HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY FOR HIGH-QUALITY TRAINING

COURSE NAME: Database Management System

FINAL PROJECT REPORT

Project name:

BUS TICKET BOOKING MANAGEMENT SYSTEM

Lecturer: Prof. Nguyễn Thành Sơn

Course ID: DBMS330284E_22_2_01FIE

Group: 5

Date: 2nd Sem/2022-2023

Ho Chi Minh city, May , 2023

(This page was intentionally left blank)

LIST OF STUDENTS – GROUP 5

Project: Bus ticket booking management system

ID	***Full name***
21110758	Lê Xuân Cường
21110092	Bùi Quốc Thông
21110066	Phạm Vũ Bảo Nhân
21110785	Mai Nguyễn Nhật Nam

Professor's comment

Ho Chi Minh city, May …, 2023

Grading

CONTENTS

PROLOGUE

Firstly, we would like to express our gratitude to Prof. *Nguyễn Thành Sơn* for his whole-hearted instructions that helped us finish our final project for the Database Management System course. Thanks to the knowledge the professor has provided us, we were able to firmly grasp the basic knowledge and foundation for building a database management system. And through this project, our group would like to present the development process of a database management system 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 and 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 Database Management, 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

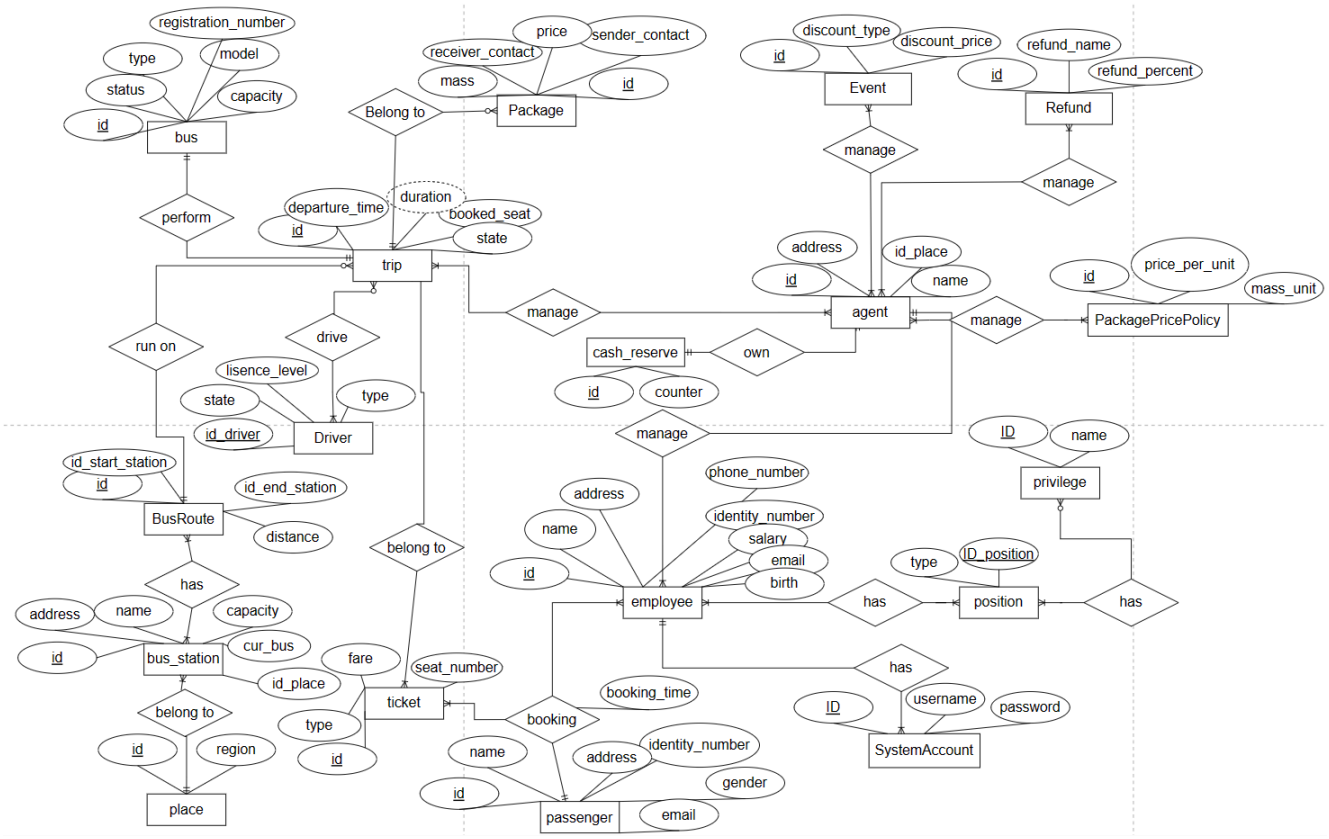
* Authorization:

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

CHAPTER II: SYSTEM ANALYSIS AND DESIGN

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)]

2. Logical level database design

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

1. BUS (id_bus, registration_number, model, capacity, status, type)
2. TRIP (id_trip, id_bus, id_bus_route, departure_time, duration, booked_seat, status)
3. TRIP_DRIVER (id_trip, id_driver)
4. AGENT (id_agent, id_place, id_cash_reserve, address, name)
5. CASHRESERVE (id_cash_reserve, counter)

6. BUSSTATION (id_bus_station, id_place, name, bus_capacity, count_current_bus)
7. DRIVER (id_driver, license_level, type, state)
8. BUSROUTE (id_route, id_start_station, id_end_station, distance)
9. PLACE (id_place, region)
10. EMPLOYEE (id_employee, id_account, id_agent, name, address, phone_number, identity_number, salary, email, birthdate, state)
11. POSITION (id_position, type)
12. PRIVILEGE (id_privilege, name)
13. SYSTEMACCOUNT (id_account, username, password)
14. TICKET (id_ticket, id_trip, status, fare, type, seat_number)
15. EVENT (id_event, discount_type, discount_percent)
16. REFUND (id_refund, refund_name, refund_percent)
17. PASSENGER (id_passenger, name, phone_number, address, identity_number, gender, email)
18. BOOKING (id_booking, id_ticket, id_passenger, id_employee, booking_time)
19. PACKAGE (id_package, id_trip, mass, price, sender_contact_phone, receiver_contact_phone)
20. PACKAGEPRICEPOLICY (id_policy, price_per_km, mass_unit)
21. AGENT_TRIP (id_agent, id_trip)
22. AGENT_EVENT (id_agent, id_event)
23. AGENT_REFUND (id_agent, id_refund)
24. AGENT_POLICY (id_agent, id_policy)
25. BUSROUTE_BUSSTATION (id_bus_route, id_bus_station)
26. EMPLOYEE_POSITION (id_employee, id_position)
27. EMPLOYEE_TICKET (id_employee, id_ticket)
28. POSITION_PRIVILEGE (id_position, id_privilege)

3. Required constraints

No.	Table	Constraint
1	BUS	Primary key: id_bus Check: CHK_bus_capacity

2	TRIP	Primary key: id_trip Foreign keys: FK_trip_id_bus, FK_trip_id_bus_route Check: CK_trip
3	TRIP_DRIVER	Foreign keys: FK_trip_driver_id_trip, FK_trip__driver_id_driver
4	DRIVER	Primary key: id_driver
5	AGENT	Primary key: id_agent Foreign key: FK_agent_id_cash_reserve, FK_agent_id_place
6	CASHRESERVE	Primary key: id_cash_reserve
7	BUSSTATION	Primary key: id_bus_station Foreign keys: FK_busstation_id_place Check: CHK_busstation
8	BUSROUTE	Primary key: id_route

		Foreign keys: FK_busroute_id_start_bus_station, FK_busroute_id_end_bus_station Check: CHK_busroute
9	PLACE	Primary key: id_place Check: CHK_place
10	EMPLOYEE	Primary key: id_employee Foreign keys: FK_employee_id_account, FK_employee_id_agent Check: CHK_employee
11	POSITION	Primary key: id_position Check: CHK_position
12	PRIVILEGE	Primary key: id_privilege Check: CHK_privilege
13	SYSTEMACCOUNT	Primary key: id_account

14	TICKET	Primary key: id_ticket Foreign key: FK_trip_id_trip Check: CHK_ticket
15	EVENT	Check: CHK_event
16	REFUND	Check: CHK_refund
17	PASSENGER	Primary key: id_passenger Check: CHK_passenger
18	BOOKING	Primary key: id_booking Foreign keys: FK_booking_id_ticket, FK_booking_id_passenger, FK_booking_id_employee
19	PACKAGE	Primary key: id_package Foreign keys: FK_package_id_trip Check: CHK_package

20	PACKAGEPRICEPOLICY	Primary key: id_policy Check: CHK_packagepricepolicy
21	AGENT_TRIP	Primary key: id_agent, id_trip Foreign key: FK_agent_trip_id_agent, FK_agent_trip_id_trip
22	AGENT_EVENT	Primary key: id_agent, id_event Foreign key: FK_agent_event_id_agent, FK_agent_event_id_event
23	AGENT_REFUND	Primary key: id_agent, id_refund Foreign key: FK_agent_refund_id_agent, FK_agent_refund_id_refund
24	AGENT_POLICY	Primary key: id_agent, id_policy Foreign key: FK_agent_policy_id_agent, FK_agent_policy_id_policy
25	BUSROUTE_BUSSTATION	Primary key: id_bus_route, id_bus_station

		Foreign key: FK_busroute_busstation_id_bus_route, FK_busroute_busstation_id_bus_station
26	EMPLOYEE_POSITION	Primary key: id_employee, id_postion Foreign key: FK_employee_position_id_employee, FK_employee_position_id_position
27	EMPLOYEE_TICKET	Primary key: id_employee, id_ticket Foreign key: FK_employee_ticket_id_employee, FK_employee_ticket_id_ticket
28	POSITION_PRIVILEGE	Primary key: id_position, id_privilege Foreign key: FK_position_privilege_id_position, FK_position_privilege_id_privilege

4. Database and constraints implementation

[BUS]

```
create table BUS
(
    id_bus varchar(20) primary key,
    registration_number char(15) unique not null,
    model varchar(50) not null,
    capacity tinyint default 32,
    status char(15) not null default 'idle',
    type bit not null default 0
    -- 0: interprovince, 1: transit
);
```

[TRIP]

```sql
create table TRIP
(
    id_trip varchar(20) primary key,
    id_bus varchar(20) ,
    id_bus_route varchar(20),
    departure_time datetime not null,
    duration int not null,
    -- unit: hour,
    booked_seat tinyint default 0,
    status char(15) default 'waiting'
);
```

[TRIP_DRIVER]

```sql
create table TRIP_DRIVER
(
    id_trip varchar(20),
    id_driver varchar(20),
    primary key(id_trip, id_driver)
);
```

[AGENT]

```sql
create table AGENT
(
    id_agent varchar(20) primary key,
    id_cash_reserve varchar(20),
    id_place varchar(20),
    name varchar(50) not null,
    address char(100) not null
);
```

[CASHRESERVE]

```sql
create table CASHRESERVE
(
    id_cash_reserve varchar(20) primary key,
    counter money default 0,
);
```

[BUSSTATION]

```sql
create table BUSSTATION
(
    id_bus_station varchar(20) primary key,
    id_place varchar(20),
    name varchar(50) not null,
    address char(100) not null,
    bus_capacity int not null,
    count_current_bus int null default 0,
);
```

[DRIVER]

```
create table DRIVER
(
    id_driver varchar(20) primary key,
    -- foreign key
    lisence_level char(10) not null,
    type bit default 0,
    -- 0: interprovince, 1: transit
    state bit default 0
    -- 0: not drive, 1: is driving
);
```

[BUSROUTE]

```
create table BUSROUTE
(
    id_route varchar(20) primary key,
    id_start_station varchar(20),
    id_end_station varchar(20),
    distance int not null
    -- unit: km
);
```

[PLACE]

```
create table PLACE
(
    id_place varchar(20) primary key,
    region char(50) default 'TP.Ho Chi Minh'
);
```

[EMPLOYEE]

```
create table EMPLOYEE
(
    id_employee varchar(20) primary key,
    id_account varchar(20),
    id_agent varchar(20),
    name varchar(50) not null,
    address char(100) not null,
    phone_number varchar(20) not null,
    identity_number char(20) not null,
    salary money not null,
    email char(50) null,
    birthdate date not null,
    state bit default 1
    -- 0: not working, 1: is working
);
```

[POSITION]

```sql
create table POSITION
(
    id_position varchar(20) primary key,
    type varchar(50) not null
);
```

[PRIVILEGE]

```sql
create table PRIVILEGE
(
    id_privilege varchar(20) primary key,
    name char(50)
);
```

[SYSTEMACCOUNT]

```sql
create table SYSTEMACCOUNT
(
    id_account varchar(20) primary key,
    username varchar(20) not null unique,
    pass varchar(50) not null,
);
```

[TICKET]

```sql
create table TICKET
(
    id_ticket varchar(20) primary key,
    id_trip varchar(20) ,
    status bit default 0,
    fare money not null,
    type bit default 0,
    -- 0: seat, 1: lie
    seat_number char(15) not null unique
);
```

[EVENT]

```sql
create table EVENT
(
    id_event varchar(20) primary key,
    discount_type char(50) not null unique default 'normal',
    discount_percent float default 0.0
);
```

[REFUND]

```sql
create table REFUND
(
    id_refund varchar(20) primary key,
    refund_name char(50) not null unique default 'cancel',
    refund_percent float default 0.0
);
```

[PASSENGER]

```sql
create table PASSENGER
(
    id_passenger varchar(20) primary key,
    name varchar(50) not null,
    phone_number varchar(20) not null,
    address char(100) not null,
    identity_number char(20) null,
    gender bit default 0,
    -- 0: male, 1: female
    email char(50) null,
);
```

[BOOKING]

```sql
create table BOOKING
(
    id_booking varchar(20) primary key,
    id_ticket varchar(20),
    id_passenger varchar(20),
    id_employee varchar(20),
    booking_time datetime default getdate(),
);
```

[PACKAGE]

```sql
create table PACKAGE
(
    id_package varchar(20) primary key,
    id_trip varchar(20),
    mass smallint default 0,
    price money ,
    --  is calculated by the formula
    sender_contact_phone char(20) not null,
    receiver_contact_phone char(20) not null
);
```

[PACKAGEPRICEPOLICY]

```sql
create table PACKAGEPRICEPOLICY
(
    id_policy varchar(20) primary key,
    price_per_km money not null,
    mass_unit int not null
    -- /5kg, /1kg
);
```

[AGENT_TRIP]

```sql
create table AGENT_TRIP
(
    id_agent varchar(20),
    id_trip varchar(20),
    primary key(id_agent, id_trip)
);
```

[AGENT_EVENT]

```sql
create table AGENT_EVENT
(
    id_agent varchar(20),
    id_event varchar(20),
    primary key(id_agent, id_event)
);
```

[AGENT_REFUND]

```sql
create table AGENT_REFUND
(
    id_agent varchar(20),
    id_refund varchar(20),
    primary key(id_agent, id_refund)
);
```

[AGENT_POLICY]

```sql
create table AGENT_POLICY
(
    id_agent varchar(20),
    id_policy varchar(20),
    primary key(id_agent, id_policy)
);
```

