Implementation of Airline Management System

100520807

Content

1 Introduction

After learning how to realize a database, in this paper, I will use the database designed by MySQL in Part 1 to implement a corresponding front-end system. This airline management system enables the end user to interact with the database. In this system, administer can use it to realize some basic management functions like manage the staff, flight and airplane. Then, a new booking for passengers can be created, and they the bookings also can be updated and deleted. Finally, the list of the passenger, airplane and flight schedule will also be generated. At the same time, some concepts such as database privileges, recovery, etc. will be considered on how to adopt in the implementation of the system to protect the security of personal data information.

In this paper, firstly make a summarize of feedback on assignment 1, then some changes are made for system E-R diagram and tables. Next, this system operation environment will be introduced. Then, the detail design and implementation of the system requirements will be described. Finally, some actions will be discussed about data security.

2. Summarize of Feedback for Assignment 1

From the feedback, what have learnt can be summarized as follows:

1. The introduction should introduce the whole study background and topic of the paper, and the conclusion section should describe what functions have been done, what has learnt, and the critical evaluation of the project. These two parts should be more detailed.
2. The figure name should describe the meaning of the figure more clearly and in detail.
3. Pay attention to writing grammar.
4. Some attributes in tables need to be updated.

At the same time, for the part 2, I will make some changes such as, modify the E-R diagram, change a few attributes in the original tables and add some new tables.

Firstly, a new system E-R diagram is created as follows:

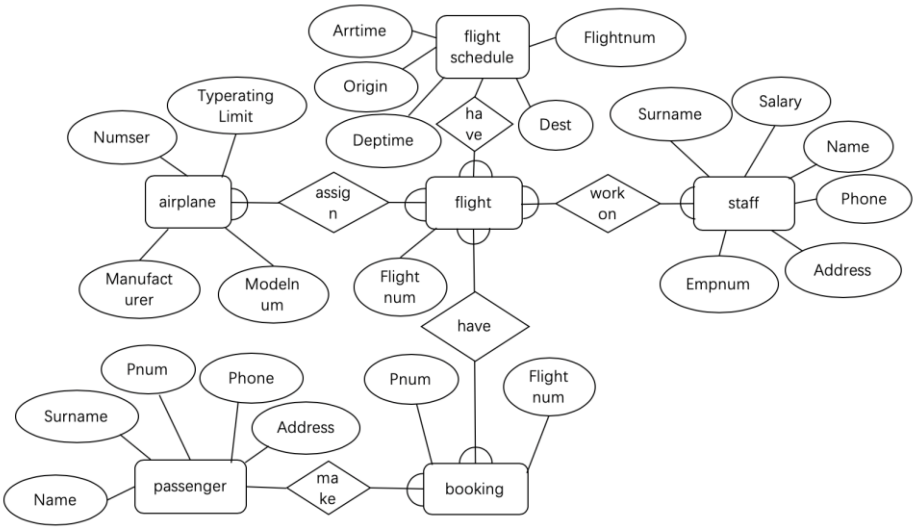
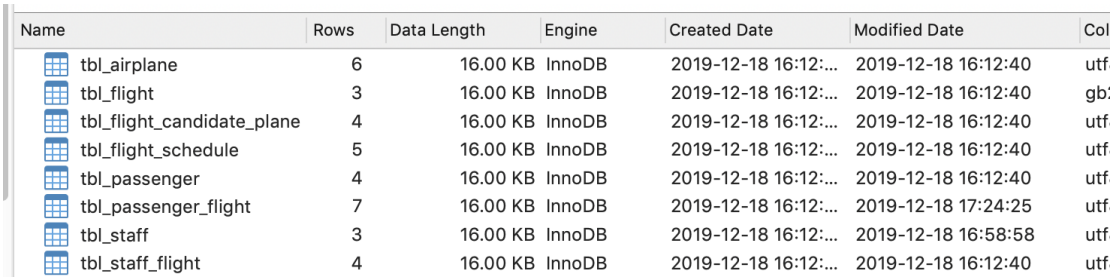


Fig 1 System E-R diagram

Then, some attributes in the tables have been changed, and some new tables have created instead of the original tables. The new database table schema looks like as follows:



Name	Rows	Data Length	Engine	Created Date	Modified Date	Col
tbl_airplane	6	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	utf
tbl_flight	3	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	gb
tbl_flight_candidate_plane	4	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	utf
tbl_flight_schedule	5	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	utf
tbl_passenger	4	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	utf
tbl_passenger_flight	7	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 17:24:25	utf
tbl_staff	3	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:58:58	utf
tbl_staff_flight	4	16.00 KB	InnoDB	2019-12-18 16:12:...	2019-12-18 16:12:40	utf

Fig 2 Database table schema

3 System Operation Environment

Operating System: macOS 10.14.6
CPU: 2.3 GHz Intel Core i5
RAM: 8GB
Resolution: 2560 x 1600
Eclipse: This article uses eclipse as the tool for project implementation, and selects tomcat 7 as the server running the project.
MySQL 5.1.21: MySQL is used as database to store and handle data of the airline system. NoSQL is a non-relational database that can be used for the storage of large-scale data, and is more suitable for storing unstructured data. MySQL database is fully capable in the system designed in this article.
Navicat Premium 12.1: This is a database management tool. I use it to connect to MySQL to manage the database more conveniently.

4 System Design & Implementation

4.1 Database Connection

In this paper, I use JDBC to connect the system and database. JDBC is a java database connection and is a standard method for Java applications to connect to a database. Because the system is implemented in Java, the program can access the database through SQL statements by using JDBC. The steps for using JDBC are as follows: First, load the driver: Class.forName (driver). Then, transfer the connect() method of driver with getconnection (), returning an object that implements the Connection interface to connect to the MySQL database. Finally, use the statement created by the connection to execute the SQL statement.
Database connection test--add function:

In the flight add page, we enter some data and click the submit. Then we can found that, there appears a new flight with the above information in the flight list. Therefore, this proves that the database is connected to the system.

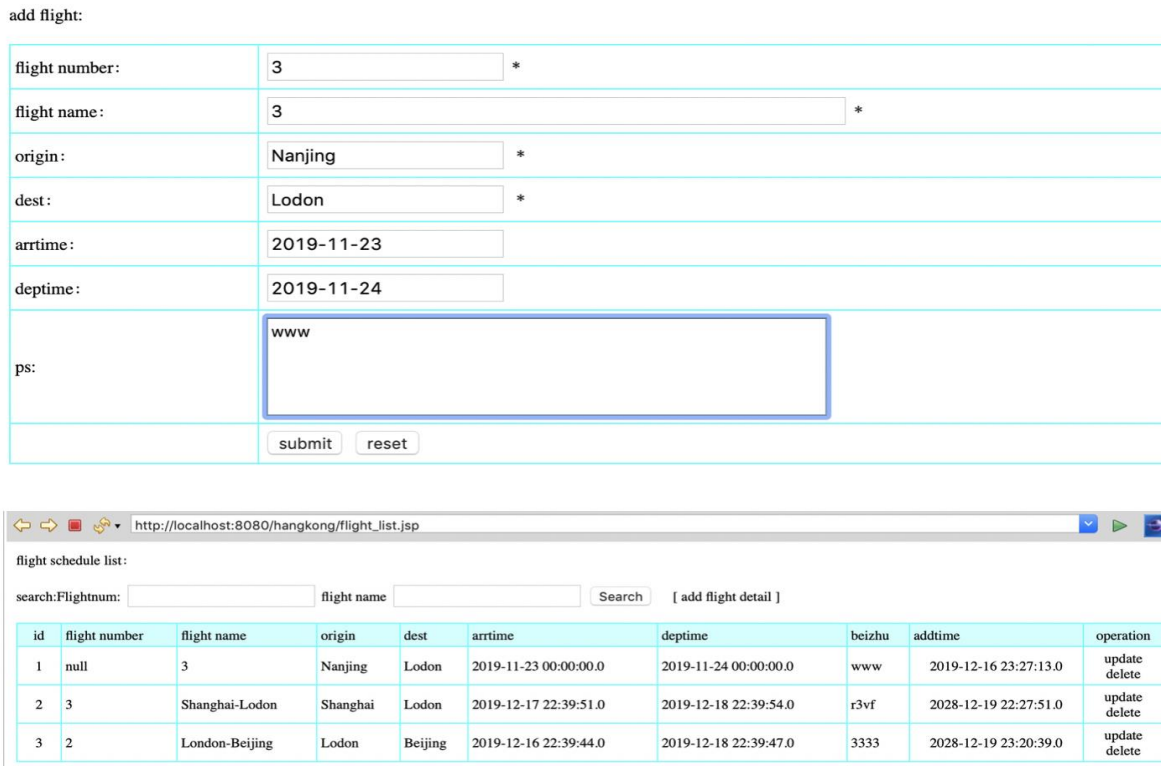


Fig 3 Database connection test--add function test

Database connection test--search function:

In the flight list page, if we enter some data about the flight that we want to find, then click the search. And the corresponding results can be shown in this page. Therefore, this proves that the database is connected to the system.

