|  |
| --- |
| 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

**CONTENT**

**Chapter 1: Overview**

**1.1 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.

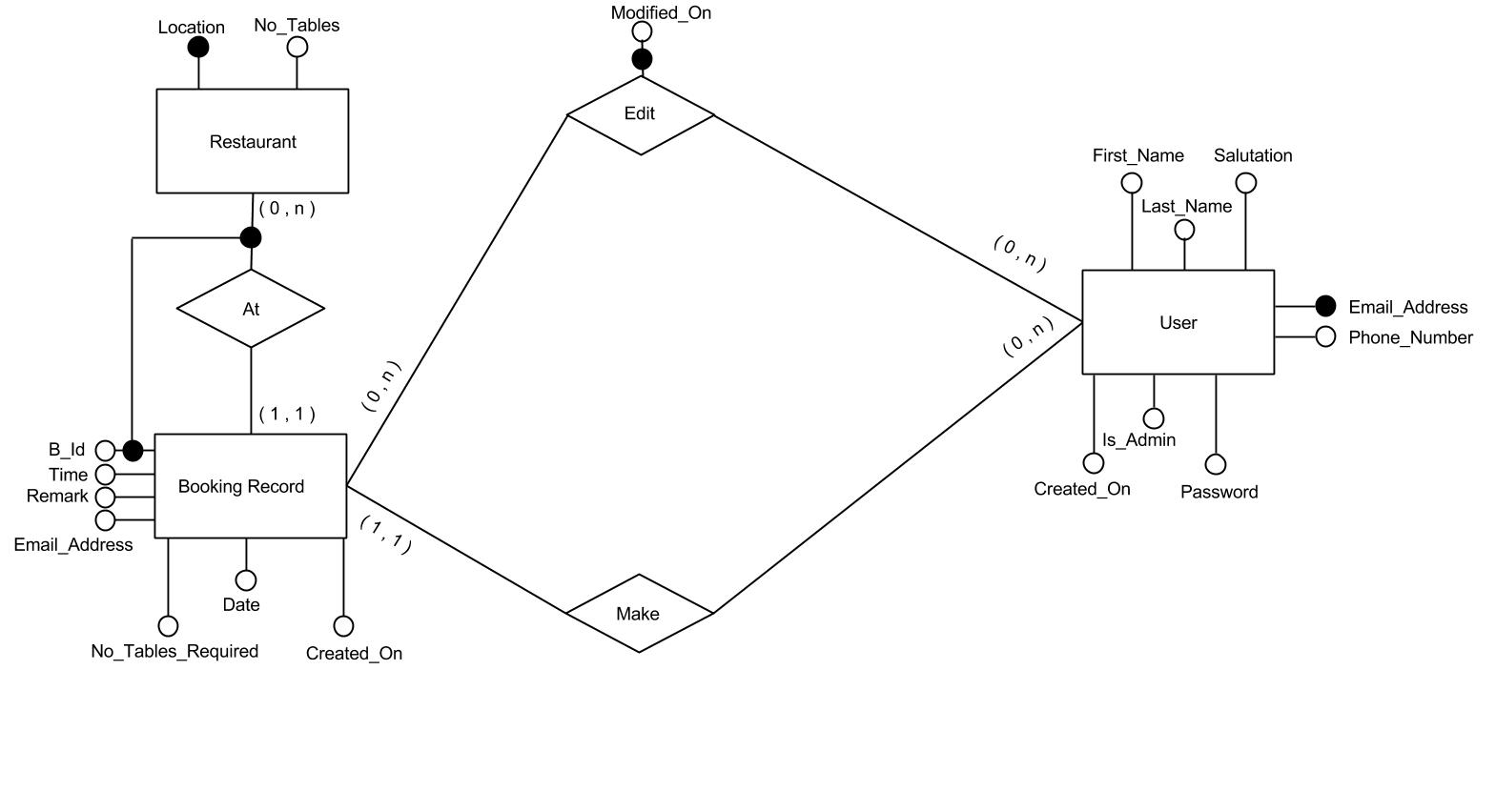
The user is required to login in order to make any reservation. A signup function is provided for user who is first timer. User who makes reservation is allowed to change their reservation details before the reservation date. However, the changes are subject to availability. Each table in each of the branches of CKY Restaurant is standardised at a table of 10 people. So the user only needs to decide how many table(s) he needs when making reservation online. More details of the system will be explained in the following sections.

**1.2 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. On the back end, 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 provides 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. We used **PHP** as our server side scripting language and **MySQL** as our database management system because it is easy to be accessed and free.

**1.3 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.

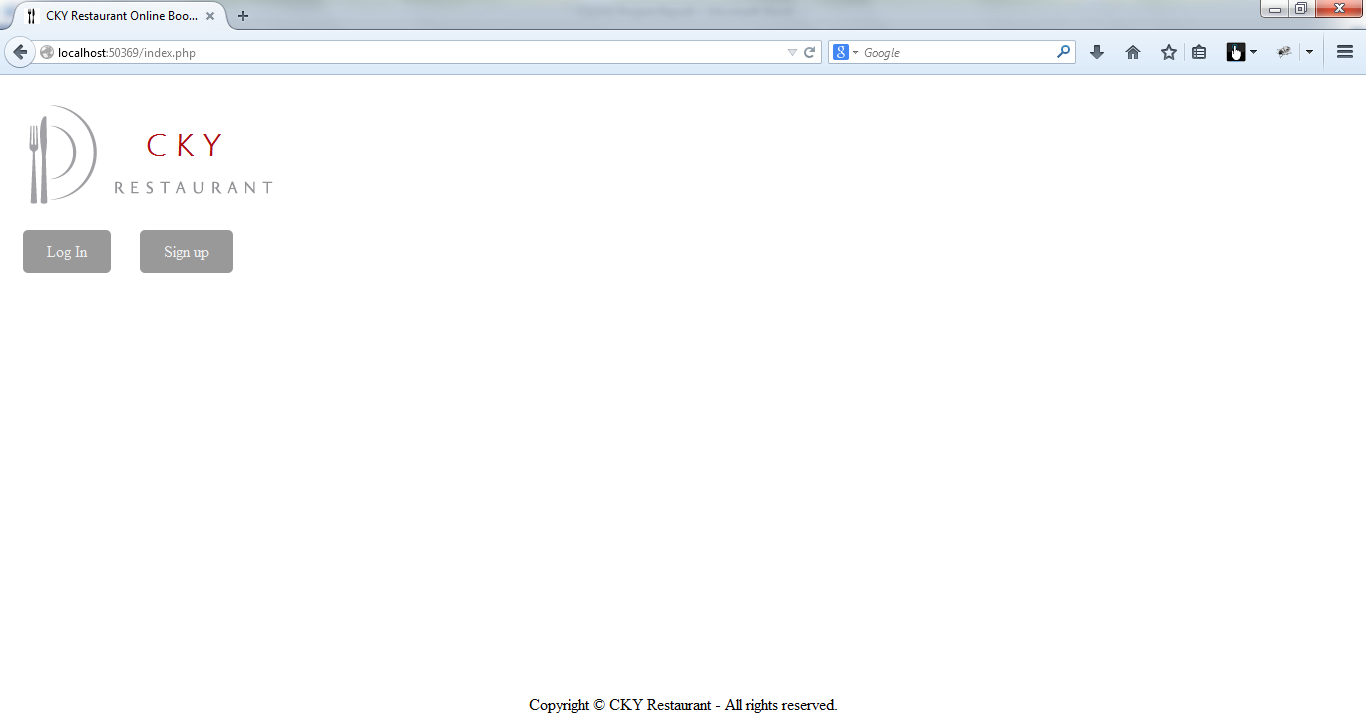
The Data Definition Language (DDL) that correspond to each component are as follow:

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 2: Sign up function and Log in function**

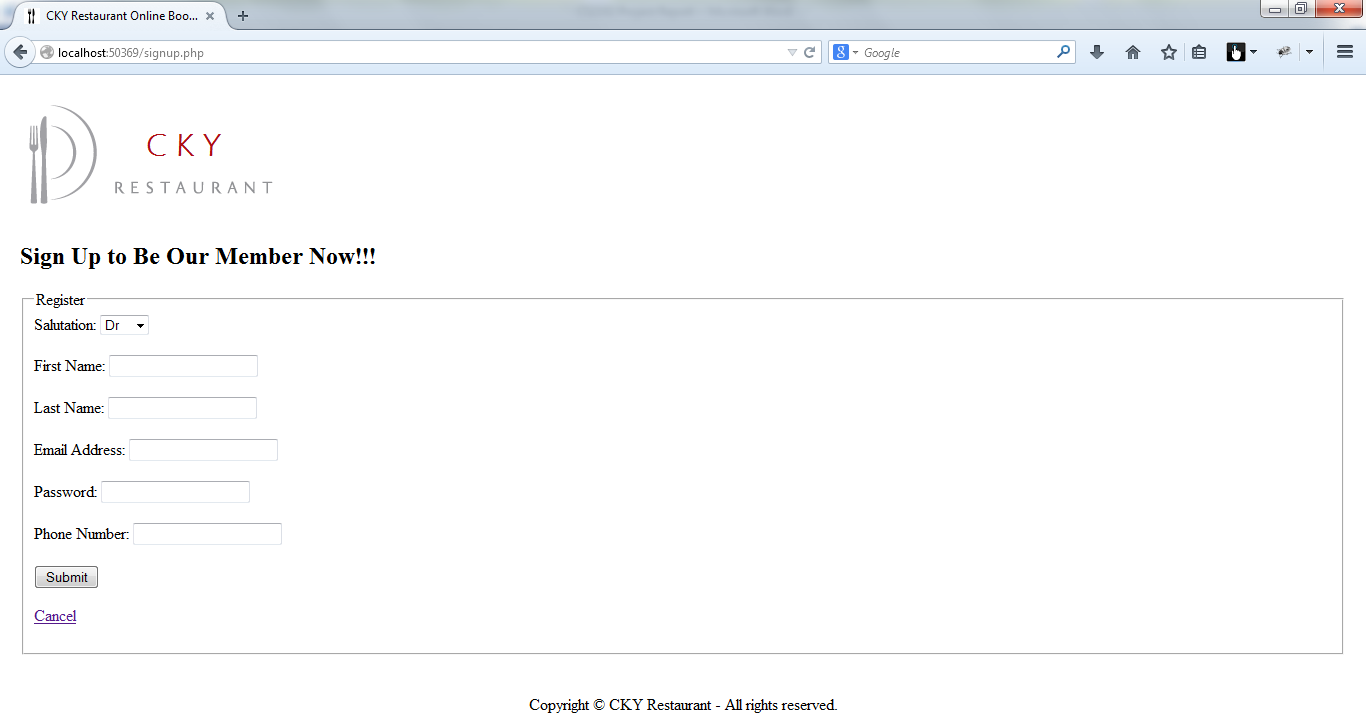
**2.1 Sign up**

**Figure 2.1.1 Interface of Our Home Page (index.php)**



On the main page of the Online Booking System, user can choose to login with their registered account or sign up if the user is a first timer. The user is required to login in order to use the functions on this website, which include making reservation.

**Figure 2.1.2 Sign up page for first time user (signup.php)**



**First layer restriction:**

All fields are required to be filled up and the format of specific fields need to be correct. The checking is done upon submission. Restrictions are as described below:

1. Email address.  
   The checking is done by using JavaScript. The JavaScript function, IsEmail(), which is used to do the checking of email address is included below.

function IsEmail(email) {

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

return regex.test(email);

}

1. Password.  
   Password field is required to be at least 6 characters long.
2. Phone number.  
   Phone number textbox is only allowed to be filled in by numerical characters, and it has to be exactly 8 digits. The checking of numerical character is also done by using JavaScript, as shown below:

function IsPhoneNum(txtPhone) {

var filter = /^[0-9]+$/;

return filter.test(txtPhone);

}

**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 the email address is not found in the database, the signing up function will just be allowed to proceed 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();

}

As an example, if the details that user key in are as follow:   
Email Address: [abc@gmail.com](mailto:abc@gmail.com)  
First Name: Hello  
Last Name: Kitty   
Salutation: Dr.   
Password: 12345678  
Phone Number: 98765432

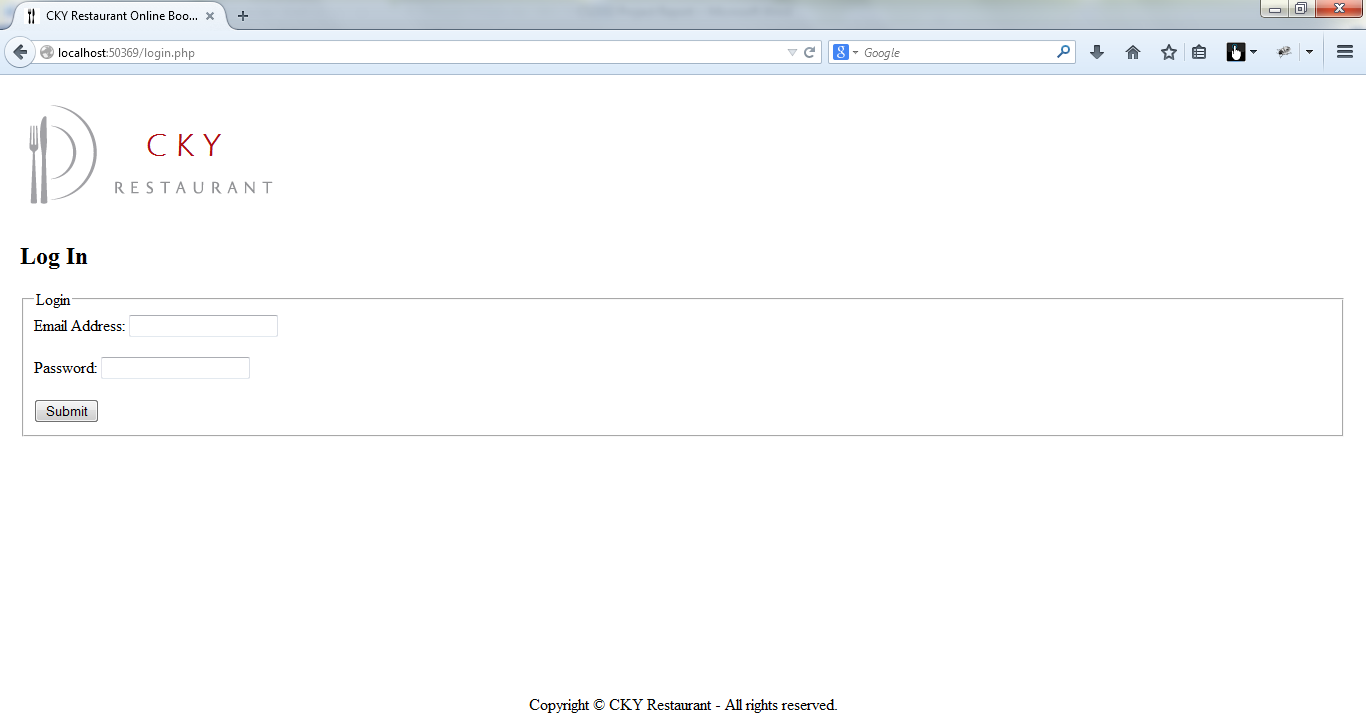
Upon checking the existence of the email address, the executed SQL query is   
***SELECT \* FROM user where Email\_Address = “abc@gmail.com”.***

If no such email is found in the database, the registration will be continued by inserting the details into the user table. The executed SQL query is   
***INSERT INTO user(Email\_Address, First\_Name, Last\_Name, Salutation, Password, Phone\_Number, Created\_On) VALUES (“abc@gmail.com”, “Hello”, “Kitty”, “Dr.”, “12345678”, 98765432, 2014/10/31 08:00:15)***

A successful sign up will redirect user into the homepage.

