

CAR Resource Booking System

COMP-4983

Supervisor: Dr. Daniel Silver

Written By: Guanglei Hu 100123792

Table of Contents

1.Introduction.....	1
1.1 Problem Definition.....	1
2.Background.....	2
3.Requirements Analysis	3
3.1 Functional Requirements.....	3
3.2 Non-Functional Requirements.....	3
3.3 Constraints.....	3
3.4 Test Scenarios.....	4
4.Detail Design.....	5
4.1Module & Layer Decomposition.....	5
4.2 Use Case Scenarios.....	7
4.3 Sequence Diagram For System.....	12
4.4 Class Design.....	13
5.Implementation Notes.....	19
6.System Testing and Experimentation.....	23
6.1 Testing the log in and register function.....	23
6.2 Testing the booking function.....	24
6.3 Testing the canceling booking function.....	25
6.4 Testing function of deleting and adding resource.....	26
6.5 Testing adding administrator function.....	27
7.Results and Discussion.....	28
8.Conclusion.....	28
8.1 Summary.....	28
8.2 Future Work.....	29
9.Reference.....	29

1. Introduction

In modern life, school and university has lots of resources that can be used by students, faculty and staff. For example, in Acadia University, the Carnegie Hall is the building of computer science faculty. The CAR(Carnegie Hall) has some rooms to study, do lab, do meeting and even hold some activities such as gaming and party. But I know that CAR doesn't have a system that can manage the booking of the resources. I think it is very inconvenient. For example, when the people want to use the resource, they may find that the resource is used by other people. So, I decide to design the Carnegie Hall resource booking system. The faculty and staff will know the booking status of every resources in CAR. The project will be a website based on PHP and MySQL. It is very convenient for user to choose when and how long they will use the resources.

1.1 Problem Definition

Objectives:

- The system provides the user correct information about the available resource.
- The system allows booking of resource without errors and without creating conflicts.
- The system should allow user to register and log in.
- The system should have administrator function that can manage the resource.

Scope:

- The system should be able to run on any browser compatible with HTML5.
- The system can be run on the device which can connect to internet.

Success Criteria:

- The system shows user the correct booking information, so that the user can book the system without creating conflicts.
- The system provides a administrator function that can manage the resource without error.
- The whole system should work well without errors.
- The system provides user a friendly interface, so that the user can use the system easily.

2. Background

Software

The application area of the project is Acadia University. The user of the project should be faculty and staff of computer science. The project will be a website. The website will be built by PHP, Javascript, HTML, bootstrap and MySQL database. I will use the MAMP to build the website. MAMP is Apache, MySQL and PHP runs on Mac, but Windows user can use MAMP too. I can use Apache to build a local server easily. So, the project will be developed and tested on the local server. PHP is a server-side language that can connect to the MySQL database easily. The website should be able to run on any device with a browser compatible with HTML5. So, I decide to use bootstrap. Because the mobile-first is one of the advantages of bootstrap. So, my project will not only be able to run on PC, but also be able to run on mobile device.

Hardware

I will use my PC to develop the project. The information below is some important parameters of hardware.

Lenovo IdeaPad U530 Touch

Operating System: Microsoft Windows 10

Ram: 8GB

CPU: 4th Gen Intel® Core™ i7-4500U (1.8 GHz 200 MHz 4MB)

3. Requirements Analysis

3.1 Functional Requirements

1. The system should allow user to register an account.
2. The system should allow user to log in.
3. The system should show all the resource to user.
4. Users can choose the resource and the date and time when they want to book the resource .
5. Users can cancel a booking that is made by themselves.
6. Administrator can add the new resource.
7. Administrator can delete the resource.
8. Administrator can add the new administrator.

3.2 Non-Functional requirements

1. System should help user to fill the log in form, register form by providing the hint message or error message.
2. System should show a confirm message to user when he/she finish the booking.
3. System should be able to run on the mobile device that has the browser compatible with HTML5.

3.3 Constraints

1. Each user should have the unique username.
2. Each user should have the unique e-mail address.
3. Every resource should have the unique name and id.
4. The project must be completed before March 31st.

3.4 Test Scenarios

1. Testing the register function.

I will input correct value and incorrect value. The incorrect input means that the input format is wrong. When the input is correct, a new account can be generated. Otherwise, the system should show user the error message.

2. Testing the log in function

I will input correct and incorrect value. The incorrect input means that the input value does not match the database record. When the input is correct, user can log in the system. Otherwise, the system show user the error message.

3. Testing the booking function

I will choose each resource first and see whether the system shows the correct booking information about the resource. If the correct booking information is shown to user, then I can start the second step. I choose the date and time to book the resource. Then I will see whether the correct booking is created.

4. Testing the canceling function.

I will choose the booking record and see if the chosen record is canceled.

5. Testing function of adding resource.

I will input the proper or improper value and see if the resource is added successfully or the error message is shown.

6. Testing function of deleting resource.

I will choose the resource that I want to delete and see if the chosen resource is deleted.

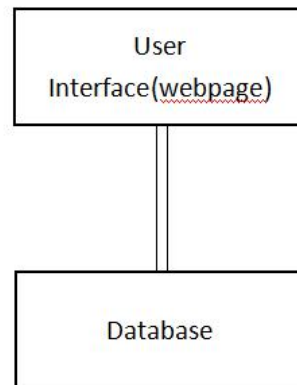
7. Testing function of adding administrator

I will input the proper and improper value and see whether the administrator is

added successfully or the system shows error message.

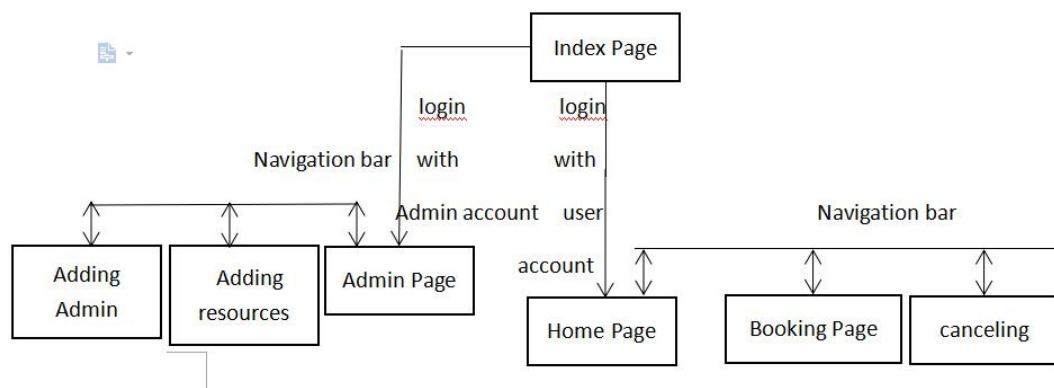
4. Detailed Design

4.1 Module & Layer Decomposition



4.1.1 User Interface Layer

The user interface for the system is the web page that is shown to user. We have the index page. User can log in the system on index page. The system has the homepage, booking page and canceling booking page for users. The pages are connected by a navigation bar. The system also has a page for administrator to manage the booking information and administrator account. The diagram below is the site map for the system.

