

HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
COURSE NAME: Windows Programming



FINAL PROJECT REPORT

Project name:

BUS TICKET BOOKING MANAGEMENT APPLICATION

Lecturer: Doctor Le Van Vinh

Course ID: WIPR230579E_22_2_02FIE

Group: 2

Date: 2nd Sem/2022-2023

Ho Chi Minh City, May , 2023

(This page was intentionally left blank)

LIST OF STUDENTS – GROUP 2

Project: Bus ticket booking management system

<i>ID</i>	<i>Full name</i>
21110758	Lê Xuân Cường
21110092	Bùi Quốc Thông
21110019	Trần Gia Huy

Doctor's comment

Ho Chi Minh City, May ..., 2023

Grading

CONTENTS

PROLOGUE	5
INTRODUCTION	6
CHAPTER I: SYSTEM OVERVIEW	7
1. Specifications	7
1.1. Problem statement	7
1.2. Overview	7
2. Real-life functions	10
2.1. Booking period	10
2.2. Departure period	11
2.3. Drop-off period	11
2.4. Ticket cancellation/time change period	11
2.5. Delivery period	11
3. Main application functions	11
CHAPTER II : DATABASE DESIGN	13
1. Conceptual level database design	13
2. Logical level database design	13
3. Database diagram	14
CHAPTER III: SOFTWARE ARCHITECTURE	17
1. Project structure	17
2. Class analysis	18
2.1. Class overview	18
2.2. Class explanation	19
CHAPTER IV: SYSTEM INTERFACE DESIGN	64
1. Applications and services used	64
2. Software interface	64
2.1. User function	64

2.2. Employee function (only for employee account)	76
2.3. Admin function (only accessible by admins)	78

PROLOGUE

Firstly, we would like to express our gratitude to Doctor Le Van Vinh for his whole-hearted instructions that helped us finish our final project for the Windows Programming course. Thanks to the knowledge the professor has provided us, we were able to firmly grasp the basic knowledge and foundation for building a bus ticket management system. And through this project, our group would like to present the development process of an application and demonstrate by programming a related project once again.

During the process of executing this project, it will be hard to avoid mistakes. Because of that, we would love to get the professor's suggestion on improving our work so it would be more functional and complete. We wish you good health and the best of luck pursuing the path of teaching.

Finally, we would like to thank all the teachers and classmates who studied with us on this course and offered us support while we carried out our final project.

INTRODUCTION

In recent years, the Information Technology (IT) field has been integrated into our society and daily lives, regardless of any field and/or occupations. It also plays an important part of booking management in Vietnam and especially in almost every country as there are many applications made to help fix problems that big organizations frequently face.

The creation of the bus ticket booking management system is the result of many developers' creativity and hard work with the aim of aiding companies in managing their businesses.

With that in mind, to better understand the application and role of Information and Technology (IT) in Windows Programming, we have decided on the “**Bus ticket booking management system**” as our final project.

CHAPTER I: SYSTEM OVERVIEW

1. Specifications

1.1. Problem statement

The bus ticket booking management system will:

- Manage the employees, passengers, bus, trips, routes easier.
- Convenient for users to check and book trips.
- Check the state and location of the trip more clearly through a map.
- More convenient for the bus company to obtain statistics: revenue, number of passengers, number of trips, employee salary, outcome, etc. per day, per month, per year.

Vehicle management: Manage travel vehicles including their location, date and time of arrival/departure, price, etc.

System management: Manage employees, drivers, customers, travel curriculum.

Statistics: Employee statistics, vehicle statistics, daily sales, etc.

1.2. Overview

A bus company needs to have a bus ticket reservation system. The bus ticket reservation system should contain the following data:

The bus company manages a lot of agents. Each agent has: agent ID, place id, cash reserve ID, address, agent name.

Each agent has only one cash reserve. A cash reserve includes cash reserve ID and counter.

An agent has many employees. Each employee has: employee ID, position ID, account ID, agent ID, name, address, phone number, identity number, salary, email, date of birth, state.

The employee state can be:

- Not working
- Working

Each employee is provided with an account to access into the system (username and password). Each employee type has a different position.

The information of the position group contains: position ID, type.

There are several types:

- Administrator
- Travel planner
- Travel supervisor
- Driver
- Ticket seller
- Service guide
- Security guard
- Porter

Each position group has separate privileges. The information of the privileges group includes: privilege ID, name.

The agent manages passengers. Each passenger has: passenger ID, name, phone number, address, identity number, gender, email.

The gender attribute of passenger above has two options:

- Male
- Female

Easily manage and filter the address of stations in the general local area, there is information of places: place ID, region.

Each passenger can choose a pickup station and drop-off station. Each station has: station ID, detailed address, name, capacity, parked bus number.

The bus of each brand has: bus ID, registration number, model, capacity, status, type.

Status of the bus can be:

- Ongoing
- Idle
- Break
- Incident

Type of the bus can be:

- Interprovince
- Transit

Routes involving the journey have: route ID, start bus station ID, final bus station ID, travel distance.

Each trip is set up by the travel planner which includes: trip ID, drivers ID, bus ID, route ID, departure time, duration, number of booked seats, state.

The state of trip above has three options:

- Waiting
- Going
- Finish
- Cancel

The drivers ID in the trip relation is an attribute of TRIP_DRIVER relation: trip ID, driver ID. Note that driver ID is a multivalued attribute.

The agent distributes tickets to the passenger. Each ticket has: ticket ID, trip ID, passenger ID, status, fare, type, seat number.

The status of the ticket can be:

- Available
- Bought

The type of ticket has two options:

- Seat ticket
- Sleeper ticket

The agent manages the booking transaction. Each booking transaction includes: transaction ID, ticket ID, passenger ID, employee ID, booking time.

Each driver has an employee ID number, license level and type of driver (long-haul driver and transit driver), state.

The state of driver can be:

- Not drive
- Is driving

General rules:

- Each employee can take on more than 1 position
- Each passenger can book more than 1 ticket
- Each trip can have more than 1 driver

The bus company provides a delivery service so that the customers can send a package without booking a ticket. They must provide information about their packages such as: mass, the phone number of sender and receiver. This package will have an ID and price. The package's price is determined by a pre-determined pricing policy: ID, mass of package and price_per_km.

When a big event happens, the bus company will hold discount periods to lower the price of tickets.

Besides, the refund policy can help the passengers receive part of the fare when they cancel their trips and tickets.

2. Real-life functions

2.1. Booking period

**** Offline booking:***

The service guide records the passenger's full field information including: their name, ID number, phone number, address, gender, email. Then, the ticket seller checks again to guarantee all the required fields are correctly fielded.

Then, the passenger picks a trip by choosing from multiple options: destination, pickup station, drop-off station, departure time, the available seat, ticket type. Options will be planned by the travel planner, so the passenger must follow this template.

Then, the ticket seller verifies the customer's selection. If valid, the ticket seller informs the passenger and waits for their confirmation. If they confirm, the ticket seller prints the ticket, gives it to them and reminds them to arrive at the correct time on the ticket. Else if they refuse, the customer needs to modify the information.

**** Online booking:***

First of all, the passengers must have an account to access the bus ticket booking application. If they don't have an account yet, they have to register and log in to book the ticket. If they have an account, they only need to log in to book.

Afterwards, passengers will access the system to book their ticket. They will fill in the information about their name, ID number, phone number, address, gender, email, destination, pickup station, drop-off station, departure time, the available

seat, ticket type. The system will send a verification code through email, then passengers fill in the app to verify their booking action.

Next, the system will provide information about the ticket and passengers will have to pay the ticket fare via online payment.

2.2. Departure period

The passengers wait for the agent. 15 minutes before the departure time of the trip, the vehicle will take the passengers to the bus station.

At the bus station, the porter put the passengers' luggage into the trunk.

When it's time, the service guide instructs passengers to the vehicle, and provides water and tissues to them.

2.3. Drop-off period

When the bus arrives at the last bus station, the porter takes passengers' luggage from the bus and gives it to the passenger.

2.4. Ticket cancellation/time change period

** Offline cancellation/time change (in the agent):*

The passengers must go to the agent of the bus system and have the ticket-selling employee cancel or change their ticket. In some different cases, the fee of the cancellation/change is also different.

** Online cancellation/time change (on the application system):*

The passengers must access the application that they had booked their tickets to change or cancel their ticket. In this case, they must pay for this change.

2.5. Delivery period

The customer must take their packages before the time of departure of the bus. The employee will measure the weight of the packages and inform the customers with the incurred fees.

3. Main application functions

Administrator (global):

- Add, modify, delete, authorize for positions
- Add, modify, delete employee of the position
- Statistic information about trip, the number of sold tickets

Travel planner:

- Add, modify, delete trips
- Add, modify, delete routes
- Add (distribute the tickets of the trip), modify, delete tickets

Travel supervisor:

- Add, delete passengers of the trip
- Report errors (trip, route, passenger, booking)

Ticket-selling:

- Add, modify, delete passenger
- Export bill
- Export ticket
- Change the state attribute of trips

Passenger (when booking online):

- Check price ticket of each route
- Check the information about booked tickets
- Book one or many tickets
- Change the information about ticket (information of passenger, the route, departure time, departure date)
- Cancel their tickets
- Export their tickets

* Authorization:

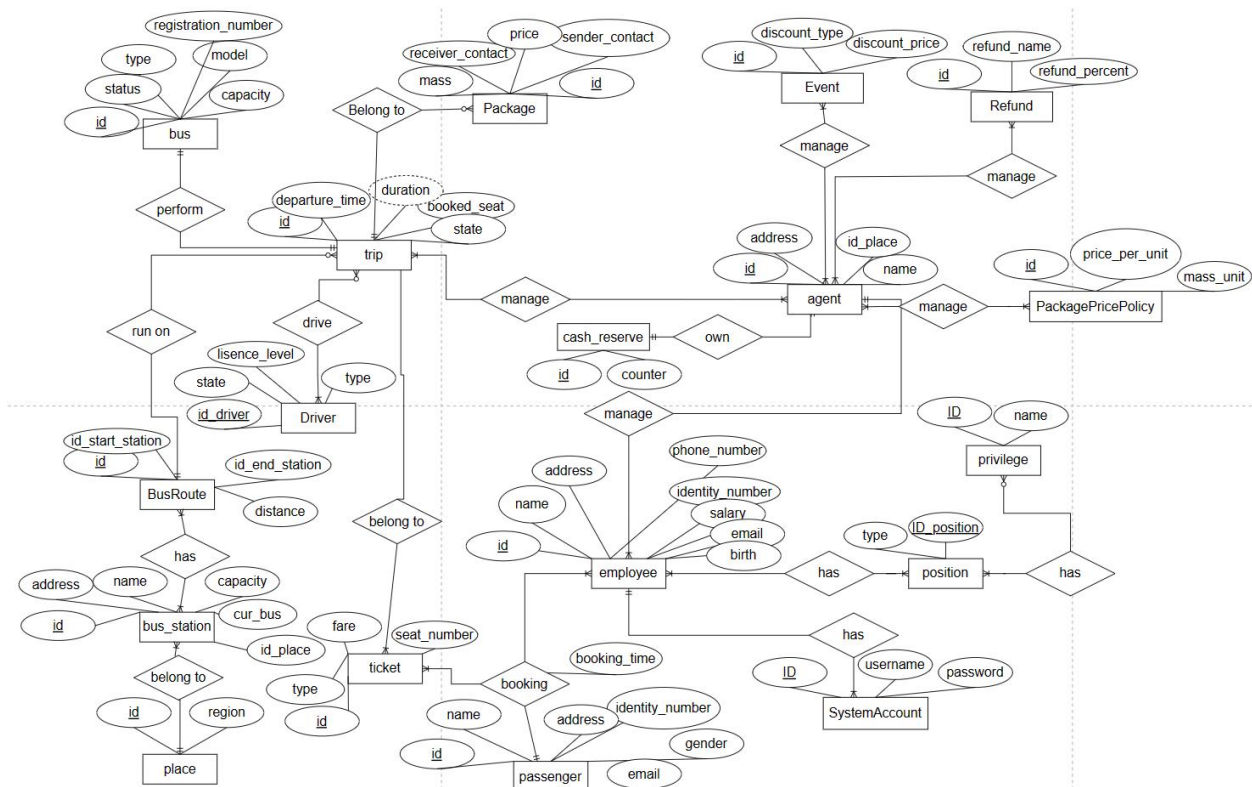
- Admin: Full control on the whole system – Global privilege
- [...] (Other privileges): Local privilege

CHAPTER II : DATABASE DESIGN

(Database developer's perspective)

1. Conceptual level database design

From the necessary data in description of the specifications, the following Entity Relationship Model (ERD) is formed.



Total 18 relations with 9 N-N, 1 three-way relation.

