

## Secured Restaurant Reservation System using Role-Based Access Control Approach

Nur Hanis Hishamudin<sup>1</sup>, Nurul Azma Abdullah<sup>1\*</sup>

<sup>1</sup>Faculty of Computer Science and Information Technology  
Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, 86400 MALAYSIA

DOI: <https://doi.org/10.30880/aitcs.2022.03.02.011>

Received 15 September 2022; Accepted 06 October 2022; Available online 30 November 2022

**Abstract:** Restaurant reservation is a pre-arranged plan that assists in ensuring the available table and reserved table do not overlap and meals can be prepared in time. However, the existing system might have unauthorized access which lead to information theft. Hence, the Secured Restaurant Reservation System solved the concerns highlighted earlier by using password policy, two-factor authentication and access control to prevent unauthorized. The system uses Prototype Model and contains eight modules which are the sign in and sign-up, table, reservation, course, cart, order, user, and logs module. Next, user testing was performed by 10 respondents including students and staffs of a restaurant. The page that can be access by administrator, staff, and user are different depending on their role. As a result, the system prioritized the confidentiality and integrity concepts by limiting access to the system database and the system made the business more organized and saved much time.

**Keywords:** Password Policy, Restaurant Reservation System, Role-Based Access Control, Two-factor Authentication

### 1. Introduction

Restaurant reservation system is a pre-arranged plan to have a table at a restaurant available. Many restaurants still prefer a conventional style of making reservations by walk-in or phone call. Reservation system depends on staff to take calls, as well as handle systems to organize the process of reserving tables [1]. There are also restaurants that are already using the online system to promote businesses. The system assists the owner in ensuring the available table and reserved table do not overlap and meals can be prepared in time so the customers do not have to wait so long which is a waste of time.

However, the existing system is less concerned about security as it might have unauthorized access and fraudulent activities which lead to information theft and the system allows users to use a weak password. The objectives of the system are to design a secured restaurant reservation system using a web-based approach, develop a secured restaurant reservation system equipped with access control in which limit the user access, and test the functionality of a secured restaurant reservation system in which customers are familiar with interface elements. The system was developed for internet customers to

make an online table reservation. The software used for this project is Sublime Text and the language used is Hypertext Preprocessor (PHP). The system used access control approach which is Role-Based Access Control. Role-Based Access Control (RBAC) is a security method that grants users access to the system based on the role and permissions granted to them. The access control contains three elements which are identification, authentication, and authorization. The system built with two-factor authentication to secure the system. The people involved in the restaurant reservation system includes the administrator, the staff, and the customer.

As a result, the Secured Reservation Restaurant System limits the staff from alter any information from the database. Next, the system also uses a password policy and two-factor authentication to strengthen the security in the sign up and sign in session.

## **2. Literature Review**

Security is the most important problem in an information system. Data security policies and procedures are expressed in terms of confidentiality, integrity, and availability. Confidentiality refers to the ability to verify that only authorized individuals have access, and to keep information from being accessed or denied access with unauthorized people. Next, confidentiality assures all resources are shielded against unauthorized access, ensuring that illegal read access is impossible [2]. It connects to information security since it necessitates access control to protected data. Confidentiality requires that data be safeguarded so that it may only be used for permitted reasons by authorized people. Data integrity refers to the correctness and consistency of information across its entire lifecycle. Integrity includes three purposes that enable accomplish data security which are preventing unauthorized users from altering information, preventing unallowed changes to the information by authorized users, and maintaining internal and external consistency. Furthermore, integrity ensures that the message sent matches the message received so the communication is not tampered with while in transmission. In addition, availability guarantees that authorized users of a system have timely and reliable access to and use of information toward the system. Besides, access control is to provide access data that requires the user to sign into a system using authentication techniques to have access to a system. Access control models are viewed as a system for establishing and guaranteeing the integrity of security policies that govern how information on a system may be accessed and shared. Mandatory Access Control and Discretionary Access Control, and Role Based Access Control are the access control models.

### **2.1 Mandatory Access Control (MAC)**

Mandatory Access Control (MAC) is a part of access control or security mechanism in which the system or database restricts a user's ability to access the information. The administrator may grant users access to the data as the administrator has authority to provide permissions to objects. Only the administrator may alter the object in this model. Mandatory Access Control is more dependable than Discretionary Access Control since it allows for the blocking of information leakage pathways "by memory" [3]. Mandatory Access Control is commonly connected with the BellLaPadula Model.

#### **a. Bell-LaPadula Model**

The Bell–LaPadula Model is a model used to implement access control in military and government activities. The model was established by David Elliott Bell and Leonard J. LaPadula, and it specifies a set of access control rules based on the use of security labels on objects and subject clearances. The Bell–LaPadula concept is based on data secrecy and controlled access to documents which limits information flow from a lower security label to a higher security label to only move upward to avoid jeopardizing information confidentiality [4].

b. Biba Model

The Biba Model is built to prevent people from corrupting data at a level higher than their own or being corrupted by data at a lower level than their own. In general, data integrity preservation has two objectives which are to prevent unauthorized individuals from altering data and maintain uniformity both internally and externally. The model was created in general to handle integrity as a basic value, which is the polar opposite of the Bell–LaPadula paradigm. It specifies that a person can access a document if the document security level is lower than the person's clearance level [5].

2.2. Discretionary Access Control (DAC)

Discretionary Access Control (DAC) is a security approach in which access is granted based on the user's identification. A user in the system is only granted access to a system resource if they are added to an access control list (ACL) linked with the system. The user or group can also give permissions to other users and groups in the same system [6].

a. Access Matrix Model

An Access Control Matrix is a table that connects a set of users' privileges to operate on a group of objects inside a system. The authorization state is defined in the access matrix model as (S, O, A). S represents the group of users that have access, O represents the set of objects with which access can be done, and A represents the access matrix. The matrix is a table containing columns of subjects and rows of objects [7].

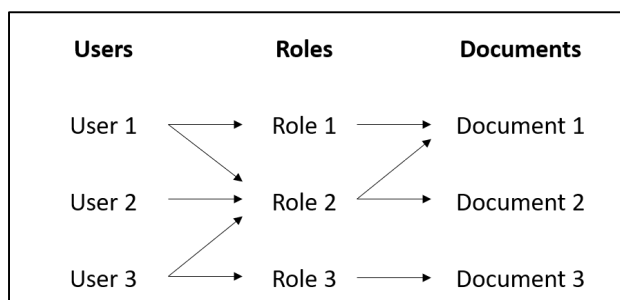
	Document 1	Document 2	Document 3	Program 1
User 1	own read write	read write		execute
User 2	read		read write	
User 3			read	execute read

Figure 1: Access Matrix Model [8]

Figure 1 shows the access matrix model that contains of three processes, three documents, and a program. The first process, User 1 the owner of Document 1 is able to read and write Document 2 and execute Program 1. The second process, User 2 is able to read Document 1 and read and write Document 3. The last process, User 3 is able to read Document 3 and execute and read the Program 1.

2.3 Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is a security mechanism in which access to system resources is granted depending on the role and privileges assigned to a user by the administrator. It is a rule access-control technique that revolves on roles and privileges. RBAC components such as role-permissions, user-role and role-role connections make user tasks straightforward. Although Role-Based Access Control (RBAC) differs from Mandatory Access Control (MAC) and Discretionary Access Control (DAC) access control structures, it can easily implement these restrictions.



