Implementation of Queries and Interactions with Database for Airlines System

Rui Fu

University of Derby

r.fu2@unimail.derby.ac.uk

Table of Content

1 Introduction

As a web application applied for a certain airline company, it is necessary for this system to provide two individual aspects of accesses to meet fundamental requirements for normal users, who want to book flight tickets of this company on the website, and administrators of this company whose responsibility is to manage all affairs of operating system efficiently and effectively. In the process of operation, the role of administrators always has higher privileges and more accesses to get detailed data of whole company, which is used to collect and analyse before generating reports per-week or per-month, including pilot schedule, flight schedule and so on.

This project achieves these functionalities by using MySQL as the database to store data tables` structure and specific data. As related database, every slide of data in the database has been identified by identifying unique column like the primary key. And foreign key is utilized to deliver and band the connection between different tables. Besides, the coding language used by MySQL is called SQL, which can be inserted and implemented to implement functions in plenty of other languages. CRUD (Create, Read, Update and Delete) (Martin,1981) is the basic operations in any RDBMS (Relational Database Management System). In the MySQL, it is commonly considered to map to Insert, Select, Update and Delete respectively. In addition, PHP is used as the programming language to implement UI (User Interface) and functions in this project.

This essay is a brief report for every part of this system and principles used back to them. Based on the 3rd normalized tables, E-R diagram and feedbacks got from the lecturer in the last part of assignment in this module, this essay starts with improvement on contents and structure of E-R diagram has been improved. This is followed by the development and operation environment through brands and versions of these software used during this process. The next chapter introduces the whole process of implementation in four aspects, database connection, functions for normal users and functions for administrators, and database security. Finally, this essay presents reflection and conclusion for this project, and analyse the strength and weakness for this implement process.

2 Improvement of Database Based on Feedbacks of Part 1

According to the feedback given by lecturer on the 1st part of the assignment, a series of problems about the database structure in this project has been discovered. Considering the importance of database for this web application project, these problems cannot be ignored. Otherwise, it is entirely possible that we simply cannot afford to the price and consequences taken by them. Therefore, the reconstruction and missing of structure and content in this project will be shown below.

Firstly, I need to make clear the reasons for normalization database tables, which should be presented in the 1st part. As we all know, database normalization (Codd,1970) is the process of restructuring a relational database in accordance with a series of normal forms in order to reduce data redundancy and improve data integrity. After identifying entities and creating table structure, the process of normalization helps

to avoid undesirable effects in three aspects, update anomaly, insert anomaly and delete anomaly, and maintain data atomicity, which means related data should be updated or delete in the same time and data without redundancy can take up smaller memory space (Storey & Song, 2017). Even though, this kind of operation is likely to lead to increasement on the amount of displaying data from multiple tables, it is still an effective way to maintain ACID (Atomicity, Consistency, Isolation and Durability) (Peltz, 2003) for database.

In addition, because of the inappropriate attributes' management, the construct of some tables has been changed to fit the fact and requirements asked by sponsors. After proper adjustment on tables, there are eleven tables created to store structuring data, the table`s name and attributes are shown below.

Table 1. Tables and Attributes

Table Name	Attributes
Booking	(book_id, user_id, b_time, b_state, flight_id, passenger_amount, b_amount, b_payway)
Admin	(name, password)
User	(user_id, user_name, user_pswd)
Ticket	(ticket_id, book_id, passenger_id, t_state, t_from_cityid, t_to_cityid
T_info	(id, flight_id, ticket_id, t_seat_no, t_seat_type, t_price, t_baggage_requirement)
Passenger	(passenger_id, p_surname, p_givenname, p_e_surname, p_e_givenname, p_id_num, p_birth, p_gender, p_nation, p_address, p_phone, p_level)
Flight	(flight_id, f_no, aiecraft_type, f_level, f_date, f_duration, f_arr_punct_rate, f_avg_delay_time, f_ticket_price, f_ticket_discount, f_seat_amount, f_seat_avail_num, f_origin_cityid, f_board_terminal, f_board_gate, f_board_time, f_dept_time, f_stop_cityid, f_stop_terminal, f_stop_arr_time, f_stop_dept_time, f_dest_time, f_dest_cityid, f_dest_terminal)
F_crew	(id, flight_id, position, staff_id)
Staff	(staff_id, s_surname, s_givenname, s_e_surname, s_e_givenname, s_type_id, s_phone, s_salary, s_address, s_gender, s_age, s_workage)
S_type	(s_rating_id, s_type_name, s_work_content)
City	(city_id, city_name, city_level, city_terminal_num)

Compared with the last database design, there are some changes shown in this table. Details and reasons are listed below in five points.

1. To begin with, the table admin is created to join in this group to store login information for administrators, as a precondition to identify roles and manage information of this airlines company for running normally.

2. And then, an attribute called “b_date” of table booking has been replaced by “b_time”, because during the process of test, I found only the date of booking is not an accuracy way to record data, but the exact time can solve this problem.
3. After that, due to considering the situation of stopover, the place of origin and destination might be different from the information filled in the table of flight. So, attributes called “t_from_cityid” and “t_to_cityid” are inserted into table of ticket to ensure the accuracy of data.
4. In addition, the entity of passenger is an independent one to store information of passenger themselves, rather than information related to flight trip. That is the reason why tables of ticket and t_info cover “flight_id”, “passenger_id” and their “booking_id” together.
5. Finally, some attributes of flights cannot be identified before their precondition happen, such as the board gate. Passengers cannot get the entity number of their gate before their flight date. So, I set up the property of this kind of attributes as “Can Be Null” before being filled in by administrators, not only for the normal operation of the whole system, but also for respecting the facts.

In summary, these are changes of essay critical analysis expression and database structure to make the design and implement of this project better. And I am deeply appropriate for my lecturer`s feedback and suggestion. Based on this new database, the programming of implementation has been started.

3 Development and Operation Environment

In this chapter, the list of developing environments has been shown in three parts, hardware environment, development tools and operation & test browsers.

1. Hardware Environment:

Intel® Core (TM) i7-4470 CPU @ 3.40GHz 3.40GHz (Window 8.1).

2. Development Tools:

XAMPP (Version: 7.2.12): Control Panel (Version: 3.2.2 [Compiled: Nov 12th, 2015]);
MySQL Workbench 6.3.10 Build 12092614 CE (64 Bits) Community;
HeidiSQL (Version 9.5.0.5196 (64 Bits)).

3. Operation and Test Environment:

Browser: Google Chrome (Version: 69.0.3497.81 (32 Bits));
Firefox (Version: 63.0.3 (64 Bits)).

4 Implementation

In this charter, following content will show most of primary pages about the implementation process of database connection, client-side implementation, server-side implementation and data security with text and screenshots.

4.1 Connection with Database

Before the implementation actions, the developing environment need to be built completely. In this project, the accomplishment of developing environment involves a series of software, including MySQL workbench, HeidiSQL and XAMPP, with different responsibility in different aspects.

1. Firstly, as a simple and friendly user interface view for users to see and edit data and structures from computers running one of the database system MySQL and so on (Becker, 2018), HeidiSQL is used to create and improve database created for this subject. At the same time, test records can be stored in the database in this step. (Patinge & Talmale, 2017) After that, we can also observe the database and test data from the MYSQL workbench.