Sharp image: [ERD sharp image \(busticketbookingerd.netlify.app\)](http://busticketbookingerd.netlify.app)

2. Logical level database design

From the Entity Relationship Model (ERD), we have the following tables:

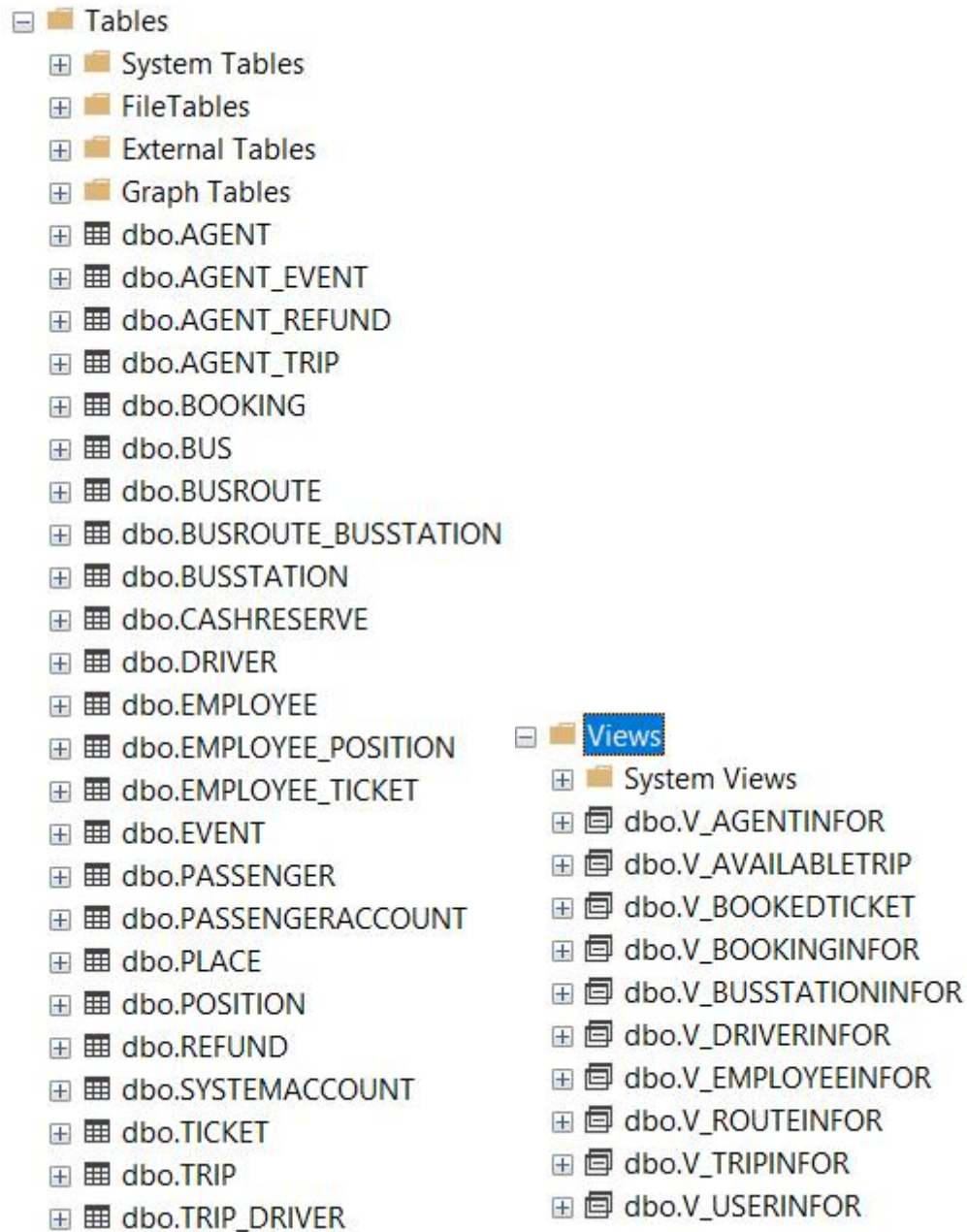
1. AGENT (id_agent, id_place, id_cash_reserve, address, name)
2. AGENT_EVENT (id_agent, id_event)
3. AGENT_REFUND (id_agent, id_refund)
4. AGENT_TRIP (id_agent, id_trip)

5. BOOKING (id_booking, id_ticket, id_passenger, id_employee, booking_time)
6. BUS (id_bus, registration_number, model, capacity, status, type)
7. BUSROUTE (id_route, id_start_station, id_end_station, distance)
8. BUSROUTE_BUSSTATION (id_bus_route, id_bus_station)
9. BUSSTATION (id_bus_station, id_place, name, bus_capacity, count_current_bus)
10. CASHRESERVE (id_cash_reserve, counter)
11. DRIVER (id_driver, license_level, type, state)
12. EMPLOYEE (id_employee, id_account, id_agent, name, address, phone_number, identity_number, salary, email, birthdate, state)
13. EMPLOYEE_POSITION (id_employee, id_position)
14. EMPLOYEE_TICKET (id_employee, id_ticket)
15. EVENT (id_event, discount_type, discount_percent)
16. PASSENGER (id_passenger, name, phone_number, address, identity_number, gender, email)
17. PASSENGERACCOUNT (id_passenger, username, password)
18. PLACE (id_place, region)
19. POSITION (id_position, type)
20. REFUND (id_refund, refund_name, refund_percent)
21. SYSTEMACCOUNT (id_account, username, password)
22. TICKET (id_ticket, id_trip, status, fare, type, seat_number)
23. TRIP (id_trip, id_bus, id_bus_route, departure_time, duration, booked_seat, status)
24. TRIP_DRIVER (id_trip, id_driver)

3. Database diagram

After setting up the necessary tables and relations along with their respective constraints and triggers, a Physical level diagram will be created:

The structure of the database after deployment:

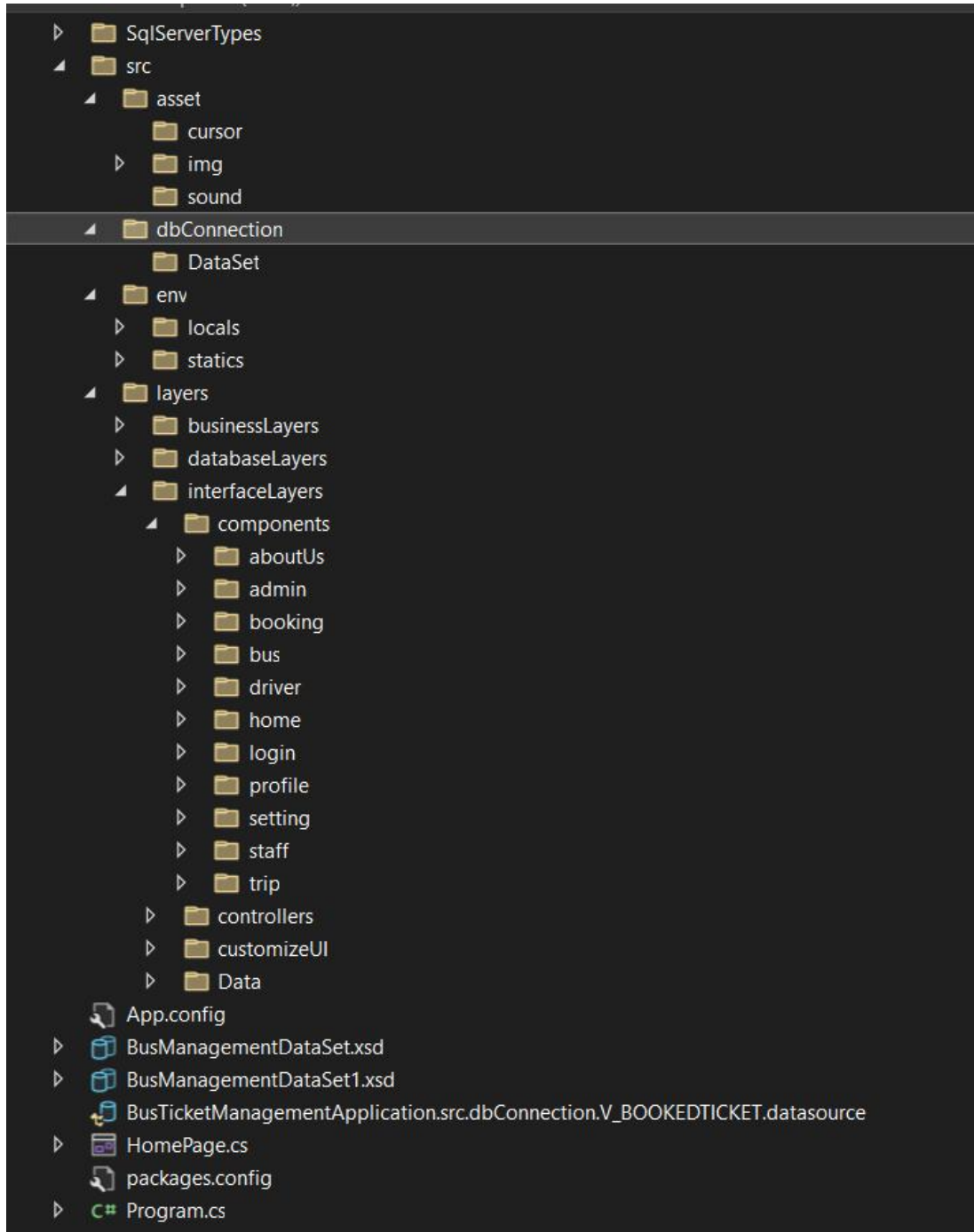


CHAPTER III: SOFTWARE ARCHITECTURE

(Software developer's perspective)

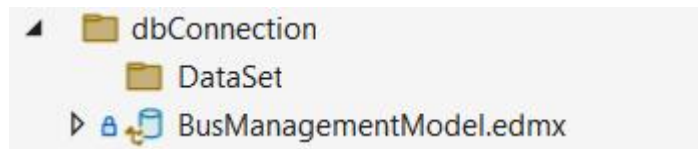
1. Project structure

This is the general structure for 3 versions in this project.



In Entity Framework version, it has some differences:

- dbConnection folder

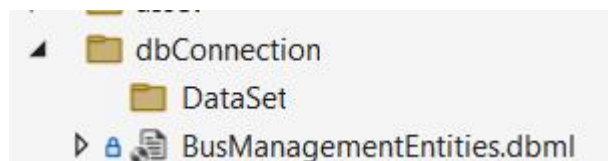


- layers folder



In LinQ to SQL version, it has some differences:

- dbConnection folder



- layers folder



2. Class analysis

2.1. Class overview

Class diagram: The following picture shows all of classes of the software



2.2. Class explanation

* env folder

- ./Locals/LocalEnv.cs

The purpose of this class is to provide private data (Email servername, encodedEmailServerPassword)

- ./statics/StaticEnv.cs

This class used to provide global (public) data (size of controls, connection strings, HTML code for the confirm email).

```

internal static class StaticEnv
{
    0 references
    public static Size AppSize { get => new Size(1200, 700); }
    0 references
    public static Size DashboardSize { get => new Size(235, 700); }
    0 references
    public static Size FillContentSize { get => new Size(965, 598); }
    0 references
    public static Size NavigationBarSize { get => new Size(965, 44); }
    //
    1 reference
    public static string GetConfirmSubject(string ticketId)
    => $"Mua vé thành công. Mã vé: {ticketId}";
    1 reference
    public static string GetTemPlateConfirmMessage(string ticketId, string bookingTime, string name, string email, string phone,
    {
        string departureDate = departureTime.ToLongDateString();
        string departureHour = departureTime.ToLongTimeString();
        return $"<div class=\">\r\n    <div class=\"aHL\"></div>\r\n    <div id=\":od\" tabindex=\"-1\"></div>\r\n    <div\r\n
    }
    0 references
    public static string GetEFConnectionString(string username, string password)
    => $"metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnection.BusManagementModel.ssdl|res://*/src.
    1 reference
    public static string GetEFConnectionString()
    => $"metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnection.BusManagementModel.ssdl|res://*/s
    4 references
    public static string GetDefaultEFConnectionString()
    => "metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnection.BusManagementModel.ssdl|res://*/s
}

```

* databaseLayers folder (only for ADO.NET version)

- ./DBMain.cs

This class provides necessary functions which help C# run/execute functionalities/commands under SQL server.

- ./IDBMAIN.CS

This interface publish fundamental functions related to interaction between C# and SQL server.

```

7 references
List<DataRowView> ExecuteQueryList(string query, CommandType type, ref string errMsg);
1 reference
SqlDataReader ExecuteQueryDataReader(string query, CommandType type, ref string errMsg);
2 references
bool ExecuteNonQuery(string query, CommandType type, ref string errMsg);
5 references
int ExecuteScalar(string query, CommandType type, ref string errMsg);
5 references
string ExecuteScalarByString(string query, CommandType type, ref string errMsg);

```

*business layers

In 3 versions, we have a separate class which provides entities of our database

- Entity Framework: BusManagementEntities
- LinQ to SQL: BusManagementEntitiesDataContext

- ADO.NET: BusManagementEntities (this class is not provided by default, we model (create) it ourself, we will show it later)

Move into the detail:

- ***./BSAgent.CS***

This method searches and returns a list of agents based on input criteria. It accepts two parameters:

- + input - A search string to match against the Agent_ID column
- + region - A region to filter the results by. If "All" is passed, all regions are included.

The purpose here is to provide an interface to search and retrieve agent information for use elsewhere in the application.

- ***./BSBooked.CS***

By using the functions GetAllBookedTrips and SearchBookedTickets, the system can provide methods to search trips and get all information of booked trips for users.

The class provides two methods:

- + GetAllBookedTickets(): This method retrieves all the records in the V_BOOKEDTICKET view from the BusManagementEntities database context and returns them as a list of DataRowView objects.
- + SearchBookedTickets(string passengerId, string input, string src, string des, DateTime dateTime): This method takes in five parameters - passengerId, input, src, des, and dateTime. It queries the V_BOOKEDTICKET view from the database context and filters the results based on the search criteria specified by these parameters.

- ***./BSBooking.CS***

This class supplies some methods to get the trips, the tickets, available seat, booked ticket and add passenger or booking.

It has the following functionality:

1. GetTrip() - Fetches trip details from the V_AVAILABLETRIP view based on trip ID.

2. GetRouteId() - Gets the route ID for a given trip ID from the TRIPS table.
3. GetAvailableSeat() - Calls a stored procedure func_GetAvailabelSeat() to get available seat numbers for a trip ID.
4. GetTicket() - Fetches a ticket from the TICKET table based on ticket details.
5. AddPassenger() - Calls a stored procedure pro_AddPassenger() to add a new passenger to the system.
6. GetBookedTicket() - Fetches a booked ticket from the V_BOOKEDTICKET view.
7. AddBooking() - Calls pro_AddBooking() to add a new booking, linking a ticket, passenger and employee.

- ***./BSBus.CS***

This class illustrates the functionality of getting the bus, searching for the bus by ID and registration number.

It has the following functionality:

1. GetAllBus() - Fetches all buses from the BUSES table.
2. SearchBusById() - Searches for buses by ID. It accepts an input string and a type (true for AC, false for non-AC). It calls the FilterBus() method to filter by type, then performs the ID search on the result.
3. SearchBusByRegistrationNumber() - Similar to SearchBusById(), but searches by registration number instead of ID.
4. FilterBus() - Filters the buses by AC or non-AC type.

- ***./BSBusStation.CS***

This class illustrates the functionality of getting to the information bus station from the database.

