Chapter 4: System design and implementation

4.1 Introduction

In this chapter, the implementation and testing phase of the proposed system will be done and shown. The implementation part of the proposed system will be utilizing and applying the system design that have been done in Chapter 3. The system user interfaces and functionalities with their respective code screenshots will be attached throughout this chapter for better understanding. After the development phase, the testing phase will be carried out to ensure the system to be working as expected with all the errors and bugs are fixed. This chapter consist of a few parts namely environment setup, software installation, system development phase and system testing phase.

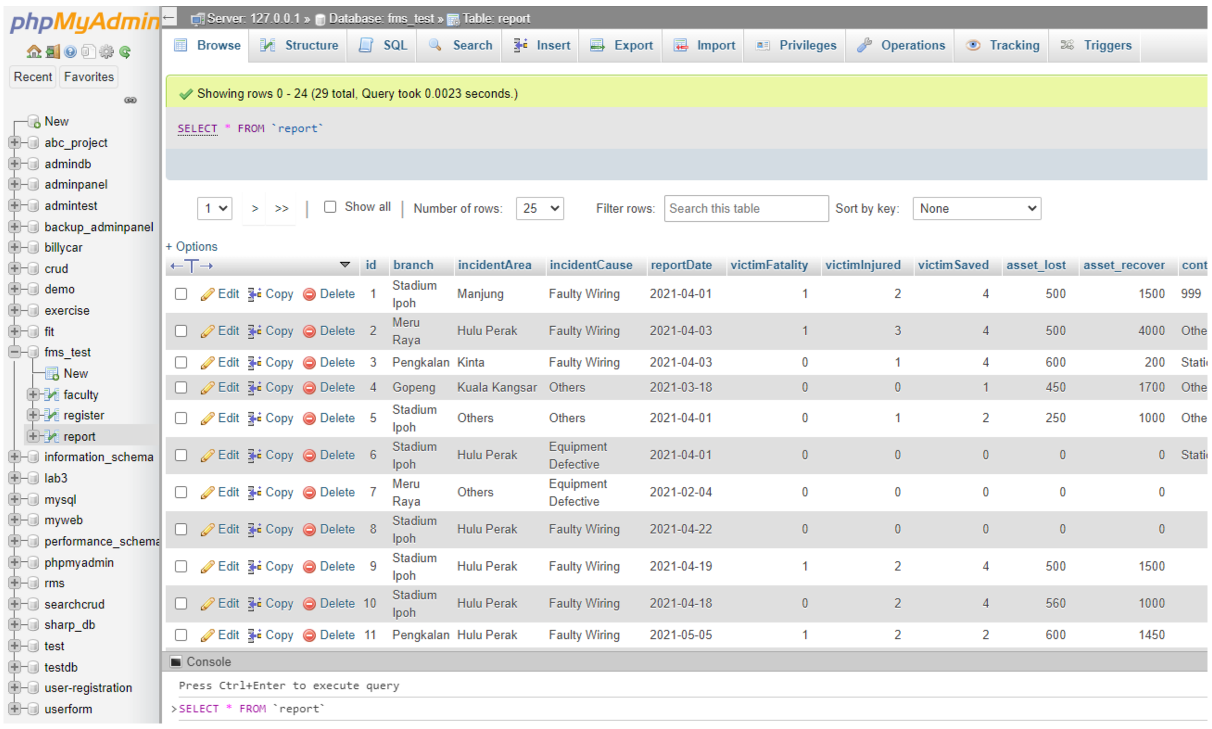
4.2 Environment setup

In this phase, the software required for the system development are installed upon the programming and scripting language chosen for the system development. Then, the database setup will be done through the software as well. No external hardware installation is needed for this proposed system development.

4.2.1 Software installation

The software that are needed for the development will be listed and explained here. For the system development, the backend development will be done using PHP. On the other hand, HTML, CSS, JavaScript, Bootstrap and jQuery will be used for the system's front-end development.

As PHP is used for backend development, XAMPP is installed as phpMyAdmin is used for handling and managing the system database where the tables and columns are created and stored. Then, XAMPP is also necessary for hosting a local server on the computer for the system to run on. Figure 4.1 shows the database tables management interface of phpMyAdmin.



Showing rows 0 - 24 (29 total, Query took 0.0023 seconds.)

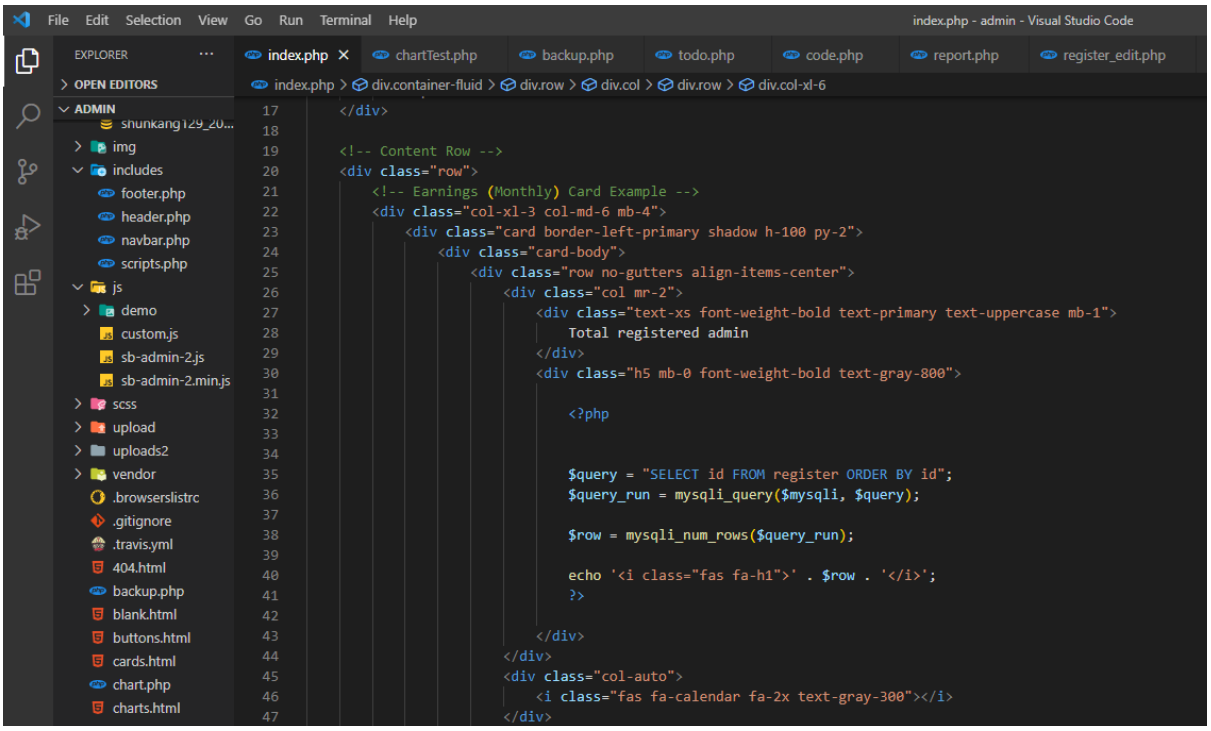
```
SELECT * FROM `report`
```

id	branch	incidentArea	incidentCause	reportDate	victimFatality	victimInjured	victimSaved	asset_lost	asset_recover	cont
1	Stadium Ipoh	Manjung	Faulty Wiring	2021-04-01	1	2	4	500	1500	999
2	Meru Raya	Hulu Perak	Faulty Wiring	2021-04-03	1	3	4	500	4000	Othe
3	Pengkalan	Kinta	Faulty Wiring	2021-04-03	0	1	4	600	200	Stati
4	Gopeng	Kuala Kangsar	Others	2021-03-18	0	0	1	450	1700	Othe
5	Stadium Ipoh	Others	Others	2021-04-01	0	1	2	250	1000	Othe
6	Stadium Ipoh	Hulu Perak	Equipment Defective	2021-04-01	0	0	0	0	0	Stati
7	Meru Raya	Others	Equipment Defective	2021-02-04	0	0	0	0	0	
8	Stadium Ipoh	Hulu Perak	Faulty Wiring	2021-04-22	0	0	0	0	0	
9	Stadium Ipoh	Hulu Perak	Faulty Wiring	2021-04-19	1	2	4	500	1500	
10	Stadium Ipoh	Hulu Perak	Faulty Wiring	2021-04-18	0	2	4	560	1000	
11	Pengkalan	Hulu Perak	Faulty Wiring	2021-05-05	1	2	2	600	1450	

```
Press Ctrl+Enter to execute query
>SELECT * FROM `report`
```

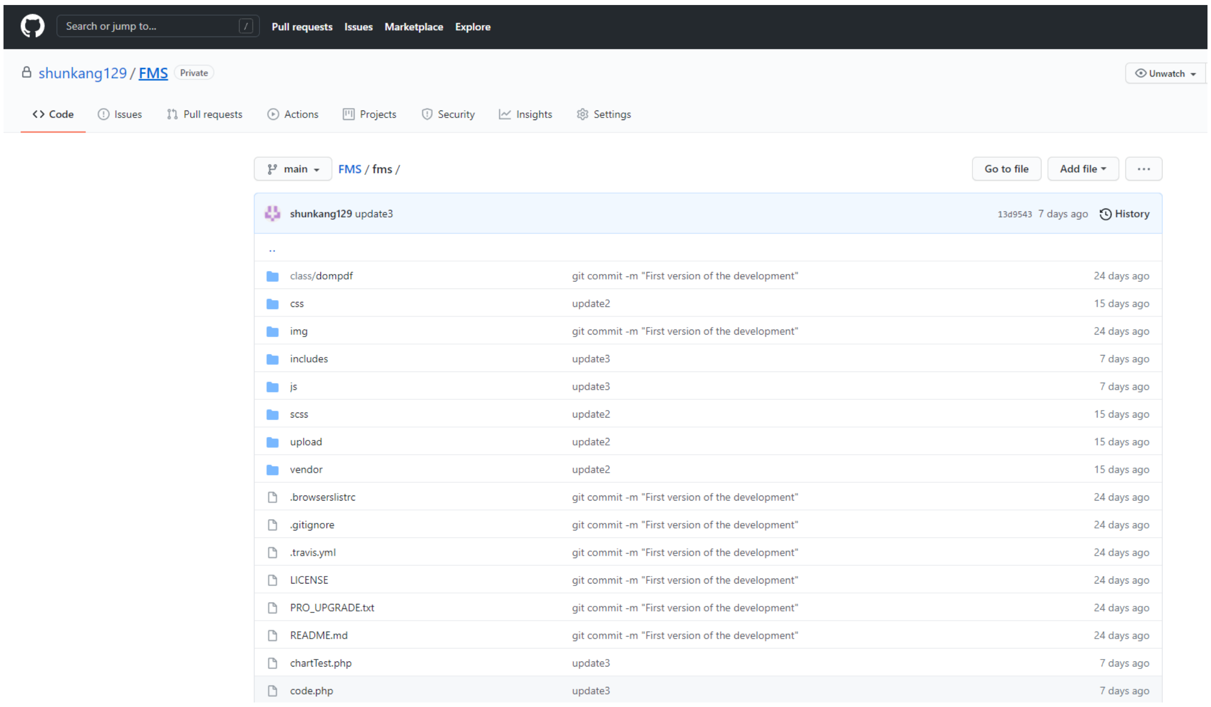
Figure 4.1: phpMyAdmin interface

For the code editor, I have chosen and installed Microsoft VS code to write and edit the code for the system. As most of the other text editors, Microsoft VS code is free and compatible for editing most programming or scripts languages. Other than that, Microsoft VS code is also best known for its customizability where it allows users to install side extensions that assist and increase the user's productivity. (Faulds, 2021). Furthermore, Git is installed for this development as well for better version control as the updates to the system will be pushed to the GitHub remote repository. Figure 4.2 shows the interface of Microsoft VS code while Figure 4.3 shows the GitHub private remote repository of the system.



```
</div>

<!-- Content Row -->
<div class="row">
    <!-- Earnings (Monthly) Card Example -->
    <div class="col-xl-3 col-md-6 mb-4">
        <div class="card border-left-primary shadow h-100 py-2">
            <div class="card-body">
                <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                        <div class="text-xs font-weight-bold text-primary text-uppercase mb-1">
                            Total registered admin
                        </div>
                        <div class="h5 mb-0 font-weight-bold text-gray-800">

                            <?php


                            $query = "SELECT id FROM register ORDER BY id";
                            $query_run = mysqli_query($mysqli, $query);

                            $row = mysqli_num_rows($query_run);

                            echo '<i class="fas fa-h1">' . $row . '</i>';
                            ?>

                        </div>
                    </div>
                    <div class="col-auto">
                        <i class="fas fa-calendar fa-2x text-gray-300"></i>
                    </div>
```

Figure 4.2: Microsoft VS code editor interface


Search or jump to...
Pull requests Issues Marketplace Explore
shunkang129 / FMS Private
Unwatch
Code Issues Pull requests Actions Projects Security Insights Settings
main
FMS / fms /
Go to file
Add file
shunkang129 update3
13d9543 7 days ago
History

..		
class/dompdf	git commit -m "First version of the development"	24 days ago
css	update2	15 days ago
img	git commit -m "First version of the development"	24 days ago
includes	update3	7 days ago
js	update3	7 days ago
scss	update2	15 days ago
upload	update2	15 days ago
vendor	update2	15 days ago
.browserslistrc	git commit -m "First version of the development"	24 days ago
.gitignore	git commit -m "First version of the development"	24 days ago
.travis.yml	git commit -m "First version of the development"	24 days ago
LICENSE	git commit -m "First version of the development"	24 days ago
PRO_UPGRADE.txt	git commit -m "First version of the development"	24 days ago
README.md	git commit -m "First version of the development"	24 days ago
chartTest.php	update3	7 days ago
code.php	update3	7 days ago

Figure 4.3: GitHub remote repository interface

4.2.2 System database setup

The database setup and configuration will be explained here with images attached for better clarification. Firstly, a database named ‘fms’ will be created to handle the data tables of this system. Then, two tables will be created for this system, the tables are named ‘register’ and ‘report’. The ‘register’ will be used to store the user information like name, email, password, user ID, role, and status. Besides, the ‘report’ table will be storing data that will be passed from the report form like incident area, incident date, victim amount, report status and more. Figures below show the database ‘fms’ and the two tables.

