INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY, HCM CITY

School of Computer Science & Engineering

PROJECT REPORT

TOPIC 8: ANDROID LOCAL TRAIN TICKET SYSTEM

Lecturer: Nguyen Thi Thuy Loan

Course: Principle of Database Management

Group members:

Trần Hoàng Thịnh (Leader)	ITITIU19054
Lê Thị Huỳnh Giao	ITITIU19012
Nguyễn Gia Phúc	ITITIU19041
Phạm Hồng Đăng	ITITIU19009
Phan Thị Duyên Anh	ITDSIU19030

CONTENTS

Chapter 1. INTRODUCTION

Currently, the epidemic disease is becoming strained. However, some people working away from home want to reunite with their families. In addition, most of them choose to travel by train. It is mean that, there are thousands of people waiting in line to buy tickets which cause a negative impact on the epidemic. On the other hand, buying tickets at the ticket stations take times of the buyer and disputes can arise while waiting in line to buy tickets. Therefore, our team decide to choose the Android Local Train Ticketing for our topic.

Getting tickets online bring many benefits to customers. For instance, the passenger gets tickets easily. They are also no need to stand in line for getting ticket. It is more convenient when need not print tickets. In addition, for ticket companies, they have significant benefits such as easier access to customers, saving staff costs, advertising and easier payment methods with a bank account of customer.

Our project is an application ticket purchase system for an existing train station. This application is built on two main subjects. Each participant in the app is provided with an account. Through the initial login interface, the users depending on the role are given different functions.

a. Users:
   Users will be provided for the ability to look up trips based on place and date. The booking will be made when the user chooses the desired trip. The payment also happens online, via online transfer using a debit or credit card, directly on the application. Moreover, the users can search their tickets by ID Ticket if they forget.
b. Admin:
   Each admin has a unique account to add new users as well as their balance, and check view transactions.

Chapter 2. ENTITY – RELATIONSHIP DIAGRAM

2.1. Requirement

The Ticket booking system allows users to order and buy tickets over the systems. The manager can manage all the trip's information, customer's information so on. The user must log in to access the system by a unique User ID and password. Each User has a unique User ID, a name, address, mail, phone, age, gender, security question, answer.

The user system is organized into 3 roles: Admin, Ticket Collector, User.

The Admin can manage User through adding new User by Add Users and can manage the User's Balance by adding balance.

The User can book the ticket that is provided by Train information. And the Collector can collect the ticket by unique Ticket ID.

Each train has its own ID, source, destination, seat's information, time, date, price. Each ticket has its ID, seat number, and total price of train it is provided. Each train include many seats which has a unique Seat number.

The balance will be recharged from the users by card. Each user's Balance has a unique card number and balance.

2.2. Entity Relationship Diagram (ERD)

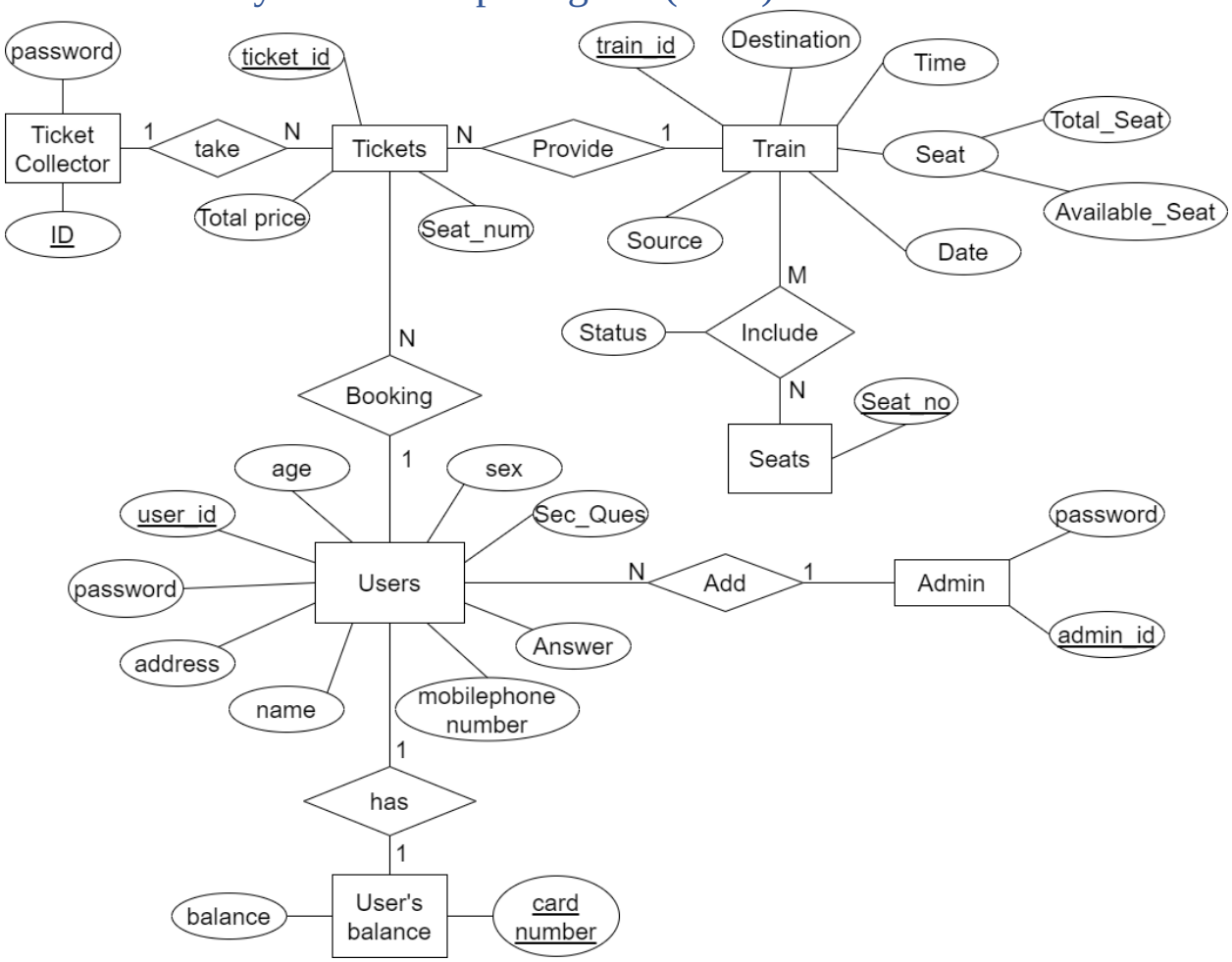


Figure 2.2 Entity Relationship Diagram of Booking Ticket System

Advantage:

Easy to visualize the relationship among entities and relationships.
It is an effective communication tool for database designer.
It is highly integrated with the relational model.