The responsibilities of this class are:

1. Retrieve bus station data from the database
2. Hide database details from callers
3. Provide a consistent API to get all bus station information

- ***./BSDriver.CS***

The purpose of this is to populate driver data for use in other parts of the application, like showing driver details, assigning drivers to buses, etc. The business layer abstracts away the database details and provides a clean interface.

It contains a business layer class called BSDriver which has 2 methods:

GetAllDrivers() - This fetches all drivers from the database table V_DRIVERINFOR using Entity Framework and returns the results as a list.

SearchDriver() - This method searches drivers based on input criteria. It accepts an input string and a tag (0 or 1). Based on the tag, it searches either:

- + Employees_ID column (if tag is 0)
- + Lisence_Level column (if tag is 1)

- ***./BSEmployee.CS***

The BSEmployee allows for searching and deleting employees in the system. The class utilizes a BusManagementEntitiesDataContext instance to interact with the database and execute the necessary database operations.

The responsibilities of this class are:

1. Retrieve employee data from the database
2. Provide search and filter functionality on employee data
3. Execute operations on employees (like disabling)
4. Hide database details from callers

- ***./BSLogin.CS***

The BSLogin allows for user authentication, retrieval of user information, checking user roles, creating new user accounts, and changing user passwords. The class relies on a BusManagementEntitiesDataContext instance to interact with the database and perform the necessary database operations.

It has the following functionality:

1. GetUser() - Fetches a passenger's details from the V_USERINFOR view based on their ID.
2. GetEmployee() - Fetches an employee's details from the V_EMPLOYEEINFOR view based on their employee ID.
3. IsAdmin() - Checks if an employee ID belongs to an administrator.
4. ValidateUser() - Validates a username and password and returns the corresponding passenger ID or employee ID. It also returns an error message.
5. CreateNewUser() - Creates a new passenger account and stores the details in various tables. It returns the new passenger ID.
6. ChangeUserPassword() - Calls a stored procedure to change the password for a passenger or employee account.

- ***./BSMain.CS***

The BSMain class provides utility methods to execute functions and procedures in the database. It interacts with the database through an instance of BusManagementEntitiesDataContext and handles different parameter types for different function and procedure calls.

The main responsibilities of this class are:

- Executing table valued functions and returning the results
- Executing scalar valued functions and returning the result
- Executing stored procedures
- Execute SELECT queries and return results as DataRowViews
- Provides utilities like fetching data from tables

- ***./BSPassenger.CS***

The BSPassenger class provides functionality to retrieve a new passenger ID, retrieve specific passengers by ID, update passenger and employee details, and search for passengers based on various criteria. It interacts with the PASSENGERs and EMPLOYEEs tables through an instance of BusManagementEntitiesDataContext.

The class contains business logic related to managing passenger data:

1. GetNewPassengerId() - Fetches a new passenger ID from a scalar valued function.
2. GetPassenger() - Fetches a passenger's details from the PASSENGER table given their ID.
3. UpdatePassenger() - Updates a passenger's details like name, phone number, email and gender.
4. UpdateEmployee() - Similar to UpdatePassenger(), but updates an employee's details.
5. SearchPassenger() - Searches for passengers based on an input string and tag

- ***./BSPlace.CS***

The BSPlace class provides functionality to retrieve place names, retrieve the ID of a place based on its name, and retrieve the name of a place based on its ID. It interacts with the PLACES table through an instance of BusManagementEntitiesDataContext.

This class contains business logic related to managing places in the system:

1. GetPlaceNames() - Fetches all place names from the PLACES table.
2. GetPlaceId() - Given a place name, fetches the corresponding place ID from the PLACES table.

3. GetPlaceName() - Given a place ID, fetches the corresponding place name from the PLACES table.

- ***./BSPosition.CS***

The BSPosition class provides functionality to retrieve the names of all positions from the database. It interacts with the POSITIONS table through an instance of BusManagementEntitiesDataContext and returns the position names as a list of strings.

This code contains business logic related to managing positions in the system GetPositionNames() - Fetches all position names from the POSITIONS table.

- ***./BSRoute.CS***

The BSRoute class provides functionality to fetch specific routes by ID or retrieve all routes available in the database. It interacts with the V_ROUTEINFORs table through an instance of BusManagementEntitiesDataContext.

This class contains business logic related to managing routes:

1. GetRoute() - Fetches details of a single route given its ID, from the V_ROUTEINFOR view.

2. GetAllRoutes() - Fetches details of all routes from the V_ROUTEINFOR view

- ***./BSTicket.CS***

The BSTicket class provides a method to obtain a new ticket ID from the database by calling a function or procedure defined in the BSMain class. The details of the database interaction and the implementation of the RunFunc method are not included in the provided code snippet.

This class contains logic related to generating new ticket IDs:

GetNewTicketId() - Calls a scalar valued function func_auto_id_ticket() to generate a new ticket ID.

- ***./BSTrip.CS***

The BSTrip class provides functionality for retrieving trip information, searching for available trips, searching for trips based on specific criteria, and updating the status of trips in the database.

This class contains business logic related trips:

1. GetAllTrips() - Fetches details of all trips from the V_TRIPINFOR view.
2. SearchAvailableTrips() - Searches for available trips based on input, source place, destination place and departure time.
3. SearchTrips() - Similar to SearchAvailableTrips() but searches all trips, not just available ones.
4. SetCancelTrip() - Calls a stored procedure to cancel a trip.
5. SetFinish() - Calls a stored procedure to mark a trip as finished.
6. SetGoing() - Calls a stored procedure to mark a trip as going.

- ***./BusManagementEntities.CS (only ADO.NET version)***

This code defines a class that represents the database context for the bus ticket management system. It contains:

- Properties to access tables and views
- Methods to call stored procedures and functions
- Definitions of column names for each table

This context class provides a unified way to access and manipulate data in the bus ticket management database.

- ***./IBSMain.CS (only ADO.NET version)***

This method retrieves the data from a table with the given name.

Classes that implement this interface will be able to:

- Retrieve data from tables in the database
- Interact with stored procedures and functions

The purpose of this interface is to define a consistent API for accessing the database. Any class that implements this interface can then be used interchangeably.

- ***./IBusManagementEntities.CS (only ADO.NET version)***

This interface defines the functionality for a database context class that specifically manages data for a bus ticket management system.

The benefits of defining this interface are:

- Loose coupling between layers
- Increased testability
- Segregation of concerns

*** interfaceLayers folder (the same for three versions)**

This folder includes child folders:

- components

About US

./aboutUs/AboutUs.cs

Purpose:

The "AboutUs" class represents a form in a Windows Forms application that serves as the About Us page or section. It provides the necessary functionality for the form, including initialization and handling the loading event.

./aboutUs/AboutUsNavigationBar.cs

Purpose:

The "AboutUsNavigationBar" class represents a form in a Windows Forms application that serves as a navigation bar for the About Us section. It provides functionality to handle button clicks, update the navigation index, and visually indicate the selected button in the panel.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class, providing basic window functionality.

Fields and Properties:

`navIndex` is a static field that represents the current navigation index. It keeps track of the selected button.

`parentForm` is a field that holds a reference to the parent form (of type "App") to interact with it.

`NavIndex` is a static property that allows accessing and updating the value of the `navIndex` field.

Constructors: There are two constructors available. The first one is a parameterless constructor that initializes a new instance of the class. The second one accepts a parameter of type "App" and initializes the `parentForm` field with it.

Handler_NavBtn_Click Event Handler: This event handler is triggered when a navigation button is clicked. It updates the `navIndex` field based on the clicked button's tag. It also sets the `MainFeatureIndex` property of the parent form to a virtual number (5 in this case).

PnlMainContainer_Paint Event Handler: This event handler is triggered when the panel is being painted. It determines the selected button based on the `navIndex` field and draws a line at the bottom of the selected button using the `Graphics.DrawLine` method.

./aboutUs/BusRoute.cs

Purpose:

The "BusRoute" class represents a form in a Windows Forms application that displays bus routes information. It initializes the form and loads the default data for the bus routes.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class, providing basic window functionality.

Constructor: The constructor is a method that initializes a new instance of the "BusRoute" class. It is responsible for initializing the components of the form using the "InitializeComponent()" method.

BusRoute_Load Event Handler: This event handler is triggered when the "BusRoute" form is loaded. It is associated with the "Load" event of the form. Inside this event handler, the "LoadDefault()" method is called to load the default data for the bus routes.

LoadDefault Method: The "LoadDefault()" method is a private method that loads the default data for the bus routes. It creates an instance of the "BSRoute" class, which is presumably a business service or data access class. It then calls the "GetAllRoutes()" method from the "bSBusRoute" instance to retrieve the data for all bus routes. Finally, it sets the data source of the "dgvBusRoute" DataGridView control to the retrieved data, presumably displaying it in a table format.

./aboutUs/BusStation.cs

Purpose:

The "BusStation" class represents a form in a Windows Forms application that displays bus station information. It initializes the form and loads the default data for the bus stations.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class, providing basic window functionality.

Constructor: The constructor is a method that initializes a new instance of the "BusStation" class. It is responsible for initializing the components of the form using the "InitializeComponent()" method.

BusStation_Load Event Handler: This event handler is triggered when the "BusStation" form is loaded. It is associated with the "Load" event of the form. Inside this event handler, the "LoadDefault()" method is called to load the default data for the bus stations.

LoadDefault Method: The "LoadDefault()" method is a private method that loads the default data for the bus stations. It creates an instance of the "BSBusStation" class, which is presumably a business service or data access class. It then calls the "GetAllStations()" method from the "bSBusStation" instance to retrieve the data for all bus stations. Finally, it sets the data source of the "dgvBusStation" DataGridView control to the retrieved data, presumably displaying it in a table format.

Admin

./admin/reports/CashReserveStatisticDataSet.xsd

./admin/reports/CashReserveStatistic.rdlc

./admin/AdminNavigationBar.cs

Purpose:

The "AdminNavigationBar" class represents a form in a Windows Forms application that serves as a navigation bar for administrative features. It implements the INavigationBar interface and provides functionality to handle button clicks, update the navigation index, and visually indicate the selected button in the panel.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class and implements the "INavigationBar" interface, which likely defines additional navigation-related methods or properties.

Fields and Properties:

navIndex is a static field that represents the current navigation index. It keeps track of the selected button.

parentForm is a field that holds a reference to the parent form (of type "App") to interact with it.

NavIndex is a static property that allows accessing and updating the value of the navIndex field.

Constructors: There are two constructors available. The first one is a parameterless constructor that initializes a new instance of the class. The second one accepts a parameter of type "App" and initializes the parentForm field with it.

Handler_NavBtn_Click Method: This method is called when a navigation button is clicked. It updates the navIndex field based on the clicked button's tag. It

also sets the `MainFeatureIndex` property of the parent form to a virtual number (7 in this case).

`PnlMainContainer_Paint` Method: This method is called when the panel is being painted. It determines the selected button based on the `navIndex` field and draws a line at the bottom of the selected button using the `Graphics.DrawLine` method.

./admin/CashReserve.cs

Purpose:

The "CashReserve" class represents a form in a Windows Forms application that displays cash reserve information. It allows the user to search and filter cash reserve data based on specific criteria.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class, providing basic window functionality.

Fields:

`searchInput` is a private field that stores the user's search input.

Constructor: The constructor is a method that initializes a new instance of the "CashReserve" class. It is responsible for initializing the components of the form using the "InitializeComponent()" method.

CashReserve_Load Event Handler: This event handler is triggered when the "CashReserve" form is loaded. It is associated with the "Load" event of the form.

Inside this event handler, the "LoadDefaultPlaces()" method is called to load the default places.

LoadDefaultPlaces Method: The "LoadDefaultPlaces()" method loads the default places into the "CbRegion" ComboBox control. It retrieves the place names using the "GetPlaceNames()" method from a "BSPlace" class (presumably a business service or data access class) and adds them to the ComboBox control. The "All" option is added as the first item, and the ComboBox is initially set to the first item.

FilterAgent Method: The "FilterAgent()" method filters the agents based on the search input and selected region. It creates an instance of the "BSAgent" class (presumably a business service or data access class) and calls the "SearchAgents()" method to retrieve the filtered agents based on the search input and selected region. The retrieved data is then set as the data source for the "DgvMainData" DataGridView control. Additionally, the sum of the

./admin/Employee.cs

Purpose:

The "Employee" class represents a form in a Windows Forms application that displays employee information. It allows the user to search and filter employees based on different criteria and perform operations such as deleting an employee.

Details:

Class Declaration: The class is declared as a public partial class, indicating it can be accessed from other parts of the application. It inherits from the "Form" class, providing basic window functionality.

Fields:

searchInput is a private field that stores the user's search input.

Constructor: The constructor is a method that initializes a new instance of the "Employee" class. It is responsible for initializing the components of the form using the "InitializeComponent()" method.

Employee_Load Event Handler: This event handler is triggered when the "Employee" form is loaded. It is associated with the "Load" event of the form. Inside this event handler, the "LoadDefaultFilterItems()" method is called to load the default filter items.

LoadDefaultFilterItems Method: The "LoadDefaultFilterItems()" method loads the default filter items into the "CbField" ComboBox control and the "CbPosition" ComboBox control. It adds the filter options "ID" and "Name" to the "CbField" ComboBox control and retrieves position names using the "GetPositionNames()" method from a "BSPosition" class (presumably a business service or data access class) and adds them to the "CbPosition" ComboBox control. The "All" option is added as the first item, and both ComboBox controls are initially set to the first item.

DgvMainData_CellClick Event Handler: This event handler is triggered when a cell in the "DgvMainData" DataGridView control is clicked. It is associated with the "CellClick" event of the DataGridView control. Inside this event handler, the selected employee's ID and name are displayed in the corresponding labels.

TbSearch_Leave Event Handler: This event handler is triggered when the focus leaves the "TbSearch" TextBox control. It is associated with the "Leave" event of the TextBox control. Inside this event handler, the text from the TextBox control is trimmed and stored in the "searchInput" field.

FilterEmployees Method: The "FilterEmployees()" method filters the employees based on the search input, selected field, and position. It creates an instance of the "BSEmployee" class (presumably a business service or data access class) and calls the "SearchEmployees()" method to retrieve the filtered employees based on the search input, selected field, and position. The retrieved data is then set as the data source for the "DgvMainData" DataGridView control.

