|  |
| --- |
| National University of singapore  nus_logo.gif |
| CKY Restaurant  Online Booking System |
| CS2102 Database Systems – Project Report |
|  |
|  |
| **YEAR**  **14/15 Semester 1** |

|  |
| --- |
|  |

GROUP 62  
TEAM MEMBERS:

1. SIEW KAE YEE A0098127X
2. Lui Yu Yao A0098155W
3. Chen Tze Cheng A0112092W

**Introduction**

In this project, our group had built up an online booking system for CKY Restaurant to facilitate it with providing online reservation services. This platform provides a range of services such as sign up, log in/out, make/edit/delete reservation, statistical report for administrators and etc.

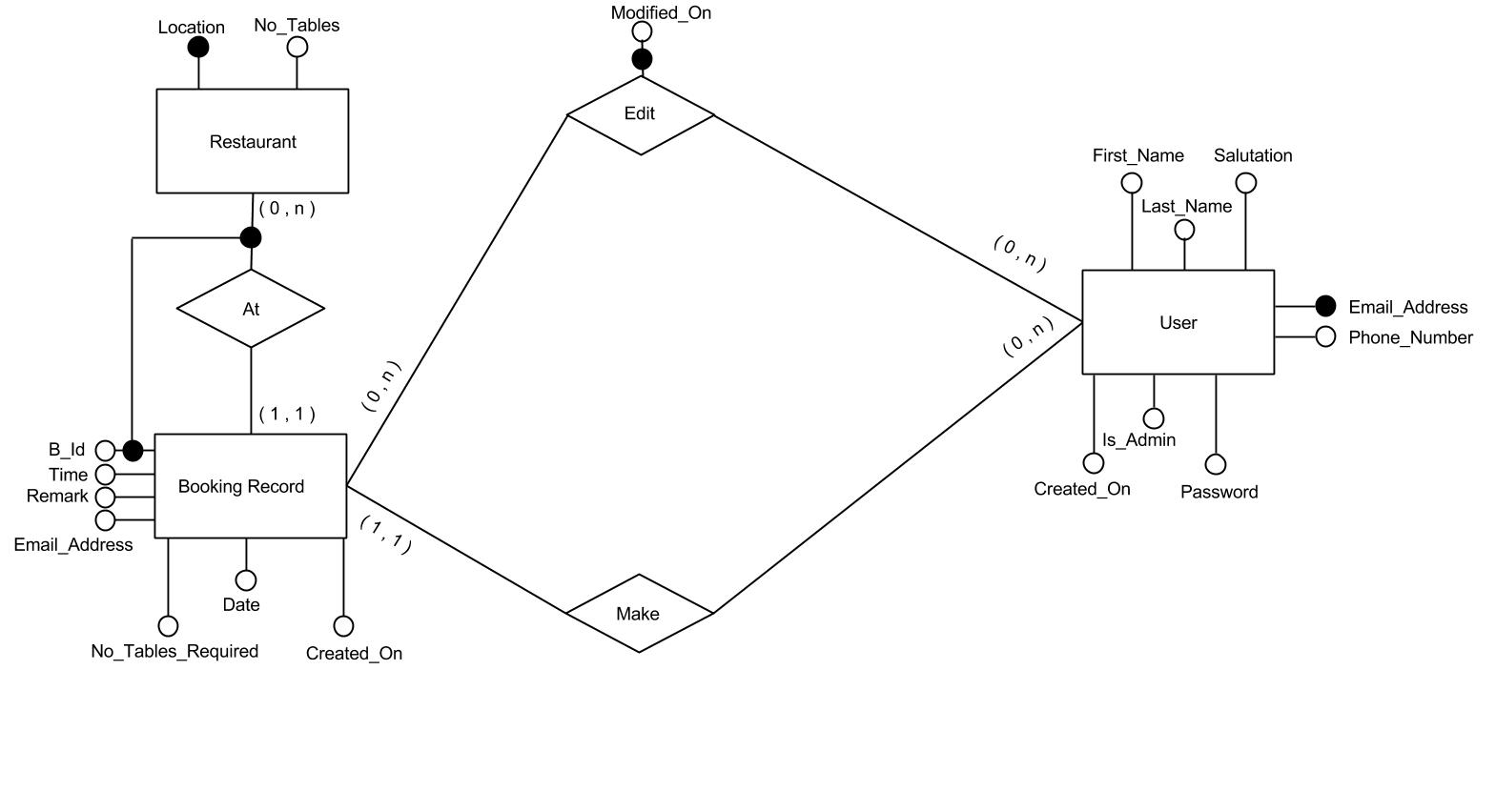
The user is required to login in order to make any reservation. A signup function is provided for user who is first timer. User who make reservation is allowed to change their reservation details at least one day before the reservation date and also subject to availability. Only Admin can cancel any reservation record, so if the user wants to cancel reservation, he/she has to contact the admin in order to do so. Each table in each of the branches of CKY Restaurant is standardised at a table of 10 people. So the user only need to make selection of how many table they need when making reservation online. More details of the system will be provided in the following sections.

**Implementation of the Online Booking System**

Our group had used **HTML** together with **Javascript/JQuery** and **CSS** in building up the user interface of this system. We used **Microsoft Web Matrix** to develop this system, in which the server is built up together with the installation of Web Matrix. Web Matrix provide a function to debug the website directly from it, so there is no need for us to build up additional web server to run the online booking system. The server side language that we used is **PHP** which is the same language as the one used by Facebook. We had decided to use **MySQL** as our database management system due to its easy accessibility and also because it is a freeware.

**Database Management System**

The following is the ER diagram corresponding to our database management system.



The restaurant table is used to store the details of the restaurant, such as the location of the restaurant and also the total number of tables available in the restaurant.

The booking record table is used to store the details of each record that is made by the user. It contains details such as the user’s email address, number of tables required, the date and time of the booking.

On the other hand, the user table is used to store the particulars of the user. When user is trying to make new reservation, all the fields that contain details in this table will be auto-filled up.

Lastly, the edit table contains record that is used to identify the editing of any reservation record.