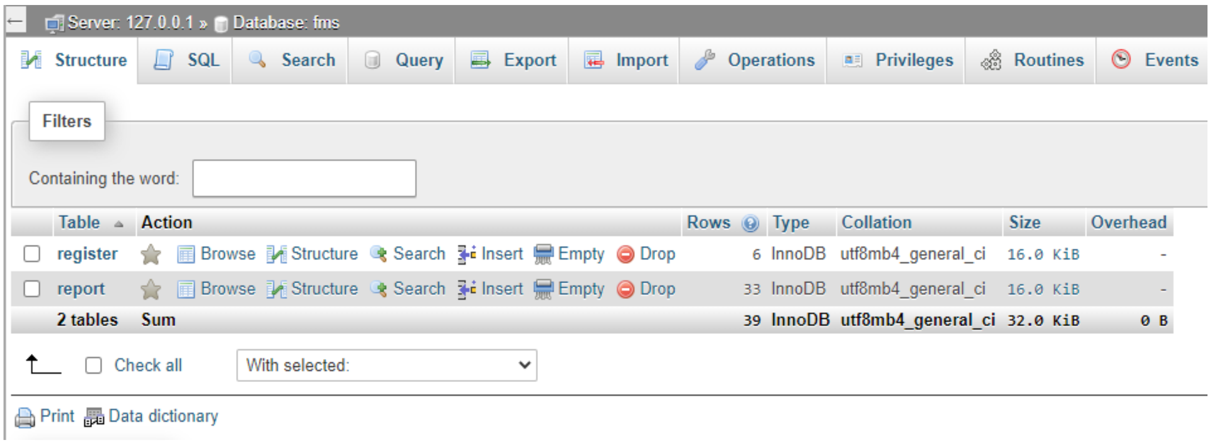
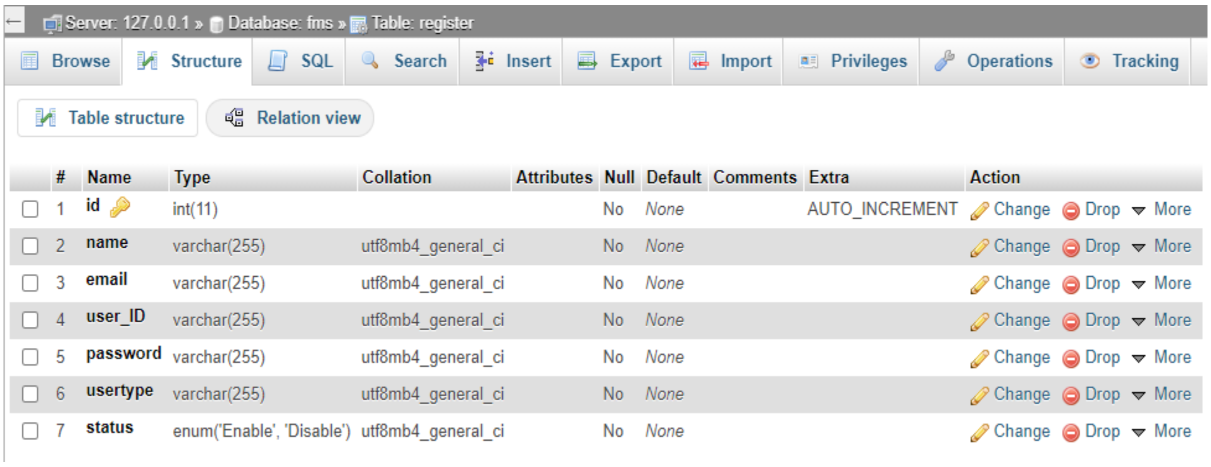


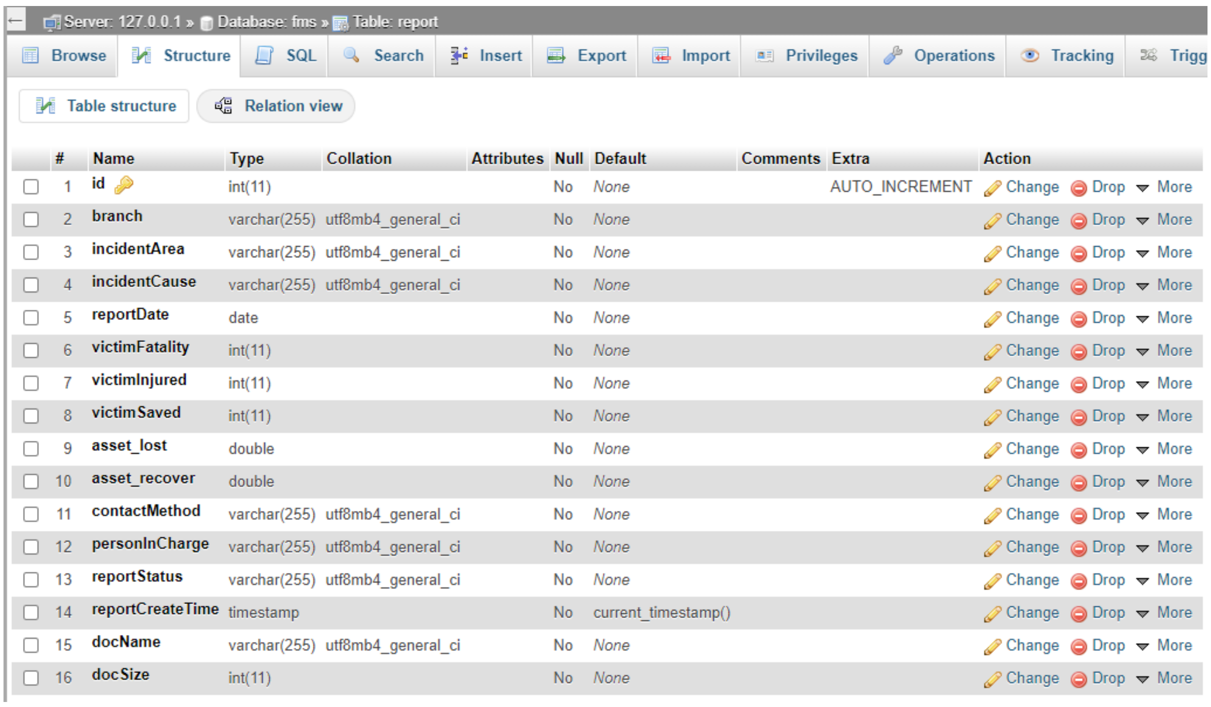
	Table	Action	Rows	Type	Collation	Size	Overhead
☐	register	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	report	Browse Structure Search Insert Empty Drop	33	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	2 tables	Sum	39	InnoDB	utf8mb4_general_ci	32.0 KiB	0 B

Figure 4.4: Database setup



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	email	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	4	user_ID	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	5	password	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	6	usertype	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	7	status	enum('Enable', 'Disable')	utf8mb4_general_ci		No	*None*			Change Drop More

Figure 4.5: 'register' table setup



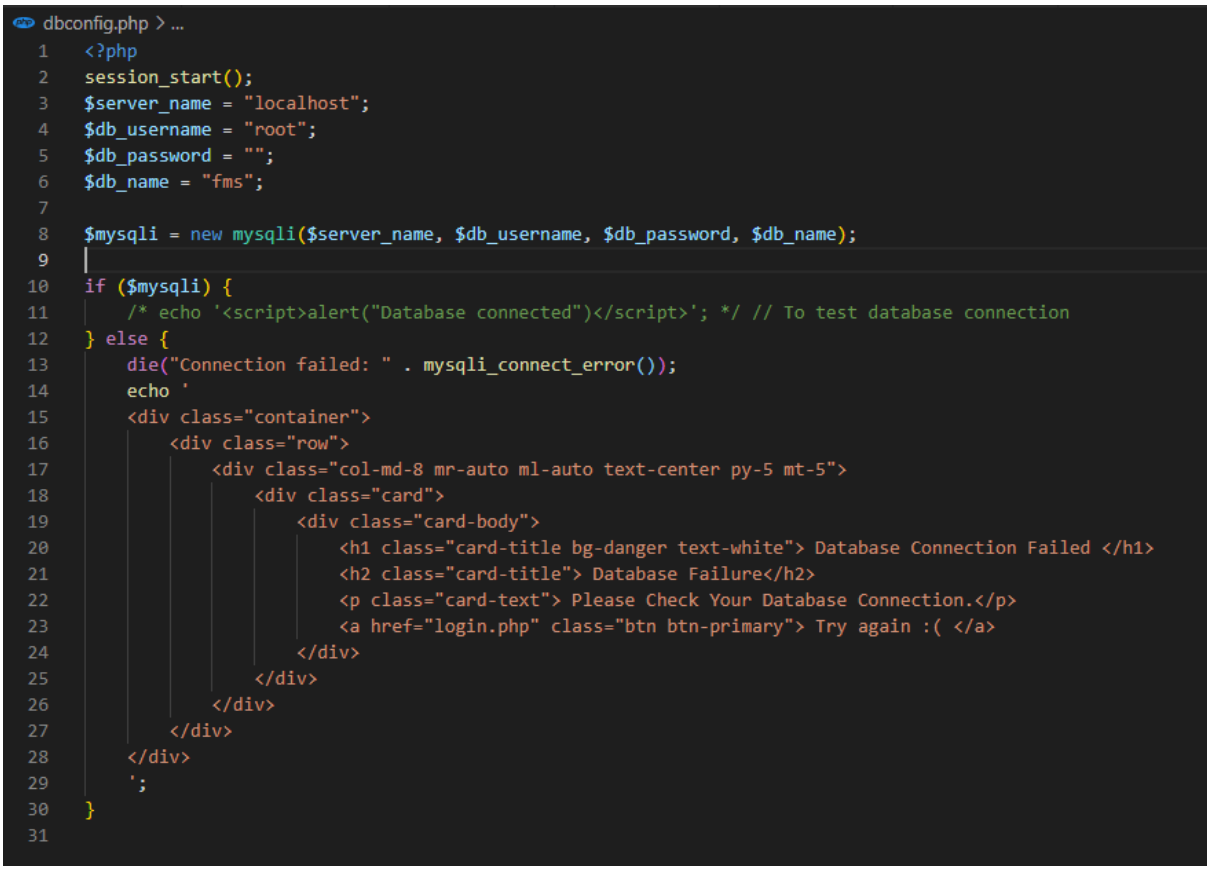
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	branch	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
3	incidentArea	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
4	incidentCause	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
5	reportDate	date			No	*None*			Change Drop More
6	victimFatality	int(11)			No	*None*			Change Drop More
7	victimInjured	int(11)			No	*None*			Change Drop More
8	victimSaved	int(11)			No	*None*			Change Drop More
9	asset_lost	double			No	*None*			Change Drop More
10	asset_recover	double			No	*None*			Change Drop More
11	contactMethod	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
12	personInCharge	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
13	reportStatus	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
14	reportCreateTime	timestamp			No	current_timestamp()			Change Drop More
15	docName	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
16	docSize	int(11)			No	*None*			Change Drop More

Figure 4.6: 'report' table setup

4.3 System implementation and development

4.3.1 Database connection establishment

Before the implementation of the interfaces and features of the proposed system take place, the database connectivity will be established first to ensure that the data insertion and retrieval which will be carried out later by the system can be done without any issue or error. A PHP file named dbconfig will be created to establish the connection to the MySQL database. In this file, several variables are created and assigned with the values required for the database connection. For instance, the 'localhost', 'root', '', and 'fms' are all assigned to the server's name, database user, database password, and database name respectively. Figure 4.7 shows the setup of the database connection written in the dbconfig.php file.



dbconfig.php > ...

```php
<?php
session_start();
$server_name = "localhost";
$db_username = "root";
$db_password = "";
$db_name = "fms";

$mysqli = new mysqli($server_name, $db_username, $db_password, $db_name);

if ($mysqli) {
    /* echo '<script>alert("Database connected")</script>'; */ // To test database connection
} else {
    die("Connection failed: " . mysqli_connect_error());
    echo '
    <div class="container">
        <div class="row">
            <div class="col-md-8 mr-auto ml-auto text-center py-5 mt-5">
                <div class="card">
                    <div class="card-body">
                        <h1 class="card-title bg-danger text-white"> Database Connection Failed </h1>
                        <h2 class="card-title"> Database Failure</h2>
                        <p class="card-text"> Please Check Your Database Connection.</p>
                        <a href="login.php" class="btn btn-primary"> Try again :( </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    ';
}

```

Figure 4.7: Database connection setup

4.3.2 View of the signup page

As users are required to create an account so that they can access the system with the created account. Hence, a sign-up function is written in the code.php file where it stores all the operations code. Then, Figure 4.8 shows the sign-up page interface while Figure 4.9 shows the implementation of the sign-up features where the HTTP POST data will be handled and stored into the database. Users are needed to fill in the information before they proceed with the sign-up. In addition, a small error message underneath the input filed will be displayed to the users if they are entering an existing name, email, or user ID. Other than that, the password entered by users will be hashed using the PHP password_hash() function before they are stored into the database to ensure better security and privacy. After that, the user will be redirected to the login page if an account is successfully created.

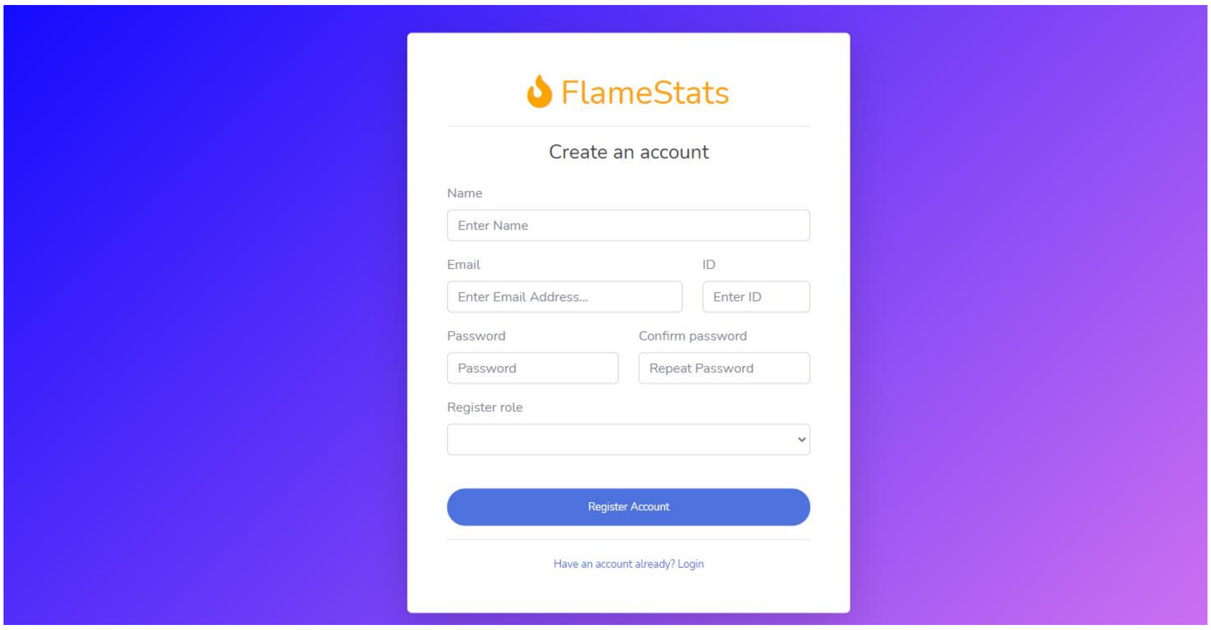
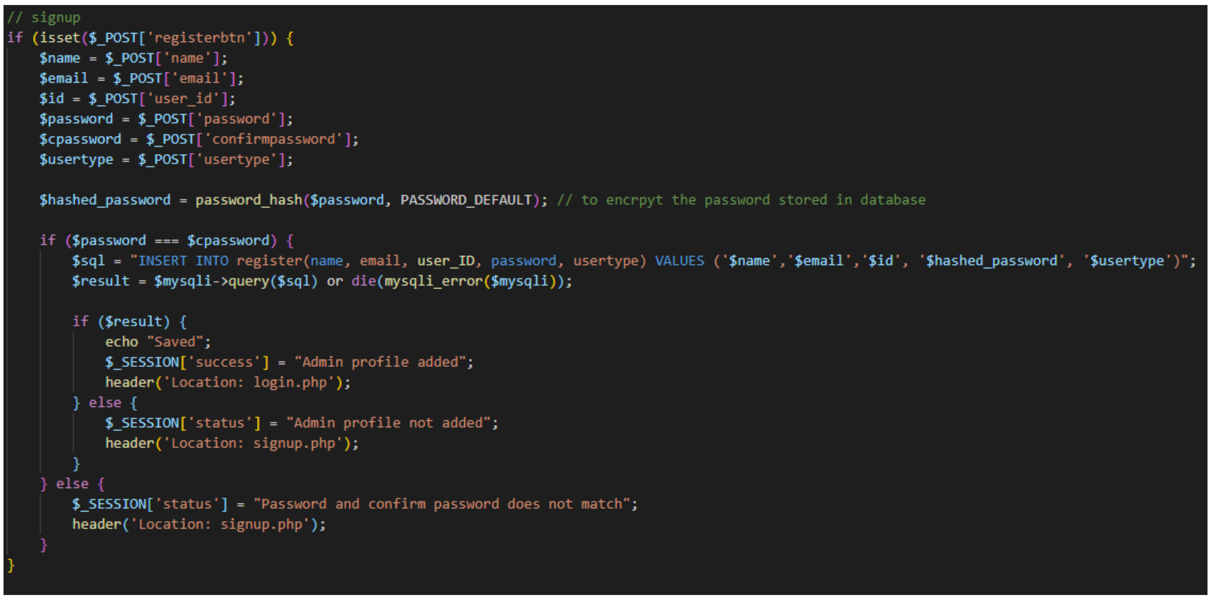