Disadvantages:

Some information could be hidden in ER model.
Limited relationship representation
No representation of data manipulation
Popular for high level design

Chapter 3. RELATIONAL MODEL

3.1. Relational Model

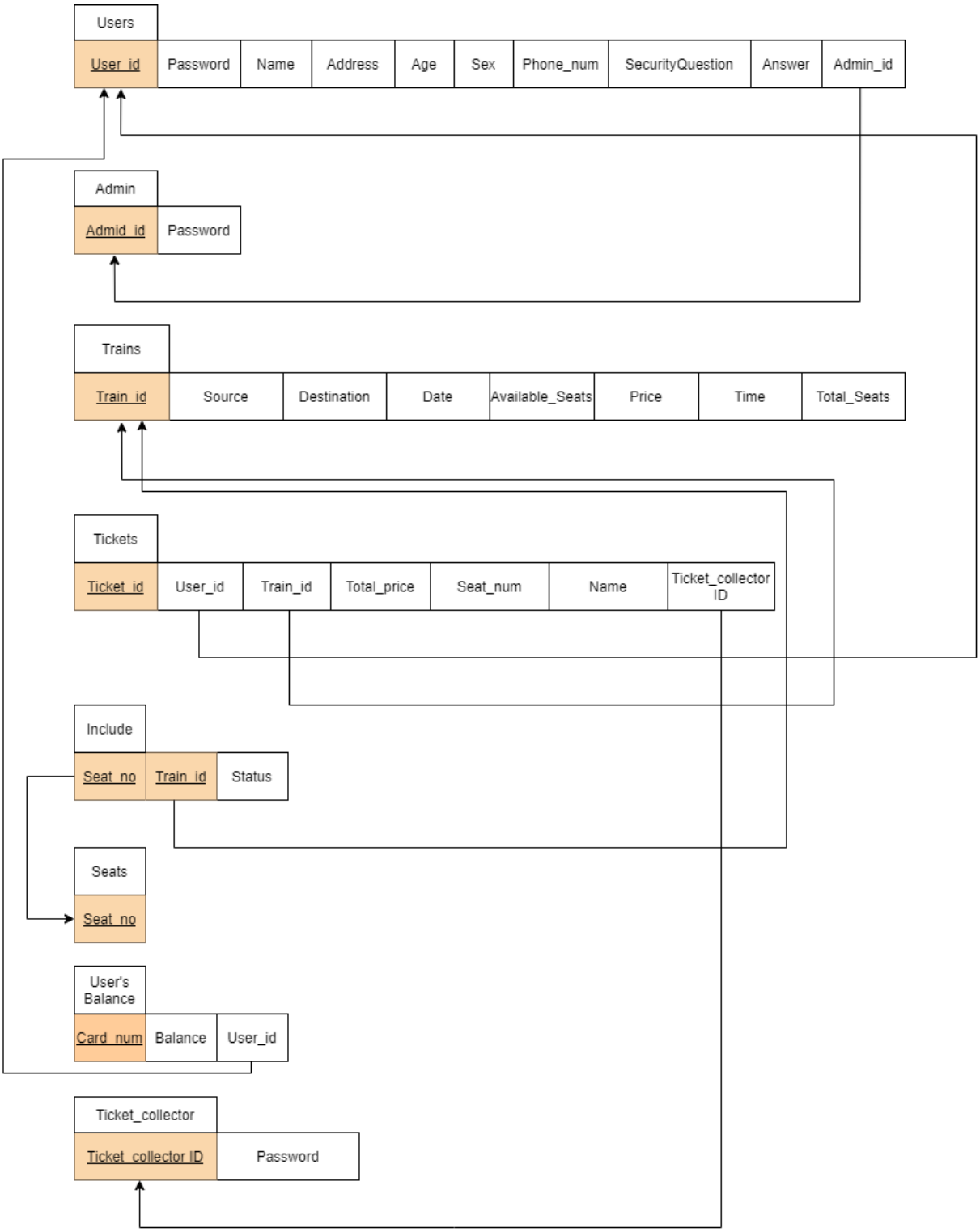


Figure 3.1. Relational Model

3.2. Explanation:

3.2.1. For the entity:

We have seven entities in total (Users, Admin, Train, Tickets, User's Balance, Ticket Collector, Seats). Thus, changing from ER diagram to relational model gives out seven schemas.

Each has the primary keys as given:

Users (User_id, Password, Name, Address, Age, Sex, Phone Number, Security Question, Answer)

Admin (Admin_id, Password)

Trains (Train_id, Source, Destination, Date, Available_Seats, Price, Time, Total_Seats)

Tickets (Ticket_id, Total_price, Seat_num, Name)

User's Balance (Card_num, Balance)

Ticket Collector (Ticket_collector ID, Password)

Seats (Seat_no)

3.2.2. For the relationship:

Add relationship (between Users and Admin): It is a 1-N relationship. Therefore, we will place the primary key of Admin (Admin_id) in the schema of Users as foreign key.

Book relationship (between Users and Ticket): It is a 1-N relationship. Therefore, we will only have a way to present, which is putting the primary key of Users, in detail, User_id in the schema of Ticket as foreign key.

Has relationship (between Users and User's Balance): It is a 1-1 relationship. So, we will place the primary key of Users (User_id) in the schema of User's Balance as foreign key or we can also do the opposite.

Provide relationship (between Train and Ticket): It is a 1-N relationship. Therefore, we will place the primary key of Train (Train_id) in the schema of Ticket as foreign key.

Take relationship (between Ticket Collector and Tickets): It is 1-N relationship. So, we will only have a way to present, which is placing the primary key of Ticket Collector (Ticket_Collector ID) in the schema of Tickets as foreign key.

Include relationship (between Train and Seats): It is a M-N relationship. Therefore, we put both the primary key of Train (Train_id) and Seats (Seat_no) to the new schema named Include as primary key.

Combining (1) and (2), the relation schema is:

Users (User_id, Password, Name, Address, Age, Sex, Phone Number, Security Question, Answer, Admin_id)

Admin (Admin_id, Password)

Train (Train_id, Source, Destination, Date, Available_Seats, Price, Time, Total_Seats)

Tickets (Ticket_id, Users_id, Train_id, Total_price, Seat_num, Name, Ticket_collector ID)

Seats (Seat_no)

User's Balance (Card_num, Balance, User_id)

Ticket Collector (Ticket_collector ID, Password)

Include (Seat_no, Train_id, Status)

Chapter 4. DATABASE STRUCTURE

4.1. Database Diagram

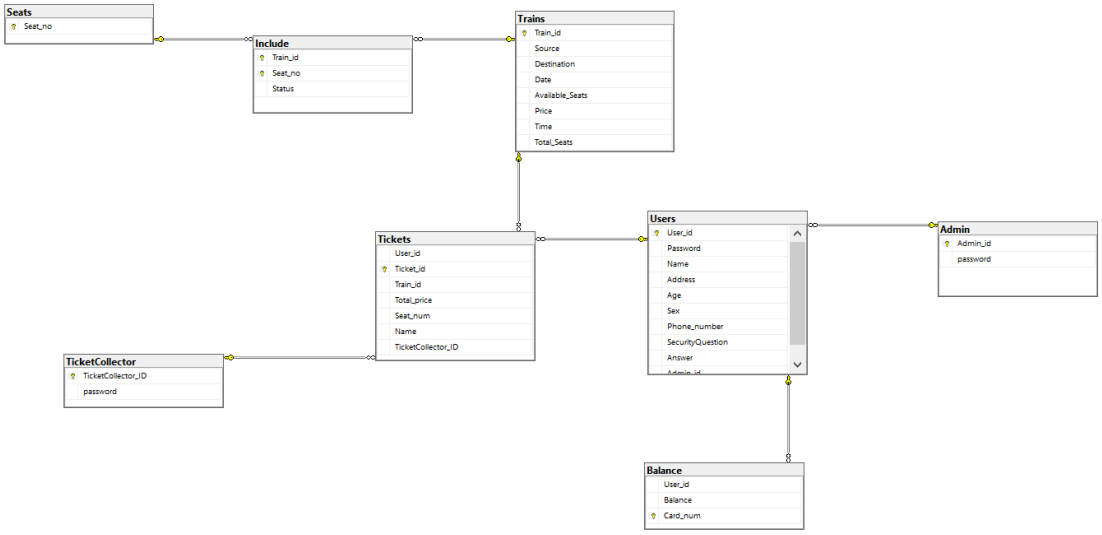


Figure IV.1 Database Diagram

4.2. Explanation

Users: It contains all personal information of User with distinguish by ID User. Each user has its own role in using different functions of the system.

Tickets: It contains information of User, train, total price and seat number. There are 3 foreign keys: ID User to get information of User who bought this ticket, ID tickets to get information of the Trip that was booked by User, and ID Ticketcollector.