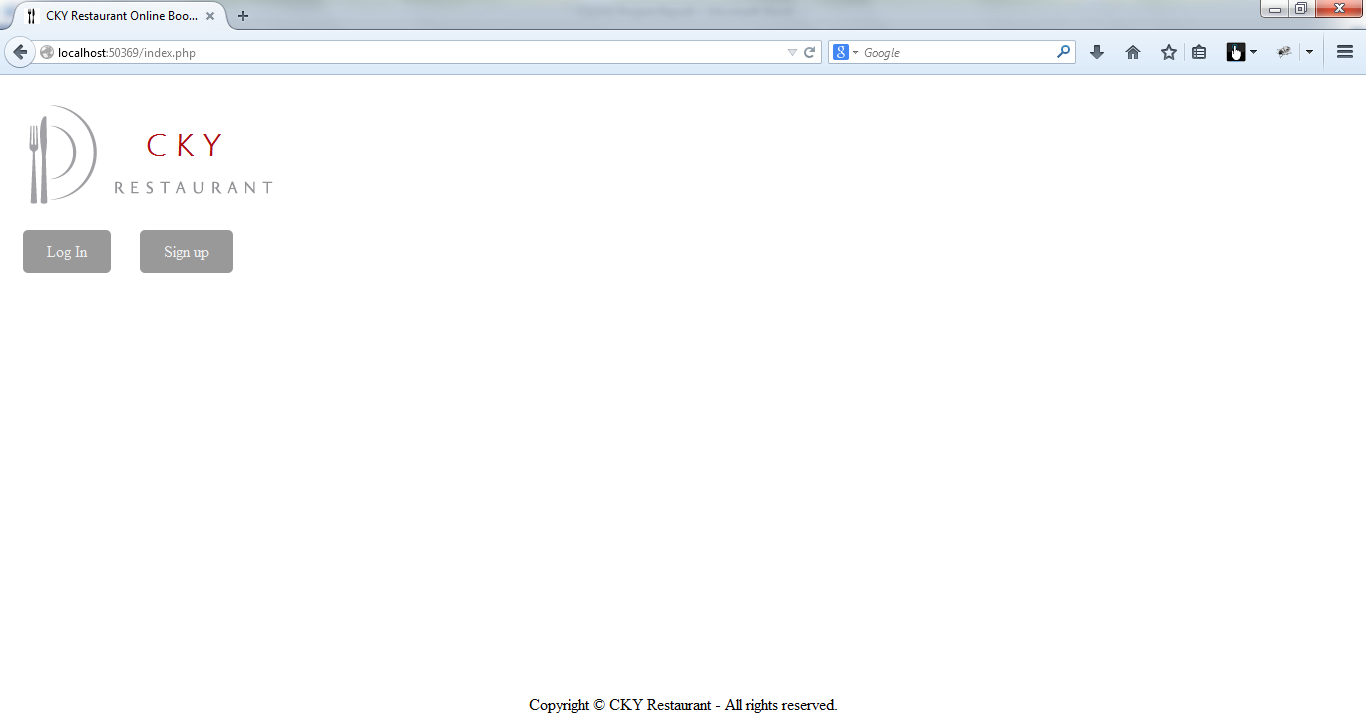
The Data Definition Language (DDL) that correspond to each component are as follow:

1. Restaurant   
   CREATE TABLE restaurant(  
   Location VARCHAR(255) NOT NULL,  
   No\_Tables INTEGER NOT NULL DEFAULT 10,  
   PRIMARY KEY (Location),  
   );
2. Booking\_Record  
   CREATE TABLE booking\_record (  
   B\_Id INTEGER NOT NULL AUTO\_INCREMENT,   
   Time INT NOT NULL,   
   Date DATE NOT NULL,  
   No\_Tables\_Required INTEGER NOT NULL DEFAULT 0,  
   Location VARCHAR(255) NOT NULL,  
   Remark VARCHAR(255),  
   PRIMARY KEY (B\_Id, Location),  
   FOREIGN KEY(Location)   
    REFERENCES restaurant(Location)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
   );
3. User  
   CREATE TABLE user(  
   Email\_Address VARCHAR(255) NOT NULL,  
   First\_Name VARCHAR(255),  
   Last\_Name VARCHAR(255),  
   Salutation VARCHAR(255),  
   Password VARCHAR(255),  
   Phone\_Number INTEGER,  
   Is\_Admin BIT NOT NULL,  
   Created\_On DATETIME NOT NULL,  
   PRIMARY KEY (Email\_Address)  
   );
4. Edit  
   CREATE TABLE edit (  
   B\_Id INTEGER NOT NULL,  
   Email\_Address VARCHAR(255) NOT NULL,  
   Modified\_On DATETIME NOT NULL,  
   PRIMARY KEY (B\_Id, Email\_Address , Modified\_On),  
   FOREIGN KEY (B\_Id)  
    REFERENCES booking\_record (B\_Id),  
   FOREIGN KEY (Email\_Address)  
    REFERENCES user(Email\_Address)  
   );