[BUSROUTE_BUSSTATION]

```sql
create table BUSROUTE_BUSSTATION
(
    id_bus_route varchar(20),
    id_bus_station varchar(20),
    primary key(id_bus_route, id_bus_station)
);
```

[EMPLOYEE_POSITION]

```sql
create table EMPLOYEE_POSITION
(
    id_employee varchar(20),
    id_position varchar(20),
    primary key(id_employee, id_position)
);
```

[EMPLOYEE_TICKET]

```sql
create table EMPLOYEE_TICKET
(
    id_employee varchar(20),
    id_ticket varchar(20),
    primary key(id_employee, id_ticket)
);
```

[POSITION_PRIVILEGE]

```sql
create table POSITION_PRIVILEGE
(
    id_position varchar(20),
    id_privilege varchar(20),
    primary key(id_position, id_privilege)
);
```

5. Other constraints

* Constrain bus ID after add one

```sql
-- Set constraint bus identity automatically.
USE BusManagement
ALTER TABLE Bus
ADD CONSTRAINT AUTO_ID_Bus
DEFAULT DBO.AUTO_ID_Bus() FOR ID_bus;
GO
```

* Constrain passenger ID and ticket ID after customer buy a ticket

```sql
-- Set constraint ticket identity automatically.
USE BusManagement
ALTER TABLE TICKET
ADD CONSTRAINT AUTO_ID_ticket
DEFAULT DBO.AUTO_ID_ticket() FOR ID_ticket;
GO
```

```sql
-- Set constraint passenger identity automatically.
USE BusManagement
ALTER TABLE PASSENGER
ADD CONSTRAINT AUTO_ID_passenger
DEFAULT DBO.AUTO_ID_passenger() FOR ID_passenger;
GO
```

* Constrain trip ID after add one

```sql
-- Set constraint trip identity automatically.
USE BusManagement
ALTER TABLE TRIP
ADD CONSTRAINT AUTO_ID_trip
DEFAULT DBO.AUTO_ID_trip() FOR ID_trip;
GO
```

* Constrain route ID after add one

```sql
-- Set constraint route identity automatically.
USE BusManagement
ALTER TABLE BUS_ROUTE
ADD CONSTRAINT AUTO_ID_route
DEFAULT DBO.AUTO_ID_route() FOR ID_route;
GO
```

* Constrain position ID after add one

```sql
-- Set constraint route identity automatically.
USE BusManagement
ALTER TABLE POSITION
ADD CONSTRAINT AUTO_ID_position
DEFAULT DBO.AUTO_ID_position() FOR ID_position;
GO
```

* Constrain employee ID after add one

```sql
-- Set constraint route identity automatically.
USE BusManagement
ALTER TABLE EMPLOYEE
ADD CONSTRAINT AUTO_ID_employee
DEFAULT DBO.AUTO_ID_employee() FOR ID_employee;
GO
```

Constrain agent ID after add one

```
-- Set constraint bus identity automatically.
USE BusManagement
ALTER TABLE Agent
ADD CONSTRAINT AUTO_ID_Agent
DEFAULT DBO.AUTO_ID_Agent() FOR ID_agent;
GO
```

* Constrain booking ID after add one

```
-- Set constraint bus identity automatically.
USE BusManagement
ALTER TABLE Booking
ADD CONSTRAINT AUTO_ID_Booking
DEFAULT DBO.AUTO_ID_Booking() FOR ID_booking;
GO
```

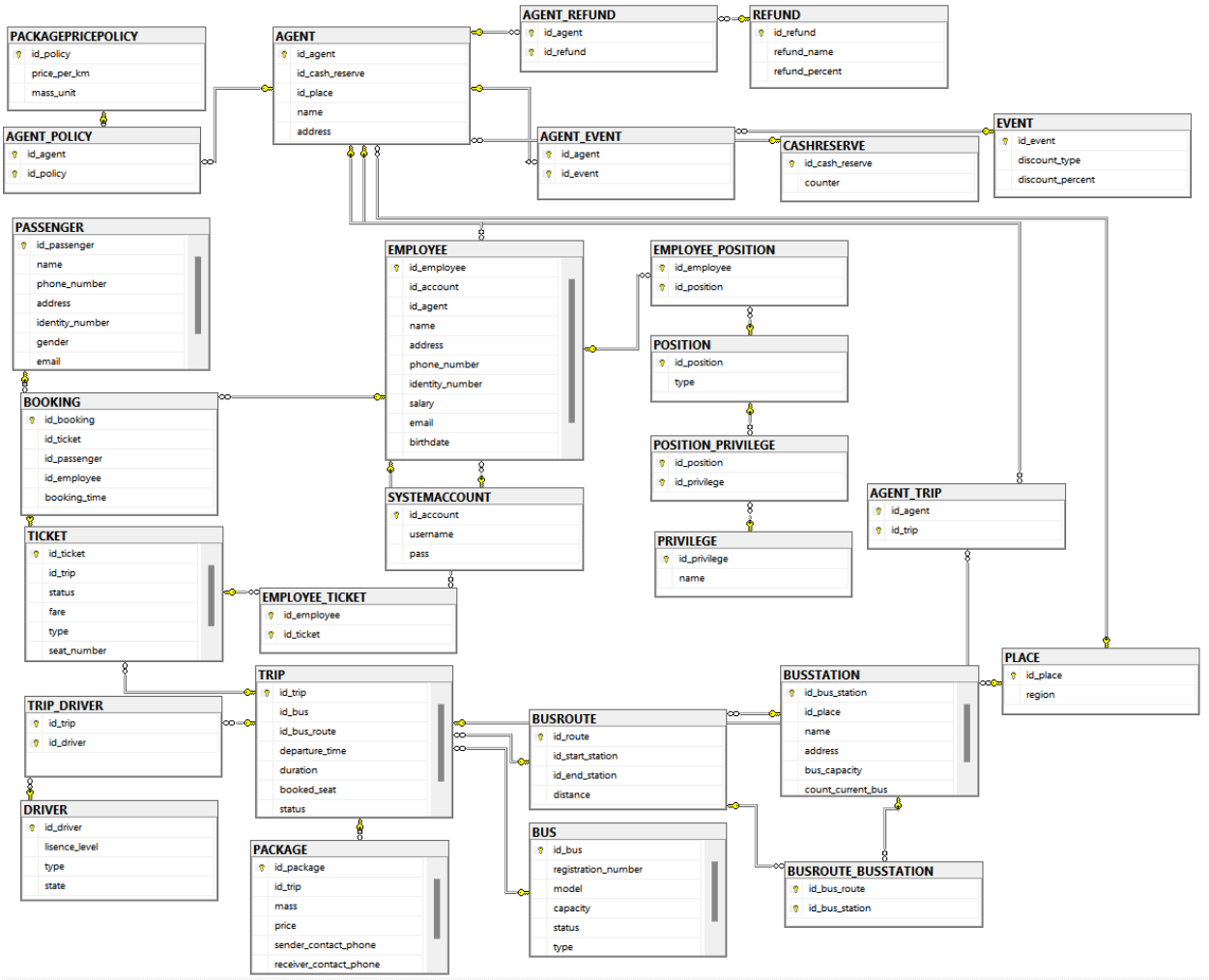
* Constrain package ID after add one

```
-- Set constraint bus identity automatically.
USE BusManagement
ALTER TABLE Package
ADD CONSTRAINT AUTO_ID_Package
DEFAULT DBO.AUTO_ID_Package() FOR ID_package;
GO
```

* Constrain package price policy ID after add one

```
-- Set constraint bus identity automatically.
USE BusManagement
ALTER TABLE PackagePricePolicy
ADD CONSTRAINT AUTO_ID_PackagePricePolicy
DEFAULT DBO.AUTO_ID_PackagePricePolicy() FOR ID_policy;
GO
```

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



6. Constraint-checking triggers

* Update state of an employee on Employee and Driver relation

```sql
CREATE TRIGGER tr_employee_update_stateEmployee
ON Employee
AFTER UPDATE
AS
BEGIN
	DECLARE @employee_id CHAR
	DECLARE @new_state CHAR
	SELECT @employee_id = ID_employee, @new_state = state FROM inserted
	-- Kiểm tra nếu trạng thái mới của nhân viên là 0
	IF @new_state = 0
	BEGIN
	-- Cập nhật trạng thái của nhân viên trong bảng Employee thành 0
	UPDATE Employee SET state = 0 WHERE ID_employee = @employee_id
		UPDATE Driver SET state = 0 WHERE ID_driver = @employee_id
	END
	SELECT state FROM Driver WHERE ID_driver = @employee_id


END
```

* Delete employee account and update state of inactive Employee (state = 0)

```sql
CREATE TRIGGER tr_employee_deleteAccount
ON Employee
AFTER UPDATE
AS
BEGIN
	DECLARE @account_id CHAR
	DECLARE @new_state CHAR
	SELECT @account_id = ID_account, @new_state = inserted.state FROM inserted

	IF @new_state = 0
	BEGIN
	-- Delete account
	DELETE FROM SystemAccount WHERE @account_id = ID_account
	END
END
```

7. Database views implementation

* View for list of active employee which is working (state = 1)

```sql
CREATE VIEW [dbo].[ActiveEmployee] AS
SELECT Employee.ID_employee, Employee_ID_account, Employee.name, Employee.address,
Employee.phone_number, Employee.identity_number, Employee.salary, Employee.email,
Employee.birthday, Agent.name, Position.type
FROM
Employee AS temp1 INNER JOIN Agent AS temp 2
ON temp1.ID_agent = temp2.ID_agent
INNER JOIN Position as temp3
ON temp1.ID_position = temp3.ID_position
WHERE temp1.status = 1
```

* View for list of waiting trip:

```sql
CREATE VIEW [dbo].[WaitingTrip] AS
temp1.ID_trip, temp1.departure_time, temp1.duration, temp1.booked_seat,
temp1.registration_number, temp1.type,
temp2.name AS start_point, temp3.name AS end_point
FROM
((SELECT Trip.*, Bus.registration_number, Bus.type
FROM Trip INNER JOIN Bus
ON Trip.ID_bus = Bus.ID_bus) AS temp0
INNER JOIN BusRoute
ON BusRoute.ID_route = temp0.ID_route) AS temp1
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station1, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station1 = BusStation.ID_bus_station
) AS temp2
ON temp1.ID_route = temp2.ID_route
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station2, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station2 = BusStation.ID_bus_station
) AS temp3
ON temp1.ID_route = temp3.ID_route
WHERE temp1.status = 'Waiting'
```

* View for of going trip:

```sql
CREATE VIEW [dbo].[GoingTrip] AS
temp1.ID_trip, temp1.departure_time, temp1.duration, temp1.booked_seat,
temp1.registration_number, temp1.type,
temp2.name AS start_point, temp3.name AS end_point
FROM
((SELECT Trip.*, Bus.registration_number, Bus.type
FROM Trip INNER JOIN Bus
ON Trip.ID_bus = Bus.ID_bus) AS temp0
INNER JOIN BusRoute
ON BusRoute.ID_route = temp0.ID_route) AS temp1
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station1, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station1 = BusStation.ID_bus_station
) AS temp2
ON temp1.ID_route = temp2.ID_route
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station2, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station2 = BusStation.ID_bus_station
) AS temp3
ON temp1.ID_route = temp3.ID_route
WHERE temp1.status = ‘Going’
```

* View for list of finished trip:

```sql
CREATE VIEW [dbo].[FinishTrip] AS
temp1.ID_trip, temp1.departure_time, temp1.duration, temp1.booked_seat,
temp1.registration_number, temp1.type,
temp2.name AS start_point, temp3.name AS end_point
FROM
((SELECT Trip.*, Bus.registration_number, Bus.type
FROM Trip INNER JOIN Bus
ON Trip.ID_bus = Bus.ID_bus) AS temp0
INNER JOIN BusRoute
ON BusRoute.ID_route = temp0.ID_route) AS temp1
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station1, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station1 = BusStation.ID_bus_station
) AS temp2
ON temp1.ID_route = temp2.ID_route
INNER JOIN (
SELECT temp1.ID_route, temp1.ID_bus_station2, BusStation.name
FROM temp1 INNER JOIN BusStation
ON temp1.ID_bus_station2 = BusStation.ID_bus_station
) AS temp3
ON temp1.ID_route = temp3.ID_route
WHERE temp1.status = ‘Finish'
```

* View for list of idle interprovince bus:

```sql
CREATE VIEW [dbo].[IdleInterprovinceBus]
AS
SELECT rel.ID_bus, rel.registration_number, rel.model, rel.capacity
FROM Bus as rel
WHERE Bus.status = 'idle' AND Bus.type = 'interprovince'
```

* View for list of break interprovince bus:

```sql
CREATE VIEW [dbo].[BreakInterprovinceBus]
AS
SELECT rel.ID_bus, rel.registration_number, rel.model, rel.capacity
FROM Bus as rel
WHERE Bus.status = 'break' AND Bus.type = 'interprovince'
```

* View for list of incident interprovince bus:

```sql
CREATE VIEW [dbo].[IncidentInterprovinceBus]
AS
SELECT rel.ID_bus, rel.registration_number, rel.model, rel.capacity
FROM Bus as rel
WHERE Bus.status = 'incident' AND Bus.type = 'interprovince'
```

* View for list of ongoing interprovince bus:

```sql
CREATE VIEW [dbo].[OnGoingInterprovinceBus]
AS
SELECT rel.ID_bus, rel.registration_number, rel.model, rel.capacity
FROM Bus AS rel INNER JOIN (
Select Trip.ID_bus, Trip.ID_trip
FROM Trip
WHERE Trip.status = 'going'
) as rel2
ON rel.ID_bus = rel2.ID_bus
WHERE Bus.status = 'ongoing' AND Bus.type = 'interprovince'
```

* View for (detailed) list of passenger booking information:

```sql
CREATE VIEW [dbo].[BookingInfor]
AS
SELECT rel.ID_booking, temp1.ID_ticket, temp1.ID_trip, temp1.seat_number, temp1.type,
temp1.fare, temp2.name AS passenger_name, temp2.phone_number AS passenger_phone_number,
temp2.address AS passenger_address, temp2.email AS passenger_email, temp2.gender AS
passenger_gender, temp3.name AS employee_name, temp3.phone_number AS
employee_phonee_number
FROM Booking AS rel INNER JOIN Ticket AS temp1
ON rel.ID_ticket = temp1.ID_ticket
INNER JOIN Passenger AS temp2
ON rel.ID_passenger = temp2.ID_passenger
INNER JOIN Employee AS temp3
ON rel.ID_employee = temp3.ID_employee
```

* View for (detailed) list of current bus route information:

```sql
CREATE VIEW  [dbo].[BusRouteInfor]
AS
SELECT rel.ID_route, temp1.start_point, temp2.end_point, rel.distance
FROM BusRoute AS rel INNER JOIN (
SELECT rel.ID_Route, BusStation.name as start_point
FROM rel INNER JOIN BusStation
ON rel.ID_bus_station1 = BusStation.ID_bus_station
) AS temp1
ON rel.ID_route = temp1.ID_route
INNER JOIN (
SELECT rel.ID_Route, BusStation.name as end_point
FROM rel INNER JOIN BusStation
ON rel.ID_bus_station2 = BusStation.ID_bus_station
) AS temp2
ON rel.ID_route = temp2.ID_route
```

