateuser","userid2: " + userID);
                String trainNumber = jsonObject.optString("TrainNumber");
```

```
                String trainName = jsonObject.optString("TrainName");
                String departureStation =
jsonObject.optString("DepartureStation");
                String destinationStation =
jsonObject.optString("DestinationStation");
                int totalPassengers = jsonObject.optInt("TotalPassengers");
                String ticketClass = jsonObject.optString("TicketClass");
                String email = jsonObject.optString("Email");
                String phone = jsonObject.optString("ContactNumber");
                String reservationDate =
jsonObject.optString("FormattedReservationDate");
                String bookingDate =
jsonObject.optString("FormattedBookingDate");
                Log.d("ReservationApiClient", "Parsed reservations: " +
reservationList.size());

                Reservation reservation = new Reservation(ID, trainNumber,
trainName, userID, bookingDate,
                        reservationDate, totalPassengers, mainPassengerName,
phone, departureStation, destinationStation, email, nic, ticketClass);

                reservationList.add(reservation);
            }
        } catch (JSONException e) {
            e.printStackTrace();
            Log.e("ReservationApiClient", "Error parsing JSON: " +
e.getMessage());
        }

        return reservationList;
    }

    // Method to update a reservation via API
    public static void updateReservationInAPI(final Reservation reservation,
final OnReservationUpdatedListener listener) {
        AsyncTask<Void, Void, String> task = new AsyncTask<Void, Void, String>()
{
            @Override
            protected String doInBackground(Void... voids) {
                try {
                    URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/updateticketbooking/" +
reservation.getBookingID());
                    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
```

```
                    connection.setRequestMethod("PUT");
                    connection.setRequestProperty("Content-Type",
"application/json");

                    JSONObject requestBody = new JSONObject();
                    requestBody.put("MainPassengerName",
reservation.getMainPassengerName());
                    requestBody.put("NIC", reservation.getNIC());
                    requestBody.put("TrainName", reservation.getTrainName());
                    requestBody.put("DepartureStation",
reservation.getDepartureStation());
                    requestBody.put("DestinationStation",
reservation.getDestinationStation());
                    requestBody.put("TotalPassengers",
reservation.getTotalPassengers());
                    requestBody.put("TicketClass", reservation.getTicketClass());
                    requestBody.put("Email", reservation.getEmail());
                    requestBody.put("ContactNumber",
reservation.getContactNumber());
                    requestBody.put("ReservationDate",
reservation.getReservationDate());

                    connection.setDoOutput(true);
                    OutputStream os = connection.getOutputStream();
                    os.write(requestBody.toString().getBytes("UTF-8"));
                    os.close();

                    int responseCode = connection.getResponseCode();

                    if (responseCode == HttpURLConnection.HTTP_OK) {
                        return "Success";
                    } else {
                        return "Error";
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                }
            }

            @Override
            protected void onPostExecute(String response) {
                if (response != null && response.equals("Success")) {
                    listener.onReservationUpdated();
                } else {
```

```
                    listener.onError("Error updating reservation");
                }
            }
        };

        task.execute();
    }

    // Method to handle the response after updating a reservation
    public interface OnReservationUpdatedListener {
        void onReservationUpdated();
        void onError(String errorMessage);
    }
}
```

ReservationListAdapter.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.trainbooking_mobileapp.R;
import java.util.List;

public class ReservationListAdapter extends
RecyclerView.Adapter<ReservationListAdapter.ViewHolder> {

    private List<Reservation> reservationList;
    private LayoutInflater inflater;
    private Context context;
    private String userID;

    // Constructor
    public ReservationListAdapter(Context context, List<Reservation>
reservationList) {
```

```
        this.inflater = LayoutInflater.from(context);
        this.reservationList = reservationList;
        this.context = context;
        this.userID = userID;
    }

    // Set user ID
    public void setUserID(String userID) {
        this.userID = userID;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view = inflater.inflate(R.layout.reservation_list_item, parent,
false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Reservation reservation = reservationList.get(position);

        // Bind reservation data to views
        holder.reservationTrainNameTextView.setText(reservation.getTrainName());
        holder.reservationBookingStatusTextView.setText(reservation.getMainPassen
gerName());
        holder.reservationReservationDateTextView.setText(reservation.getReservat
ionDate());
        holder.reservationBookingDateTextView.setText(reservation.getBookingDate(
));

        // Set click listeners for buttons
        Button updateButton = holder.itemView.findViewById(R.id.updateButton);
        Button viewButton = holder.itemView.findViewById(R.id.viewButton);
        Button cancelButton = holder.itemView.findViewById(R.id.cancelButton);

        // Update Button Click Listener
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openUpdateReservationActivity(reservation);
            }
        });
```

```
        // View Button Click Listener
        viewButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openViewReservationActivity(reservation);
            }
        });

        // Cancel Button Click Listener
        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cancelReservation(reservation);
            }
        });
    }

    @Override
    public int getItemCount() {
        return reservationList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView reservationTrainNameTextView, reservationBookingStatusTextView,
reservationReservationDateTextView, reservationBookingDateTextView;

        // ViewHolder constructor
        public ViewHolder(@NonNull View itemView) {
            super(itemView);

            reservationTrainNameTextView =
itemView.findViewById(R.id.reservationTrainNameTextView);
            reservationBookingStatusTextView =
itemView.findViewById(R.id.reservationBookingStatusTextView);
            reservationReservationDateTextView =
itemView.findViewById(R.id.reservationReservationDateTextView);
            reservationBookingDateTextView =
itemView.findViewById(R.id.reservationBookingDateTextView);
        }
    }

    // Cancel Reservation
    private void cancelReservation(Reservation reservation) {
```

```
        ReservationApiClient.cancelReservationFromAPI(reservation, new
ReservationApiClient.OnReservationCanceledListener() {
            @Override
            public void onReservationCanceled() {
                reservationList.remove(reservation);
                notifyDataSetChanged();
                Toast.makeText(context, "Reservation canceled successfully",
Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onError(String errorMessage) {
                Toast.makeText(context, "You can only cancel reservations at
least 5 days before the your ticket booking date.", Toast.LENGTH_SHORT).show();
            }
        });
    }

    // Open Update Reservation Activity
    private void openUpdateReservationActivity(Reservation reservation) {
        Intent intent = new Intent(context, UpdateReservationActivity.class);
        intent.putExtra("reservation", reservation);
        intent.putExtra("userID", userID);
        ((Activity) context).startActivityForResult(intent, 1001);
    }

    // Open View Reservation Activity
    private void openViewReservationActivity(Reservation reservation) {
        Intent intent = new Intent(context, ReservationViewActivity.class);
        intent.putExtra("reservation", reservation);
        intent.putExtra("userID", userID);
        ((Activity) context).startActivityForResult(intent, 1001);
    }
}
```

