****

**BAHRIA UNIVERSITY, (Karachi Campus)**

*Department of Software Engineering*

**REPORT**

**COURSE: SEL 221**

**Software design and archotecture**

**CLASS: BSE – 4 C (SPRING - 2024)**

**Project Title**

**“Transport Management System”**

**Group Name: “Transportify”**

|  |  |
| --- | --- |
| **Student Name** | **Enrollment#** |
| Ayesha | 02-131222-090 |
| Arifa Naseem | 02-131222-097 |
| Hafsa Shahid | 02-131222-088 |

**Submitted to:**

Course Instructor Lab Instructor . Eng. Majid Kaleem Engr. Asma Shaheen

Submission Date**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abstract**

The Transport Management System is a web-based application designed using the 3-tier architecture and Abstract Factory design pattern in C#. It offers comprehensive functionalities for both administrators and users, facilitating easy management of buses, routes, schedules, bookings, and passenger information. Through a user-friendly interface, the system aims to enhance the efficiency, reliability, and convenience of transportation services.

Table of Contents

[1. INTRODUCTION: 4](#_Toc169110845)

[2. ABSTRACT: 4](#_Toc169110846)

[3. PROJECT SCOPE: 4](#_Toc169110847)

[4. PROJECT FUNCTIONALITIES: 4](#_Toc169110848)

[5. ASSUMPTIONS AND CONSTRAINTS: 4](#_Toc169110849)

[6. UML DIAGRAMS 5](#_Toc169110850)

[7. MODULE DISTRIBUTION 8](#_Toc169110851)

[8. CODE (Design Pattern Code) 8](#_Toc169110852)

[9. INTERFACE 13](#_Toc169110853)

[10. REFRENCES 16](#_Toc169110854)

# **INTRODUCTION:**

Efficient transport management systems are crucial for ensuring smooth operations of transportation services. The proposed project aims to develop a Transport Management System utilizing modern software design principles and technologies. This system will highlight various aspects of transport management, including route planning, scheduling, ticket booking, and passenger management.

# **ABSTRACT:**

The Transport Management System is a web-based application designed using the 3-tier architecture and Abstract Factory design pattern in C#. It offers comprehensive functionalities for both administrators and users, facilitating easy management of buses, routes, schedules, bookings, and passenger information. Through a user-friendly interface, the system aims to enhance the efficiency, reliability, and convenience of transportation services.

# **PROJECT SCOPE:**

The scope of the project encompasses the development of a robust and scalable web application capable of handling various aspects of transport management. This includes features such as user authentication, bus and route management, schedule creation and management, ticket booking and management, and passenger information management. The system will be developed using .NET framework and SQL Server database, ensuring compatibility, reliability, and scalability.

# **PROJECT FUNCTIONALITIES:**

The Transport Management System will offer functionalities such as:

• User authentication and authorization

• Bus and route management

• Schedule creation and management

• Ticket booking and management

• Passenger information management

# **ASSUMPTIONS AND CONSTRAINTS:**

Assumptions:

• Users will have basic knowledge of using web-based applications.

• The system will be accessed through modern web browsers.

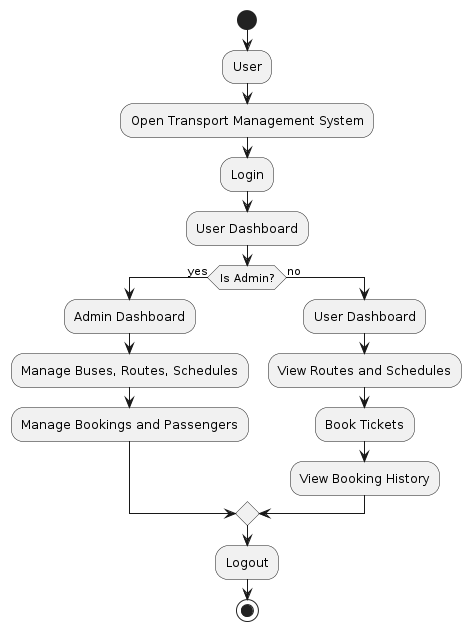
Constraints:

• Limited resources and time constraints for development.