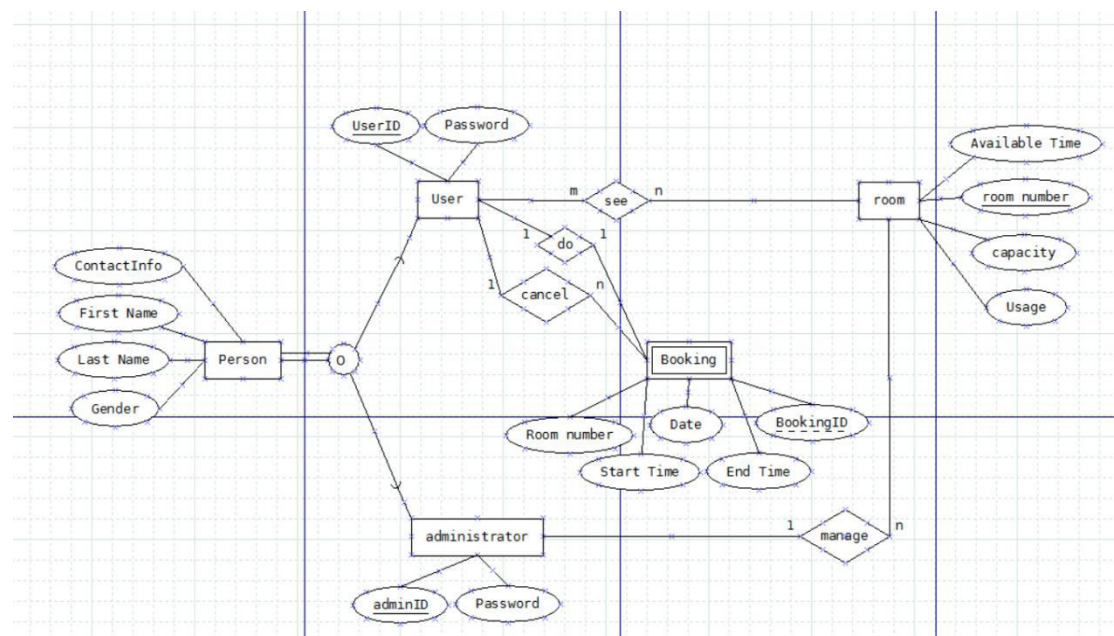


4.1.2 Database

The database has 3 entities: room information, booking information, and account information. So, the database has 3 tables corresponding to the 3 entities.

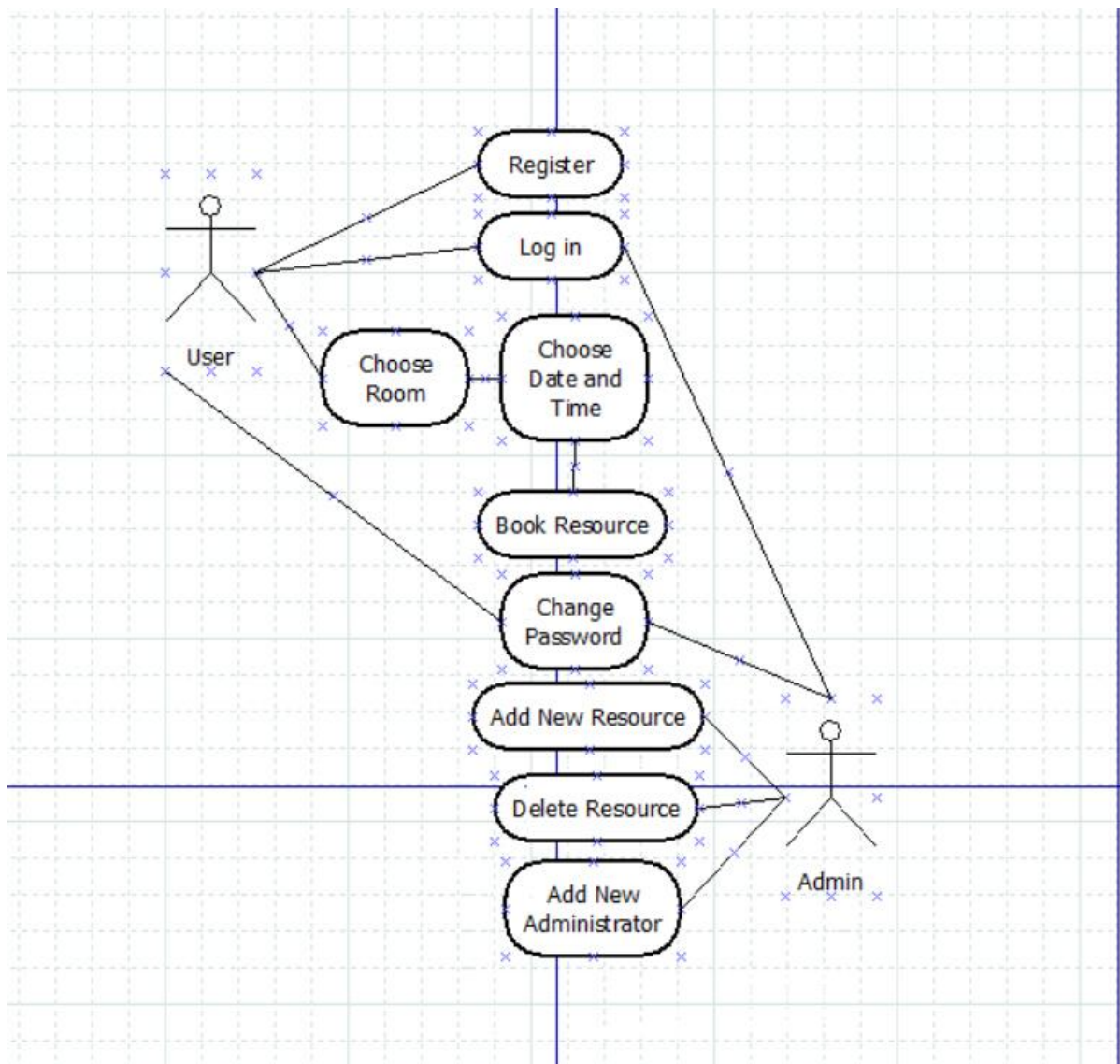
The room information table has the columns: Room Name, Room Usage, Room Capacity and Room ID. The user account information table has the columns: Username, Password, Gender, Firstname and Lastname. The booking table has the columns: BookingID, Date, Room Name, Start Time and End Time.