**Figure 2: Role-Based Access Control Model**

Figure 2 shows the Role-Based Access Control model. There are three users, three roles, and three documents. User 1 has two roles which are Role 1 and Role 2. User 1 as Role 1 can access Document 1 while as Role 2 can access Document 1 and Document 2. For User 2 has one role which is Role 2 that can access Document 1 and Document 2. As User 3 who has two roles which are Role 2 and Role 3. User 3 as Role 2 can access Document 1 and Document 2 while Role 3 can access Document 3.

## 2.4 Analysis of the Existing System

This section explains three restaurant reservation websites, which are Dineplan, Eatigo, and OpenTable.

### 2.4.1 Dineplan

This system consists of four modules which are login module, verification module, reservation module, and administrator module. Login module can be done by both administrator and the users. The signup session is to register a user that requires the user to enter first name, last name, email, country code, mobile, and password. Verification module purpose is to validate the user by sending verification code to the user's mobile number. Reservation module is to allow the user to book a table by choosing date, pax, and time. Administrator module is for updating, and deleting user, reservation, and menu.

### 2.4.2 Eatigo

This system consists of three modules which are register and login module, reservation module, and administrator module. Login module can be done by both administrator and the users. The register session is to register a user that requires the user to enter name, email, password, and re-type password. Reservation module is to allow the user to book a table by choosing date, pax, and time. Administrator module is for updating, and deleting user, reservation, and menu.

### 2.4.3 OpenTable

This system consists of four modules which are login module, verification module, reservation module, and administrator module. Login module can be done by both administrator and the users. The signup session is to register a user that requires the user to enter first name, last name, email, password, re-enter password and primary dining location. Reservation module is to allow the user to book a table by choosing date, pax, time, and choosing location. Administrator module is for updating, and deleting user, reservation, and menu.

### 2.4.4 Comparison of existing system with proposed system

The dineplan, eatigo, and OpenTable are among the reviewed systems, sharing a few similarities and contrasts in certain features. As a result, the differences may be used to enhance the developed system. Sign-up, sign-in, two-factor authentication, password policy, easy access, and password reset are the six aspects that are studied in these comparisons. The table below displays the comparisons between the equivalent and developed systems.

**Table 1: Comparison of existing systems with developed system**

Security Feature	dineplan	eatigo	OpenTable	Developed System
Sign up	Yes	Yes	Yes	Yes
Sign in	Yes	Yes	Yes	Yes
Two-factor Authentication	Yes	No	No	Yes
Password Policy	No	No	No	Yes
Password Reset	Yes	Yes	Yes	Yes

Based on Table 1, dineplan does not use password policy but it is not easy to access. eatigo and OpenTable are not using two-factor authentication and password policy. The developed system has all security features.

### 3. Methodology

In this project, Prototype Model was chosen as a methodology that consists of five phases which are planning, analysis, design, prototype, and implementation and testing phase.

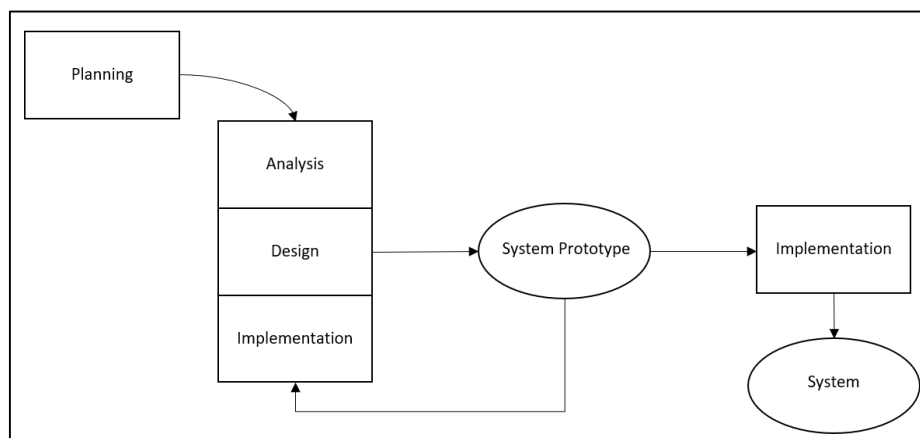


Figure 3: Prototype Model [9]

Based on the developed system, in the planning phase, the problem of the existing system, the objectives of the developed system, and study scope are specified. Furthermore, a planned timetable for the next phase to operate smoothly has been prepared to make the developed system development project to be successful.

In the analysis phase, an analysis of the existing system and its requirements is carried out to ensure the development of a developed system more planned and systematic. Furthermore, the proper programming language, software, and modules analyzed for the developed system.

In the design phase, context diagrams (CD), data flow diagrams (DFD), and system architecture were used to represent the processes in developed system. In addition, this phase creates a database using entity relationship diagrams (ERD), and data dictionaries.

In the prototype phase, user interfaces and the database were designed for the developed system. However, if any weaknesses or user requirements are identified at this phase, the process will be repeated until the developed system can function correctly and accepted by users.

In the implementation and testing phase, the prototype model of the developed system will be available as a complete system and accepted in real-time. Next, user testing is carried out to ensure that the system corresponds to the demands and needs of the users who will use it. The test case was performed by 10 respondents including students and staffs of a restaurant in Johor Bahru.

### 4. Analysis and Design

This section explains the designs of the system.

#### 4.1 Functional and Non-functional requirements

##### a. Functional requirement

The developed system illustrates all operations that the system can execute in each module as shown in Table 2. Based on Table 2, there are eight modules which are the sign-in and sign-up, table, reservation, course, cart, order, user, and logs module. Each module explains the functionalities that should be accomplished to fulfil the requirements of the system.

**Table 2: Functional Requirement of the system**

Module	Functionalities
Sign up and Sign in	<ul style="list-style-type: none"> <li>- Administrator register staff into the system.</li> <li>- Customer register an account by using username, email, NRIC, mobile number, PIN, password, and confirm password.</li> <li>- Customer signs in to the system by using email and password as registered.</li> </ul>
Table	<ul style="list-style-type: none"> <li>- Administrator decide which table should be choose for the customer.</li> </ul>
Reservation	<ul style="list-style-type: none"> <li>- Staff updates table availability status.</li> <li>- Customer reserves a table.</li> <li>- Customer views the courses and make orders.</li> <li>- Administrator adds, updates and deletes the courses.</li> <li>- Staff view the reservation.</li> <li>- Administrator views, adds, deletes the reservation.</li> </ul>
Course	<ul style="list-style-type: none"> <li>- Administrator to add, update, and delete the courses.</li> <li>- Customer view available course.</li> </ul>
Cart	<ul style="list-style-type: none"> <li>- Customer add to cart the desired courses before ordering.</li> <li>- Administrator to view cart course added by the customer.</li> </ul>
Order	<ul style="list-style-type: none"> <li>- Customer order the course by filling ordering form.</li> <li>- Administrator view the orders made by customers.</li> </ul>
User	<ul style="list-style-type: none"> <li>- Administrator view, add, update, and delete users if necessary.</li> </ul>
Logs	<ul style="list-style-type: none"> <li>- Administrator monitor activities performed by staff and customer.</li> </ul>

##### b. Non-functional requirement

The non-functional requirements which explain the performance, operational, and security of the developed system. Table 3 shows the non-functional requirements of the developed system.