BtnSearchIcon_Click Event Handler: This event handler is triggered when the "BtnSearchIcon" button is clicked. It is associated with the "Click" event of the button. Inside this event handler, the "FilterEmployees()" method is called to filter the employees based on the current criteria.

CbField_TextChanged Event Handler: This event handler is triggered when the text in the "CbField" ComboBox control changes. It is associated with the "TextChanged" event of the ComboBox control. Inside this event handler, the "FilterEmployees()" method is called to filter the employees based on the current criteria.

CbPosition_TextChanged Event Handler: This event handler is triggered when the text in the "CbPosition" ComboBox control changes. It is associated with the "TextChanged" event of the ComboBox control. Inside this event handler, the "FilterEmployees()" method is called to filter the employees based on the current criteria.

TbSearch_KeyDown Event Handler: This event handler is triggered when a key is pressed while the focus is on the "TbSearch" TextBox control. It is associated with the "KeyDown" event of the TextBox control. Inside this event handler, if the Enter key is pressed, the text from the TextBox control is stored in the "searchInput" field, and the "FilterEmployees()" method is called to filter the employees based on the current criteria.

BtnDelete_Click Event Handler: This event handler is triggered when the "BtnDelete" button is clicked. It is associated with the "Click" event of the button. Inside this event handler, it checks if an employee is selected for deletion. If not, it displays a message to select an employee. Otherwise, it creates an instance of the "BSEmployee" class and calls the "DeleteEmployee()" method to delete the selected employee. If the deletion is successful, a success message is displayed, and the employees are filtered again using the "FilterEmployees()" method.

./admin/Statistic.cs

The purpose of the Statistic class is to display and manage statistical data in a Windows Forms application. It provides functionality to initialize the data and user interface, as well as load and display statistical data based on the selected year.

Here are the details of the class:

Statistic class is defined as a partial class, indicating that it is part of a larger class implementation.

The class contains a private list variable filterYears to store a range of years for filtering the statistics data.

The class has a default constructor Statistic() which is responsible for initializing the class and calling the InitializeComponent() method.

The Statistic_Load event handler is triggered when the form is loaded. It is responsible for calling various methods to initialize data and UI, as well as load the statistical data.

The InitData() method initializes the filterYears list with a range of years from 2000 to the current year.

The InitUI() method is responsible for initializing the user interface. It calls the LoadDefaultDataOfYearFilter() method to load the default data into the CbYearFilter ComboBox control.

The LoadDefaultDataOfYearFilter() method takes an array of years as input and loads them into the CbYearFilter ComboBox control. It sets the selected index of the ComboBox control to the last year in the array.

The LoadStatisticData() method takes a year as input and loads the statistic data for that year. It checks if the year is within the valid range (2000 to the current year). If it is, it fills the cashReserveStatisticDataSet with the corresponding data using the cashReserveStatisticTableAdapter. Then, it refreshes the RWCashReserveStatistic report viewer to display the updated data. The method returns a boolean value indicating the success of loading the data.

The ConvertToInt() method takes a string as input and converts it to an integer using the Convert.ToInt32() method.

The CbYearFilter_TextChanged event handler is triggered when the text in the CbYearFilter ComboBox control changes. It is associated with the TextChanged event of the ComboBox control. Inside this event handler, the ConvertToInt() method is called to convert the selected year to an integer, and the LoadStatisticData() method is called to load the statistical data based on the selected year.

Booking

./booking/Booked.cs

Purpose:

The purpose of the Booked class is to provide functionality for viewing, filtering, and canceling booked tickets. It interacts with the user interface elements such as textboxes, dropdowns, and datagridviews to display and handle user input. The class also interacts with the BSBooked and BSPlace classes to retrieve and manipulate data related to booked tickets and place information.

Details:

The class inherits from the Form class in Windows Forms and is declared as public partial class Booked : Form.

It contains private fields searchInput, source, destination, and departureTime to store the current search input, selected source and destination places, and departure time.

The class constructor Booked() is responsible for initializing the form by calling InitializeComponent().

The Booked_Load event handler is triggered when the form is loaded and calls the LoadDefaultPlace() function to load default place options into the source and destination dropdowns.

The LoadDefaultPlace() function retrieves place names from the BSPlace class and populates the source and destination dropdowns with the retrieved data. It also sets the default selection to "All".

The FilterBookedTickets() function is called to filter and display booked tickets based on the selected criteria such as search input, source, destination, and departure time. It uses the BSBooked class to search for booked tickets and binds the retrieved data to the DgvMainData DataGridView.

The TbSearch_Leave event handler is triggered when the search textbox loses focus. It updates the searchInput field with the current text value.

The DtpDepartureTime_ValueChanged event handler is triggered when the departure time value changes. It updates the departureTime field with the new value and calls FilterBookedTickets() to update the displayed tickets based on the selected departure time.

The BtnSearchIcon_Click event handler is triggered when the search button is clicked. It calls FilterBookedTickets() to update the displayed tickets based on the selected criteria.

The DgvMainData_CellClick event handler is triggered when a cell in the DataGridView is clicked. It retrieves the selected ticket ID from the clicked row and updates the LbSelectedId label with the selected ID.

The CbSource_TextChanged event handler is triggered when the source dropdown value changes. It updates the source field with the selected value and calls FilterBookedTickets() to update the displayed tickets based on the selected source.

The CbDestination_TextChanged event handler is triggered when the destination dropdown value changes. It updates the destination field with the selected value and calls FilterBookedTickets() to update the displayed tickets based on the selected destination.

The TbSearch_KeyDown event handler is triggered when a key is pressed in the search textbox. If the pressed key is Enter, it updates the searchInput field with the current text value and calls FilterBookedTickets() to update the displayed tickets based on the entered search input.

The BtnCancel_Click event handler is triggered when the cancel button is clicked. It checks if a ticket is selected and displays a message if no ticket is selected. If a ticket is selected, it cancels the ticket using the pro_CancelTicket stored procedure and displays a success message. It then updates the displayed tickets by calling FilterBookedTickets().

./booking/Booking.cs

Purpose:

The purpose of the Booking class is to provide functionality for booking tickets, displaying trip information, selecting ticket types and seats, and managing the booking process. It interacts with the user interface elements such as textboxes, dropdowns, and buttons to handle user input and display relevant information. The class also interacts with the BSBooking, BSRoute, BSPlace, and BusManagementEntities classes to retrieve and manipulate data related to trips, tickets, and bookings.

Details:

The class inherits from the Form class in Windows Forms and is declared as public partial class Booking : Form.

It contains a private field parentForm of type App to store a reference to the parent form.

The class provides two constructors. The default constructor Booking() initializes the form by calling InitializeComponent(). The second constructor Booking(App parent) also initializes the form and sets the parentForm field with the provided parent form.

The Booking_Load event handler is triggered when the form is loaded and calls the following functions: LoadSelectedTrip(), LoadDefaultTicketType(), and LoadUserInfor().

The LoadUserInfor() function sets the user information (name, phone, and email) in the respective textboxes based on the UserData values.

The LoadDefaultTicketType() function populates the ticket type dropdown (CbType) with "Seat" and "Sleeper" options and selects the first item by default.

The LoadSelectedTrip() function retrieves the selected trip information using the BSBooking class and displays relevant data in the respective textboxes (TbFrom, TbTo, TbIDTrip, TbDistance, and TbDuration).

The LoadAvailableSeat(int type) function loads the available seats based on the selected ticket type (Seat or Sleeper) using the BSBooking class. It populates the seat dropdown (CbBookedSeat) with the available seats and selects the first item by default.

The LoadSelectedTicket() function retrieves the selected ticket information using the BSBooking class based on the selected seat and ticket type. It displays the ticket ID and fare in the respective textboxes (TbIDTicket and TbFare).

The CbBookedSeat_TextChanged event handler is triggered when the selected seat in the dropdown changes. It calls LoadSelectedTicket() to update the displayed ticket information based on the selected seat.

The CbType_TextChanged event handler is triggered when the selected ticket type in the dropdown changes. It calls LoadAvailableSeat() to update the available seats based on the selected ticket type.

The BtnCancel_Click event handler is triggered when the cancel button is clicked. It displays a confirmation dialog and if the user confirms, it resets the CurrentSelectedTripId in UserData and sets the MainFeatureIndex of the parent form to 4 (presumably for navigating to a specific feature).

The BtnBooking_Click event handler is triggered when the booking button is clicked. It validates the selected ticket and email input. If valid, it adds the booking to the database using the pro_AddDefaultBooking stored procedure and initiates the payment and confirmation logic. It also displays appropriate messages based on the success of the confirmation email sending. Finally, it reloads the available seats.

The SendConfirmEmail() function is responsible for sending a confirmation email to the provided email address. It retrieves the booked ticket information using the BSBooking class and uses an EmailSender object to send the email asynchronously. The email subject and message are generated using static methods from the StaticEnv class.

./booking/BookingNavigationBar.cs

Purpose:

The purpose of the BookingNavigationBar class is to provide navigation functionality for the booking feature. It allows the user to switch between different views or sections related to booking, such as booking a ticket and viewing booked tickets. The class tracks the current navigation index and communicates with the parent form (App) to update the main feature index accordingly.

Details:

The BookingNavigationBar class inherits from the Form class and implements the INavigationBar interface.

The class includes a private static field navIndex to store the current navigation index. It is initialized to 0.

There are two constructors available. The default constructor initializes the form by calling InitializeComponent(). The second constructor BookingNavigationBar(App parent) also initializes the form and sets the parentForm field with the provided parent form.

The class includes a static property NavIndex to get or set the value of the navIndex field.

The Handler_NavBtn_Click method is an event handler triggered when any of the navigation buttons (BtnBooking or BtnBooked) is clicked. It retrieves the clicked button's tag value, sets the navIndex to that value, and updates the MainFeatureIndex of the parent form to 4.

The PnlMainContainer_Paint method is an event handler triggered when the main container panel (PnlMainContainer) is being painted. It is responsible for drawing a horizontal line under the selected navigation button. Based on the current navIndex value, it determines the selected button and uses the Graphics.DrawLine method to draw a line at the bottom of the selected button to indicate the selection.

./booking/UnselectBooking.cs

This form is the empty form, it is shown when the user does not pick any trip.

Bus

./bus/Bus.cs

Details:

Fields and Constructor:

parentForm: A private field that represents the parent form (an instance of the App class).

type: A boolean field that represents the type of the bus (true for Interprovince and false for Transit).

Default constructor: Initializes the form components.

Constructor with parent: Initializes the form components and sets the parentForm field to the provided parent form.

Event Handlers:

BtnSearchIcon_Click: Handles the click event of the search icon button. Calls the SearchBus method with the text entered in the search textbox.

Bus_Load: Handles the load event of the Bus form. Calls the LoadDefault and LoadDefaultBusSearchItems methods to populate the DataGridView with default bus data and set the default values for the search comboboxes.

cbType_TextChanged: Handles the text change event of the type combobox. Updates the type field based on the selected type (Interprovince or Transit). Calls the SearchBus method with the current search input.

LoadDefault method:

Creates an instance of the BSBus class.

Retrieves all buses using the GetAllBus method from BSBus.

Binds the retrieved buses to the dgvMainData DataGridView for display.

LoadDefaultBusSearchItems method:

Initializes the bus type and search type arrays.

Adds the bus types ("Interprovince" and "Transit") to the cbType combobox.

Adds the search types ("ID of bus" and "Registration number") to the cbSearch combobox.

Sets the selected index of both comboboxes to 0 (the first option).

SearchBus method:

Refreshes the DataGridView to clear any previous search results.

Creates an instance of the BSBus class.

Based on the selected search type (cbSearch.SelectedIndex), calls the appropriate method from BSBus to search for buses by ID or registration number.

Binds the search results to the dgvMainData DataGridView for display.

Purpose:

The purpose of the Bus class is to provide functionality for searching and displaying bus data. It allows the user to search for buses by ID or registration number and filter them by bus type (Interprovince or Transit). The class interacts with the user through form components and handles events to perform searches and display the search results in the DataGridView.

Driver

./driver/Driver.cs

Details:

Fields and Constructor:

searchInput: A private string field that holds the search input entered by the user.

Default constructor: Initializes the form components.

Event Handlers:

Driver_Load: Handles the load event of the Driver form. Calls the LoadMainData and LoadDefaultFilterItems methods to populate the DataGridView with default driver data and set the default filter items.

DgvMainData_CellClick: Handles the cell click event of the DgvMainData DataGridView. Retrieves the selected driver's ID and name from the DataGridView and displays them in the corresponding labels.

BtnSearchIcon_Click: Handles the click event of the search icon button. Calls the FilterDrivers method to filter the drivers based on the selected filter field and search input.

TbSearch_Leave: Handles the leave event of the search textbox. Updates the searchInput field with the trimmed text from the search textbox.

TbSearch_KeyDown: Handles the key down event of the search textbox. If the Enter key is pressed, it updates the searchInput field with the text from the search textbox and calls the FilterDrivers method.

CbField_SelectedIndexChanged: Handles the selected index change event of the filter field combobox. Calls the FilterDrivers method to filter the drivers based on the selected filter field.

LoadMainData method:

Creates an instance of the BSDriver class.

Retrieves all drivers using the GetAllDrivers method from BSDriver.

Binds the retrieved drivers to the DgvMainData DataGridView for display.

LoadDefaultFilterItems method:

Initializes the filter items array with values "ID" and "License level".

Adds the filter items to the CbField combobox.

Sets the selected index of the combobox to 0 (the first option).

FilterDrivers method:

Creates an instance of the BSDriver class.

Retrieves the selected filter tag from the CbField combobox.

Calls the SearchDriver method from BSDriver with the search input and filter tag.

Binds the search results to the DgvMainData DataGridView for display.

Purpose:

The purpose of the Driver class is to provide functionality for searching and displaying driver data. It allows the user to search for drivers by ID or license level and filter them based on the selected filter field. The class interacts with the user through form components and handles events to perform searches and display the search results in the DataGridView.

Home

./home/Home.cs

Purpose:

The purpose of the Home class is to display a rotating slideshow of images representing different cities or destinations. It provides navigation buttons to move to the previous or next image in the slideshow. Additionally, it includes buttons for specific actions related to routes and finding information.

Details:

The Home class inherits from the Form class.