Figure 4.8: Interface of sign-up



```php
// signup
if (isset($_POST['registerbtn'])) {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $id = $_POST['user_id'];
    $password = $_POST['password'];
    $cpassword = $_POST['confirmpassword'];
    $usertype = $_POST['usertype'];

    $hashed_password = password_hash($password, PASSWORD_DEFAULT); // to encrpyt the password stored in database

    if ($password === $cpassword) {
        $sql = "INSERT INTO register(name, email, user_ID, password, usertype) VALUES ('$name','$email','$id', '$hashed_password', '$usertype')";
        $result = $mysqli->query($sql) or die(mysqli_error($mysqli));

        if ($result) {
            echo "Saved";
            $_SESSION['success'] = "Admin profile added";
            header('Location: login.php');
        } else {
            $_SESSION['status'] = "Admin profile not added";
            header('Location: signup.php');
        }
    } else {
        $_SESSION['status'] = "Password and confirm password does not match";
        header('Location: signup.php');
    }
}
```

Figure 4.9: Sign-up implementation

4.3.3 View of the login page

The login interface of the system with its implementation will be discussed here. Figure 4.10 shows the interface of the login page where users need to enter their email and password before accessing the system. Then Figure 4.12 shows the implementation of the login function written in the code.php file to collect the data using HTTP POST method from the login form. After receiving the data input by user from the login form, the system will first determine the user status to check whether the user is disabled, if yes, the user will not be able to log in even though they entered the correct credentials. After that, the system will also determine the role of user to check whether the user is an admin or normal user. Then, the function password_verify() will be used to verify the hashed password stored mentioned earlier to verify the login process. If the credentials are verified, the sessions of login will be initialized then the users will then be redirected to the main page. For any access to the system without the granted sessions, it will be redirected to the error page as shown in Figure 4.11.

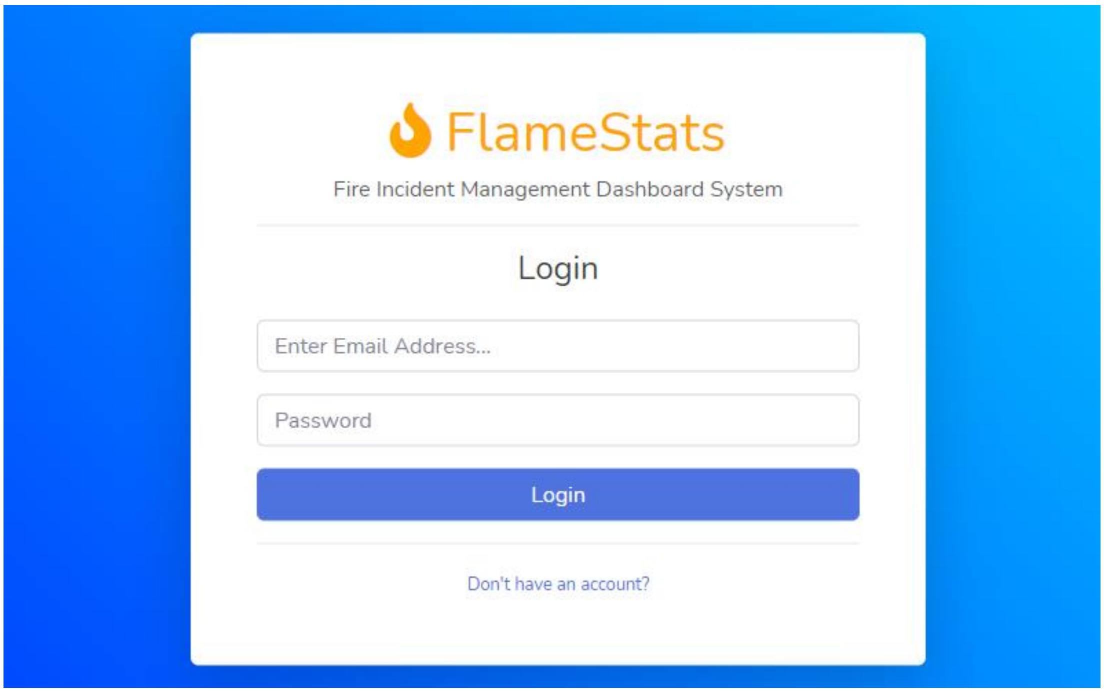
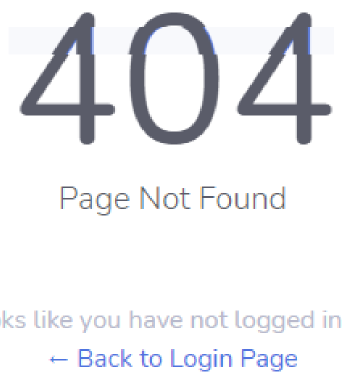


Figure 4.10: Interface of login


404

Page Not Found

It looks like you have not logged in yet...

← Back to Login Page

Figure 4.11: Error page

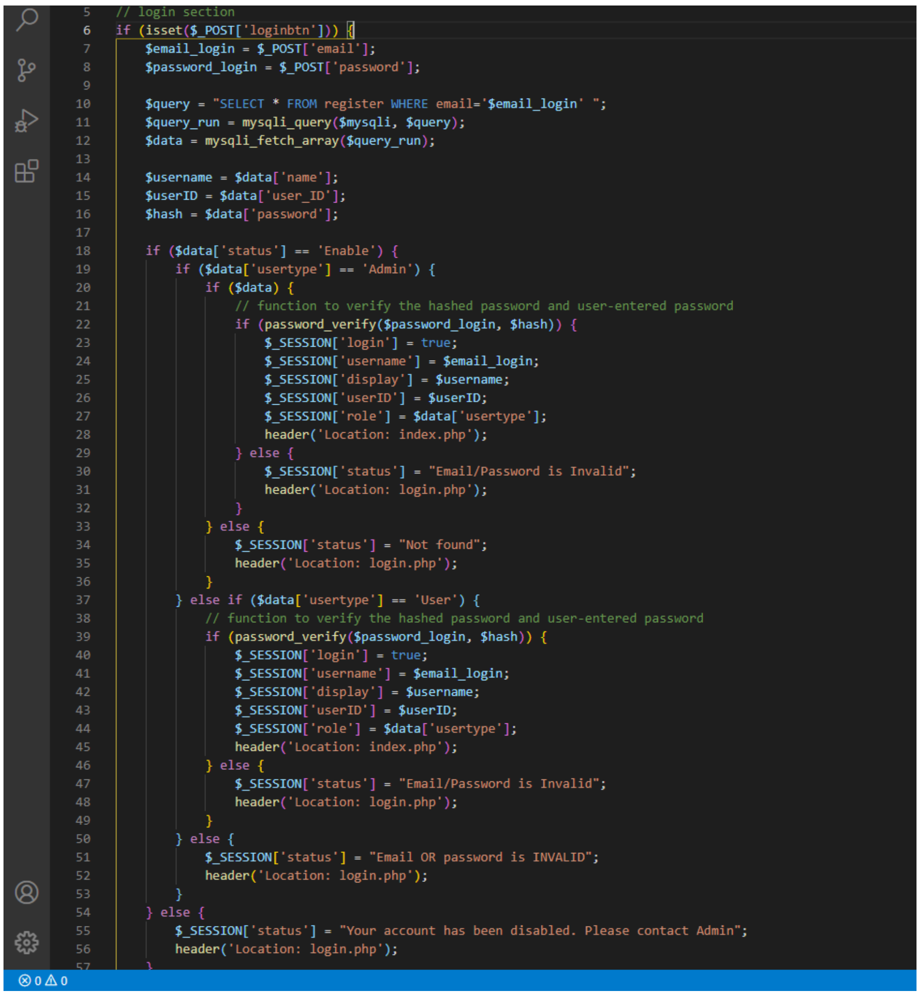

```php
// login section
if (isset($_POST['loginbtn'])) {
    $email_login = $_POST['email'];
    $password_login = $_POST['password'];

    $query = "SELECT * FROM register WHERE email='$email_login' ";
    $query_run = mysqli_query($mysqli, $query);
    $data = mysqli_fetch_array($query_run);

    $username = $data['name'];
    $userID = $data['user_ID'];
    $hash = $data['password'];

    if ($data['status'] == 'Enable') {
        if ($data['usertype'] == 'Admin') {
            if ($data) {
                // function to verify the hashed password and user-entered password
                if (password_verify($password_login, $hash)) {
                    $_SESSION['login'] = true;
                    $_SESSION['username'] = $email_login;
                    $_SESSION['display'] = $username;
                    $_SESSION['userID'] = $userID;
                    $_SESSION['role'] = $data['usertype'];
                    header('Location: index.php');
                } else {
                    $_SESSION['status'] = "Email/Password is Invalid";
                    header('Location: login.php');
                }
            } else {
                $_SESSION['status'] = "Not found";
                header('Location: login.php');
            }
        } else if ($data['usertype'] == 'User') {
            // function to verify the hashed password and user-entered password
            if (password_verify($password_login, $hash)) {
                $_SESSION['login'] = true;
                $_SESSION['username'] = $email_login;
                $_SESSION['display'] = $username;
                $_SESSION['userID'] = $userID;
                $_SESSION['role'] = $data['usertype'];
                header('Location: index.php');
            } else {
                $_SESSION['status'] = "Email/Password is Invalid";
                header('Location: login.php');
            }
        } else {
            $_SESSION['status'] = "Email OR password is INVALID";
            header('Location: login.php');
        }
    } else {
        $_SESSION['status'] = "Your account has been disabled. Please contact Admin";
        header('Location: login.php');
    }
```

Figure 4.12: Login implementation

4.3.4 View of the main dashboard page

The main page of the system will be displaying the overview data and charts to the users. As a dashboard-based management system, there will be a side navigation panel which allow users to access other features and pages while the top navigation bar will have the link to the users' profile page. If the user logged in as an admin, they would be able to access more features compared to normal users. Features like user managements and backup database will be displayed at the side panel only to the admin users. Figure 4.13 and Figure 4.14 will be showing the interface of the main page with admin users or normal users logging in, respectively. The side navigation panel design for normal user's system will be changed to be in red in order to provide better differentiation for users.

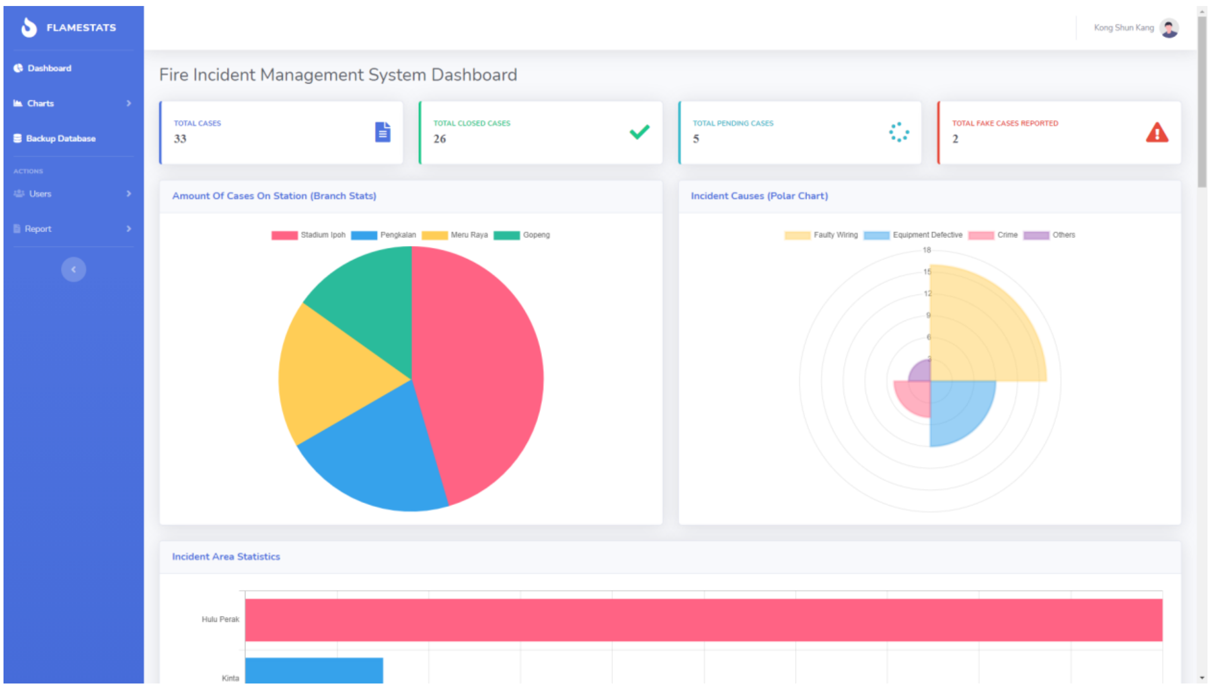


Figure 4.13 Main dashboard interface (admin view)

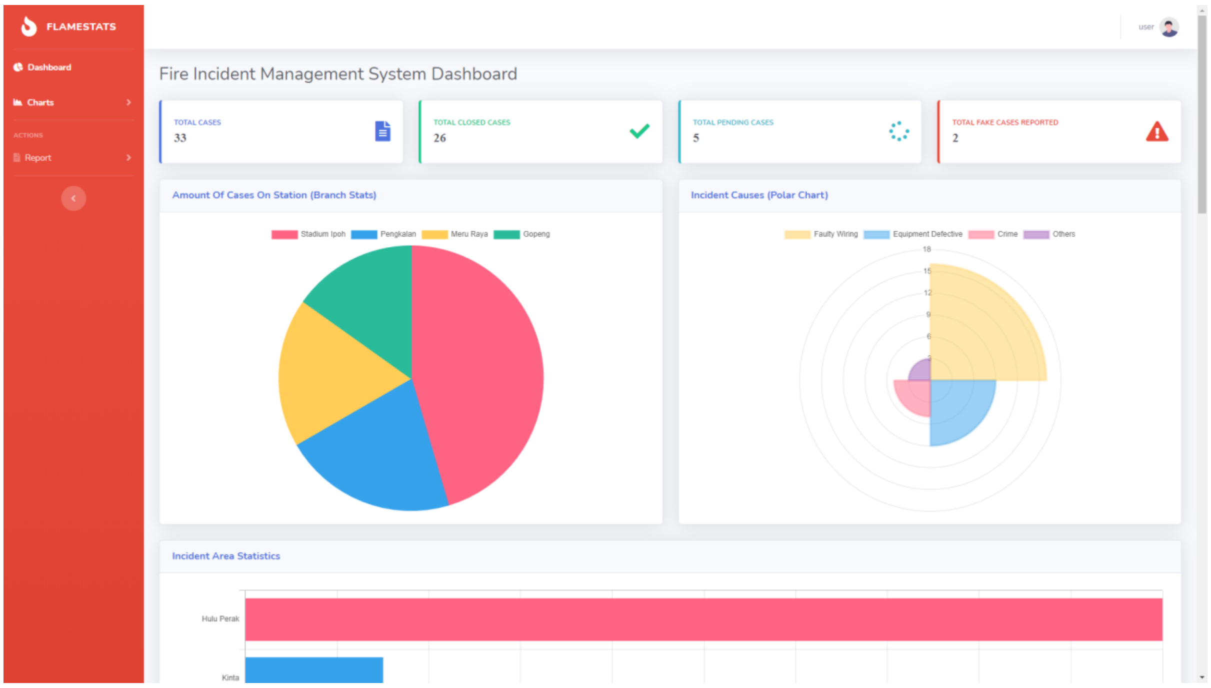


Figure 4.14: Main dashboard interface (Normal user view)

The frontend of this system is implemented using the Bootstrap v4.6 and Bootstrap SB admin 2 plugins which provide more responsive and automatic features. Furthermore, the charts and graphs are all generated using the Chart.js 2.9.4 version plugins. The reason that I chose Chart.js 2.9.4 version instead of the latest released version is because this version of the plugins works better with the data setup and other plugins that I have used for this system such as DataTables, moment.js etc.