The diagram below is the E-R diagram for the database:



4.2 Use Case Model Decomposition

Use case diagram for the system:



4.2.1 Use case scenarios

Use Case ID: register

Primary Actor: User who doesn't have an account

Pre-Condition: User visit the booking system and want to register.

Success-Post-Condition: A new account is registered successfully.

Failed-Post-Condition: The registration is failed.

Basic Flow:

- 1.The system provide a webpage that contains a register button.
- 2.User click the register button and a register form will be shown.
- 3.User fill the register form and click submit button.
- 4.A new account is registered successfully, or it is failed and shows an error message.

Use Case ID: Log In

Primary Actor: User who has an account and want to log in the system.

Pre-Condition: User visit the index page of website and want to log in.

Success-Post-Condition: User log in successfully.

Failed-Post-Condition: The log in is failed.

Basic flow:

1. User visit the index page of the booking system and see a log in form.
2. User fill the log in form and click log in button.
3. User log in successfully. Or the log in is failed and an error message will be shown.

Use Case ID: Booking

Primary Actor: User who has log in the system.

Pre-Condition: User has log in the system and want to book a room.

Success-Post-Condition: The booking process is success.

Failed-Post-Condition: The booking process is failed.

Basic Flow:

1. User see a table that contains all information about every room.
2. User choose the room by clicking a 'book' button.
3. User see a table that contains the available booking time of the chosen room and select the date and time, then click 'book' button.
4. The booking process is successful. Or an error message will be shown.

Use Case ID: Canceling Booking

Primary Actor: Users who want to cancel their own booking.

Pre-Condition: User has already booked a room. User has navigated to the canceling page.

Success-Post-Condition: The booking is canceled successfully.

Failed-Post-Condition: The canceling is failed and the page shows an error message.

Basic Flow:

1. User click the 'view booking' button on the navigation bar.
2. The user see their own booking information.
3. The user choose the booking they want to cancel and click 'cancel button'.
4. The canceling process is successful.

Use Case ID: Administrator Log in

Primary Actor: Administrator

Pre-Condition: Administrator visit the index page of the system.

Success-Post-Condition: Administrator log in successfully.

Failed-Post-Condition: Administrator is failed to log in the system. And system will show an error message.

Basic Flow:

1. Administrator visit the index page, and see a log in form.
2. Administrator fill the log in form and click 'log in' button.
3. Administrator log in successfully. Or the log in is failed and system shows an error.

Use Case ID: Adding New Resources

Primary Actor: Administrator

Pre-Condition: Administrator has log in the system using administrator account.

Success-Post-Condition: New resource is added successfully.

Failed-Post-Condition: The process is failed.

Basic Flow:

1. Administrator click 'add resource' button.
2. Administrator see a form about adding new resource.
3. Administrator fill the form and click 'submit' button.
4. The adding process is successful or failed.

Use Case ID: Deleting Resources

Primary Actor: Administrator

Pre-Condition: Administrator has log in the system using administrator account.

Success-Post-Condition: Chosen resource is deleted successfully.

Failed-Post-Condition: The process is failed.

Basic Flow:

1. Administrator click 'delete' button on the resource record that he/she want to delete.
2. The resource is deleted successful or the process is failed.

Use Case ID: Add New Administrator

Primary Actor: Administrator

Pre-Condition: Administrator has log in the system using administrator account.

Success-Post-Condition: New Administrator is added successfully.

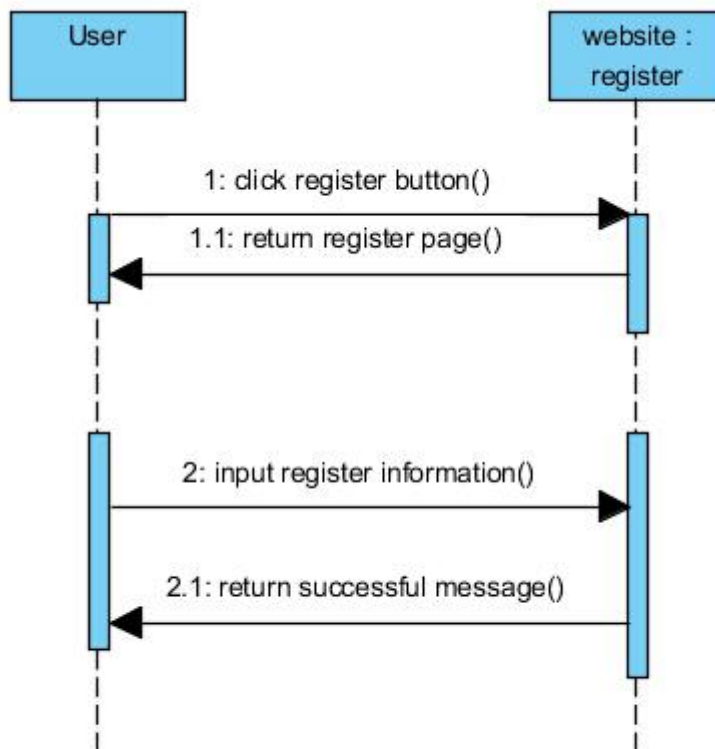
Failed-Post-Condition: The process is failed.