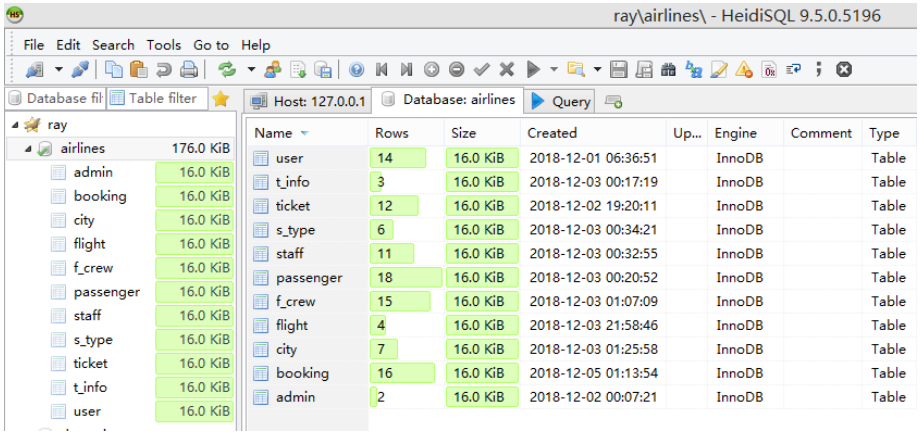


Fig. 1. User Interface of HeidiSQL

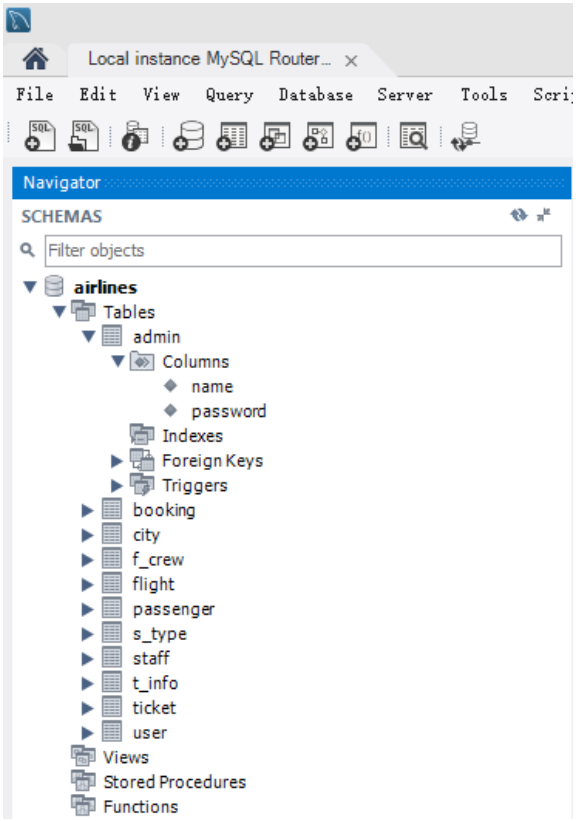


Fig. 2. Database Tables Structure Shown in MySQL Workbench

2. The next step is to identify the consistency that PHP coding software XAMPP can connect with database which has been created in the same IP and port before and get all the permission of accessing and editing. After starting the action ports of Apache and MySQL, the button "admin" of MySQL provides an access to "localhost/phpMyAdmin" that is also an interface for MySQL database, like HeidiSQL.

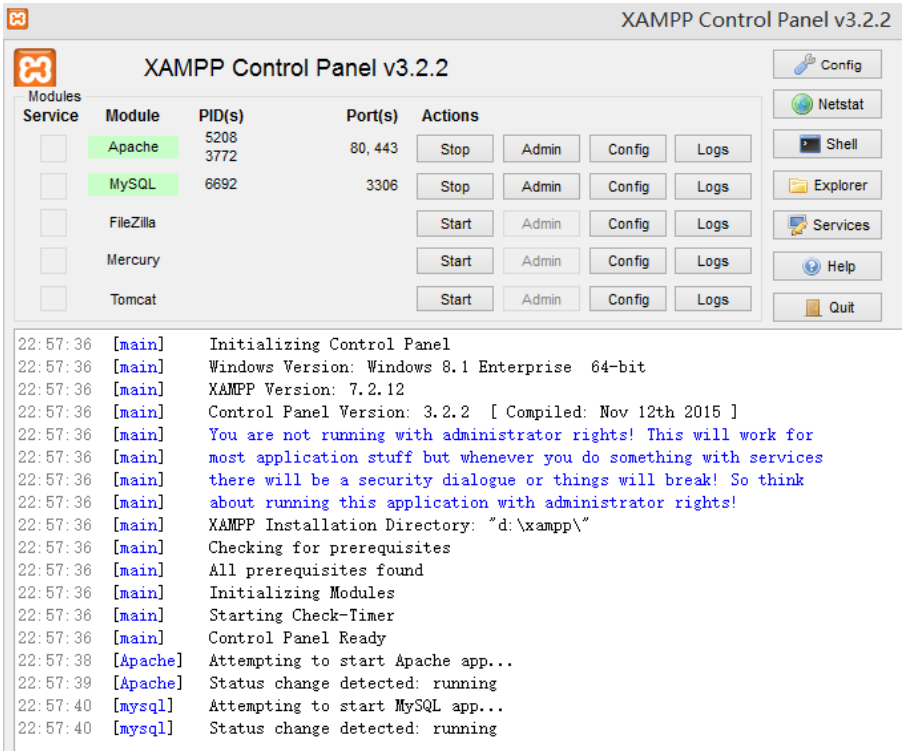


Fig. 3. XAMPP Starts Actions to Provide Ports for Apache and MySQL

Fig. 4. Check database stored in MySQL through phpMyAdmin

3. Finally, php document connected with MySQL is based on host, username and password of database. This file is used to create an interface with MySQL to ensure normal running of adding and controlling data stored in the database.

```php
<?php
class Database {
    private $_connection;
    private static $_instance; //The single instance
    private $_host = "localhost";
    private $_username = "root";
    private $_password = "1234";
    private $_database = "airlines";

    public static function getInstance() {
        if(!self::$_instance) { // If no instance then make one
            self::$_instance = new self();
        }
        return self::$_instance;
    }

    // Constructor
    private function __construct() {
        $this->_connection = new mysqli($this->_host, $this->_username,
            $this->_password, $this->_database);

        // Error handling
        if(mysqli_connect_error()) {
            trigger_error("Failed to connect to MySQL: " . mysql_connect_error(),
                E_USER_ERROR);
        }

        //echo "connection successful";
    }

    // Magic method clone is empty to prevent duplication of connection
    private function __clone() { }

    // Get mysqli connection
    public function getConnection() {
        return $this->_connection;
    }
}
?>
```

4.2 Operation Processes for Users

In this project, the operation process for users includes login, register, the flow of booking flight tickets and check bookings.

Login. Login is a basic function for almost all the websites, which is used store track of users and booking with id of every user and its interface is shown below. I take it as an example to show the user interface, form created for user interface to transfer data inputted by post method in HTML and php part to execute the action of transferring data with SQL language.

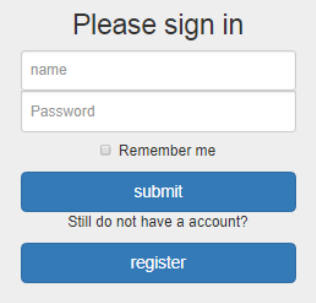


Fig. 5. User Interface for Users to Log In

Codes for user interface:

```
<div class="container">
  <form action="" method="post" class="form-signin">
    <h2 class="form-signin-heading" align="center">Please sign in</h2>
    <input type="text" name="name" placeholder="name" class="
    form-control" required autofocus>
    <input type="password" name="password"  class="form-control"
    placeholder="Password" required>
    <div class="checkbox" align="center">
        <label>
        <input type="checkbox" name="remember-me" value="remember-me">
         Remember me
        </label>
    </div>
   <input type="submit" class="btn btn-lg btn-primary btn-block" name=
   "submit" value="submit">
   <P align="center">Still do not have a account?</P>
   <a type="submit" class="btn btn-lg btn-primary btn-block" href="
   register.php" name="Register" value="Register">register</a><br />
  </form>
</div> <!-- /container -->
```

And codes for transfer data action in PHP:

```php
<?php
session_start();
include('connection.php');
if (isset($_POST["submit"])) {
  $name = trim($_POST["name"]);
  $pswd = trim($_POST["password"]);
      $sql = "SELECT user_name,user_pswd from user where user_name = '$name
        ' and user_pswd = '$pswd'";
      $result = mysqli_query($db->getConnection(), $sql);
      if (mysqli_num_rows($result)) {
              $_SESSION['user_name']=$name;
              $_SESSION['islogin']=1;
              if($_POST["remember-me"]=="remember-me"){
                setcookie('name',$name,time()+7*24*60*60);
                setcookie('pswd',md5($name.md5($pswd)),time()+7*24*60*60);
              }else{
                setcookie('name','',time()-999);
                setcookie('pswd','',time()-999);
              }
              echo "<script>alert('log in success！')</script>";
              header("Location: index.php");
      }else{
       echo "<script>alert('name or password is wrong！')</script>";
      }
}
?>
```

Register. Register is also an important action for websites, and always appears with login page. If user does not have an account in the server, he needs to register first before booking or other actions. The form and transfer method used by register is similar with the login page, so this part just shows the user interface and part of php background coding.