In order to generate the charts, it is required to first setup the data that will be passed to the chart generate function that comes with the Chart.js plugins. For this system development, the chart data setup and preparation will be done inside the chartSetup.php file as shown in Figure 4.15. For instance, the function mysqli_num_rows() will be used here to collect the number of rows of that particular data which will be assigned to a new variable. Then, the variable will be used to display out the data in charts later on.

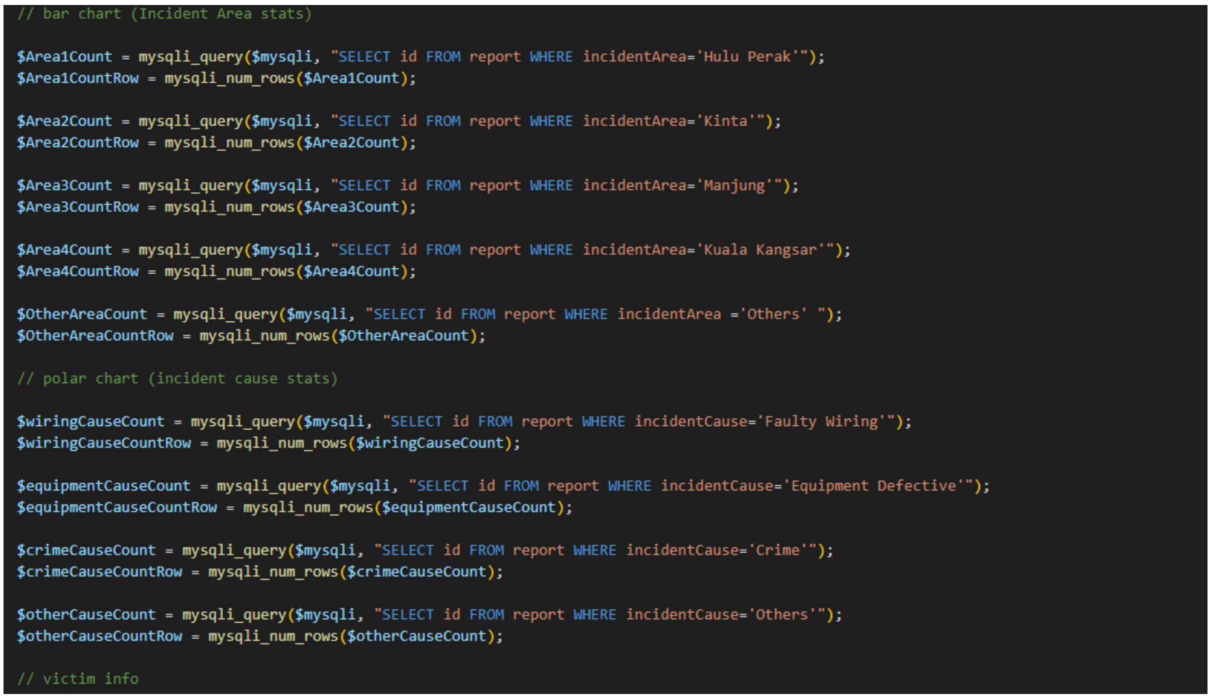
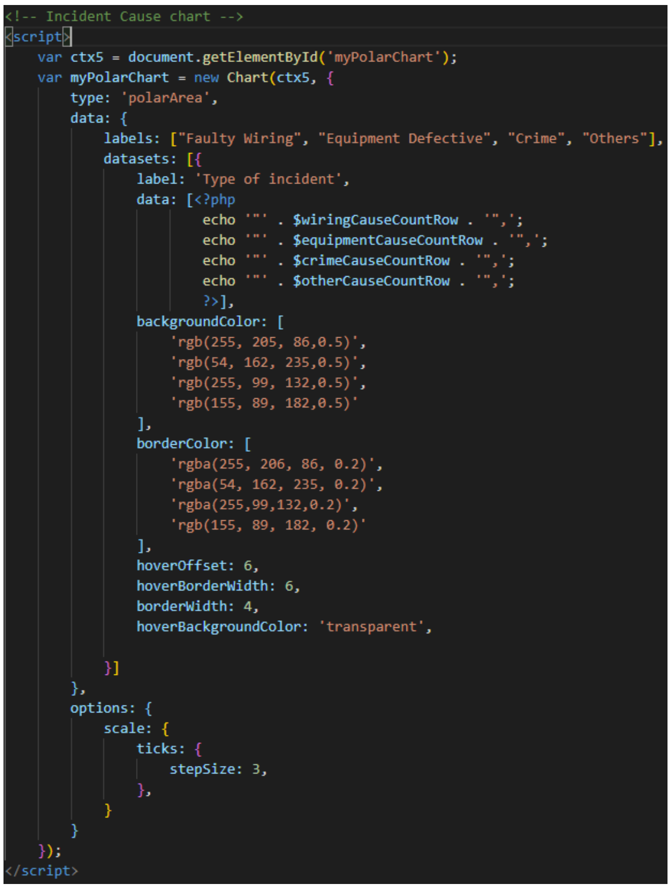

```php
// bar chart (Incident Area stats)

$Area1Count = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentArea='Hulu Perak'");
$Area1CountRow = mysqli_num_rows($Area1Count);

$Area2Count = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentArea='Kinta'");
$Area2CountRow = mysqli_num_rows($Area2Count);

$Area3Count = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentArea='Manjung'");
$Area3CountRow = mysqli_num_rows($Area3Count);

$Area4Count = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentArea='Kuala Kangsar'");
$Area4CountRow = mysqli_num_rows($Area4Count);

$OtherAreaCount = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentArea ='Others' ");
$OtherAreaCountRow = mysqli_num_rows($OtherAreaCount);

// polar chart (incident cause stats)

$wiringCauseCount = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentCause='Faulty Wiring'");
$wiringCauseCountRow = mysqli_num_rows($wiringCauseCount);

$equipmentCauseCount = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentCause='Equipment Defective'");
$equipmentCauseCountRow = mysqli_num_rows($equipmentCauseCount);

$crimeCauseCount = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentCause='Crime'");
$crimeCauseCountRow = mysqli_num_rows($crimeCauseCount);

$otherCauseCount = mysqli_query($mysqli, "SELECT id FROM report WHERE incidentCause='Others'");
$otherCauseCountRow = mysqli_num_rows($otherCauseCount);

// victim info
```

Figure 4.15: chartSetup implementation

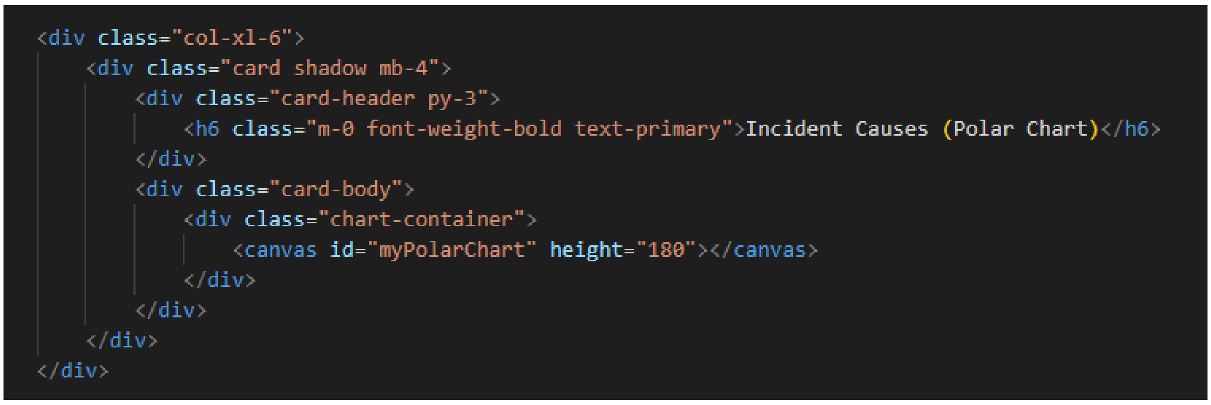
After the setup, the chartSetup.php will be included at the beginning of all the php files that will display charts. Other than the setup, the data will also be passed to the chart scripts to generate and display the charts as mentioned previously. These scripts will all be stored inside the scripts.php file together with all the other scripts declaration. Figure 4.16 shows one of the chart scripts where it displays the data of incident causes in a polar area chart. The script is started with assigning and displaying the chart to the canvas tag with that specific id tag, in this example, the tag ‘myPolarChart’ is used. Then, the type of chart to be generated and displayed will be set here as well, the type ‘polarArea’ is used here to display a polar area chart that showcase the number of cases with each type of incident causes. After setting up the chart type, the data will be set here by using the PHP echo function to produce the number of cases of each incident causes. Other than that, the chart can also be further customized in terms of the scale sizes, colour and more.



```
<!-- Incident Cause chart -->
<script>
    var ctx5 = document.getElementById('myPolarChart');
    var myPolarChart = new Chart(ctx5, {
        type: 'polarArea',
        data: {
            labels: ["Faulty Wiring", "Equipment Defective", "Crime", "Others"],
            datasets: [{
                label: 'Type of incident',
                data: [<?php
                        echo '"' . $wiringCauseCountRow . '",';
                        echo '"' . $equipmentCauseCountRow . '",';
                        echo '"' . $crimeCauseCountRow . '",';
                        echo '"' . $otherCauseCountRow . '",';
                        ?>],
                backgroundColor: [
                    'rgb(255, 205, 86,0.5)',
                    'rgb(54, 162, 235,0.5)',
                    'rgb(255, 99, 132,0.5)',
                    'rgb(155, 89, 182,0.5)'
                ],
                borderColor: [
                    'rgba(255, 206, 86, 0.2)',
                    'rgba(54, 162, 235, 0.2)',
                    'rgba(255,99,132,0.2)',
                    'rgb(155, 89, 182, 0.2)'
                ],
                hoverOffset: 6,
                hoverBorderWidth: 6,
                borderWidth: 4,
                hoverBackgroundColor: 'transparent',

            }]
        },
        options: {
            scale: {
                ticks: {
                    stepSize: 3,
                },
            }
        }
    });
</script>
```

Figure 4.16: Polar Area chart scripts

For the frontend part of the Chartjs implementation, the charts will be called using the canvas tag with the specific tag id as shown in Figure 4.17.



```
<div class="col-xl-6">
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-primary">Incident Causes (Polar Chart)</h6>
        </div>
        <div class="card-body">
            <div class="chart-container">
                <canvas id="myPolarChart" height="180"></canvas>
            </div>
        </div>
    </div>
</div>
```

Figure 4.17: Chartjs canvas tag

The charts can be filtered upon users’ preferences by clicking on the legends. For instance, Figure 4.18 shows the mix chart that involve in using two types of chart namely bar chart and line chart. A click on the legend of that particular dataset will remove that dataset from the chart until you click the legend again as shown in Figure 4.19.