**2.2 Log in**

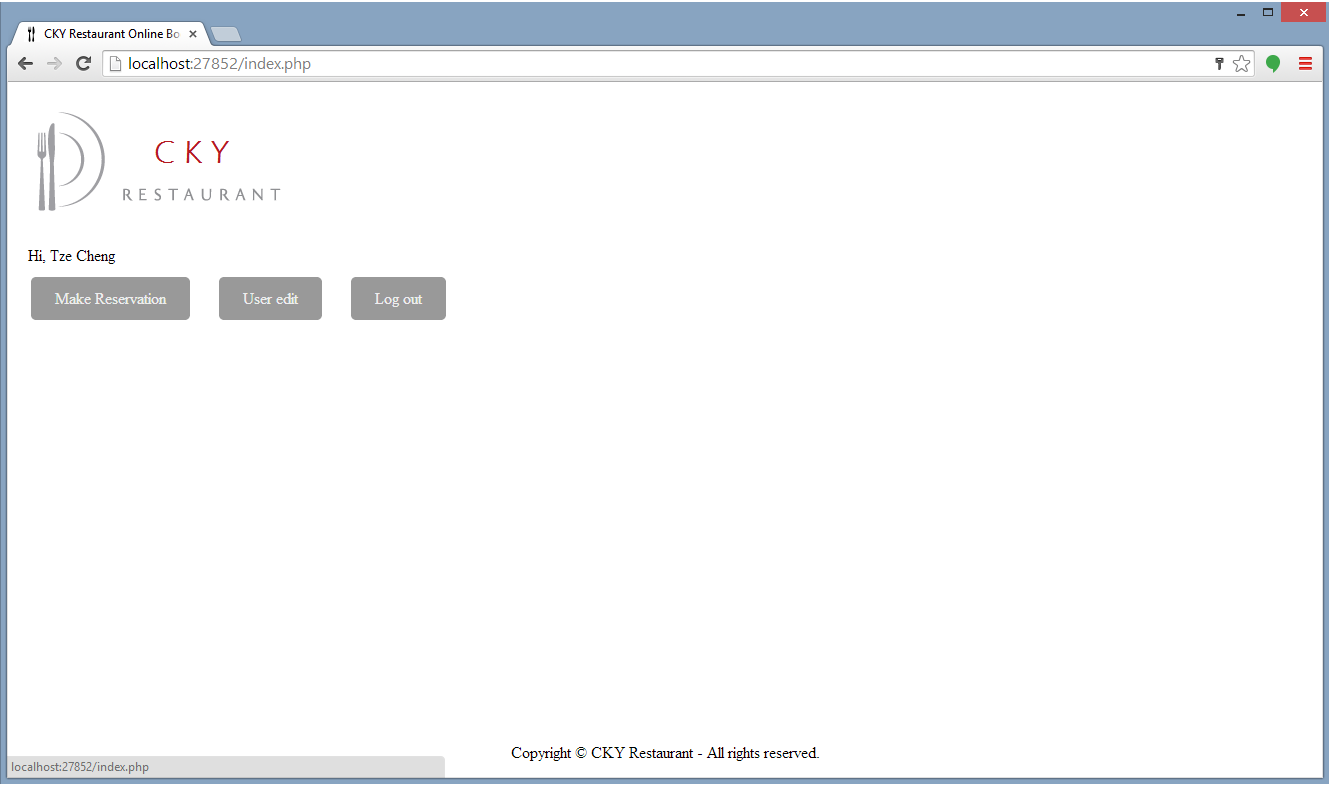
**Figure 2.2.1 log in page for registered user (login.php)**



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 the details are not matched with the one in database. For example, “Email Address/ Password is incorrect!!”.

A successful log in will auto redirect registered user into the homepage and show their last name on homepage. Different functions will be shown according to the user’s role. There are two roles available, namely member and admin. Please look at chapter 3 for the details of the admin part.

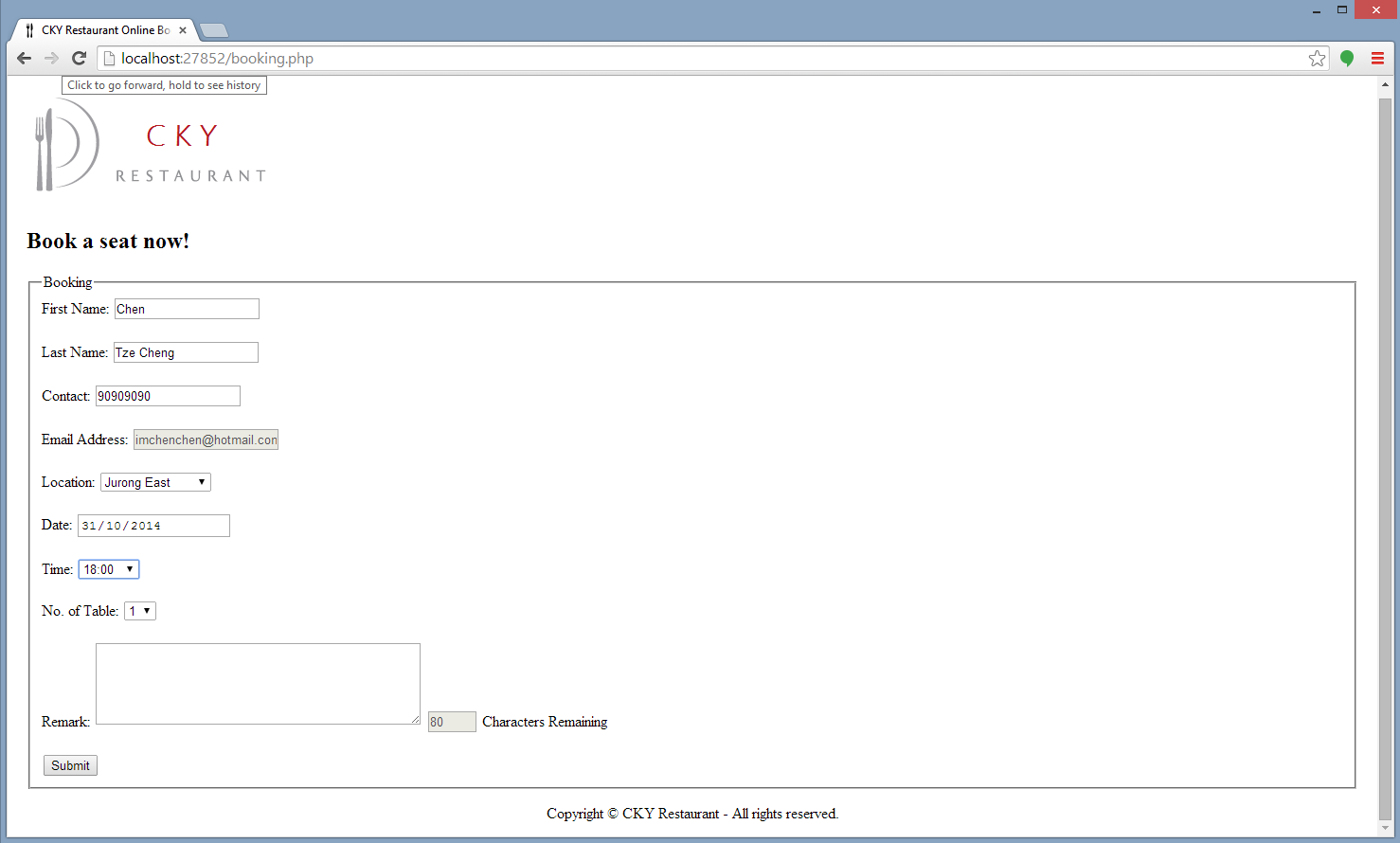
**Figure 2.2.2 Home page after log in (member)**