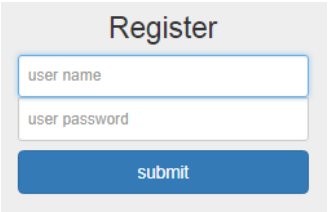


Fig. 6. User Interface for Users to Register

PHP coding to judge if this user name has existed, if not, insert this slide of record in the database and then return to the login page.

```
        $sql = "SELECT user_name from user where user_name = '$name' ";
        $result = mysqli_query($db->getConnection(), $sql);
        if (mysqli_num_rows($result)) {
            echo "<script>alert('user name has
              exist');history.go(-1);</script>";
        } else {
            $sql1 = "INSERT INTO user (user_name,user_pswd) values ('$name','$
              pswd') ";
            mysqli_query($db->getConnection(), $sql1);
            echo "<script>alert('Register succeed')</script>";
            header("Location: login.php");
        }
    }
```

The Flow of Booking Ticket. The flow of booking ticket happens from index of this system as below.

Users have access button to select flights by flights list or search them by inputting the origin city and destination city to get flight list from index page. Based on the information provided by flights list, users can check the details of the flight they prefer after they log in. The button “Book it” in detail page is used to book a ticket and redirect to page of adding passenger. After adding information of passenger, the booking system provide a table involving information of flight and the passenger to let users identify their booking details again and pay for it. Then, the information of this booking would be stored in the database and redirect to the “booking success” page. The process of booking ticket has finished, and user interface is presented by screenshots below.

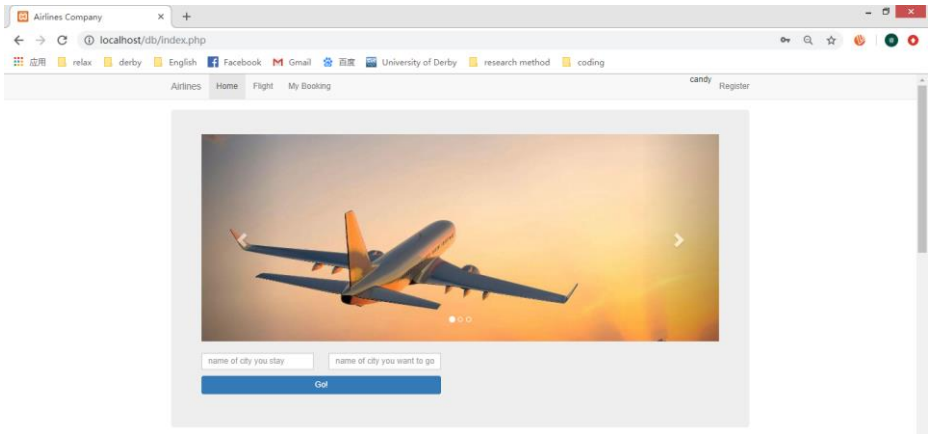
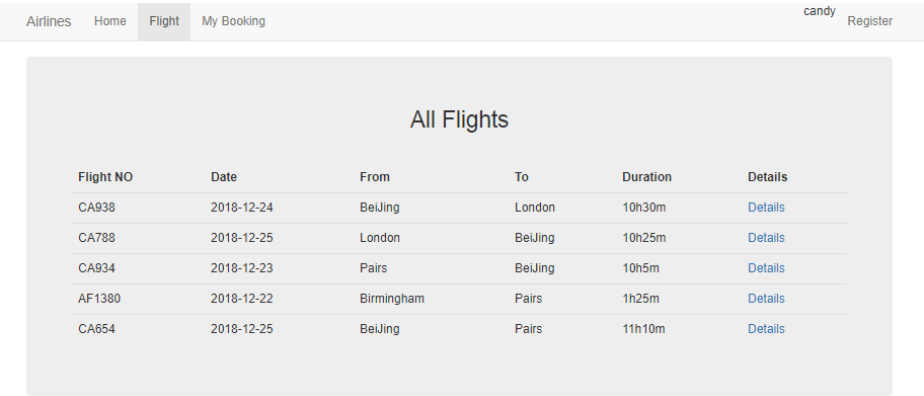


Fig. 7. Index Page for Users



Flight NO	Date	From	To	Duration	Details
CA938	2018-12-24	BeiJing	London	10h30m	Details
CA788	2018-12-25	London	BeiJing	10h25m	Details
CA934	2018-12-23	Pairs	BeiJing	10h5m	Details
AF1380	2018-12-22	Birmingham	Pairs	1h25m	Details
CA654	2018-12-25	BeiJing	Pairs	11h10m	Details

Fig. 8. Flights List Page

Codes for showing flights information as an example in php:

```
$sql_1 = " SELECT flight_id,flight.f_no,flight.f_date,city.city_name,
  flight.f_duration FROM flight,city where flight.f_origin_cityid=
  city.city_id";
$result_1 = mysqli_query($db->getConnection(), $sql_1);
$sql_2 = " SELECT city_name FROM flight,city where flight.
  f_dest_cityid=city.city_id";
$result_2 = mysqli_query($db->getConnection(), $sql_2);

while($row_1 = mysqli_fetch_array($result_1)) {
  $row_2 = mysqli_fetch_array($result_2);
    echo "<tr>".
          "<td>".$row_1["f_no"]."</td>".
          "<td>".$row_1["f_date"]."</td>".
          "<td>".$row_1["city_name"]."</td>".
          "<td>".$row_2["city_name"]."</td>".
          "<td>".$row_1["f_duration"]."</td>".
          "<td>"."<a href='f_details.php?id=".$row_1["flight_id"]."
            '>Details</a></td>".
          "</tr>";
        }
```