CreateReservationActivity.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;
import com.google.gson.Gson;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.time.LocalDateTime;
import java.util.List;

public class CreateReservationActivity extends AppCompatActivity {

    private EditText editTextMainPassengerName;
    private EditText editTextNIC;
    private EditText editTextDepartureStation;
    private EditText editTextDestinationStation;
    private EditText editTextEmail;
    private EditText editTextPhone;
    private EditText editTextTotalPassengers;
    private TextView textviewReservationDate;
    private Toolbar toolbar;
    private Spinner ticketClassSpinner;
    private Spinner trainNameSpinner;
    private String userID;
    private ReservationApiClient trainBookingApiClient;

    // Method for checking if the provided email is valid
    private boolean isValidEmail(String email) {
```

```
        String emailPattern = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-
.]+$";
        return email.matches(emailPattern);
    }

    // Method for checking if the provided NIC is valid
    private boolean isValidNIC(String nic) {
        String nicPattern = "^\\d{12}$";
        return nic.matches(nicPattern);
    }

    // Method for checking if the provided contact number is valid
    private boolean isValidContactNumber(String contactNumber) {
        String contactNumberPattern = "^\\d{10}$";
        return contactNumber.matches(contactNumberPattern);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_rservation);

        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        setTitle("Add Reservation");

        getSupportActionBar().setDisplayHomeAsUpEnabled(false);

        editTextMainPassengerName = findViewById(R.id.editTextMainPassengerName);
        editTextNIC = findViewById(R.id.editTextNIC);
        trainNameSpinner = findViewById(R.id.trainNameSpinner);
        editTextDepartureStation = findViewById(R.id.editTextDepartureStation);
        editTextDestinationStation =
findViewById(R.id.editTextDestinationStation);
        editTextTotalPassengers = findViewById(R.id.editTextTotalPassengers);
        ticketClassSpinner = findViewById(R.id.ticketClassSpinner);
        editTextEmail = findViewById(R.id.editTextEmail);
        editTextPhone = findViewById(R.id.editTextPhone);
        textviewReservationDate = findViewById(R.id.editTextReservationDate);

        Button buttonSubmitReservation =
findViewById(R.id.buttonSubmitReservation);
        buttonSubmitReservation.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                createReservation();
            }
        });

        userID = getIntent().getStringExtra("userID");
        Log.d("ReservationActivity", "Received userID: " + userID);

        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button6 = findViewById(R.id.button6);
        ImageButton Button5 = findViewById(R.id.button5);
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CreateReservationActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);

            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CreateReservationActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);

            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CreateReservationActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });


        ImageButton calendarButton = findViewById(R.id.calendarButton);
        calendarButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                showDatePickerDialog();
            }
        });

        Spinner ticketClassSpinner = findViewById(R.id.ticketClassSpinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
                R.array.ticket_classes, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        ticketClassSpinner.setAdapter(adapter);


        trainBookingApiClient = new ReservationApiClient();

        Spinner trainNameSpinner = findViewById(R.id.trainNameSpinner);

        trainBookingApiClient.getTrains(new
ReservationApiClient.OnTrainNamesReceivedListener() {
            @Override
            public void onTrainNamesReceived(List<String> trainNames) {
                // Create an ArrayAdapter and set it to the spinner
                ArrayAdapter<String> adapter = new
ArrayAdapter<>(CreateReservationActivity.this,
android.R.layout.simple_spinner_item, trainNames);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_d
ropdown_item);
                trainNameSpinner.setAdapter(adapter);
            }
            @Override
            public void onError(String errorMessage) {
            }
        });

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
```

```
        } else if (id == android.R.id.home) {
            onBackPressed();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // Method for showing the date picker dialog
    private void showDatePickerDialog() {
        DatePickerDialog datePickerDialog = new DatePickerDialog(
                this,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                        // Handle the selected date
                        String formattedDate = String.format("%04d-%02d-%02d",
year, monthOfYear + 1, dayOfMonth);
                        textviewReservationDate.setText(formattedDate);
                    }
                },
                2023, 9, 17
        );
        datePickerDialog.show();
    }

    // Method for signing out
    private void signOut() {
        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }

    // Method for creating a reservation
    private void createReservation() {
        String mainPassengerName =
editTextMainPassengerName.getText().toString();
        String nic = editTextNIC.getText().toString();
        String userID = getIntent().getStringExtra("userID");
        String totalPassengersString =
editTextTotalPassengers.getText().toString();
        String departureStation = editTextDepartureStation.getText().toString();
        String destinationStation =
editTextDestinationStation.getText().toString();
```

```
        String bookingStatus = "Active";
        String email = editTextEmail.getText().toString();
        String phone = editTextPhone.getText().toString();
        String reservationDate = textviewReservationDate.getText().toString();
        LocalDateTime currentDateTime = LocalDateTime.now();
        String bookingDate = currentDateTime.toString();

        if (!isValidEmail(email)) {
            Toast.makeText(CreateReservationActivity.this, "Invalid email
format.", Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidNIC(nic)) {
            Toast.makeText(CreateReservationActivity.this, "Invalid NIC format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidContactNumber(phone)) {
            Toast.makeText(CreateReservationActivity.this, "Invalid contact
number format.", Toast.LENGTH_SHORT).show();
            return;
        }

        Log.d("ReservationActivity", "booking Date: " + bookingDate);