* View for list of Going trip driver:

```sql
CREATE VIEW [dbo].[WaitingTripDriverInfor]
AS
SELECT rel.*, temp2.ID_driver, temp2.name AS driver_name, temp2.phone_number AS
driver_phone_number
FROM WaitingTrip AS rel INNER JOIN TripDriver AS temp1
ON rel.ID_trip = temp1.ID_trip
INNER JOIN (
SELECT Driver.ID_driver, Employee.name, Employee.phone_number
FROM Driver INNER JOIN Employee
ON Driver.ID_driver = Employee.ID_employee
) AS temp2
ON temp1.ID_driver = temp2.ID_driver
```

* View for list of Going trip driver:

```sql
CREATE VIEW [dbo].[GoingTripDriverInfor]
AS
SELECT rel.*, temp2.ID_driver, temp2.name AS driver_name, temp2.phone_number AS
driver_phone_number
FROM GoingTrip AS rel INNER JOIN TripDriver AS temp1
ON rel.ID_trip = temp1.ID_trip
INNER JOIN (
SELECT Driver.ID_driver, Employee.name, Employee.phone_number
FROM Driver INNER JOIN Employee
ON Driver.ID_driver = Employee.ID_employee
) AS temp2
ON temp1.ID_driver = temp2.ID_driver
```

* View for list of Going trip driver:

```sql
CREATE VIEW [dbo].[FinishTripDriverInfor]
AS
SELECT rel.*, temp2.ID_driver, temp2.name AS driver_name, temp2.phone_number AS
driver_phone_number
FROM FinishTrip AS rel INNER JOIN TripDriver AS temp1
ON rel.ID_trip = temp1.ID_trip
INNER JOIN (
SELECT Driver.ID_driver, Employee.name, Employee.phone_number
FROM Driver INNER JOIN Employee
ON Driver.ID_driver = Employee.ID_employee
) AS temp2
ON temp1.ID_driver = temp2.ID_driver
```

* View for list of employee accounts:

```sql
CREATE VIEW [dbo].[EmployeeAccount]
AS
SELECT temp1.ID_employee, temp1.name, temp2.username, temp2.password
FROM Employee AS temp1, SystemAccount AS temp2
```

* View for list of waiting trip which still has empty seats:

```sql
CREATE VIEW [dbo].[TripWithAvailableChair]
AS
SELECT temp1.*, temp2.capacity - temp1.booked_seat AS available_position
FROM WaitingTrip AS temp1 INNER JOIN (
SELECT  Bus.capacity, Trip.ID_trip
FROM Trip INNER JOIN Bus
ON Trip.ID_bus = Bus.ID_bus
) as temp2
ON temp1.ID_trip temp2.ID_trip
WHERE temp2.capacity - temp1.booked_seat > 0
```

* View for list of waiting trip which has full seats:

```sql
CREATE VIEW [dbo].[TripWithAvailableChair]
AS
SELECT temp1.*, temp2.capacity - temp1.booked_seat AS available_position
FROM WaitingTrip AS temp1 INNER JOIN (
SELECT  Bus.capacity, Trip.ID_trip
FROM Trip INNER JOIN Bus
ON Trip.ID_bus = Bus.ID_bus
) as temp2
ON temp1.ID_trip temp2.ID_trip
WHERE temp2.capacity - temp1.booked_seat = 0
```

* View for sum of all agents' cash reserve:

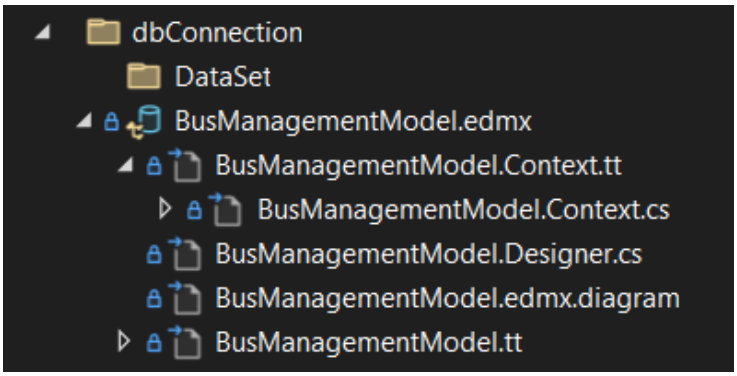
```sql
CREATE VIEW [dbo].[V_TotalCashReserve]
AS
SELECT SUM(counter) AS sum_counter
FROM   dbo.CASHRESERVE
```

CHAPTER III: FUNCTION AND PROCEDURE DESIGN

1. Connect to database

This project uses Entity Framework in order to connect to the BusManagement database, hence the Connection string is stored in App.config instead of a variable in the application, but this feature is still part of the .NET framework that the project is based on.

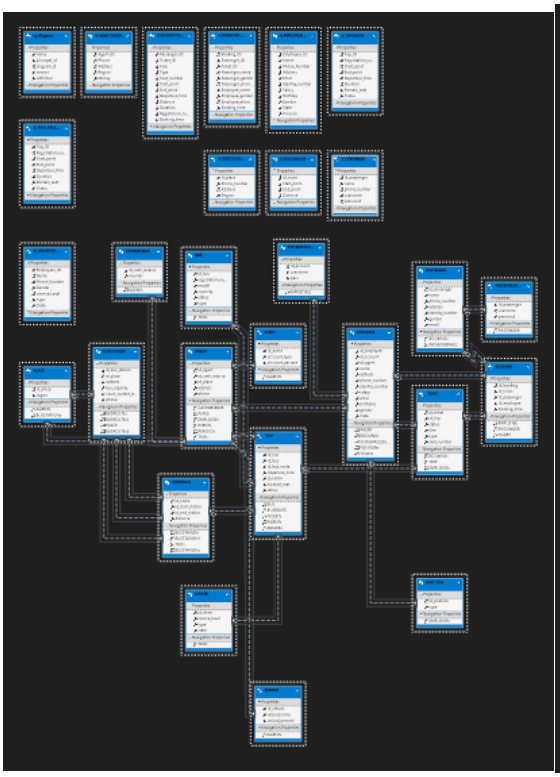
The Entity Framework library maps the database into the C# Project so we will have a dbConnection file that helps us perform operations (functions and procedures) and get database entities (views, relations,…) without having to directly use SQL commands.

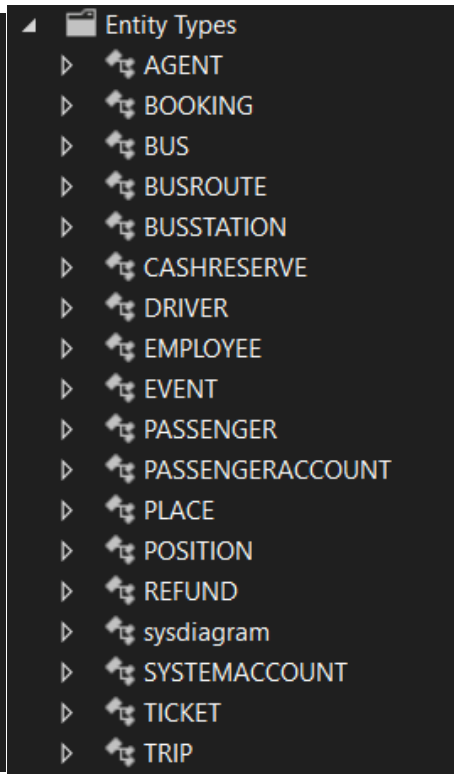


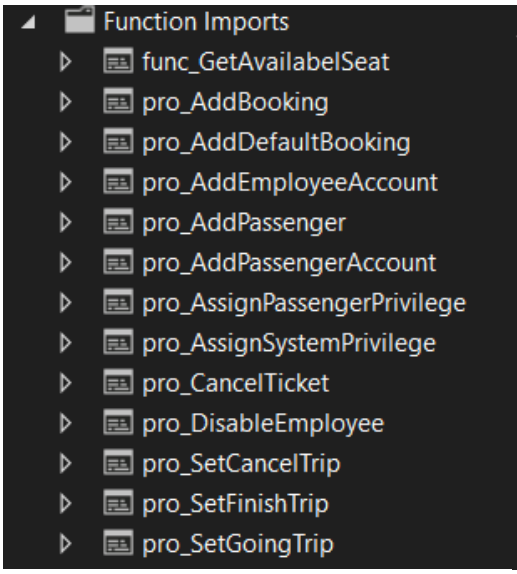
Connection string (For Server Admins):

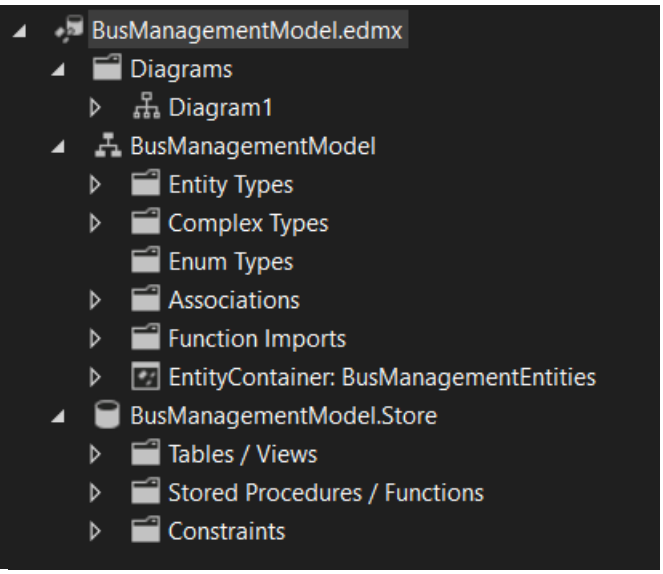
```
<connectionStrings>
    <add
name="BusTicketManagementApplication.Properties.Settings.BusManagementConnectionS
tring" connectionString="Data Source=.;Initial Catalog=BusManagement;Integrated
Security=True" providerName="System.Data.SqlClient" /><add
name="BusManagementEntities"
connectionString="metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://
/*/src.dbConnection.BusManagementModel.ssdl|res://*/src.dbConnection.BusManagemen
tModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
source=(local);initial catalog=BusManagement;persist security info=True;user
id=admin01;password=admin;MultipleActiveResultSets=True;App=EntityFramework&quot;
" providerName="System.Data.EntityClient" />
</connectionStrings>
```

Model tree of BusManagementModel.edmx:









2. Function design

NOTE: All functions with the name [func_auto_...] have the same functionality and call syntax in the program so we will display only ONE example.

- dbo.func_auto_id_account
- dbo.func_auto_id_agent
- dbo.func_auto_id_booking
- dbo.func_auto_id_bus
- dbo.func_auto_id_busroute
- dbo.func_auto_id_busstation
- dbo.func_auto_id_cashreserve
- dbo.func_auto_id_employee
- dbo.func_auto_id_event
- dbo.func_auto_id_package
- dbo.func_auto_id_passenger
- dbo.func_auto_id_policy
- dbo.func_auto_id_position
- dbo.func_auto_id_privilege
- dbo.func_auto_id_refund
- dbo.func_auto_id_ticket
- dbo.func_auto_id_trip
- dbo.func_AutoDefaultIdEmployee

All SQL functions are passed in this function - using only their names - in order to run in the program.

```
        public static string RunFunc(string funcName)
        {
            BusManagementEntities db = new BusManagementEntities();
            string query = $"select dbo.{funcName}()";
            return
db.Database.SqlQuery<string>(query).ToList().FirstOrDefault().ToString();
        }
```

*** Example for [func_auto_...] to automatically create a default passenger ID:**

```
CREATE function [dbo].[func_auto_id_passenger]()
returns char(20)
as
begin
declare @id_no char(20)
set @id_no = (
    select max(id_passenger)
    from PASSENGER
)
if( @id_no is null)
    set @id_no = concat('pas_', '0000000000')
declare @no int
set @no = right(@id_no, 10) + 1;
return concat('pas_', format(@no, '0000000000'))
end
```

```
        public string GetNewPassengerId()
        {
            BusManagementEntities db = new BusManagementEntities();
            string funcName = "func_auto_id_passenger";
            return BSMain.RunFunc(funcName);
        }
```

* Function to add a new passenger:

```
create function func_AddPassenger(@name nvarchar(50), @phone char(20))
returns char(20)
as
begin
       declare @id_passenger char(20)
       set @id_passenger = dbo.func_auto_id_passenger();
       exec dbo.pro_AddPassenger @id_passenger, @name, @phone;
       return @id_passenger
end
```

```
        public void AddPassenger(string name, string phone)
        {
            BusManagementEntities db = new BusManagementEntities();
            string idPassenger = BSMain.RunTableValuedFunc("func_AddPassenger",
new List<string> { name, phone }).FirstOrDefault();
        }
```

* Function to get available seats in a trip:

```
create function func_GetAvailableSeat(@idTrip char(20), @type bit)
returns table
as
return (select TICKET.seat_number from TICKET where TICKET.id_trip = @idTrip and
TICKET.type = @type and TICKET.status = 0)
```

```
        public List<string> GetAvailableSeat(string idTrip, int type) // type 0:
seat, 1: sleeper
        {
            BusManagementEntities db = new BusManagementEntities();
            string funcName = "func_GetAvailabelSeat";
            List<string> ticketList = BSMain.RunTableValuedFunc(funcName, new
List<string> { idTrip, type.ToString() });
            return ticketList;
        }
```

3. Procedure design

* Procedure for cancelling a ticket:

```sql
create proc pro_CancelTicket @id_ticket char(20)
as
begin
	update TICKET set status = 0 where id_ticket = @id_ticket
end
```

```
        private void BtnCancel_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(this.LbSelectedId.Text.Trim()))
            {
                MessageBox.Show("Please select the ticket to cancel!");
                return;
            }
            BusManagementEntities db = new BusManagementEntities();
            //
            db.pro_CancelTicket(this.LbSelectedId.Text.Trim());
            this.LbSelectedId.Text = string.Empty;
            MessageBox.Show("Cancel successfully!");
            FilterBookedTickets();
        }
```

*** Procedure for setting a trip status as 'cancel':**

```sql
create proc pro_SetCancelTrip @id_trip char(20)
as
begin
	update TRIP set status = 'cancel' where id_trip = @id_trip
end
```

```
        public void SetCancelTrip(string tripId)
        {
            BusManagementEntities db = new BusManagementEntities();
            db.pro_SetCancelTrip(tripId);
        }
```

*** Procedure for setting a trip status as 'finish':**

```sql
create proc pro_SetFinishTrip @id_trip char(20)
as
begin
	update TRIP set status = 'finish' where id_trip = @id_trip
end
```

```
        public void SetFinish(string tripId)
        {
            BusManagementEntities db = new BusManagementEntities();
            db.pro_SetFinishTrip(tripId);
        }
```

*** Procedure for setting a trip status as 'going':**

```
create proc pro_SetGoingTrip @id_trip char(20)
as
begin
	update TRIP set status = 'going' where id_trip = @id_trip
end
```

```
        public void SetGoing(string tripId)
        {
            BusManagementEntities db = new BusManagementEntities();
            db.pro_SetGoingTrip(tripId);
        }
```

* Procedure for adding default ticket information while booking:

```
create proc pro_AddDefaultBooking @id_ticket char(20), @id_passenger char(20)
as
begin
	update TICKET set TICKET.status = 1 where TICKET.id_ticket = @id_ticket;
	insert into BOOKING(id_ticket, id_passenger, id_employee, booking_time) values
(@id_ticket, @id_passenger, dbo.func_AutoDefaultIdEmployee(), GETDATE());
end
```

```
        BusManagementEntities db = new BusManagementEntities();
            //
            db.pro_AddDefaultBooking(this.TbIDTicket.Text.Trim(),
UserData.GetPassengerId());
            // payment logic ( directly, online )
```