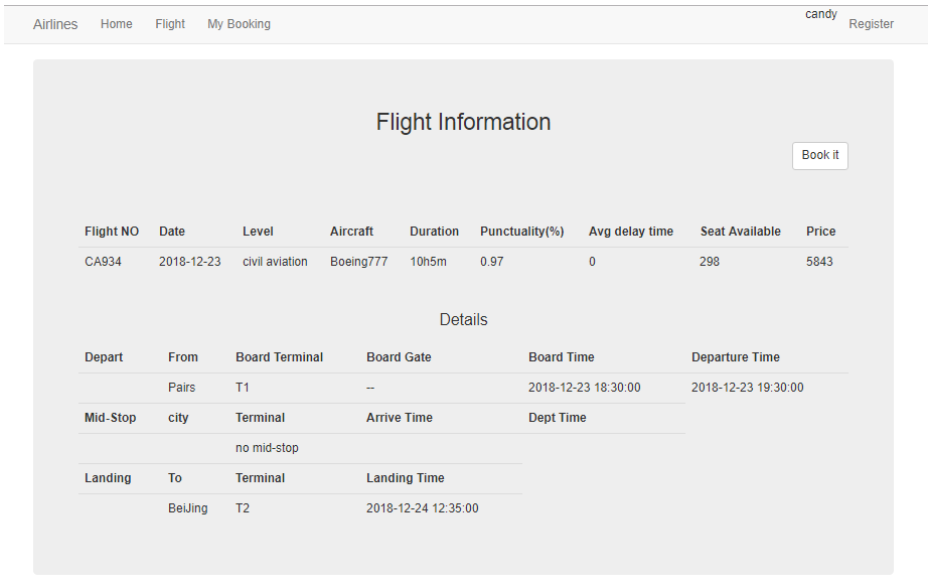


Fig. 9. Flight Details Page

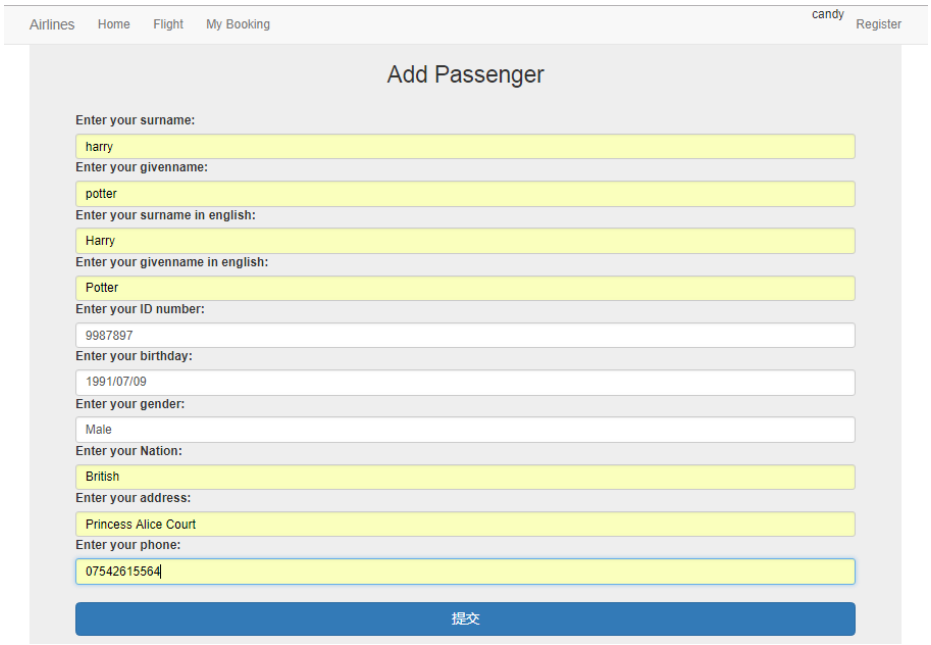


Fig. 10. Add Passenger Page

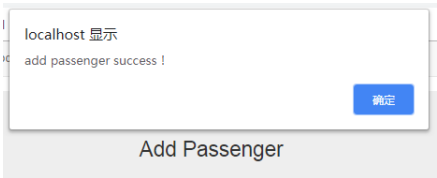


Fig. 11. Tap for Adding Passenger Success

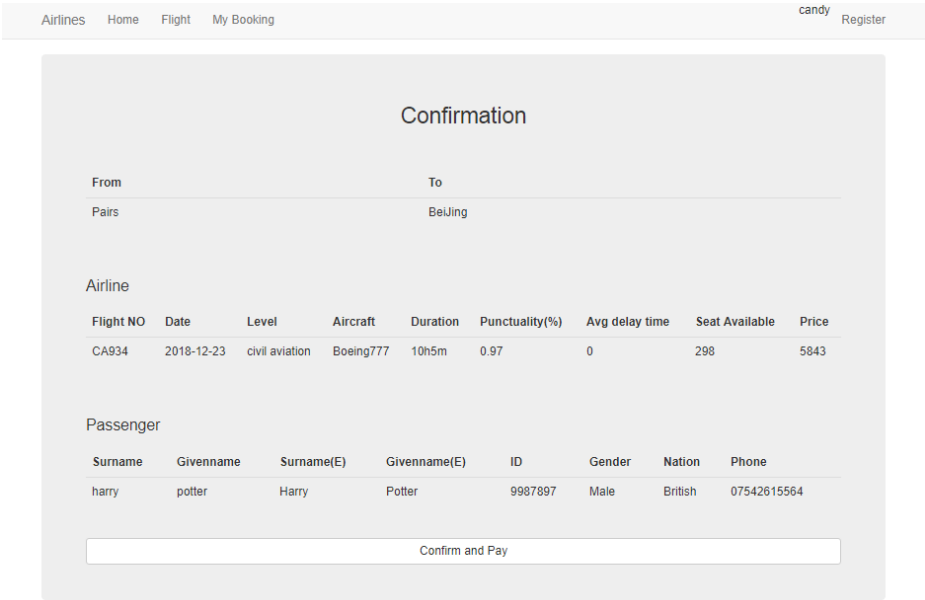


Fig. 12. Details Confirmation Page

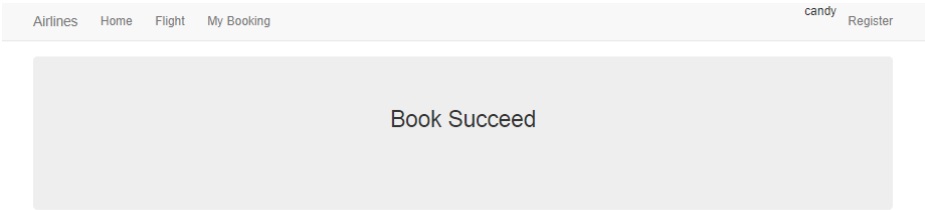


Fig. 13. Booking Success Page

Check Booking by User. Through clicking the button “My Booking” in the navigation bar, users can check their own booking details and they have access to delete the booking as screenshot follows.

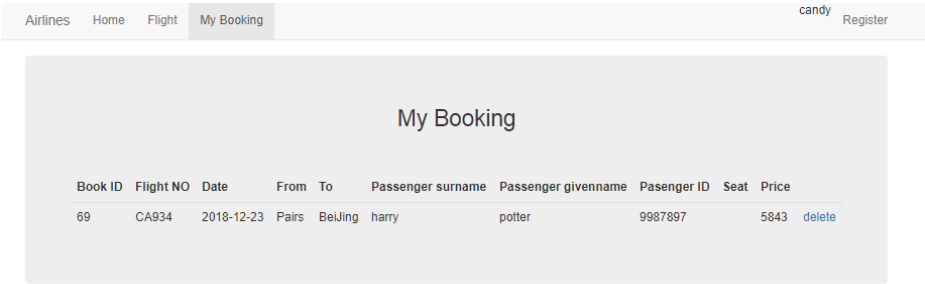


Fig. 14. Check Booking by The User

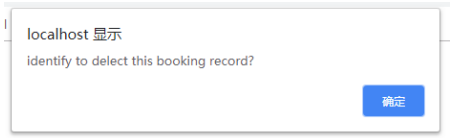


Fig. 15. Tap for Record Delete

4.3 Operation Processes for Administrators

Operation processes for administrators are divided into several parts. After login page, four reports of flight-passenger number, pilot schedule and pilot working hours (ascending &descending) are shown in the index of administrator, and the details button after each flight in the first form redirect to details of passenger who book this flight.

From the side bar, administrators in this system need to manage staffs in eight aspects, flights, staff, crew, machines, city, passengers, books and tickets. The management of flights, staff and city covers the access to view all records, add a new one, update and delete the existing record. And other management function only covers the access to log out from this system and redirect to the log in page.

Login &Reports &Log out

Fig. 16. Login Page for Administrators

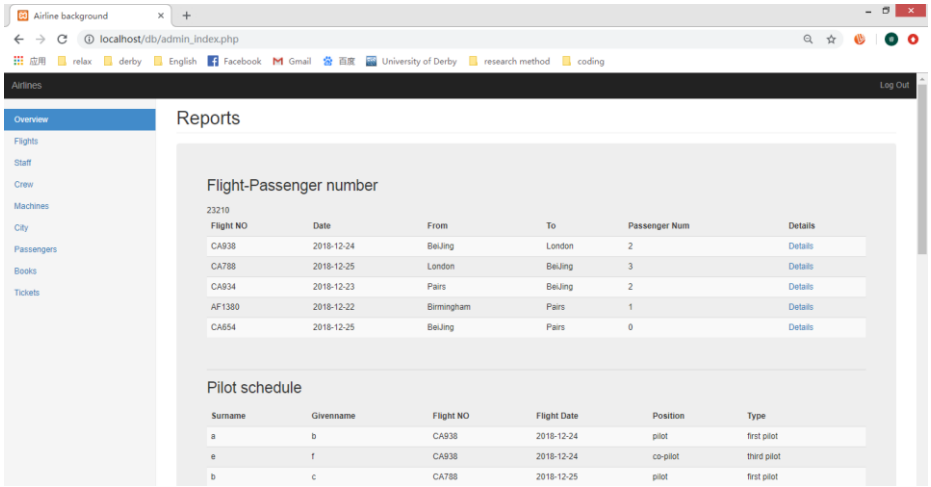


Fig. 17. Reports for Administrators (1)