        String ticketClass = ticketClassSpinner.getSelectedItem().toString();

        int totalPassengers = Integer.parseInt(totalPassengersString);
        String trainName = trainNameSpinner.getSelectedItem().toString();

        Reservation reservation = new Reservation(null, "", trainName, userID,
bookingDate,
                reservationDate, totalPassengers, mainPassengerName, phone,
departureStation, destinationStation, email, nic, ticketClass);

        CreateReservationTask task = new CreateReservationTask(userID);
        task.execute(reservation);

        Log.d("ReservationActivity", "Reservation: " + reservation);
    }

    // AsyncTask class for creating a reservation in the background
```

```
    private class CreateReservationTask extends AsyncTask<Reservation, Void,
String> {
        private String userID;

        public CreateReservationTask(String userID) {
            this.userID = userID;
        }

        @Override
        protected String doInBackground(Reservation... reservations) {
            Reservation reservation = reservations[0];
            try {
                URL url = new URL("http://pasinduperera-001-
site1.atempurl.com/api/trainbooking/createticketbooking");
                HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                connection.setRequestMethod("POST");
                connection.setRequestProperty("Content-Type",
"application/json;charset=UTF-8");
                connection.setRequestProperty("Accept","application/json");
                connection.setDoOutput(true);

                reservation.setUserID(userID);

                Gson gson = new Gson();
                String jsonInputString = gson.toJson(reservation);
                Log.d("ReservationActivity", "jsonInputString: " +
jsonInputString);
                OutputStream os = connection.getOutputStream();
                os.write(jsonInputString.getBytes("utf-8"));

                BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream(), "utf-8"));
                StringBuilder response = new StringBuilder();
                String responseLine;
                while ((responseLine = br.readLine()) != null) {
                    response.append(responseLine.trim());
                }

                return response.toString();
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
        }
```

```
        @Override
        protected void onPostExecute(String response) {
            if (response != null) {
                Log.d("ReservationActivity", "Reservation created successfully.
Response: " + response);
                Toast.makeText(CreateReservationActivity.this, "Reservation
created successfully!", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(CreateReservationActivity.this,
ReservationDetailsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            } else {
                Log.e("ReservationActivity", "Error creating reservation.");
                Toast.makeText(CreateReservationActivity.this, "Reservation date
must be within 30 days from the your booking date.", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

ReservationDetailsActivity.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

import java.util.List;
```

```
public class ReservationDetailsActivity extends AppCompatActivity implements
ReservationApiClient.OnReservationDataReceivedListener {

    private RecyclerView recyclerView;
    private ReservationListAdapter reservationListAdapter;
    private ReservationApiClient reservationApiClient;
    private List<Reservation> reservationList;
    private Toolbar toolbar;
    private String userID;

    @SuppressLint("LongLogTag")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reservation_list);

        // Initialize the toolbar
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        // Set the title for this activity
        setTitle("Reservation History");

        // Disable the up button in the action bar
        getSupportActionBar().setDisplayHomeAsUpEnabled(false);

        // Find the RecyclerView for reservations
        recyclerView = findViewById(R.id.reservationRecyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        // Create an instance of ReservationApiClient
        reservationApiClient = new ReservationApiClient();

        // Get the user ID passed from the previous activity
        userID = getIntent().getStringExtra("userID");

        // Fetch reservations from the API
        reservationApiClient.getReservationsForUserFromAPI(userID, this);

        // Set click listeners for the ImageButtons
        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button6 = findViewById(R.id.button6);
        ImageButton Button5 = findViewById(R.id.button5);

        // Handle button clicks
```

```
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to the main activity
                Intent intent = new Intent(ReservationDetailsActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);

            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ReservationDetailsActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);

            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ReservationDetailsActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
        } else if (id == android.R.id.home) {
            // Handle the up button in the action bar
            onBackPressed();
            return true;
        }
```

```
        return super.onOptionsItemSelected(item);
    }

    private void signOut() {
        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }

    @Override
    public void onReservationDataReceived(List<Reservation> reservationList) {

        for (Reservation reservation : reservationList) {
        }

        reservationListAdapter = new ReservationListAdapter(this,
reservationList);
        reservationListAdapter.setUserID(userID); // Set the userID in the
adapter
        recyclerView.setAdapter(reservationListAdapter);
    }

    @Override
    public void onError(String errorMessage) {
        Log.e("ReservationListActivity", errorMessage);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == 1001 && resultCode == RESULT_OK) {
            Reservation updatedReservation = (Reservation)
data.getSerializableExtra("updatedReservation");
            String updatedUserID = data.getStringExtra("userID");

            int position = reservationList.indexOf(updatedReservation);
            if (position != -1) {
                reservationList.set(position, updatedReservation);
                reservationListAdapter.notifyItemChanged(position);
                reservationListAdapter.setUserID(updatedUserID);
            }
        }
    }
```