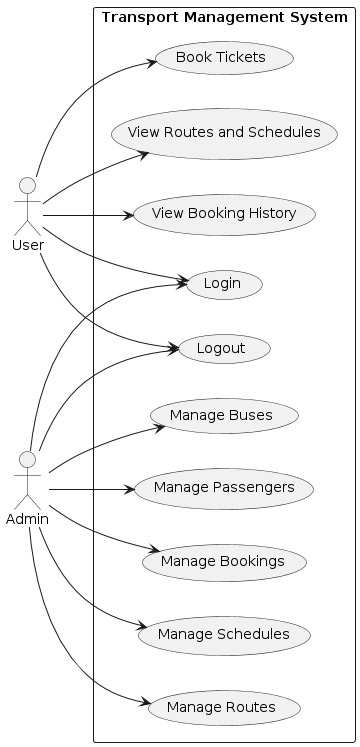
• Compatibility with specific versions of web browsers and operating systems.

# **UML DIAGRAMS**

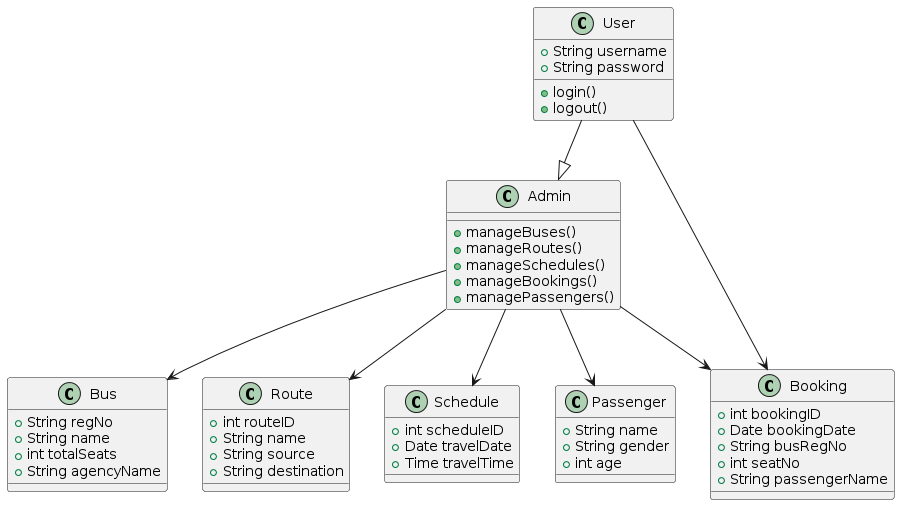
**FLOW DIAGRAM**:



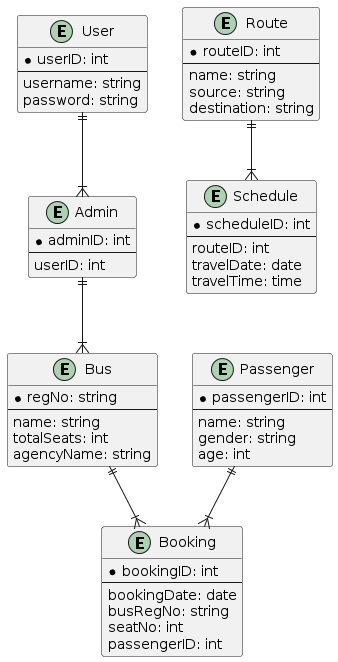
**USE CASE:**



**CLASS DIAGRAM:**



**ENTITY RELATIONSHIP DIAGRAM:**



# **MODULE DISTRIBUTION**

**Hafsa:**

User interface, web forms management, database creation and management.