**Chapter 1 Sign up function and Log in function**

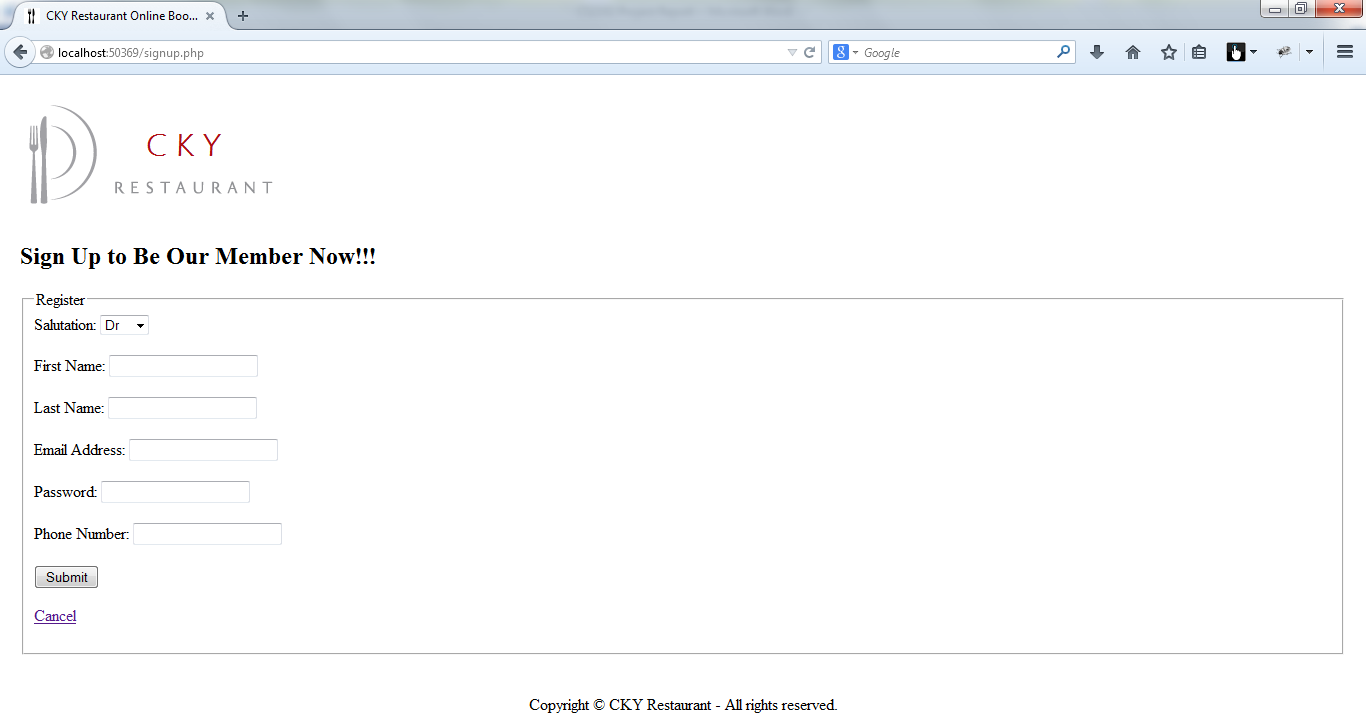
**1.1 Sign up**

Figure 1.1.1 Interface of Our Home Page (index.php)



As shown in our home page, for customers, we have two options for them. They can either sign up or log in (using their previously registered account).

Figure 1.1.2 Sign up page for first time user (signup.php)



**First layer restriction:**

All fields are to be filled up. These restrictions are enforced by Javascript:

1. Email Address is checked to ensure it is in correct form by using REGEX.

Var regex = /^([a-zA-Z0-9\_.+-])+\@(([a-zA-Z0-9-])+\.)+([a-zA-Z0-9]{2,4})+$/

1. Password needs to be at least 6 characters long.
2. Phone number has to be exactly 8 digits

**Second layer restriction:**

After the form has passed first layer restriction, our system will check the email provided by the user to see if our database already contains that email address. If so, the registration will fail and subsequent error message will be shown to the user like so “Email Address has been used by another account. Please provide another email address.”.

A snippet of PHP code showing how the checking is done by constructing and executing an SQL SELECT query. Only if our database does not contain the provided email address, do we confirm the sign up is successful and insert a new tuple into the user table. That being said, Email Address is the primary key of user table.

$email = $\_POST['email'];

$checkQuery ="SELECT \* FROM user where Email\_Address = ?";

$chkStatement = $databaseConnection -> prepare($checkQuery);

$chkStatement -> bind\_param('s', $email);

$chkStatement -> execute();

$chkStatement -> store\_result();

if($chkStatement -> num\_rows > 0){

echo "Email Address has been used by another account. <br>";

echo "Please provide another email address.";

}else{

$query = "INSERT INTO user(Email\_Address, First\_Name, Last\_Name, Salutation, Password, Phone\_Number, Created\_On) VALUES (?, ?, ?, ?, ?, ?, ?) ";

$statement = $databaseConnection -> prepare($query);

$statement -> bind\_param('sssssss', $email, $fName, $lName, $salutation, $password, $phoneNum, $createdOn);

$statement ->execute();

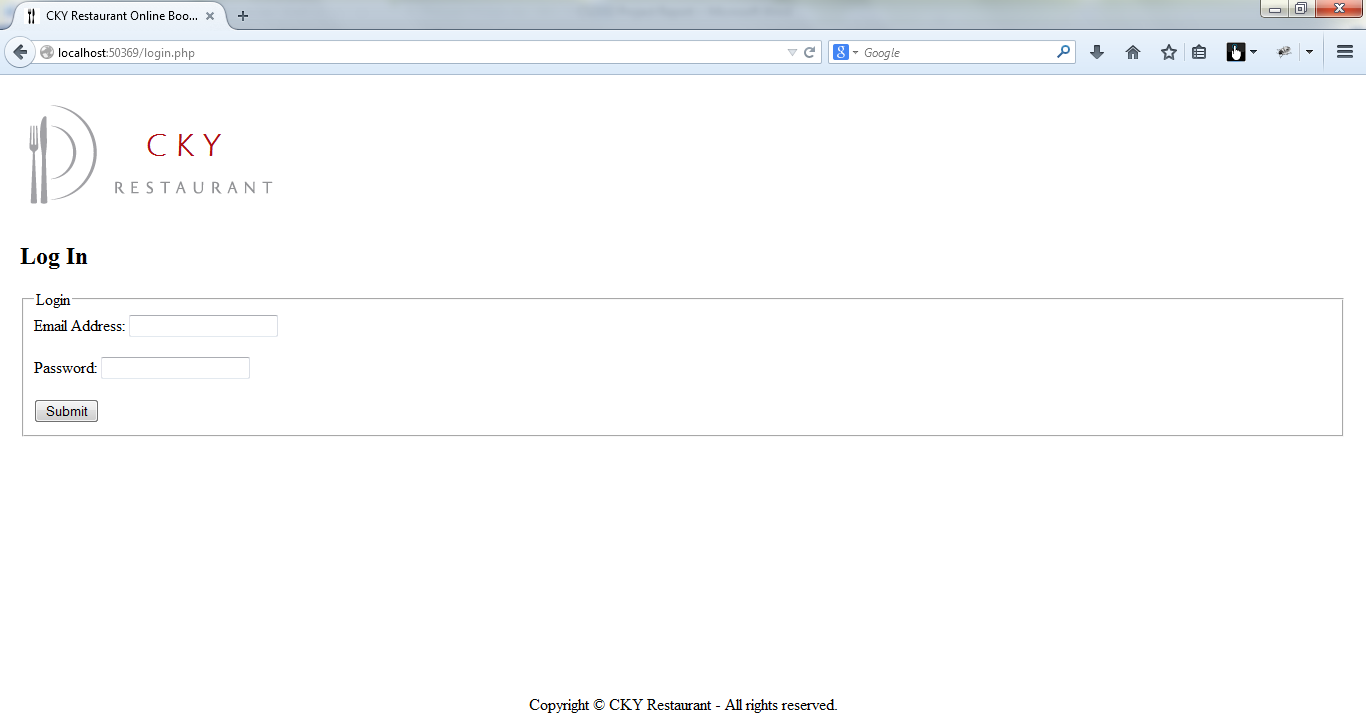
$statement ->store\_result();

}

A successful sign up will redirect user into the homepage.