```
}
```

ReservationViewActivity.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

public class ReservationViewActivity extends AppCompatActivity {

    private TextView mainPassengerNameTextText, nicTextText, trainNameTextText,
departureStationTextText,
            destinationStationTextText, totalPassengersTextText,
ticketClassTextText, emailTextText, contactNumberTextText,
reservationDateTextText;
    private Reservation reservation;

    private Toolbar toolbar;
    private String userID;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        // Initialize the activity
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reservation_view);

        // Set up the toolbar
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        setTitle("Reservation Details");
```

```
        // Retrieve reservation data
        reservation = (Reservation)
getIntent().getSerializableExtra("reservation");

        // Initialize TextViews
        mainPassengerNameTextText = findViewById(R.id.mainPassengerNameTextText);
        nicTextText = findViewById(R.id.nicTextText);
        trainNameTextText = findViewById(R.id.trainNameTextText);
        departureStationTextText = findViewById(R.id.departureStationTextText);
        destinationStationTextText =
findViewById(R.id.destinationStationTextText);
        totalPassengersTextText = findViewById(R.id.totalPassengersTextText);
        ticketClassTextText = findViewById(R.id.ticketClassTextText);
        emailTextText = findViewById(R.id.emailTextText);
        contactNumberTextText = findViewById(R.id.contactNumberTextText);
        reservationDateTextText = findViewById(R.id.reservationDateTextText);

        // Populate TextViews with reservation data
        populateFields();

        // Set up buttons
        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button5 = findViewById(R.id.button5);
        ImageButton Button6 = findViewById(R.id.button6);
        userID = getIntent().getStringExtra("userID");

        // Set up button click listeners
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Handle button click
                Intent intent = new Intent(ReservationViewActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Handle button click
                Intent intent = new Intent(ReservationViewActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
```

```
                startActivity(intent);
            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Handle button click
                Intent intent = new Intent(ReservationViewActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle menu item selection
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
        } else if (id == android.R.id.home) {
            onBackPressed();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    // Handle sign out
    private void signOut() {
        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }

    // Populate TextViews with reservation data
    private void populateFields() {
        mainPassengerNameTextText.setText(reservation.getMainPassengerName());
        nicTextText.setText(reservation.getNIC());
        trainNameTextText.setText(reservation.getTrainName());
        departureStationTextText.setText(reservation.getDepartureStation());
        destinationStationTextText.setText(reservation.getDestinationStation());
```

```
        totalPassengersTextText.setText(String.valueOf(reservation.getTotalPassen
gers()));
        ticketClassTextText.setText(reservation.getTicketClass());
        emailTextText.setText(reservation.getEmail());
        contactNumberTextText.setText(reservation.getContactNumber());
        reservationDateTextText.setText(reservation.getReservationDate());
    }
}
```

UpdateReservationActivity.java

```
package com.example.trainbooking_mobileapp.ReservationManagement;

import android.annotation.SuppressLint;
import android.app.DatePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.AboutUsActivity;
import com.example.trainbooking_mobileapp.MainActivity;
import com.example.trainbooking_mobileapp.R;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

import java.util.List;

public class UpdateReservationActivity extends AppCompatActivity {

    private EditText mainPassengerNameEditText, nicEditText,
departureStationEditText,
            destinationStationEditText, emailEditText, contactNumberEditText,
totalPassengersEditText;
```

```
    private TextView reservationDateTextView;
    private Button updateButton;
    private Reservation reservation;

    private Toolbar toolbar;
    private String userID;

    private Spinner ticketClassSpinner;

    private ReservationApiClient trainBookingApiClient;

    @SuppressLint("LongLogTag")
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        // Initialize the activity
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_reservation);
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        setTitle("Update Reservation");
        userID = getIntent().getStringExtra("userID");
        reservation = (Reservation)
getIntent().getSerializableExtra("reservation");
        userID = getIntent().getStringExtra("userID");
        mainPassengerNameEditText = findViewById(R.id.mainPassengerNameEditText);
        nicEditText = findViewById(R.id.nicEditText);
        totalPassengersEditText = findViewById(R.id.totalPassengersEditText);
        departureStationEditText = findViewById(R.id.departureStationEditText);
        destinationStationEditText =
findViewById(R.id.destinationStationEditText);
        ticketClassSpinner = findViewById(R.id.ticketClassSpinner);
        emailEditText = findViewById(R.id.emailEditText);
        contactNumberEditText = findViewById(R.id.contactNumberEditText);
        reservationDateTextView = findViewById(R.id.editTextReservationDate);
        updateButton = findViewById(R.id.updateButton);

        populateFields();

        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateReservation();
            }
        });
```

```
        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button5 = findViewById(R.id.button5);
        ImageButton Button6 = findViewById(R.id.button6);
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateReservationActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateReservationActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(UpdateReservationActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        reservationDateTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDatePickerDialog();
            }
        });

        Spinner ticketClassSpinner = findViewById(R.id.ticketClassSpinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
                R.array.ticket_classes, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
```

```
        ticketClassSpinner.setAdapter(adapter);

        trainBookingApiClient = new ReservationApiClient();

        Spinner trainNameSpinner = findViewById(R.id.trainNameSpinner);

        trainBookingApiClient.getTrains(new
ReservationApiClient.OnTrainNamesReceivedListener() {
            @Override
            public void onTrainNamesReceived(List<String> trainNames) {
                ArrayAdapter<String> adapter = new
ArrayAdapter<>(UpdateReservationActivity.this,
android.R.layout.simple_spinner_item, trainNames);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_d
ropdown_item);
                trainNameSpinner.setAdapter(adapter);
            }

            @Override
            public void onError(String errorMessage) {
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle menu item selection
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
        } else if (id == android.R.id.home) {
            onBackPressed();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    private DatePickerDialog.OnDateSetListener dateSetListener = new
DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear, int
dayOfMonth) {
```

```
            String formattedDate = String.format("%04d-%02d-%02d", year,
monthOfYear + 1, dayOfMonth);
            reservationDateTextView.setText(formattedDate);
        }
    };

    private void showDatePickerDialog() {
        // Show date picker dialog
        DatePickerDialog datePickerDialog = new DatePickerDialog(
                this,
                dateSetListener,
                2023, 9, 17
        );
        datePickerDialog.show();
    }

    private void signOut() {
        Intent intentone = new Intent(this, SignInActivity.class);
        startActivity(intentone);
        finish();
    }

    private void populateFields() {
        mainPassengerNameEditText.setText(reservation.getMainPassengerName());
        nicEditText.setText(reservation.getNIC());
        departureStationEditText.setText(reservation.getDepartureStation());
        totalPassengersEditText.setText(String.valueOf(reservation.getTotalPassen
gers()));
        destinationStationEditText.setText(reservation.getDestinationStation());
        emailEditText.setText(reservation.getEmail());
        contactNumberEditText.setText(reservation.getContactNumber());
        reservationDateTextView.setText(reservation.getReservationDate());
    }

    private void updateReservation() {
        // Handle updating reservation
        String mainPassengerName =
mainPassengerNameEditText.getText().toString();
        String nic = nicEditText.getText().toString();
        String departureStation = departureStationEditText.getText().toString();
        String totalPassengersConvert =
totalPassengersEditText.getText().toString();
        String destinationStation =
destinationStationEditText.getText().toString();
        String email = emailEditText.getText().toString();
```

```
        String contactNumber = contactNumberEditText.getText().toString();
        String reservationDate = reservationDateTextView.getText().toString();

        int totalPassengers = Integer.parseInt(totalPassengersConvert);

        if (!isValidEmail(email)) {
            // Check if email is valid
            Toast.makeText(UpdateReservationActivity.this, "Invalid email
format.", Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidNIC(nic)) {
            // Check if NIC is valid
            Toast.makeText(UpdateReservationActivity.this, "Invalid NIC format.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!isValidContactNumber(contactNumber)) {
            // Check if contact number is valid
            Toast.makeText(UpdateReservationActivity.this, "Invalid contact
number format.", Toast.LENGTH_SHORT).show();
            return;
        }

        Reservation updatedReservation = new
Reservation(reservation.getBookingID(), reservation.getTrainNumber(),
reservation.getTrainName(), userID, reservation.getBookingDate(),
                reservationDate, totalPassengers, mainPassengerName,
contactNumber, departureStation, destinationStation, email, nic,
reservation.getTicketClass());

        ReservationApiClient.updateReservationInAPI(updatedReservation, new
ReservationApiClient.OnReservationUpdatedListener() {
            @Override
            public void onReservationUpdated() {
                Intent intent = new Intent(UpdateReservationActivity.this,
ReservationDetailsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
                Toast.makeText(UpdateReservationActivity.this, "Reservation
updated successfully", Toast.LENGTH_SHORT).show();
                finish();
            }
```

```
            @Override
            public void onError(String errorMessage) {
                Toast.makeText(UpdateReservationActivity.this, "You can only
update reservations at least 5 days before the your ticket booking date",
Toast.LENGTH_SHORT).show();
            }
        });
    }

    private boolean isValidEmail(String email) {
        String emailPattern = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-
.]+$";
        return email.matches(emailPattern);
    }

    private boolean isValidNIC(String nic) {
        String nicPattern = "^\\d{12}$";
        return nic.matches(nicPattern);
    }

    private boolean isValidContactNumber(String contactNumber) {
        String contactNumberPattern = "^\\d{10}$";
        return contactNumber.matches(contactNumberPattern);
    }
}
```