**Table 3: Non-functional Requirement of the system**

Module	Functionalities
Performance	<ul style="list-style-type: none"> <li>- System interface must be familiar to the users.</li> <li>- Navigations, menus, and buttons should be easy to use and designed in the right places.</li> </ul>
Operational	<ul style="list-style-type: none"> <li>- System is only accessible when an internet connection is available.</li> <li>- System is compatible with multiple web browsers.</li> </ul>
Security	<ul style="list-style-type: none"> <li>- Customer must first sign up</li> <li>- All users must sign in with the email and password.</li> <li>- Password must be at least 8 characters long, contain capital and lowercase letters, special characters, and numerical digits to construct a strong password</li> <li>- Users must be verified by entering their registered PIN.</li> </ul>

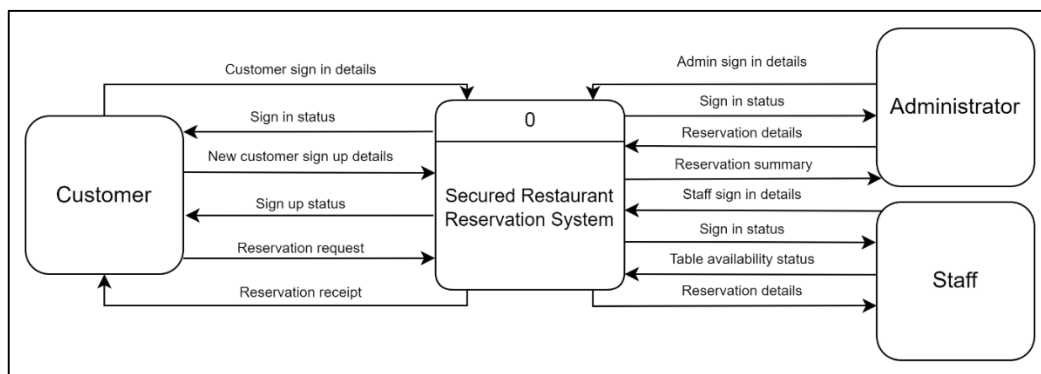
Based on Table 3, the performance is what the users can do when using the system, the operational is how the user can access the system, and security is what is the feature that secures the system and all the information.

#### 4.2 System Analysis

System analysis is an overview of the system to be developed.

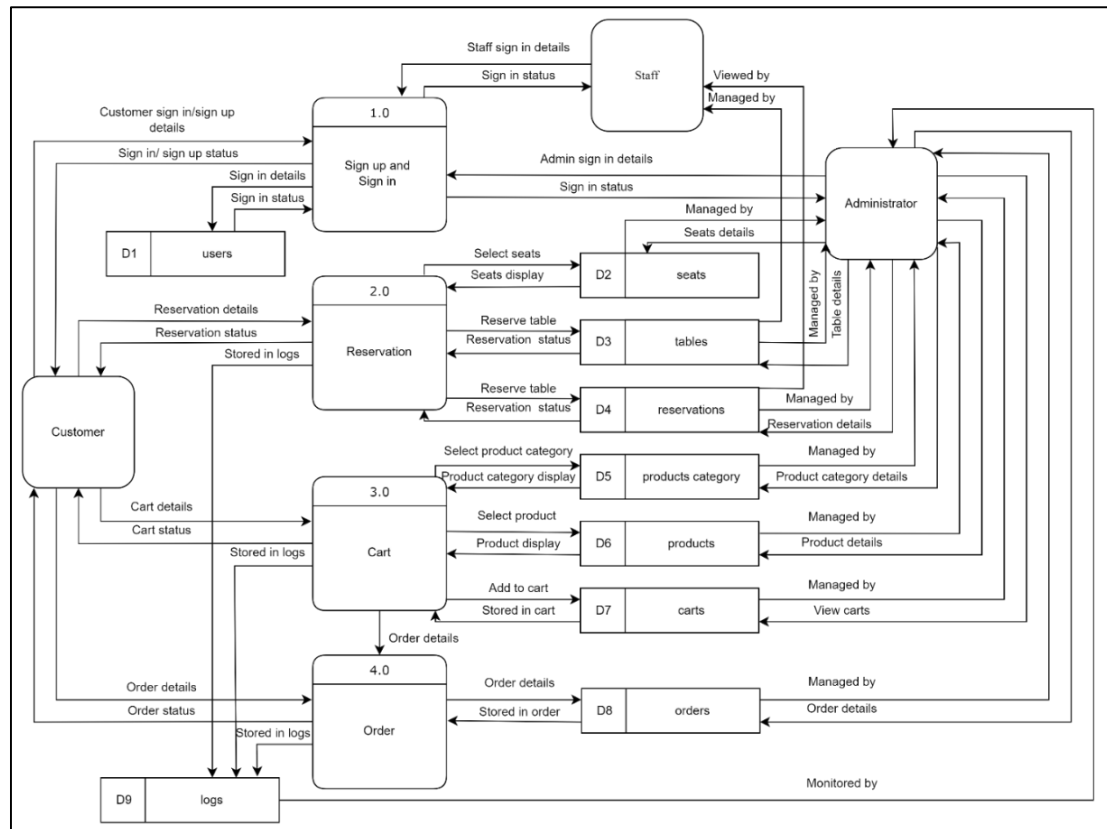
##### 4.2.1 Data Flow Diagram (DFD)

This section illustrates the data flow diagram which contains representation of a data flow that follows a specified structure. Entities, processes, data storage, and data flow are the elements of the DFD design. Figure 4 shows the data flow begins with the customer.



**Figure 4: Context Diagram**

Figure 4 shows the data flow begins with the customer. New customer needs to sign up for an account while the current customer needs to sign in to the system to make a reservation. In addition, the context diagram above also shows that restaurant administrator and staff need to access before using the system where administrator can view reservation details and summary by customers while the staff can view reservation details and manage the table availability status.