**Ayesha:**

Design pattern implementation, receipt module, Business Logic Layer connection with presentation layer.

**Arifa:**

Business logic layer management and database access layer management.

# **CODE (**Design Pattern Code**)**

namespace TMSystem1.AbstractFactory

{

public interface IReceiptFactory

{

IReceipt CreateReceipt();

}}

namespace TMSystem1.AbstractFactory

{

public class ConcreteReceiptFactory : IReceiptFactory

{

public IReceipt CreateReceipt()

{

return new Receipt();

}}}

public interface IReceipt

{

void SetDetails(string busRegNo, int seatNo, int rowNo, string bookingDate, string travelDate, int travelTime);

string GetDetails();

}

public class Receipt : IReceipt

{

private string BusRegNo;

private int SeatNo;

private int RowNo;

private string BookingDate;

private string TravelDate;

private int TravelTime;

public void SetDetails(string busRegNo, int seatNo, int rowNo, string bookingDate, string travelDate, int travelTime)

{

BusRegNo = busRegNo;

SeatNo = seatNo;

RowNo = rowNo;

BookingDate = bookingDate;

TravelDate = travelDate;

TravelTime = travelTime;

}

public string GetDetails()

{

return $"Bus Name: {BusRegNo}\nSeat No: {SeatNo}\nRow No: {RowNo}\nBooking Date: {BookingDate}\nTravel Date: {TravelDate}\nTravel Time: {TravelTime}";SSS}}

**BUSINESS LOGIC LAYER CODE:**

using DataAccessLayer;

namespace BusinessLogicLayer

{

public class BLL

{

DAL obj = new DAL();

public void setData(string PassengerName, string PassengerGender, string Age)

{

obj.InsertData(PassengerName, PassengerGender, Age);

}

public void setDataRoute(string RouteName, string Source, string Destination)

{

obj.InsertDataRoutes(RouteName, Source, Destination);

}

public void UpdateDataRoute(int RouteID, string RouteName, string Source, string Destination)

{

obj.UpdateDataRoutes(RouteID, RouteName, Source, Destination);

}

public void DeleteDataRoute(int RouteID)

{

obj.DeleteDataRoutes(RouteID);

}

public DataTable ReadDataRoutes()

{

return obj.ReadDataRoutes();}

public DataTable GetRouteDetailsByID(int routeID)

{

return obj.GetRouteDetailsByID(routeID);

}

//Tickets management

public DataTable ReadBookedSeatsData()

{

return obj.ReadBookedSeatsData();

}

public DataTable GetBookedSeatsByNo(int SeatNo)

{

return obj.GetBookedSeatsByNo(SeatNo);

}

//tickets read and add

public DataTable ReadTicketsData()

{return obj.ReadTicketsData();

}

public DataTable GetTicketsByPNR(int SeatNo)

{

return obj.GetBookedSeatsByNo(SeatNo);}

public SeatStatus setTicketData(string busRegNo, int seatNo, int rowNo, string bookingDate, string travelDate, int travelTime)

{

SeatStatus seatStatus = obj.CheckSeatStatus(busRegNo, seatNo, rowNo);

if (seatStatus == SeatStatus.Available)

{

bool isInserted = obj.InsertTicketData(busRegNo, seatNo, rowNo, bookingDate, travelDate, travelTime);

return isInserted ? SeatStatus.Available : SeatStatus.NotFound;

}

else

{

return seatStatus;

}

}

public bool BookTicket( string busRegNo, int seatNo, int rowNo, string bookingDate, string travelDate, int travelTime)

{

return obj.BookTicket( busRegNo, seatNo, rowNo, bookingDate, travelDate, travelTime);

}

//profiles

public DataTable ReadDataPassenger()

{

return obj.ReadDataPassenger();

}

public DataTable GetPassengerDetailsByID(int routeID)

{

return obj.GetPassengerDetailsByID(routeID);

}

//admin profile

public DataTable ReadDataAdmin()