Activity_create_reservation.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background"
    tools:context=".ReservationManagement.CreateReservationActivity">

    <include layout="@layout/top_bar" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginBottom="16dp">
```

```
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="16dp"
            android:gravity="center_horizontal">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:background="@drawable/rounded_border"
                android:layout_marginTop="16dp"
                android:padding="8dp">

                <TextView
                    android:id="@+id/TrainNameLabel"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginRight="16dp"
                    android:text="Train Name"
                    android:textColor="@color/white"
                    android:textSize="16sp" />

                <Spinner
                    android:id="@+id/trainNameSpinner"
                    android:layout_width="128dp"
                    android:layout_height="match_parent"
                    android:layout_marginLeft="87dp"
                    android:textColor="@color/white"
                    android:textColorHint="@color/white"
                    android:background="@color/white" />

            </LinearLayout>

<!--            <Spinner-->
<!--                android:id="@+id/trainNameSpinner"-->
<!--                android:layout_width="wrap_content"-->
<!--                android:layout_height="wrap_content" />-->

    <EditText
        android:id="@+id/editTextMainPassengerName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:hint="Traveler Name"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

    <EditText
        android:id="@+id/editTextNIC"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="NIC"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:padding="8dp"
                android:background="@drawable/rounded_border">

                <TextView
                    android:id="@+id/editTextReservationDate"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:textSize="18dp"
                    android:hint="Reservation Date"
                    android:inputType="date"
                    android:textColor="@color/white"
                    android:textColorHint="@color/white"/>

                <ImageButton
                    android:id="@+id/calendarButton"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@drawable/ic_baseline_calendar_month_24"
                    android:contentDescription="Calendar Icon"
                    android:layout_alignParentRight="true"
                    android:background="@android:color/transparent"/>

            </RelativeLayout>
```

```
<!--            <EditText-->
<!--        android:id="@+id/editTextTrainNumber"-->
<!--        android:layout_width="match_parent"-->
<!--        android:layout_height="wrap_content"-->
<!--        android:hint="Train Number"-->
<!--        android:textColor="@color/white"-->
<!--        android:textColorHint="@color/white"-->
<!--        android:layout_marginTop="16dp"-->
<!--        android:padding="8dp"-->
<!--        android:background="@drawable/rounded_border"/>-->

<!--            <EditText-->
<!--                android:id="@+id/editTextTrainName"-->
<!--                android:layout_width="match_parent"-->
<!--                android:layout_height="wrap_content"-->
<!--                android:hint="Train Name"-->
<!--                android:textColor="@color/white"-->
<!--                android:textColorHint="@color/white"-->
<!--                android:layout_marginTop="16dp"-->
<!--                android:padding="8dp"-->
<!--                android:background="@drawable/rounded_border"/>-->

    <EditText
        android:id="@+id/editTextDepartureStation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Departure Station"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

    <EditText
        android:id="@+id/editTextDestinationStation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Destination Station"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>
```

```
    <EditText
        android:id="@+id/editTextTotalPassengers"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Total Passengers"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:layout_marginTop="16dp"
        android:padding="8dp">

        <TextView
            android:id="@+id/ticketClassLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="16dp"
            android:text="Ticket Class"
            android:textColor="@color/white"
            android:textSize="16sp" />

        <Spinner
            android:id="@+id/ticketClassSpinner"
            android:layout_width="470px"
            android:layout_height="match_parent"
            android:layout_marginLeft="78dp"
            android:textColor="@color/white"
            android:textColorHint="@color/white"
            android:background="#B2FFFFFF"
            style="@style/SpinnerDropDownItem" />

    </LinearLayout>

    <EditText
android:id="@+id/editTextEmail"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:inputType="textEmailAddress"
```

```
        android:hint="Email"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

    <EditText
        android:id="@+id/editTextPhone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="phone"
        android:hint="Contact Number"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:layout_marginTop="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border"/>