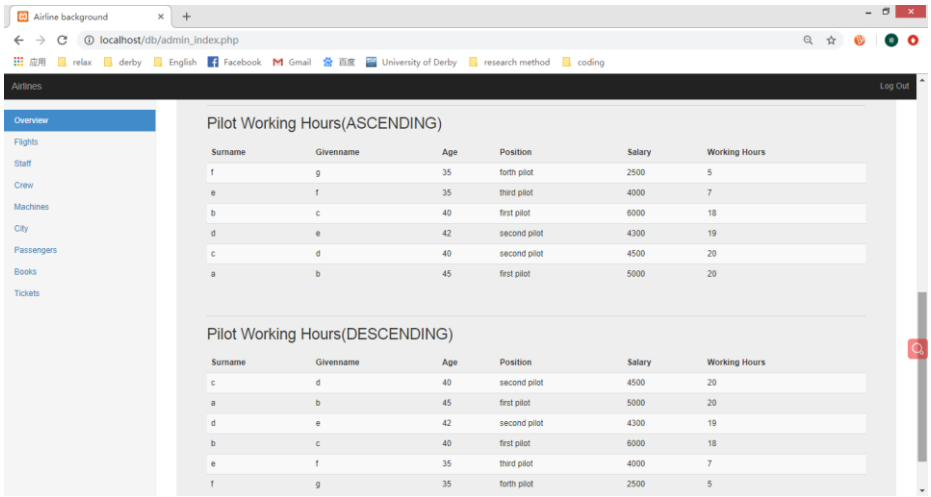


Fig. 18. Reports for Administrators (2)

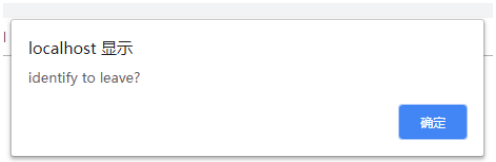


Fig. 19. Tap for Log Out

Flight Management Span and Queries. In this chapter, I pick up the flight management span to present the implement of CRUB (Create Read Update Delete) in admin

side. The first page shows all the records of flight in the database based on “select” function. The update page is the access to change record of flight, origin data has showed on it, administrator can change the option he wants to change directly by “update”. The delete link is used to delete the record that administrator choose from database by “delete”. Besides, the button “add a new flight” creates a new page for user to fill in the flight information table and submit it to store in the database by “insert” function. User interfaces are presented by screenshots below.

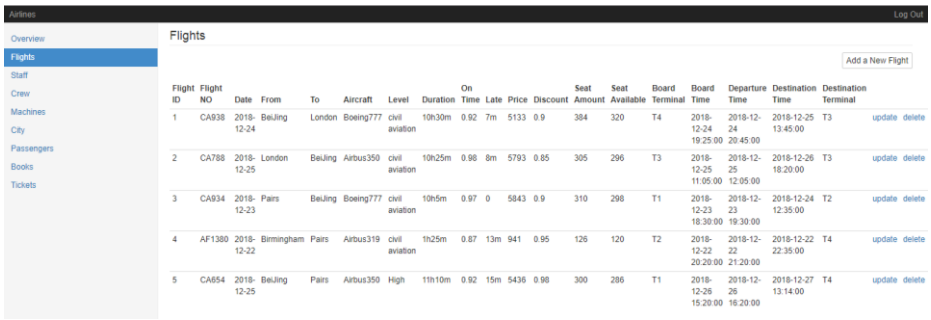


Fig. 20. Flight Management Span

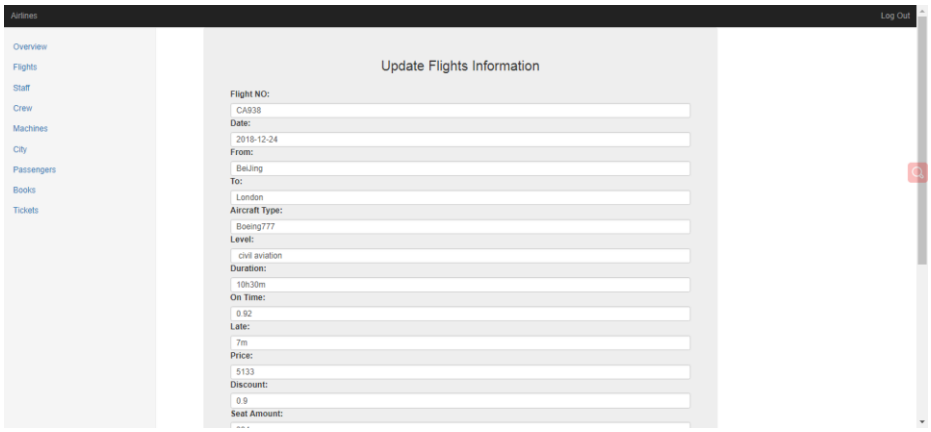


Fig. 21. Update Flight Information

Fig. 22. Add A New Flight Information

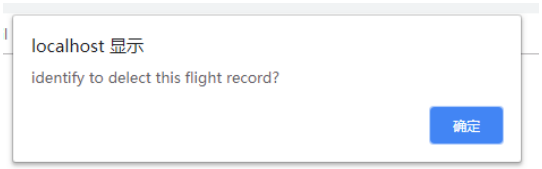
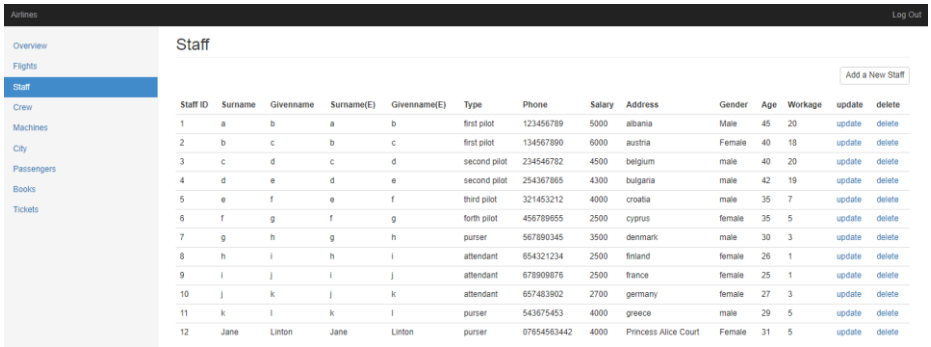


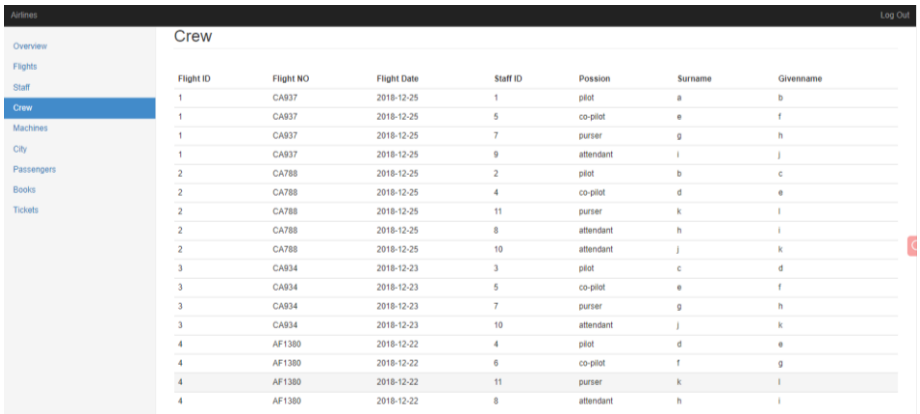
Fig. 23. Delete Flight Information Record

Other Management Spans



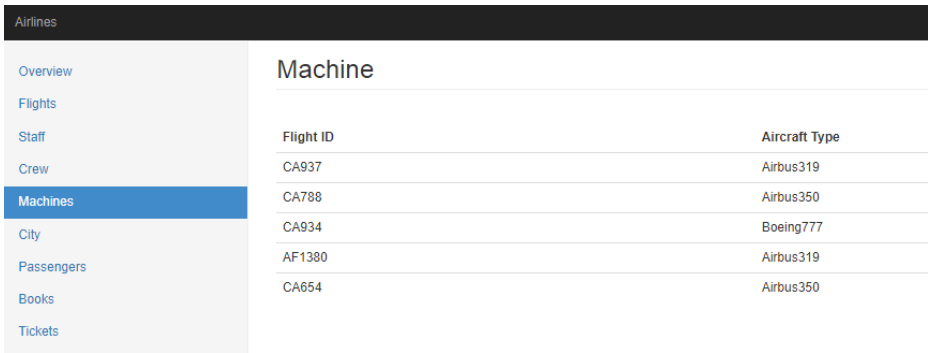
Staff ID	Surname	Givenname	Surname(E)	Givenname(E)	Type	Phone	Salary	Address	Gender	Age	Workage	update	delete
1	a	b	a	b	first pilot	123456789	5000	albania	Male	45	20	update	delete
2	b	c	b	c	first pilot	134567890	6000	austria	Female	40	18	update	delete
3	c	d	c	d	second pilot	234546782	4500	belgium	male	40	20	update	delete
4	d	e	d	e	second pilot	254367865	4300	bulgaria	male	42	19	update	delete
5	e	f	e	f	third pilot	321453212	4000	croatia	male	35	7	update	delete
6	f	g	f	g	forth pilot	456789655	2500	cyprus	female	35	5	update	delete
7	g	h	g	h	purser	567890345	3500	denmark	male	30	3	update	delete
8	h	i	h	i	attendant	654321234	2500	finland	female	26	1	update	delete
9	i	j	i	j	attendant	678909876	2500	france	female	25	1	update	delete
10	j	k	j	k	attendant	657483902	2700	germany	female	27	3	update	delete
11	k	l	k	l	purser	543675453	4000	greece	male	29	5	update	delete
12	Jane	Linton	Jane	Linton	purser	07654563442	4000	Princess Alice Court	Female	31	5	update	delete