{

return obj.ReadDataAdmin();}

public DataTable GetAdminDetailsByID(int routeID)

{

return obj.GetAdminDetailsByID(routeID);

}

public void UpdateDataAdmin(string AdminName, string AdminPhone, string AdminGender, int AdminAge)

{

obj.UpdateDataAdmin(AdminName, AdminPhone, AdminGender, AdminAge);}

//drivers

public void setDataDriver( string DriverName, string DriverPhone, int Age, string Date\_Of\_Join)

{

obj.InsertDataDriver(DriverName, DriverPhone, Age, Date\_Of\_Join);

}

public DataTable ReadDataDriver()

{

return obj.ReadDataDriver();

}

public DataTable GetDriverDetailsByID(int DriverID)

{

return obj.GetDriverDetailsByID(DriverID);

}

public void UpdateDataDriver(int DriverID, string DriverName, string DriverPhone, int Age, string Date\_Of\_Join)

{

obj.UpdateDataDriver(DriverID, DriverName, DriverPhone, Age, Date\_Of\_Join);

}

public void DeleteDataDriver(int DriverID){

obj.DeleteDataDriver(DriverID);

}}}

**DATA ACCESS LAYER CODE:**

using System.Data.SqlClient;

namespace DataAccessLayer

{

public enum SeatStatus

{

Available,

Booked,

NotFound

}

public class DAL

{

SqlConnection conn = new SqlConnection("Data Source=DESKTOP-3GE0L0B;Initial Catalog=TMSdb;Integrated Security=True;");

public void InsertData(string PassengerName, string PassengerGender, string Age)

{try

{

SqlDataAdapter adp = new SqlDataAdapter("INSERT INTO Passenger VALUES ('" + PassengerName + "', '" + PassengerGender + "', '" + Age + "')", conn);

DataTable db = new DataTable();

adp.Fill(db);

}

catch

{

Console.WriteLine("Exception occurred while inserting data.");}}

public void InsertDataRoutes(string RouteName, string Source, string Destination)

{

try{

SqlDataAdapter adp = new SqlDataAdapter("INSERT INTO RouteDetails VALUES ('" + RouteName + "', '" + Source + "', '" + Destination + "')", conn);

DataTable db = new DataTable();

adp.Fill(db);

}catch

{Console.WriteLine("Exception occurred while inserting data.");

}}

public void UpdateDataRoutes(int RouteID, string RouteName, string Source, string Destination)

{try{

string query = "UPDATE RouteDetails SET RouteName = @RouteName, Source = @Source, Destination = @Destination WHERE RouteID = @RouteID";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@RouteID", RouteID);

cmd.Parameters.AddWithValue("@RouteName", RouteName);

cmd.Parameters.AddWithValue("@Source", Source);

cmd.Parameters.AddWithValue("@Destination", Destination);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch

{Console.WriteLine("Exception occurred while updating data.");}}

public void DeleteDataRoutes(int RouteID)

{try

{

string query = "DELETE FROM RouteDetails WHERE RouteID = @RouteID";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@RouteID", RouteID);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

catch

{

Console.WriteLine("Exception occurred while deleting data.");}}

public DataTable ReadDataRoutes()

{

DataTable dt = new DataTable();

try

{SqlDataAdapter adp = new SqlDataAdapter("SELECT \* FROM RouteDetails", conn);

adp.Fill(dt);}

catch

{Console.WriteLine("Exception occurred while reading data.");}

return dt;}

public DataTable GetRouteDetailsByID(int routeID)

{DataTable dt = new DataTable();

try

{string query = "SELECT \* FROM RouteDetails WHERE RouteID = @RouteID";

SqlDataAdapter adp = new SqlDataAdapter(query, conn);

adp.SelectCommand.Parameters.AddWithValue("@RouteID", routeID);

adp.Fill(dt);}

catch{

Console.WriteLine("Exception occurred while fetching route details.");}

return dt;}

public void InsertBusInfo(string BusRegnNo, string BusName, int TotalSeats, int Latitude, int Longitude, string AgencyName)