    <Button
        android:id="@+id/buttonSubmitReservation"
        android:layout_width="170dp"
        android:layout_height="wrap_content"
        android:text="Create Reservation"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:background="@drawable/rounded_button_bg"
        android:textColor="#FFFFFF"/>

        </LinearLayout>
    </ScrollView>

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:background="#00284d"
        android:elevation="8dp">

        <ImageButton
```

```
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_home_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button5"
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</LinearLayout>
```

Activity_update_reservation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".ReservationManagement.UpdateReservationActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
android:orientation="vertical">

<include layout="@layout/top_bar" />

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:layout_weight="1">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:background="@drawable/rounded_border"
            android:layout_marginBottom="16dp"
            android:padding="8dp">

            <TextView
                android:id="@+id/TrainNameLabel"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginRight="16dp"
                android:text="Train Name"
                android:textColor="@color/white"
                android:textSize="16sp" />

            <Spinner
                android:id="@+id/trainNameSpinner"
                android:layout_width="128dp"
                android:layout_height="match_parent"
                android:layout_marginLeft="87dp"
                android:textColor="@color/white"
                android:textColorHint="@color/white"
                android:background="@color/white" />

        </LinearLayout>

        <EditText
            android:id="@+id/mainPassengerNameEditText"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:background="@drawable/rounded_border"
        android:hint="Main Passenger Name"
        android:padding="8dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />

    <EditText
        android:id="@+id/nicEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:background="@drawable/rounded_border"
        android:hint="NIC"
        android:padding="8dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:padding="8dp"
        android:background="@drawable/rounded_border">

        <TextView
            android:id="@+id/editTextReservationDate"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="18dp"
            android:hint="Reservation Date"
            android:inputType="date"
            android:textColor="@color/white"
            android:textColorHint="@color/white"/>

        <ImageButton
            android:id="@+id/calendarButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_baseline_calendar_month_24"
            android:contentDescription="Calendar Icon"
            android:layout_alignParentRight="true"
            android:background="@android:color/transparent"/>
```

```
</RelativeLayout>

<EditText
    android:id="@+id/departureStationEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:background="@drawable/rounded_border"
    android:hint="Departure Station"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white" />

<EditText
    android:id="@+id/destinationStationEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:background="@drawable/rounded_border"
    android:hint="Destination Station"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white" />

<EditText
    android:id="@+id/emailEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:background="@drawable/rounded_border"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:padding="8dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white" />

<EditText
    android:id="@+id/totalPassengersEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:background="@drawable/rounded_border"
    android:hint="Total Passengers"
    android:inputType="textEmailAddress"
```

```
        android:padding="8dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:layout_marginBottom="16dp"
        android:padding="8dp">

        <TextView
            android:id="@+id/ticketClassLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="16dp"
            android:text="Ticket Class"
            android:textColor="@color/white"
            android:textSize="16sp" />

        <Spinner
            android:id="@+id/ticketClassSpinner"
            android:layout_width="470px"
            android:layout_height="match_parent"
            android:layout_marginLeft="78dp"
            android:textColor="@color/white"
            android:textColorHint="@color/white"
            android:background="#B2FFFFFF"
            style="@style/SpinnerDropDownItem" />

    </LinearLayout>

    <EditText
        android:id="@+id/contactNumberEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_border"
        android:hint="Contact Number"
        android:inputType="phone"
        android:padding="8dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white" />

    <Button
```

```
            android:id="@+id/updateButton"
            android:layout_width="187dp"
            android:layout_height="wrap_content"
            android:background="@drawable/rounded_button_bg"
            android:layout_gravity="center"
            android:layout_marginTop="16dp"
            android:padding="12dp"
            android:text="Update Reservation"
            android:textColor="#FFFFFF"
            android:layout_marginBottom="16dp"
            />

    </LinearLayout>
</ScrollView>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bottom_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_alignParentBottom="true"
    android:background="#00284d"
    android:elevation="8dp">

    <ImageButton
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="47dp"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_home_24"
        android:tint="#ffff" />

    <ImageButton
        android:id="@+id/button5"
        android:layout_width="47dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#00284d"
        android:src="@drawable/ic_baseline_account_circle_24"
        android:tint="#ffff" />
```

```
            <ImageButton
                android:id="@+id/button6"
                android:layout_width="0dp"
                android:layout_height="47dp"
                android:layout_weight="1"
                android:background="#00284d"
                android:src="@drawable/ic_baseline_groups_24"
                android:tint="#ffff" />

        </LinearLayout>

    </LinearLayout>
</LinearLayout>
```

Activity_reservation_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background"
    tools:context=".ReservationManagement.ReservationViewActivity">

    <include layout="@layout/top_bar" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="#B3DAFF"
        android:layout_marginBottom="16dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center_horizontal">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Main Passenger Name:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"
        />

        <TextView
            android:id="@+id/mainPassengerNameTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="NIC:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/nicTextText"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reservation Date:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/reservationDateTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Train Name:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/trainNameTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Departure Station:"
            android:layout_marginRight="16dp"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

        <TextView
            android:id="@+id/departureStationTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Destination Station:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/destinationStationTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Total Passengers:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/totalPassengersTextText"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Email:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/emailTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Contact Number:"
            android:layout_marginRight="16dp"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

        <TextView
            android:id="@+id/contactNumberTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@drawable/rounded_border"
        android:gravity="center_horizontal"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Ticket Class:"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"
            android:layout_marginRight="16dp"/>

        <TextView
            android:id="@+id/ticketClassTextText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textColor="@color/white"
            android:layout_marginTop="16dp"/>

    </LinearLayout>

</LinearLayout>
```

```
    </ScrollView>

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:background="#00284d"
        android:elevation="16dp">

        <ImageButton
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_home_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button5"
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</LinearLayout>
```

Activity_reservation_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background"
    tools:context=".ReservationManagement.ReservationDetailsActivity">

    <include layout="@layout/top_bar"
        android:padding="16dp"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/reservationRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:padding="16dp"
        android:clipToPadding="false"
        android:scrollbars="vertical"
        tools:listitem="@layout/train_list_item"
        android:layout_marginBottom="8dp"/>

    <!-- Add BottomNavigationView here -->
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:background="#00284d"
        android:elevation="8dp">

        <ImageButton
            android:id="@+id/button1"
            android:layout_width="0dp"
```

```
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_home_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button5"
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</androidx.appcompat.widget.LinearLayoutCompat>
```