Trains: It contains all information of Trip with distinguish by unique ID train and information of the train.

Admin: It saves all account of the system includes password and ID Admin Reference control, repair and update the system

Balance: It contains information of User money with a primary key is Card_num.

Seat: It identified by train ID seat number.

Ticket Collector: It saves all account of the system includes password and ID TicketsCollector to check the ticket.

Include: 2 primary keys are ID train and seat_no, also status of that seat.

CHAPTER 5: DATABASE CREATION:

5.1. Admin Table:

```
CREATE TABLE Admin (
Admin_id varchar(10) NOT NULL,
password varchar(50) NOT NULL,
PRIMARY KEY (Admin_id)
)
```

	Admin_id	password
▶*	NULL	NULL

5.2. Users Table

```
CREATE TABLE Users (
    User_id varchar(10) NOT NULL,
    Password varchar(50) NOT NULL,
```

```
    Name nvarchar(50) NOT NULL,
    Address nvarchar(50) NOT NULL,
    Age int NOT NULL,
    Sex nvarchar(10) NOT NULL,
    Phone_number varchar(20) NOT NULL,
    SecurityQuestion nvarchar(50) NOT NULL,
    Answer nvarchar(50) NOT NULL,
    Admin_id varchar(10),
    PRIMARY KEY (User_id),
    FOREIGN KEY(Admin_id) REFERENCES Admin
)
```

	User_id	Password	Name	Address	Age	Sex	Phone_n...	Security...	Answer	Admin_id
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.3. Trains Table

```
CREATE TABLE Trains(
    Train_id varchar(10) NOT NULL,
    Source nvarchar(50) NOT NULL,
    Destination nvarchar(50) NOT NULL,
    Date date NOT NULL,
    Available_Seats bigint NOT NULL,
    Price bigint NOT NULL,
    Time time(7) NOT NULL,
    Total_Seats bigint NOT NULL,
    PRIMARY KEY (Train_id)
)
```

	Train_id	Source	Destinati...	Date	Availabl...	Price	Time	Total_Se...
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.4. Tickets Table

```
CREATE TABLE Tickets (
    User_id varchar(10) NOT NULL,
    Ticket_id varchar(10) NOT NULL,
    Train_id varchar(10) NOT NULL,
    Total_price bigint NOT NULL,
    Seat_num int NOT NULL,
    Name varchar(30) NOT NULL,
    TicketCollector_ID varchar(10),
    PRIMARY KEY (Ticket_id),
    FOREIGN KEY(Train_id) REFERENCES Trains,
    FOREIGN KEY(User_id) REFERENCES Users
)
```

	User_id	Ticket_id	Train_id	Total_price	Seat_num	Name	TicketCo...
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.5. TicketCollector Table

```
CREATE TABLE TicketCollector(
    TicketCollector_ID varchar(10) NOT NULL,
    password varchar(50) NOT NULL,
    PRIMARY KEY (TicketCollector_ID)
)
```

	TicketCo...	password
▶*	NULL	NULL

5.6. Seats Table

```
CREATE TABLE Seats(
    Seat_no int NOT NULL,
```

```
        PRIMARY KEY (Seat_no)
)
```

	Seat_no
▶*	NULL

5.7. Balance Table

```
CREATE TABLE Balance(
        User_id varchar(10) NOT NULL,
        Balance bigint NOT NULL,
        Card_num nvarchar(50) NOT NULL,
        PRIMARY KEY (Card_num),
        FOREIGN KEY(User_id) REFERENCES Users
)
```

	User_id	Balance	Card_num
▶*	NULL	NULL	NULL

5.8. Include Table

```
CREATE TABLE Include (
        Train_id varchar(10) NOT NULL,
        Seat_no int NOT NULL,
        Status varchar(30) ,
        PRIMARY KEY(Train_id, Seat_no),
        FOREIGN KEY(Train_id) REFERENCES Trains,
        FOREIGN KEY(Seat_no) REFERENCES Seats
)
```

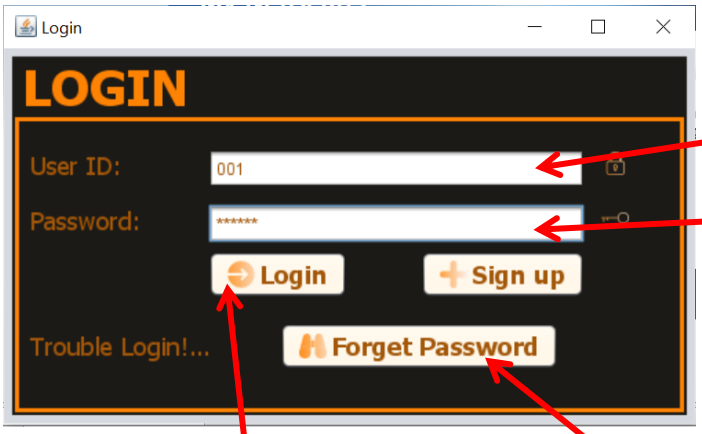
	Train_id	Seat_no	Status
▶*	NULL	NULL	NULL

Chapter 6. EXECUTION

The application is designed based on "*Android Local Train Ticketing Project.*" [1]

6.1. User

6.1.1. Login & Logout



Input User

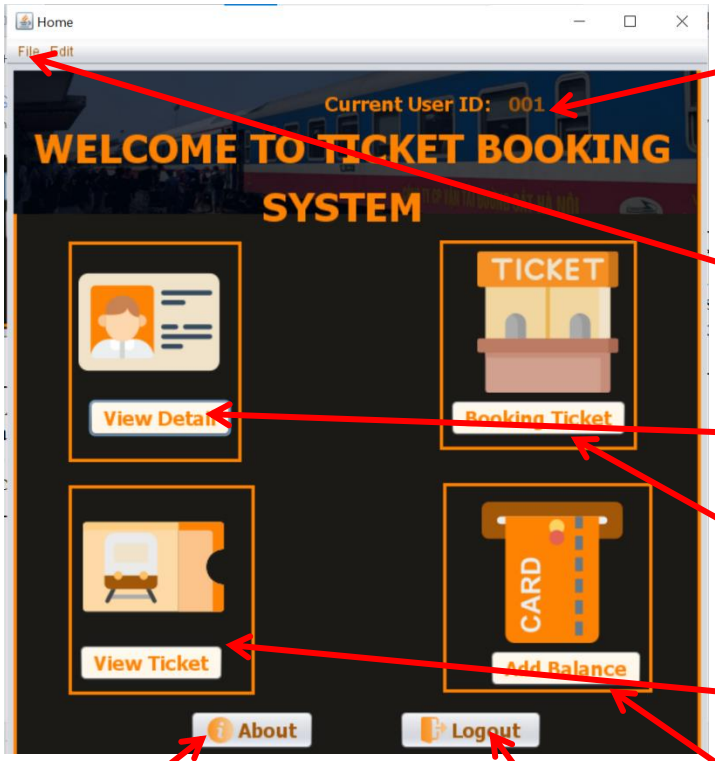
Input User's

Sign up Button: When the user login for the first time, they do not have their existed account on the system. So, they can push

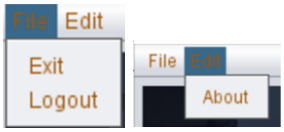
Login Button: After inputting the user's id and password, they push the Login button to access to the main frame called Home.

Forget Password Button: If the user forgot their password and they cannot login to the system, they could push this button to know their password.

6.1.2. Home



Show the current user's id so that we can know the user who are currently in the system.



File: includes the Exit button and Logout button.

View Detail Button: when the user clicks to this button, their details

Booking Ticket Button: when the user clicks to this button, they can book the train ticket.