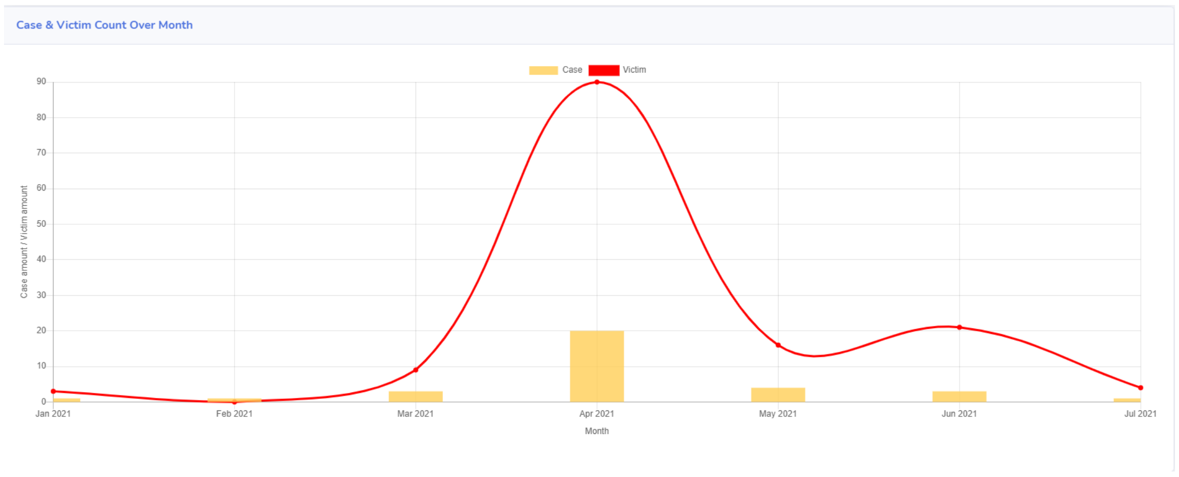


Figure 4.18: Mix chart before filter

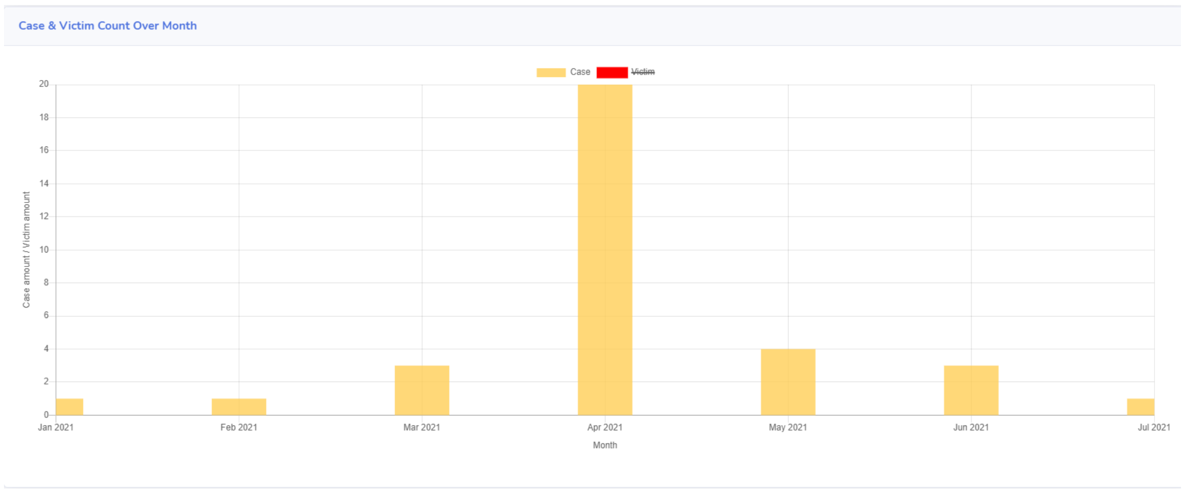
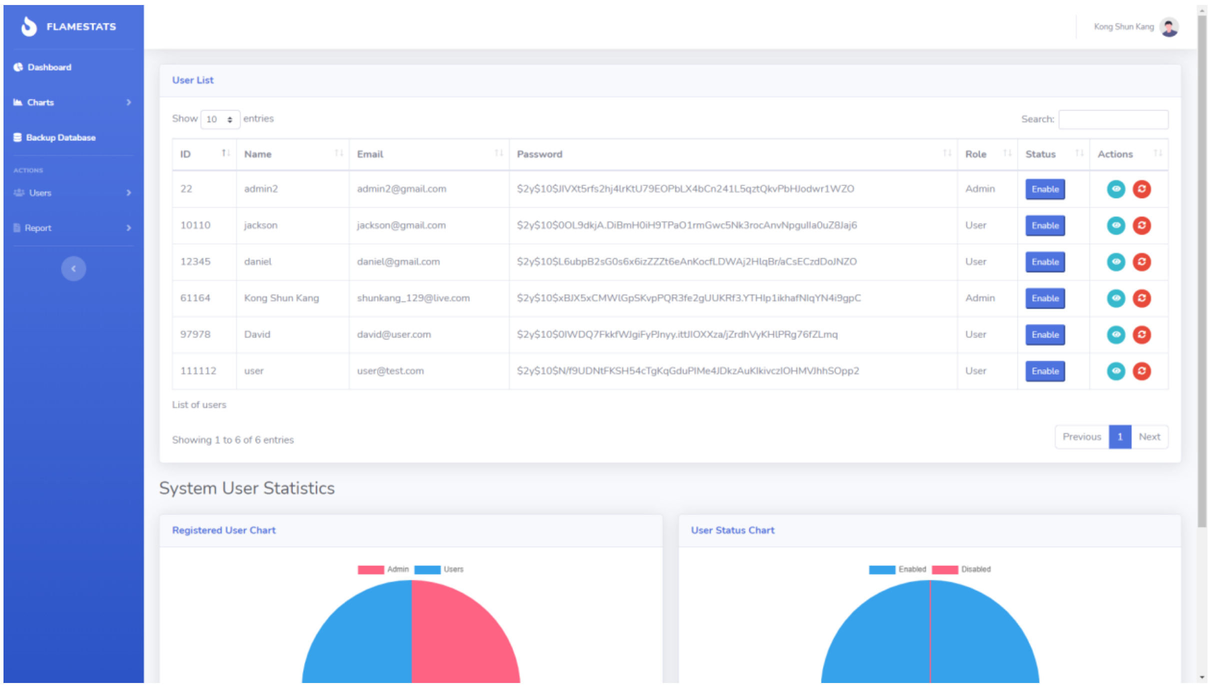


Figure 4.19: Mix chart after filtering out one data

4.3.4 View of the user management function

The user management feature that implemented for the admin users will be discussed here. The user management page will showcase a table that displays current system users' data. Then, the admin users can choose to either view the profile or swap the other user's account status by clicking on the respective button prepared at the end of the row. The interface will display two charts that show the current number of admins and normal users of the system and another chart will be showing the data of users' status. Figure 4.20 shows the interface of the user management interface.



User List

Show 10 entries Search:

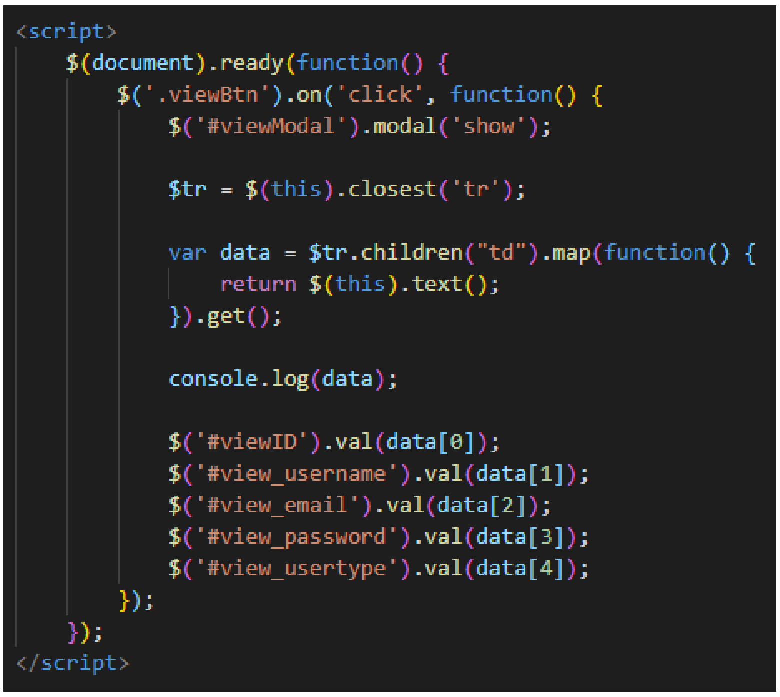
ID	Name	Email	Password	Role	Status	Actions
22	admin2	admin2@gmail.com	$2y$10$JIVXt5rfs2hj4lrKtU79EOPbLX4bCn241L5qztQkvPbHJodwr1WZO	Admin	Enable	
10110	jackson	jackson@gmail.com	$2y$10$0OL9dkjA.DiBmH0iH9TPaO1rmGwc5Nk3rocAnvNpgulla0uZ8Jaj6	User	Enable	
12345	daniel	daniel@gmail.com	$2y$10$L6ubpB2sG0s6x6izZZZt6eAnKocfLDWAj2HlqBr/aCsECzdDoJNZO	User	Enable	
61164	Kong Shun Kang	shunkang_129@live.com	$2y$10$xBJX5xCMWlGpSKvpPQR3fe2gUUKRf3.YTHIp1ikhafNlqYN4i9gpC	Admin	Enable	
97978	David	david@user.com	$2y$10$0IWDQ7FkkfWJgiFyPJnyy.ittJIOXXza/jZrdhVyKHlPRg76fZLmq	User	Enable	
111112	user	user@test.com	$2y$10$N/f9UDNtFKSH54cTgKqGduPIMe4JDkzAuKlkivczIOHMVJhhSOpp2	User	Enable	

List of users

Showing 1 to 6 of 6 entries Previous 1 Next

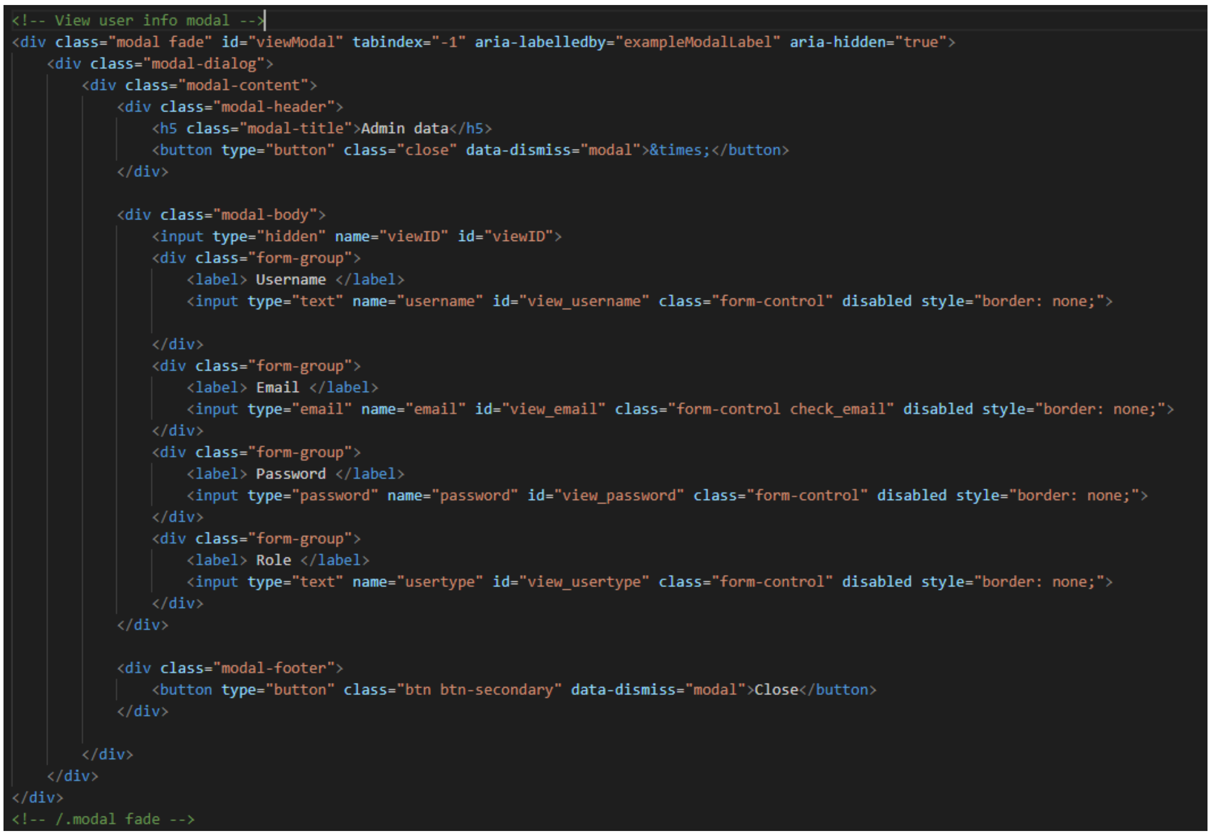
Figure 4.20: Interface of user management

When click on the view button, a popup modal filled with the selected user's information will be displayed in the interface. This feature is done using AJAX by passing the data from the data table to the Bootstrap modal. Figure 4.21 shows the AJAX implementation while Figure 4.22 shows the Bootstrap popup modal front-end implementation where the id tag are specified in order to display data in the modal when called.



```
<script>
    $(document).ready(function() {
        $('.viewBtn').on('click', function() {
            $('#viewModal').modal('show');

            $tr = $(this).closest('tr');

            var data = $tr.children("td").map(function() {
                return $(this).text();
            }).get();

            console.log(data);

            $('#viewID').val(data[0]);
            $('#view_username').val(data[1]);
            $('#view_email').val(data[2]);
            $('#view_password').val(data[3]);
            $('#view_usertype').val(data[4]);
        });
    });
</script>
```

Figure 4.21 View user information modal AJAX implementation

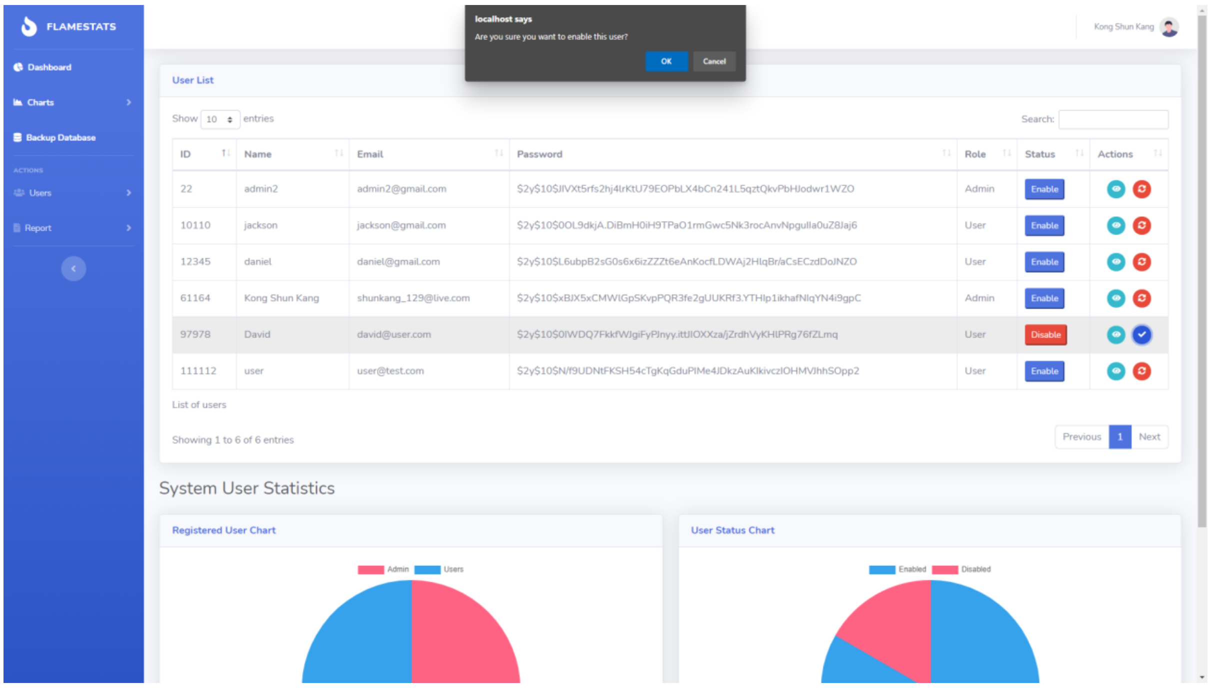


```
<!-- View user info modal -->
<div class="modal fade" id="viewModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Admin data</h5>
                <button type="button" class="close" data-dismiss="modal">&times;</button>
            </div>

            <div class="modal-body">
                <input type="hidden" name="viewID" id="viewID">
                <div class="form-group">
                    <label> Username </label>
                    <input type="text" name="username" id="view_username" class="form-control" disabled style="border: none;">

                </div>
                <div class="form-group">
                    <label> Email </label>
                    <input type="email" name="email" id="view_email" class="form-control check_email" disabled style="border: none;">
                </div>
                <div class="form-group">
                    <label> Password </label>
                    <input type="password" name="password" id="view_password" class="form-control" disabled style="border: none;">
                </div>
                <div class="form-group">
                    <label> Role </label>
                    <input type="text" name="usertype" id="view_usertype" class="form-control" disabled style="border: none;">
                </div>
            </div>

            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
            </div>

        </div>
    </div>
</div>
<!-- /.modal fade -->
```

Figure 4.22: User info modal implementation

If the users click on the swap status button beside the view button, a confirmation message will be prompted by the system to confirm the users' action before the status is being switched.