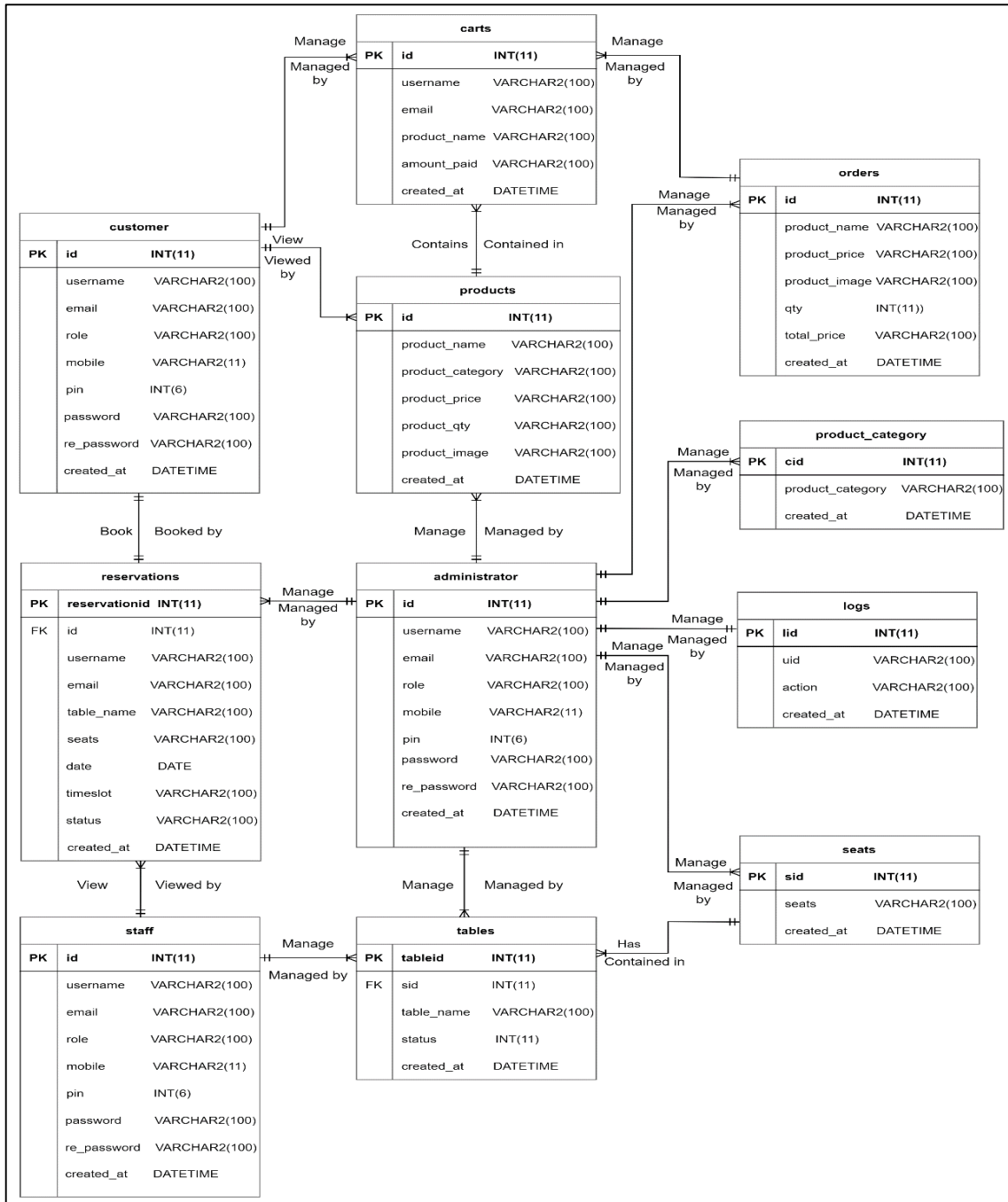
**Figure 5: Level 0 Data Flow Diagram**

Figure 5 shows the level 0 data flow diagram is a breakdown of the data flow from the context diagram. It provides a clearer overview of the processes in this system here. There are nine tables which are users, seats, tables, reservations, product category, products, carts, orders, and logs. Customers, staff, and administrator need to enter their details to sign in to the system. For new customers, they need to sign up before signing in. After signing in successfully, customers may reserve a table, add to cart the course, and order it. Administrator can manage seats, tables, reservation, product category, products, carts, orders, and logs. Staff can manage table availability status and view reservations.

#### 4.2.2 Entity Relationship Diagram (ERD)

The entity connection diagram was created to provide a more detailed picture of the structure of the database created for the Secured Restaurant Reservation System. The diagram is constructed from the data repository. Entity relationship diagram (ERD) has three main elements which are entities, attributes and relationships. The Entity Relationship Diagram (ERD) is illustrated as in Figure 6.





**Figure 6: Entity Relationship Diagram**

Figure 6 depicted the Entity Relationship Diagram of the Secured Restaurant Reservation System. The database of the system has seven entities which are users, seats, tables, reservation, product category, product, carts, orders, bills, and logs table. Based on the figure, customer can view the courses and reserve a table. Next, the admin can manage the course category, courses, and reservation. The staff can view reservations and manage the table availability.

### 4.3 Implementation

The development process of this system used two software which are Sublime Text and XAMPP. Sublime Text is used to develop and design the interface of each page such as sign in and sign-up, table, reservation, course, cart, order, user, and logs module. Furthermore, XAMPP is used to execute SQL

queries capable of saving all data connected to the modules created by linking the user interface and the database.

#### 4.3.1 Sign Up

Appendix 1, 2, and 3 depict sign up modules which is the account registration interface and a part of users sign up programming code. Customers who do not have an account must create one by providing a username, email address, mobile number, Personal Identification Number (PIN), password, and confirm password. Once registration is complete, customer can sign in the account.

#### 4.3.2 Sign In

Appendix 4 and 5 are the sign in modules showing the sign in interface and a part of user sign in programming code. Users sign in by entering the email and password they have registered.

#### 4.3.3 Verify

Appendix 6 and 7 are the verify interface and a part of verify programming code. Users verify by entering the correct PIN they have registered.

#### 4.3.4 Change Password

Appendix 8 and 9 are the change password interface and a part of change password programming code. Users can change password by entering the old password, new password, and confirm new password.

#### 4.3.5 Table Reservation

The table reservation interface and a piece of the table reservation code are depicted in Appendix 10, 11, and 12. On this page, customer can book a reservation by selecting a date and time from a virtual calendar. Furthermore, customer can only book on dates and times when there are vacancies, which are denoted by a green square button with the phrase "Book" on it.

#### 4.3.6 Course

The course interface and a part of the code are depicted in Appendix 13 and 14. This page allows customer to view or to order the course by clicking "Add to Cart" button.

#### 4.3.7 Cart

The cart interface and a part of the code are depicted in Appendix 15 and 16. This page allows customer to view the cart item. Customer can proceed to the checkout page if they wish to order the cart item.

#### 4.3.8 Checkout

The checkout interface and a part of the code are depicted in Appendix 17 and 18. This page allows customer to order the cart item by filling the ordering form. After ordering, the information will be delivered to administrator.

#### 4.4 Test Case

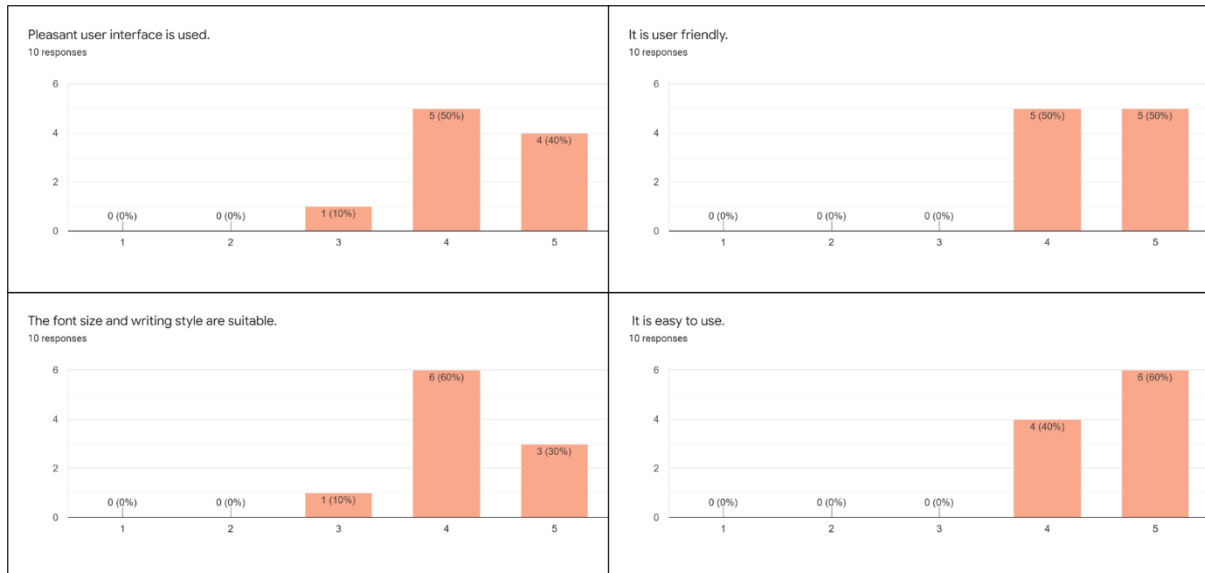
The test case used to explain the intended outcome of a function and as a guide in finding error as the test execute. The Secured Restaurant Reservation System test cases are listed in Table 4. Based on Table 4, there are five test cases which are sign in and sign up, reservation, course, cart, checkout, and access control module with all the expected result.