**1.2 Log in**

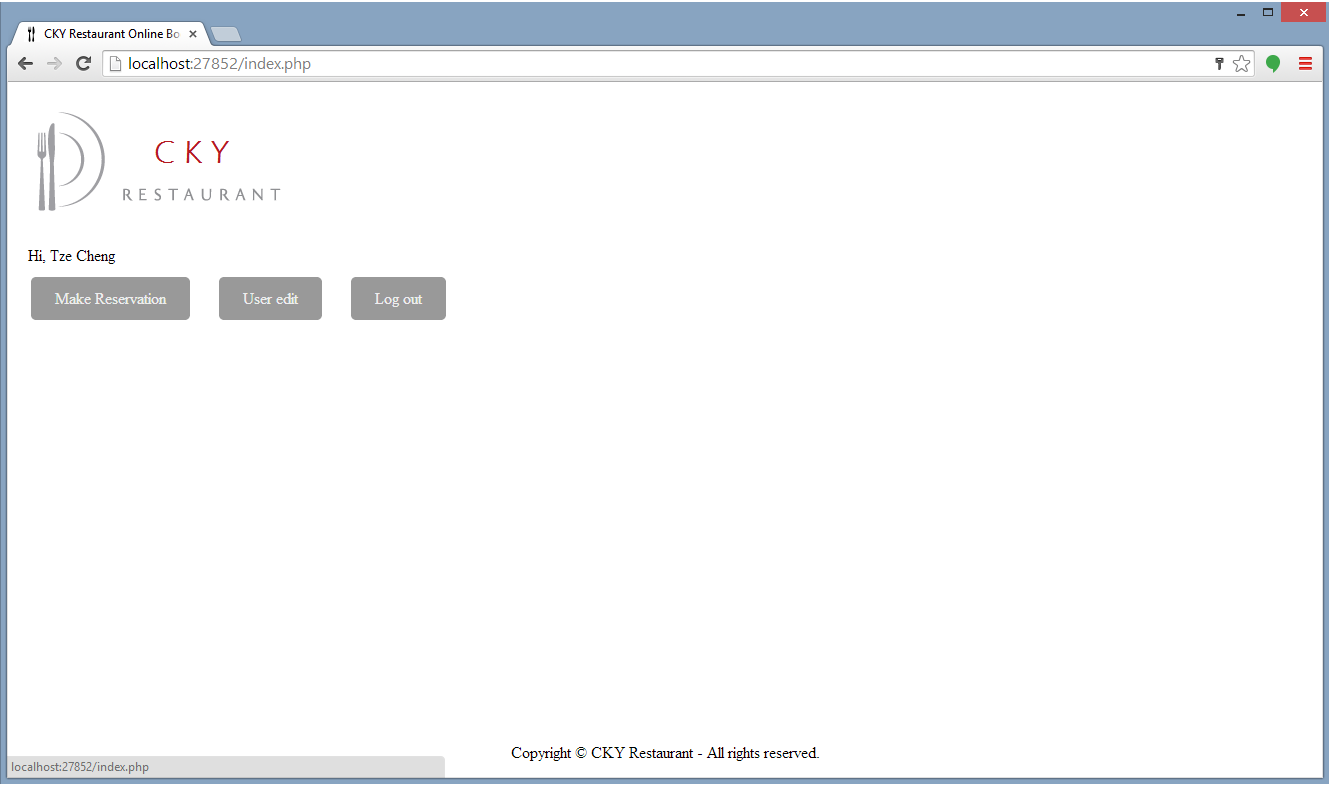
**Figure 1.2.1 log in page for registered user (login.php)**



A registered user who wishes to make reservation can go to log in page to fill in two fields, namely Email Address and Password. Error message will be thrown to screen if either one of the field does not match the other, like so “Email Address/ Password is incorrect!!”.

A successful log in will redirect registered user into the homepage and show their last name on homepage.