The class includes a private field `images` of type `List<Image>` to store a collection of images representing different cities or destinations. The images are added to the list in the constructor using the `Properties.Resources` syntax.

There is an additional private field `currentImageIndex` to keep track of the currently displayed image in the slideshow. It is initialized to 0.

The default constructor initializes the form by calling `InitializeComponent()`.

The class includes event handlers for various button clicks (`btnRoute1_Click`, `btnFind_Click`, `btnPrevious_Click`, `btnNext_Click`), but the implementation for these event handlers is not provided in the code snippet.

The `Home_Load` event handler is triggered when the form is loaded. It sets the initial image in the picture box (`pbHome`) to the image at the `currentImageIndex` position in the images list.

The `btnPrevious_Click` event handler is triggered when the "Previous" button is clicked. If the `currentImageIndex` is greater than 0, it decrements the index and updates the image in the picture box.

The `btnNext_Click` event handler is triggered when the "Next" button is clicked. If the `currentImageIndex` is less than the count of images minus 1, it increments the index and updates the image in the picture box.

Login

./Login.cs

The Login class is a Windows Forms form that represents a login screen for a bus management application. Here is a detailed description of the class:

The Login class inherits from the Form class.

The default constructor initializes the form by calling InitializeComponent().

The DrawUI method is used to draw the user interface elements. It sets the background image for the login button and draws a rectangle with a border on the main container panel.

The setBackgroundImageBtnLogin method is responsible for setting the background image of the login button (BtnLogin) using a linear gradient brush. The angle parameter specifies the angle of the gradient, and the default value is 0.

The BtnLogin_MouseEnter and BtnLogin_MouseLeave event handlers are triggered when the mouse enters or leaves the login button. They change the background image of the button to create a hover effect.

The Handler_UsernameInput_Click and Handler_PasswordInput_Click event handlers are triggered when the corresponding labels are clicked. They set the focus to the respective text boxes (TbUsername and TbPassword).

The Handler_ShowPlaceholder method is used to show or hide the placeholder labels (LbPlaceholderUsername and LbPlaceholderPassword) based on whether the text boxes are focused or empty.

The TbUsername_GotFocus and TbUsername_LostFocus event handlers are triggered when the username text box gains or loses focus. They call the Handler_ShowPlaceholder method to handle the visibility of the placeholder label.

The TbPassword_GotFocus and TbPassword_LostFocus event handlers are triggered when the password text box gains or loses focus. They also call the Handler_ShowPlaceholder method.

The Handler_Focus event handler is triggered when any control within the main container panel (PnlMainContainer) receives focus. It sets the focus to the panel itself to remove focus from other controls.

The `TbPassword_PreviewKeyDown` event handler is triggered when a key is pressed while the password textbox has focus. If the Enter key is pressed, it simulates a click on the login button (`BtnLogin`).

The `Handler_LoginSuccessfully` method is called when the login process is successful. It sets the `DialogResult` of the form to `DialogResult.OK` and closes any other forms that are contained within the Login form.

The `BtnLogin_Click` event handler is triggered when the login button is clicked. It retrieves the entered username and password from the text boxes (`TbUsername` and `TbPassword`), validates them using the `BSLogin` class, and determines whether the login is for a passenger or an employee. Based on the login result, it sets the appropriate user data using the `UserData` class and calls the `Handler_LoginSuccessfully` method.

The `PnlMainContainer_Paint` event handler is empty in the provided code snippet.

The `RenderActiveForm` method is used to render an active form within a specified control. It creates the specified form, sets its properties, and adds it as a child control to the given container control.

The `LbSignUp_Click` event handler is triggered when the "Sign Up" label (`LbSignUp`) is clicked. It renders an instance of the `SignUp` form using the `RenderActiveForm` method.

The `ChbShow_CheckedChanged` event handler is triggered when the "Show Password" checkbox (`ChbShow`) is checked or unchecked. It changes the `PasswordChar` property of the password text box (`TbPassword`) to either display the password characters or hide them.

./SignUp.cs

The `SignUp` class is a Windows Forms form that represents a sign-up screen for the bus management application. Here is a detailed description of the class:

The `SignUp` class inherits from the `Form` class.

The default constructor initializes the form by calling `InitializeComponent()`.

The DrawUI method is used to draw the user interface elements. It sets the background image for the sign-up button and draws rectangles with borders on various panels (PnlUsername, PnlPassword, PnlFullName, PnlPhone). It also sets a background image for these panels with a horizontal line at the bottom.

The SetBackgroundImageBtnLogin method is responsible for setting the background image of the sign-up button (BtnSignUp) using a linear gradient brush. The angle parameter specifies the angle of the gradient, and the default value is 0.

The RenderActiveForm method is similar to the one in the Login class. It is used to render an active form within a specified control.

The BtnSignUp_MouseEnter and BtnSignUp_MouseLeave event handlers are triggered when the mouse enters or leaves the sign-up button. They change the background image of the button to create a hover effect.

The DrawHorizontalBar method is used to draw a horizontal line at the bottom of the specified control. It creates a background image for the control and draws a line using a graphics object.

The LbLogin_Click event handler is triggered when the "Login" label (LbLogin) is clicked. It renders an instance of the Login form using the RenderActiveForm method.

The Handler_Click event handler is triggered when any control within the main container panel (PnlMainContainer) receives focus. It sets the focus to the panel itself to remove focus from other controls.

The BtnSignUp_Click event handler is triggered when the sign-up button is clicked. It retrieves the entered username, password, fullname, and phone from the text boxes (TbUsername, TbPassword, TbFullName, TbPhone). It checks if all the fields are filled, and if not, it displays an error message. Otherwise, it uses the BSLogin class to create a new user, passing the necessary information. The method returns a boolean value indicating whether the user creation was successful, and the error message is updated accordingly.

The ChbShow_CheckedChanged event handler is triggered when the "Show Password" checkbox (ChbShow) is checked or unchecked. It changes the PasswordChar property of the password text box (TbPassword) to either display the password characters or hide them.

Profile

./profile/Profile.cs

The purpose of the Profile class is to provide functionality for managing and updating user profile information in a bus management application. It is a Windows Forms form that allows users to view and modify their profile details.

Here is a breakdown of the class:

The class inherits from the Form class, indicating that it represents a form in the user interface.

The constructor Profile() is responsible for initializing the form. It calls InitializeComponent() to set up the form's components and user interface elements. The line System.Windows.Input.Keyboard.IsKeyDown(Key.A); does not seem to have any direct relevance to the profile functionality and could be removed.

The Profile_Load event handler is triggered when the profile form is loaded. It calls the LoadUserInfor method to populate the user information fields with the data stored in the UserData class.

The ToggleControls method takes a list of controls and toggles the Enabled property of each control in the list. This method is used to enable or disable a group of controls together.

The BtnUpdate_Click event handler is triggered when the "Update" button (BtnUpdate) is clicked. It checks if the fullname text box (TbFullname) is enabled, which indicates that the user is in editing mode.

Inside the editing mode block, it validates the entered email, fullname, phone, and gender. If the email is provided, it checks if it is in the correct format by verifying if it contains "@gmail.com". If any of the required fields (fullname and phone) are empty, it displays an error message using a message box and returns.

Depending on whether the user is an employee or a passenger (determined by the UserData.IsPassenger property), it calls the corresponding method from the BSPassenger class to update the user information in the database (UpdateEmployee or UpdatePassenger).

If the update operation fails (the response value is false), it displays an error message using a message box and returns. Otherwise, it calls the SetUserInfo method to update the user information stored in the UserData class and displays a success message using a message box.

The LoadUserInfo method retrieves the user information from the UserData class and populates the corresponding text boxes and radio buttons on the form with the retrieved values.

The SetUserInfo method takes the updated user information (fullname, phone, email, and gender) as parameters and updates the corresponding properties in the UserData class.

Setting

./setting/ChangePassword.cs

Details:

The class is named ChangePassword and is defined as a partial class.

It extends the Form class.

Methods and Event Handlers:

ChangePassword(): This is the default constructor for the ChangePassword class. It initializes the form components by calling the InitializeComponent() method.

BtnUpdate_Click(): This method is an event handler for the click event of the "Update" button (BtnUpdate). It retrieves the values entered in the current password, new password, and retype new password textboxes. It performs validation checks to ensure that all fields are filled. If any field is empty, it displays an error message and returns. It then checks if the entered current password matches the stored password in the UserData object. If not, it displays an error

message and returns. It further checks if the new password matches the retype new password. If not, it displays an error message and returns. Finally, it uses an instance of the BSLogin class to change the user's password by calling the ChangeUserPassword method. If the password change is successful, it updates the stored password in the UserData object, displays a success message, and clears the input fields.

ClearInput(): This method is used to clear the input fields by setting their text to empty strings.

BtnCancel_Click(): This method is an event handler for the click event of the "Cancel" button (BtnCancel). It closes the form by calling the Close() method.

Purpose:

The purpose of the ChangePassword class is to provide a form for users to change their password. The class handles the logic for validating the input, checking the correctness of the current password, matching the new password with the retype new password, and updating the password in the UserData object. It also provides functionality to display notifications and clear the input fields. The class allows users to update their password securely within the application.

Staff

./staff/reports/BookedTicketPassengerDataSet.xsd

./staff/reports/BookedTicketPassengerStatistic.rdlc

./staff/Passenger.cs

Details:

The class is named `Passenger` and is defined within the `BusTicketManagementApplication.src.layers.interfaceLayers.components.staff` namespace.

It extends the `Form` class, indicating that it is a Windows Forms form.

The class contains private fields, including `searchInput`, which stores the search input provided by the user.

Methods and Event Handlers:

`Passenger()`: This is the constructor for the `Passenger` class. It initializes the form components by calling the `InitializeComponent()` method.

`Passenger_Load()`: This event handler is triggered when the form is loaded. It loads data into the `PASSENGER` table and calls the `LoadDefaultFilterItems()` method.

`LoadDefaultFilterItems()`: This method populates the filter combo box (`CbField`) with options for filtering passengers by ID or Name.

`DgvMainData_CellClick()`: This event handler is triggered when a cell in the `DataGridView` (`DgvMainData`) is clicked. It updates the selected passenger's ID and name labels based on the clicked row.

`FilterPassengers()`: This method filters the passenger data based on the search input and the selected filter option. It uses the `BSPassenger` class to perform the search and updates the `DataGridView` with the search results.

`BtnSearchIcon_Click()`: This event handler is triggered when the search icon button is clicked. It calls the `FilterPassengers()` method to perform the passenger filtering.

`TbSearch_Leave()`: This event handler is triggered when the focus leaves the search text box (`TbSearch`). It updates the `searchInput` variable with the trimmed text of the search box.

`TbSearch_KeyDown()`: This event handler is triggered when a key is pressed while the search text box has focus. If the Enter key is pressed, it updates the `searchInput` variable with the current text and calls the `FilterPassengers()` method.

CbField_TextChanged(): This event handler is triggered when the text in the filter combo box (CbField) is changed. It is currently not implemented.

CbField_SelectedIndexChanged(): This event handler is triggered when the selected item in the filter combo box is changed. It calls the FilterPassengers() method to perform the passenger filtering.

Purpose:

The purpose of the Passenger class is to provide functionality for displaying and filtering passenger data in the bus ticket management application. It allows the user to search for passengers based on their ID or name and updates the DataGridView with the search results. The class interacts with the BSPassenger class to perform the actual search operations. Additionally, it provides event handlers for various user interactions, such as clicking the search icon, selecting filter options, and clicking on cells in the DataGridView to view details of selected passengers.

./staff/PassengerStatistic.cs

Details:

The class is named PassengerStatistic and is defined as a partial class.

It extends the Form class, indicating that it is a Windows Forms form.

Methods and Event Handlers:

PassengerStatistic(): This is the constructor for the PassengerStatistic class. It initializes the form components by calling the InitializeComponent() method.

PassengerStatistic_Load(): This event handler is triggered when the form is loaded. It loads data into the BOOKEDTICKETPASSENGER table using the BOOKEDTICKETPASSENGERTableAdapter. It then refreshes the report viewer (RVPassengerStatistic) to display the passenger statistics report.

Purpose:

The purpose of the PassengerStatistic class is to display a form that shows passenger statistics. It loads data into the BOOKEDTICKETPASSENGER table and uses a report viewer (RVPassengerStatistic) to display the passenger statistics report. The class provides the PassengerStatistic_Load() event handler to perform these operations when the form is loaded.

./staff/StaffNavigationBar.cs

Details:

The class is named StaffNavigationBar and is defined as a partial class.

It extends the Form class and implements the INavigationBar interface.

Properties:

navIndex: This is a static property of type int that represents the current navigation index. It is accessed through the NavIndex property.

Methods and Event Handlers:

StaffNavigationBar(): This is the default constructor for the StaffNavigationBar class. It initializes the form components by calling the InitializeComponent() method.

StaffNavigationBar(App parent): This is a constructor that takes an instance of the App class as a parameter. It initializes the form components and sets the parentForm field to the provided App instance.

Handler_NavBtn_Click(): This method is an event handler for the click event of the navigation buttons. It updates the navIndex property based on the clicked button's tag, and sets the MainFeatureIndex property of the parentForm to 6 (a virtual number).

PnlMainContainer_Paint(): This method is an event handler for the Paint event of the main container panel (PnlMainContainer). It is responsible for drawing a line at the bottom of the selected navigation button. It determines the selected button based on the navIndex value and uses the Graphics object to draw a line on the panel.

Purpose:

The purpose of the StaffNavigationbar class is to provide a navigation bar for the staff interface. It allows the user to navigate between different features of the application. The class tracks the current navigation index through the navIndex property and updates the MainFeatureIndex property of the parent form accordingly. It also handles the visual representation of the selected navigation button by drawing a line at the bottom of the button.

Trip

./trip/Trip.cs

Details:

Fields and Constructor:

parentForm: A private field that represents the parent form (an instance of the App class).

searchInput: A private field that stores the search input provided by the user.

source: A private field that stores the selected source place.

destination: A private field that stores the selected destination place.

departureTime: A private field that stores the selected departure time. It is initially set to January 1, 1990.

Default constructor: Initializes the form components.

Constructor with parent: Initializes the form components, sets the parentForm field to the provided parent form, and hides the "BtnBooking" button if the current user is not a passenger.