Reservation_list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="8dp"
        android:gravity="center">
```

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/TrainIDLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Train Name :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:textStyle="bold"
        android:gravity="center_vertical"
        />

    <TextView
        android:id="@+id/reservationTrainNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
    android:padding="8dp"
    android:gravity="center_vertical"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="8dp"
    />

</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">
```

```
    <TextView
        android:id="@+id/bookingIDLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Passenger Name :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:textStyle="bold"
        android:gravity="center_vertical"
        />



    <TextView
        android:id="@+id/reservationBookingStatusTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:gravity="center_vertical"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="8dp" />

</LinearLayout>

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_border"
        android:orientation="horizontal"
        android:padding="8dp"
        android:gravity="center_horizontal"
        android:layout_margin="4dp">

        <TextView
            android:id="@+id/bookingDateLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Booking Date :"
            android:textSize="16sp"
            android:textColor="@color/white"
```

```
            android:padding="8dp"
            android:textStyle="bold"
            android:gravity="center_vertical"
            />



        <TextView
            android:id="@+id/reservationBookingDateTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="@color/white"
            android:padding="8dp"
            android:gravity="center_vertical"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="8dp" />

    </LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rounded_border"
    android:orientation="horizontal"
    android:padding="8dp"
    android:gravity="center_horizontal"
    android:layout_margin="4dp">

    <TextView
        android:id="@+id/reservationDateLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reservation Date :"
        android:textSize="16sp"
        android:textColor="@color/white"
        android:padding="8dp"
        android:textStyle="bold"
        android:gravity="center_vertical"
        />

    <TextView
        android:id="@+id/reservationReservationDateTextView"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="@color/white"
            android:padding="8dp"
            android:gravity="center_vertical"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="8dp" />

    </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center"
            android:layout_marginTop="8dp">

            <Button
                android:id="@+id/updateButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFFFFF"
                android:text="Update"
                android:layout_margin="4dp"
                android:background="@drawable/rounded_border_update"/>

            <Button
                android:id="@+id/viewButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFFFFF"
                android:text="View"
                android:layout_margin="4dp"
                android:background="@drawable/rounded_border_details"/>

            <Button
                android:id="@+id/cancelButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#FFFFFF"
                android:text="Cancel"
                android:layout_margin="4dp"
                android:background="@drawable/rounded_border_delete"/>

        </LinearLayout>
```

```
    </LinearLayout>

    <View
        android:layout_width="wrap_content"
        android:layout_height="1dp"
        android:background="#4CAF50" />

</RelativeLayout>
```

AboutUsActivity.java

```
package com.example.trainbooking_mobileapp;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import com.example.trainbooking_mobileapp.UserManagement.UserProfileActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

public class AboutUsActivity extends AppCompatActivity {

    private Toolbar toolbar;
    private String userID;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about_us);

        // Set up the toolbar
        toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        setTitle("About Us");

        getSupportActionBar().setDisplayHomeAsUpEnabled(false);

        // Initialize buttons
        ImageButton Button1 = findViewById(R.id.button1);
        ImageButton Button5 = findViewById(R.id.button5);
```

```
        ImageButton Button6 = findViewById(R.id.button6);

        // Get the user ID from the intent
        userID = getIntent().getStringExtra("userID");

        // Set up click listeners for the buttons
        Button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(AboutUsActivity.this,
MainActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(AboutUsActivity.this,
UserProfileActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });

        Button6.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(AboutUsActivity.this,
AboutUsActivity.class);
                intent.putExtra("userID", userID);
                startActivity(intent);
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_sign_out) {
            signOut();
            return true;
        } else if (id == android.R.id.home) {
```

```
            onBackPressed();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    private void signOut() {
        Intent intent = new Intent(this, SignInActivity.class);
        startActivity(intent);
        finish();
    }
}
```

Activity_about_us.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background"
    tools:context=".AboutUsActivity">

    <include layout="@layout/top_bar" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="#B3DAFF"
        android:layout_marginBottom="16dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="center_horizontal">

            <ImageView
                android:id="@+id/logoImageView"
                android:layout_width="160dp"
```

```
                android:layout_height="160dp"
                android:layout_marginTop="16dp"
                android:src="@drawable/logo"
                android:layout_gravity="center"
                android:layout_marginBottom="16dp"/>

            <TextView
                android:id="@+id/aboutUsDescription"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:layout_centerHorizontal="true"
                android:textColor="@color/white"
                android:background="#8B00284D"
                android:text="We are a team of four undergraduate students in our
fourth year, specializing in software engineering. Our project, the Train Ticket
Booking System, aims to provide a seamless and user-friendly experience for
booking train tickets. We have leveraged modern technologies and best practices
to develop this system, ensuring reliability and efficiency in ticket booking and
management."
                android:layout_gravity="center"
                android:textSize="21sp"/>

        </LinearLayout>

    </ScrollView>

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/bottom_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:background="#00284d"
        android:elevation="16dp">

        <ImageButton
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
```

```
            android:src="@drawable/ic_baseline_home_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button5"
            android:layout_width="47dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_account_circle_24"
            android:tint="#ffff" />

        <ImageButton
            android:id="@+id/button6"
            android:layout_width="0dp"
            android:layout_height="47dp"
            android:layout_weight="1"
            android:background="#00284d"
            android:src="@drawable/ic_baseline_groups_24"
            android:tint="#ffff" />

    </LinearLayout>

</LinearLayout>
```

SplashScreenActivity.java

```
package com.example.trainbooking_mobileapp;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import com.example.trainbooking_mobileapp.UserManagement.SignInActivity;

public class SplashScreenActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);
```