Figure 4.23 shows the pop out message and also the result after the status is switched. Figure 4.24 shows the implementation of the status swapping feature written in the code.php file.



localhost says

Are you sure you want to enable this user?

OK Cancel

User List

Show 10 entries Search:

ID	Name	Email	Password	Role	Status	Actions
22	admin2	admin2@gmail.com	$2y$10$JIVXt5rfs2hj4lrKtU79EOPbLX4bCn241L5qztQkvPbHJodwr1WZO	Admin	Enable	
10110	jackson	jackson@gmail.com	$2y$10$0OL9dkjA.DiBmH0iH9TPaO1rmGwc5Nk3rocAnvNpgulla0uZ8Jaj6	User	Enable	
12345	daniel	daniel@gmail.com	$2y$10$L6ubpB2sG0s6x6izZZZt6eAnKocfLDWAj2HlqBr/aCsECzdDoJNZO	User	Enable	
61164	Kong Shun Kang	shunkang_129@live.com	$2y$10$xBJX5xCMWlGpSKvpPQR3fe2gUUKRf3.YTHlp1ikhafNlqYN4i9gpC	Admin	Enable	
97978	David	david@user.com	$2y$10$0IWDQ7FkkfWJgiFyPJnyy.itUIOXXza/jZrdhVyKHlPRg76fZLmq	User	Disable	
111112	user	user@test.com	$2y$10$N/f9UDNtFKSH54cTgKqGduPIMe4JDkzAuKlkivczIOHMVJhhSOpp2	User	Enable	

List of users

Showing 1 to 6 of 6 entries

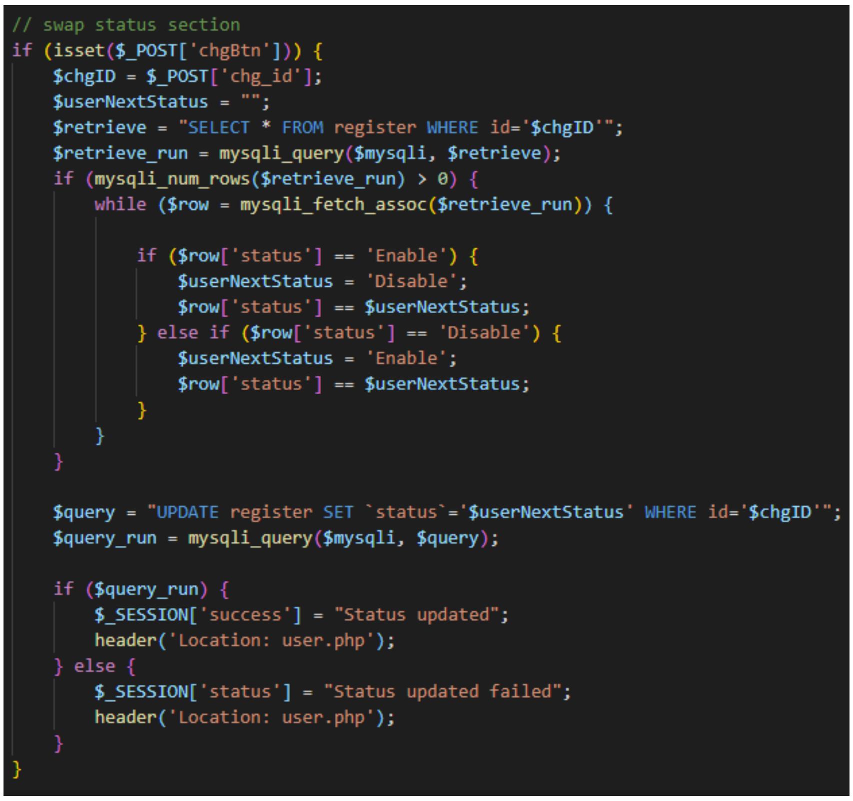
Previous 1 Next

System User Statistics

Registered User Chart

User Status Chart

Figure 4.23: Swap status pop up confirmation

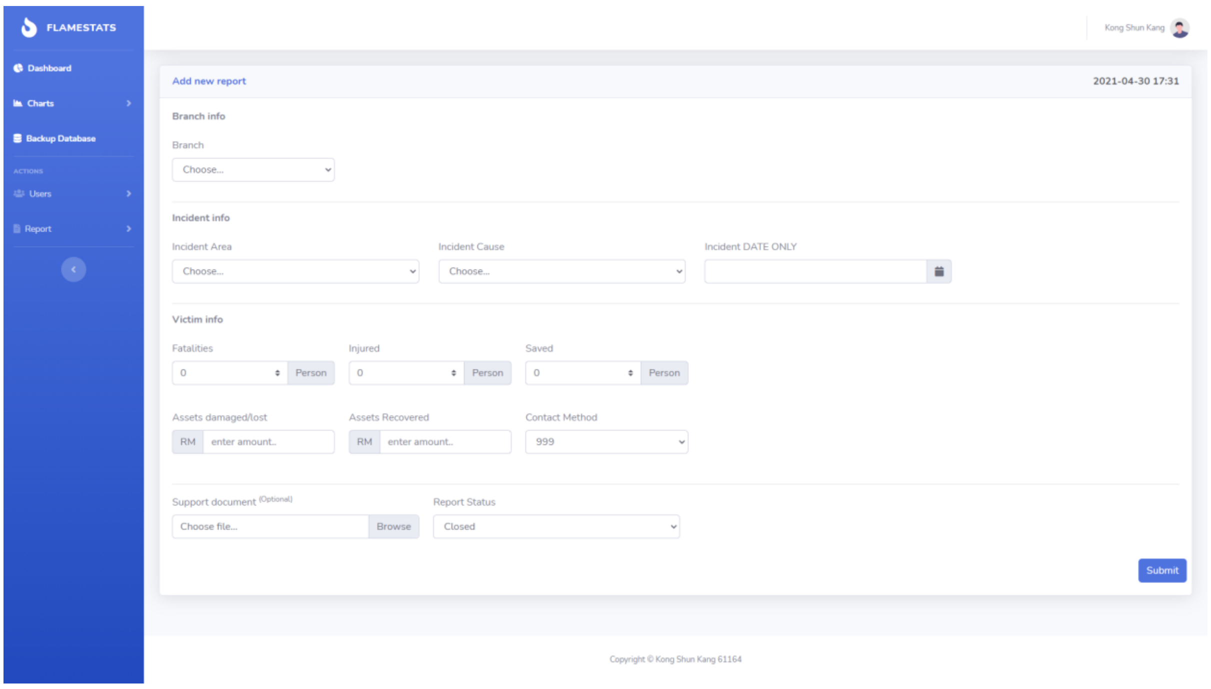


```php
// swap status section
if (isset($_POST['chgBtn'])) {
    $chgID = $_POST['chg_id'];
    $userNextStatus = "";
    $retrieve = "SELECT * FROM register WHERE id='$chgID'";
    $retrieve_run = mysqli_query($mysqli, $retrieve);
    if (mysqli_num_rows($retrieve_run) > 0) {
        while ($row = mysqli_fetch_assoc($retrieve_run)) {

            if ($row['status'] == 'Enable') {
                $userNextStatus = 'Disable';
                $row['status'] == $userNextStatus;
            } else if ($row['status'] == 'Disable') {
                $userNextStatus = 'Enable';
                $row['status'] == $userNextStatus;
            }
        }
    }

    $query = "UPDATE register SET `status`='$userNextStatus' WHERE id='$chgID'";
    $query_run = mysqli_query($mysqli, $query);

    if ($query_run) {
        $_SESSION['success'] = "Status updated";
        header('Location: user.php');
    } else {
        $_SESSION['status'] = "Status updated failed";
        header('Location: user.php');
    }
}
```

Figure 4.24: Swap status function implementation

4.3.5 View of the add report function

Users are required to fill in the report before submitting and storing it into the database. In the report filling part, the upload support document is optional, and the file uploaded should not be more than 10MB. The add report features is also written inside code.php to receive the HTTP POST request data from the form in the report.php file. Figure 4.25 shows the add report interface while Figure 4.26 and Figure 4.27 show the implementation of the add report feature as it will determine whether the users are uploading any document before insert the data into the database. After the users successfully or failure to add a report, a session will be initialized and act as a message to display and notify users at the report.php page to inform the users whether the report has been added or vice versa.



Add new report 2021-04-30 17:31

Branch info

Branch: Choose...

Incident info

Incident Area: Choose... | Incident Cause: Choose... | Incident DATE ONLY:

Victim info

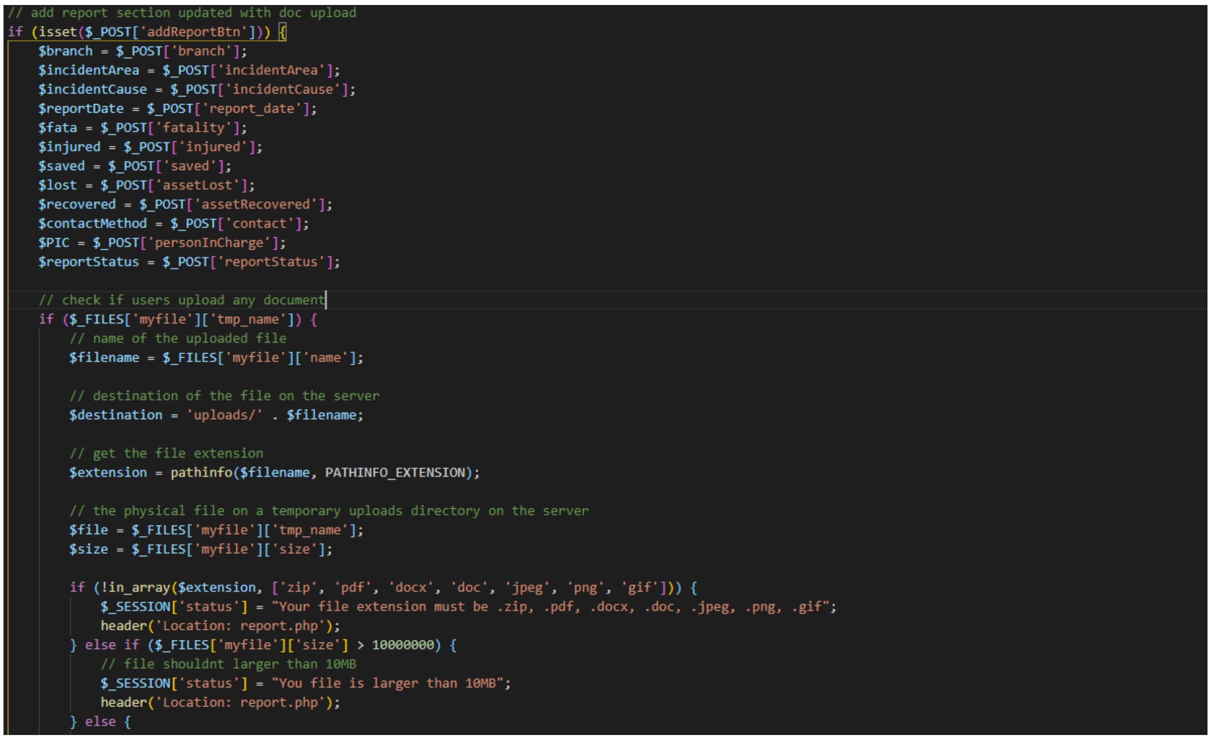
Fatalities: 0 Person | Injured: 0 Person | Saved: 0 Person

Assets damaged/lost: RM enter amount.. | Assets Recovered: RM enter amount.. | Contact Method: 999

Support document (Optional): Choose file... Browse | Report Status: Closed

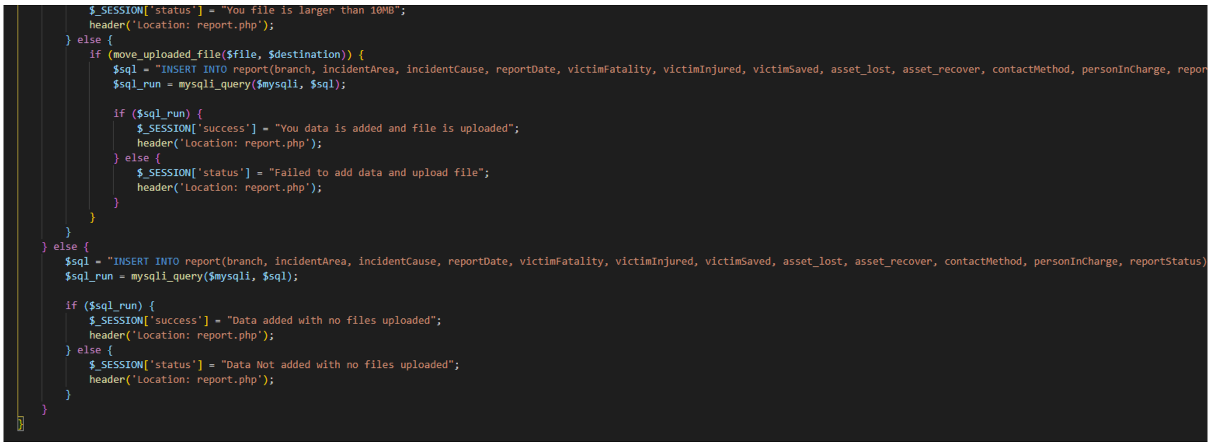
Copyright © Kong Shun Kang 61164

Figure 4.25: Interface of add report



```php
// add report section updated with doc upload
if (isset($_POST['addReportBtn'])) {
    $branch = $_POST['branch'];
    $incidentArea = $_POST['incidentArea'];
    $incidentCause = $_POST['incidentCause'];
    $reportDate = $_POST['report_date'];
    $fata = $_POST['fatality'];
    $injured = $_POST['injured'];
    $saved = $_POST['saved'];
    $lost = $_POST['assetLost'];
    $recovered = $_POST['assetRecovered'];
    $contactMethod = $_POST['contact'];
    $PIC = $_POST['personInCharge'];
    $reportStatus = $_POST['reportStatus'];

    // check if users upload any document
    if ($_FILES['myfile']['tmp_name']) {
        // name of the uploaded file
        $filename = $_FILES['myfile']['name'];

        // destination of the file on the server
        $destination = 'uploads/' . $filename;

        // get the file extension
        $extension = pathinfo($filename, PATHINFO_EXTENSION);

        // the physical file on a temporary uploads directory on the server
        $file = $_FILES['myfile']['tmp_name'];
        $size = $_FILES['myfile']['size'];

        if (!in_array($extension, ['zip', 'pdf', 'docx', 'doc', 'jpeg', 'png', 'gif'])) {
            $_SESSION['status'] = "Your file extension must be .zip, .pdf, .docx, .doc, .jpeg, .png, .gif";
            header('Location: report.php');
        } else if ($_FILES['myfile']['size'] > 10000000) {
            // file shouldnt larger than 10MB
            $_SESSION['status'] = "You file is larger than 10MB";
            header('Location: report.php');
        } else {
```