Trip_Load event handler:

Triggered when the Trip form is loaded.

Calls the LoadDefaultPlace method to populate the source and destination comboboxes with default place names.

LoadDefaultPlace method:

Retrieves default place names using the GetPlaceNames method from the BSPlace class.

Adds the "All" option to both the source and destination comboboxes.

Adds the default place names to the comboboxes and sets the selected index to 0 (which is the "All" option).

FilterTrips method:

Creates an instance of the BSTrip class.

Based on the current state of the parentForm and StaffNavigationbar.NavIndex, it calls the appropriate method from BSTrip to search for and retrieve trips.

Binds the resulting trips to the DgvMainData DataGridView for display.

Event Handlers:

TbSearch_Leave: Updates the searchInput field with the text entered in the search textbox.

DtpDepartureTime_ValueChanged: Updates the departureTime field with the selected value from the departure time DateTimePicker. Calls FilterTrips to apply the updated filters.

BtnSearchIcon_Click: Calls FilterTrips to apply the filters and search for trips.

BtnBooking_Click: Retrieves the ID of the selected trip from the DataGridView and sets it as the CurrentSelectedTripId in the UserData class. Sets the MainFeatureIndex in the parentForm to 4 (indicating the booking feature).

DgvMainData_CellClick: Retrieves the selected ID from the DataGridView and displays it in the LbSelectedId label.

CbSource_TextChanged: Updates the source field with the selected source from the source combobox. Calls FilterTrips to apply the updated filters.

CbDestination_TextChanged: Updates the destination field with the selected destination from the destination combobox. Calls FilterTrips to apply the updated filters.

TbSearch_KeyDown: Handles the Enter key press event in the search textbox. Updates the searchInput field and calls FilterTrips to apply the filters and search for trips.

Purpose:

The purpose of the Trip class is to provide functionality for searching and displaying trips. It interacts with the user through the form's components and handles events to update the search criteria and retrieve the corresponding trips from the database.

- controllers

./interfaces/ISender.cs

ISender interface publishes functions to send email

```
1 reference
internal interface ISender
{
    2 references
    Task SendEmailAsync(string email, string subject, string body);
}
```

./interfaces/INavigationBar.cs

INavigationBar interface publishes functions to control navigation bar

```
internal interface INavigationBar
{
    11 references
    void Handler_NavBtn_Click(object sender, EventArgs e);
    6 references
    void PnlMainContainer_Paint(object sender, PaintEventArgs e);
}
```

./EmailSender.cs

EmailSender.cs class provides functions to send email

```

internal class EmailSender : IEmailSender
{
    2 references
    public Task SendEmailAsync(string email, string subject, string body)
    {
        string host = "smtp.gmail.com";
        int port = 465;
        string username = LocalEnv.EmailServerName;
        string password = LocalEnv.EncodedEmailServerPassword;
        //
        MimeMessage message = new MimeMessage();
        message.From.Add(new MailboxAddress("FUTA Bus Lines", "noreply03@futa.vn"));
        message.To.Add(MailboxAddress.Parse(email));
        message.Subject = subject;
        //
        BodyBuilder builder = new BodyBuilder();
        builder.HtmlBody = body;
        message.Body = builder.ToMessageBody();
        //
        SmtplibClient smtpClient = new SmtplibClient();
        //
        smtpClient.Connect(host, port, true);
        smtpClient.Authenticate(username, password);
        return smtpClient.SendAsync(message);
    }
}

```

- customizeUI

The following classes used to initialize UI of the application

./initialUI/InitUI.cs

./initialData.cs

- Data

./UserData.cs

this class used to store user's information when they logged and provides some function to help set/get user's data easier.

65 references

```
internal class UserData
{
    private static bool islogin = false;
    private static string username;
    private static string password;
    private static string passengerId = string.Empty;
    //
    private static bool isAdmin = false;
    private static bool isStaff = false;
    private static bool isPassenger = true;
    private static string systemId = string.Empty;
    //
    //
    private static string phone;
    private static string email;
    private static string address;
    private static bool? gender; // 0: male, 1: female
    private static string identity_number;
    private static DateTime birthday;
    //
    private static string fullName;
    //
    private static string currentSelectedTripId = string.Empty;
    //
}
```

```

public static void ClearUserData()
{
    islogin = false;
    username = string.Empty;
    password = string.Empty;
    passengerId = string.Empty;
    //
    isAdmin = false;
    isStaff = false;
    isPassenger = false;
    systemId = string.Empty;
    //
    fullName = string.Empty;
    phone = string.Empty;
    email = string.Empty;
    address = string.Empty;
    gender = null;
    identity_number = string.Empty;
    birthday = DateTime.Now;
    //
    currentSelectedTripId = string.Empty;
}

```

2 references

```

public static void SetUserLoginData(string username, string password)...

```

1 reference

```

public static void SetPassengerId(string passengerId)...

```

2 references

```

public static void SetUserData(string fullName, string phone)...

```

3 references

```

public static string GetPassengerId()...
//

```

1 reference

```

public static void SetSystemId(string systemId)...

```

2 references

```

public static string GetSystemId()...

```

CHAPTER IV: SYSTEM INTERFACE DESIGN

(User perspective)

1. Applications and services used

This part contains all the applications used during the making of this project.

- Microsoft SQL Server 2022
- Microsoft SQL Server Management Studio 19 (SSMS 19)
- Windows Forms App (.NET Framework) built using Visual Studio 2022
- Packages: Entity Framework, Mailkit, Mimekit, BouncyCastle.Cryptography

2. Software interface

Step by step from the very beginning:



2.1. User function

* Login & Sign up page

This is where users will have to input their username and password in order to use the software's booking functions or access the database in general.

When admins or employees input their account (username and password) which is provided by system, system gives them their privileges

In the case a user does not possess an account to use the software, they can sign up for a new account after filling out certain information criteria.

Login	Sign Up
<p>Username</p> <p></p>	<p>Username</p>
<p>Password</p> <p> Type your password</p> <p><input type="checkbox"/> Show password</p> <p>Forgot password?</p>	<p>Password</p>
<p>Full Name</p>	<p>Full Name</p>
<p>Phone</p>	<p>Phone</p>
<p><input type="checkbox"/> Show password</p>	<p><input type="checkbox"/> Show password</p>
<p>Login</p>	<p>Sign Up</p>
<p>Sign Up</p>	<p>Login</p>

*** Home screen**

This form shows some information and pictures about the bus system and popular routes

When logged

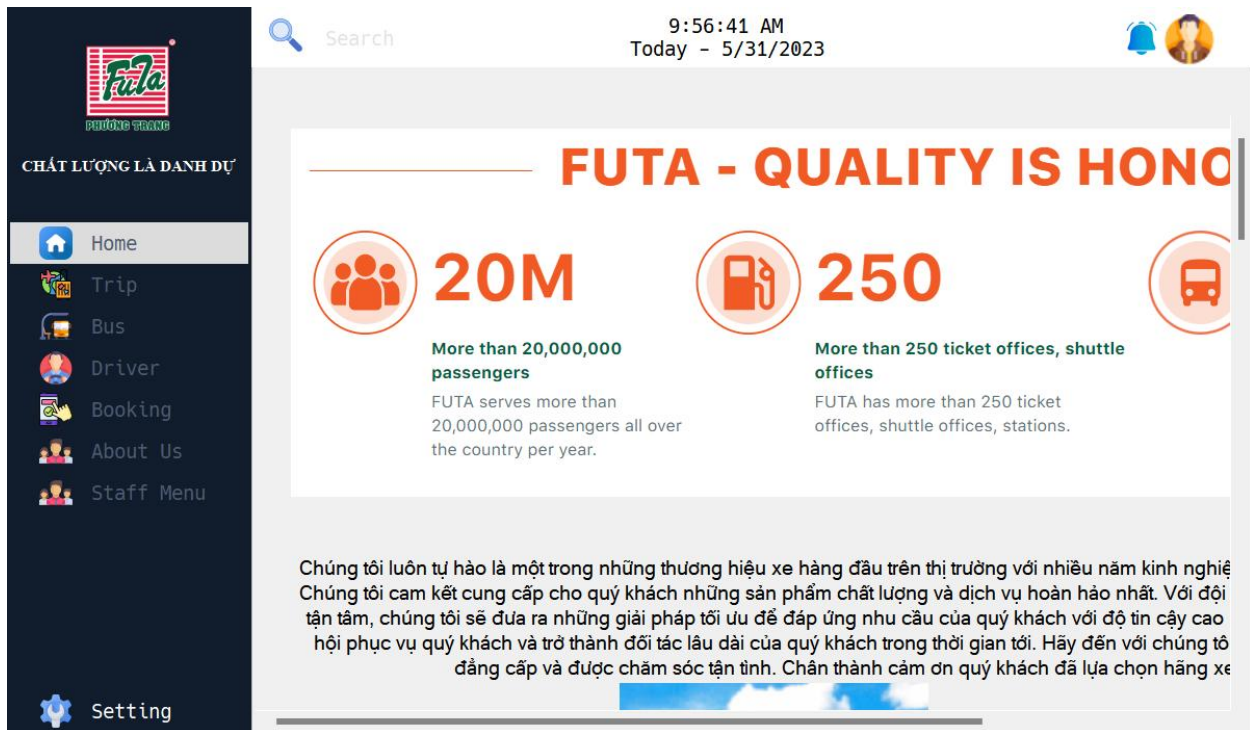
- For passengers



- For server admins



- For employee



* Change password screen

For the user's password change purpose, we supply this form (Setting -> Change password). The new password (at least 6 character, only [a-zA-Z0-9]) in a specific criteria.

8:18:31 PM
Today - 04/26/2023

Change Password

Current password: *****

New password: *****

Retype new password: *****

Save Cancel

* Profile screen

For the user's information change purpose, we supply this form (Avatar -> Profile).


The user can change their information including: name, phone, email, gender.

The screenshot shows the profile screen of the Futa mobile application. On the left is a dark blue sidebar with the Futa logo and navigation options: Home, Trip, Bus, Driver, Booking, About Us, and Setting. The main area is light gray and displays the user's profile information. At the top right, there is a search bar, the time (8:02:33 PM), and the date (Today - 04/26/2023). Below the time is a notification bell icon and a user profile icon. The profile information includes a circular avatar, Full name (nhan), Phone (123456789), Email (nhanpham@gmail.com), and Gender (Male selected, Female unselected). There is an 'Update' button at the bottom.

*Trip screen

In this form, the application will inform the users about the trips which are available and passengers can book them. It shows ID of trip, registration number of bus, start and end location, departure time, duration, seat number and status of trip.

- For passengers



CHẤT LƯỢNG LÀ DANH DỰ

- Home
- Trip**
- Bus
- Driver
- Booking
- About Us
- Setting

7:00:30 PM
Today - 04/24/2023

ID: tri_0000000009

From

To

Departure time


All

All

24/ 4/2023

Trip_ID	Registration_r	Start_point	End_point	Departure_tim	Duration	Remain_seat	Stat
tri_0000000009	58B-456.32 ...	Lâm Đồng	Lâm Đồng	04/25/2023 ...	10	7	finis
tri_0000000010	14P-678.90 ...	Lâm Đồng	An Giang	04/26/2023 ...	15	23	can

Trip screen (for admin and employee)



CHẤT LƯỢNG LÀ DANH DỰ

- Home
- Trip**
- Bus
- Driver
- Booking
- About Us
- Staff Menu
- Admin Menu
- Setting

9:58:23 AM
Today - 5/31/2023

ID:

From

To

Departure time

All

All

31/ 5/2023

Trip_ID	Registra	Start_po	End_poin	Departur	Duration	Remain_s	Status
tri_0...	58B-4...	Lâm Đồng	Lâm Đồng	8/25/...	10	4	finis...
tri_0...	14P-6...	Lâm Đồng	An Giang	10/26...	15	19	cance...

* Bus screen


Bus form show the bus information helps users who want to research information of the bus in their booked trip. They can search by id or registration number of the bus. In addition, there are the filter features to help users filter their results more accurately.

The screenshot displays the 'Bus' screen of the Fula application. On the left is a dark sidebar menu with icons and labels for 'Home', 'Trip', 'Bus' (highlighted), 'Driver', 'Booking', 'About Us', and 'Setting'. The main content area has a top header with a search icon, the time '7:06:11 PM', and the date 'Today - 04/24/2023'. Below this is a search section with a 'Search by:' dropdown set to 'ID of bus' and a 'Filter:' dropdown set to 'Interprovince'. A table lists bus details with columns: id_bus, registration_nun, model, capacity, status, type, and TRIPs. The first row is highlighted in blue.

id_bus	registration_nun	model	capacity	status	type	TRIPs
bus_0000000001	78-FN10532	Suzuki	30	idle	<input type="checkbox"/>	
bus_0000000002	78-FN15642	Suzuki	30	idle	<input type="checkbox"/>	
bus_0000000003	78-FN17253	Suzuki	16	idle	<input checked="" type="checkbox"/>	
bus_0000000004	29A-102.43	Toyota	27	idle	<input type="checkbox"/>	
bus_0000000005	51G-987.65	Honda	34	idle	<input type="checkbox"/>	
bus_0000000006	34M-115.78	Mercedes-...	20	idle	<input checked="" type="checkbox"/>	
bus_0000000007	72F-963.24	BMW	40	idle	<input type="checkbox"/>	
bus_0000000008	43N-789.01	Audi	31	idle	<input checked="" type="checkbox"/>	
bus_0000000009	58B-456.32	Hyundai	26	idle	<input checked="" type="checkbox"/>	
bus_0000000010	14P-678.90	Kia	33	idle	<input checked="" type="checkbox"/>	


* Driver screen

Driver screen illustrates the driver information included: ID, name of driver, phone number, gender, license level, type of driver (interprovince or transit) and state of driver.