Fig. 24. Staff Management Span



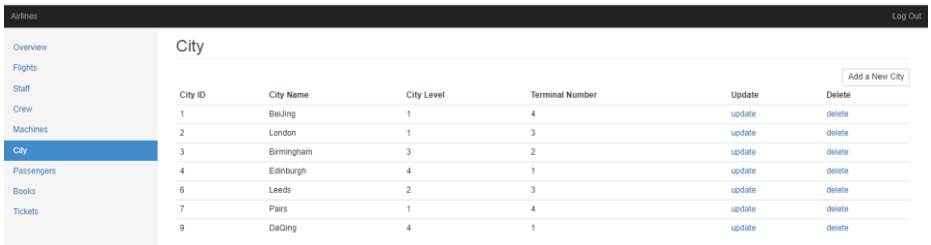
Flight ID	Flight NO	Flight Date	Staff ID	Possion	Surname	Givenname
1	CA937	2018-12-25	1	pilot	a	b
1	CA937	2018-12-25	5	co-pilot	e	f
1	CA937	2018-12-25	7	purser	g	h
1	CA937	2018-12-25	9	attendant	i	j
2	CA788	2018-12-25	2	pilot	b	c
2	CA788	2018-12-25	4	co-pilot	d	e
2	CA788	2018-12-25	11	purser	k	l
2	CA788	2018-12-25	8	attendant	h	i
2	CA788	2018-12-25	10	attendant	j	k
3	CA934	2018-12-23	3	pilot	c	d
3	CA934	2018-12-23	5	co-pilot	e	f
3	CA934	2018-12-23	7	purser	g	h
3	CA934	2018-12-23	10	attendant	j	k
4	AF1380	2018-12-22	4	pilot	d	e
4	AF1380	2018-12-22	6	co-pilot	f	g
4	AF1380	2018-12-22	11	purser	k	l
4	AF1380	2018-12-22	8	attendant	h	i

Fig. 25. Crew Management Span



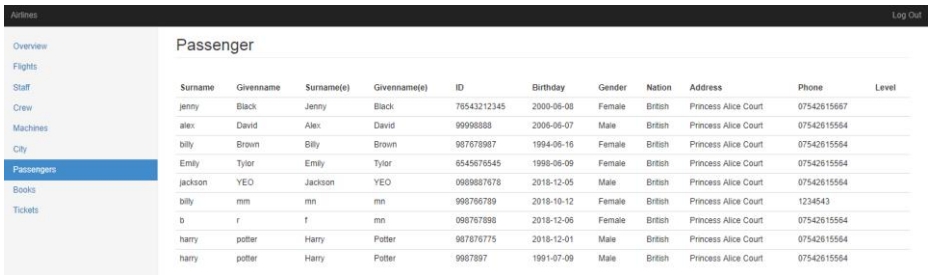
Flight ID	Aircraft Type
CA937	Airbus319
CA788	Airbus350
CA934	Boeing777
AF1380	Airbus319
CA654	Airbus350

Fig. 26. Machine Management Span



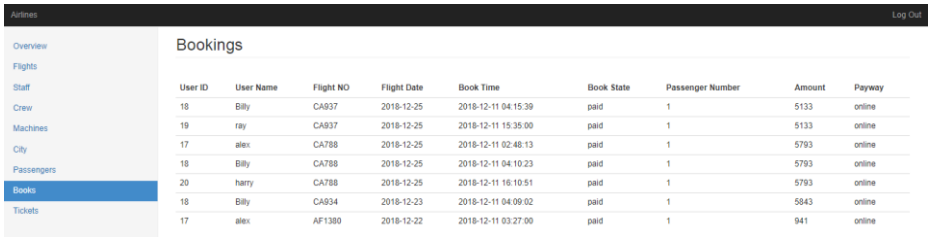
City ID	City Name	City Level	Terminal Number	Update	Delete
1	BeiJing	1	4	update	delete
2	London	1	3	update	delete
3	Birmingham	3	2	update	delete
4	Edinburgh	4	1	update	delete
6	Leeds	2	3	update	delete
7	Pairs	1	4	update	delete
9	DaQing	4	1	update	delete

Fig. 27. City Management Span



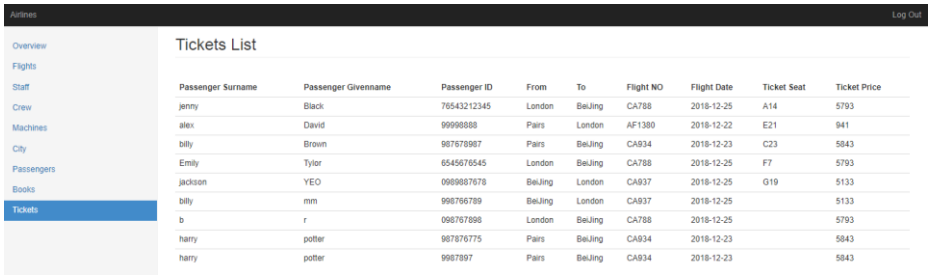
Surname	Givenname	Surname(e)	Givenname(e)	ID	Birthday	Gender	Nation	Address	Phone	Level
jenny	Black	Jenny	Black	76543212345	2000-06-08	Female	British	Princess Alice Court	07542615667	
alex	David	Alex	David	99998888	2006-06-07	Male	British	Princess Alice Court	07542615564	
billy	Brown	Billy	Brown	987678987	1994-06-16	Female	British	Princess Alice Court	07542615564	
Emily	Tylor	Emily	Tylor	6545676545	1998-06-09	Female	British	Princess Alice Court	07542615564	
jackson	YEO	Jackson	YEO	0989887678	2018-12-05	Male	British	Princess Alice Court	07542615564	
billy	mm	mn	mn	998766789	2018-10-12	Female	British	Princess Alice Court	1234543	
b	r	f	mn	098767898	2018-12-06	Female	British	Princess Alice Court	07542615564	
harry	potter	Harry	Potter	987876775	2018-12-01	Male	British	Princess Alice Court	07542615564	
harry	potter	Harry	Potter	9987897	1991-07-09	Male	British	Princess Alice Court	07542615564	

Fig. 28. Passenger Management Span



User ID	User Name	Flight NO	Flight Date	Book Time	Book State	Passenger Number	Amount	Payway
18	Billy	CA937	2018-12-25	2018-12-11 04:15:39	paid	1	5133	online
19	ray	CA937	2018-12-25	2018-12-11 15:35:00	paid	1	5133	online
17	alex	CA788	2018-12-25	2018-12-11 02:48:13	paid	1	5793	online
18	Billy	CA788	2018-12-25	2018-12-11 04:10:23	paid	1	5793	online
20	harry	CA788	2018-12-25	2018-12-11 16:10:51	paid	1	5793	online
18	Billy	CA934	2018-12-23	2018-12-11 04:09:02	paid	1	5843	online
17	alex	AF1380	2018-12-22	2018-12-11 03:27:00	paid	1	941	online

Fig. 29. Booking Management Span



Passenger Surname	Passenger Givenname	Passenger ID	From	To	Flight NO	Flight Date	Ticket Seat	Ticket Price
jenny	Black	76543212345	London	BeiJing	CA788	2018-12-25	A14	5793
alex	David	99998888	Pairs	London	AF1380	2018-12-22	E21	941
billy	Brown	987678987	Pairs	BeiJing	CA934	2018-12-23	C23	5843
Emily	Tylor	6545676545	London	BeiJing	CA788	2018-12-25	F7	5793
jackson	YEO	0989887678	BeiJing	London	CA937	2018-12-25	G19	5133
billy	mm	998766789	BeiJing	London	CA937	2018-12-25		5133
b	r	098767898	London	BeiJing	CA788	2018-12-25		5793
harry	potter	987876775	Pairs	BeiJing	CA934	2018-12-23		5843
harry	potter	9987897	Pairs	BeiJing	CA934	2018-12-23		5843

Fig. 30. Ticket Management Span

4.4 Data Security

Integrity Rules. Database integrity rule is a kind of method to ensure completeness, accuracy and consistency of data. It covers three aspects to follow, entity Integrity, referential integrity and domain integrity (Codd, 1970). Having a single, well-defined and well-controlled data integrity system increases stability, performance, reusability and maintainability. Data integrity is usually imposed during the database design phase through the use of standard procedures and rules (Techopedia.com, 2018). Data integrity can be maintained through the use of various error-checking methods and validation procedures. In this project, they present in different aspects.

