Bus reservation Documentation

Team Lead: Anjela Vinodini

DB Lead: S. Rohit

UI Lead: Manoj Kumar. V

Controller Lead: Arfath Pathan

Technology used:

- ASP.NET Core MVC
- Microsoft SQL server

Objective:

Online bus reservation system is a project which provides a portal for bus ticket reservation. This application allows customers to book bus tickets from anywhere and anytime. The customer can easily book their tickets and cancel tickets at his comfort. The customer can view all the details of the seats, bus, and route along with picture. The customer can also view and download the details of the journey and journey timings. This will give customer an overview of the journey and will be a hassle free task of doing bus reservation at fingertips.

Modules:

1. Customer Module
2. Admin Module

Customer Module:

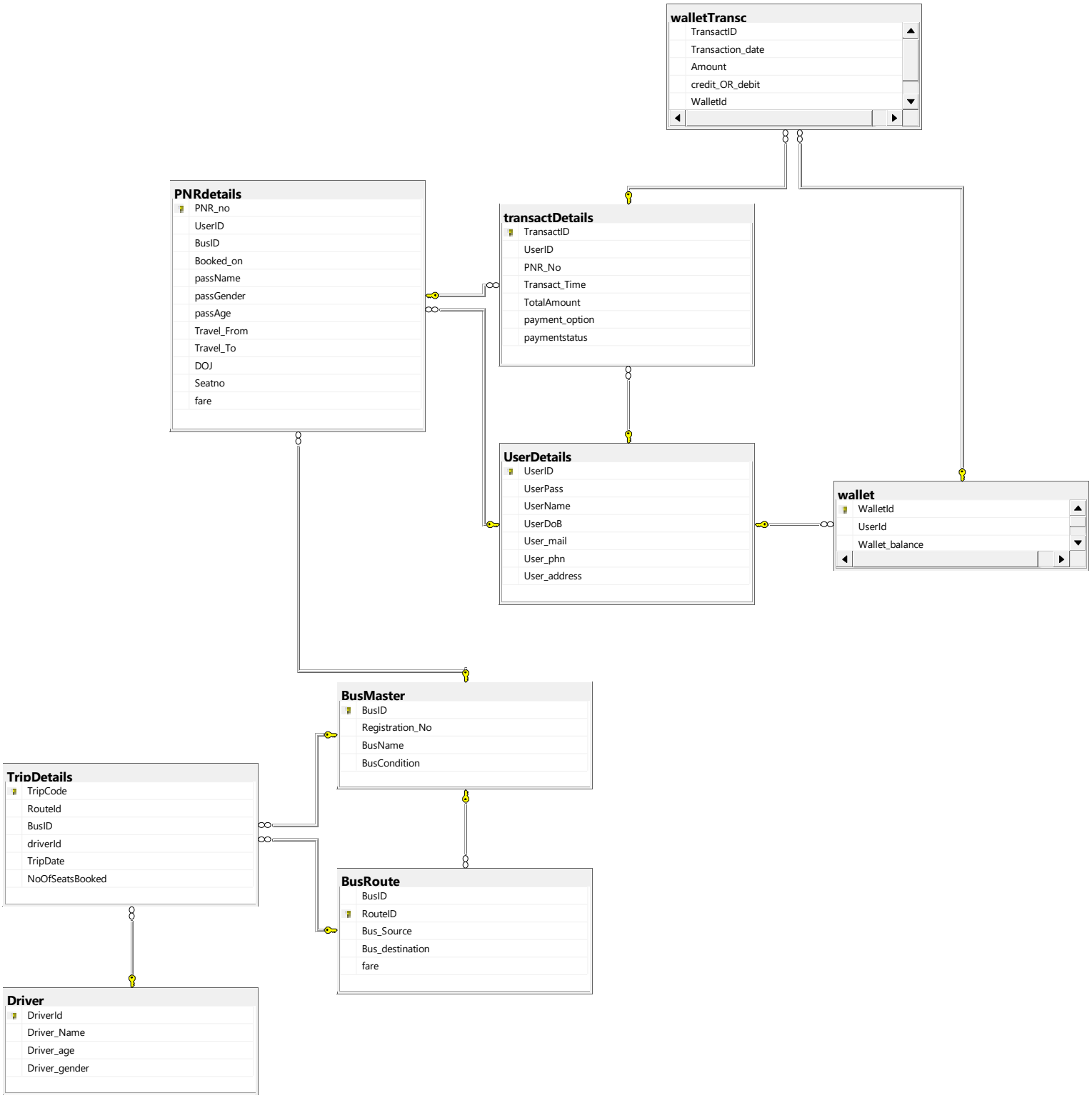
1. Registration:
   Customer has to register him/herself by entering the required input in the registration form, the registration form validation is done through the model validation. Hence if the user forgot to enter the required fields, the customer gets a acknowledge message to enter the field skipped. After successful validation all the input fields will be inserted to the USER DETAILS table in Data Base.