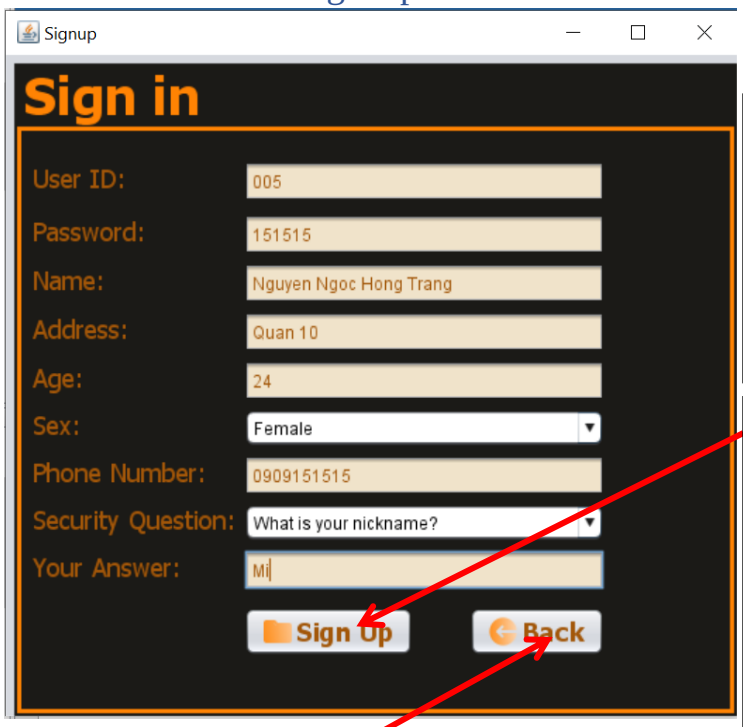
View Ticket Button: View the ticket's details that the user has

About Button: Show the system information.

Logout Button: Exit Home interface and turn back to the Login interface.

Add Balance Button: The user can add balance to pay for the ticket.

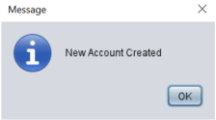
6.1.3. Sign Up



When Users do not have an account. They must sign up. The UI will appear and requires the users to input their information includes the username and password of the account.

Sign Up Button: When the users push this button, the system will check whether the User's id is exited in the system or not.

+ If it is not existed the following message will be shown to announce that the account has been created. Then, the system will insert user's information into database to store.

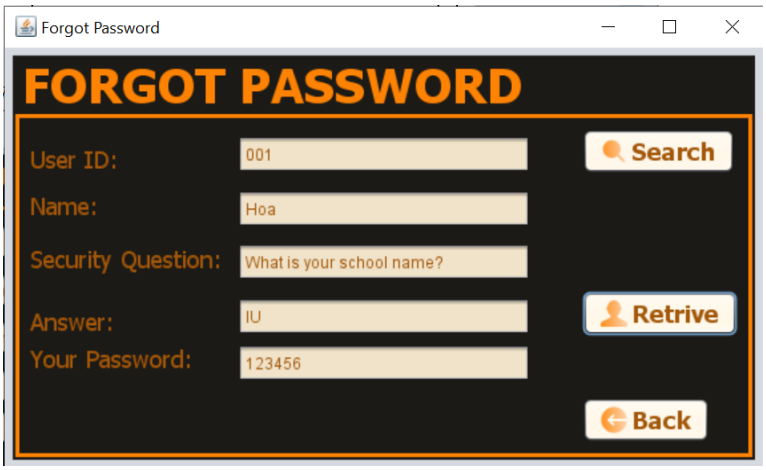


Back Button: Return to the Login interface when the user pushes it.

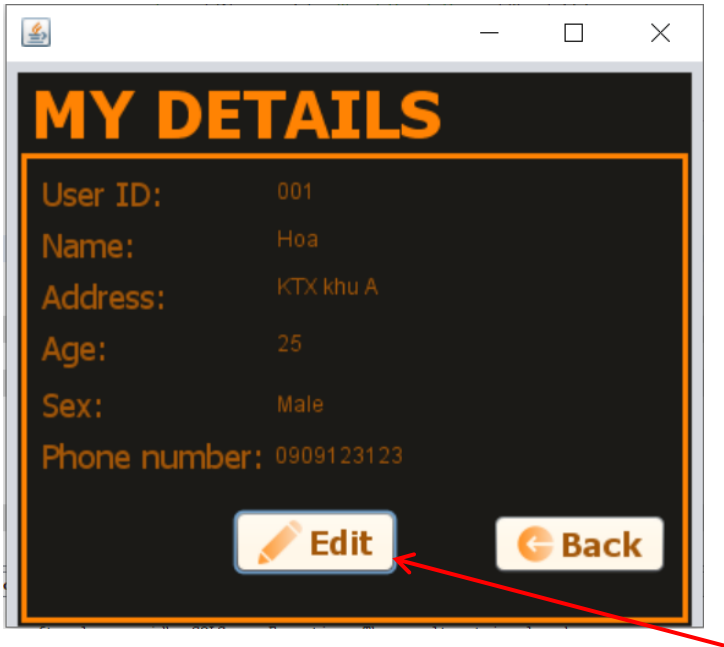
6.1.4. Forgot Password

Search Button: When the user forgot their password, they can input their id and push the Search button, then their name and security question will be shown.

Retrive Button: The user input their answer for the security question and push the Retrive button, then the user's password will be shown.

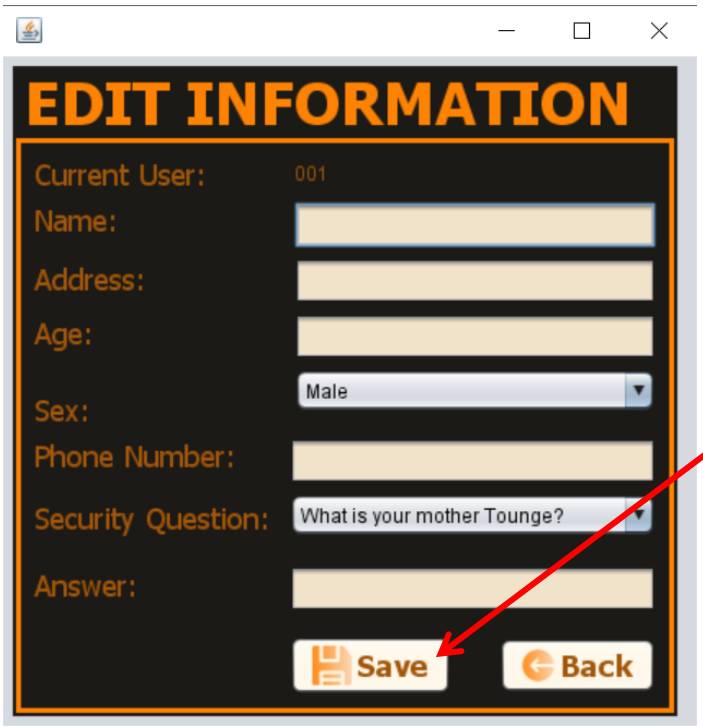


6.1.5. View User's Details



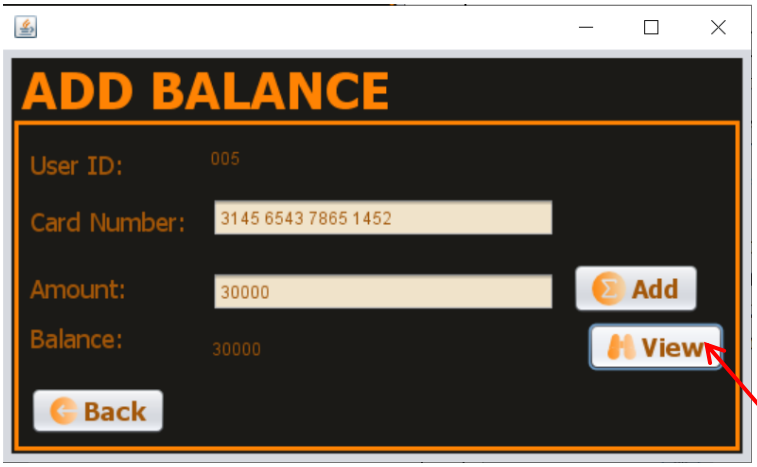
After click on the **View Details** Button in Home UI, the user's details including some basic information (ID, Name, Address, Age, Sex,

Edit Button: Allows users to modify their individual information. After clicking



Save Button: when clicking this button, the information will be updated after inputting the information that need to be

6.1.6. Add Balance



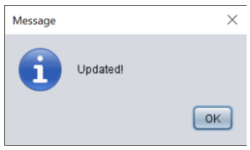
When clicking the **Add Balance** Button in Home UI, the user can create or updated their balance to pay for the ticket.