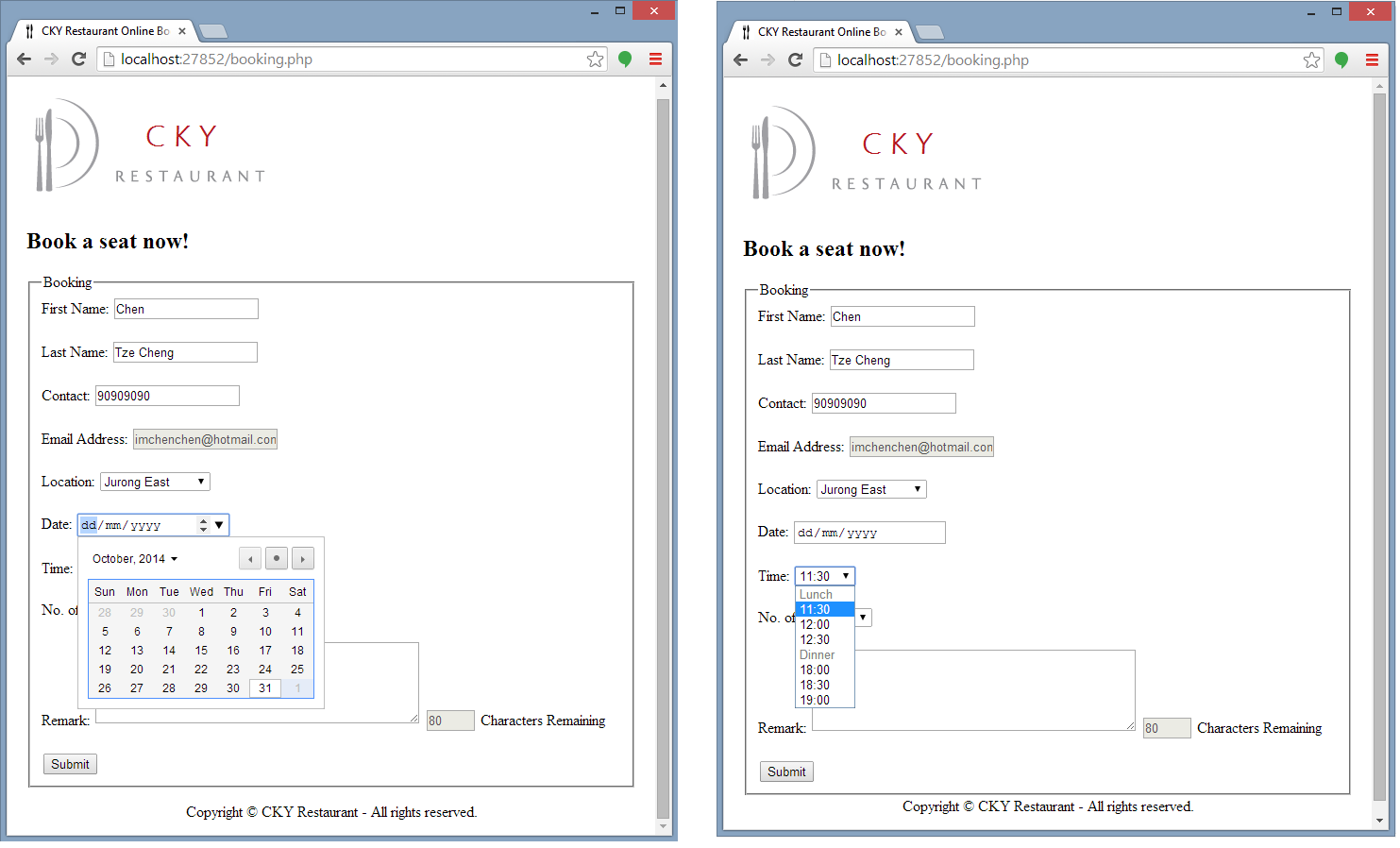
**Chapter 3: Making reservation and User Edit**

**3.1 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 using JavaScript ajax function to retrieve the user’s details from database.

**Figure 3.1.1 Automatic pre-filled fields of booking page. (booking.php)**

User is required to fill up the remaining parts which are the location, date and time they prefer, and number of table they want to reserve. If the user is just a normal member, he/she is not allowed to change the email address but is free to make changes to other fields that are pre-filled by the system. If the user has any special requirements, there is also a Remark field for him to fill in.

**Figure 3.1.2 User can select date and time via calendar and drop down list respectively.** 

If all the fields are entered correctly, there might be a failing scenario which is when 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 =?";

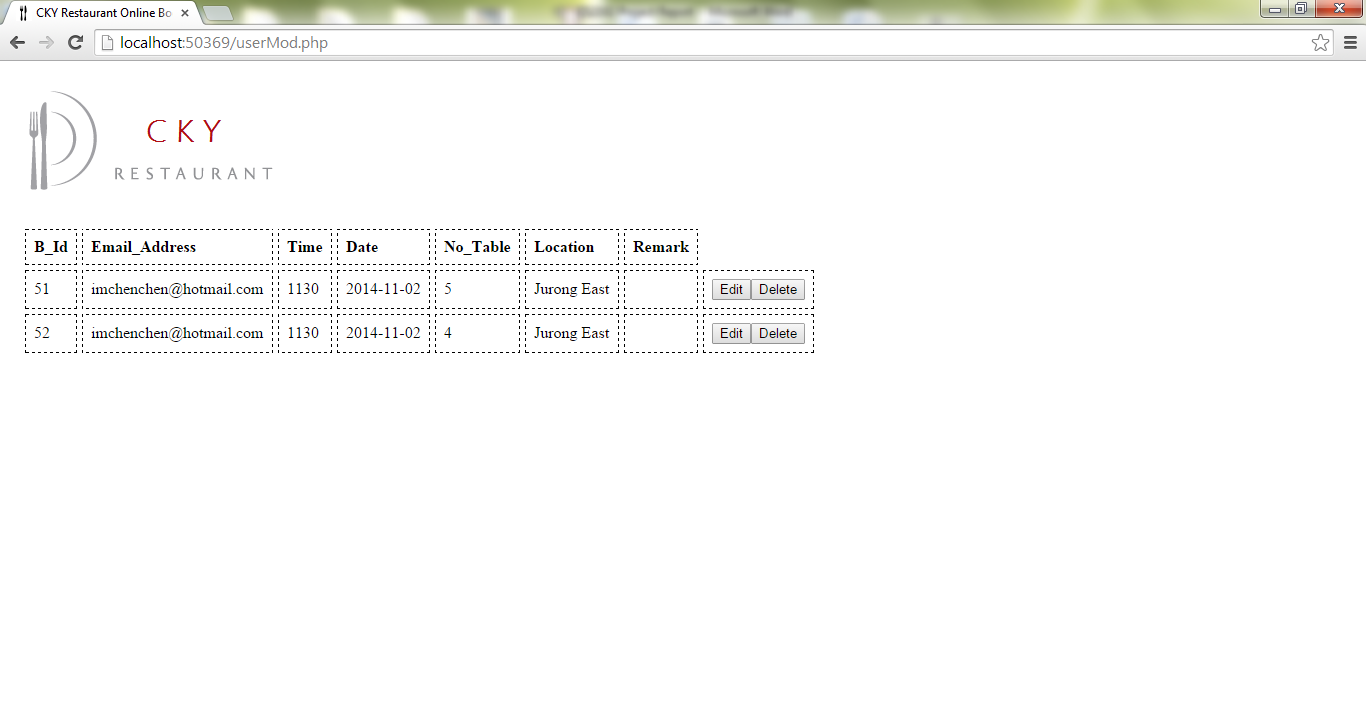
}

Once the booking succeeds, user will be redirect to bookingsuccess.php which displays “Booking Success! Redirection to main page will be proceeded in 3 seconds.