{try

{SqlDataAdapter adp = new SqlDataAdapter("INSERT INTO BusInfo VALUES ('" + BusRegnNo + "', '" + BusName + "', '" + TotalSeats + "' , '" + Latitude + "' , '" + Longitude + "' , '" + AgencyName + "')", conn);

DataTable db = new DataTable();

adp.Fill(db);}

catch

{Console.WriteLine("Exception occurred while inserting data.");}}

public void UpdateBusInfo(string BusRegnNo, string BusName, int TotalSeats, int Latitude, int Longitude, string AgencyName)

{try

{string query = "UPDATE BusInfo SET BusName = @BusName, TotalSeats = @TotalSeats, Latitude = @Latitude, Longitude = @Longitude, AgencyName = @AgencyName WHERE BusRegNo = @BusRegnNo";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@BusRegnNo", BusRegnNo);

cmd.Parameters.AddWithValue("@BusName", BusName);

cmd.Parameters.AddWithValue("@TotalSeats", TotalSeats);

cmd.Parameters.AddWithValue("@Latitude", Latitude);

cmd.Parameters.AddWithValue("@Longitude", Longitude);

cmd.Parameters.AddWithValue("@AgencyName", AgencyName);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();}

catch (Exception ex)

{Console.WriteLine("Exception occurred while updating data: " + ex.Message);}}**}**

**PRESENTATION LAYER CODE:**

using System.Web.UI;

using System.Web.Services;

namespace TMSystem1{

public partial class receipt : System.Web.UI.Page{

protected void Page\_Load(object sender, EventArgs e){

if (!IsPostBack){

if (Session["Receipt"] != null){

string receiptDetails = Session["Receipt"].ToString();

if (Session["Source"] != null)

{receiptDetails += "\nSource: " + Session["Source"].ToString();}

if (Session["Destination"] != null)

{receiptDetails += "\nDestination: " + Session["Destination"].ToString();}

receiptDetails += "\nFare: 100 rs";

ReceiptDetailsTextBox.Text = receiptDetails;}}}

[WebMethod]

public static void SaveImage(string imgData)

{try

{string base64 = imgData.Replace("data:image/png;base64,", "");

byte[] bytes = Convert.FromBase64String(base64);

string filePath = @"C:\Users\Administrator\Downloads\captured\_screen.png";

using (MemoryStream ms = new MemoryStream(bytes))

{using (Bitmap bitmap = new Bitmap(ms))

{bitmap.Save(filePath, ImageFormat.Png);}}}

catch (Exception ex)

{Console.WriteLine(ex.Message); throw new Exception("An error occurred while saving the receipt. Please try again.", ex);}}}}

# **INTERFACE**

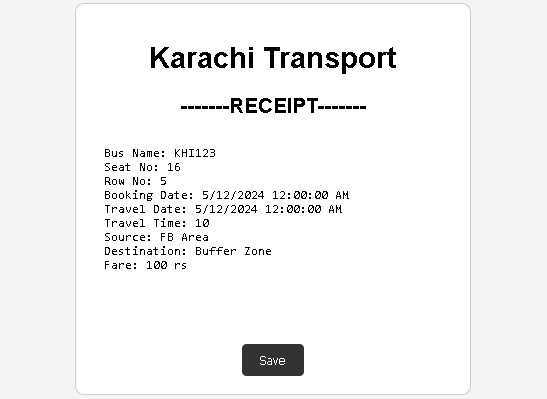
|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |



# **REFRENCES**

<https://acropolium.com/blog/transportation-management-software-development-a-detailed-overview/>

<https://ijrpr.com/uploads/V3ISSUE5/IJRPR4070.pdf>

**Drive Link:**

**Teacher Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Remarks**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission Date**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_