```
        // Use a Handler to delay the transition to SignInActivity
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                // Create an Intent to navigate to SignInActivity
                Intent intent = new Intent(SplashScreenActivity.this,
SignInActivity.class);
                // Start the SignInActivity
                startActivity(intent);
                // Finish the SplashScreenActivity to prevent going back to it
                finish();
            }
        }, 2000); // Delay for 2000 milliseconds (2 seconds)
    }
}
```

Activity_splash_scrren.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:background="#00284d"
    tools:context=".SplashScreenActivity">

    <ImageView
        android:layout_width="301dp"
        android:layout_height="280dp"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:contentDescription="@string/app_name"
        android:src="@drawable/logo" />

</LinearLayout>
```

Colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="green">#17831C</color>
    <color name="red">#DF0B1D</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="blue">#03A9F4</color>
    <color name="orange">#FF5722</color>
    <style name="SpinnerDropDownItem">
        <item name="android:textColor">@color/white</item>
    </style>
    <style name="AppTheme.Toolbar" parent="Widget.AppCompat.Toolbar">
        <item name="android:background">#00284d</item>
        <item name="android:titleTextColor">#FFFFFF</item>
        <item name="android:subtitleTextColor">#FFFFFF</item>
    </style>
</resources>
```

Strings.xml

```xml
<resources>
    <string name="app_name">Train Journey</string>
    <string-array name="gender_array">
        <item>Gender</item>
        <item>Male</item>
        <item>Female</item>
    </string-array>
    <string-array name="ticket_classes">
        <item>First Class</item>
        <item>Second Class</item>
        <item>Third Class</item>
    </string-array>
    <string name="open_drawer">Open Drawer</string>
    <string name="close_drawer">Close Drawer</string>
    <string name="title_activity_about_us">AboutUsActivity</string>
    <!-- Strings used for fragments for navigation -->
    <string name="first_fragment_label">First Fragment</string>
    <string name="second_fragment_label">Second Fragment</string>
    <string name="next">Next</string>
```

```
    <string name="previous">Previous</string>

    <string name="hello_first_fragment">Hello first fragment</string>
    <string name="hello_second_fragment">Hello second fragment. Arg:
%1$s</string>

</resources>
```

8 Challenges

- Error handling

As a developer, we will frequently come across scenarios in which the ability to recognize and correct errors is essential. We must find errors in front-end web applications and mobile applications. We use the console.log() method in JavaScript for web development and log.d() for Android Studio mobile development to locate and repair these errors.

- Difficult to connect android studio emulator with localhost

we've faced while creating our Android application is setting up our Android Studio emulator to connect to a local server. To remedy this, a change is necessary. The goal now is to connect to a live server hosted on myasp.net, rather than depending on a local server. This change reflects how apps communicate with distant servers in real-world scenarios. Configuring the app to interact with the live server, making sure the internet rights are required, and putting security measures in place to ensure secure connection are the main tasks. Verifying smooth interaction with the live server requires testing on real Android devices.

- MongoDB connection

In our project, we came across a new difficulty. connecting an ASP.NET web service with a MongoDB database. To facilitate effective data retrieval and storage, we must create a smooth connection between the web service and the MongoDB database. The solution to this is to use the MongoDB BSON library and driver, among other necessary parts. The web service and the MongoDB database can communicate much more easily thanks to these features. The driver acts as a bridge for communication, and the BSON library makes sure that data is serialized and

deserialized correctly. Our goal is to meticulously integrate these components, considering aspects such as version compatibility, error management, and authentication.

- Environment setup

We had a major environment setup difficulty during our assignment. This challenge was centered around version mismatches and compatibility problems that surfaced during the configuration of our development environments. Making sure all the parts interacted well became crucial because the online application and mobile app were both written in Java. It was immediately clear that there were discrepancies in the way that various libraries, frameworks, and tools were interacting. Our development workflow was hampered by frustrating compatibility issues resulting from this difference. It took a great deal of time and effort to debug and resolve these difficulties.

There were times when we had to find and install particular library versions or update parts, so they were compatible. To keep everything in sync, this approach required careful version control and was quite complex. To stay up to date on the most recent developments and guarantee consistent development procedures, it also required efficient communication within the development team. Resolving compatibility problems and version mismatches was a significant challenge that we conquered with much care, teamwork, and a resolute dedication to providing a reliable and useful ticket reservation system.

- Emulator issues

Throughout production, we had a few noteworthy difficulties, the most notable of them being emulator problems. We used Java to construct our mobile application, and emulators were essential to its development. Although these tools are necessary to test and optimize our app's functionality across a range of mobile devices, they also brought with them a new set of challenges.

Our challenge was configuring and setting up the emulators. It took a lot of time to make sure our emulators faithfully reproduced the intended device and operating system. The setup process became more complex due to compatibility concerns between different versions of the emulator and the requirements of the mobile app. In addition, there were times when our performance was slow. Although helpful for testing, the emulators frequently performed slower than real devices, which slowed down our development process. It was essential to locate and fix these performance bottlenecks if we were to continue being productive.

Additionally, we found emulator-specific issues that did not exist on actual hardware. Because of this, we had to set aside more time for debugging and making the necessary modifications to guarantee that our mobile app operated flawlessly on a variety of devices. Overcoming these obstacles allowed us to obtain priceless expertise in emulator management and problem-solving, which in turn helped our ticket reservation system project succeed.

9 References

[1] "ASP Tutorial," *W3schools.com*, 2016. https://www.w3schools.com/asp/

[2] Microsoft, "ASP.NET | Open-source web framework for .NET," *Microsoft*. https://dotnet.microsoft.com/en-us/apps/aspnet

[3] Android Studio, "Download Android Studio and SDK tools," *Android Developers*, 2019. https://developer.android.com/studio

[4] Meta Open Source, "React," *react.dev*, 2023. https://react.dev/

[5] "React Tutorial," *www.w3schools.com*. https://www.w3schools.com/react/

[6] "React with .NET Web API – Basic App Tutorial," *www.youtube.com*. https://youtu.be/4RKuyp_bOhY?si=7R6ojgPyw5Qhs8HP (accessed Oct. 13, 2023).

[7] "myASP.NET," *www.myasp.net*. https://www.myasp.net/

[8] MongoDB, "The most popular database for modern apps," *MongoDB*, 2019. https://www.mongodb.com/