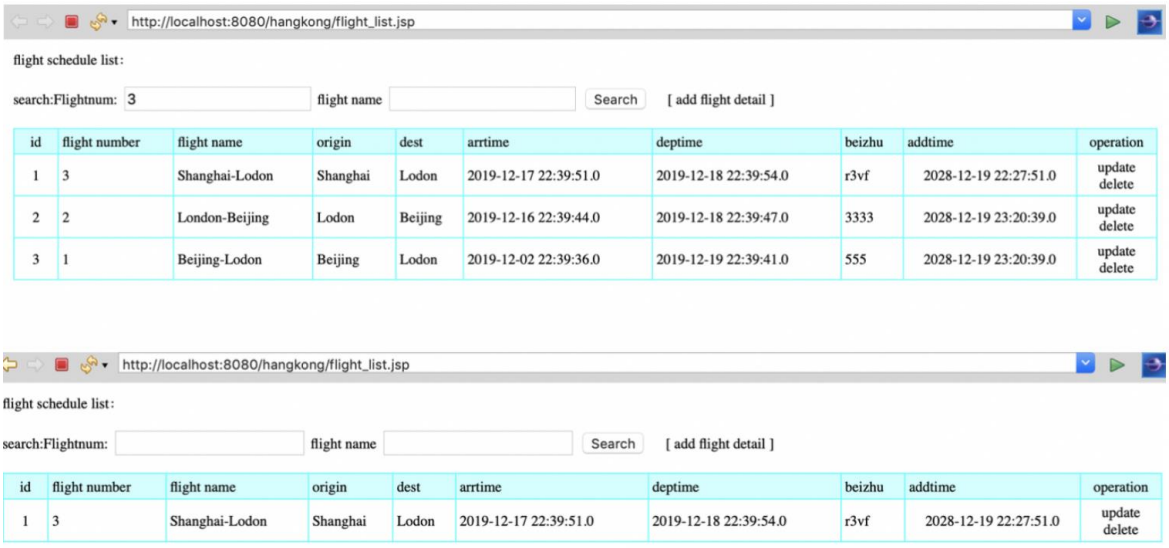


Fig 4 Database connection test—search function test

Database connection test--update function:

In the flight update page, after we enter some new data and submitting them. When we back to the flight list page, it can be found that that flight information has be changed successfully. Therefore, this proves that the database is connected to the system.

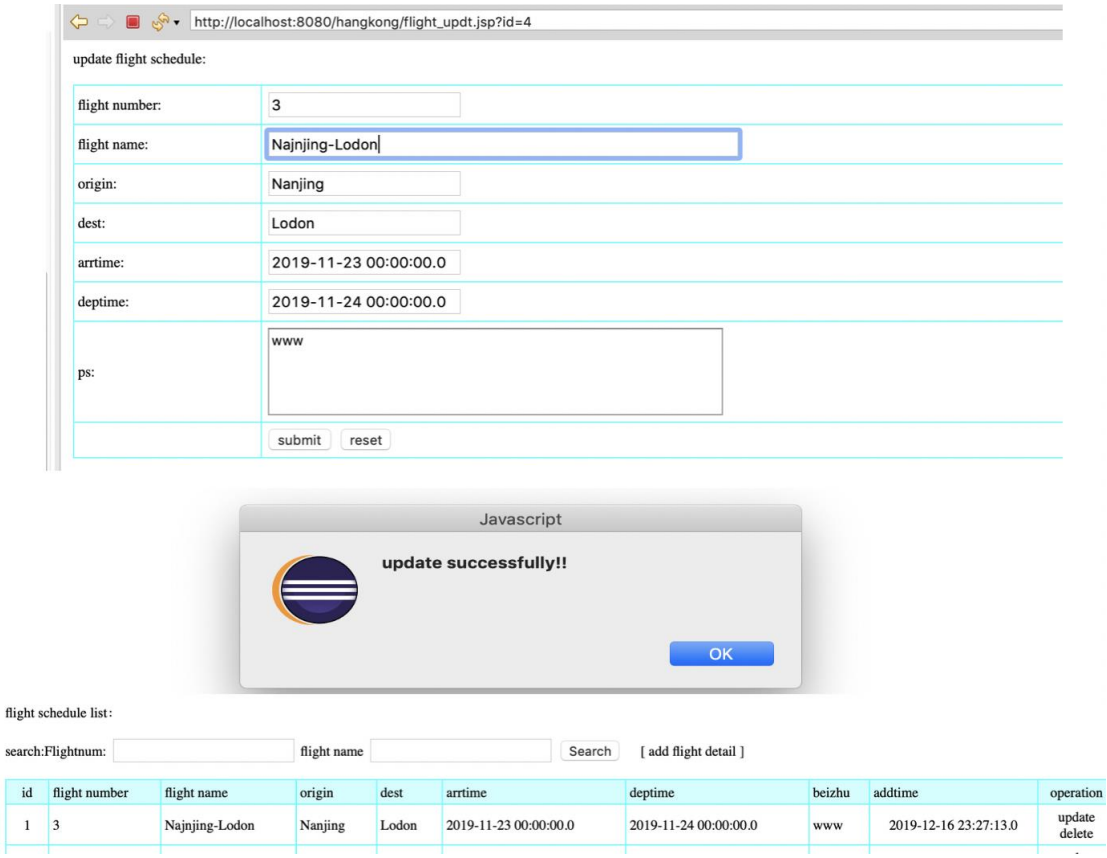


Fig 5 Database connection test—update function test

Database connection test--delete function:

In the flight management page, after we choose a flight and delete it. When we back to the flight list page, the information about this flight is no longer there. Therefore, this proves that the database is connected to the system.

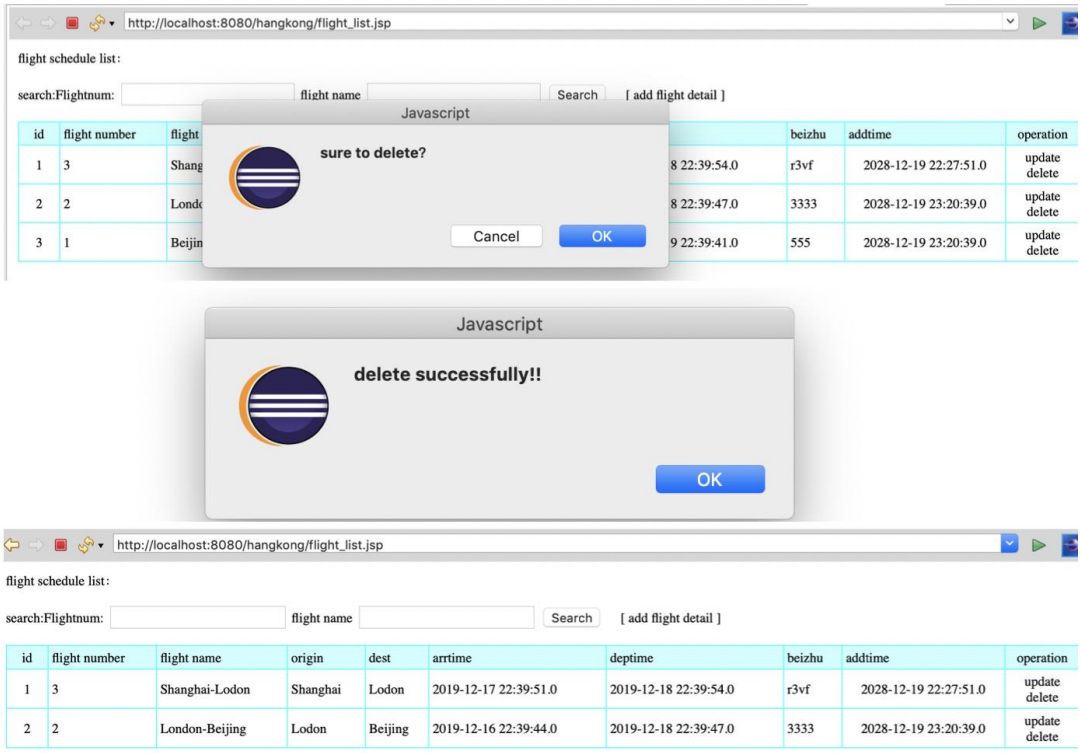


Fig 6 Database connection test—delete function test

4.2 System Requirements Analysis

There are three different user roles in this system, including staff, passenger, and administrator. For each role, they have different permissions and can use different system functions.