Entity Integrity. Entity integrity is concerned with ensuring that each row of a table has a unique and non-null primary key value (Sqa.org.uk, 2018), which means the unique of each row data to avoid mess of records.

1	user_id	INT	11	☐	☐	☐	AUTO_INCREMENT
2	user_name	VARCHAR	20	☐	☑	☐	NULL
3	user_pswd	VARCHAR	20	☐	☑	☐	NULL

Fig. 31. Example of Primary Key in Tables of Database

Sql codes for primary key:

```
CREATE TABLE `user` (
  `user_id` INT(11) NOT NULL AUTO_INCREMENT,
  `user_name` VARCHAR(20) NULL DEFAULT NULL,
  `user_pswd` VARCHAR(20) NULL DEFAULT NULL,
  PRIMARY KEY (`user_id`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
AUTO_INCREMENT=22
;
```

Referential Integrity. Referential integrity requires every value of one attribute (column) of a relation (table) to exist as a value of another attribute (column) in a different (or the same) relation (table) (Chapple, 2018). The function "foreign key" is used to follow this rule.

1	id	INT	11	☐	☐	☐	AUTO_INCREMENT
2	flight_id	INT	11	☐	☑	☐	NULL
3	ticket_id	INT	11	☐	☑	☐	NULL
4	t_seat_no	VARCHAR	50	☐	☑	☐	NULL
5	t_seat_type	VARCHAR	50	☐	☑	☐	NULL
6	t_price	DOUBLE		☐	☑	☐	NULL
7	t_baggage_requirement	VARCHAR	50	☐	☑	☐	NULL

Fig. 32. Example of Foreign Keys among Tables of Database

Sql codes for foreign key:

```
CREATE TABLE `t_info` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `flight_id` INT(11) NULL DEFAULT NULL,
  `ticket_id` INT(11) NULL DEFAULT NULL,
  `t_seat_no` VARCHAR(50) NULL DEFAULT NULL,
  `t_seat_type` VARCHAR(50) NULL DEFAULT NULL,
  `t_price` DOUBLE NULL DEFAULT NULL,
  `t_baggage_requirement` VARCHAR(50) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `FK_t_info_ticket` (`ticket_id`),
  INDEX `FK_t_info_flight` (`flight_id`),
  CONSTRAINT `FK_t_info_flight` FOREIGN KEY (`flight_id`) REFERENCES `flight` (`flight_id`),
  CONSTRAINT `FK_t_info_ticket` FOREIGN KEY (`ticket_id`) REFERENCES `ticket` (`ticket_id`)
)
COLLATE='latin1_swedish_ci'
ENGINE=InnoDB
AUTO_INCREMENT=14
;
```

Domain Integrity. Domain integrity specifies that all columns in a relational database must be declared upon a defined domain (En.wikipedia.org, 2018). The accuracy of data would be broken without domain integrity rules.

1	flight_id	INT	11	☐	☐	☐	AUTO_INCREMENT
2	f_no	VARCHAR	50	☐	☑	☐	NULL
3	aircraft_type	VARCHAR	50	☐	☑	☐	NULL
4	f_level	VARCHAR	50	☐	☑	☐	NULL
5	f_date	DATE		☐	☑	☐	NULL
6	f_duration	VARCHAR	50	☐	☑	☐	NULL
7	f_arr_punct_rate	DOUBLE		☐	☑	☐	NULL
8	f_avg_delay_time	VARCHAR	50	☐	☑	☐	NULL
9	f_ticket_price	DOUBLE		☐	☑	☐	NULL
10	f_ticket_discount	DOUBLE		☐	☑	☐	NULL
11	f_seat_amount	INT	11	☐	☑	☐	NULL
12	f_seat_avail_num	INT	11	☐	☑	☐	NULL
13	f_origin_cityid	INT	11	☐	☑	☐	NULL
14	f_board_terminal	VARCHAR	50	☐	☑	☐	NULL
15	f_board_gate	VARCHAR	50	☐	☑	☐	--
16	f_board_time	DATETIME		☐	☑	☐	NULL
17	f_dept_time	DATETIME		☐	☑	☐	NULL

Fig. 33. Domain Rules among Tables of Database

Code about setting domain rules:

```
CREATE TABLE `flight` (
  `flight_id` INT(11) NOT NULL AUTO_INCREMENT,
  `f_no` VARCHAR(50) NULL DEFAULT NULL,
  `aircraft_type` VARCHAR(50) NULL DEFAULT NULL,
  `f_level` VARCHAR(50) NULL DEFAULT NULL,
  `f_date` DATE NULL DEFAULT NULL,
  `f_duration` VARCHAR(50) NULL DEFAULT NULL,
  `f_arr_punct_rate` DOUBLE NULL DEFAULT NULL,
  `f_avg_delay_time` VARCHAR(50) NULL DEFAULT NULL,
  `f_ticket_price` DOUBLE NULL DEFAULT NULL,
  `f_ticket_discount` DOUBLE NULL DEFAULT NULL,
  `f_seat_amount` INT(11) NULL DEFAULT NULL,
  `f_seat_avail_num` INT(11) NULL DEFAULT NULL,
  `f_origin_cityid` INT(11) NULL DEFAULT NULL,
  `f_board_terminal` VARCHAR(50) NULL DEFAULT NULL,
  `f_board_gate` VARCHAR(50) NULL DEFAULT '--',
  `f_board_time` DATETIME NULL DEFAULT NULL,
  `f_dept_time` DATETIME NULL DEFAULT NULL,
```

Database Recovery. Recovery should protect the database and associated users from unnecessary problems and avoid or reduce the possibility of having to duplicate work manually (Docs.oracle.com, 2018). Recovery processes vary depending on the type of failure that occurred, the structures affected, and the type of recovery. There are three methods to maintain the stability of database and execute recovery rule, including transaction, system recovery and roll back.

Transaction. Transaction is serious of database commands with clear semantics. Transactions are completed by COMMIT or ROLLBACK SQL statements, which indicate a transaction's beginning or end. The ACID acronym defines the properties of a database transaction. In this project, the process of payment is the step following this rule, the code is presented below.

```
1 begin
2       begin tran Tran_Money
3         DECLARE @tran_error int ;
4       set @tran_error =0;
5       begin try
6         insert into MoneyRecord values(@U_Id,@MR_money,@MR_time,@MR_state ,cast (@MR_depict as varchar),@MR_typeId,@MR_number
7         set @tran_error =@tran_error+@@ERROR;
8         select @@IDENTITY as 'identity';
9         set @tran_error = @tran_error+@@ERROR;
10        insert into CZRecord values(@@IDENTITY);
11         set @tran_error = @tran_error+@@ERROR;
12         update Capital set C_state="payed" where C_Id = (select C_Id from UserInfo where U_Id =@U_Id)
13        set @tran_error = @tran_error+@@ERROR;
14      end try
15     begin catch
16         print 'something wrong'+convert(varchar,error_number())+','+error_message()
17         set @tran_error=@tran_error+1;
18     end catch
19 if(@tran_error>0)
20     begin
21      ROLLBACK TRAN;
22      Print 'pay fail'
23      end
24 else
25    begin
26     COMMIT TRAN;
27     Print 'pay success'
28     end
29 end
```

System Recovery. System recovery is through database backup built before to recovery the database to a previous version, aiming to recovery most of losing data. In this project, the operation below is the statement for database to store the backup.

```
C:\Users\100491558>mysqldump -u root -p airlines->D:BackupName.sql
```

When this database needs to be recovery, the statement need to be inputted like follows.

```
mysql>source D:\BackupName.sql
```

Roll Back. Database roll back is different from the rollback of transaction. It is based on the complete database backup and log to recovery the whole database or one of tables to a special time point, which has ability to recovery nearly all records. This operation is supported by all database, so, we can operate it in the user interface span without coding.