**Table 4: Test Case**

Test Case	Expected Result	Result
Sign In and Sign Up	- Users can sign in the system into different interfaces based on their user type or role.	Success
Reservation	- Customer can make reservation by selecting date, timeslot and number of seats needed.	Success
	- Administrator can manage details stored in the database.	Success
Course	- Customer can view courses displays in the course page.	Success
	- Administrator can add, update or delete course in the database	Success
Cart	- Customer can view courses displays in the course page.	Success
	- Administrator can add, update or delete course in the database.	Success
Checkout	- Customer can order the course in checkout page.	Success
	- Administrator can view order in the database.	Success
Access Control	- Users can access the system into different page based on their role.	Success
	- Administrator can access all page while staff can only access certain page.	Success

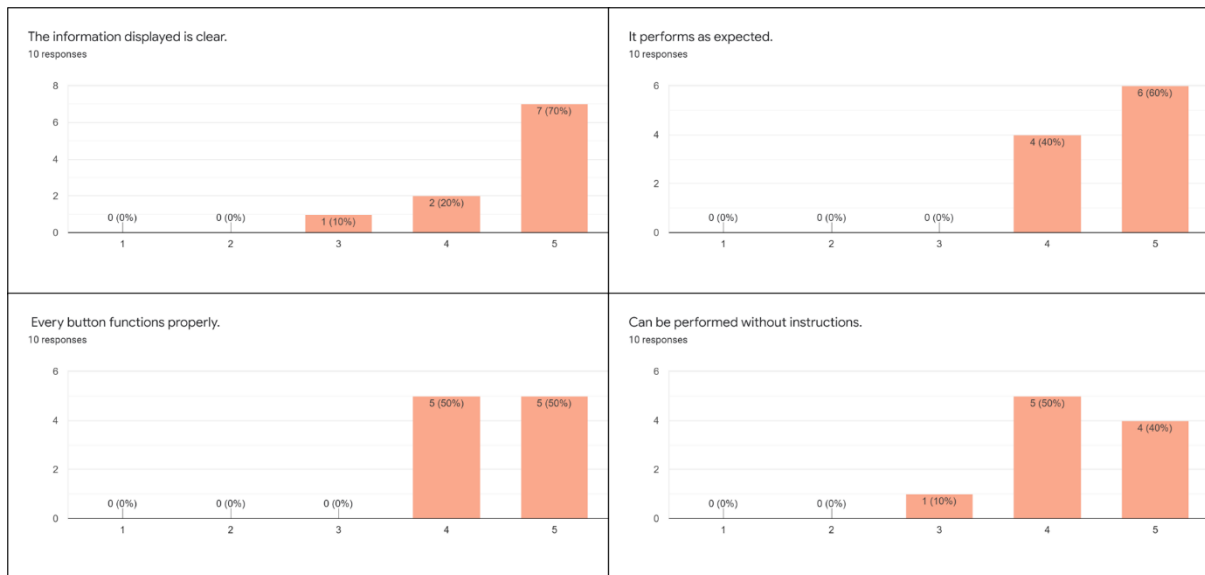
#### 4.5 Test Case Result

User satisfaction testing is carried out to confirm that the developed system meets the desired goals. Furthermore, this testing is done to acquire opinions and levels of user satisfaction with the established system. Furthermore, user satisfaction testing is separated into two sections which are system interface and system functioning. The test results for each system interface-related question are shown in Figure 7.



**Figure 7: Graph of Test Results for System Interface**

Based on Figure 7, the testing section for the system interface contains 4 questions. Regarding the pleasant user interface is used, 4 respondents chose highly agree, 5 respondents chose agree, and 1 respondent chose neutral. Next, for the second question, the font size and writing style are suitable, 3 respondents chose highly agree, 6 agreed, and 1 chose neutral. Concerning user friendly, 5 respondents chose highly agree and 5 respondents chose agree. For the easy-to-use system interface, 6 respondents chose highly agree and 4 respondents chose agree. According to the findings of the system interface testing, all respondents were satisfied with the system interface designed



**Figure 8: Graph of Test Results for System Functioning**

Figure 8 depicts the system function test results. In this second section there are 4 questions. For the first question, 7 respondents chose strongly agree, 2 respondents chose agree and 1 respondent chose neutral for the information displayed clearly. Next, for the second question which is every button functions properly, 5 respondents chose strongly agree and 5 respondents chose. For the third question 6 respondents chose strongly agree and 4 respondents chose agree for the question system performs as expected. In addition, for the last question 4 respondents chose strongly agree, 5 respondents chose agree and 1 respondent chose neutral for the question of Can be performed without instructions. According to the findings of system function testing, all respondents are satisfied with the system function developed.

## 5. Conclusion

In conclusion, the development of restaurant reservation system will make customers reserve table much easier. Furthermore, it can help Secured Restaurant Reservation System manage tables and orders more systematically. This is due to the fact that improper booking management may produce confusion in everyday management. Therefore, with the Secured Restaurant Reservation System, restaurant reservation system will be more organized and structured. Secured Restaurant Reservation System using Role-Based Access Control Approach developed is secured when using access control method.