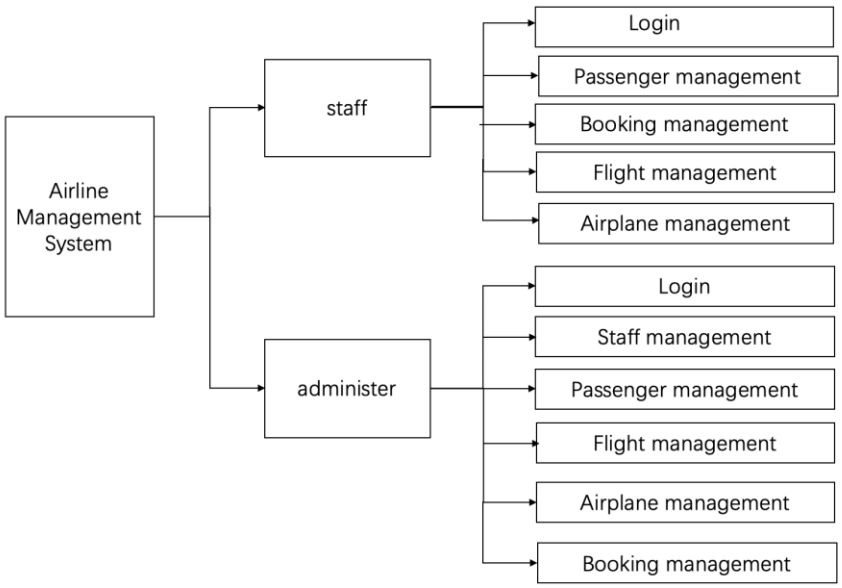


Fig 7 System function block diagram

4.3 System Implementation

1. Login

All users can use the login function. The user enters the user name and password correctly, selects their corresponding role, and then can successfully log in to enter the system.

```
if(cx.equals("staff"))
{
   sql="select * from tbl_flight_candidate_plane where gonghao='"+uid+"' and mima='"+pwd+"' ";
}
```

Fig 8 SQL statement in JSP file for login function

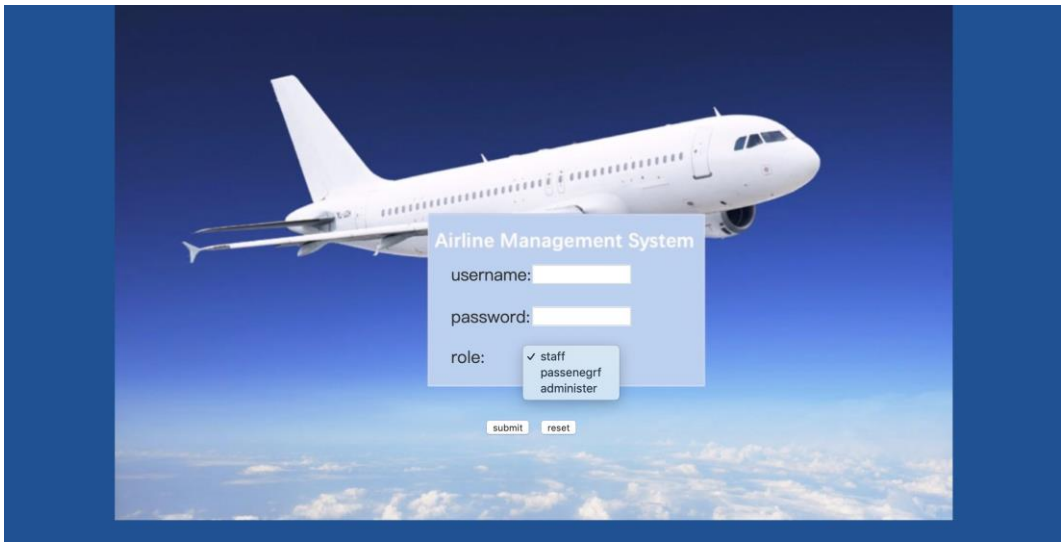


Fig 9 User login interface

After successfully logging in to the system, the user will enter the homepage of the system as follows. But for users with different permissions, the functions that they can use are also different. In the top of the system home page, it shows current user and date. When we click on the Home, we will go back to the system home page. When we choose User, we can enter the user management interface. And when we choose Exit, we will exit the system and return to the login page.

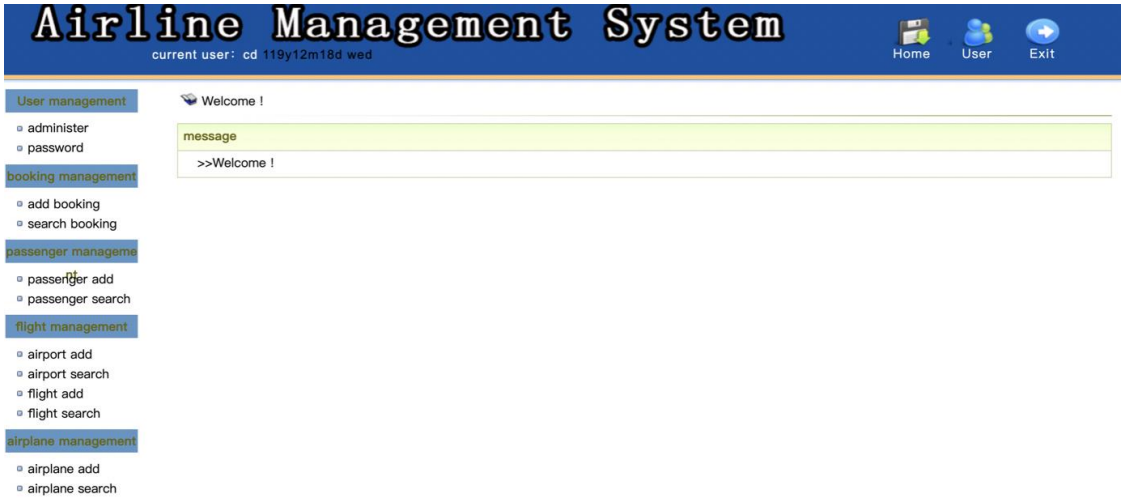


Fig 10 System home page interface

2. User Management

This function is belong to staff and administer. In this module, there are two parts: administer management and password modification.

1) Administer management

When administer click on the administer, they can enter a administer management page. In this page, we also can see the personal information of other staff and administer.

Add administer: Administer can add a new administer through entering new username and password, and submit them.

```
String username=request.getParameter("username");
 String pwd=request.getParameter("pwd1");
 ResultSet RS_result=connDbBean.executeQuery("select * from tbl_staff where username='"+username+"'");

 if(RS_result.next())
{
out.print("<script>alert('This username already exists, please change !');window.history.go(-1);</script>");
    }
else{


        String sql="insert into tbl_staff(username,pwd) values('"+username+"','"+pwd+"')";

        connDbBean.executeUpdate(sql);
        out.print("<script>alert('add sucessfully!!');location.href='yhzhgl.jsp';</script>");

    }
```

Fig 11 SQL statement in JSP file for administer management function

Delete staff and administer: For administer and staff, they also have the right to delete existing staff and administer user.

```
String id=request.getParameter("id");
String tablename=request.getParameter("tablename");

String sql="delete from "+tablename+" where id="+id+"";

connDbBean.executeUpdate(sql);
```

Fig 12 SQL statement in JSP file for delete function

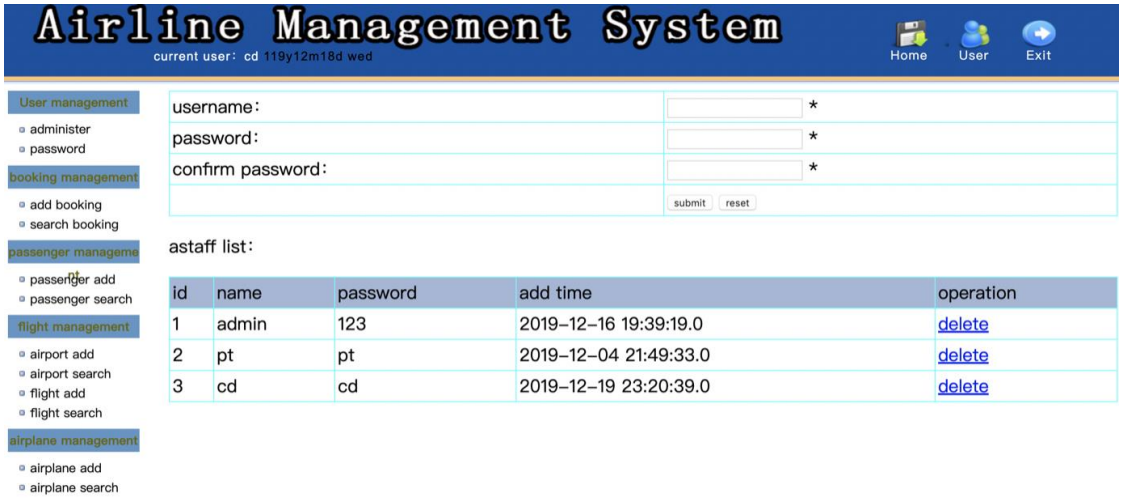


Fig 13 Administer management interface

2) Modify password

After selecting password, administer and staff can enter the page as follows. If administer and staff want to change their password, they need to enter the old password firstly, then enter new password and confirm password. If something wrong with the information they entered, they can select the reset to set a new password again. After submitting these information, administer can modify their password successfully.

```
if(pwd.equals(ymm))
{
  String sql="update tbl_staff set pwd='"+xmm1+"' where username='"+request.getSession().getAttribute("username")+"'";
  connDbBean.executeUpdate(sql);
  out.print("<script>alert('update successfully!!');window.history.go(-1);</script>");
}
else
{
  out.print("<script>alert('soorry,please try again!');window.history.go(-1);</script>");
```