CHẤT LƯỢNG LÀ DANH DỰ

- Home
- Trip
- Bus
- Driver**
- Booking
- About Us
- Staff Menu
- Admin Menu
- Setting



1:14:20 PM
Today - 05/04/2023



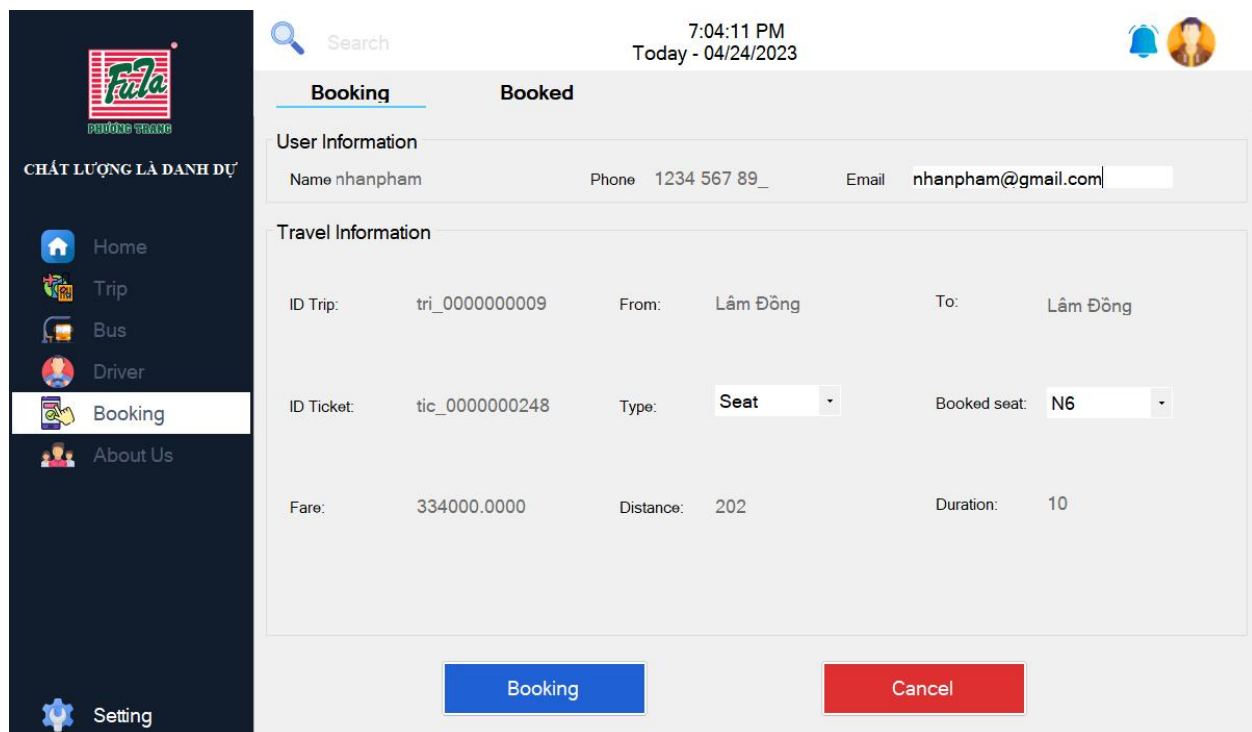
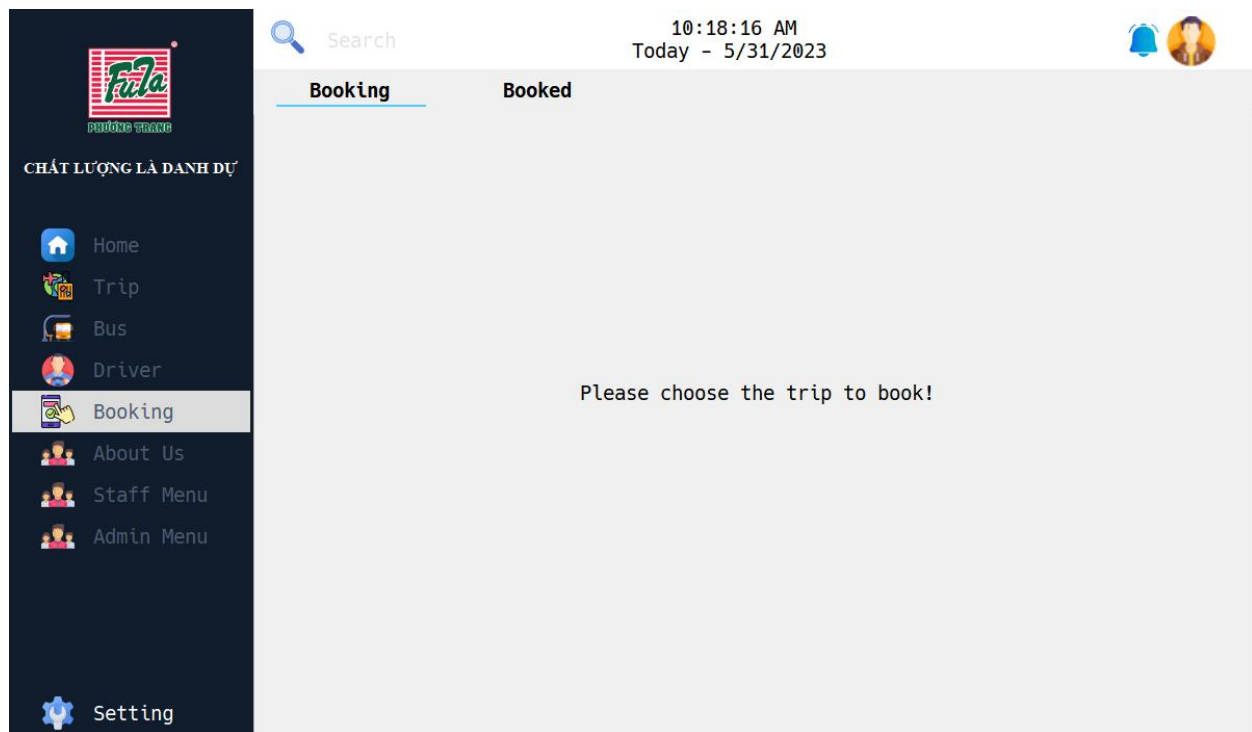
ID:
Lisence level:
ID

Employees_ID	Name	Phone_Number	Gender	Lisence_Level	Type	State
emp_00000000...	Lucky D	0888888888 ...	True	F	Interprovince	Idle

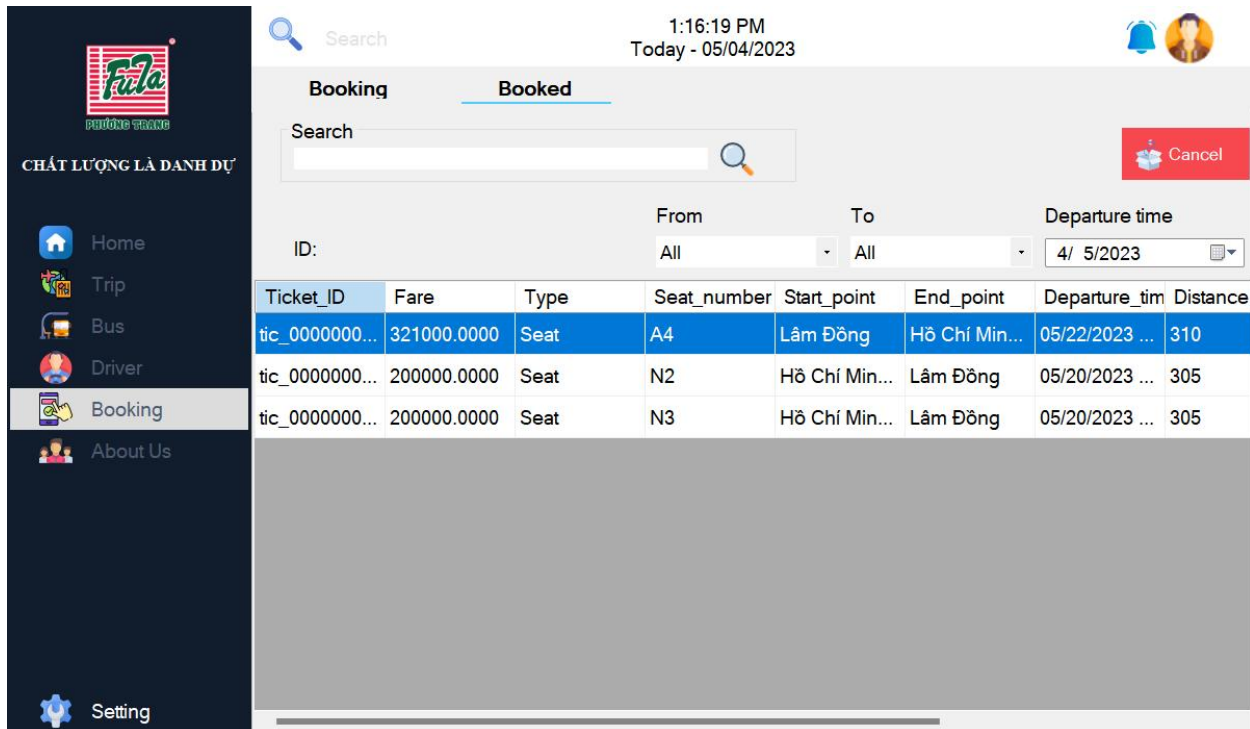
* Booking & Booked screen

There is a place where the user can book the ticket and check again the booked trip.

When user have not picked any trip yet:



After users verify their tickets, they will appear at Booked tab in the following picture:



*** Email received to confirm after successful booking**

When a user booked a trip, the system will send an email to confirm.

Dear: Dear Valued Guest nhan,

Congratulations on your successful booking on the FUTA Bus Lines system.
Your ticket number **tic_0000000029** booked on 04/26/2023 8:03:20 PM with the form of payment is VatoPay .



Contact Info:
nhan
Email: nhanpham@gmail.com
Tel: 1234 567 89

Trip information:

Hồ Chí Minh (TP) to Lâm Đồng	Ticket price: 200000.0000 VND
Hours: 3:30:00 PM	Number of tickets: 1
Departure date: Tuesday, June 20, 2023	Total: 200000.0000 VND
Number of seats: N3	
Boarding point: Thu Duc office: 798 XLHN , Hiep Phu Ward, District 9, HCMC	
Into money: 200000.0000 VND	
Discount (Voucher): 0 VND	
Total: 200000.0000 VND	
(Free drinking water, cold towels, Wi-Fi, TV)	

You can use the following QR code to check-in:



Contact Info:
nhan
Route: Lâm Đồng
Hours: 3:30:00 PM
Departure date: Tuesday, June 20, 2023
Seat: N3

[Download QR code](#)

Note before boarding

- Please bring your email containing the ticket code to the office to redeem your ticket at least 60 minutes before departure time.
- For routes from Ho Chi Minh City to Western provinces, please be at the office **231 - 233 Le Hong Phong** at least 60 minutes before departure time for us to transfer.
- For routes from Ho Chi Minh City to Mui Ne, Nha Trang, please be at **272 De Tham** office at least 60 minutes before departure time for us to transfer.
- For the route from Ho Chi Minh City to Da Lat, please be at the office at **231 - 233 Le Hong Phong** at least 60 minutes before the departure time for us to transfer, guests to the BXMT will be present 60 minutes before.
- For routes from BXM, please be at the station for at least 60 minutes. • Passenger information must be correct, otherwise it will not be possible to board the bus or cancel/change tickets.
- You are not allowed to change / return tickets on New Year's Day (weekdays you are entitled to change or cancel tickets only once before 24 hours), 10% cancellation fee.

Need further assistance?

If you have any questions, please call the FUTA Bus Lines support hotline **1900 6067** . Or leave a message in the **Contact** section of the website futabus.vn .

Phuong Trang Passenger Car Joint Stock Company FUTA Bus Lines

Address: 80 Tran Hung Dao, District 1, Ho Chi Minh City
Phone: 08 3838 6852 - Fax: 08 3838 6853
Email: hotro@futabus.vn


* About Us screen

This form has three tabs (About Us, Bus Station and Bus Route).


In the first tab, it introduces the bus system and some routes.




In the second tab, it shows information about all of bus station the bus system




CHẤT LƯỢNG LÀ DANH DỰ




Home




Trip




Bus




Driver




Booking




About Us




Staff Menu



Admin Menu




Setting




Search

1:17:42 PM

Today - 05/04/2023






About Us

Bus Station


Bus Route


Phone_number	Address	Region
0262 393 6868	172 Lê Duẩn, TP Buôn Ma Thuột, ...	Đắk Lắk
02613 67 67 67	226 Hai Bà Trưng, Nghĩa Thành, ...	Đắk Nông
02913 93 2345	QL1A, Khóm 2, P.7, TP.Bạc Liêu, ...	Bạc Liêu
02753646464	Đường Võ Nguyên Giáp, Quốc lộ ...	Bến Tre
02903 651 651	309 Lý Thường Kiệt, P.6, TP.Cà M...	Cà Mau
0283 511 9808	292 Đinh Bộ Lĩnh, phường 26, Bìn...	Hồ Chí Minh (TP)
02633 651 651	695-697, QL20 Liên Nghĩa, H.Đức ...	Lâm Đồng
02633 788 799	735 Hùng Vương, TT.Di Linh, H.D...	Lâm Đồng
02633 731 731	280 Trần Phú, TX.Bảo Lộc, Lâm ...	Lâm Đồng
02973 66 88 66	QL80, KP 5, P.Bình San, TX.Hà Ti...	Kiên Giang
02973 769 768	397 QL 80, KP Ngã ba, TT.Kiên L...	Kiên Giang
02973 699 688	QL 80, Tổ 3, KP Kiên Tân, TT.Kiê...	Kiên Giang
02933 868 866	BX Ngã 7, P.Ngã Bảy, TX.Ngã Bả...	Hậu Giang
02773 898 777	Ngã 4 Võ Văn Kiệt - Điện Biên Ph...	Đồng Tháp


The last tab will show information about all of the routes.