View Button:

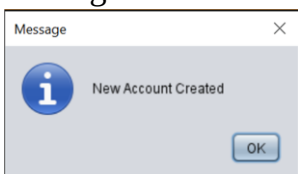
This button will allow the user to view their current balance.

Add Button: After inputting the card number and the amount, the users push this button. Then, the system will check if the account has existed in the database or not.

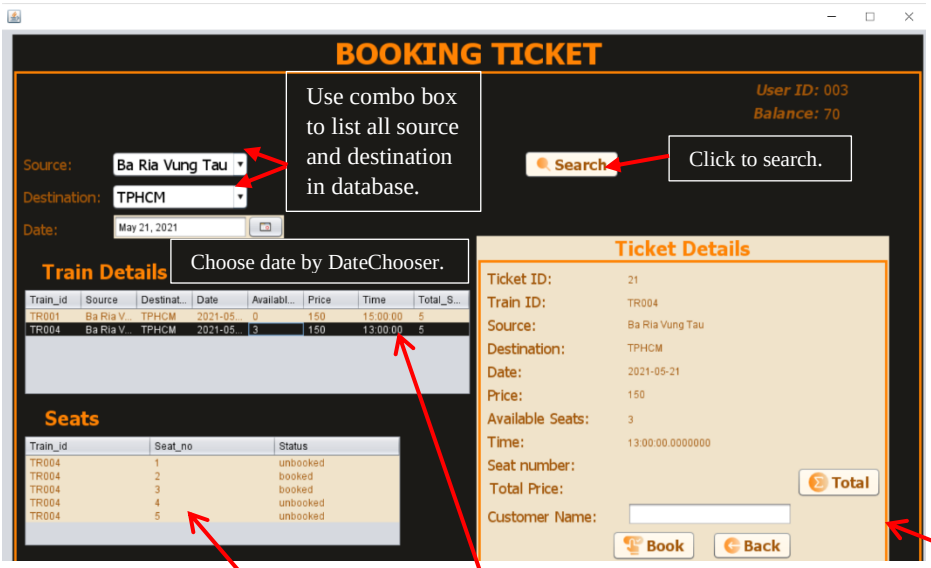
+ If the account has been existed, the card number and amount will be updated for the user's account. The message below will also be shown.



+ If the account has not been existed in the system (it means this is the new users), the system will create the new account on the system. The message will be shown.



6.1.7. Booking Ticket



When clicking the **Booking Ticket** Button in Home UI, the user can select the source, destination, date to find the Train.

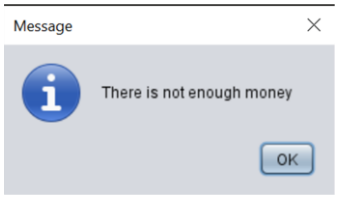
The system will list all Train that is satisfied with source, destination, date.

After choosing the train, the list of seat information will appear. You can only choose the rows with Status = "unbooked".

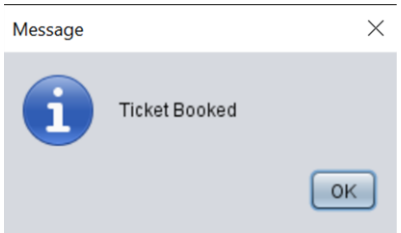
The list of train's information. Click on row to choose the train.

The information of the train and seat will transfer to the Ticket details. Click Book Button to confirm your ticket.

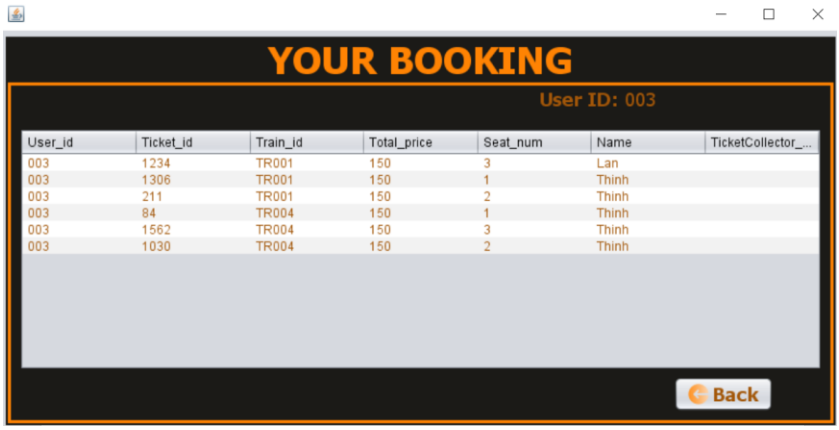
+ If Balance < Price, the system will send you the message below and your ticket will not be booked.



+ If not, the ticket will be booked, the Balance will be updated, the Available seats will be minus by 1.



6.1.8. View ticket:

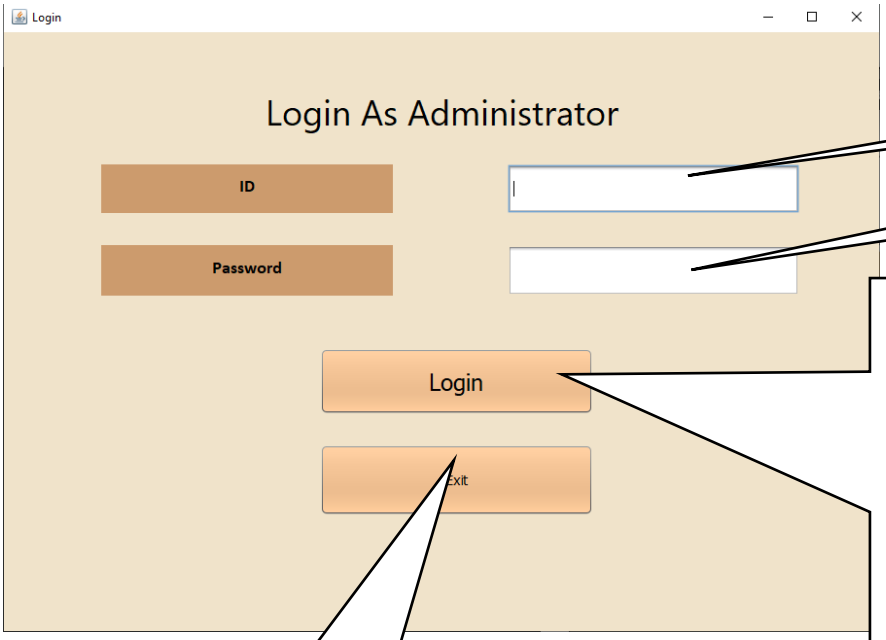


When clicking the **View Ticket** Button in Home UI, the user can view all tickets that have been booked ordered by Train ID.