* Procedure for booking a ticket for a passenger:

```
create proc pro_AddBooking @id_ticket char(20), @id_passenger char(20), @id_employee
char(20)
as
begin
	update TICKET set TICKET.status = 1 where TICKET.id_ticket = @id_ticket;
	insert into BOOKING(id_ticket, id_passenger, id_employee, booking_time) values
(@id_ticket, @id_passenger, @id_employee, GETDATE());
end
```

```
        public void AddBooking(string ticketId, string passengerId, string
employeeId)
        {
            BusManagementEntities db = new BusManagementEntities();
            db.pro_AddBooking(ticketId, passengerId, employeeId);
        }
```

* Procedure for adding a passenger:

```sql
create proc pro_AddPassenger @id_passenger char(20), @name nvarchar(50), @phone
char(20)
as
begin
       insert into PASSENGER(id_passenger, name, phone_number) values (@id_passenger,
@name, @phone);
end
```

```
string funcName = "func_auto_id_passenger";
         passengerId =  BSMain.RunFunc(funcName);
         if (!string.IsNullOrEmpty(passengerId))
         {
            db.pro_AddPassenger(passengerId, name, phone);
            db.pro_AddPassengerAccount(passengerId, username, password); // add
passenger account and assign privilege
            errMsg = "Create new user successfully!. No error";
         }
            else
         {
            errMsg = "Can't get new passengerId";
         }
```

*** Simulate [C#]: Try/Catch SQL exception in ValidateUser function (Check whether the user information is in the database or not)**

The function will pass an error variable into the function to get the exception message. When an exception occurs, the function will automatically assign the message to the error variable then the program will show it to the user.

```sql
create PROCEDURE pro_CheckUniqueUser(@username varchar(50))
AS
BEGIN
    DECLARE @count int, @errMsg nvarchar(MAX)
    SET @count = 0
    SELECT @count = COUNT(*) FROM PASSENGERACCOUNT WHERE PASSENGERACCOUNT.username =
@username
    IF @count > 0
    BEGIN
        SET @errMsg = 'Username has already been taken.';
        RAISERROR(@errMsg, 16, 1);
    END
    SELECT @count
END
```

```
public bool CreateNewUser(string username, string password,string name, string
phone,ref string passengerId, ref string errMsg)
        {
            passengerId = string.Empty;
            try
            {
                foreach (char c in username)
                {
                    if (c < 48 || (c > 57 && c < 65) || (c > 90 && c < 97) || c >
122)
                    {
                        errMsg = "Username is invalid. Username must be letters
in alphabet and digits";
                        return false;
                    }
                }
                BusManagementEntities db = new BusManagementEntities();
                // way 1
                //bool uniqueUser = db.PASSENGERACCOUNTs.Count(d => d.username ==
username) == 0;
                //if(!uniqueUser){
                //    errMsg = "Username has exist in the system!";
                //    return false;
                //}
                // way 2: check whether unique username
                db.Database.SqlQuery<int>("EXEC pro_CheckUniqueUser @username",
new SqlParameter("@username", username)); //if user exist,throw an sql exception
                // incase of unique username
                string funcName = "func_auto_id_passenger";
                passengerId =  BSMain.RunFunc(funcName);
                if (!string.IsNullOrEmpty(passengerId))
                {
                    db.pro_AddPassenger(passengerId, name, phone);
                    db.pro_AddPassengerAccount(passengerId, username, password);
// add passenger account and assign privilege
                    errMsg = "Create new user successfully!. No error";
                }
                else
                {
                    errMsg = "Can't get new passengerId";
                }
            }
            catch (SqlException err)
            {
                errMsg = err.Message;
                MessageBox.Show(err.Message.ToString());
                return false;
            }
            catch (Exception ex)
            {
                MessageBox.Show("An error occurred: " + ex.Message);
                return false;
            }
            return true;
        }
```

*** Procedure for changing user passwords**

```sql
CREATE PROC [dbo].[pro_ChangeSystemPassword] (@username varchar(50), @new_password
varchar(50))
AS
BEGIN
	SET XACT_ABORT ON;

	BEGIN TRAN;
	BEGIN TRY
		UPDATE SYSTEMACCOUNT SET pass = @new_password WHERE username =
@username;

		DECLARE @query nvarchar(MAX);
		SET @query = 'ALTER LOGIN ' + QUOTENAME(@username) + ' WITH PASSWORD =
''' + @new_password + ''';';
		EXEC (@query);

		COMMIT TRAN;
	END TRY
	BEGIN CATCH
		ROLLBACK TRAN;
		THROW;
	END CATCH;
END;
```

```sql
CREATE PROC [dbo].[pro_ChangePassengerPassword] (@username varchar(50), @new_password
varchar(50))
AS
BEGIN
	SET XACT_ABORT ON;

	BEGIN TRAN;
	BEGIN TRY
		UPDATE PASSENGERACCOUNT SET password = @new_password WHERE username =
@username;

		DECLARE @query nvarchar(MAX);
		SET @query = 'ALTER LOGIN ' + QUOTENAME(@username) + ' WITH PASSWORD =
''' + @new_password + ''';';
		EXEC (@query);

		COMMIT TRAN;
	END TRY
	BEGIN CATCH
		ROLLBACK TRAN;
		THROW;
	END CATCH;
END;
```

```
public bool ChangeUserPassword(string username, string newPassword)
        {
            bool res = true;
            try
            {
                BusManagementEntities db = new
BusManagementEntities(StaticEnv.GetDefaultEFConnectionString());
                if (UserData.IsPassenger)
                {
                    db.pro_ChangePassengerPassword(username, newPassword);
                }
                else
                {
                    db.pro_ChangeSystemPassword(username, newPassword);
                }
            }
            catch (SqlException err)
            {
                MessageBox.Show(err.Message);
                return false;
            }
            catch (Exception err)
            {
                MessageBox.Show(err.Message);
                return false;
            }
            return res;
        }
```

CHAPTER IV: USER CREATION AND PRIVILEGE DISTRIBUTION

Applicable towards users (Passengers, Staff and Server Admin)

* For passengers

```
exec sp_addrole rol_Passenger;
-- grant tables
grant select on AGENT to rol_Passenger;
grant select, insert, delete, references on BOOKING to rol_Passenger;
grant select on BUS to rol_Passenger;
grant select on BUSROUTE to rol_Passenger;
grant select on BUSSTATION to rol_Passenger;
grant select, insert, update, references on PASSENGER to rol_Passenger;
grant select, insert, update, references on PASSENGERACCOUNT to rol_Passenger;
grant select on PLACE to rol_Passenger;
grant select on TICKET to rol_Passenger;
grant select on TRIP to rol_Passenger;
--
--grant views
grant select on V_AGENTINFOR to rol_Passenger;
grant select on V_AVAILABLETRIP to rol_Passenger;
grant select on V_BOOKEDTICKET to rol_Passenger;
grant select on V_BOOKINGINFOR to rol_Passenger;
grant select on V_BUSSTATIONINFOR to rol_Passenger;
grant select on V_ROUTEINFOR to rol_Passenger;
grant select on V_TRIPINFOR to rol_Passenger;
grant select on V_USERINFOR to rol_Passenger;
--
-- grant procedures
grant execute on dbo.pro_AddBooking to rol_Passenger;
grant execute on dbo.pro_AddDefaultBooking to rol_Passenger;
grant execute on dbo.pro_AddPassenger to rol_Passenger;
grant execute on dbo.pro_AddPassengerAccount to rol_Passenger
grant execute on dbo.pro_CancelTicket to rol_Passenger;
--
-- grant functions
grant execute on dbo.func_auto_id_booking to rol_Passenger;
grant execute on dbo.func_auto_id_passenger to rol_Passenger;
--grant execute on dbo.func_GetAvailabelSeat to rol_Passenger;
--
-- deny
deny delete on PASSENGER to rol_Passenger;
deny update, insert, delete, references on AGENT to rol_Passenger;
deny update, insert, delete, references on BUS to rol_Passenger;
deny update, insert, delete, references on BUSROUTE to rol_Passenger;
deny update, insert, delete, references on BUSSTATION to rol_Passenger;
deny select, update, insert, delete, references on CASHRESERVE to rol_Passenger;
deny select, update, insert, delete, references on EMPLOYEE to rol_Passenger;
deny delete on PACKAGE to rol_Passenger;
deny update, insert, delete, references on PLACE to rol_Passenger;
deny select, update, insert, delete, references on POSITION to rol_Passenger;
deny select, update, insert, delete, references on SYSTEMACCOUNT to rol_Passenger;
deny update, insert, delete, references on TRIP to rol_Passenger;
```

* For staff

```sql
exec sp_addrole rol_Staff;
-- grant tables
grant select on AGENT to rol_Staff;
grant select, insert, delete, references on BOOKING to rol_Staff;
grant select on BUS to rol_Staff;
grant select on BUSROUTE to rol_Staff;
grant select on BUSSTATION to rol_Staff;
grant select, insert, update, delete, references on PASSENGER to rol_Staff;
grant select, insert, update, delete, references on PASSENGERACCOUNT to rol_Staff;
grant select on PLACE to rol_Staff;
grant select, update on TICKET to rol_Staff;
grant select on TRIP to rol_Staff;
grant select, update on EMPLOYEE to rol_Staff;
grant select, update on SYSTEMACCOUNT to rol_Staff;
--
--grant views
grant select on V_AGENTINFOR to rol_Staff;
grant select on V_AVAILABLETRIP to rol_Staff;
grant select on V_BOOKEDTICKET to rol_Staff;
grant select on V_BOOKINGINFOR to rol_Staff;
grant select on V_BUSSTATIONINFOR to rol_Staff;
grant select on V_ROUTEINFOR to rol_Staff;
grant select on V_TRIPINFOR to rol_Staff;
grant select on V_USERINFOR to rol_Staff;
grant select on V_EMPLOYEEINFOR to rol_Staff;
grant select on V_DRIVERINFOR to rol_Staff;
--
-- grant procedures
grant execute on dbo.pro_AddBooking to rol_Staff;
grant execute on dbo.pro_AddDefaultBooking to rol_Staff;
grant execute on dbo.pro_AddPassenger to rol_Staff;
grant execute on dbo.pro_AddPassengerAccount to rol_Staff
grant execute on dbo.pro_CancelTicket to rol_Staff;
--
-- grant functions
grant execute on dbo.func_auto_id_booking to rol_Staff;
grant execute on dbo.func_auto_id_passenger to rol_Staff;
--grant execute on dbo.func_GetAvailabelSeat to rol_Staff;
grant execute on dbo.func_auto_id_employee to rol_Staff;
--
-- deny
deny delete on PASSENGER to rol_Staff;
deny update, insert, delete, references on AGENT to rol_Staff;
deny update, insert, delete, references on BUS to rol_Staff;
deny update, insert, delete, references on BUSROUTE to rol_Staff;
deny update, insert, delete, references on BUSSTATION to rol_Staff;
deny select, update on CASHRESERVE to rol_Staff;
deny insert, delete, references on EMPLOYEE to rol_Staff;
deny update, insert, delete, references on PLACE to rol_Staff;
deny select, update, insert, delete, references on POSITION to rol_Staff;
deny insert, delete, references on SYSTEMACCOUNT to rol_Staff;
deny update, insert, delete, references on TRIP to rol_Staff;
```

* For server admin

```
exec sp_addrole rol_admin;
grant control on DATABASE::BusManagement to rol_admin;
```

*** [C#] Create an account for a passenger and assign the Passenger role to the account**

When the user doesn't have an account to use the app, they will have the option to sign up for a brand new account.

This is when they will input certain information including their name, phone number, the account's username and password, which will be used by these procedures: pro_AddPassenger, pro_AddPassengerAccount.

```
if (!string.IsNullOrEmpty(passengerId))
{
    db.pro_AddPassenger(passengerId, name, phone);
    db.pro_AddPassengerAccount(passengerId, username, password); // add passenger account and assign privilege
    errMsg = "Create new user successfully!. No error";
}
```

pro_AddPassenger will insert the new information into Passenger table.

```
CREATE proc [dbo].[pro_AddPassenger] @id_passenger char(20), @name nvarchar(50),
@phone char(20)
as
begin
insert into PASSENGER(id_passenger, name, phone_number) values (@id_passenger, @name,
@phone);
end;
```

pro_AddPassengerAccount will do 2 things: Insert new username and password into PassengerAccount table then assign Passenger role to the user.

```
CREATE proc [dbo].[pro_AddPassengerAccount] @id_passenger char(20), @username
varchar(50), @password varchar(50)
as
begin
insert into PASSENGERACCOUNT values (@id_passenger, @username, @password);
exec dbo.pro_AssignPassengerPrivilege @id_passenger;
end;
```

*** Procedure to assign privileges to passengers**

```
CREATE proc [dbo].[pro_AssignPassengerPrivilege] (@id_passenger char(20))
as
begin
SET XACT_ABORT ON
	begin tran
		begin try
			declare @username varchar(50), @sqlString nvarchar(MAX)
			select @username = PASSENGERACCOUNT.username from PASSENGERACCOUNT
where PASSENGERACCOUNT.id_passenger = @id_passenger
			set @sqlString = 'exec sp_addrolemember ''rol_Passenger'', ''' +
@username + ''''
			exec (@sqlString)
			commit tran
		end try
		begin catch
			rollback
		end catch
end
```

* Trigger for PassengerAccount table when a new row of data is created

```
CREATE trigger [dbo].[tr_CreatePassengerAccount] on [dbo].[PASSENGERACCOUNT]
after insert
as
declare @username varchar(30), @password varchar(10)
select @username = ins.username, @password = ins.password from inserted ins
begin
	begin tran
		begin try
			declare @sql nvarchar(max);
			set @sql = 'create login ' + quotename(@username) + ' with
password = ''' + @password + ''', DEFAULT_DATABASE=[BusManagement],
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF';
			exec sp_executesql @sql;
			set @sql = 'create user ' + quotename(@username) + ' for login ' +
quotename(@username);
			exec sp_executesql @sql;
			commit tran;
		end try
		begin catch
			rollback
		end catch
end
```

* Procedure to assign system privileges to staff members and admins

```
CREATE proc [dbo].[pro_AssignSystemPrivilege] (@id_employee char(20), @id_position
char(20))
as
begin
	SET XACT_ABORT ON
	begin tran
		begin try
			declare @username varchar(50), @position_name varchar(50),
@sqlString varchar(1000)
		--
			select @username = b.username
			from EMPLOYEE as a inner join SYSTEMACCOUNT as b on a.id_account =
b.id_account
			where a.id_employee = @id_employee;
			--
			select @position_name = a.type from POSITION as a where
a.id_position = @id_position;
		--
			if(@position_name = 'administrator')
				set @sqlString = 'exec sp_addrolemember ''rol_Admin'', '''
+ @username + '''';
			else
				set @sqlString = 'exec sp_addrolemember ''rol_Staff'', '''
+ @username + '''';
			exec (@sqlString);
			insert into EMPLOYEE_POSITION values(@id_employee, @id_position);
			commit tran;
		end try
		begin catch
		rollback;
		end catch
end
```

*** Sensitive class to load users' information after logging in**

```
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;
    //
```

*** C# Function to check whether the login credentials are of a Passenger's, an Admin's or a Staff's and save their information**

```
private void BtnLogin_Click(object sender, EventArgs e)
{
    string username = this.TbUsername.Text;
    string password = this.TbPassword.Text;
    string passengerId = string.Empty;
    string employeeId = string.Empty;
    string errMsg = string.Empty;
    //
    bool isLogin = new BSLogin().ValidateUser(username, password,ref passengerId, ref employeeId, ref errMsg);
    UserData.ClearUserData();
    if (isLogin && !string.IsNullOrEmpty(passengerId))
    {
        UserData.IsPassenger = true;
        UserData.SetUserLoginData(username, password);
        UserData.SetPassengerId(passengerId);
        //
        V_USERINFOR curUser = new BSLogin().GetUser(passengerId);
        UserData.SetUserData(curUser.name.Trim(), curUser.phone_number.Trim());
        //
        PASSENGER curPassenger = new BSPassenger().GetPassenger(passengerId);
        UserData.Email = curPassenger?.email?.Trim();
        UserData.Gender = curPassenger.gender;
        //
        //
        Handler_LoginSuccessfully();
    }
```

```
else if(isLogin && !string.IsNullOrEmpty(employeeId))
{
    bool isAdmin = new BSLogin().IsAdmin(employeeId);
    UserData.ClearUserData();
    if (isAdmin)
    {
        UserData.IsAdmin = true;
    }
    else
    {
        UserData.IsStaff = true;
    }
    UserData.SetUserLoginData(username, password);
    UserData.SetSystemId(employeeId);
    //
    V_EMPLOYEEINFOR curEmployee = new BSLogin().GetEmployee(employeeId);
    UserData.SetUserData(curEmployee.Name.Trim(), curEmployee.Phone_Number.Trim());
    //
    UserData.Email = curEmployee.Email?.Trim();
    UserData.Gender = curEmployee.Gender;
    Handler_LoginSuccessfully();
}
else
{
    this.LbErrorMessage.Text = errMsg;
}
```

When logging in, the app will use the default connection string. After logging in, the app will switch to a different connection string that contains the saved information that was input by the user.