If the booking fails, on the booking page itself, statement like “Sorry, we left with 1 available table(s) in this location.” will be displayed. In the message, the available number of tables in that particular restaurant will be shown.

**3.2 Editing reservation**

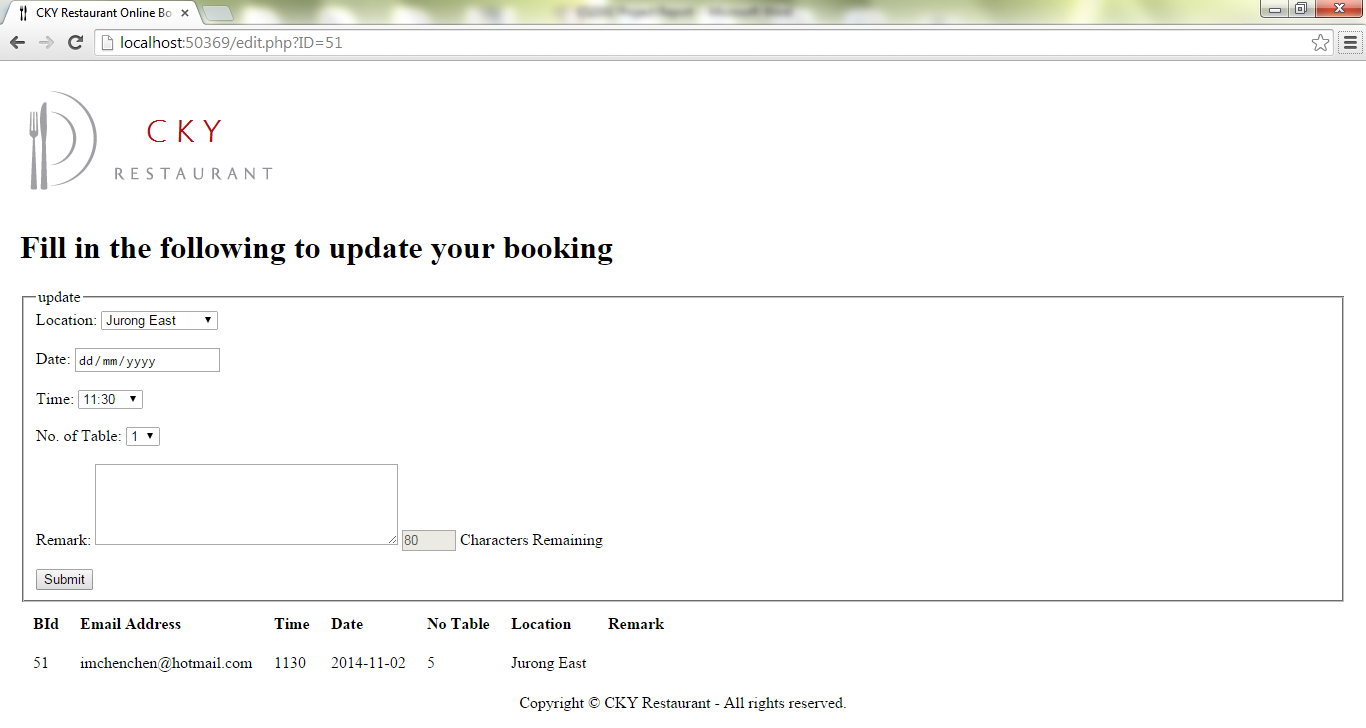
**Fig 3.2.1 A sample user edit page (userMod.php)**



On the homepage, user can go to user edit page (userMod.php), to inspect all the bookings he has made, and make changes if he wants to and if the editing is still allowed. There are two buttons, namely, Edit and Delete, beside each record. User can click on the respective button to make changes. However, reservations with date before the current date will not be displayed.   
As an example of retrieving data from MySQL database, if the user is with email address [abc@gmail.com](mailto:abc@gmail.com), the select query is as follows:

***SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Email\_Address = “abc@gmail.com” AND Date > NOW() ";***

**Fig 3.2.2 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");

}

}

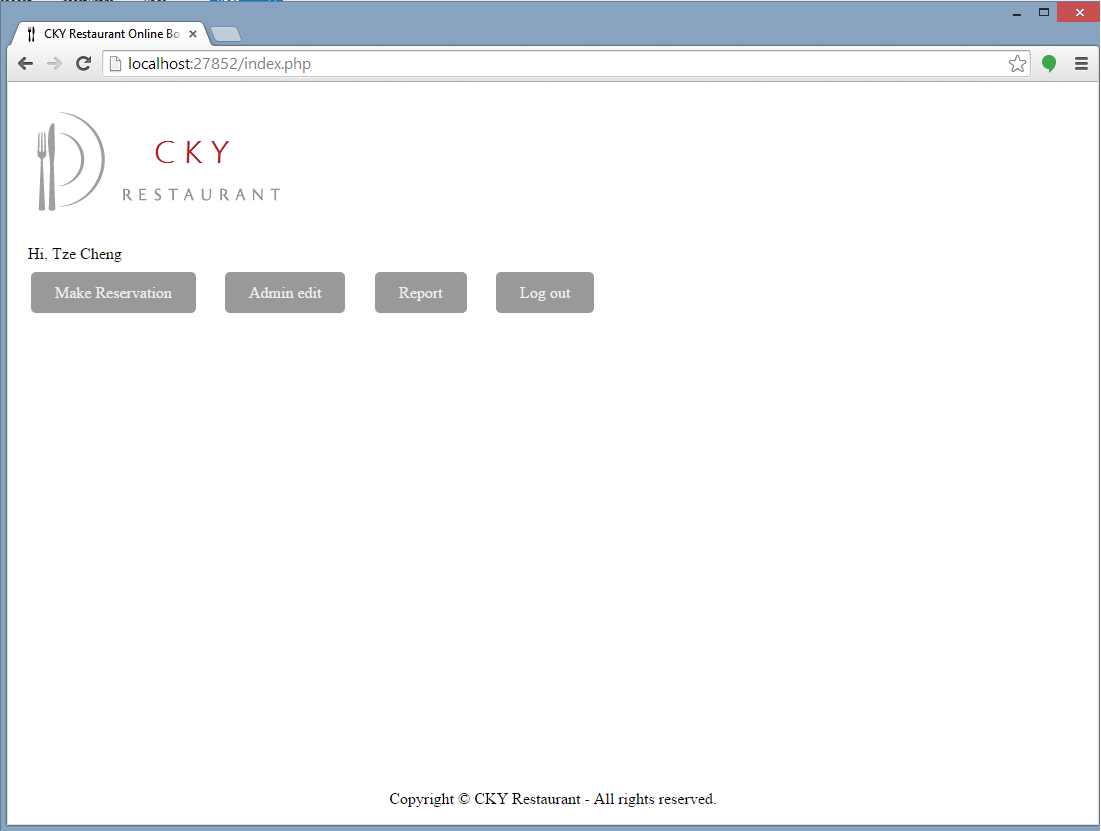
If Delete button is clicked, user will be prompted a warning message to confirm whether he really meant to cancel the reservation. If true, the reservation will be deleted permanently from booking\_record table. Otherwise, nothing will be changed. The eligibility of the user to delete the record is the same as the method mentioned above.