Basic Flow:

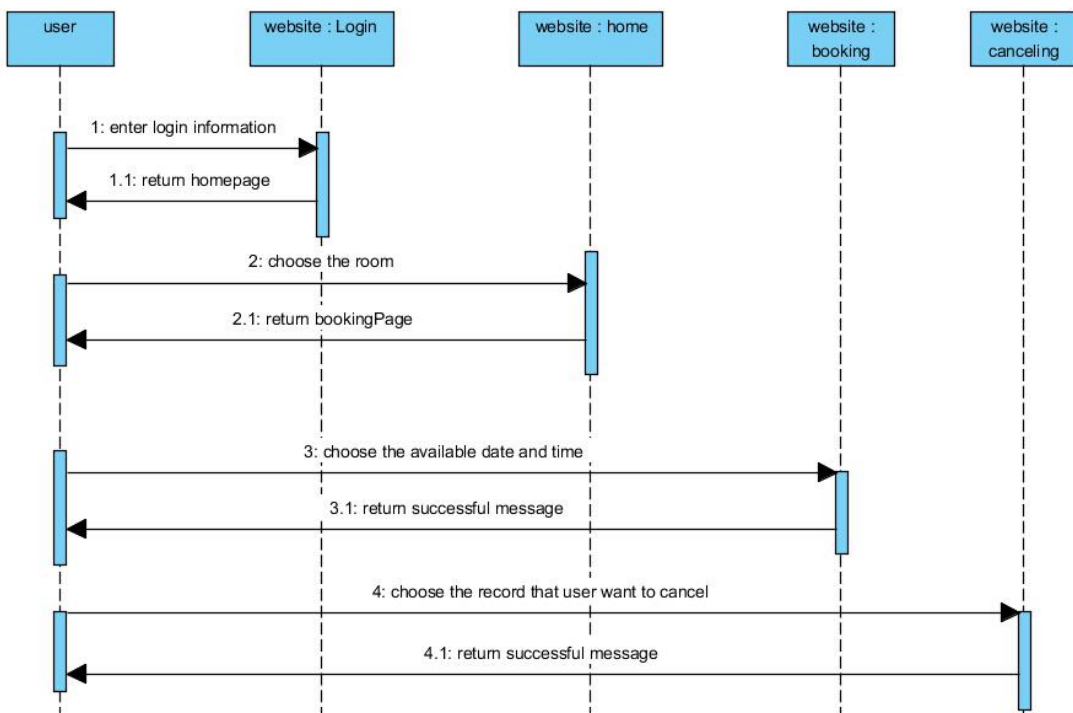
- 1.Administrator click 'Add new admin' button.
- 2.Administrator see a form about adding new administrator.
- 3.Administrator fill the form and click 'submit' button.
- 4.The adding process is successful or failed.

4.3 Sequence Diagram For System

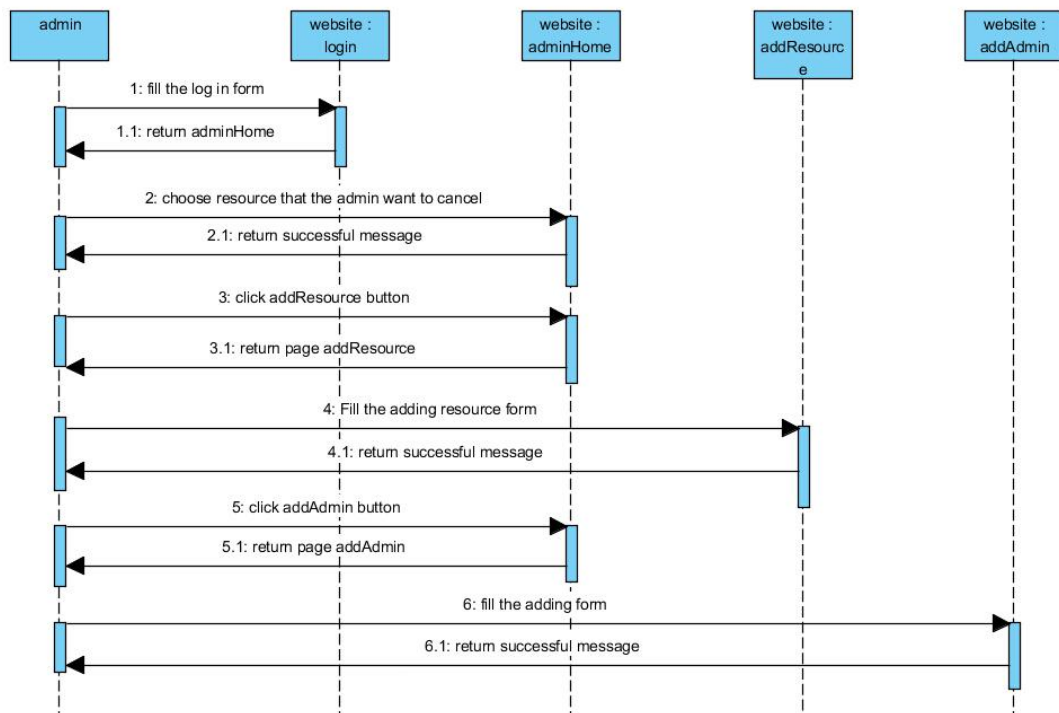
Register:



User log in and booking:

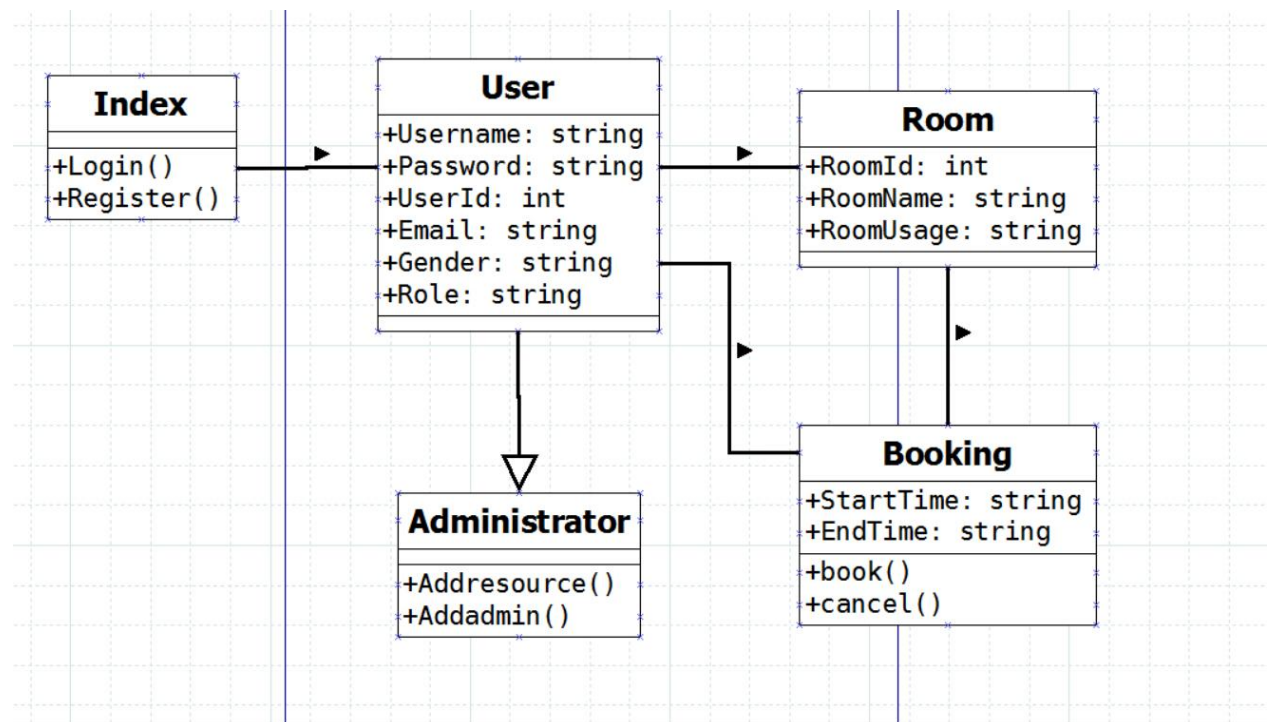


Administrator Functions:

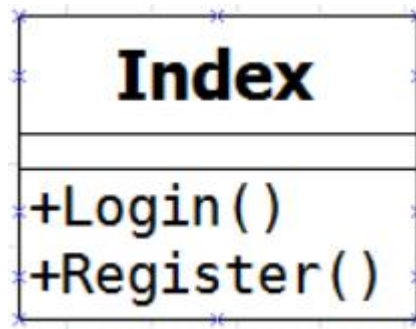


4.4 Class Design

Class diagram for system:



4.4.1 Index Class



Attributes:

The class is related to the class user. So, it can use the attribute in User Class.

String Username:

Store the username as string variable.

String Password:

Store the password as string variable.

Int UserId:

Store the UserId as an int number.

String Email:

Store the user's email as a String variable

String Gender:

Store the Gender information of user as a String variable.

String Role:

Store the role of user as a string variable.

Methods:

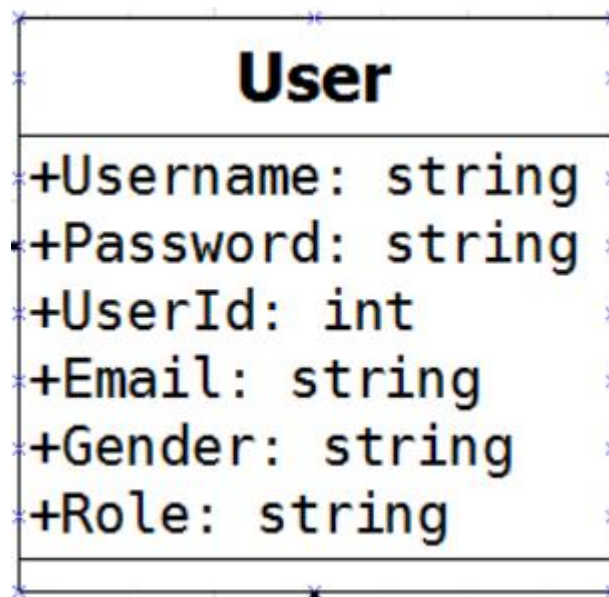
Login():

Log in the system using user name and password.

Register():

Register an account of the system.

4.4.2 User Class



Attributes:

String Username:

Store the username as string variable.

String Password:

Store the password as string variable.

Int UserId:

Store the UserId as an int number.

String Email:

Store the user's email as an String variable

String Gender:

Store the Gender information of user as a String variable.

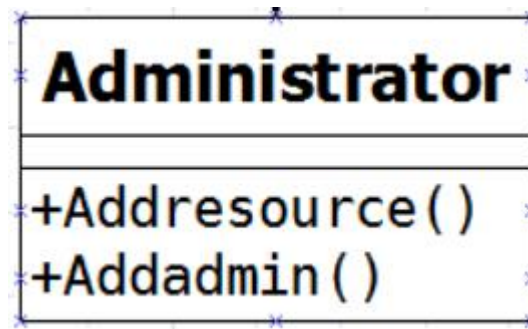
String Role:

Store the role of user as a string variable.

Method:

There is no method in this class.

4.4.3 Administrator Class



Attributes:

The Administrator Class is inherited from the class User. So, it uses the attributes in the User Class.

String Username:

Store the username as string variable.

String Password:

Store the password as string variable.

Int UserId:

Store the UserId as an int number.

String Email:

Store the user's email as an String variable

String Gender:

Store the Gender information of user as a String variable.

String Role:

Store the role of user as a string variable.

Methods:

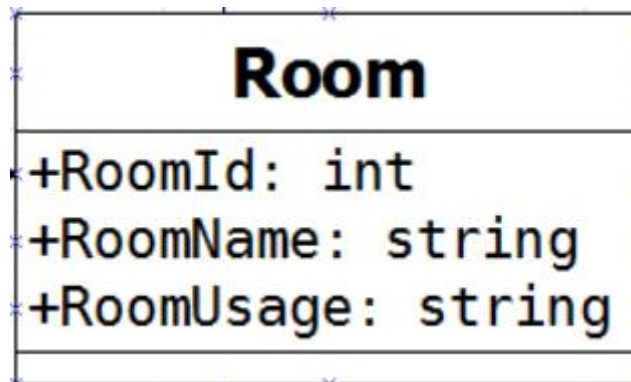
Addresource():

The method Addresouce() allows administrator add new resource into the system.

Addadmin():

This method allows administrators add a new administrator to the system.

4.4.4 Room Class



Attributes:

Int RoomId:

The attribute stores the Room Id as a int number.

String RoomName:

The attribute stores the room name as a string variable.

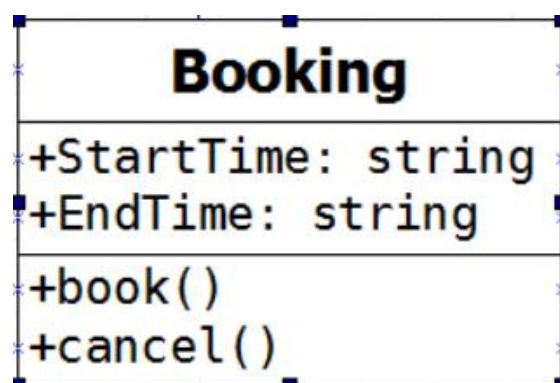
String RoomUsage:

The attribute stores the room usage as a string variable.