*** [C#] Code inside StaticEnv**

```
public static string GetEFConnectionString(string username, string password)
        =>
$"metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnect
ion.BusManagementModel.ssdl|res://*/src.dbConnection.BusManagementModel.msl;provi
der=System.Data.SqlClient;provider connection string=\"data
source=(local);initial catalog=BusManagement;user
id={username};password={password};MultipleActiveResultSets=True;App=EntityFramewo
rk\"";
        public static string GetEFConnectionString()
            =>
$"metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnect
ion.BusManagementModel.ssdl|res://*/src.dbConnection.BusManagementModel.msl;provi
der=System.Data.SqlClient;provider connection string=\"data
source=(local);initial catalog=BusManagement;user
id={UserData.Username};password={UserData.Password};MultipleActiveResultSets=True
;App=EntityFramework\"";

        public static string GetDefaultEFConnectionString()
            =>
"metadata=res://*/src.dbConnection.BusManagementModel.csdl|res://*/src.dbConnecti
on.BusManagementModel.ssdl|res://*/src.dbConnection.BusManagementModel.msl;provid
er=System.Data.SqlClient;provider connection string=\"data source=(local);initial
catalog=BusManagement;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework\"";
```

*** Creating a new database instance to connect to the database using the Staff/Employee Connection String:**

```
public BusManagementEntities(string connectionStr) : base(connectionStr)
        {
        }
```

```
BusManagementEntities db = new
BusManagementEntities(StaticEnv.GetDefaultEFConnectionString());
```

*** Creating a new database instance to connect to the database using the User Connection String:**

```
public BusManagementEntities() : base(StaticEnv.GetEFConnectionString())
        {
        }
```

```
BusManagementEntities db = new BusManagementEntities();
```

CHAPTER V: SYSTEM INTERFACE DESIGN

1. Applications and services used

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

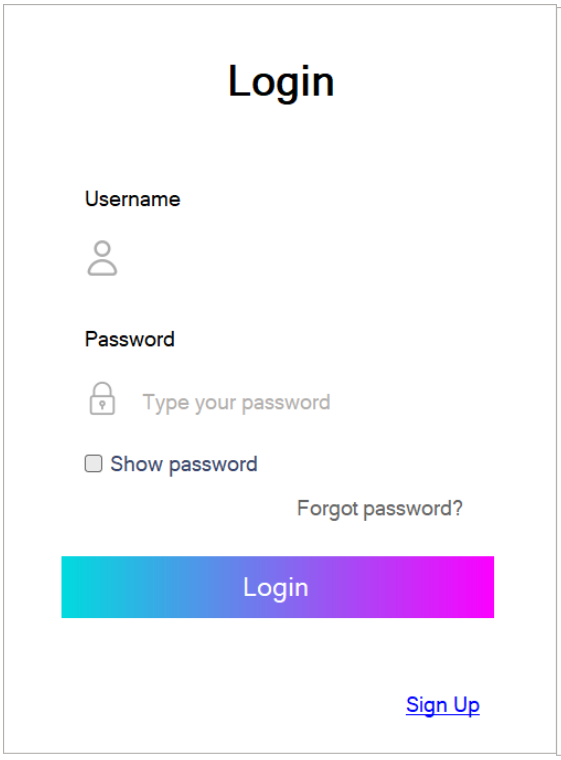
- Microsoft SQL Server 2022
- Microsoft SQL Server Management Studio 19 (SSMS 19)
- Windows Forms App (.NET Framework) built using Visual Studio 2022
- Packages: Entity Framework, Mailkit, Mimekit, BouncyCastle.Cryptography

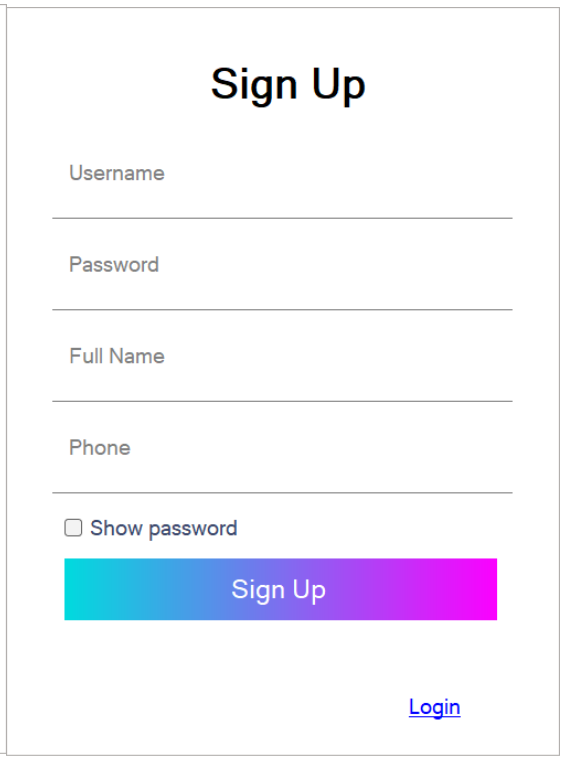
2. Software interface

* 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.

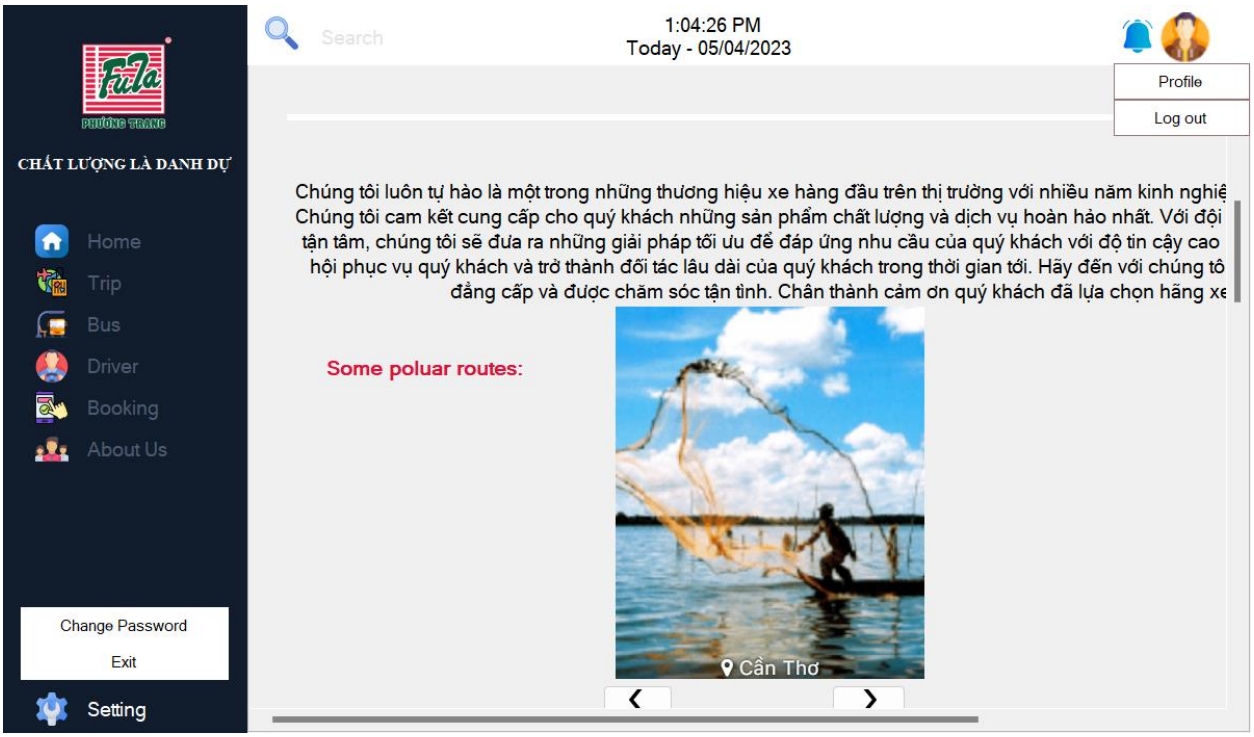
In the case a user does not possess an account to use the software, they can sign up for a new account after filling out certain information criteria.