## Acknowledgement

The authors would also like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

## Appendix

Appendix 1: Account Sign Up Interface

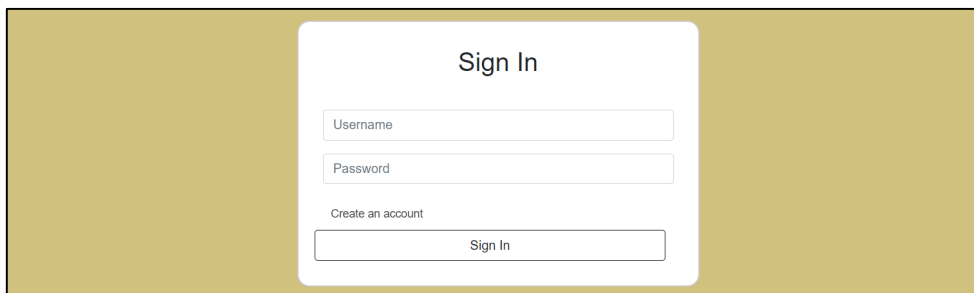
```

5  if (isset($_POST['username']) && isset($_POST['name']) && isset($_POST['email']) && isset($_POST['mobile']) && isset($_POST['role']) &&
6  isset($_POST['pin']) && isset($_POST['password']) && isset($_POST['re_password'])){
7      function validate($data){
8          $data = trim($data);
9          $data = stripslashes($data);
10         $data = htmlspecialchars($data);
11         return $data;
12     }
13     $username = validate($_POST['username']);
14     $name = validate($_POST['name']);
15     $email = validate($_POST['email']);
16     $mobile = validate($_POST['mobile']);
17     $role = validate($_POST['role']);
18     $pin = validate($_POST['pin']);
19     $password = validate($_POST['password']);
20     $re_password = validate($_POST['re_password']);
21     $user_data = 'username=' . $username . '&name=' . $name;
22
23     if (empty($username) && empty($name) && empty($email) && empty($mobile) && empty($role) && empty($pin) && empty($password)
24         && empty($re_password)){
25         if (mysql_num_rows($result) > 0){
26             header("Location: signup.php?error=The username is taken. Try another&$user_data");
27             exit();
28         }
29     }
30     if (!preg_match("/^[a-zA-Z ]*$/", $username)) {
31         header("Location: signup.php?error=Only letters and white space allowed&$user_data");
32         exit();
33     }
34     if (!preg_match("/^[a-zA-Z ]*$/", $name)) {
35         header("Location: signup.php?error=Only letters and white space allowed&$user_data");
36         exit();
37     }
38     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
39         header("Location: signup.php?error=Email format is invalid&$user_data");
40         exit();
41     }
42     if (!preg_match("/^[0-9]{11}$/", $mobile)) {
43         header("Location: signup.php?error=Only 11-digit mobile numbers allowed&$user_data");
44         exit();
45     }
46     if (!preg_match("/^[0-9]{6}$/", $pin)) {
47         header("Location: signup.php?error=Only 6-digit PIN allowed&$user_data");
48         exit();
49     }
50     if (!preg_match("/^(?=.*[!@#-_%&+=$!~?])(?=.*[a-z])(?=.*[A-Z])[0-9A-Za-z@#_-_%&+=$!~?]{8,20}$/", $password)) {
51         header("Location: signup.php?error=Password should be between 8 to 20 characters long, contains at least one special
52         character, lowercase, uppercase and a digit&$user_data");
53         exit();
54     }
55     if ((preg_match("/^[a-zA-Z ]*$/", $username) && (preg_match("/^[a-zA-Z ]*$/", $name) && (filter_var($email,
56         FILTER_VALIDATE_EMAIL) && (preg_match("/^[0-9]{11}$/", $mobile) && (preg_match("/^[0-9]{6}$/", $pin)) && (preg_match(
57         "/^(?=.*[!@#-_%&+=$!~?])(?=.*[a-z])(?=.*[A-Z])[0-9A-Za-z@#_-_%&+=$!~?]{8,20}$/", $password)))){
58         $password = md5($password);
59         $re_password = md5($re_password);
60         $pin = md5($pin);
    
```

Appendix 2: Account Sign Up Programming Code

```
55      $sql = "INSERT INTO users(username, password, re_password, name, pin, email, mobile, role) VALUES('$username', '$
56      password', '$re_password', '$name', '$pin', '$email', '$mobile', '$role)";
57      $result = mysqli_query($con, $sql);
58
59      if ($result){
60          header("Location: signup.php?success=Your account has been created successfully");
61          exit();
62      }else{
63          header("Location: signup.php?error=Unknown error occurred&$user_data");
64          exit();
65      }
66  }
67  }else{
68      if (empty($username)) {
69          header("Location: signup.php?error=Username is required&$user_data");
70          exit();
71      }
72      else if (empty($name)){
73          header("Location: signup.php?error=Name is required&$user_data");
74          exit();
75      }
76      else if (empty($email)){
77          header("Location: signup.php?error=Email is required&$user_data");
78          exit();
79      }
80      else if (empty($mobile)){
81          header("Location: signup.php?error=Mobile is required&$user_data");
82          exit();
83      }
84      else if (empty($role)){
85          header("Location: signup.php?error=Click Me&$user_data");
86          exit();
87      }
88      else if (empty($password)){
89          header("Location: signup.php?error=Password is required&$user_data");
90          exit();
91      }
92      else if (empty($re_password)){
93          header("Location: signup.php?error=Confirm password is required&$user_data");
94          exit();
95      }
96      else if ($password != $re_password){
97          header("Location: signup.php?error=Passwords does not match&$user_data");
98          exit();
99      }
100     else if (empty($pin)){
101         header("Location: signup.php?error=PIN is required&$user_data");
102         exit();
103     }
104 }
105 }else{
106     header("Location: signup.php");
107     exit();
108 }
```

### Appendix 3: Account Sign Up Programming Code



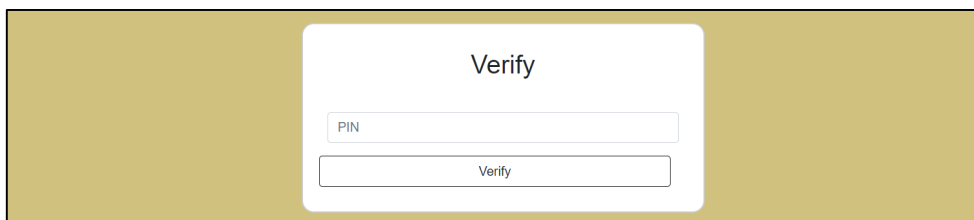
### Appendix 4: Account Sign In Interface

```

4
5 if (isset($_POST['username']) && isset($_POST['password'])) {
6     function validate($data){
7         $data = trim($data);
8         $data = stripslashes($data);
9         $data = htmlspecialchars($data);
10        return $data;
11    }
12
13    $username = validate($_POST['username']);
14    $password = validate($_POST['password']);
15
16    if (empty($username)) {
17        header("Location: index.php?error=Username is required");
18        exit();
19    }else if (empty($password)){
20        header("Location: index.php?error=Password is required");
21        exit();
22    }else{
23        // hashing the password
24        $password = md5($password);
25
26        $sql = "SELECT * FROM users WHERE username='$username' AND password='$password'";
27
28        $result = mysqli_query($con, $sql);
29
30        if (mysqli_num_rows($result) == 1) {
31            $row = mysqli_fetch_assoc($result);
32            if ($row['username'] == $username && $row['password'] == $password) {
33                $_SESSION['username'] = $row['username'];
34                $_SESSION['name'] = $row['name'];
35                $_SESSION['id'] = $row['id'];
36                header("Location: verify.php");
37                exit();
38            }else{
39                header("Location: index.php?error=Incorrect Username or password");
40                exit();
41            }
42        }else{
43            header("Location: index.php?error=Incorrect Username or password");
44            exit();
45        }
46    }
47 }
48 }else{
49     header("Location: index.php");
50     exit();
51 }