Methods:

The Room Class has no method.

4.4.5 Booking Class



Attributes:

String StartTime:

The attribute stores the start time of a booking as a String variable.

String EndTime:

The attribute stores the end time of a booking as a String variable.

The Booking Class is related to the Room Class and the User Class. So, it can use the attributes in Room Class and User Class.

String Username:

Store the username as string variable.

String RoomName:

The attribute stores the room name as a string variable.

Method:

Book():

The book() method allows user to book the resource in the system.

Cancel():

The cancel() method allows users to cancel the their own booking.

4.4.6 Pseudo Code Examples:

Login():

Get input[username], input[password]

if (table 'User' does not contain input[username])

 output error message.

else if (table 'User' contains input[username])

 if(input[password] matches the password in database)

 Log in to the system.

 else

 Output error message.

Booking():

User click the book button.

Get the 'roomName' that is chosen by user.

User choose date and time.

Get the 'date' and 'time' that is chosen by user.

Insert the table 'booking' the data: room name = roomName, date of the booking = date, time of the booking = time.

5.Implementation Notes

1. Use PHP Session to memorize the user's log in status.

When user log in the the system. PHP create a variable `$_SESSION['username']` in every PHP file of the system. I use the following code:

```
<?php
    session_start();
    if(isset($_SESSION['username']))
    {
    }
    else
    {
        header("Location: index.php");
    }
?>
```

2. When user log out , the system will kill the session variable:

```
session_destroy();
```

3. I need to use date and time in the system.

```
date_default_timezone_set("America/Halifax");
date();
```

The date() can present a date: date("Y-m-d"). It can also present a time:

```
date("H:i",$time), $time is made by the function mktime();
```

```
mktime(int, int, int);
```

```
strtotime();
```

When I need to compare two variable of time, I need to use this function. The type of result strtotime() is same with mktime().

4. When the user book the resource, the system needs to know that which checkbox is checked and which time slot the user choose.

Room name	Time	Status	Booked By
CAR211	08:00-09:00	available	<input type="checkbox"/>
CAR211	09:00-10:00	available	<input type="checkbox"/>
CAR211	10:00-11:00	available	<input type="checkbox"/>
CAR211	11:00-12:00	available	<input type="checkbox"/>
CAR211	12:00-13:00	available	<input type="checkbox"/>
CAR211	13:00-14:00	available	<input type="checkbox"/>
CAR211	14:00-15:00	available	<input type="checkbox"/>
CAR211	15:00-16:00	available	<input type="checkbox"/>
CAR211	16:00-17:00	available	<input type="checkbox"/>
CAR211	17:00-18:00	available	<input type="checkbox"/>
CAR211	18:00-19:00	available	<input type="checkbox"/>
CAR211	19:00-20:00	available	<input type="checkbox"/>

book

Figure1

I give every checkbox a name "check", and use getElementByName("check").

Then I store the result into a variable and use ".checked" event to know the checked status. Here is the code:

```
function myFunction1()
{
    var a = document.getElementsByName("check");
    var n = a.length;
    for (var i=0; i<n; i++)
    {
        if(a[i].checked)
        {
            //do something
        }
    }
}
```

I know that PHP is the server-side language and javascript is client-side language. So, I cannot use javascript variable directly in PHP code. After some hardworking, I found the solution: Ajax.

First I give every checkbox a unique id that is related to time slot on the same line. It means the checkbox on the first line has id='8', the checkbox on the second line has id='9' and so on. Then I use the Ajax pass this id value to PHP. Here is code:

Javascript:

```
function myFunction1(){
    var a = document.getElementsByName("check");
    var n = a.length;
    var b = "";
    for (var i=0; i<n; i++)
    {
        if(a[i].checked)
        {
            var b=a[i].id;
            $.ajax({
                                type:'GET',
```

```

        url:"book.php",
        data:{text: b},
        success: function(data)
        {
        }
    });
    }
}
}

```

PHP:

Get['text'] will get the value of id.

5. The code for "Figure1" is written by the for loop, this table should consider the current time. For example, if the current time is 10:10, the table will not show the time slot before 11:00. So, I need to add condition in for loop to echo table.

6. In the column of status of Figure1, I need compare the time information to the time in database. This comparison happens in the while loop which is embedded in the for loop:

```

while($row = mysqli_fetch_assoc($result)){
}

```

I find a problem when I design this part. The status information is correct only in the first line. After some working, I find that the pointer of this loop never go back to the original position after it is executed. So I use the following function to solve this problem:

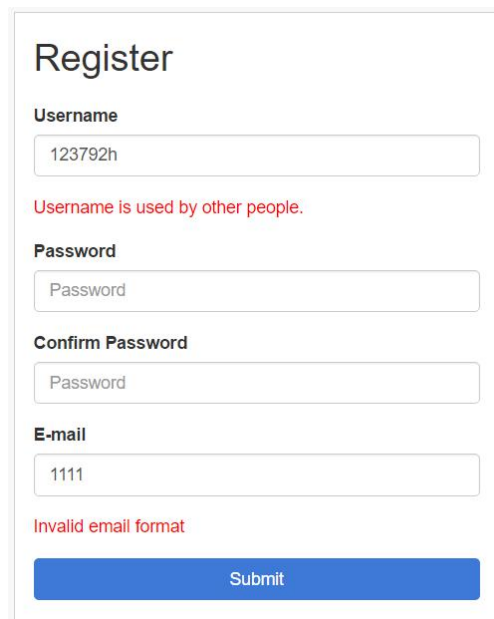
```

mysqli_data_seek($result,0);

```


6. System Testing and Experimentation

6.1 Testing the log in and register function.



Register

Username

Username is used by other people.