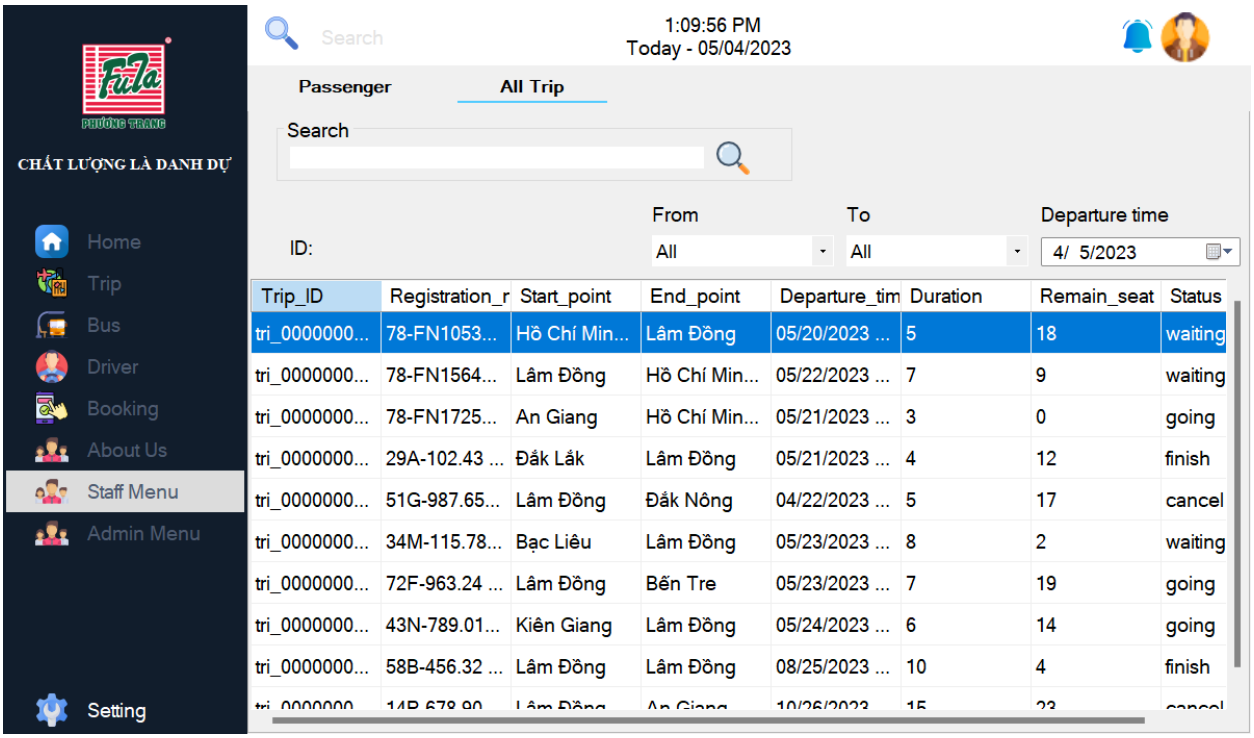


* Home screen

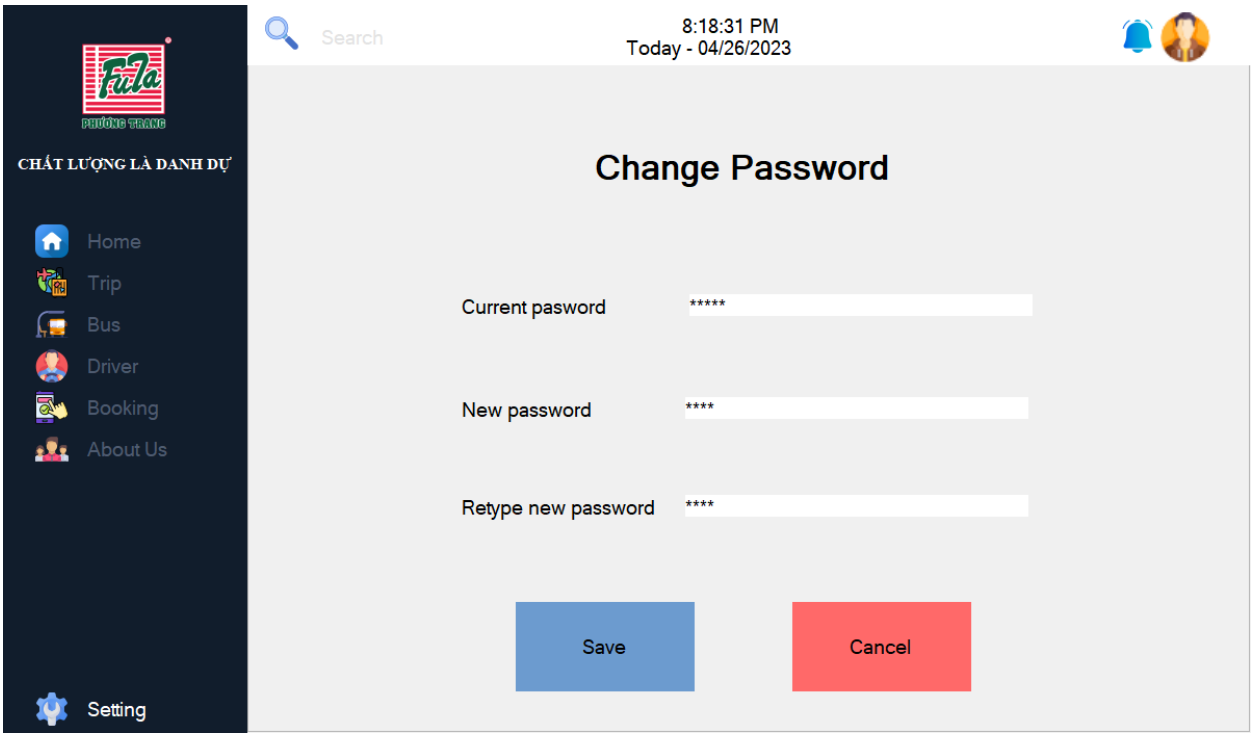
- For passengers



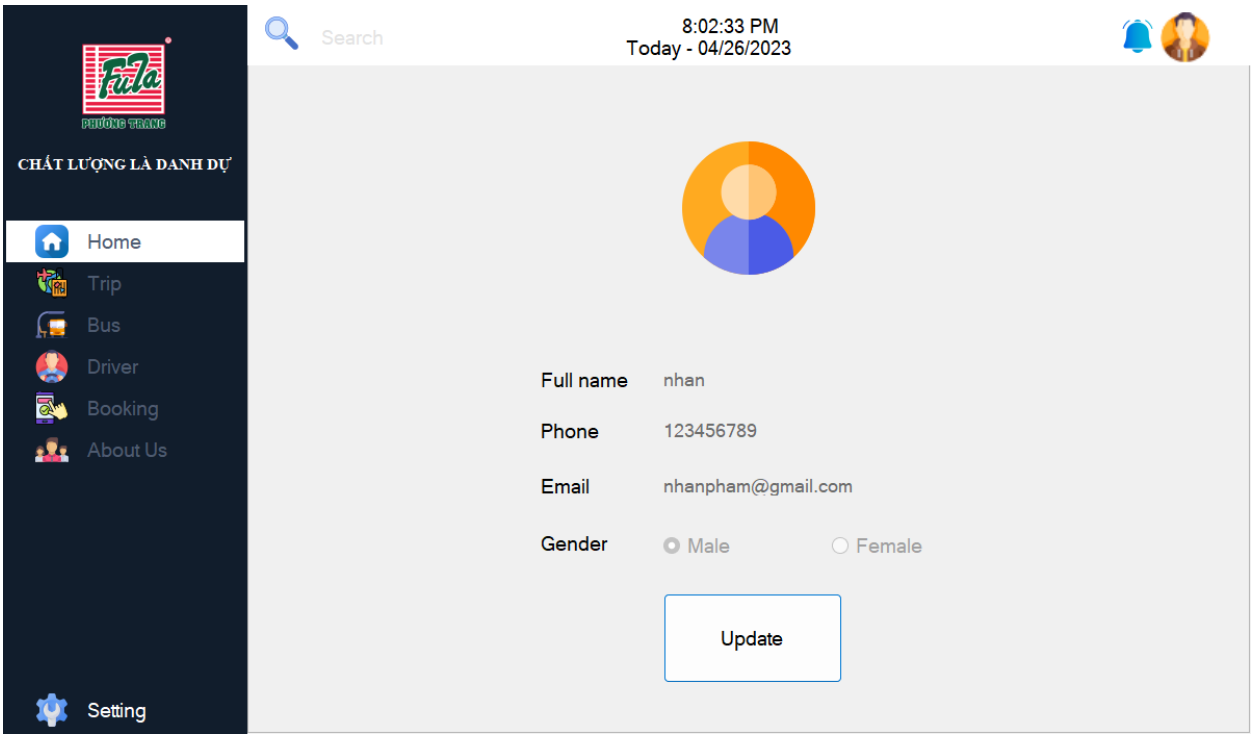
- For server admins



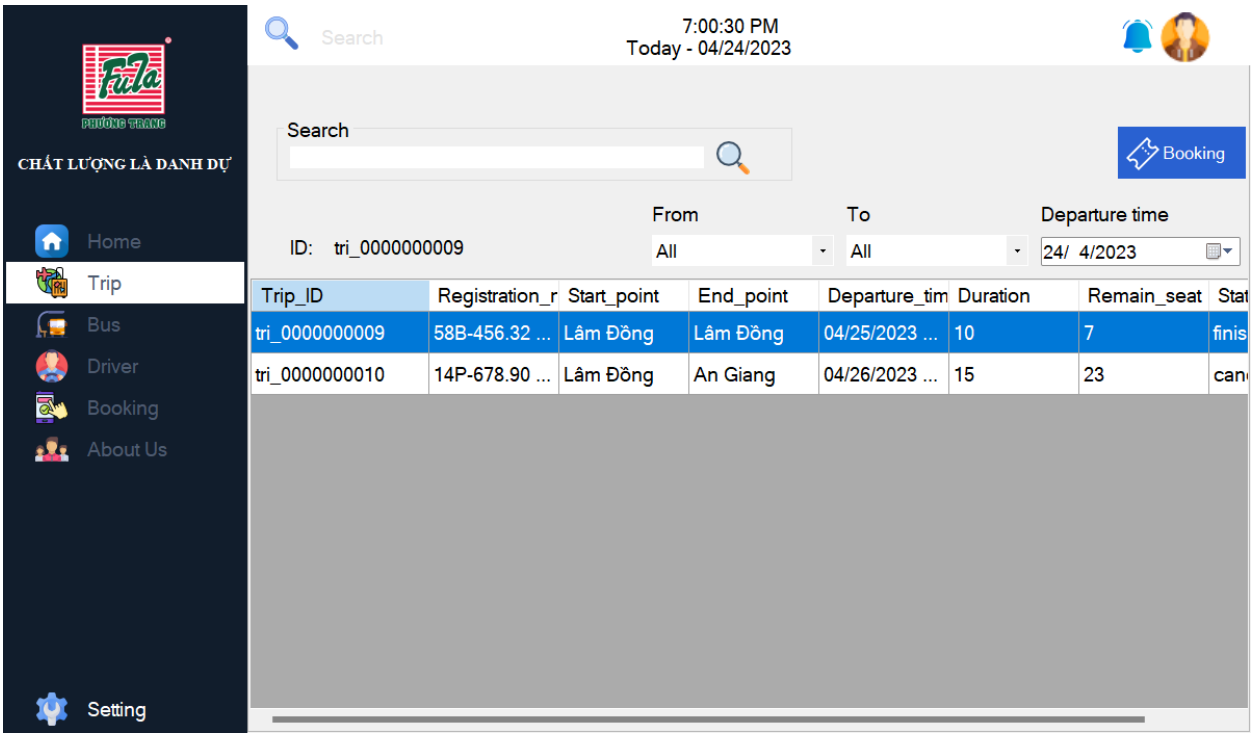
* Change password screen



*** Profile screen**



*** Trip screen**



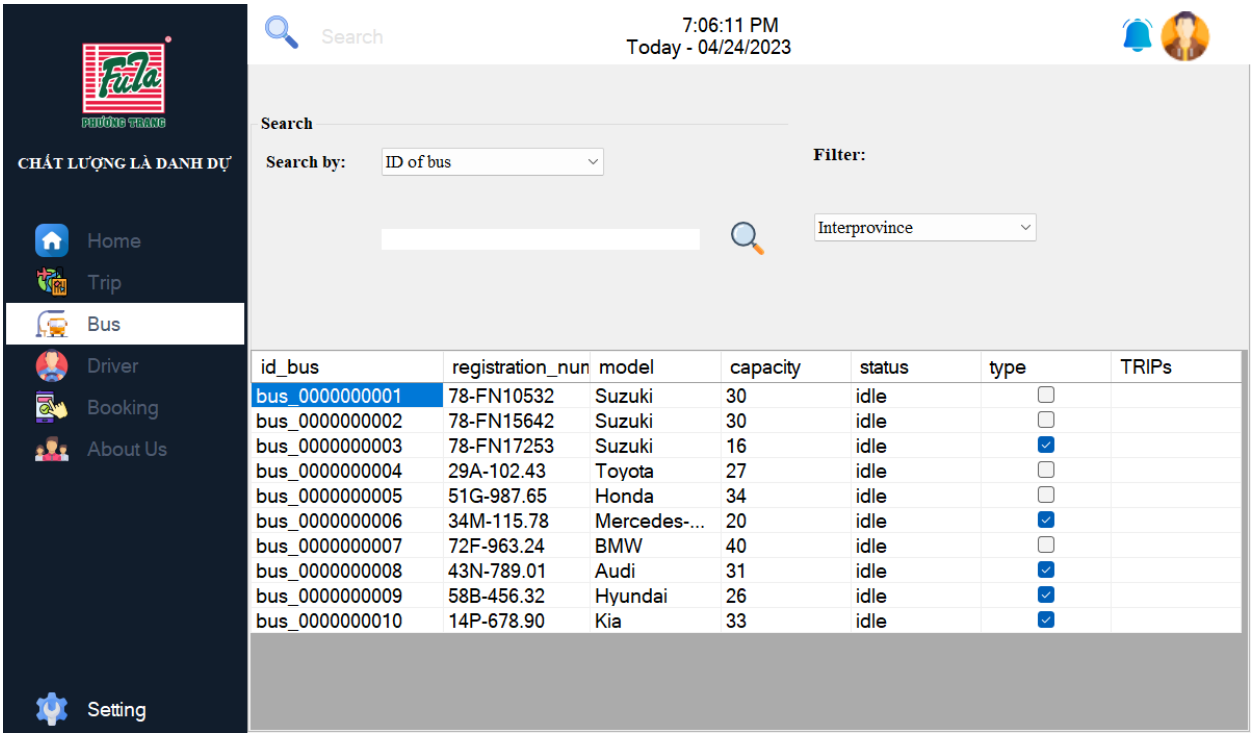
- Get all trip data

```
public List<V_TRIPINFOR> GetAllTrips(DateTime dateTime)
{

    BusManagementEntities db = new BusManagementEntities();
    var res = db.V_TRIPINFOR.ToList();
    return res;
}
```

- Search for available trips

```
        public List<V_AVAILABLETRIP> SearchAvailableTrips(string input, string
src, string des, DateTime dateTime)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = db.V_AVAILABLETRIP.Where(d => d.Departure_time > dateTime);
            if (src != "All")
            {
                res = res.Where(d => d.Start_point == src);
            }
            if (des != "All")
            {
                res = res.Where(d => d.End_point == des);
            }
            if (!string.IsNullOrEmpty(input))
            {
                res = res.Where(d => d.Trip_ID.Contains(input.Trim()));
            }
            return res.ToList();
        }
```

* Bus screen



- Get all bus information

```
        public List<BUS> GetAllBus()
        {

            BusManagementEntities db = new BusManagementEntities();
            var res = db.BUSes.ToList();
            return res;
        }
```

- Search bus by ID

```
        public List<BUS> SearchBusByID(string input, bool type)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = FilterBus(type);
            if (!string.IsNullOrEmpty(input))
            {
                res = res.Where(d => d.id_bus.Contains(input)).ToList();
                return res.ToList();
            }
            return res.ToList();
        }
```

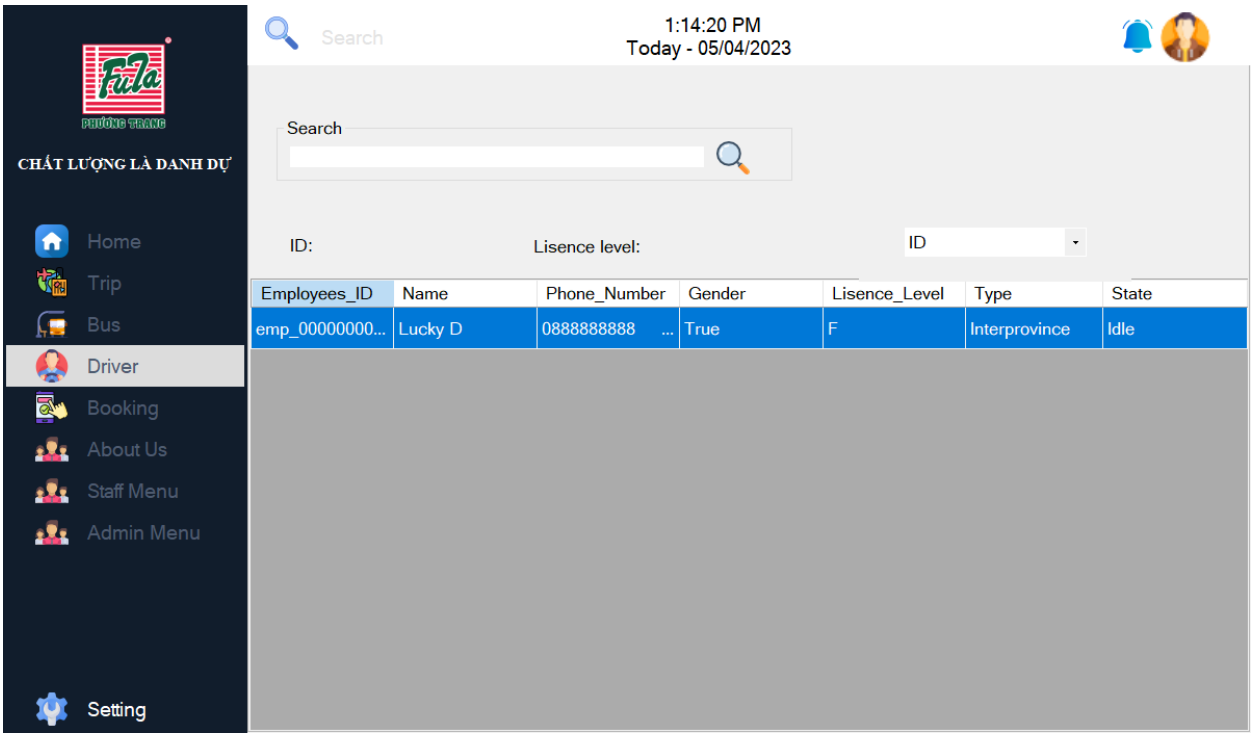
- Search bus by registration number

```
        public List<BUS> SearchBusByRegistrationNumber(string input, bool type)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = FilterBus(type);
            if (!string.IsNullOrEmpty(input))
            {
                res = res.Where(d =>
d.registration_number.Contains(input)).ToList();
                return res.ToList();
            }
            return res.ToList();
        }
```

- Filter bus by type (Interprovince/Transit)

```
        public List<BUS> FilterBus(bool type)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = db.BUSes.Where(d => d.type == type);
            return res.ToList();
        }
```