Concurrency. Database concurrency means two transactions read/write on the same part of database, although transactions execute correctly, results may interleave in different ways causing lost update, uncommitted dependency and inconsistent analysis (Techopedia.com, 2018). There are two technique to control concurrency of database, locking and deadlock.

Locking. Locking is a procedure used to control concurrent access to data. With the transaction, a lock may deny access to other transactions to prevent incorrect results. So, a transaction must obtain a read (shared) or a write (exclusive) lock on a data item before the corresponding database read or write operation is carried out. The code below is a simple example for lock the table flight with a S(shared) lock in the transaction until this transaction commits.

```
1 begin tran
2 select * from flight with HOLDLOCK
3 ...
4 ...
5 commit tran
```

Deadlock. Deadlock occurs when two or more transactions are in a simultaneous wait state (each holds the lock the other needs). When it happens, the system must choose one transaction as a victim and rolling it back and return an error code to the victim and leaving it up to program to handle situation.

SQL Injection. As one of the most common web hacking techniques, SQL injection is a code injection technique that might destroy database, and it is the placement of malicious code in SQL statements, via web page input (W3schools.com, 2018). This "smart" input way is based on 1=1 is always true, "=" is always true or batched SQL statements. In this project, every input needs to mention this kind of problem. Codes below is a simple example to show how this kind of technique works.
This is the code before SQL injection.

```
1  txtUserId = getRequestString("UserId");
2  txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

If "1=1" is filled in this blank place, the SQL statement will become like code below.

```
"SELECT * FROM Users WHERE UserId = 33 or 1=1;
```

In this case, the judgement will be always true. And the hacker can get all information of database in this way.
However, SQL parameters can be used to protect a web site from SQL injection. They are values that are added to an SQL query at execution time in a controlled manner. Each parameter is stated by a @ marker and set up the SQL engine to check each parameter to ensure that it is correct for its column and are treated literally, not as part of SQL to be executed. Codes following is a simple example for using parameters to protect database.

```
1  txtNam = getRequestString("PassengerName");
2  txtAdd = getRequestString("Address");
3  txtCit = getRequestString("City");
4  txtSQL = "INSERT INTO passenger (PassengerName,Address,City) Values(@0,@1,@2)";
5  db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

5 Summary

5.1 Reflection and Conclusion

To sum up, this essay presents a brief report of implementing processes and related user interfaces for booking ticket system, affairs management system and their function details clearly and completely. Requirements asked from sponsors and test users have been implemented and running accurately under control. Looking through this essay, based on feedbacks got from the lecturer in the last part of assignment in this module, the content presents the improvement of database structure. After introduce running environment of these software used during this process, this essay divides the presentation into four parts, database connection, functions for normal users and functions for administrators, and database security. This flow presents the achievements of this project and brief rationales for technique I used.

For the database, the rule of ACID is the key point I cannot ignore during the whole process to build tables and attributes. Following this rule, every table need to be normalized to 3rd normalized table to make sure there is no reliable between attributes except for the primary key and the primary key makes every record to be a unique one, which avoid harm of data redundancy in the database. Besides, E-R diagram also helps programmers to make the thoughts clear about the connection among tables. This is also an effective way to find problems and hidden danger in the database.

As an official software, the workbench of mysql is unfriendly for new users to start their stories with mysql, that is the reason why plenty of mysql assistance software appear and are used widely, such as HeidiSQL and Navicat. Nowadays, HeidiSQL is becoming a popular software to manage database instead of use mysql workbench directly. This kind of software still need to be running based on the mysql environment, but users can manage tables and data in a friendlier interface in their own national language, which is much easy to operate database effectively and efficiently.

The environment provider XAMPP is a free and open-source cross-platform web server solution stack package, which really make the process of environment creation much easier for me. The most obvious characteristic of XAMPP is the ease at which a WAMP webserver stack can be deployed and instantiated. As a development tool, XAMPP allows website designers and programmers to test their work on their own computers without any access to the Internet. In this project, XAMPP provides the control switches of Apache, MySQL and path to view php pages to let me check how it is working. However, many important security features are disabled by default in this software, but in my project, few of them need to be consider in this small case.

Based on the database which has been created, this website is created by PHP coding language to implement the function of connect database, transfer data, and present the user interfaces. In this project, php is friendly with SQL, which means that the SQL statement can be written in the php document directly and called by php functions to control data in CRUB.

5.2 Evaluation of This Project

Strength of This Project

1. All requirements and needs posted by sponsors and users have been met through developing in PHP basically. These functions make sure this system running normally under control with no bug.
2. The building of basic database in this project follows ACID rules. The database in this project covers 11 tables, every table has its primary key to ensure the unique of records. At the same time, foreign keys describe the connections among attributes of different tables to maintain the consistency and integrity of data. In addition, these tables have followed normalization rules to become 3rd normalized tables in order to control data following ACID rules much more.
3. The problem of security has been considered in this project to protect the security of data and maintain these data smooth operation from different aspects. The in-

tegrity rules maintain the database stability of the whole database. Database recovery is used to make sure the database can be recovered to a normal statement with help of log and backup. In particular, SQL injection can crack down on hacker attacks in input area. Because of these security methods, this system can be running and operating safely.

Weakness of This Project

1. Owing to some questions about the requirement document, the details of these needs are not clear enough for programmers. During the implement process, the programmer has communicated with sponsors about details for several times, but the developer cannot promise every page meets sponsors` expectation.
2. In the area of programming, this project follows the rules of Procedure Oriented Design to create the whole website, which waste much time to repeat the similar code, which has the same develop methods. At the same time, this project has poor flexibility to update to a new version because of the high workload.
3. This project is only a small case for the module of database, so, there are plenty of reality factors have not been considered in this project, including the process and way of payment, the dynamic condition of flights and so on. If we want to learn more about the database, we need to practise it in a much bigger formal case with massive data to experience the security and data management.

References

1. Lecture materials Week 6-8
2. Martin, J. (1981). Managing the data base environment.
3. Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.
4. Peltz, C. (2003). Web services orchestration and choreography. Computer, 36(10), 46-52.
5. Storey, V. C., & Song, I. Y. (2017). Big data technologies and management: What conceptual modeling can do. Data & Knowledge Engineering, 108, 50-67.
6. Patinge, S. G., & Talmale, M. G. (2017). A Review: Runtime Environment for Addition Services for TOMCAT and MYSQL using JAVA Swing. International Journal on Recent and Innovation Trends in Computing and Communication, 5(3), 130-132.
7. Becker, A. (2018). HeidiSQL - MySQL, MSSQL and PostgreSQL made easy. [online] Heidisql.com. Available at: https://www.heidisql.com/#featurelist [Accessed 12 Dec. 2018].
8. Sqa.org.uk. (2018). Higher National Computing: E-Learning Materials. Entity Integrity. [online] Available at: https://www.sqa.org.uk/e-learning/MDBS01CD/page_39.htm [Accessed 12 Dec. 2018].
9. Chapple, M. (2018). The Role of Referential Integrity in Your Database. [online] Lifewire. Available at: https://www.lifewire.com/referential-integrity-definition-1019181 [Accessed 12 Dec. 2018].
10. En.wikipedia.org. (2018). Data integrity. [online] Available at: https://en.wikipedia.org/wiki/Data_integrity [Accessed 12 Dec. 2018].
11. Techopedia.com. (2018). What is Data Integrity in Databases? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/811/data-integrity-databases [Accessed 12 Dec. 2018].
12. Docs.oracle.com. (2018). Database Recovery. [online] Available at: https://docs.oracle.com/cd/A87860_01/doc/server.817/a76965/c28recov.htm#2804 [Accessed 12 Dec. 2018].
13. Techopedia.com. (2018). What does Concurrency mean? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/27385/concurrency-databases [Accessed 12 Dec. 2018].
14. W3schools.com. (2018). SQL Injection. [online] Available at: https://www.w3schools.com/sql/sql_injection.asp [Accessed 12 Dec. 2018].