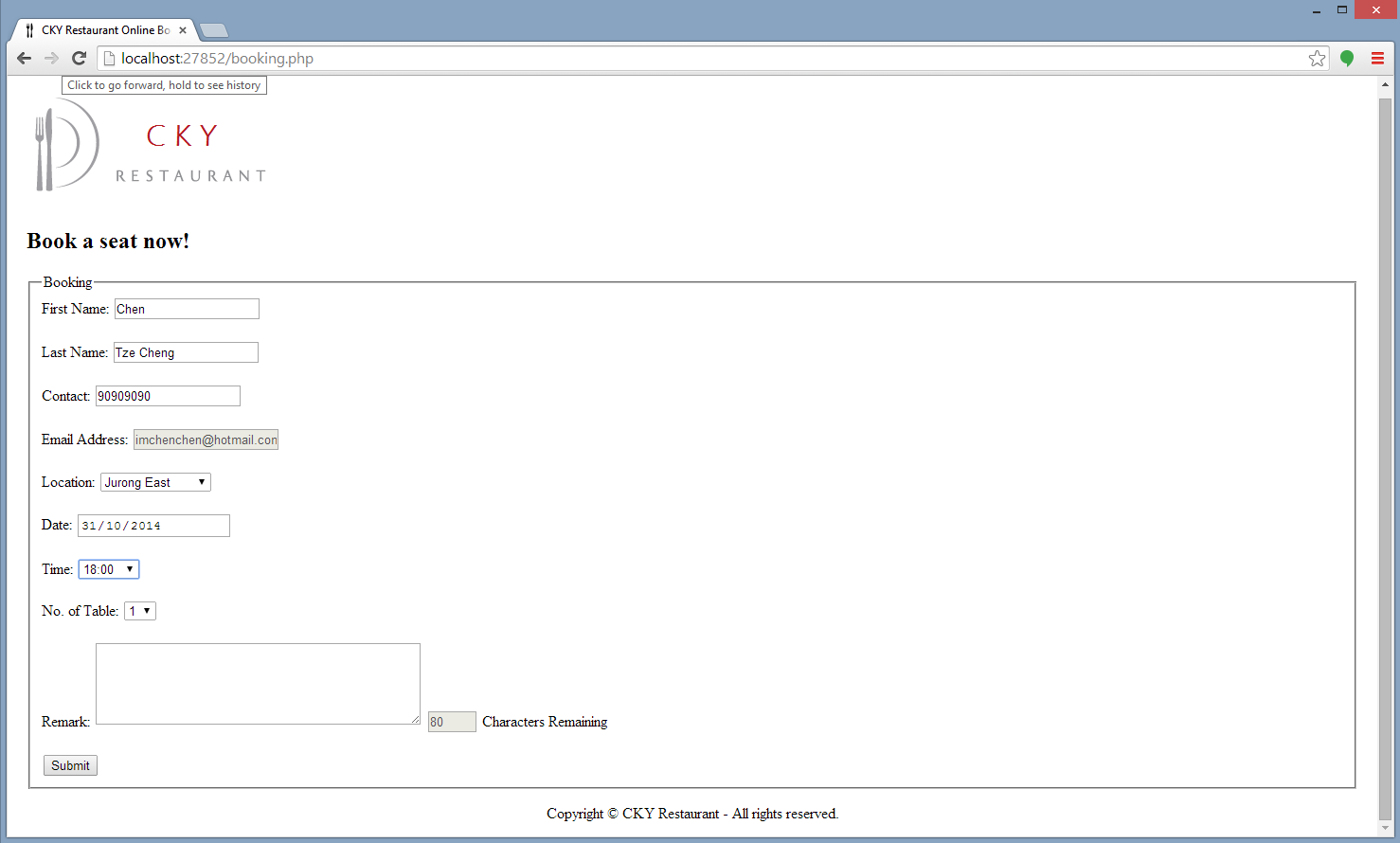
**Figure 1.2.2 Home page after log in**



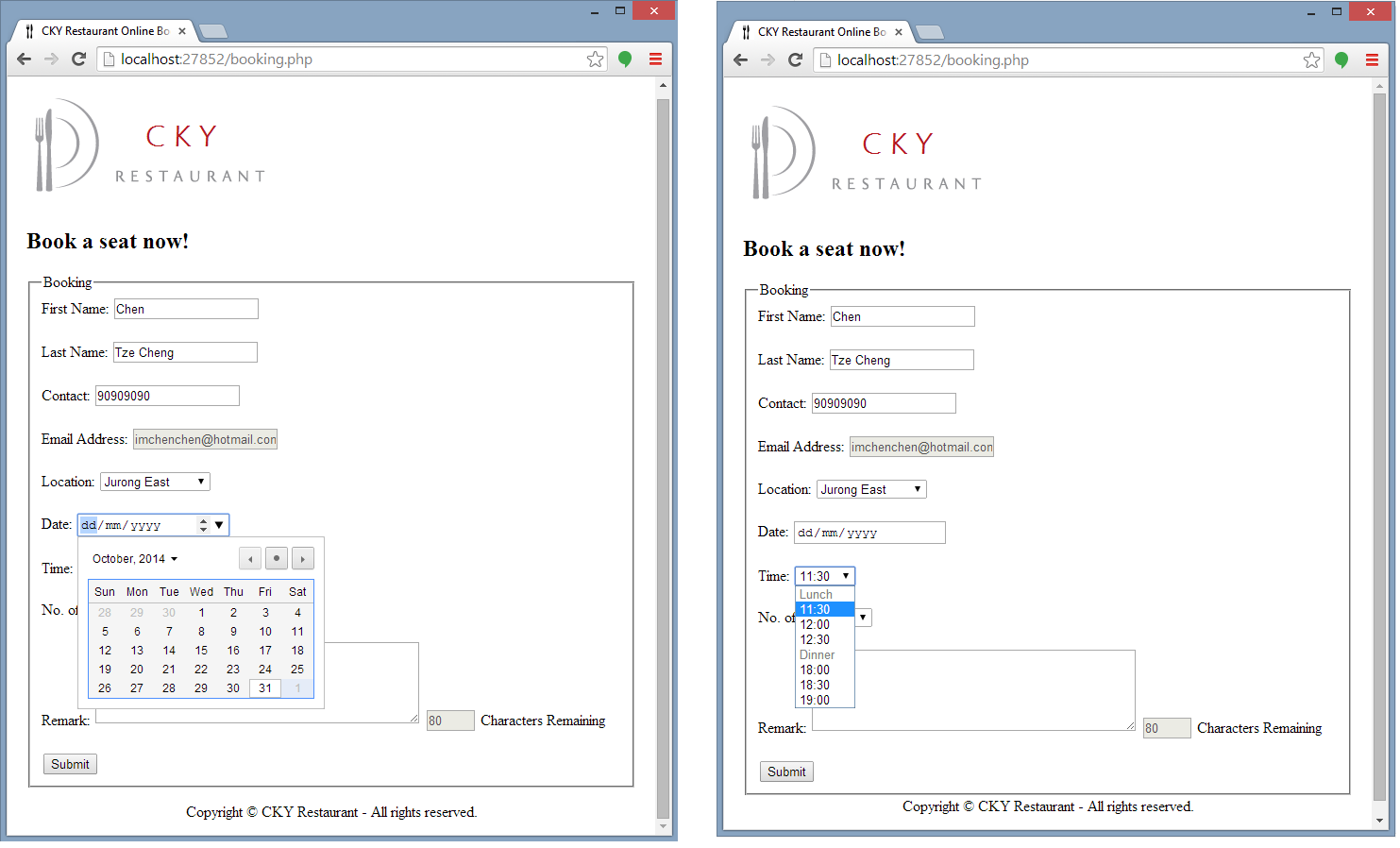
**Chapter 2: Making reservation and User Edit**

**2.1 Making reservation**

After user logged in, user can go to Make Reservation page to fill in the details of the reservation. The user’s personal details such as First Name, Last Name, Email Address and contact number will be automatically pre-filled in into fields for user’s convenience. This pre-fill is achieved by selecting user’s details from MYSQL database using the Email Address the user used to log in.

**Figure 2.1.1 Automatic pre-filled fields of booking page. (booking.php)**

The only information left for user to fill in are the location, date and time they prefer, and number of table they want to reserve. If the user has any special requirements, there is also a Remark field for him to fill in.

**Figure 2.1.2 User can select date and time via calendar and drop down list respectively.** 

If all the fields are entered correctly, the only reason of failing the reservation is that the particular Restaurant has no sufficient tables for the user to reserve. This checking is accomplished by summing the tables that have been reserved at lunch time (if the user selects one of the lunch time, i.e. 11:30, 12:00 or 12:30) and examining if the sum plus the tables user entered will exceed the total tables that the particular Restaurant has. We treat 11:30, 12:00 and 12:30 as lunch time and 18:00, 18:30, 19:00 as dinner time.

If ($time == 1130 || $time == 1200 || $time == 1230) {

$checkBookedTableQuery = "SELECT SUM(No\_Table) FROM booking\_record WHERE (Time=1130 or Time=1200 or Time=1230) AND location =? AND Date =?";

} else if ($time == 1800 || $time == 1830 || $time == 1900) {

$checkBookedTableQuery = "SELECT SUM(No\_Table) FROM booking\_record WHERE (Time=1800 or Time=1830 or Time=1900) AND location =? AND Date =?";