6.2. Admin:

6.2.1. Login Frame:

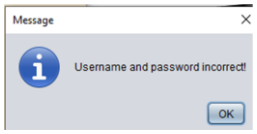
When open the application, the Login Frame will appear.



Input Admin ID

Input Admin password

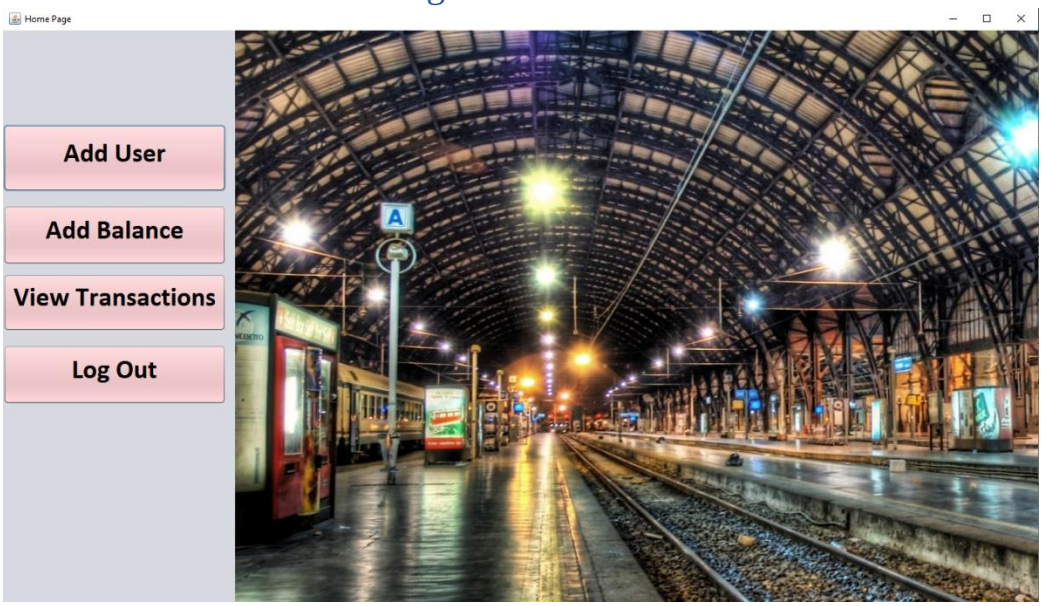
Login Button: After input ID and password, the Administrator push this button to access **Home Page**. System will check this account valid or invalid. If valid, Admin can access Home Page, if not, an error message



will appear.

Exit Button: when push this button, the program will close.

6.2.2. Home Page:



There are 4 buttons in this Frame:

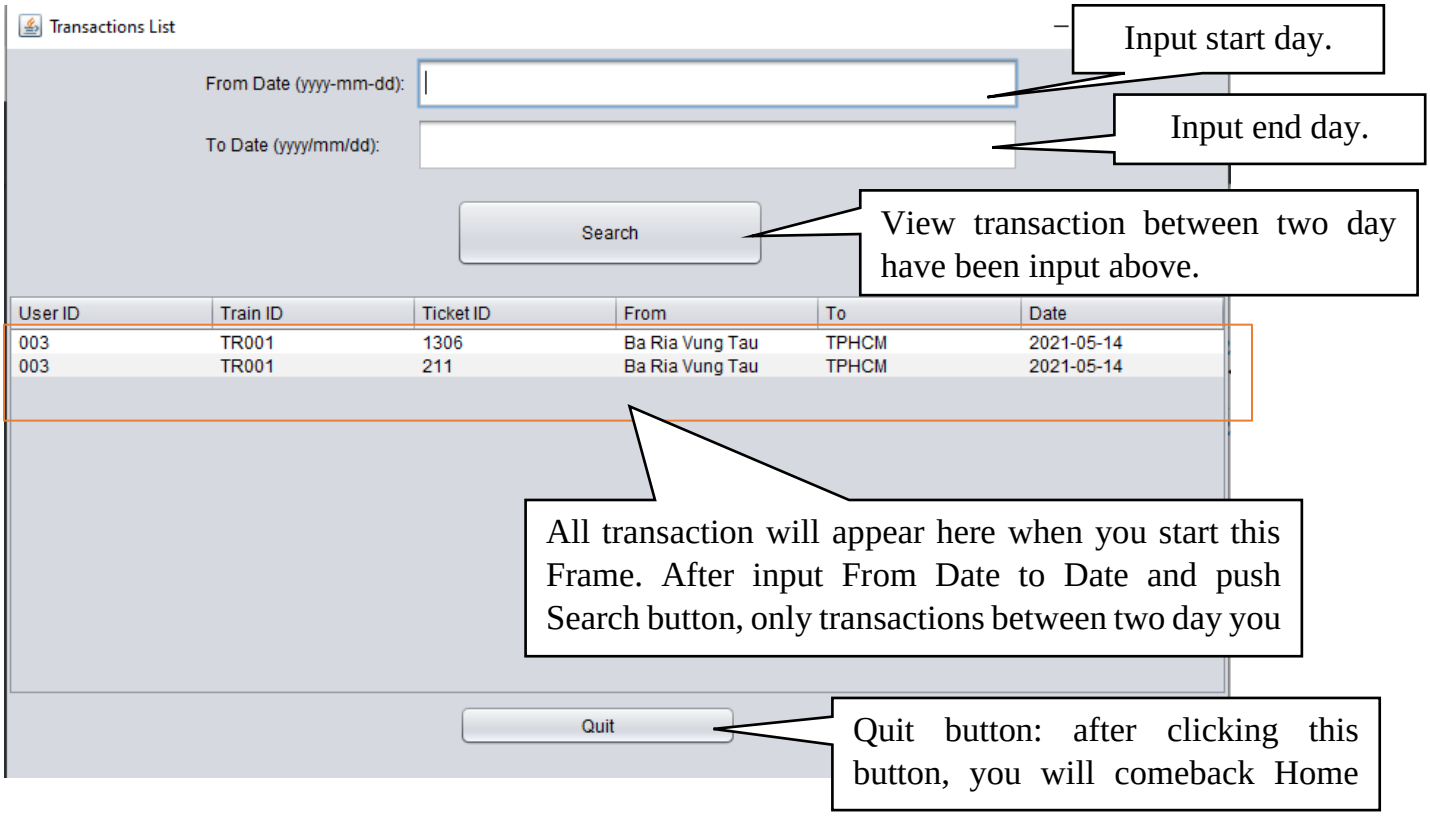
Add User button: Admin can add a user to system.

Add Balance button: Admin can change user's balance.