Fig 14 SQL statement in JSP file for password modification function

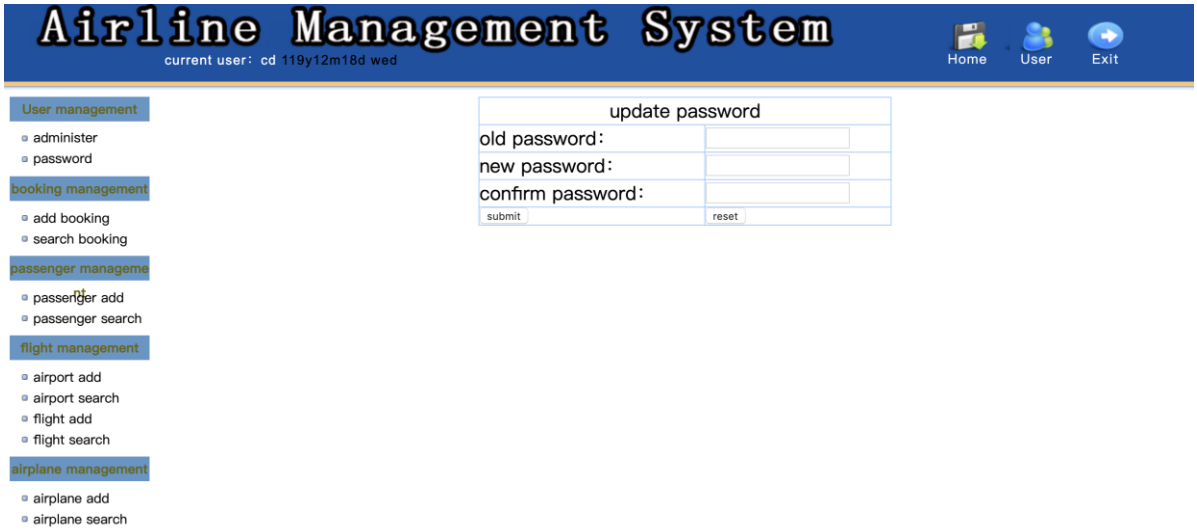


Fig 15 Administer password modification interface

3. Booking Management

This function is belong to staff and administer and they can manage the information of booking.

1) Add booking:

Staff and administer can add a new booking for the passengers. They should enter passenger number and passenger name, select flight, airplane and airport, enter note and price, then submit them to make a new booking for the passenger successfully.

```
String Airport=request.getParameter("Airport");
String PNum=request.getParameter("PNum");String Price=request.getParameter("Price");String PName=request.getParameter("PName");String I

   String sql="insert into tbl_passenger_flight(PNum,PName,Flight,Airplane,Airport,beizhu,Price) values('"+PNum+"','"+PName+"','"+
 Flight+"','"+Airplane+"','"+Airport+"','"+beizhu+"','"+Price+"') ";

       connDbBean.executeUpdate(sql);
```

Fig 16 SQL statement in JSP file for booking add function

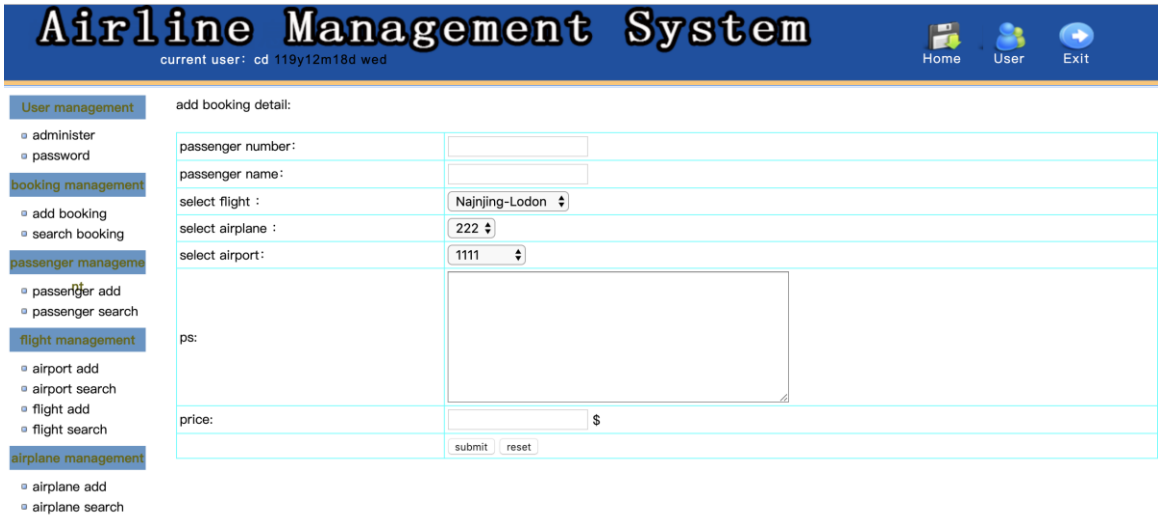


Fig 17 Add new booking interface

2) Search booking:

Search booking: After entering the number or name of the passenger that we want to query, then click search. The booking details of corresponding passenger will be shown.

```
sql="select * from tbl_passenger_flight where 1=1";
if(request.getParameter("Pnum")=="" ||request.getParameter("Pnum")==null )
{}
else
{
  sql=sql+" and Pnum like '%"+new String(request.getParameter("Pnum").getBytes("8859_1"))+"%'";
}
if(request.getParameter("Pname")=="" ||request.getParameter("Pname")==null )
{}
else
{
  sql=sql+" and Pname like '%"+new String(request.getParameter("Pname").getBytes("8859_1"))+"%'";
}
sql=sql+" order by id desc";
ResultSet RS_result=connDbBean.executeQuery(sql);
```

Fig 18 SQL statement in JSP file for booking search function

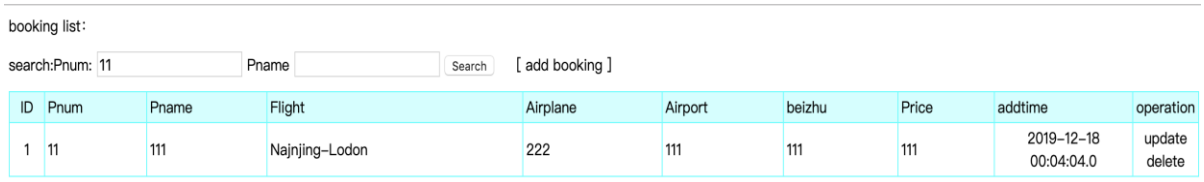


Fig 19 Search booking interface—search and delete

Delete booking: If passenger want to cancel their booking, we can choose delete to help them cancel the booking.

Update airport: Click the update, it will take us to the selected booking details page, where we can update its information.

```
  String Price=request.getParameter("Price");  String Airport=request.getParameter("Airport");
String PNum=request.getParameter("PNumo");String PName=request.getParameter("PName");
String Flight=request.getParameter("Flight");String Airplane=request.getParameter("Airplane");
String beizhu=request.getParameter("beizhu");
String id=request.getParameter("id");
String sql="update tbl_passenger_flight set PNum='"+PNum+"',PName='"+PName+"',Flight='"+Flight+"',Airplane='"
+Airplane+"',beizhu='"+beizhu+"',Price='"+Price+"',Airport='"+Airport+"'  where id= "+id;
connDbBean.executeUpdate(sql);
out.print("<script>alert('update successfully!!');location.href='booking_list.jsp';</script>");
```

Fig 20 SQL statement in JSP file for booking update function

update booking detail:

passenger number:	11
passenger name:	111
select flight:	
select airplane:	222
select airport:	1111
ps:	111
price:	111 $
	submit reset

Fig 21 Booking update interface

4. Passenger Management

This function is belong to staff and administer and they can manage the information of passenger.