```

Appendix 5: Account Sign In Programming Code



Appendix 6: Account Verify Interface

```

5
6 if (isset($_POST['pin'])){
7     function validate($data){
8         $data = trim($data);
9         $data = stripslashes($data);
10        $data = htmlspecialchars($data);
11        return $data;
12    }
13    $pin = validate($_POST['pin']);
14    if (empty($pin)){
15        header("Location: verify.php?error=PIN is required");
16        exit();
17    }else{
18        $pin = md5($pin);
19        $sql = "SELECT * FROM users WHERE pin='$pin' ";
20        $result = mysqli_query($con, $sql);
21        if (mysqli_num_rows($result) == 1){
22            $row = mysqli_fetch_assoc($result);
23            if ($row['pin'] == $pin){
24                if ($row['role'] == "Admin"){
25                    $_SESSION['Admin'] = $row['pin'];
26                    $_SESSION['role'] = $row['role'];
27                    header("Location: admin/dashboard.php");
28                    exit();
29                }elseif ($row['role'] == "Staff"){
30                    $_SESSION['Staff'] = $row['pin'];
31                    $_SESSION['role'] = $row['role'];
32                    header("Location: staff/dashboard.php");
33                    exit();
34                }elseif ($row['role'] == "User"){
35                    $_SESSION['User'] = $row['pin'];
36                    $_SESSION['role'] = $row['role'];
37                    header("Location: customer/about.php");
38                    exit();
39                }
40            }else{
41                header("Location: verify.php?error=Incorrect PIN");
42                exit();
43            }
44        }else{
45            header("Location: verify.php?error=Incorrect PIN");
46            exit();
47        }
48    }
49 }else{
50     header("Location: verify.php");
51     exit();
52 }

```

Appendix 7: Account Verify Programming Code

**Change Password**

Old Password

New Password

Confirm New Password

**Appendix 8: Change Password Interface**

```

4  if (isset($_SESSION['id']) && isset($_SESSION['username'])) {
5      include 'connect.php';
6      if (isset($_POST['op']) && isset($_POST['np'])
7          && isset($_POST['c_np'])) {
8          function validate($data) {
9              $data = trim($data);
10             $data = stripslashes($data);
11             $data = htmlspecialchars($data);
12             return $data;
13         }
14         $op = validate($_POST['op']);
15         $np = validate($_POST['np']);
16         $c_np = validate($_POST['c_np']);
17         if (empty($op)) {
18             header("Location: change-password.php?error=Old Password is required");
19             exit();
20         } else if (empty($np)) {
21             header("Location: change-password.php?error=New Password is required");
22             exit();
23         } else if ($np !== $c_np) {
24             header("Location: change-password.php?error=The confirmation password does not match");
25             exit();
26         } else {
27             $op = md5($op);
28             $np = md5($np);
29             $id = $_SESSION['id'];
30             $sql = "SELECT password FROM users WHERE id='$id' AND password='$op'";
31             $result = mysqli_query($con, $sql);
32             if (mysqli_num_rows($result) == 1) {
33                 $sql_2 = "UPDATE users SET password='$np' WHERE id='$id'";
34                 mysqli_query($con, $sql_2);
35                 header("Location: change-password.php?success=Your password has been changed successfully");
36                 exit();
37             } else {
38                 header("Location: change-password.php?error=Incorrect password");
39                 exit();
40             }
41         }
42     }
43 } else {
44     header("Location: change-password.php");
45     exit();
46 }
47 } else {
48     header("Location: index.php");
49     exit();
50 }
    
```

**Appendix 9: Change Password Programming Code**

July 2022

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
					1 <input type="button" value="Not Available"/>	2 <input type="button" value="Close"/>
3 <input type="button" value="Not Available"/>	4 <input type="button" value="Not Available"/>	5 <input type="button" value="Book"/>	6 <input type="button" value="Book"/>	7 <input type="button" value="Book"/>	8 <input type="button" value="Book"/>	9 <input type="button" value="Close"/>
10 <input type="button" value="Book"/>	11 <input type="button" value="Book"/>	12 <input type="button" value="Book"/>	13 <input type="button" value="Book"/>	14 <input type="button" value="Book"/>	15 <input type="button" value="Book"/>	16 <input type="button" value="Close"/>
17 <input type="button" value="Book"/>	18 <input type="button" value="Book"/>	19 <input type="button" value="Book"/>	20 <input type="button" value="Book"/>	21 <input type="button" value="Book"/>	22 <input type="button" value="Book"/>	23 <input type="button" value="Close"/>
24 <input type="button" value="Book"/>	25 <input type="button" value="Book"/>	26 <input type="button" value="Book"/>	27 <input type="button" value="Book"/>	28 <input type="button" value="Book"/>	29 <input type="button" value="Book"/>	30 <input type="button" value="Close"/>
31 <input type="button" value="Book"/>						

**Appendix 10: Table Reservation Date Selection Interface**





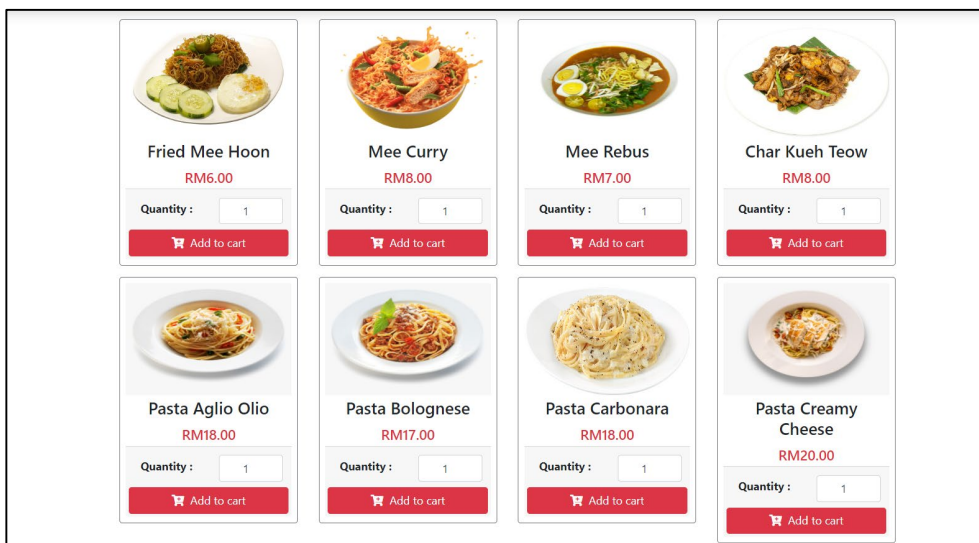
Appendix 11: Table Reservation Timeslot Selection Interface

```

3  if(isset($_GET['date'])){
4      $date = $_GET['date'];
5      $stmt = $mysqli->prepare("SELECT * FROM reservations WHERE date = ?");
6      $stmt->bind_param('s', $date);
7      $reservations = array();
8      if($stmt->execute()){
9          $result = $stmt->get_result();
10         if ($result->num_rows>0){
11             while ($row = $result->fetch_assoc()) {
12                 $reservations[] = $row['timeslot'];
13             }
14             $stmt->close();
15         }
16     }
17 }
18 if(isset($_POST['submit'])){
19     $username = $_POST['username'];
20     $email = $_POST['email'];
21     $timeslot = $_POST['timeslot'];
22     $seats = $_POST['seats'];
23     $stmt = $mysqli->prepare("SELECT * FROM reservations WHERE date = ? AND timeslot=?");
24     $stmt->bind_param('ss', $date, $timeslot);
25     if($stmt->execute()){
26         $result = $stmt->get_result();
27         if($result->num_rows>0){
28             $msg = "<div class='alert alert-danger'>Already Booked</div>";
29         }else{
30             $stmt = $mysqli->prepare("INSERT INTO reservations (username, timeslot, email, date, seats) VALUES (?, ?, ?, ?, ?)");
31             $stmt->bind_param('sssss', $username, $timeslot, $email, $date, $seats);
32             $stmt->execute();
33             $msg = "<div class='alert alert-success'>Booking Successful</div>";
34             $reservations[] = $timeslot;
35             $stmt->close();
36             $mysqli->close();
37         }
38     }
39 }
40 $duration = 30;
41 $cleanup = 0;
42 $start = "09:00";
43 $end = "24:00";
44 function timeslots($duration, $cleanup, $start, $end){
45     $start = new DateTime($start);
46     $end = new DateTime($end);
47     $interval = new DateInterval("PT".$duration."M");
48     $cleanupInterval = new DateInterval("PT".$cleanup."M");
49     $slots = array();
50     for($intStart = $start; $intStart<$end; $intStart->add($interval)->add($cleanupInterval)){
51         $endPeriod = clone $intStart;
52         $endPeriod->add($interval);
53         if($endPeriod->$end){
54             break;
55         }
56         $slots[] = $intStart->format("H:iA")." - ". $endPeriod->format("H:iA");
57     }return $slots;
58 }