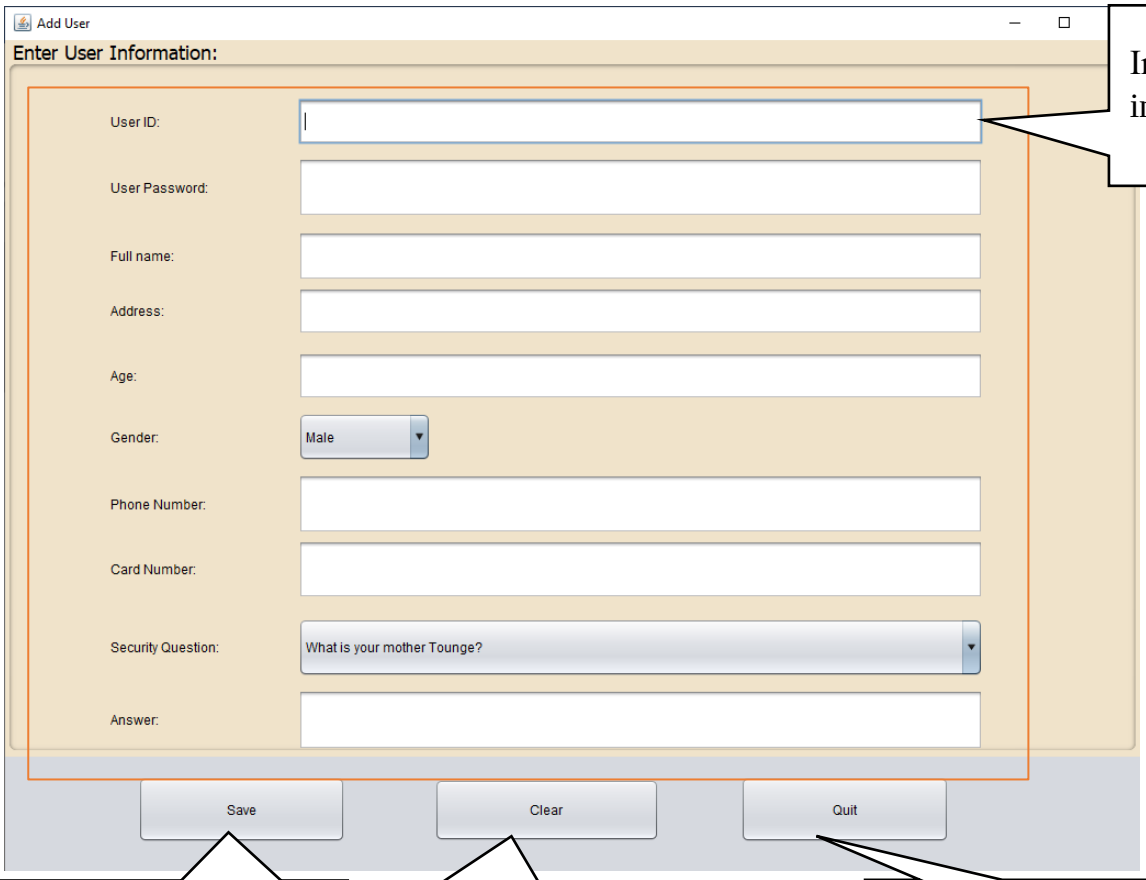
View Transaction: See what transactions have been done.

Log Out: Return to the login Frame.

6.2.3. View Transaction:



6.2.4. Add User:



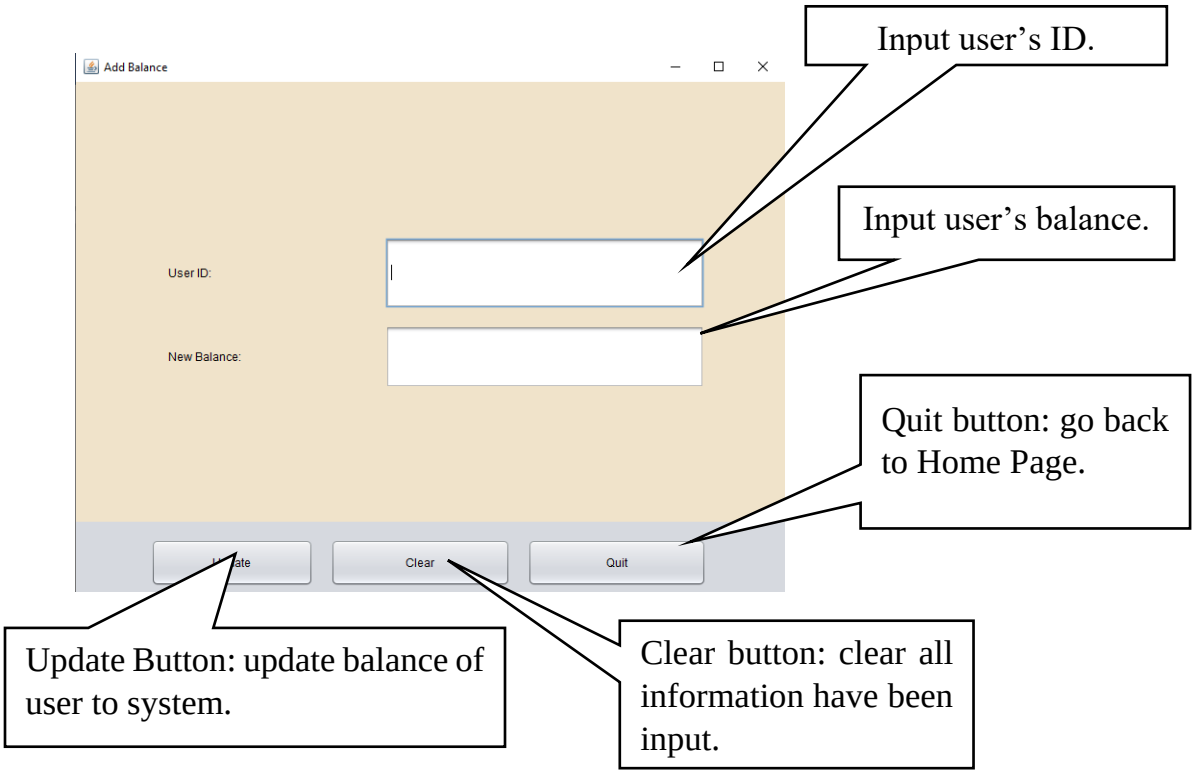
Input user's information.

Save button: after input user's information, admin push this button to add that user to system.

Clear button: clear all information have been input.

Quit button: go back to Home Page.

6.2.5. Add Balance:



Chapter 7. QUERY COMMAND

1. Find all Ticket ID that the customer's name is similar with the name of person booking ticket

SQL Statement:

```
SELECT Ticket_id
FROM Tickets T
WHERE Name IN (SELECT Name
        FROM Users U
        WHERE T.User_id=U.User_id)
```

Relational algebra:

$\pi_{\text{Ticket\_id}}$ (Ticket $\bowtie_{\text{Ticket.User\_id=Users.User\_id AND Name.Tickets=Name.Users}}$ Users)

2. Find the User_id who booked Train TR001 and not booked Train TR002

SQL Statement:

```
SELECT DISTINCT User_id
FROM Tickets T
WHERE Train_id='TR001'
EXCEPT
SELECT DISTINCT User_id
FROM Tickets T
WHERE Train_id='TR002'
```

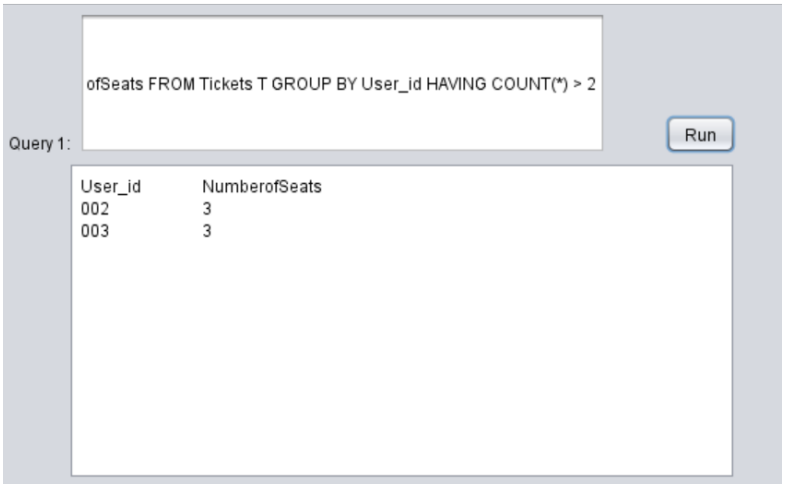
Relational algebra:

$\pi_{User\_id}(\sigma_{Train\_id='TR001'}$ Tickets) - $\pi_{User\_id}(\sigma_{Train\_id='TR001'}$ Tickets)

3. Find the User_ID who booked more than 2 seats. Sort the result based on number of seats booked.

SQL Statement:

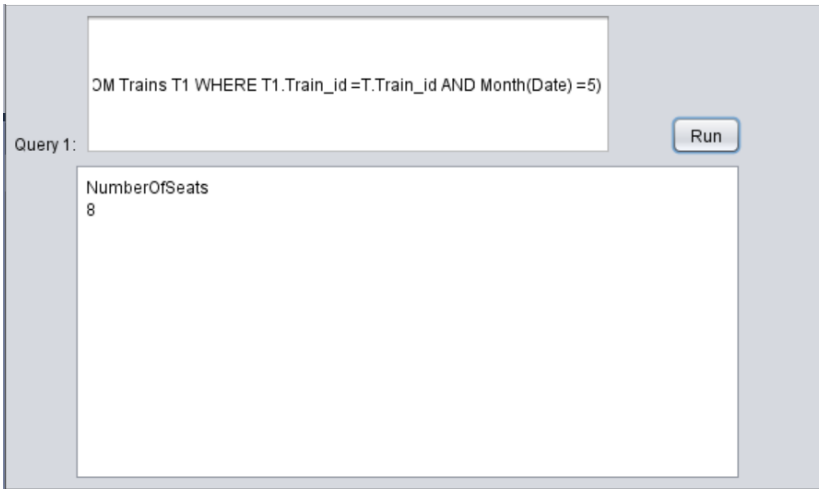
```
SELECT User_id, COUNT (*) AS NumberofSeats
FROM Tickets T
GROUP BY User_id
HAVING COUNT (*) > 2
ORDER BY NumberofSeats
```



4. Find total seats sold in May:

SQL Statement:

```
SELECT COUNT (*) AS NumberOfSeats
FROM Tickets T
WHERE EXISTS (SELECT *
        FROM Trains T1
        WHERE T1.Train_id =T.Train_id AND MONTH (Date) =5)
```



5. Find the user's name whose balance is the most

SQL Statement:

```
SELECT Name
FROM Users
WHERE User_id IN (SELECT User_id
                  FROM Balance
                  WHERE Balance = (SELECT MAX(Balance)
                                   FROM Balance))
```

6. Find the name of users who book train having more than 2 available seats

SQL Statement:

```
select DISTINCT U.Name
from Users U, Tickets
where U.User_id = Tickets.User_id
AND Tickets.Ticket_id IN (select T.Ticket_id
                          from Tickets T, Trains S
                          where T.Train_id= S.Train_id AND S.Available_Seats > 2)
```

	Name
1	Hoa
2	Linh

7. Find the user id and number of tickets booked by each user. Give an alias name as no_of_tickets. Sort the result based on number of tickets booked.

SQL Statement:

```
select User_id, count (Ticket_id) as no_of_tickets
from Tickets
group by User_id
order by no_of_tickets
```

	User_id	no_of_tickets
1	001	2
2	002	3
3	003	3

8. Find the information of users who booked more than 2 tickets

SQL Statement:

```
Select *
From Users U
Where 2 < (Select count (*) From Tickets T
           Where U.User_id = T.User_id )
```

	User_id	Password	Name	Address	Age	Sex	Phone_number	SecurityQuestion	Answer	Admin_id
1	002	123123	Linh	KTX khu B	20	Female	0909121212	What is your school name?	IU	NULL
2	003	123456	Thinh	Ba Ria Vung Tau	23	Male	0908135311	What is your nickname?	HT	NULL

9. Find all passengers booked ticket by someone else

SQL Statement:

```
Select Distinct T.Name As passengers
from Tickets T, Users U
where T.Name not in ( Select Distinct T.Name
                      from Tickets T, Users U
                      Where T.User_id = U.User_id and T.Name = U.Name )
```

	Name
1	Hinh
2	Lien
3	long

10. Find name and address of users who book train having source in Ba Ria Vung Tau

SQL Statement:

```
select DISTINCT U.Name, U.Address
from Users U, Tickets T, Trains S
where T.User_id=U.User_id AND T.Train_id=S.Train_id
                AND S.Train_id IN (Select Train_id
                                   from Trains
                                   where Trains.Source = 'Ba Ria Vung Tau')
```

	Name	Address
1	Thinh	Ba Ria Vung Tau
2	Hoa	KTX khu A
3	Linh	KTX khu B

11. Find the users that booked for someone else

SQL Statement:

```
Select Distinct U.Name As booker
```

```
from Tickets T, Users U
where T.Name not in ( Select Distinct T.Name
                      from Tickets T, Users U
                      Where T.User_id = U.User_id and T.Name = U.Name )
```

	booker
1	Hoa
2	Linh
3	Thinh

12. Find all users in the database who have more balance than each user

SQL Statement:

```
SELECT U.name
FROM Users U, Balance B
WHERE U.User_id = B.User_id and B.Balance  > all (SELECT avg (Balance)
                                                  FROM Balance)
```

	name
1	Linh

13. Find the user's name who has the least balance

SQL Statement:

```
SELECT Name
FROM Users
WHERE User_id IN (SELECT User_id
            FROM Balance
            WHERE Balance = (SELECT MIN (Balance)
                    FROM Balance))
```

	Name
1	Thinh

Chapter 8. CONTRIBUTION

WORKING		CODING	CHECK
Doing Proposal		**All members**	**DONE**
Create Databases	Users, Trains, Balance	**Giao**	**Thinh**
	Tickets, Seats, Include	**Thinh**	**Giao**
	Admin	**Dang**	**Phuc**
	Check linked in whole database	**Anh**	**All members**
	Optimize database		
Meeting, discussion and decided about general background of the application. How many functions in this, included functions of user and manager		**All members**	
Find the information and document about the similar system		**Anh**	**All members**
Querying the database using Java Database Connectivity	Design the main display of **Login, Signup. Forget Password of Users, View Details, Add Balance**	**Giao**	**Thinh**
	Design the main display of **Booking Ticket, View Ticket**	**Thinh**	**Giao**
	Function of **Login, Signup. Forget Password of Users, View Details, Add Balance**	**Giao**	**Thinh (Check and test)**
	Function of **Booking Ticket, View Ticket**	**Thinh**	**Giao (Check and Test)**

	Design the main display of **Log in, Home Page, Add User, Add Balance, View Transaction** as Administrator	**Phuc**	**Dang**
	Function of **Log in, Home Page, Add User, Add Balance, View Transaction** as Administrator	**Dang**	**Phuc**
Design and complete the ERD diagram due to requirement		**Thinh**	**All members**
Design, complete and explain the Relational Model		**Giao**	**All members**
Design, complete and explain the Database Diagram		**Phuc**	**All members**
Meeting, discussion and decided about the structure, quality of the application, improve the code (if needed). Developed the application with more new features if have new ideas.		**All members**	
Find 5 query commands each person and find out the solution for each.		**All members**	
Summary to write the report		**Thinh**	**All members**
Design the slides		**Anh**	**All members**

Chapter 9. CONCLUSION

Ticket Booking project is combined with many important skills for the student to improve their knowledge, coding skill, thinking logically, develop problems, and solve them.

While making the project, we made our progress by solving problems. This provided us experiences that will be useful in the future. We came to know that how we can use Java to make an app, create a logical database that is suitable for the project and link it with the programming language. Doing project is known as one of the best ways to learn more algorithms and optimize them.

Some important things that we learned include designing a good program architecture and analyst customer requirements, converting real-life situations into efficient code.

Therefore, besides having a deeper knowledge into Database structure, this project also helps us improve programming ability.

REFERENCES

[1] Nevon Projects, "Android Local Train Ticketing Project",2015 [Online].

Link access: https://nevonprojects.com/android-local-train-ticketing-project/?fbclid=IwAR0vA63Gu9-4aMUAQH1367yAoWQ5XnbHCOu2wSXPqWeZ25_rUv6WQXaO5r8