1) Add passenger:

Staff and administer can add a new passenger. After selecting passenger add, they need to enter passenger number, passenger name, sex, birthday, phone, email and address. And then select submit to add a new passenger successfully.

```
String gonghao=request.getParameter("gonghao");String xingming=request.getParameter("xingming");String xingbie=request.getParameter("xingbie");
String chushengnianyue=request.getParameter("chushengnianyue");String dianhua=request.getParameter("dianhua");
String youxiang=request.getParameter("youxiang");String jiguan=request.getParameter("jiguan");String zuzhijigou=request.getParameter("zuzhijigou");
String mima=request.getParameter("mima");String zhicheng=request.getParameter("zhicheng");String beizhu=request.getParameter("beizhu");
ResultSet RS_result=connDbBean.executeQuery("select * from [tbl_staff_flight] where gonghao='"+gonghao+"'");
       String sql="insert into tbl_passenger(gonghao,xingming,xingbie,chushengnianyue,dianhua,youxiang,jiguan,zuzhijigou,zhicheng,beizhu) values('"+gonghao+"','"
       +xingming+"','"+xingbie+"','"+chushengnianyue+"','"+dianhua+"','"+youxiang+"','"+jiguan+"','"+zuzhijigou+"','"+zhicheng+"','"+beizhu+"') ";

       connDbBean.executeUpdate(sql);
```

Fig 22 SQL statement in JSP file for passenger add function

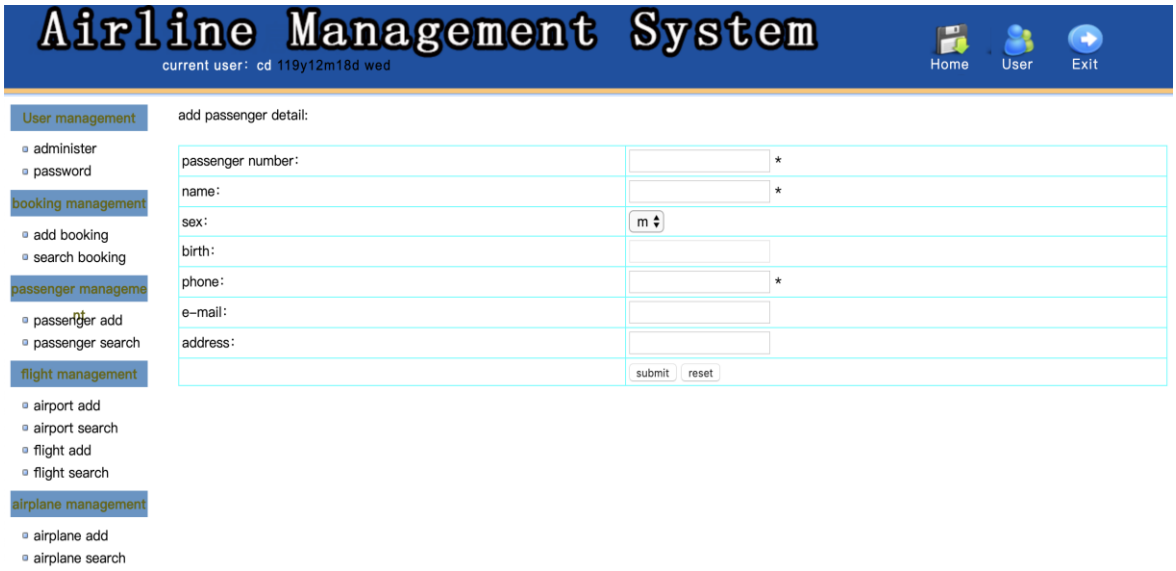


Fig 23 Add new passenger interface

2) Search passenger:

Search passenger: After entering the number or name of the passenger that we want to query, then click search, and the details of corresponding passenger will be shown.

```
 sql="select * from tbl_passenger where 1=1";
 if(request.getParameter("num")=="" ||request.getParameter("num")==null )
 {}
 else
 {
   sql=sql+" and num like '%"+new String(request.getParameter("num").getBytes("8859_1"))+"%'";
 }
 if(request.getParameter("name")=="" ||request.getParameter("name")==null )
 {}
 else
 {
   sql=sql+" and name like '%"+new String(request.getParameter("name").getBytes("8859_1"))+"%'";
 }
 sql=sql+" order by id desc";
ResultSet RS_result=connDbBean.executeQuery(sql);
```

Fig 24 SQL statement in JSP file for passenger search function

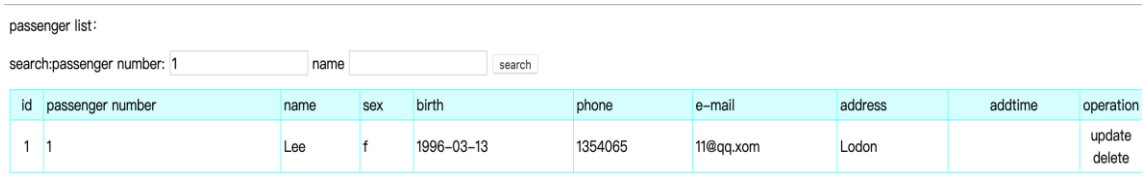


Fig 25 Search passenger interface—search and delete

Update passenger: Click the update, it will take us to the selected passenger details page, where we can make a modification of their information.

```
String num=request.getParameter("num");String name=request.getParameter("name");String sex=request.getParameter("sex");
String birth=request.getParameter("birth");String phone=request.getParameter("phone");String mail=request.getParameter("mail=");
String address=request.getParameter("address");String mima=request.getParameter("mima");String id=request.getParameter("id");

String sql="update tbl_passenger set num='"+num+"',name='"+name+"',sex='"+sex+"',birth='"+birth+"',phone='"+phone+"',mail='"
+mail+"',address='"+address+"' where id= "+id;
connDbBean.executeUpdate(sql);
out.print("<script>alert('update successfully!!');location.href='kh_list.jsp';</script>");
```

Fig 26 SQL statement in JSP file for passenger update function

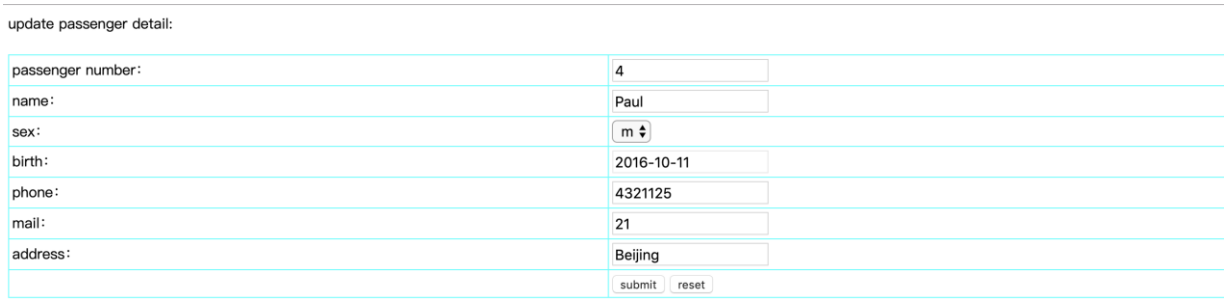


Fig 27 Update passenger interface

5. Flight Management

This function is belong to administer and they can manage the information of flight and airport.

1) Add airport:

When we choose airport add, we need to enter new airport name, and choose submit, then a new airport can be added successfully.

```
String Airport=request.getParameter("Airport");
String leibiemingcheng=request.getParameter("leibiemingcheng");

        String sql="insert into tbl_airplane(leibiemingcheng) values('"+leibiemingcheng+"') ";

        connDbBean.executeUpdate(sql);
        out.print("<script>alert('Add successfully!!');location.href='booking_add.jsp';</script>");
```

Fig 28 SQL statement in JSP file for airport add function

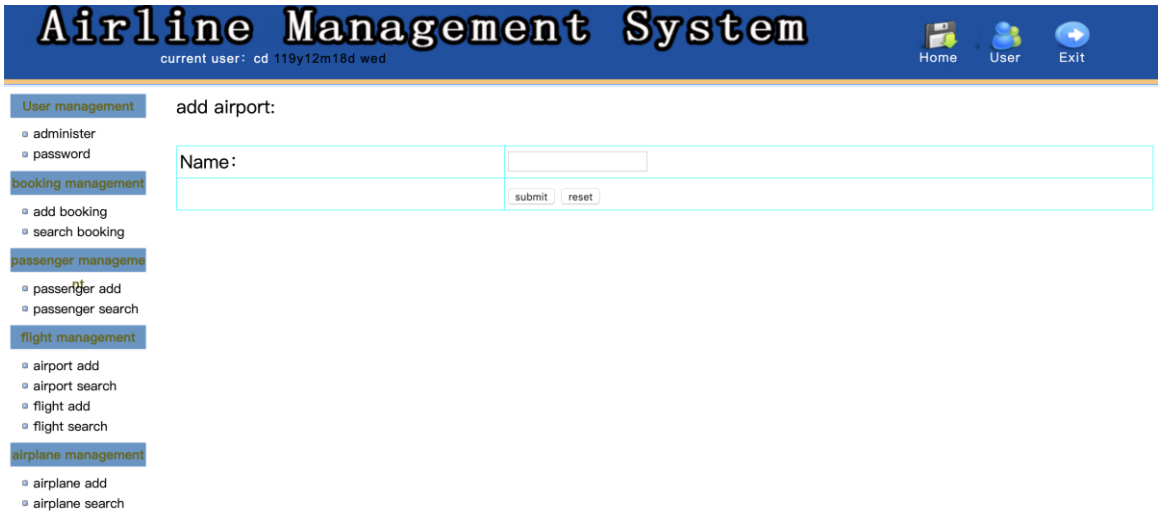


Fig 29 Add new airport interface

1) Search airport:

Search airport: After entering the name of the airport that you want to query, then click search. The results of corresponding airport information will be displayed below.

```
String sql="";
sql="select * from tbl_airplane where 1=1";
if(request.getParameter("bianhao")=="" ||request.getParameter("bianhao")==null )
{}
else
{
  sql=sql+" and leibiemingcheng like '%"+new String(request.getParameter("bianhao").getBytes("8859_1"))+"%'";
}
sql=sql+" order by id desc";
```

Fig 30 SQL statement in JSP file for airport search function

Delete airport: If some airports are no longer used, we can choose delete to delete these airports.

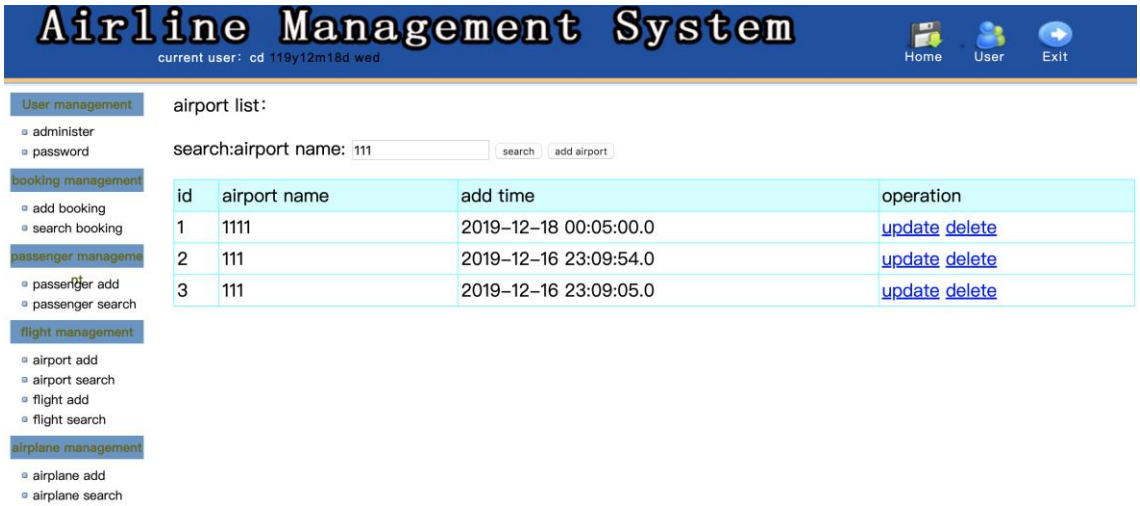


Fig 31 Airport management interface—search and delete

Update airport: Click the update, it will take us to the selected airport information page, where we can update its name.

```
String leibiemingcheng=request.getParameter("leibiemingcheng");
String id=request.getParameter("id");
String sql="update tbl_airplane set leibiemingcheng='"+leibiemingcheng+"' where id= "+id;
connDbBean.executeUpdate(sql);
```

Fig 32 SQL statement in JSP file for airport update function

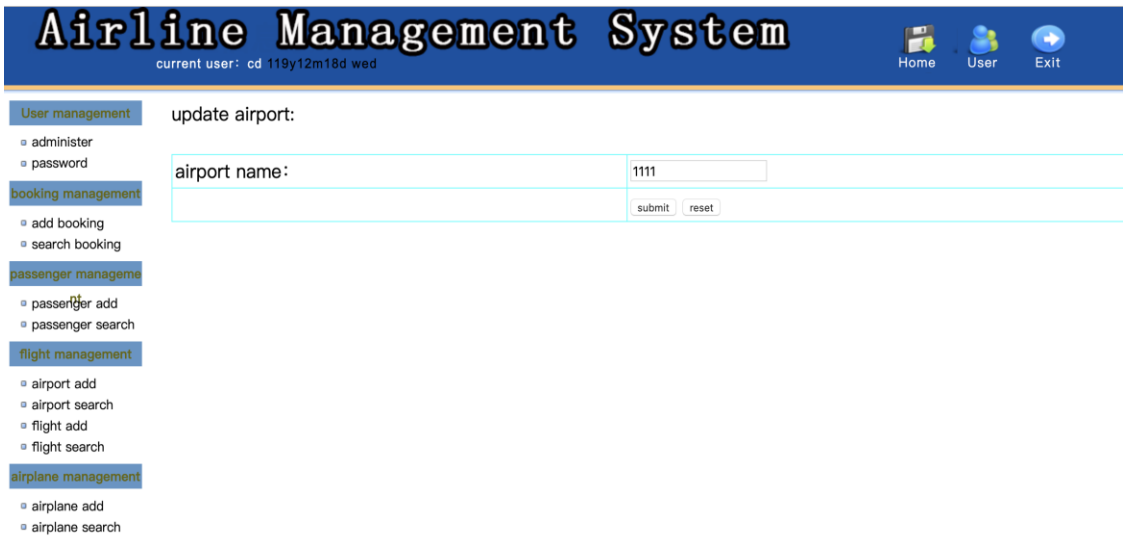


Fig 33 Airport update interface

2) Add flight:

When we choose flight add, we can add a new flight schedule. First, we should enter flight number, flight name, origin, destination, arrive time, departure time and note, then choose submit. Finally, a new flight schedule can be added successfully.

```
String Flightnum=request.getParameter("Flightnum");String zuzhimingcheng=request.getParameter("zuzhimingcheng");
String origin=request.getParameter("origin");String dest=request.getParameter("dest");
String arrtime=request.getParameter("arrtime");String deptime=request.getParameter("deptime");String beizhu=request.getParameter("beizhu");


        String sql="insert into tbl_flight(Flightnum,zuzhimingcheng,origin,dest,arrtime,deptime,beizhu) values('"+Flightnum+"','"
        +zuzhimingcheng+"','"+origin+"','"+dest+"','"+arrtime+"','"+deptime+"','"+beizhu+"') ";

        connDbBean.executeUpdate(sql);
```

Fig 34 SQL statement in JSP file for flight add function

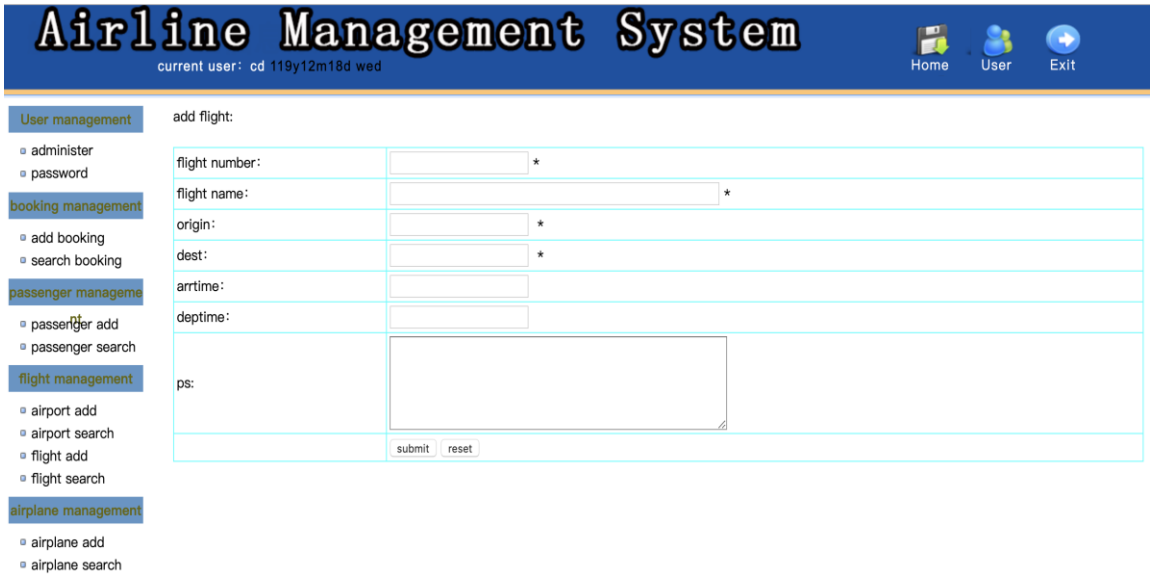


Fig 35 Add new flight interface

3) Search flight:

Search flight: After entering the flight number or flight name that we want to query, then click search. The results of corresponding flight schedule will be displayed below.

```
String sql="";
sql="select * from tbl_flight where 1=1";
if(request.getParameter("Flightnum")=="" ||request.getParameter("Flightnum")==null )
{}
else
{
  sql=sql+" and Flightnum like '%"+new String(request.getParameter("Flightnum").getBytes("8859_1"))+"%'";
}
if(request.getParameter("mingcheng")=="" ||request.getParameter("mingcheng")==null )
{}
else
{
  sql=sql+" and zuzhimingcheng like '%"+new String(request.getParameter("mingcheng").getBytes("8859_1"))+"%'";
}
sql=sql+" order by id desc";
ResultSet RS_result=connDbBean.executeQuery(sql);
```

Fig 36 SQL statement in JSP file for flight search function

Delete flight: We also can choose delete to delete some existing flights.