```

Appendix 12: Table Reservation Programming Code



Appendix 13: Course Interface

```

$(document).ready(function() {
    $(".addItemBtn").click(function(e) {
        e.preventDefault();
        var $form = $(this).closest(".form-submit");
        var pid = $form.find(".pid").val();
        var pname = $form.find(".pname").val();
        var pprice = $form.find(".pprice").val();
        var pimage = $form.find(".pimage").val();
        var pcode = $form.find(".pcode").val();

        var pqty = $form.find(".pqty").val();

        $.ajax({
            url: 'action.php',
            method: 'post',
            data: {
                pid: pid,
                pname: pname,
                pprice: pprice,
                pqty: pqty,
                pimage: pimage,
                pcode: pcode
            },
            success: function(response) {
                $("#message").html(response);
                window.scrollTo(0, 0);
                load_cart_item_number();
            }
        });
        load_cart_item_number();

        function load_cart_item_number() {
            $.ajax({
                url: 'action.php',
                method: 'get',
                data: {
                    cartItem: "cart_item"
                },
                success: function(response) {
                    $("#cart-item").html(response);
                }
            });
        }
    });
});

```

Appendix 14: Course Programming Code

PRODUCTS IN YOUR CART!						
ID	Image	Product	Price	Quantity	Total Price	Clear All
19		Spaghetti Creamy Cheese	RM22.00	<input type="text" value="1"/>	RM22.00	
20		Orange Juice	RM7.00	<input type="text" value="1"/>	RM7.00	
<b>Grand Total</b>					<b>RM29.00</b>	

[Continue Shopping](#)

Appendix 15: Cart Interface

```

71 $(document).ready(function() {
72
73     // Send product details in the server
74     $(".addItemBtn").click(function(e) {
75         e.preventDefault();
76         var $form = $(this).closest(".form-submit");
77         var pid = $form.find(".pid").val();
78         var pname = $form.find(".pname").val();
79         var pprice = $form.find(".pprice").val();
80         var pimage = $form.find(".pimage").val();
81         var pcode = $form.find(".pcode").val();
82
83         var pqty = $form.find(".pqty").val();
84
85         $.ajax({
86             url: 'action.php',
87             method: 'post',
88             data: {
89                 pid: pid,
90                 pname: pname,
91                 pprice: pprice,
92                 pqty: pqty,
93                 pimage: pimage,
94                 pcode: pcode
95             },
96             success: function(response) {
97                 $("#message").html(response);
98                 window.scrollTo(0, 0);
99                 load_cart_item_number();
100             }
101         });
102     });
103
104     // Load total no.of items added in the cart and display in the navbar
105     load_cart_item_number();
106
107     function load_cart_item_number() {
108         $.ajax({
109             url: 'action.php',
110             method: 'get',
111             data: {
112                 cartItem: "cart_item"
113             },
114             success: function(response) {
115                 $("#cart-item").html(response);
116             }
117         });
118     }
119 });

```

Appendix 16: Cart Programming Code

Complete your order!

Item(s) : Pasta Carbonara(4)  
**Total Amount Payable : RM80.00**

Place Order

### Appendix 17: Checkout Interface

```

$(document).ready(function() {
    // Change the item quantity
    $("#itemQty").on('change', function() {
        var $el = $(this).closest("tr");
        var pid = $el.find("#pid").val();
        var pprice = $el.find("#pprice").val();
        var qty = $el.find("#itemQty").val();
        location.reload(true);
        $.ajax({
            url: 'action.php',
            method: 'post',
            cache: false,
            data: {
                qty: qty,
                pid: pid,
                pprice: pprice
            },
            success: function(response) {
                console.log(response);
            }
        });
    });

    // Load total no. of items added in the cart and display in the navbar
    load_cart_item_number();

    function load_cart_item_number() {
        $.ajax({
            url: 'action.php',
            method: 'get',
            data: {
                cartItem: "cart_item"
            },
            success: function(response) {
                $("#cart-item").html(response);
            }
        });
    }
});
    
```

### Appendix 18: Checkout Programming Code

#### References

- [1] B. J. Gregorash, "Restaurant revenue management: apply reservation management?," *Inf. Technol. Tour.*, vol. 16, pp. 331–346, 2016, doi: 10.1007/s40558-016-0065-0.
- [2] R. M. Karima Echihabi, *Networked Systems: 9th International Conference, NETYS 2021, Virtual Event*. Springer, 2021.
- [3] S. V. Belim and S. Y. Belim, "Implementation of Mandatory Access Control in Distributed Systems," *Automatic Control and Computer Sciences*, vol. 52, no. 8. pp. 1124–1126, 2018, doi: 10.3103/S0146411618080357.
- [4] L. Thirupathi and V. N. R. Padmanabhuni, "Multi-level Protection (Mlp) Policy Implementation using Graph Database," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 3, pp. 421–429, 2021, doi: 10.14569/IJACSA.2021.0120350.
- [5] V. Yesin, M. Karpinski, M. Yesina, V. Vilihura, and K. Warwas, "Ensuring data integrity in databases with the universal basis of relations," *Appl. Sci.*, vol. 11, no. 18, 2021, doi: 10.3390/app11188781.
- [6] Y. Duan, X. Deng, and H. Yang, "A Multi-Tenant Access Control Method Based on Environmental Attributes and Security Labels," pp. 41–46, 2021, doi: 10.1145/3473465.3473473.
- [7] S. De Capitani di Vimercati, "Access Matrix," in *Encyclopedia of Cryptography and Security*, Springer US, 2011, pp. 14–17.
- [8] B. Jayant.D, U. Swapnaja A, A. Sulabha S, and M. Dattatray G, "Analysis of DAC MAC RBAC Access Control based Models for Security," *Int. J. Comput. Appl.*, vol. 104, no. 5, pp. 6–13, Oct.

2014, doi: 10.5120/18196-9115.

- [9] H. Aman, R. Miftah, N. Murli, A. Mustapha, and M. M. Zainuddin, "Convoy Marshal: Group-Based Navigation Mobile Application," *Adv. Sci. Lett.*, vol. 24, no. 3, pp. 1660–1665, 2018, doi: 10.1166/asl.2018.11132.