Password

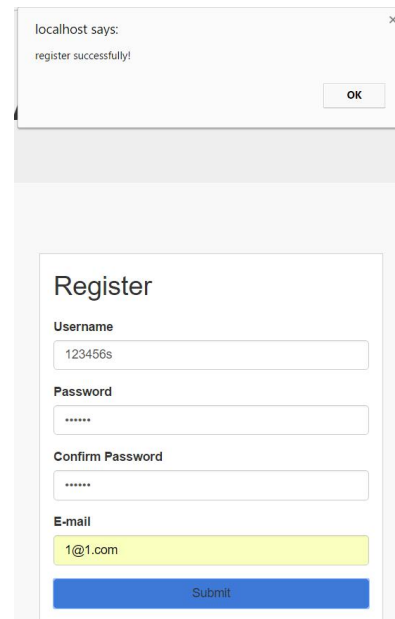
Confirm Password

E-mail

Invalid email format

Submit

Figure2.1



localhost says:
register successfully!

OK

Register

Username

Password

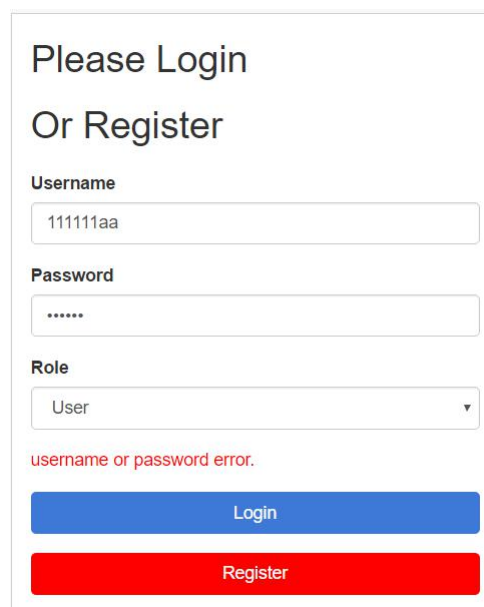
Confirm Password

E-mail

Submit

Figure2.2

When I input the exist username and invalid e-mail, the system shows the error message. When I input the correct information, the system shows the successful message.



Please Login
Or Register

Username

Password

Role

User ▼

username or password error.

Login

Register

Figure2.3

When I input the incorrect username or password, the system shows the error message. When the input is correct, the user will log in to system and go to the home page.

6.2 Testing the booking function

Room name	Usage	
CAR212	student's lounge	<input type="button" value="book"/>
CAR210	lab	<input type="button" value="book"/>
CAR211	lab	<input type="button" value="book"/>
CAR113	classroom	<input type="button" value="book"/>
CAR410	meeting	<input type="button" value="book"/>

Figure3.1

localhost says:
Booking is successful!

logou

click to choose date: 2017-04-07

Room name	Time	Status	Booked By
CAR212	08:00-09:00	available	<input checked="" type="checkbox"/>
CAR212	09:00-10:00	available	<input checked="" type="checkbox"/>
CAR212	10:00-11:00	available	<input type="checkbox"/>
CAR212	11:00-12:00	available	<input type="checkbox"/>
CAR212	12:00-13:00	available	<input type="checkbox"/>
CAR212	13:00-14:00	available	<input type="checkbox"/>
CAR212	14:00-15:00	available	<input type="checkbox"/>
CAR212	15:00-16:00	available	<input type="checkbox"/>
CAR212	16:00-17:00	available	<input type="checkbox"/>
CAR212	17:00-18:00	available	<input type="checkbox"/>
CAR212	18:00-19:00	available	<input type="checkbox"/>
CAR212	19:00-20:00	available	<input type="checkbox"/>

Figure3.2

click to choose date: 2017-04-07

Room name	Time	Status	Booked By
CAR212	08:00-09:00	unavailable	111111a
CAR212	09:00-10:00	unavailable	111111a
CAR212	10:00-11:00	available	<input type="checkbox"/>
CAR212	11:00-12:00	available	<input type="checkbox"/>
CAR212	12:00-13:00	available	<input type="checkbox"/>
CAR212	13:00-14:00	available	<input type="checkbox"/>
CAR212	14:00-15:00	available	<input type="checkbox"/>
CAR212	15:00-16:00	available	<input type="checkbox"/>
CAR212	16:00-17:00	available	<input type="checkbox"/>
CAR212	17:00-18:00	available	<input type="checkbox"/>
CAR212	18:00-19:00	available	<input type="checkbox"/>
CAR212	19:00-20:00	available	<input type="checkbox"/>

Figure3.3

Room name	Date	Start Time	End Time	
CAR212	2017-04-07	08:00:00	09:00:00	<input type="checkbox"/>
CAR212	2017-04-07	09:00:00	10:00:00	<input type="checkbox"/>
<div><input type="button" value="cancel"/></div>				

Figure3.4

Suppose that I want to book CAR212, I click the book button after CAR212(Figure3.1). The system shows me correct information about

CAR212(Figure3.2). I want to book the room from 8am to 10am. So, I click the first two check box, and then I click the book button. The system shows the successful message(Figure 3.2). And the booking information of CAR212 is updated successfully(Figure3.3). And the booking information about the user is updated correctly too(Figure 3.4). I book the other room of different time, the result is successful. So, the booking function works well.

6.3 Testing the canceling booking function

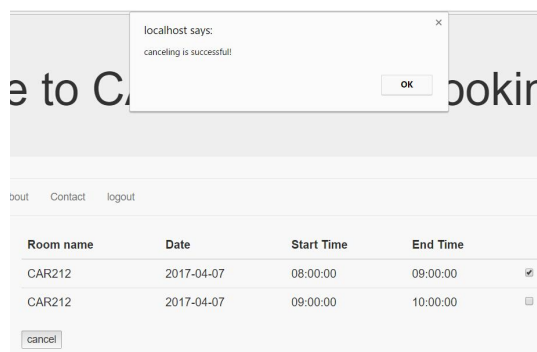


Figure 4.1

Room name	Date	Start Time	End Time	
CAR212	2017-04-07	09:00:00	10:00:00	<input type="checkbox"/>

cancel

Figure4.2

click to choose date: 2017-04-07

Room name	Time	Status	Booked By
CAR212	08:00-09:00	available	<input type="checkbox"/>
CAR212	09:00-10:00	unavailable	111111a
CAR212	10:00-11:00	available	<input type="checkbox"/>
CAR212	11:00-12:00	available	<input type="checkbox"/>
CAR212	12:00-13:00	available	<input type="checkbox"/>
CAR212	13:00-14:00	available	<input type="checkbox"/>
CAR212	14:00-15:00	available	<input type="checkbox"/>
CAR212	15:00-16:00	available	<input type="checkbox"/>
CAR212	16:00-17:00	available	<input type="checkbox"/>
CAR212	17:00-18:00	available	<input type="checkbox"/>
CAR212	18:00-19:00	available	<input type="checkbox"/>
CAR212	19:00-20:00	available	<input type="checkbox"/>

book

Figure4.3

As the Figure3.4 shows, users can see their own booking. Suppose I am the user, I click the first check box and the cancel button. The system shows me the successful message(Figure4.1). And the booking information of the user is updated correctly(Figure 4.2). The Booking information of the room is updated successful too(Figure4.3). The canceling function works well.