**Chapter 4: Administrator**

**4.1 Admin user**

Admin account cannot be registered in the website itself. It is pre-inserted into the database. After an admin has logged in, the index page will show different functions 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 4.1.1 Main page of admin after logged in**



**4.2 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);

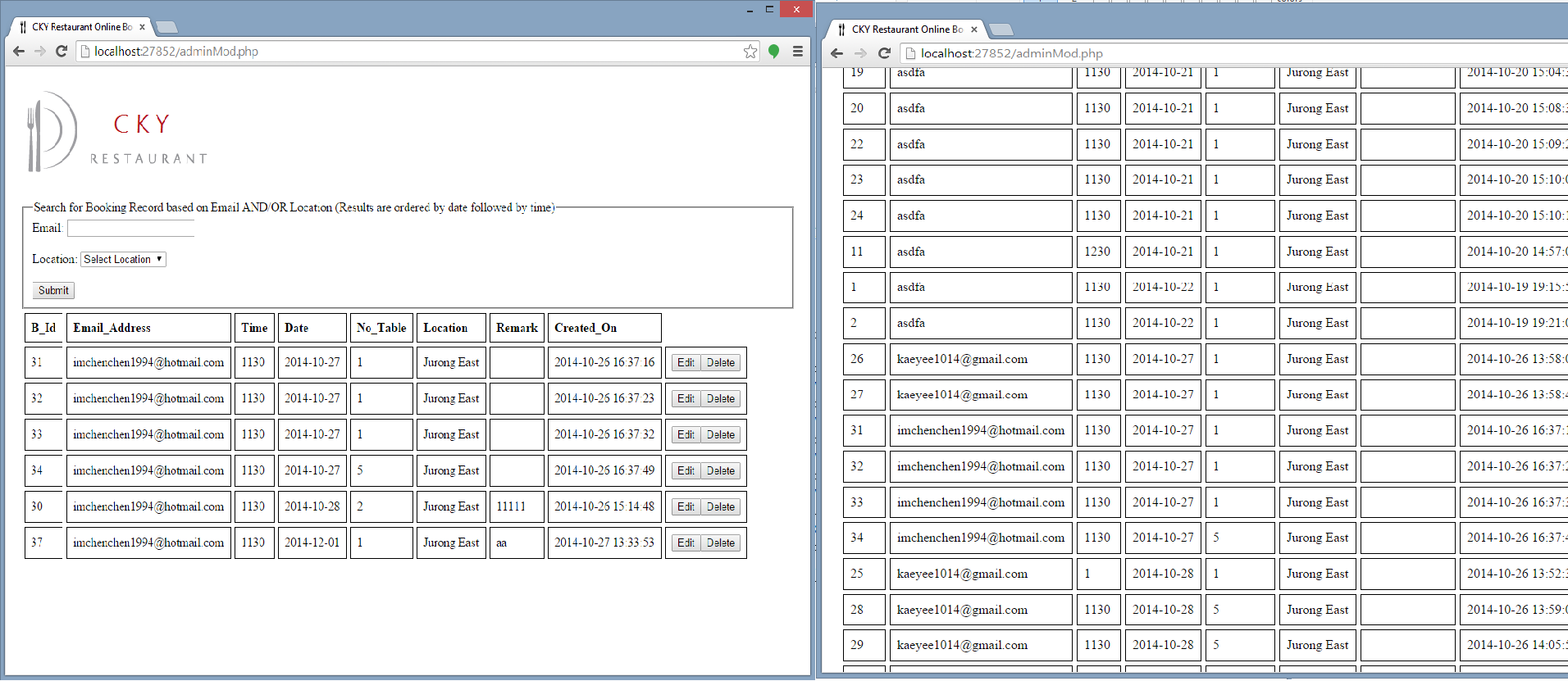
$statement -> bind\_param('s', $locationSubmitted);

}

As an example, if admin fill up the email textbox with [abc@gmail.com](mailto:abc@gmail.com) and leave the Location filed unselected, the SQL that will be execute is the second scenario in the code above, namely,   
***SELECT B\_Id, Email\_Address, Time, Date, No\_Table, Location, Remark, Created\_On FROM booking\_record WHERE Email\_Address = “abc@gmail.com” ORDER BY DATE, TIME***

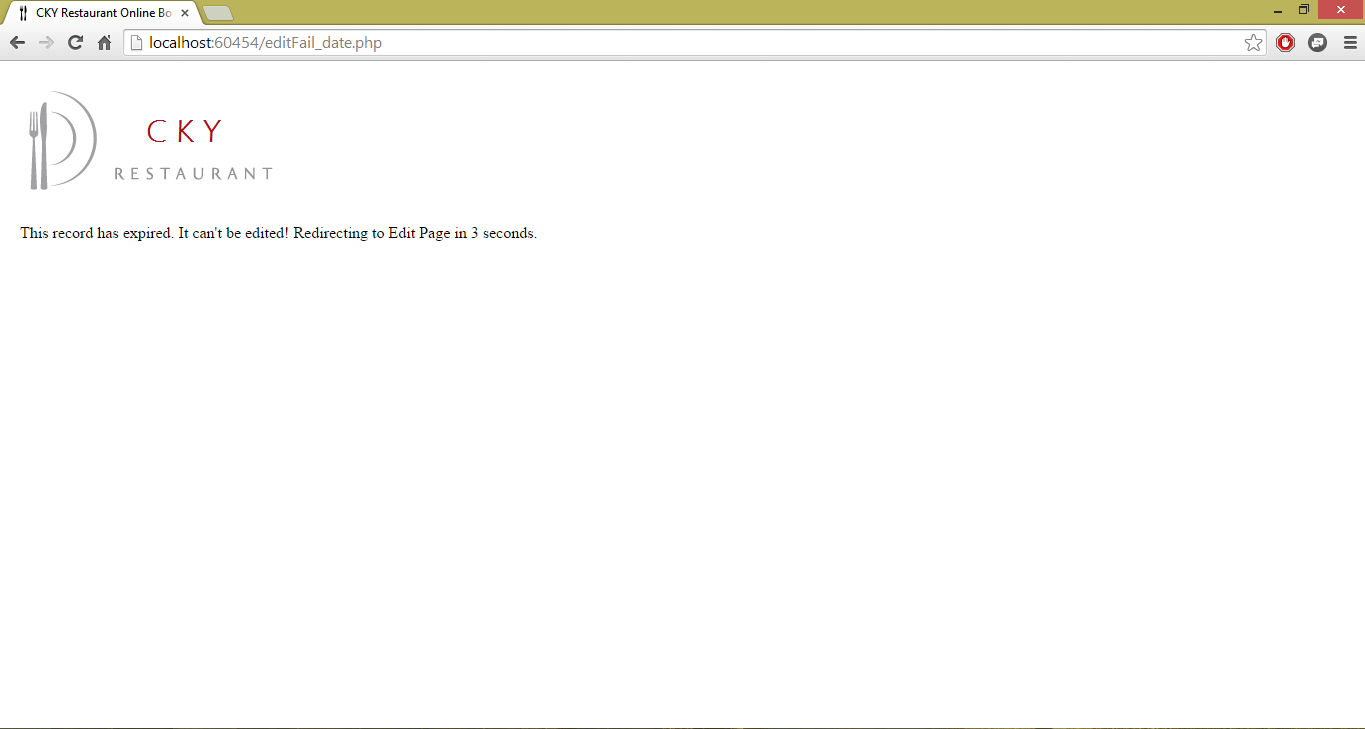
Admin is authorized to edit any record that hasn’t expired yet. Admin has also the permission to delete any records.