Figure 4.26: Add report feature and file upload checking part 1

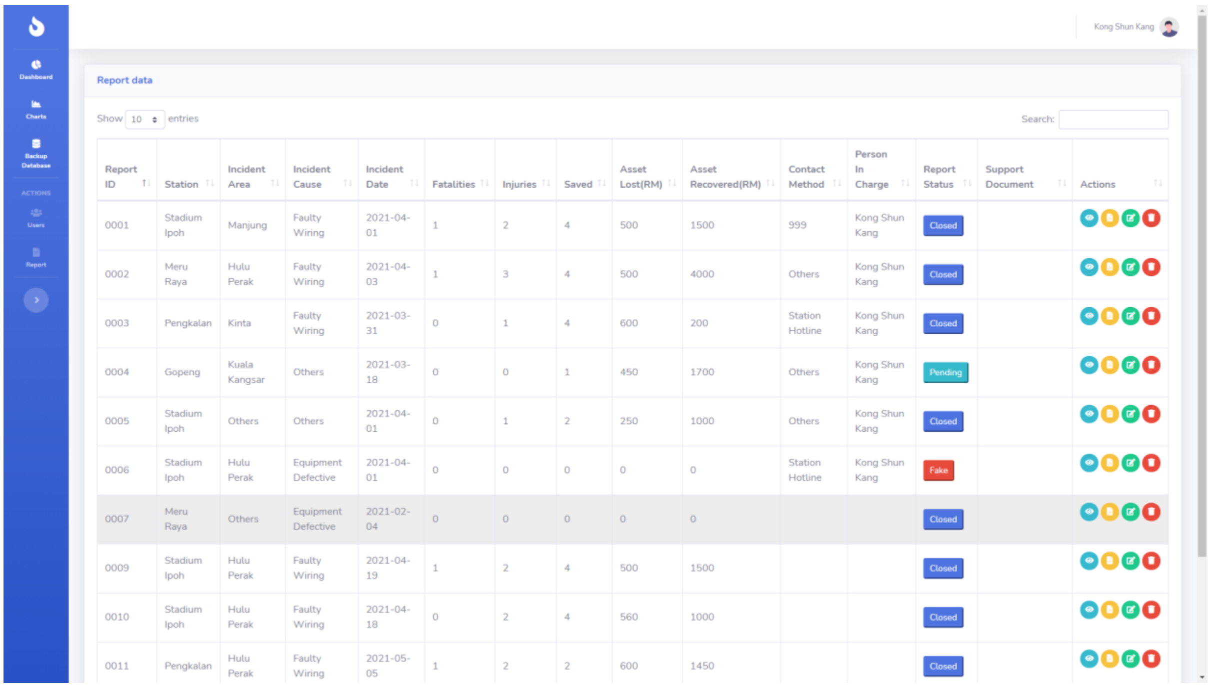


```php
            $_SESSION['status'] = "You file is larger than 10MB";
            header('Location: report.php');
        } else {
            if (move_uploaded_file($file, $destination)) {
                $sql = "INSERT INTO report(branch, incidentArea, incidentCause, reportDate, victimFatality, victimInjured, victimSaved, asset_lost, asset_recover, contactMethod, personInCharge, repor
                $sql_run = mysqli_query($mysqli, $sql);

                if ($sql_run) {
                    $_SESSION['success'] = "You data is added and file is uploaded";
                    header('Location: report.php');
                } else {
                    $_SESSION['status'] = "Failed to add data and upload file";
                    header('Location: report.php');
                }
            }
        }
    } else {
        $sql = "INSERT INTO report(branch, incidentArea, incidentCause, reportDate, victimFatality, victimInjured, victimSaved, asset_lost, asset_recover, contactMethod, personInCharge, reportStatus)
        $sql_run = mysqli_query($mysqli, $sql);

        if ($sql_run) {
            $_SESSION['success'] = "Data added with no files uploaded";
            header('Location: report.php');
        } else {
            $_SESSION['status'] = "Data Not added with no files uploaded";
            header('Location: report.php');
        }
    }
}
```

Figure 4.27: Add report feature and file upload checking part 2

4.3.6 View of the view report page

After adding report, the users can open up the view report table page to check on the existing reports. The previously filled data will be showed here in a table using the DataTable jQuery plugins. By using the DataTable plugin, users would be able to type in keywords in the search field inside the table to search for specific row of data and also filter the rows of data by clicking on the table header of each column. In the table, admin users can choose to view, print, edit or delete the report data while normal users can only view and print the report data. The interface of the view report page from the admin view and normal user view are shown in Figure 4.28 and Figure 4.29. Figure 4.29 also shows that the data of the table has been filtered with search keywords. In Figure 4.30, the data are retrieved from the database using the mysqli_fetch_assoc() function and display in the table using a while loop. The DataTable plugin is also called by declaring the id as ‘dataTable’ and also declare the function in the JavaScript file.

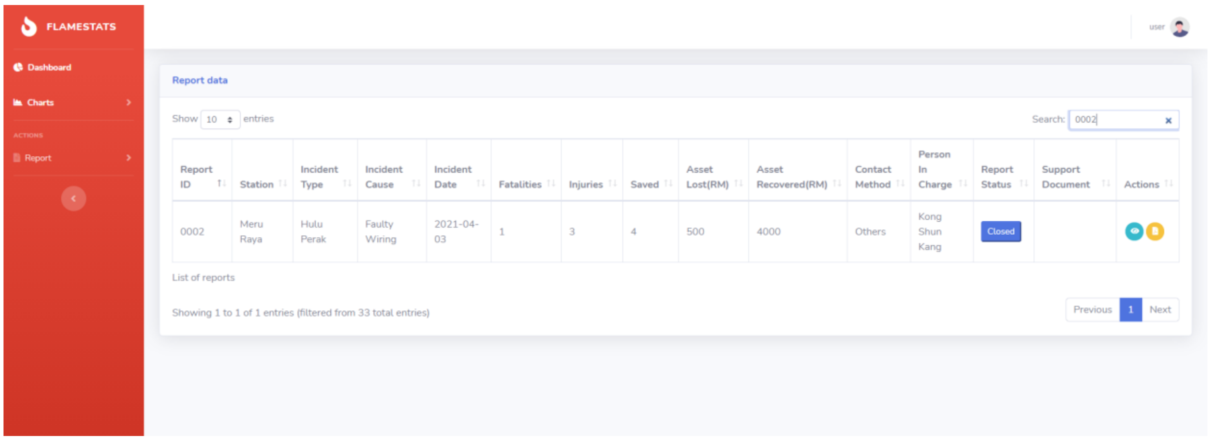


Report data

Show 10 entries Search:

Report ID	Station	Incident Area	Incident Cause	Incident Date	Fatalities	Injuries	Saved	Asset Lost(RM)	Asset Recovered(RM)	Contact Method	Person In Charge	Report Status	Support Document	Actions
0001	Stadium Ipoh	Manjung	Faulty Wiring	2021-04-01	1	2	4	500	1500	999	Kong Shun Kang	Closed		
0002	Meru Raya	Hulu Perak	Faulty Wiring	2021-04-03	1	3	4	500	4000	Others	Kong Shun Kang	Closed		
0003	Pengkalan	Kinta	Faulty Wiring	2021-03-31	0	1	4	600	200	Station Hotline	Kong Shun Kang	Closed		
0004	Gopeng	Kuala Kangsar	Others	2021-03-18	0	0	1	450	1700	Others	Kong Shun Kang	Pending		
0005	Stadium Ipoh	Others	Others	2021-04-01	0	1	2	250	1000	Others	Kong Shun Kang	Closed		
0006	Stadium Ipoh	Hulu Perak	Equipment Defective	2021-04-01	0	0	0	0	0	Station Hotline	Kong Shun Kang	Fake		
0007	Meru Raya	Others	Equipment Defective	2021-02-04	0	0	0	0	0			Closed		
0009	Stadium Ipoh	Hulu Perak	Faulty Wiring	2021-04-19	1	2	4	500	1500			Closed		
0010	Stadium Ipoh	Hulu Perak	Faulty Wiring	2021-04-18	0	2	4	560	1000			Closed		
0011	Pengkalan	Hulu Perak	Faulty Wiring	2021-05-05	1	2	2	600	1450			Closed		

Figure 4.28: Interface of admin view report



Report data

Show 10 entries Search: 0002

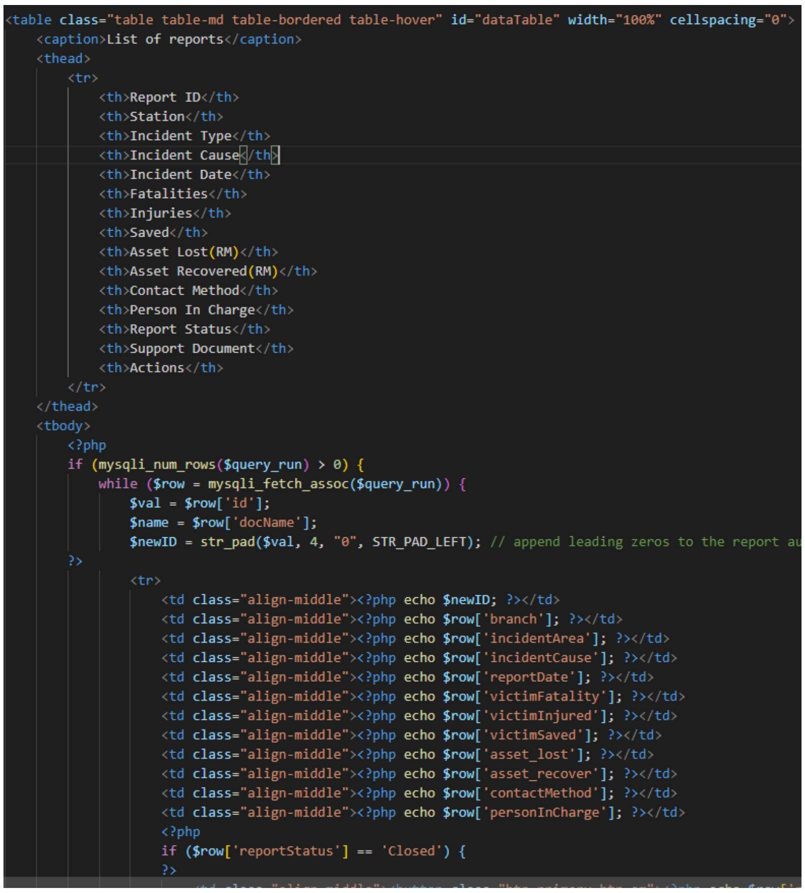
Report ID	Station	Incident Type	Incident Cause	Incident Date	Fatalities	Injuries	Saved	Asset Lost(RM)	Asset Recovered(RM)	Contact Method	Person In Charge	Report Status	Support Document	Actions
0002	Meru Raya	Hulu Perak	Faulty Wiring	2021-04-03	1	3	4	500	4000	Others	Kong Shun Kang	Closed		

List of reports

Showing 1 to 1 of 1 entries (filtered from 33 total entries)

Previous 1 Next

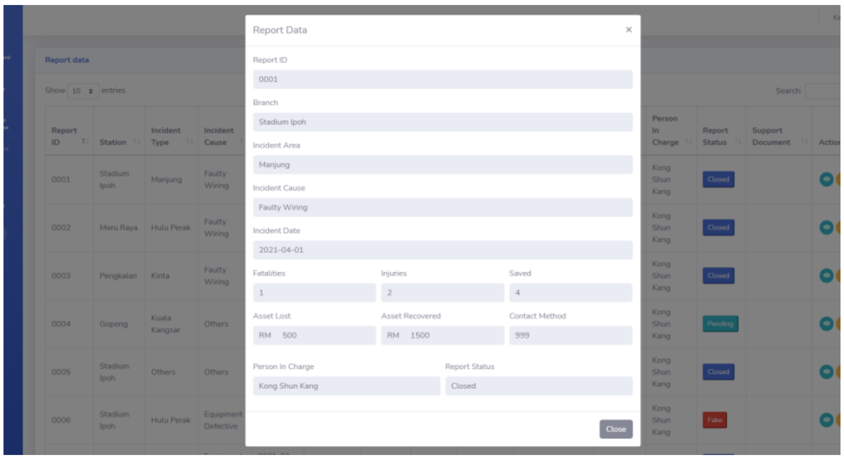
Figure 4.29: Interface of normal users view report and search report



```
<table class="table table-md table-bordered table-hover" id="dataTable" width="100%" cellspacing="0">
    <caption>List of reports</caption>
    <thead>
        <tr>
            <th>Report ID</th>
            <th>Station</th>
            <th>Incident Type</th>
            <th>Incident Cause</th>
            <th>Incident Date</th>
            <th>Fatalities</th>
            <th>Injuries</th>
            <th>Saved</th>
            <th>Asset Lost(RM)</th>
            <th>Asset Recovered(RM)</th>
            <th>Contact Method</th>
            <th>Person In Charge</th>
            <th>Report Status</th>
            <th>Support Document</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        <?php
        if (mysqli_num_rows($query_run) > 0) {
            while ($row = mysqli_fetch_assoc($query_run)) {
                $val = $row['id'];
                $name = $row['docName'];
                $newID = str_pad($val, 4, "0", STR_PAD_LEFT); // append leading zeros to the report au
        ?>
                <tr>
                    <td class="align-middle"><?php echo $newID; ?></td>
                    <td class="align-middle"><?php echo $row['branch']; ?></td>
                    <td class="align-middle"><?php echo $row['incidentArea']; ?></td>
                    <td class="align-middle"><?php echo $row['incidentCause']; ?></td>
                    <td class="align-middle"><?php echo $row['reportDate']; ?></td>
                    <td class="align-middle"><?php echo $row['victimFatality']; ?></td>
                    <td class="align-middle"><?php echo $row['victimInjured']; ?></td>
                    <td class="align-middle"><?php echo $row['victimSaved']; ?></td>
                    <td class="align-middle"><?php echo $row['asset_lost']; ?></td>
                    <td class="align-middle"><?php echo $row['asset_recover']; ?></td>
                    <td class="align-middle"><?php echo $row['contactMethod']; ?></td>
                    <td class="align-middle"><?php echo $row['personInCharge']; ?></td>
                    <?php
                    if ($row['reportStatus'] == 'Closed') {
                    ?>
```

Figure 4.30: Retrieval of data from database to display in table

For the features in the view report interface, if users choose to view a specific report data. They can click on the button with the eye icon to trigger the pop modal that display them the data of that specific report. Figure 4.31 shows the report popup modal .