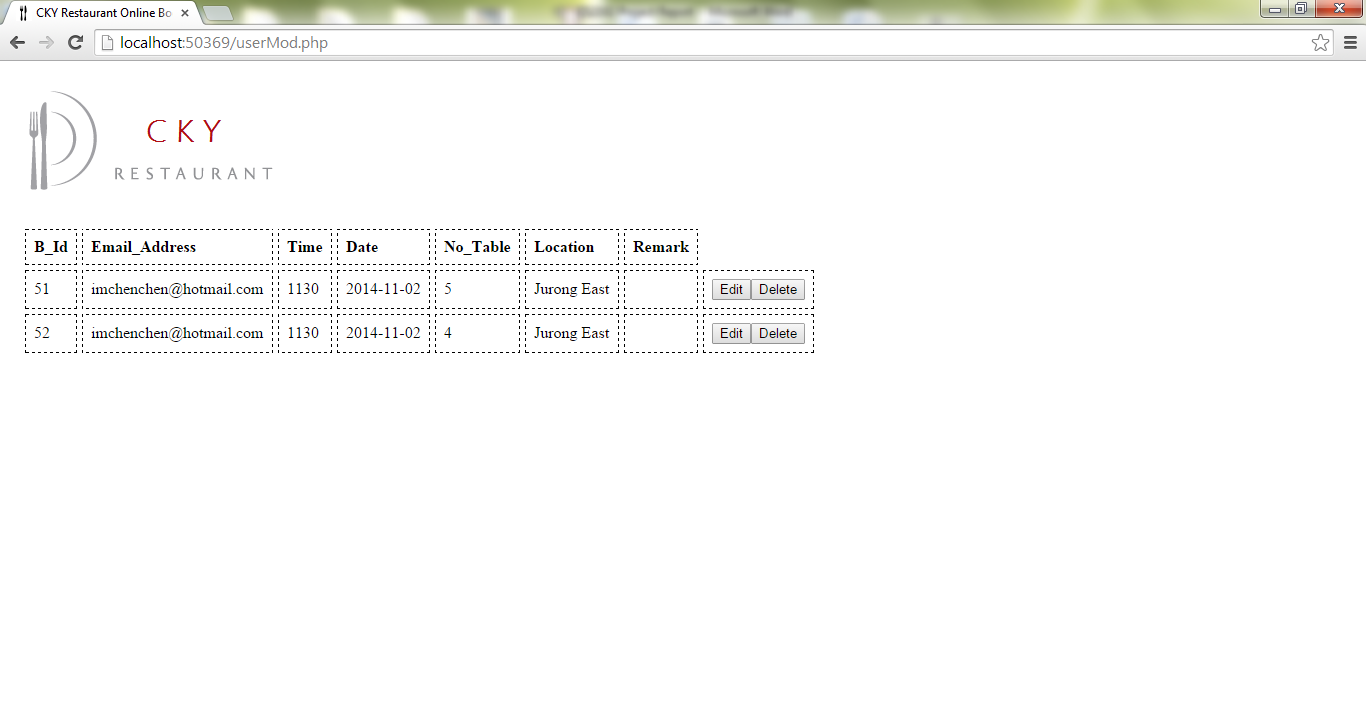
}

Once the booking succeeds, user will be redirect to bookingsuccess.php which displays “Booking Success! Redirecting to Main Page in 3 seconds.”. After three seconds, user will be redirect again to homepage.

If the booking fails, on the booking.php itself, statement like “Sorry, we left with 1 available table(s) in this location.” will be displayed.

**2.2 Editing reservation**

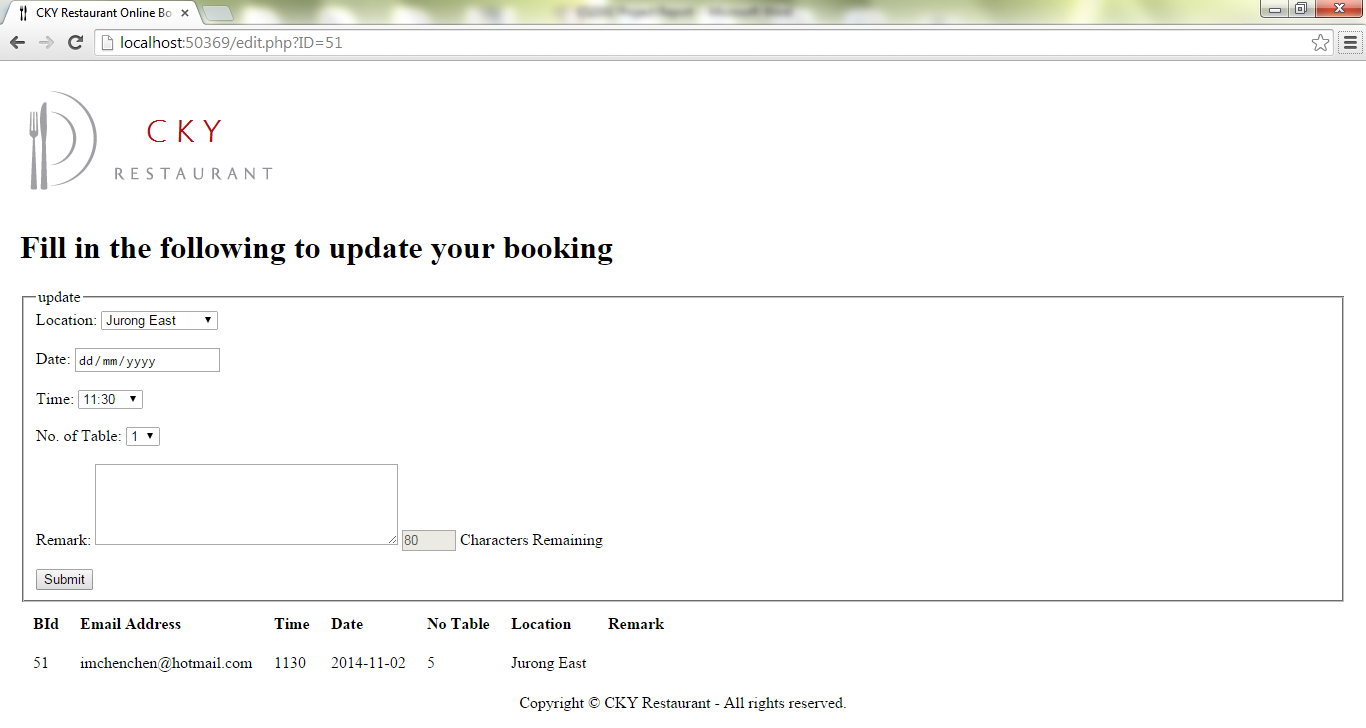
**Fig 2.2.1 A sample user edit page (userMod.php)**



On the homepage, user can go to user edit page (userMod.php), inspect all the bookings he has made, and make changes if he wants to. The changes can be Edit or Delete. However, reservations with date before the current date will not be displayed. The select query is as follows:

$query = "SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Email\_Address = ? AND Date > NOW() ";