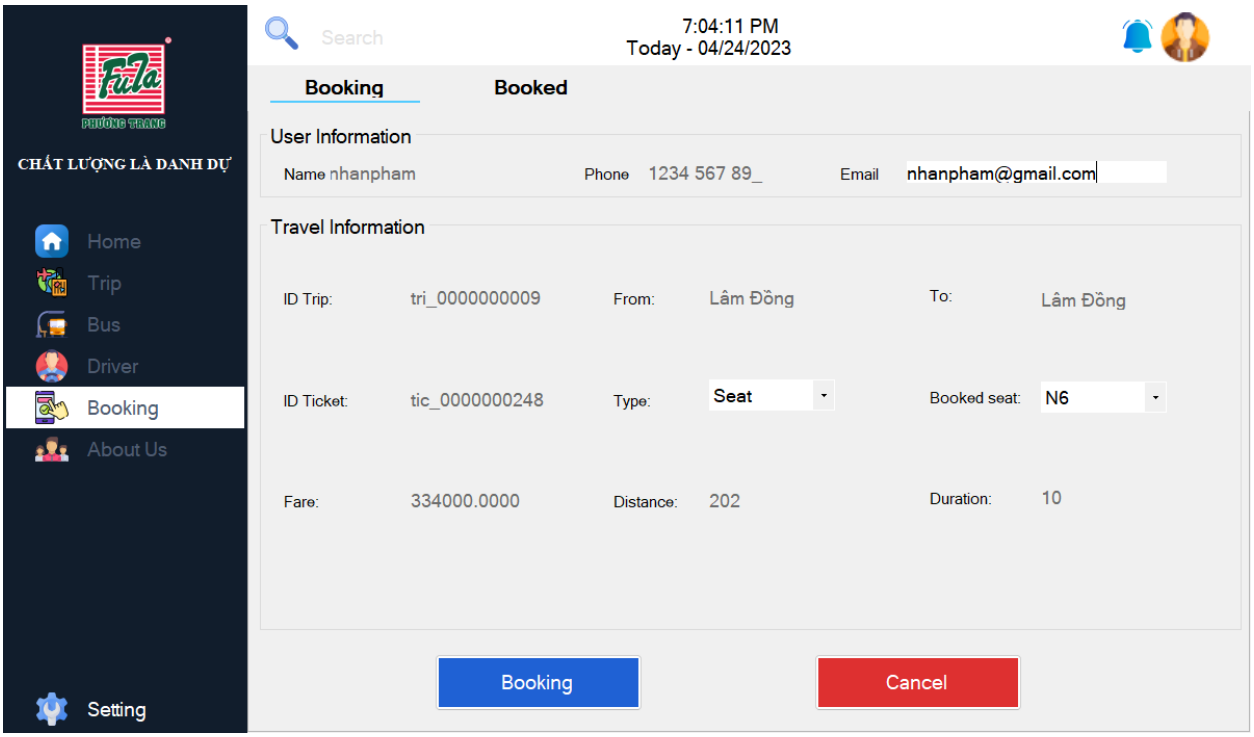
* Driver screen

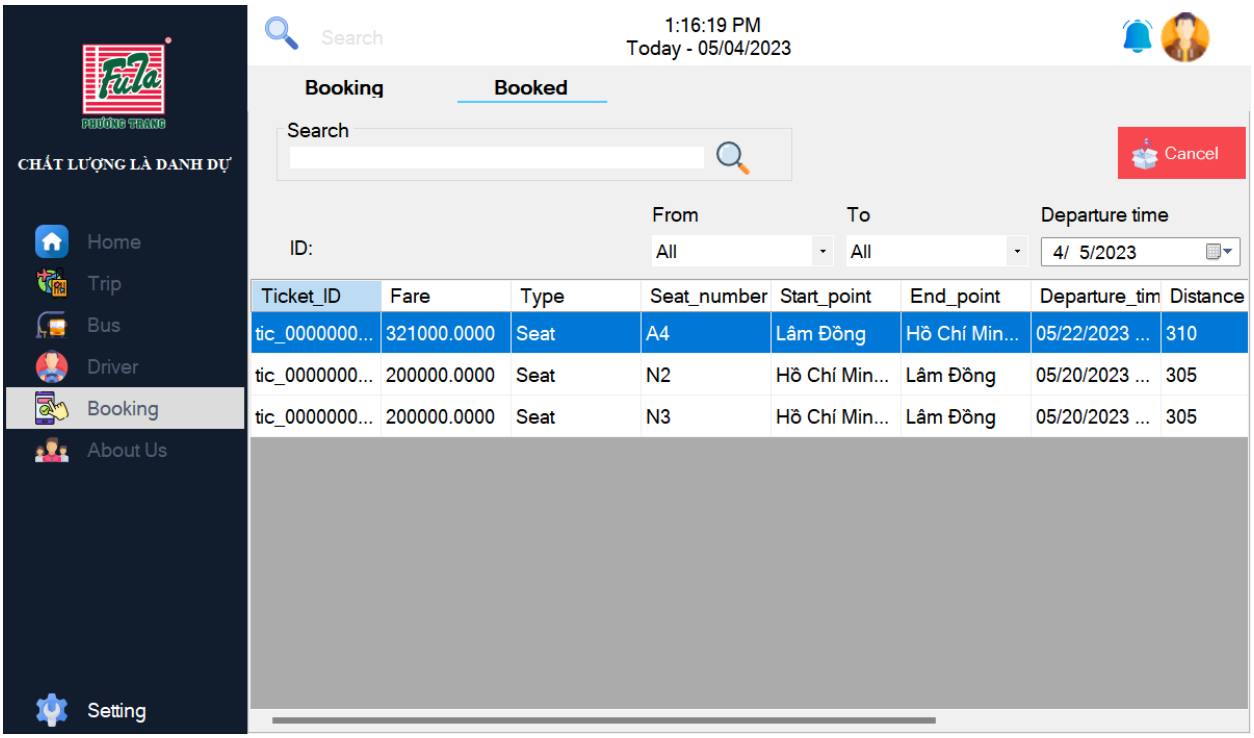


- Get all drivers information

```
private void LoadMainData()
{
    BSDriver bSDriver = new BSDriver();
    this.DgvMainData.DataSource = bSDriver.GetAllDrivers();
}
```

* Booking & Booked screen





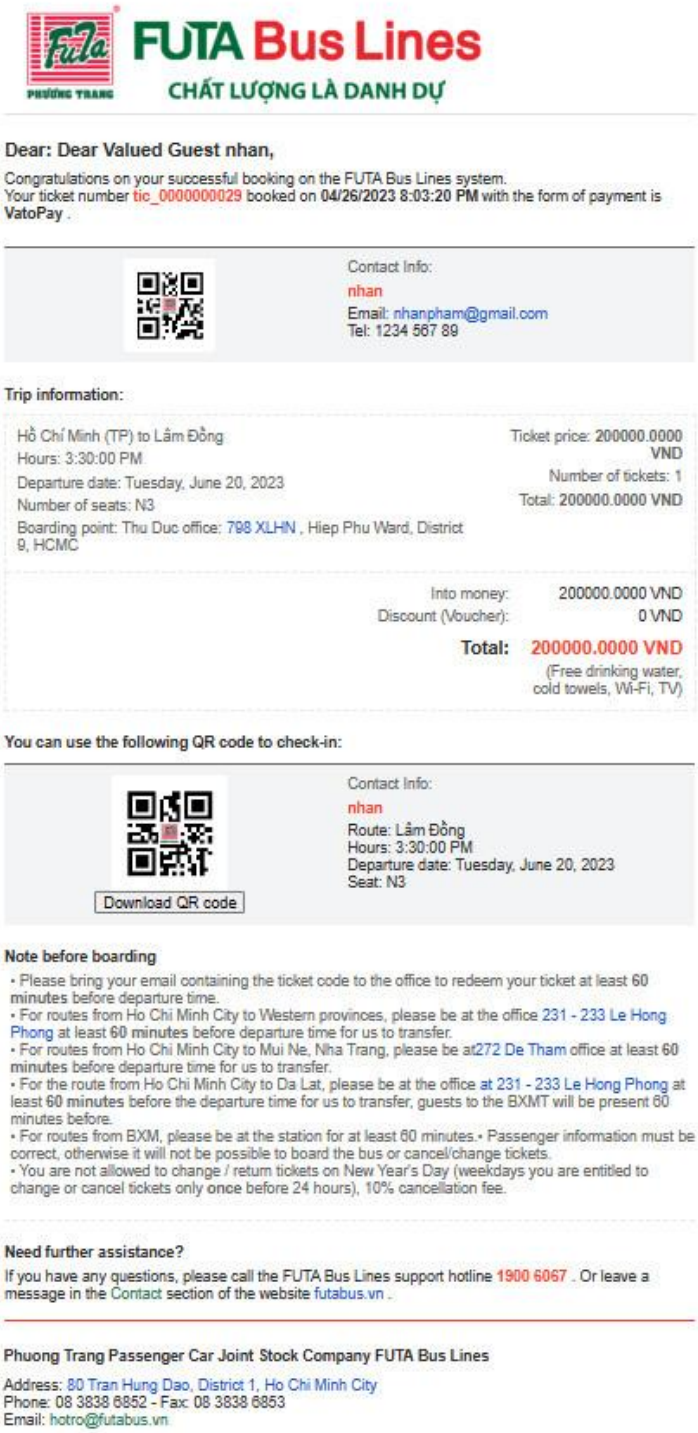
- Get available seats

```
        public List<string> GetAvailableSeat(string idTrip, int type) // type 0:
seat, 1: sleeper
        {
            BusManagementEntities db = new BusManagementEntities();
            string funcName = "func_GetAvailabelSeat";
            List<string> ticketList = BSMain.RunTableValuedFunc(funcName, new
List<string> { idTrip, type.ToString() });
            return ticketList;
        }
```

- Get booked ticket information

```
        public List<V_BOOKEDTICKET> SearchBookedTickets(string passengerId,string
input, string src, string des, DateTime dateTime)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = db.V_BOOKEDTICKET.Where(d => d.Departure_time > dateTime &&
d.Passenger_ID == passengerId);
            if (src != "All")
            {
                res = res.Where(d => d.Start_point == src);
            }
            if (des != "All")
            {
                res = res.Where(d => d.End_point == des);
            }
            if (!string.IsNullOrEmpty(input))
            {
                res = res.Where(d => d.Ticket_ID.Contains(input.Trim()));
            }
            return res.ToList();
        }
```

* Email received to confirm after successful booking



FUTA Bus Lines

CHẤT LƯỢNG LÀ DANH DỰ

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
Hours: 3:30:00 PM
Departure date: Tuesday, June 20, 2023
Number of seats: N3
Boarding point: Thu Duc office: 798 XLHN , Hiep Phu Ward, District 9, HCMC

Ticket price: **200000.0000 VND**
Number of tickets: 1
Total: **200000.0000 VND**

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:

Download QR code

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

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 at272 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

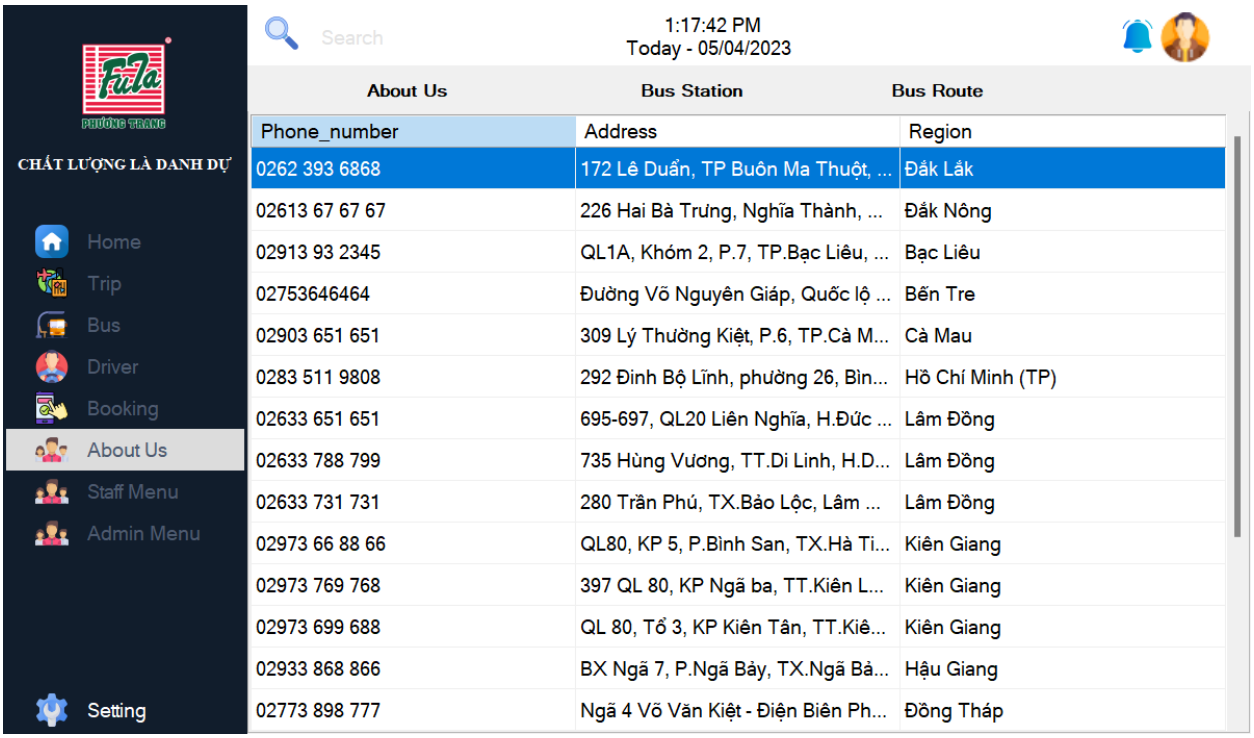
* [C#] Code inside IEmailSender.cs

```
internal interface IEmailSender
    {
        Task SendEmailAsync(string email, string subject, string body);
    }
```

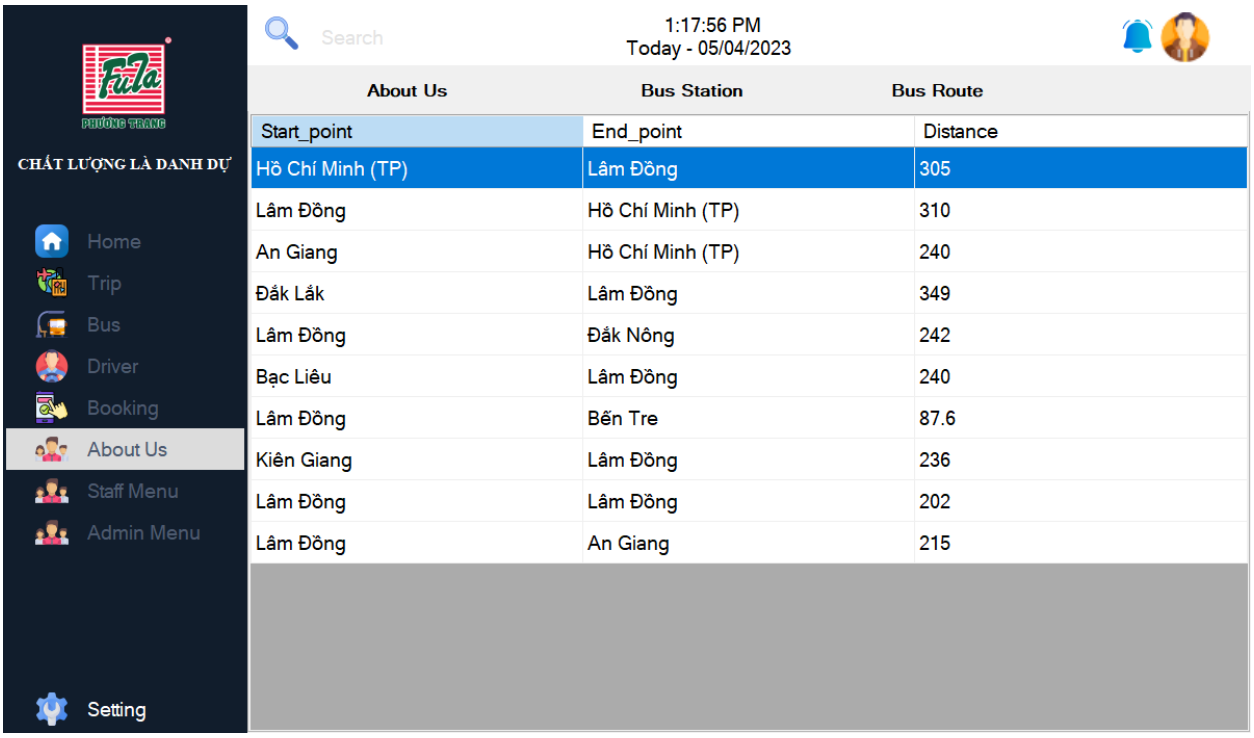
* [C#] Code inside EmailSender.cs

```
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();
            //
            SmtpClient smtpClient = new SmtpClient();
            //
            smtpClient.Connect(host, port, true);
            smtpClient.Authenticate(username, password);
            return smtpClient.SendAsync(message);
        }
```