CHẤT LƯỢNG LÀ DANH DỰ


 Home


 Trip


 Bus


 Driver


 Booking

 About Us



 Staff Menu

 Admin Menu

 Setting

 Search

1:17:56 PM
Today - 05/04/2023

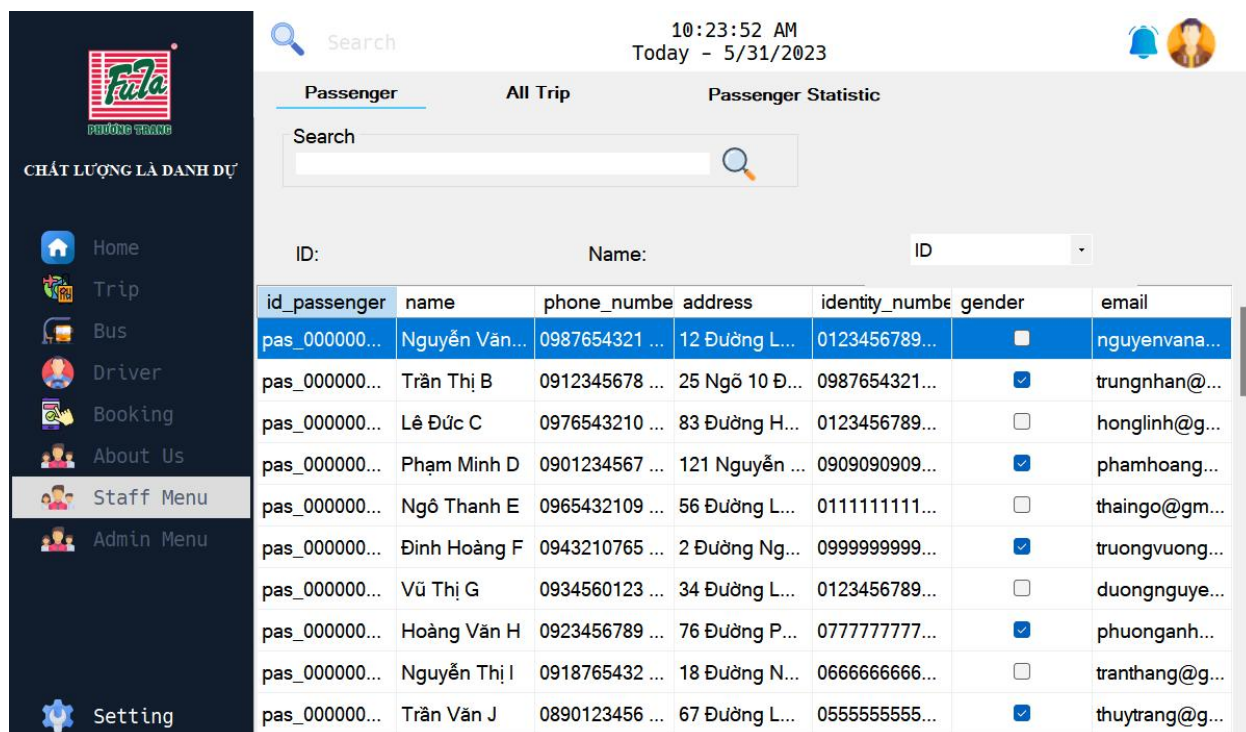


About Us	Bus Station	Bus Route
Start_point	End_point	Distance
Hồ Chí Minh (TP)	Lâm Đồng	305
Lâm Đồng	Hồ Chí Minh (TP)	310
An Giang	Hồ Chí Minh (TP)	240
Đắk Lắk	Lâm Đồng	349
Lâm Đồng	Đắk Nông	242
Bạc Liêu	Lâm Đồng	240
Lâm Đồng	Bến Tre	87.6
Kiên Giang	Lâm Đồng	236
Lâm Đồng	Lâm Đồng	202
Lâm Đồng	An Giang	215

2.2. Employee function (only for employee account)

Passenger screen

The passenger tab shows the passenger's information for the employee.



10:23:52 AM
Today - 5/31/2023

Passenger All Trip Passenger Statistic

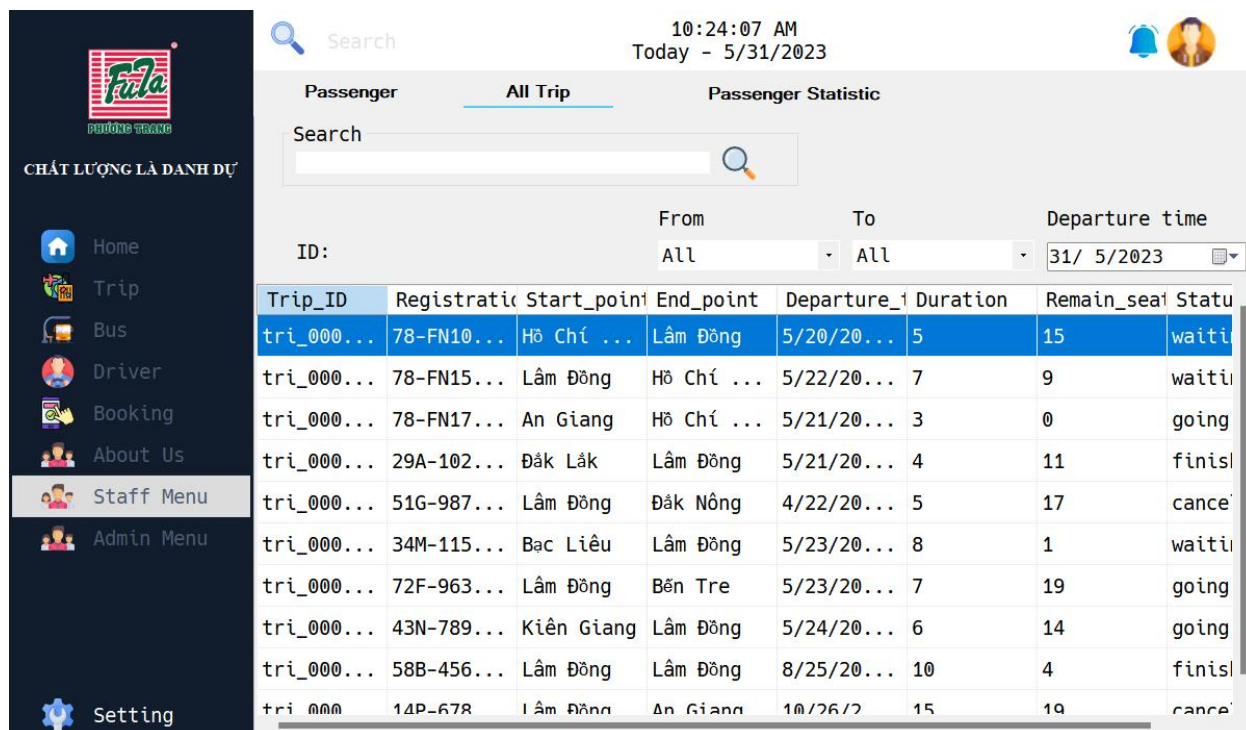
Search

ID: Name: ID

id_passenger	name	phone_numbe	address	identity_numbe	gender	email
pas_000000...	Nguyễn Văn...	0987654321 ...	12 Đường L...	0123456789...	<input type="checkbox"/>	nguyenvana...
pas_000000...	Trần Thị B	0912345678 ...	25 Ngõ 10 Đ...	0987654321...	<input checked="" type="checkbox"/>	trungnhan@...
pas_000000...	Lê Đức C	0976543210 ...	83 Đường H...	0123456789...	<input type="checkbox"/>	honglinh@g...
pas_000000...	Phạm Minh D	0901234567 ...	121 Nguyễn ...	0909090909...	<input checked="" type="checkbox"/>	phamhoang...
pas_000000...	Ngô Thanh E	0965432109 ...	56 Đường L...	0111111111...	<input type="checkbox"/>	thaingo@gm...
pas_000000...	Đinh Hoàng F	0943210765 ...	2 Đường Ng...	0999999999...	<input checked="" type="checkbox"/>	truongvuong...
pas_000000...	Vũ Thị G	0934560123 ...	34 Đường L...	0123456789...	<input type="checkbox"/>	duongnguye...
pas_000000...	Hoàng Văn H	0923456789 ...	76 Đường P...	0777777777...	<input checked="" type="checkbox"/>	phuonganh...
pas_000000...	Nguyễn Thị I	0918765432 ...	18 Đường N...	0666666666...	<input type="checkbox"/>	tranthang@g...
pas_000000...	Trần Văn J	0890123456 ...	67 Đường L...	0555555555...	<input checked="" type="checkbox"/>	thuytrang@g...

All trip screen

The All trip tab show the trips's information for the employee



10:24:07 AM
Today - 5/31/2023

Passenger All Trip Passenger Statistic

Search

ID: From To Departure time

All All 31/ 5/2023

Trip_ID	Registratio	Start_point	End_point	Departure_1	Duration	Remain_sea	Statu
tri_000...	78-FN10...	Hồ Chí ...	Lâm Đồng	5/20/20...	5	15	waiti
tri_000...	78-FN15...	Lâm Đồng	Hồ Chí ...	5/22/20...	7	9	waiti
tri_000...	78-FN17...	An Giang	Hồ Chí ...	5/21/20...	3	0	going
tri_000...	29A-102...	Đắk Lắk	Lâm Đồng	5/21/20...	4	11	finis
tri_000...	51G-987...	Lâm Đồng	Đắk Nông	4/22/20...	5	17	cance
tri_000...	34M-115...	Bạc Liêu	Lâm Đồng	5/23/20...	8	1	waiti
tri_000...	72F-963...	Lâm Đồng	Bến Tre	5/23/20...	7	19	going
tri_000...	43N-789...	Kiên Giang	Lâm Đồng	5/24/20...	6	14	going
tri_000...	58B-456...	Lâm Đồng	Lâm Đồng	8/25/20...	10	4	finis
tri_000...	14P-678...	Lâm Đồng	An Giang	10/26/2...	15	19	cance

Passenger statistic screen

The passenger statistic tab will statistic the booked trips of each passenger

Passenger **All Trip** **Passenger Statistic**


Passenger name: cuongle xuan 1 Phone: 0777988274

Ticket ID	Fare	Type	Seat number	Start point	End point	Departure time	Distance	Duration	Registration number
tic_0000000051	321.000	Seat	A4	Lâm Đồng	Hồ Chí Minh (TP)	5/22/2023 4:45:30 PM	310		7 78-FN1
tic_0000000092	426.000	Seat	N4	Đắk Lắk	Lâm Đồng	5/21/2023 3:30:00 PM	349		4 29A-10:
tic_0000000156	500.000	Sleeper	A3	Bạc Liêu	Lâm Đồng	5/23/2023 8:00:00 AM	240		8 34M-11
tic_0000000028	200.000	Seat	N2	Hồ Chí Minh (TP)	Lâm Đồng	5/20/2023 3:30:00 PM	305		5 78-FN1
tic_0000000029	200.000	Seat	N3	Hồ Chí Minh (TP)	Lâm Đồng	5/20/2023 3:30:00 PM	305		5 78-FN1
tic_0000000018	481.000	Sleeper	A14	Hồ Chí Minh (TP)	Lâm Đồng	5/20/2023 3:30:00 PM	305		5 78-FN1
tic_0000000024	432.000	Seat	N5	Lâm Đồng	An Giang	10/26/2023 10:10:30 AM	215		15 14P-67:
tic_0000000028	432.000	Seat	N9	Lâm Đồng	An Giang	10/26/2023 10:10:30 AM	215		15 14P-67:
Total:	8 2992 nghìn VND						2244		64

2.3. Admin function (only accessible by admins)


Cash reserve screen

This tab will show the money of each agent.





CHẤT LƯỢNG LÀ DANH DỰ


- Home
- Trip
- Bus
- Driver
- Booking
- About Us
- Staff Menu
- Admin Menu**
- Setting



10:29:58 AM
Today - 5/31/2023

Cash reserve
Employee
Statistic

Search



Money: 61,102,390,000

ID:
Region: All

Agent_ID	Phone	Address	Region	Money
age_0000000000...	0262 393 6868...	172 Lê Duẩn, ...	Đắk Lắk	3055152000.0000
age_0000000000...	02613 67 67 6...	226 Hai Bà Tr...	Đắk Nông	3055142000.0000
age_0000000000...	02913 93 2345...	QL1A, Khóm 2,...	Bạc Liêu	3055132000.0000
age_0000000000...	02753646464 ...	Đường Võ Nguy...	Bến Tre	3055122000.0000
age_0000000000...	02903 651 651...	309 Lý Thường...	Cà Mau	3055112000.0000
age_0000000000...	0283 511 9808...	292 Đinh Bộ L...	Hồ Chí Minh (TP)	3055102000.0000
age_0000000000...	02633 651 651...	695-697, QL20...	Lâm Đồng	3055092000.0000
age_0000000000...	02633 788 799...	735 Hùng Vươn...	Lâm Đồng	3055082000.0000
age_0000000000...	02633 731 731...	280 Trần Phú,...	Lâm Đồng	3055072000.0000
age_0000000001...	02973 66 88 6...	QL80, KP 5, P...	Kiên Giang	3055062000.0000


Employee screen

This tab will show the employee's information





CHẤT LƯỢNG LÀ DANH DỰ


- Home
- Trip
- Bus
- Driver
- Booking
- About Us
- Staff Menu
- Admin Menu**
- Setting



10:30:14 AM
Today - 5/31/2023

Cash reserve
Employee
Statistic


Search


ID:
Name:
Field: ID
Position: All

Employee	Name	Phone_N	Address	Email	Identit	Salary	Birthda	Gender	State	Positio
emp...	ADMIN	011...	Số 5...	adm...	123...	312...	12/1...	False	True	admi...
emp...	Le A	0999999999			123...	546...	12/1...	False	True	planner
emp...	Ngu...	033...	Số 3...		123...	768...	12/1...	False	True	supe...
emp...	Jus...	055...	Số 2...		123...	536...	12/1...	False	True	tick...
emp...	Luc...	088...	Số 1...		123...	546...	12/1...	True	True	driver

Statistic screen

This tab statistic the result of all agents



PHUONG TRANG

CHẤT LƯỢNG LÀ DANH DỰ

Home

Trip

Bus

Driver

Booking

About Us

Staff Menu

Admin Menu

Setting

Search

10:30:33 AM
Today - 5/31/2023

Cash reserve

Employee

Statistic

Year 2023

1 of 1

75%

Find | Next

1	0 0 VND
2	0 0 VND
3	0 0 VND
4	0 0 VND
5	6 2.128.000 VND
6	0 0 VND
7	0 0 VND
8	0 0 VND
9	0 0 VND
10	4 1.728.000 VND
11	0 0 VND
12	0 0 VND
Total	10 3.856 nghìn VND