**Fig 2.2.2 A sample edit page (edit.php?ID=51)**



If Edit button is clicked, user will be redirected to edit.php?ID=51 (51 can vary depending on which reservation the user clicked to edit. In fact, it is the booking id of a particular reservation, we use that to fetch details of that reservation on edit.php) to make changes accordingly. Previous booking details will be shown at the bottom. However, user can’t edit the booking by just entering edit.php?ID=51 if the booking record doesn’t belong to the user and the user is not an admin. We checked that by retrieving the booking record’s email address and compare it to the user’s email.

if($isAdmin==0){

if(!($r\_email == $email)){

header("Location: editFail\_authorized.php");

}

}

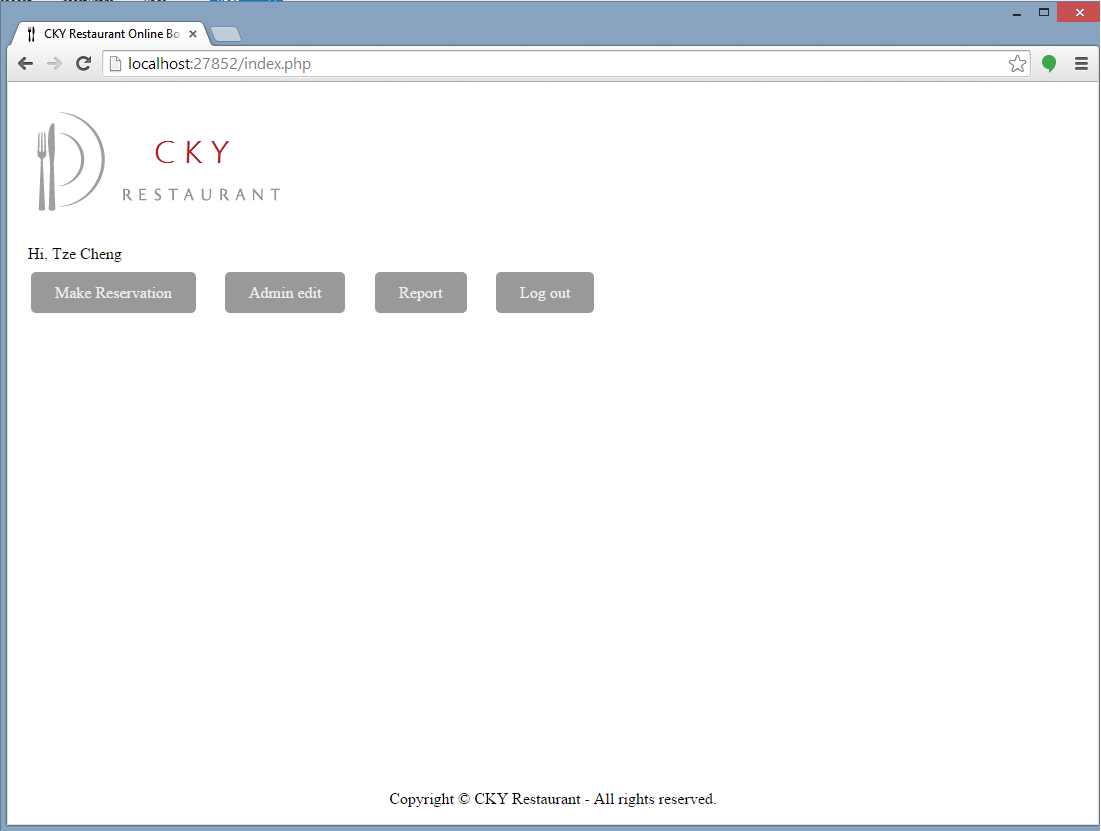
If Delete button is clicked, user will be prompted a warning message to confirm whether he really meant to cancel the reservation. If true, the reservation will be deleted permanently from booking\_record table. Otherwise, nothing changes. We check the user’s permission to delete same as the method mentioned above.

**Chapter 3: Administrator**

**3.1 Admin user**

Admin user can only be set on the database itself with the Is\_Admin bit turned to 1. After an admin has logged in, the index page will show a different main page for admin. Make Reservation button and Logout button will still be shown. However, instead of User edit button, admin gets an Admin edit button. Additionally, a Report button is added for administration purposes.

**Fig 3.1.1 Main page of admin after logged in**



**3.2 Admin Edit**

In Admin edit, admin can either search for booking records by location or email address, or both. Sample SQL code is as follows:

if($emailSubmitted != "" && $locationSubmitted != ""){

$query = "SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Email\_Address = ? AND Location = ? ORDER BY DATE, TIME";

$statement = $databaseConnection -> prepare($query);

$statement -> bind\_param('ss', $emailSubmitted, $locationSubmitted);

}else if($emailSubmitted != "" && $locationSubmitted == ""){

$query = "SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Email\_Address = ? ORDER BY DATE, TIME";

$statement = $databaseConnection -> prepare($query);

$statement -> bind\_param('s', $emailSubmitted);

}else if($emailSubmitted == "" && $locationSubmitted != ""){

$query = "SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Location = ? ORDER BY DATE, TIME";