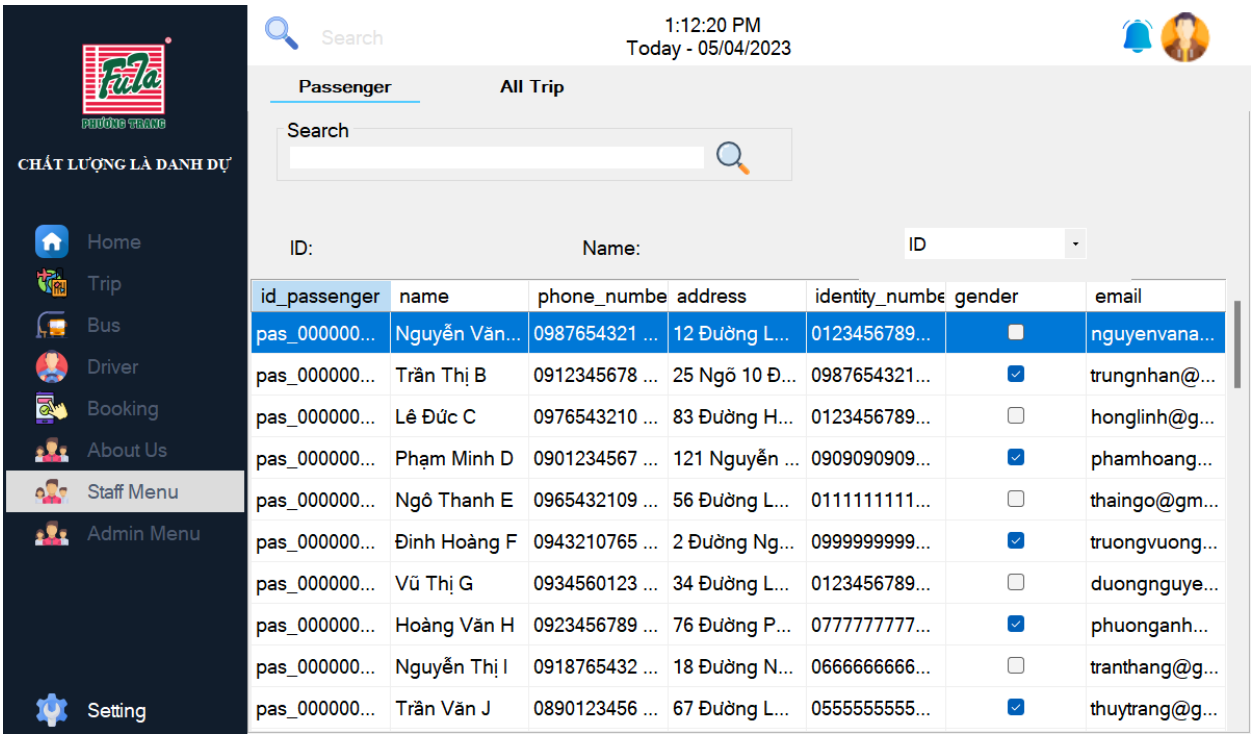
* About us screen

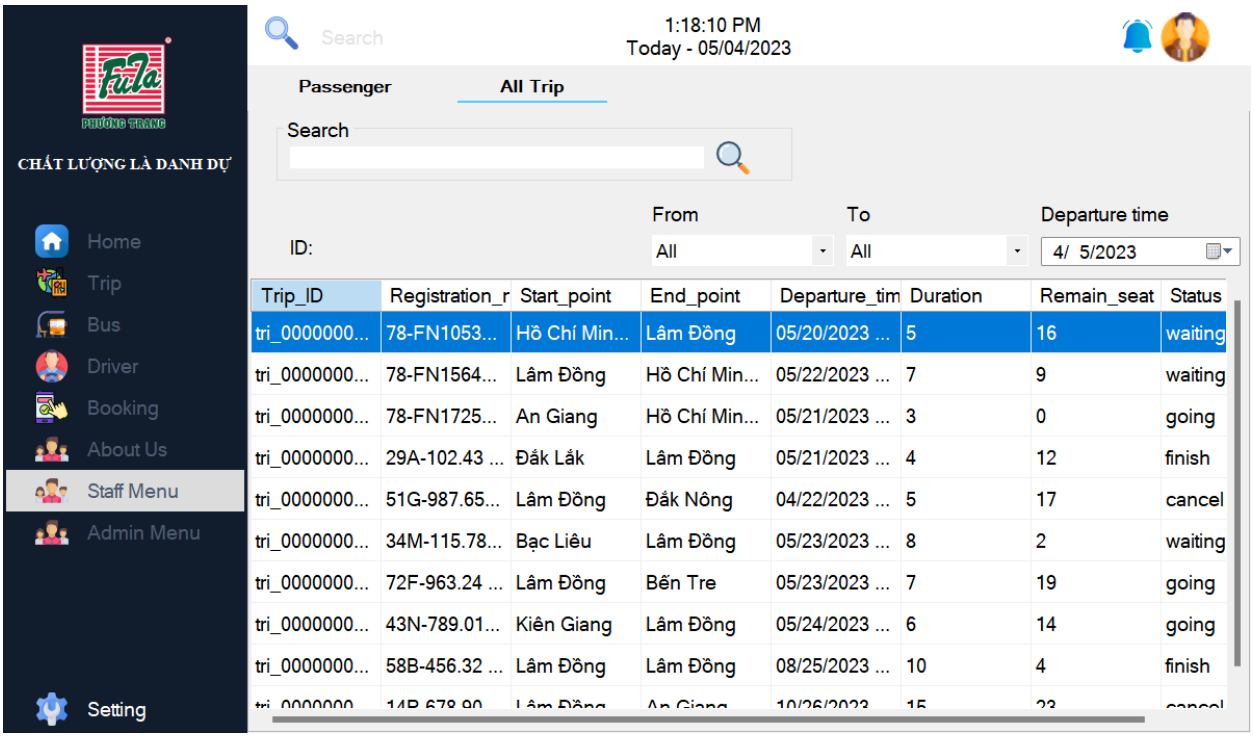


Phone_number	Address	Region
0262 393 6868	172 Lê Duẩn, TP Buôn Ma Thuột, ...	Đắk Lắk
02613 67 67 67	226 Hai Bà Trưng, Nghĩa Thành, ...	Đắk Nông
02913 93 2345	QL1A, Khóm 2, P.7, TP.Bạc Liêu, ...	Bạc Liêu
02753646464	Đường Võ Nguyên Giáp, Quốc lộ ...	Bến Tre
02903 651 651	309 Lý Thường Kiệt, P.6, TP.Cà 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



* Passenger & All trips screen (Only visible to server admin)





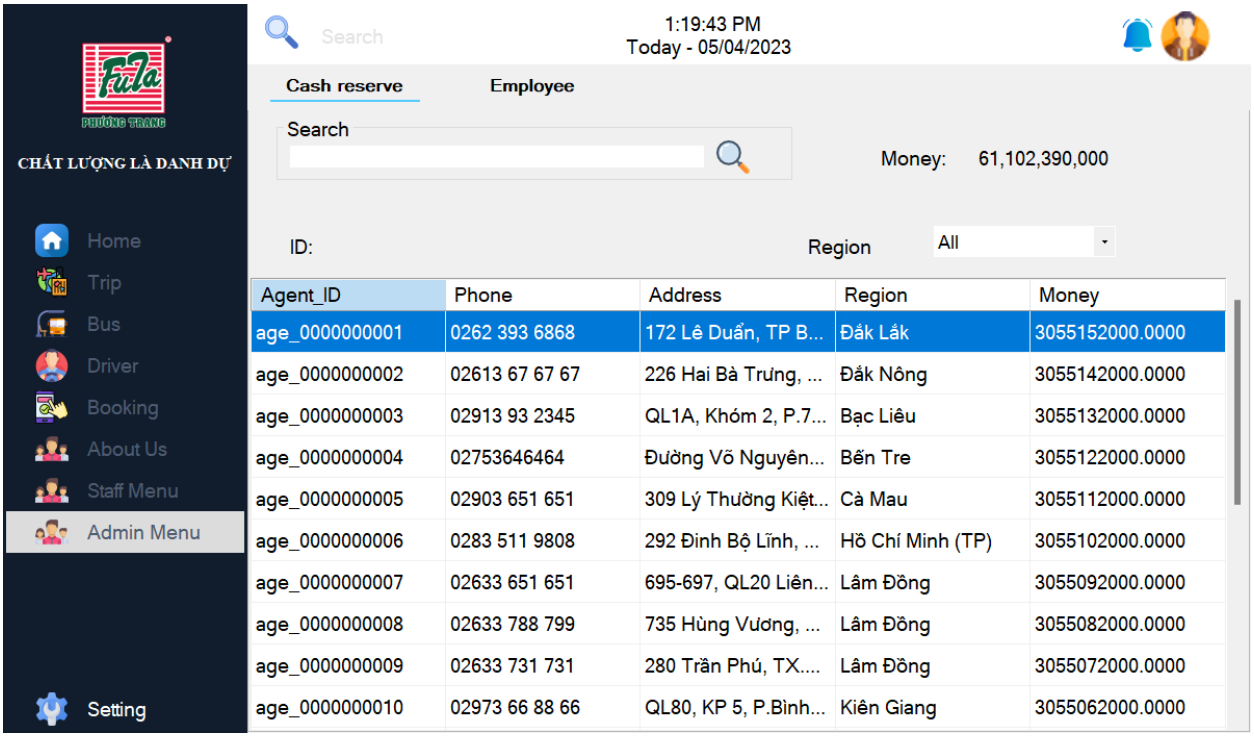
- Get passenger information

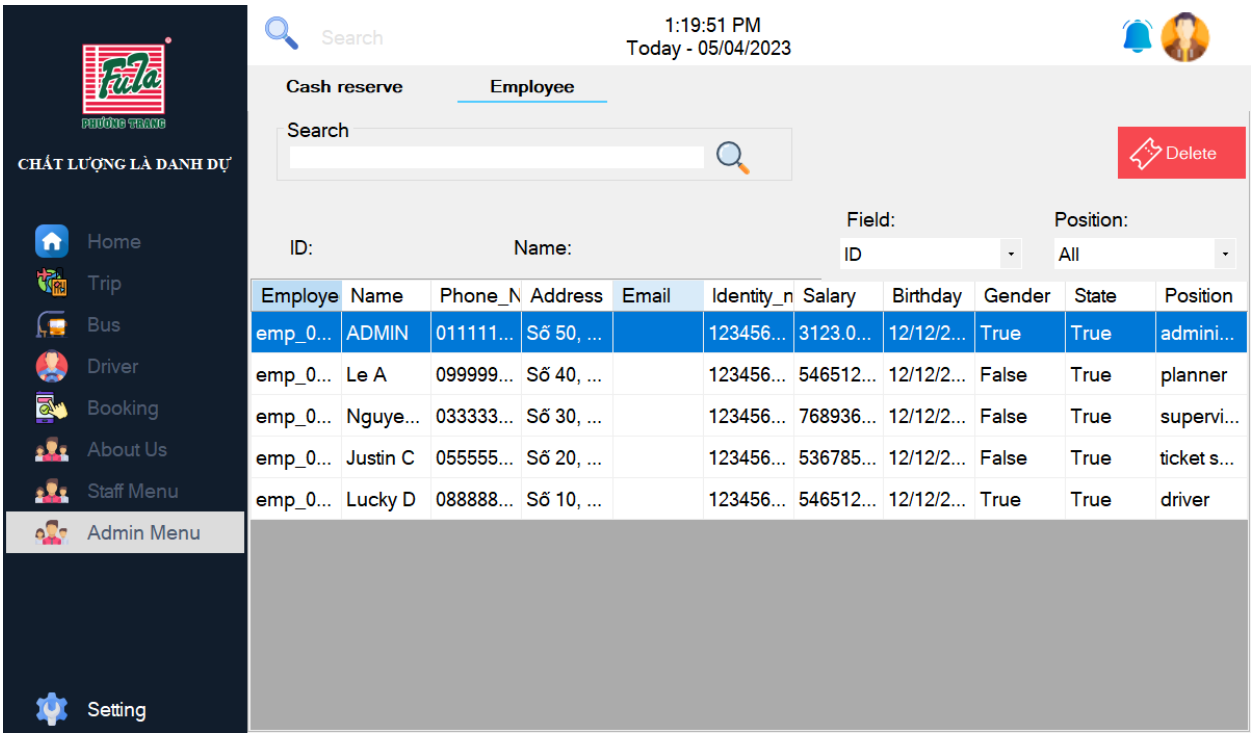
```
        private void FilterPassengers()
        {
            BSPassenger bsbooked = new BSPassenger();
            int tag = this.CbField.SelectedIndex;
            this.DgvMainData.DataSource = bsbooked.SearchPassenger(searchInput,
tag); // 0: ID, 1: Name
        }
```

- Get all trip information

```
        public List<V_TRIPINFOR> SearchTrips(string input, string src, string
des, DateTime dateTime)
        {
            BusManagementEntities db = new BusManagementEntities();

            var res = db.V_TRIPINFOR.Where(d => d.Departure_time > dateTime);
            if (src != "All")
            {
                res = res.Where(d => d.Start_point == src);
            }
            if (des != "All")
            {
                res = res.Where(d => d.End_point == des);
            }
            if (!string.IsNullOrEmpty(input))
            {
                res = res.Where(d => d.Trip_ID.Contains(input.Trim()));
            }
            return res.ToList();
        }
```

* Admin menu – Cash reserve & Employee (Only accessible by admins)





- Get cash reserve information

```
        private void FilterAgent()
        {
            BSAgent bsagent = new BSAgent();
            List<V_AGENTINFOR> dataSource =
bsagent.SearchAgents(this.searchInput.Trim(), this.CbRegion.Text.Trim());
            this.DgvMainData.DataSource = dataSource;
            this.LbSumMoney.Text = dataSource?.Aggregate(0m, (s, d) => s +
(decimal)d.Money).ToString("###,###,###,###");
        }
```

- Get employee information

```
        private void FilterEmployees()
        {
            BSEmployee bsemployee = new BSEmployee();
            int tag = this.CbField.SelectedIndex;
            string position = this.CbPosition.Text.Trim();
            this.DgvMainData.DataSource =
bsemployee.SearchEmployees(this.searchInput.Trim(), tag, position);
        }
```

(This page was intentionally left blank)