**Fig 4.2.1 Admin Edit page (AdminMod.php). Admin can search by email or location, or both.**



If admin try to edit any record that is expired, error message will be shown and no change will be done.

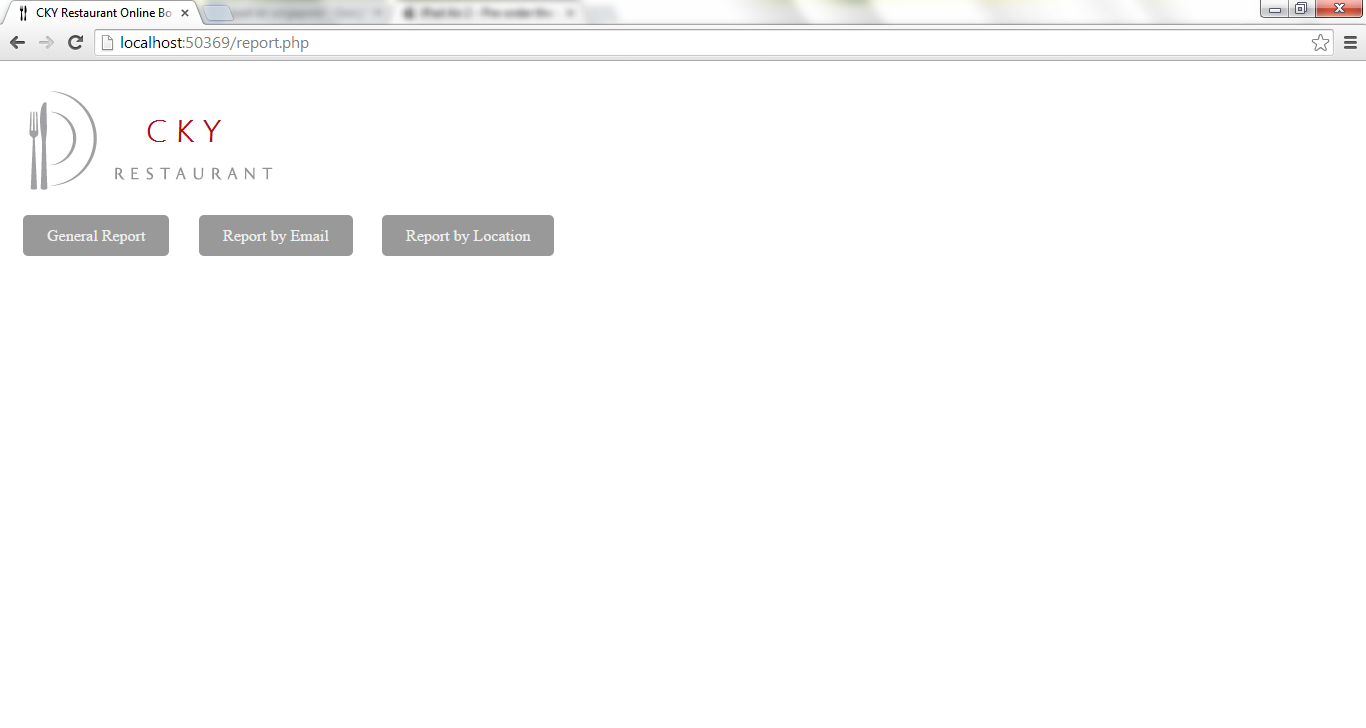
**Fig 4.2.2 Error message when trying to edit expired record.**

****

**4.3 Report**

In report function, there are three kind of report for admin to select from, which are General Report, Report By Email and Report By Location. These reports provide important statistical data for admin to work with.

**Fig 4.3.1 Report page (report.php) showing three selections of report.**



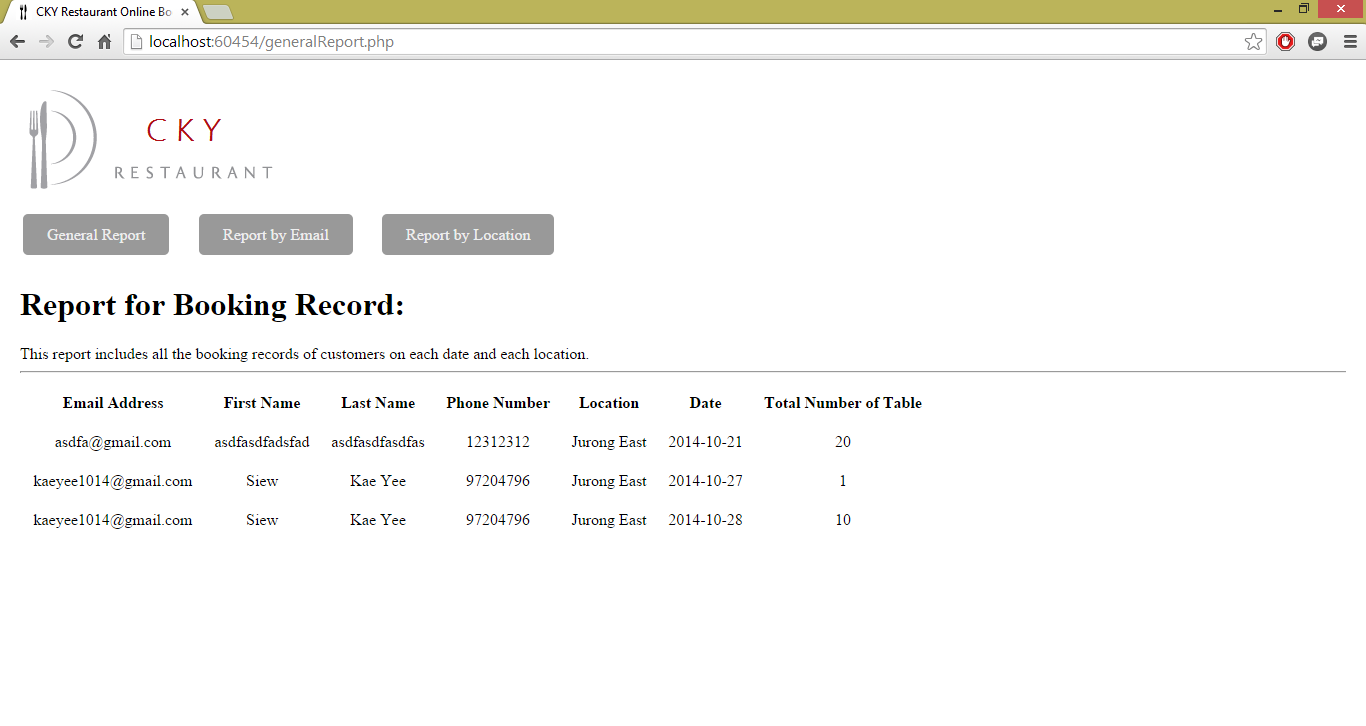
**4.3.1 General Report**

General report will retrieve all the booking records in the database and display them on the browser. Aggregated SQL function is used to do summation and the result is ordered by the email address. The executed SQL query is as follow.