Fig 37 Flight management interface—search and delete

Update airport: After clicking the update, we will enter the selected flight schedule page. We can make a modification about the information of the corresponding flight.

```
String Flightnum=request.getParameter("Flightnum");String zuzhimingcheng=request.getParameter("zuzhimingcheng");

String origin=request.getParameter("origin");String dest=request.getParameter("dest");
String arrtime=request.getParameter("arrtime");String deptime=request.getParameter("deptime");

String beizhu=request.getParameter("beizhu");
String id=request.getParameter("id");
String sql="update tbl_flight set Flightnum='"+Flightnum+"',zuzhimingcheng='"+zuzhimingcheng+"',origin='"
+origin+"',dest='"+dest+"',arrtime='"+arrtime+"',deptime='"+deptime+"',beizhu='"+beizhu+"' where id= "+id;
connDbBean.executeUpdate(sql);
```

Fig 38 SQL statement in JSP file for flight update function

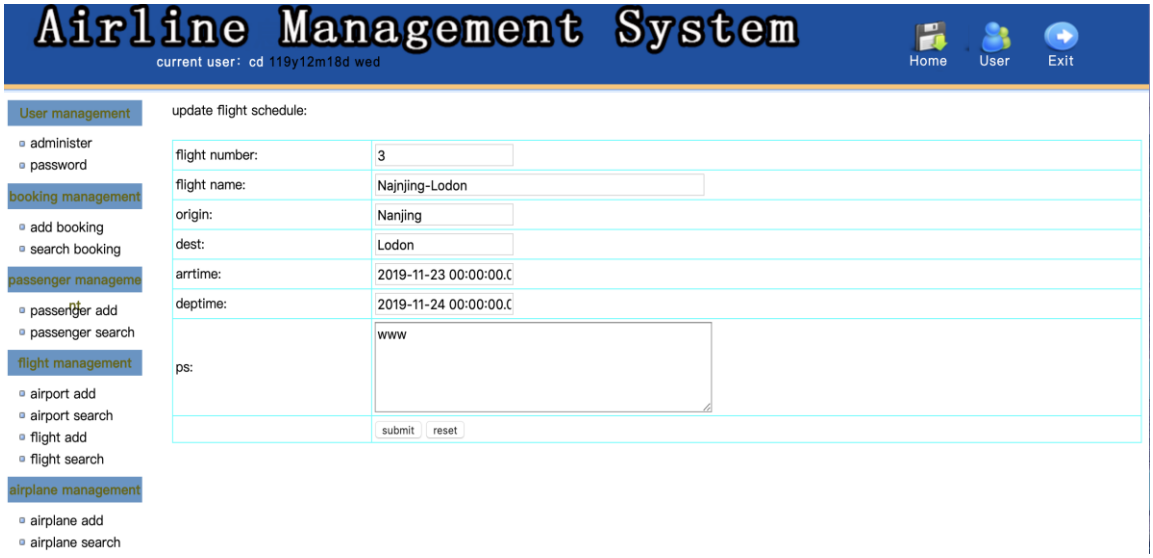


Fig 39 Airport update interface

6. Airplane Management

This function is belong to administer and staff. They can manage the information of airplane.

1) **Add airplane:**

If we want to add a new airplane, we need to enter some information on this airplane, including numser, manufacture, start place, end place, modelnum and typerating. After submitting, a new airplane can be added successfully.

```
String xianlubianhao=request.getParameter("xianlubianhao");String xianlumingcheng=request.getParameter("xianlumingcheng");
String chufadi=request.getParameter("chufadi");String mudedi=request.getParameter("mudedi");
String feiyong=request.getParameter("feiyong");String beizhu=request.getParameter("beizhu");
String faburen=request.getParameter("faburen");String dsj=request.getParameter("dsj");
String id=request.getParameter("id");
String sql="update tbl_flight_schedule set xianlubianhao='"+xianlubianhao+"',xianlumingcheng='"+xianlumingcheng+"',chufadi='"
+chufadi+"',mudedi='"+mudedi+"',dsj='"+dsj+"',feiyong='"+feiyong+"',beizhu='"+beizhu+"',faburen='"+faburen+"' where id= "+id;
connDbBean.executeUpdate(sql);
```

Fig 40 SQL statement in JSP file for airplane add function

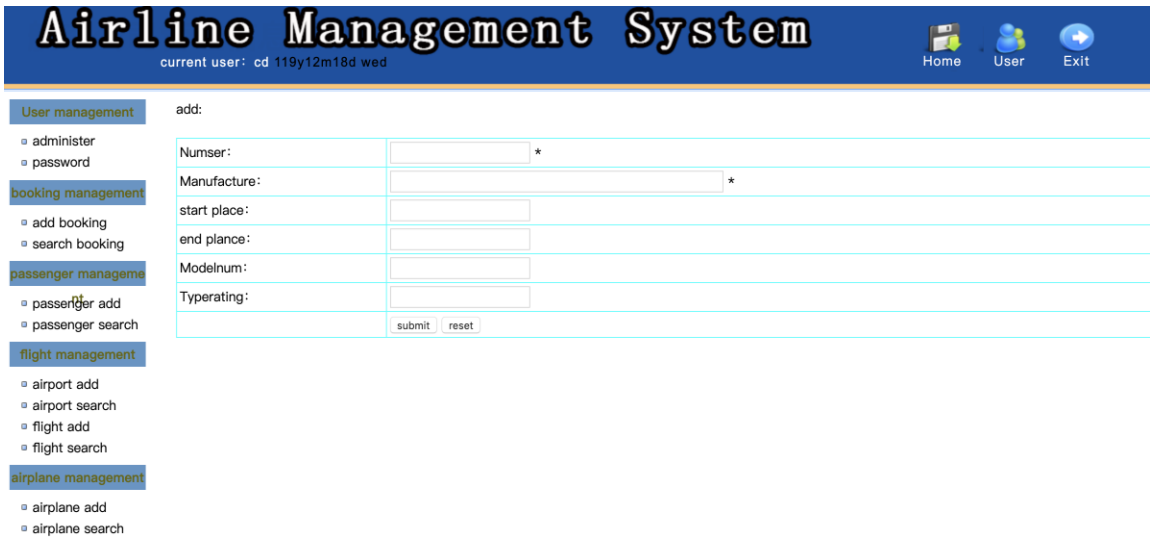


Fig 41 Add new airplane interface

2) Search airplane:

Search airplane: After entering the airplane serial number or manufacture and click search. Then the results of corresponding airplane will be displayed below.

```
String sql="";
sql="select * from tbl_airplane where 1=1";
if(request.getParameter("Numser")=="" ||request.getParameter("Numser")==null )
{}
else
{
  sql=sql+" and Numser like '%"+new String(request.getParameter("Numser").getBytes("8859_1"))+"%'";
}
if(request.getParameter("Manufacture")=="" ||request.getParameter("Manufacture")==null )
{}
else
{
  sql=sql+" and Manufacture like '%"+new String(request.getParameter("Manufacture").getBytes("8859_1"))+"%'";
}
sql=sql+" order by id desc";
ResultSet RS_result=connDbBean.executeQuery(sql);
```