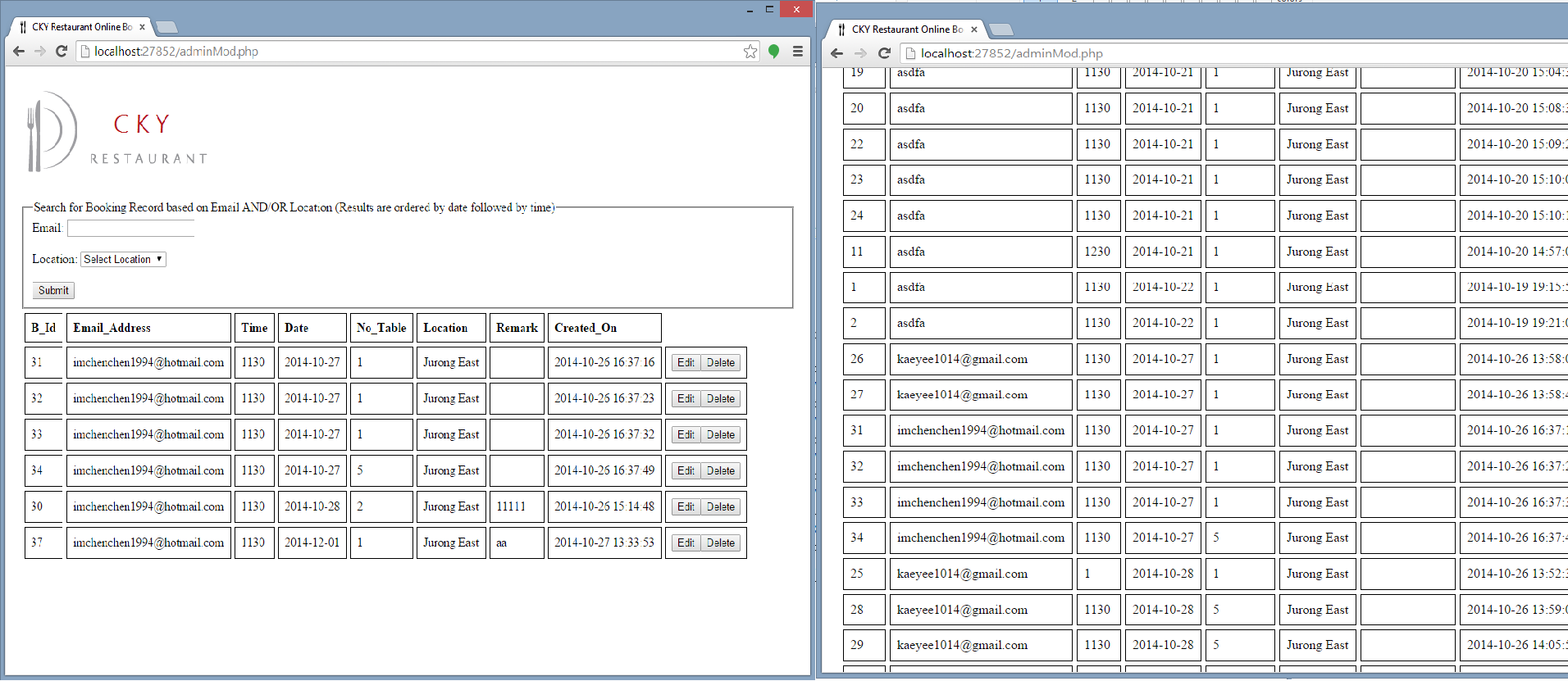
$statement = $databaseConnection -> prepare($query);

$statement -> bind\_param('s', $locationSubmitted);

}

Admin is authorized to editing any records which haven’t expired. Admin also has the permission to deleting any records.

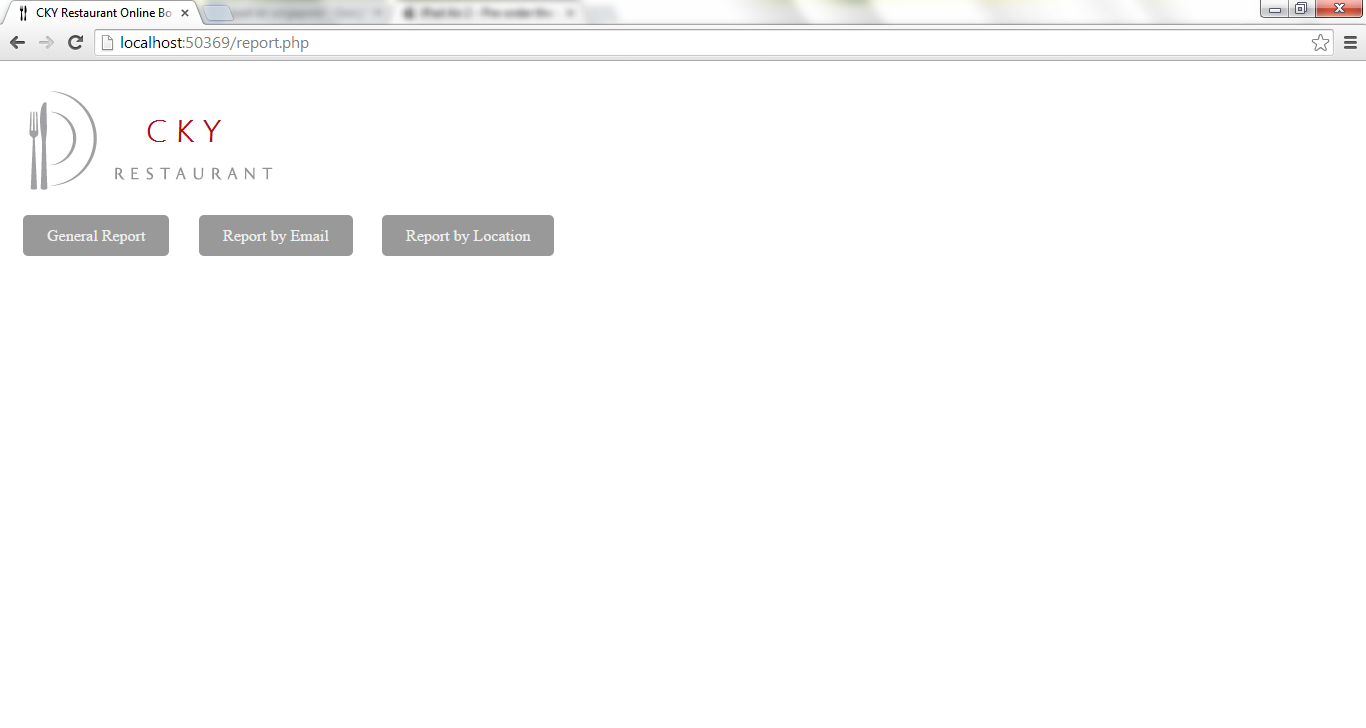
**Fig 3.2.1 Admin Edit page (AdminMod.php). Admin can search by email or location, or both.**



**3.3 Report**

In Report, there are three options, namely General Report, Report By Email and Report By Location. These reports provide important statistical facts for admin.

Fig 3.3.1 Report page (report.php) showing three kinds of reports.



3.3.1 General Report

General Report query:

$query = "SELECT

A.Email\_Address,

First\_Name,

Last\_Name,

Phone\_Number,

Location,

Date,

SUM(No\_Table) As Total\_Table

FROM

booking\_record A,

user B

WHERE

A.Email\_Address = B.Email\_Address

GROUP BY

A.Email\_Address,

First\_Name,

Last\_Name,

Phone\_Number,

Location,

Date

ORDER by

A.Email\_Address;";

3.3.2 Report By Email

Report By Email query:

$query = "SELECT

A.Email\_Address,

First\_Name,

Last\_Name,

Phone\_Number,

Location,

Date,

SUM(No\_Table) As Total\_Table

FROM

booking\_record A,

user B

WHERE

A.Email\_Address = B.Email\_Address

AND

A.Email\_Address =?

GROUP BY

A.Email\_Address,

First\_Name,

Last\_Name,

Phone\_Number,

Location,

Date

ORDER by

A.Location,

A.Date;";

3.3.3 Report By Location

Report By Location query:

$query = "SELECT

Date,

Time,

SUM(No\_Table) As Total\_Table

FROM

booking\_record

WHERE

Location =?

GROUP BY

Date,

Time

ORDER by

Time,

Date;";