***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;"***

**Figure 4.3.1.1 General Report page**

****

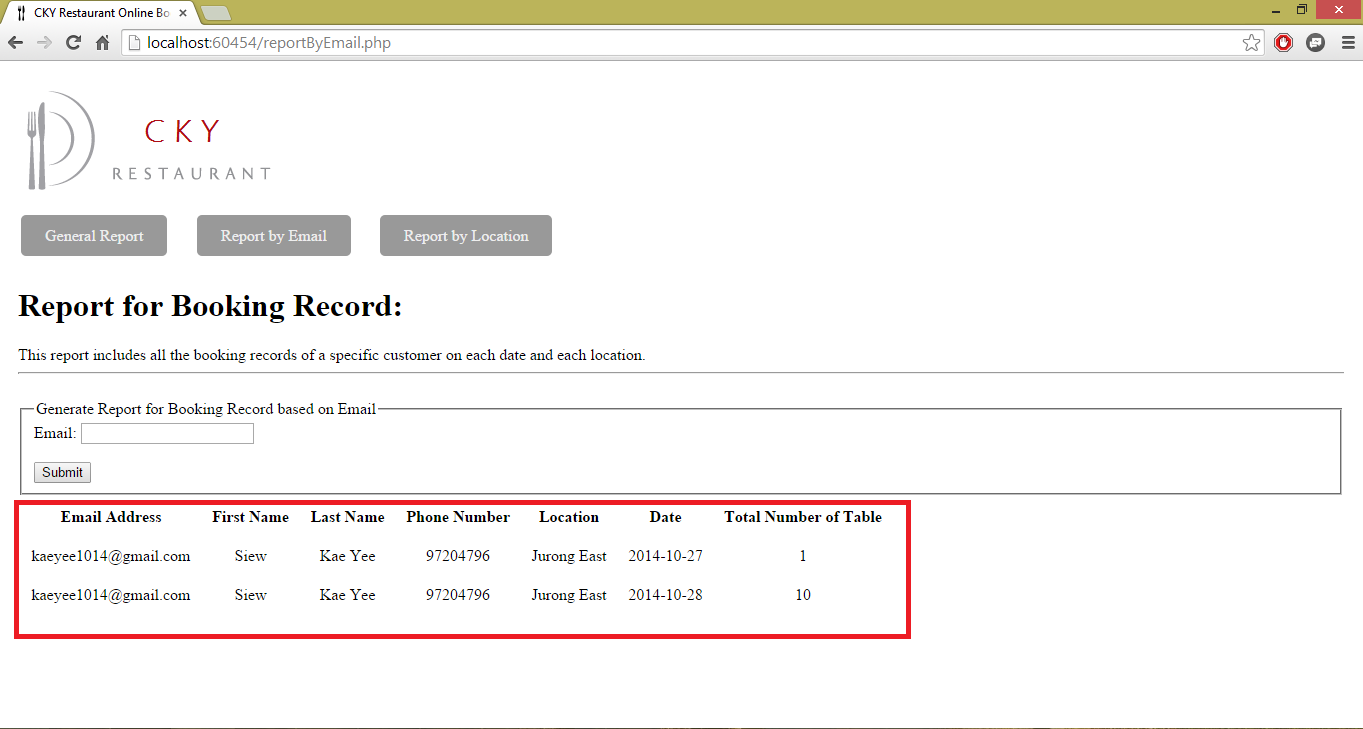
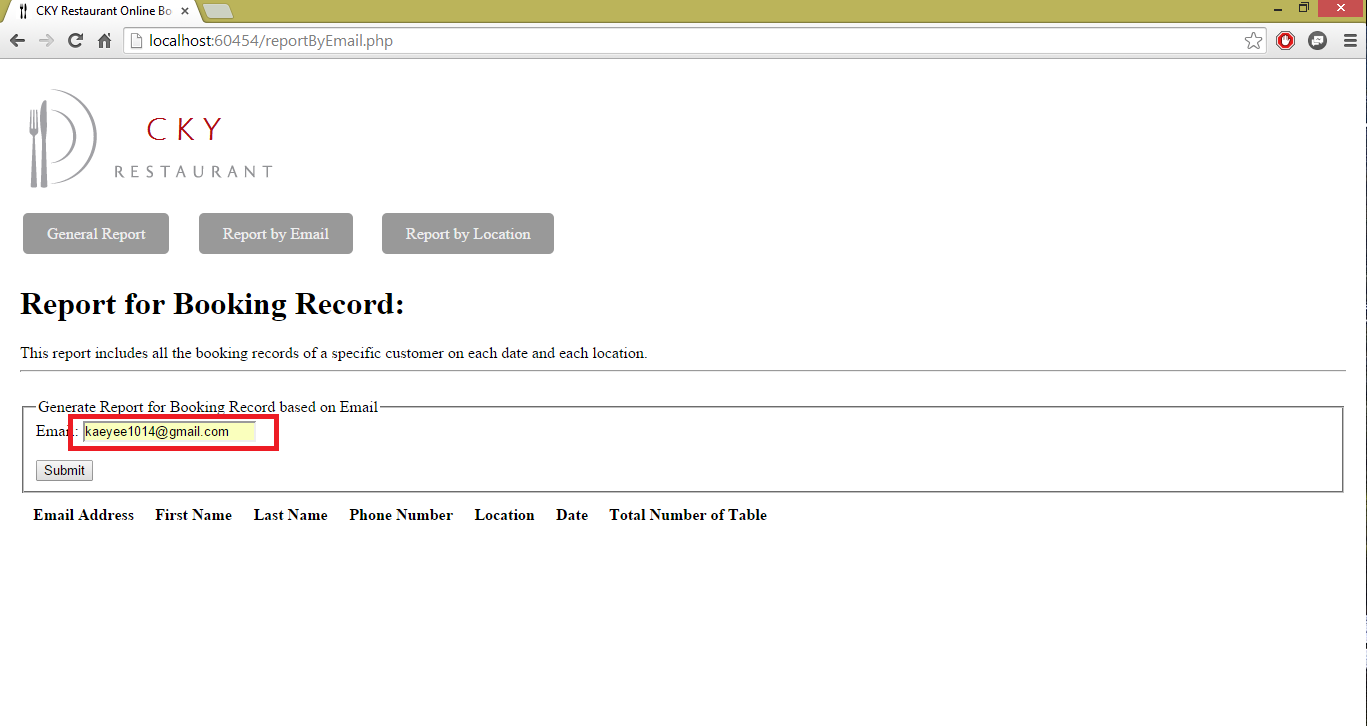
**4.3.2 Report by Email**

Admin is allowed to key in the email of a particular user to retrieve the booking records of the user. Aggregated SQL function is used to do summation. The records are separate by different date and location. The final result is ordered by location, follow by date.

As an example, if the admin keys in [kaeyee1014@gmail.com](mailto:kaeyee1014@gmail.com), the executed SQL query will be

***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 = “kaeyee1014@gmail.com” GROUP BY a.Email\_Address, First\_Name, Last\_Name, Phone\_Number, Location, Date ORDER BY***

***a.Location, a.Date;***

**Figure 4.3.2.1 Report by Email page  
**

**4.3.3 Report By Location**

Admin is allowed to generate report by using a specific restaurant location. A dropdown list with all the available restaurant’s locations is shown. Aggregated SQL function is used to do summation for the number of table booked on a specific date with different timing. The final results are ordered by date follow by time.

For example, if the admin selected location “Jurong East”, the executed SQL will be as follow:

***SELECT Date, Time, SUM(No\_Table) As Total\_Table FROM booking\_record WHERE Location = “Jurong East” GROUP BY Date, Time ORDER BY Date, Time;***

**Figure 4.3.3.1 Report by Email page**