Fig 42 SQL statement in JSP file for airplane search function

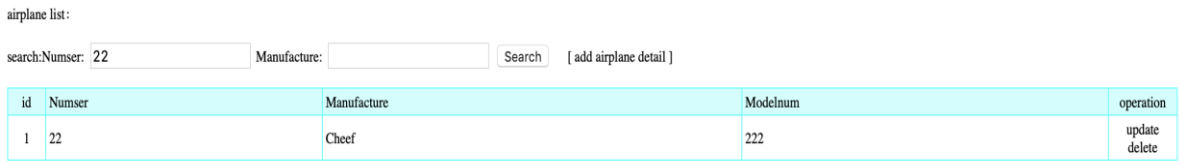


Fig 43 Airplane management interface—search and delete

Update airplane: We also can update the airplane details through entering the new information about the airplane which we would like to change. We can make a modification about the information of the corresponding flight.

```
String xianlubianhao=request.getParameter("xianlubianhao");String xianlumingcheng=request.getParameter("xianlumingcheng");
String chufadi=request.getParameter("chufadi");String mudedi=request.getParameter("mudedi");
String feiyong=request.getParameter("feiyong");String beizhu=request.getParameter("beizhu");
String faburen=request.getParameter("faburen");String dsj=request.getParameter("dsj");
String id=request.getParameter("id");

String sql="update tbl_flight_schedule set xianlubianhao='"+xianlubianhao+"',xianlumingcheng='"+xianlumingcheng+"',chufadi='"
+chufadi+"',mudedi='"+mudedi+"',dsj='"+dsj+"',feiyong='"+feiyong+"',beizhu='"+beizhu+"',faburen='"+faburen+"' where id= "+id;
connDbBean.executeUpdate(sql);
out.print("<script>alert('update successfully!!');location.href='jipiaoxx_updt.jsp?id="+id+"';</script>");
```

Fig 44 SQL statement in JSP file for airplane update function

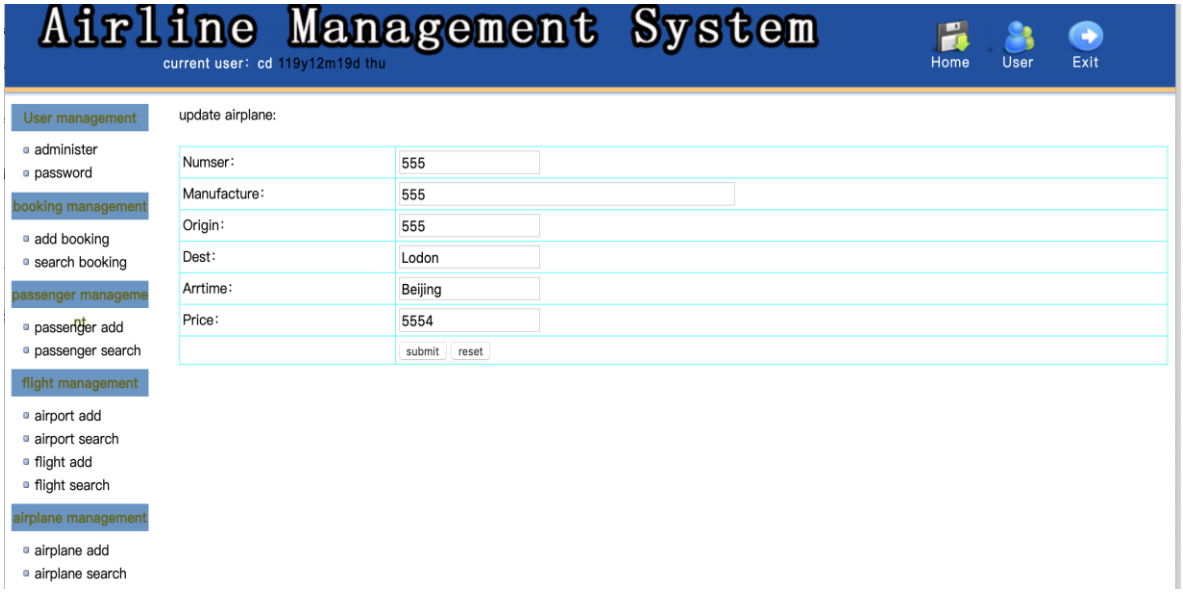


Fig 45 Airplane update interface

5 Data Security

1. Integrity:

The data integrity can be realized through setting some constraints in the process of using SQL statements. In this paper, I used 'PRIMARY KEY', 'NOT NULL' and 'FOREIGN KEY… REFERENCE…' to realize the integrity.

```
CREATE TABLE `tbl_staff_flight` (
  `Empnum` int(11) NOT NULL,
  `Flightnum` int(11) NOT NULL,
  PRIMARY KEY (`Empnum`,`Flightnum`),
  KEY `tbl_staff_flight_ibfk_2` (`Flightnum`),
  CONSTRAINT `tbl_staff_flight_ibfk_1` FOREIGN KEY (`Empnum`) REFERENCES `tbl_staff` (`Empnum`),
  CONSTRAINT `tbl_staff_flight_ibfk_2` FOREIGN KEY (`Flightnum`) REFERENCES `tbl_flight` (`Flightnum`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Fig 46 SQL statement for integrity

1) Primary key constraint

The primary key constraint enforces a unique constraint on the row, while null attributes are not allowed. In the constraint attribute, each record row can only appear once in the data table. In other words, each row of data in the table is unique.

Fields | Indexes | Foreign Keys | Triggers | Options | Comment | SQL Preview

Name	Type	Length	Decimals	Not Null	Virtual	Key
Empnum	int	11	0	☑	☐	🔑1
Flightnum	int	11	0	☑	☐	🔑2

Fig 47 Primary key and not null constraint for entity integrity

2) Foreign Key Constraint

Foreign key constraints can enforce referential integrity. Foreign key constraints are defined on a set of attributes in the referencing table and point to a set of primary keys in the referenced table.

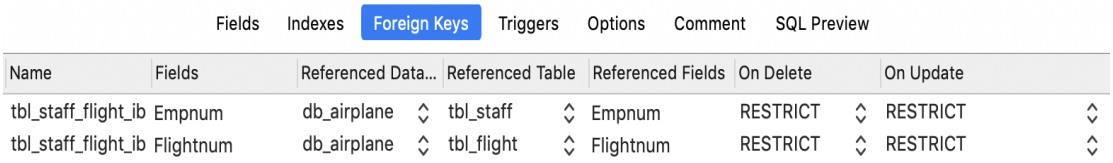


Fig 48 Foreign key constraint for referential integrity

2. Privileges

To realize the privileges, I set two different role for this system, including staff and administer. For staff user, they can manage the airplane, flight, airport and booking. For administer user, they can use the same functions like staff, and they also can manage the staff. Both users can login the system, and they can modify their personal information like password.

3. Views:

A view contains rows and columns, just like a real table. The fields in the view are the fields from a real table in one or more databases. There is a view created for tbl_flight. Views can be created and displayed through SQL statements.

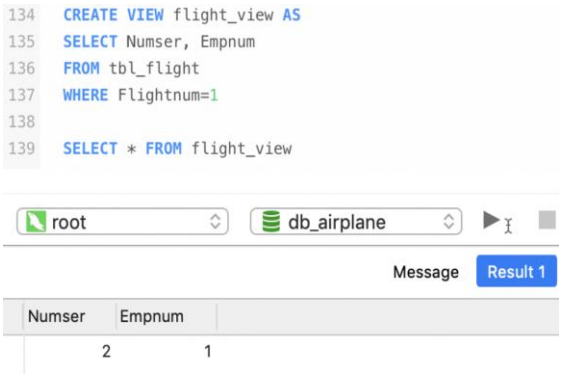


Fig 49 SQL statement for database view

4. Recovery

In this paper, to avoid the data lost, we can use the following statement to back up the data in the table.

```
use hangkong;

select * from tbl_staff

into outfile 'd:/file/myfile1.txt'

fields  terminated by ','

optionally enclosed by '"'

lines terminated by '?';
```

Fig 50 SQL statement for data backup

We can back up the database at the system terminal.

```
pc-204-80:~ yangsiqi$ mysqldump -u root -p hangkong > desktop:\db.bak
Enter password:
```

Fig 51 MySQL database backup

At the same time, we can complete the recovery of the database.

```
pc-204-80:~ yangsiqi$ mysql -u root -p hangkong < desktop:\db.bak
Enter password:
```

Fig 52 MySQL database recovery

6 Conclusion

This article completes the design and implementation of the airline system front-end interface based on the MySQL database designed in part 1. Through this system, staff user can make a booking for passenger, the information of booking, passenger, flight, airplane can also be managed by them. Administer can do the same thing like staff user, apart from this, administer can manage the staff information. In this paper, first summarize the assignment 1 and make appropriate changes. Then introduced the development and running environment of the system. Then the design and implementation of the system are explained in detail. Finally, some technical thoughts on data security are introduced.

During the process, I understood that if we want to develop a application, first thing we need to do is designing a right and suitable database. Then we need to use some technologies such as JDBC, ODBC to make a connection with the database and the application. At the same time, I learned how to use eclipse and MySQL. And how to implement a front-end interface of some basic function by using JSP, JS, CSS and other techniques. This system can basically complete the management of data information for an airline system, but it still exists many things that need to be improved.