2. LOGIN:
   Login validation done by validating the Email and password used while registration.

Highlights:

- **User Session**
- **Ticket Download**

Database Design:



Search Buses Module:

Search Bus has the source and destination where the customer wish to make the reservation on specific date, and number of seats the customer would like to reserve.

After successful search, results of the available buses will be displayed

Code snippet:

Displaying Bus

```
public IActionResult DisplayBus(Search s)
        {
            List<DisplayBus> bus = new List<DisplayBus>();
            bus = (from b in db.BusRoutes
                           join m in db.BusMasters on b.BusId equals m.BusId
                           where b.BusDestination == s.Destination && b.BusSource ==
  s.Source && m.BusStatus==true
                           select new DisplayBus
                           {
                               BusId = b.BusId,
                               RouteId = b.RouteId,
                               BusName = m.BusName,
                               BusCondition = m.BusCondition,
                               BusSource = b.BusSource,
                               BusDestination = b.BusDestination,
                               Fare=b.Fare
                           }).ToList();
            if(bus!=null && bus.Count>0)
            {
                foreach(var item in bus)
                {
                    item.Doj = s.DateOfJourney;
                }
            }
            ViewBag.Uid = HttpContext.Session.GetString("Uid");
            TempData["Doj"] = s.DateOfJourney;
            TempData["Src"] = s.Source;
            TempData["Des"] = s.Destination;
            TempData["noOfSeats"] = s.TotalSeats;
            return View(bus);
        }
```

Booking Module:

In booking module the passenger details form accepts the passenger details from the user according to the number of seats selected, and populates the customer input in the repeating fields.

Code Snippet:

Booking

```
public IActionResult Book(int id)
      {
          BookingDetails p = new BookingDetails();
          p.PassengerDetails = new List<PassengerDetails>();
          p.BusId = id;
          p.BookedOn = DateTime.Now;
          p.Doj = Convert.ToDateTime(TempData["doj"]);
          TempData.Keep("doj");
          p.TravelFrom = Convert.ToString(TempData["Src"]);
          TempData.Keep("Src");
          p.TravelTo = Convert.ToString(TempData["Des"]);
          TempData.Keep("Des");
          p.UserId = Convert.ToString(TempData["id"]);
          TempData.Keep("id");
          var fare = (from f in db.BusRoutes
                      where f.BusId == id
                      select f.Fare).SingleOrDefault();
          p.Fare = Convert.ToDouble(fare);
          var noOfseats = TempData["noOfSeats"] != null ?
Convert.ToInt32(TempData["noOfSeats"]) : 0;
          TempData.Keep("noOfSeats");
          for (int i = 1; i <= noOfseats; i++)
          {
              var passengerdetail = new PassengerDetails();
              passengerdetail.Seatno = "A" + i;
              p.PassengerDetails.Add(passengerdetail);
          }


          ViewBag.Uid = HttpContext.Session.GetString("Uid");

          return View(p);
      }
```

Payment Module:

After accepting the passenger details user/customer will be re-directed to the payment page and the amount is calculated according the number of seats selected in the total fare. Payment window accepts cards which is validated and accepting the payment from user after successful payment user will be re-directed to the bookings where user can find all his bookings.

Payment

```
public IActionResult Payment(int id)
        {
            ViewBag.Uid = HttpContext.Session.GetString("Uid");
            TransactDetail t = new TransactDetail();
            t.TransactTime = DateTime.Now;
            t.UserId = Convert.ToString(TempData["id"]);
            TempData.Keep("id");
            var userid = (from u in db.UserDetails
                          where u.UserMail == t.UserId
                          select u.UserId).SingleOrDefault();

            var pnr = (from p in db.Pnrdetails
                       where p.BusId == id && p.UserId == userid
                       orderby p.BookedOn descending
                       select p.PnrNo).FirstOrDefault();
            t.PnrNo = pnr;
            var fare = (from f in db.BusRoutes
                        where f.BusId == id
                        select f.Fare).SingleOrDefault();
            var noOfseats = Convert.ToInt32(TempData["noOfSeats"]);
            TempData.Keep("noOfSeats");
            t.TotalAmount = fare * noOfseats;
            return View(t);
        }
```

In Bookings user can view the details of the ticket and can print the ticket.

Admin Module:

Admin module has the login window and made available all the CRUD operations.

- Add New Bus Details
- Add Route Details
- Add Driver Details

Contribution:

Team Lead: Anjela Vinodini

- Implementation of Search Buses Module
- Implementation of Displaying Available Buses
- Implementation of error handling in user module
- Delete operation in PNR details
- Disabling Bus from Available Bus Result checking the bus condition

DB Lead: Rohit

- Data base Design and Creation
- Admin Module
- CRUD operations in Admin Module
- Implementation of error handling in Admin Module

UI Lead: Manoj kumar

- Data base Design
- Printing the User ticket
- Smooth Hover styling to the navigation bar
- Implementing Payment in User Module.

Controller Lead: Arfath pathan

- Registration module creation and validation
- Login Module creation and validation
- CRUD operation in admin module
- Error handling and validations.