6.4 Testing function of deleting and adding resource

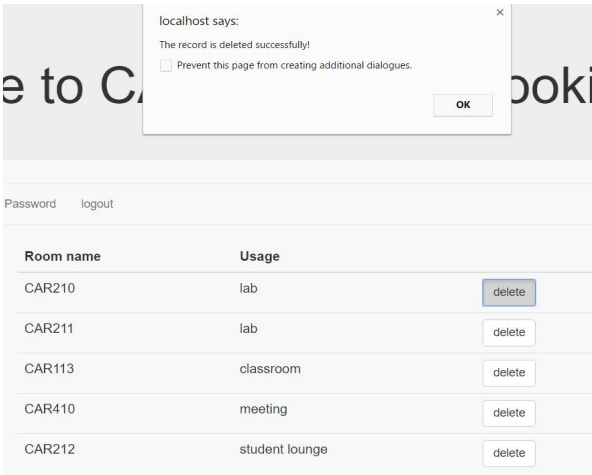


Figure5.1

Room name	Usage	
CAR211	lab	delete
CAR113	classroom	delete
CAR410	meeting	delete
CAR212	student lounge	delete

Figure5.2

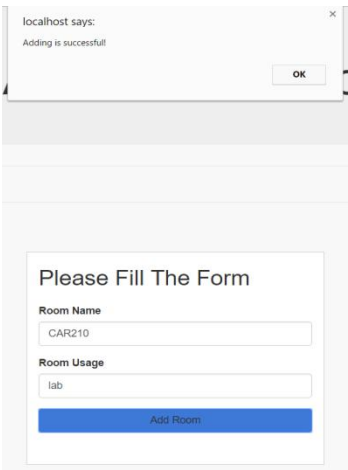


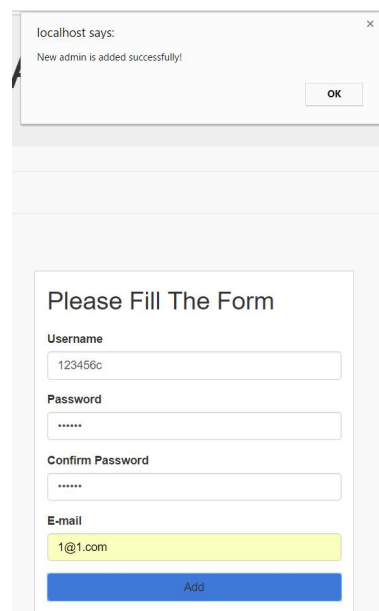
Figure5.3

Room name	Usage	
CAR211	lab	delete
CAR113	classroom	delete
CAR410	meeting	delete
CAR212	student lounge	delete
CAR210	lab	delete

Figure5.4

When I log in as administrator, I see the all resource information. I click the delete in the line of CAR210. The system shows me the successful message (Figure5.1). Then the system shows correct resource information after deleting CAR210(Figure5.2). After I click 'addResource' button, the system shows me the adding form. After I fill the adding form, I click the 'add room' button. The system shows me the successful message(Figure5.3) and the resource information is updated successfully(Figure5.4).

6.5 Testing adding administrator function



The screenshot displays a web application interface. At the top, a modal dialog box titled 'localhost says:' contains the message 'New admin is added successfully!' and an 'OK' button. Below this, the main content area features a form titled 'Please Fill The Form'. The form includes four input fields: 'Username' (containing '123456c'), 'Password' (masked with '*****'), 'Confirm Password' (masked with '*****'), and 'E-mail' (containing '1@1.com'). A blue 'Add' button is positioned at the bottom of the form.

Figure6.1

After I clicked the 'Add Admin' button, I see an adding form. After I fill the form, I click the 'Add' button. The system shows me the successful message. And I use the account I have just created to log in as administrator, the log in process is successful. So, the adding new administrator function works well.

7. Results and Discussion

After the testing on the system, the results shows that the whole system can works well without error. User can register the account and use it to log in the system. When the user log in the system. User can see the correct information of the resource. User can book the resource without creating errors. And User can cancel the booking correctly.

The administrator function works well too. Administrator can log in the admin-system using the administrator account. They can add or delete the resource correctly. They also can add a new administrator.

The testing result shows that the project meets the project objectives and successful criteria and the scope.

8. Conclusion

8.1 Summary

In conclusion, the CAR resource booking system works well. The faculty and staff of computer science can use the system to book the system without creating wrong booking. The system has a navigation bar, the error message and successful message is clear. So, the system has an user friendly interface. The user can understand how to use the system easily. About the mobile device capability, the system is built by the bootstrap framework. The mobile-first is one of the advantages of bootstrap. So, user can launch the system using the mobile device which has a browser compatible with HTML5. It is very convenient for user. But the system is not perfect yet. The system is built without considering any security problem. It can be a big problem. And I think that administrator function is not enough. The administrator should also manage the user's account. The system doesn't have the e-mail confirmation.

8.2 Future Work

As I mentioned in the section 8.1, the system is not perfect. So, the future work will be focus on the problem I mentioned above: security problem, email confirmation and enhancing the administrator function.

1. Using CAPTCHA

CAPTCHA is able to determine whether or not the user is a human. It can prevent the bot create too many accounts.

2. Encrypting user's password

The user's password is not encrypted now. It is insecure. So, I will encrypt the user's password first and then store it into the database.

3. E-mail confirmation. It is very useful to use e-mail confirmation. Because if the user forget the password, they can use the email to get it back or set a new password.

4. Adding some new administrator function.

User may have the bad behavior. For example, user booked a room and didn't cancel it, but he/she didn't use the room at the time he/she booked. So, I think the administrator should have the ability to deny the user's access when they has many bad behavior.

5. UI Problem

The UI does not looks good. I'll spend more time on it to make it look good.

9. Reference

D.L.Silver, Guidelines For Your COMP 4983 Project Report and Presentation [2016]

Braude, Eric J. SOFTWARE ENGINEERING: An Object Oriented Perspective [2001]