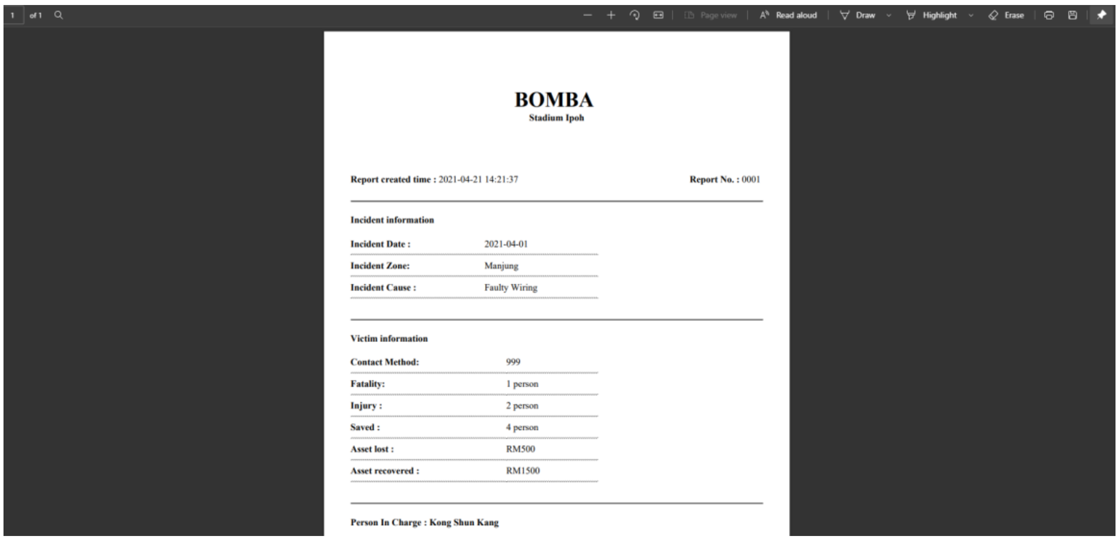


Report Data

Report ID	0001	
Branch	Stadium Ipoh	
Incident Area	Manjung	
Incident Cause	Faulty Wiring	
Incident Date	2021-04-01	
Fatalities: 1	Injuries: 2	Saved: 4
Asset Lost: RM 500	Asset Recovered: RM 1500	Contact Method: 999
Person In Charge: Kong Shun Kang	Report Status: Closed	

Figure 4.31: Report viewing popup modal

Then, for users that would like to print or download the report in PDF format, they can click the second button which will redirect the users to the report printing interface where users can choose to download the report in PDF format as well. Figure 4.32 shows the report viewing and printing page while Figure 4.33 and Figure 4.34 show the print.php file where the report layout is done here and converted to PDF format using the Dompdf html to PDF converter with the data retrieval from database.



BOMBA
Stadium Ipoh

Report created time : 2021-04-21 14:21:37 **Report No. :** 0001

Incident information

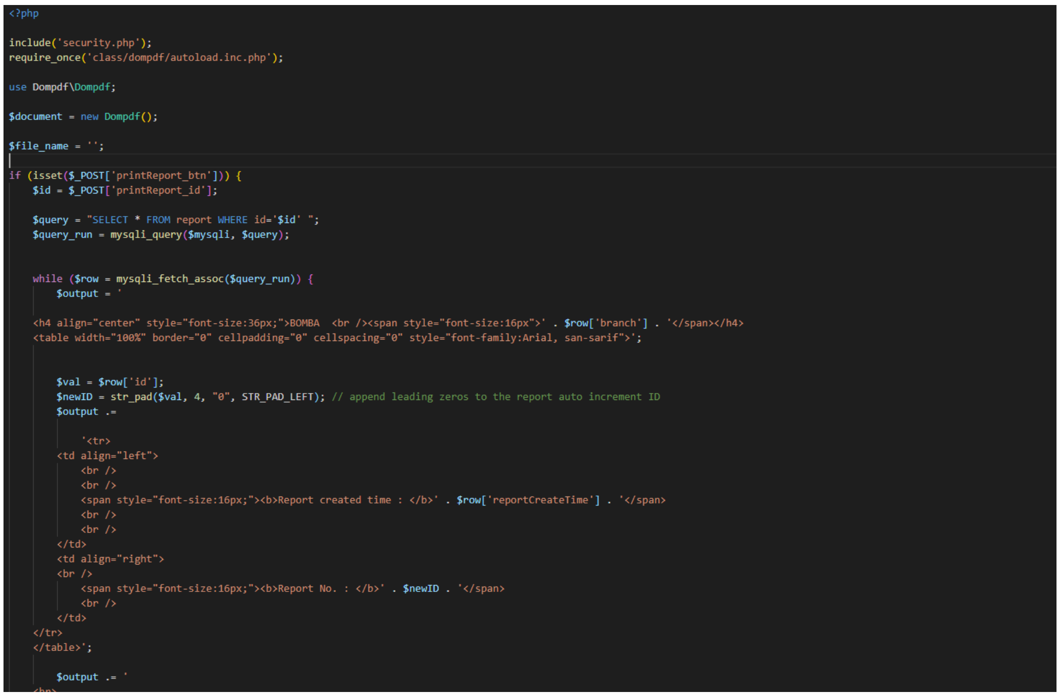
Incident Date :	2021-04-01
Incident Zone:	Manjung
Incident Cause :	Faulty Wiring

Victim information

Contact Method:	999
Fatality:	1 person
Injury :	2 person
Saved :	4 person
Asset lost :	RM500
Asset recovered :	RM1500

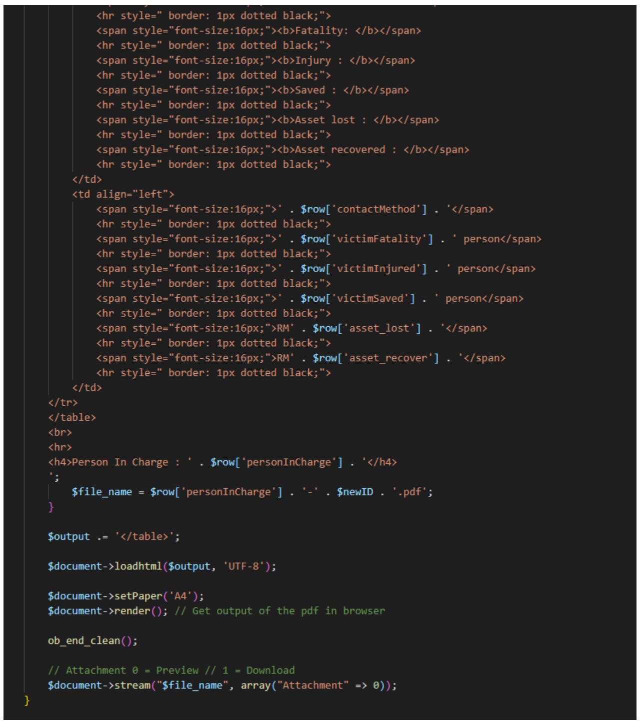
Person In Charge : Kong Shun Kang

Figure 4.32: Interface of report viewing and printing



```php
<?php

include('security.php');
require_once('class/dompdf/autoload.inc.php');

use Dompdf\Dompdf;

$document = new Dompdf();

$file_name = '';

if (isset($_POST['printReport_btn'])) {
    $id = $_POST['printReport_id'];

    $query = "SELECT * FROM report WHERE id='$id' ";
    $query_run = mysqli_query($mysqli, $query);


    while ($row = mysqli_fetch_assoc($query_run)) {
        $output = '

    <h4 align="center" style="font-size:36px;">BOMBA  <br /><span style="font-size:16px">' . $row['branch'] . '</span></h4>
    <table width="100%" border="0" cellpadding="0" cellspacing="0" style="font-family:Arial, san-sarif">';


        $val = $row['id'];
        $newID = str_pad($val, 4, "0", STR_PAD_LEFT); // append leading zeros to the report auto increment ID
        $output .=

            '<tr>
        <td align="left">
            <br />
            <br />
            <span style="font-size:16px;"><b>Report created time : </b>' . $row['reportCreateTime'] . '</span>
            <br />
            <br />
        </td>
        <td align="right">
        <br />
            <span style="font-size:16px;"><b>Report No. : </b>' . $newID . '</span>
            <br />
        </td>
    </tr>
    </table>';

        $output .= '
```

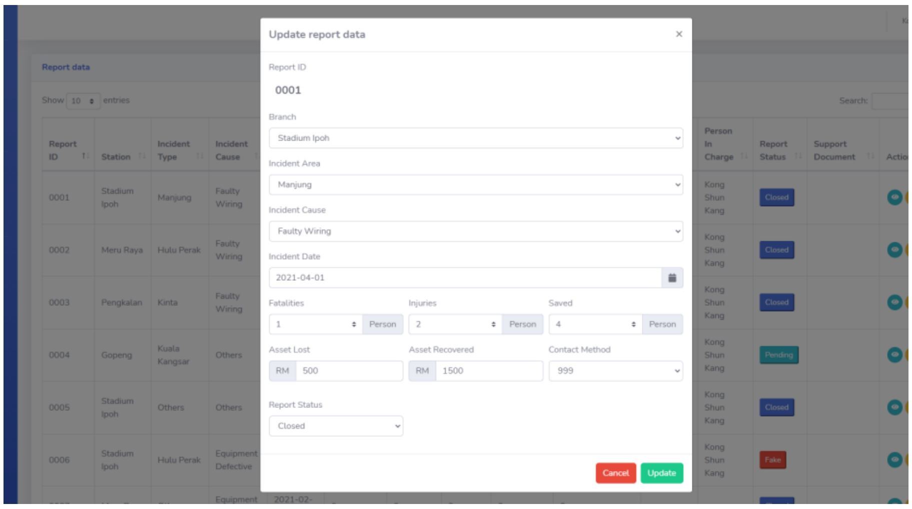
Figure 4.33: Report html to PDF convert and layout design part 1



```
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;"><b>Fatality: </b></span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;"><b>Injury : </b></span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;"><b>Saved : </b></span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;"><b>Asset lost : </b></span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;"><b>Asset recovered : </b></span>
                <hr style=" border: 1px dotted black;">
            </td>
            <td align="left">
                <span style="font-size:16px;">' . $row['contactMethod'] . '</span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;">' . $row['victimFatality'] . ' person</span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;">' . $row['victimInjured'] . ' person</span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;">' . $row['victimSaved'] . ' person</span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;">RM' . $row['asset_lost'] . '</span>
                <hr style=" border: 1px dotted black;">
                <span style="font-size:16px;">RM' . $row['asset_recover'] . '</span>
                <hr style=" border: 1px dotted black;">
            </td>
        </tr>
        </table>
        <br>
        <hr>
        <h4>Person In Charge : ' . $row['personInCharge'] . '</h4>
        ';
            $file_name = $row['personInCharge'] . '-' . $newID . '.pdf';
        }

        $output .= '</table>';

        $document->loadhtml($output, 'UTF-8');

        $document->setPaper('A4');
        $document->render(); // Get output of the pdf in browser

        ob_end_clean();

        // Attachment 0 = Preview // 1 = Download
        $document->stream("$file_name", array("Attachment" => 0));
    }
```

Figure 4.34: Report html to PDF convert and layout design part 2

Other than viewing and printing the report, admin users can edit the report by clicking the third button. A popup modal will be displayed and allow the admin users to update the data. After clicking the update button, the data will be updated and stored into the database. Figure 4.35 shows the edit report popup modal while Figure 4.36 shows the backend implementation of the edit function where the POST data from the update modal is handled and update the specific row according to the report ID.



Update report data

Report ID: 0001

Branch: Stadium Ipoh

Incident Area: Manjung

Incident Cause: Faulty Wiring

Incident Date: 2021-04-01

Fatalities: 1 Person

Injuries: 2 Person

Saved: 4 Person

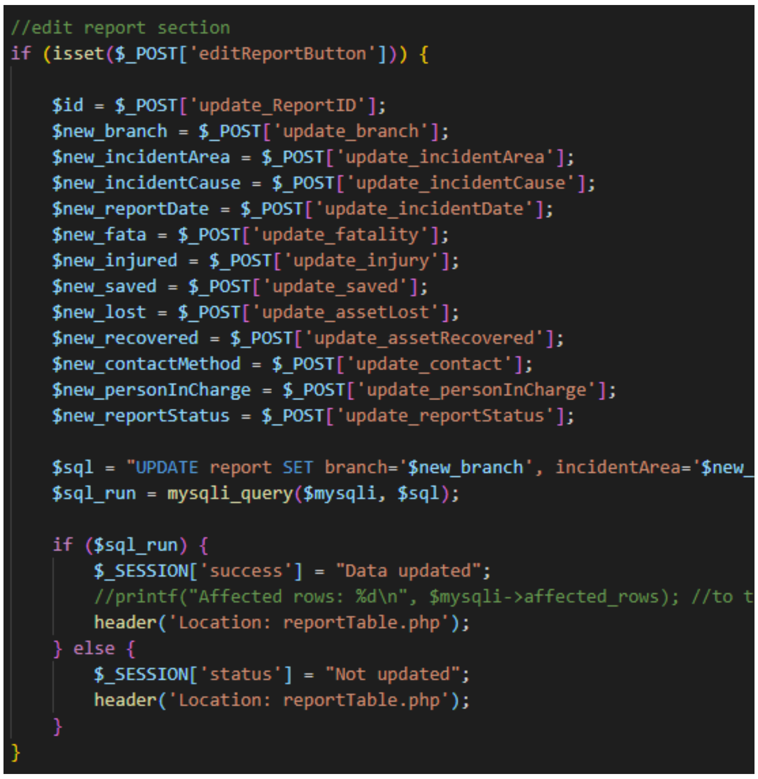
Asset Lost: RM 500

Asset Recovered: RM 1500

Contact Method: 999

Report Status: Closed

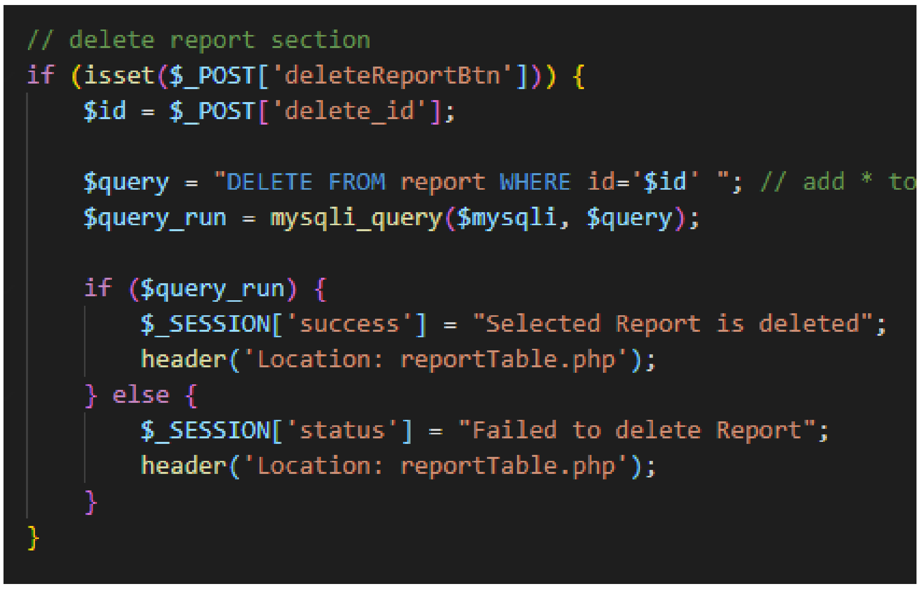
Figure 4.35: Edit report popup modal



```php
//edit report section
if (isset($_POST['editReportButton'])) {

    $id = $_POST['update_ReportID'];
    $new_branch = $_POST['update_branch'];
    $new_incidentArea = $_POST['update_incidentArea'];
    $new_incidentCause = $_POST['update_incidentCause'];
    $new_reportDate = $_POST['update_incidentDate'];
    $new_fata = $_POST['update_fatality'];
    $new_injured = $_POST['update_injury'];
    $new_saved = $_POST['update_saved'];
    $new_lost = $_POST['update_assetLost'];
    $new_recovered = $_POST['update_assetRecovered'];
    $new_contactMethod = $_POST['update_contact'];
    $new_personInCharge = $_POST['update_personInCharge'];
    $new_reportStatus = $_POST['update_reportStatus'];

    $sql = "UPDATE report SET branch='$new_branch', incidentArea='$new_
    $sql_run = mysqli_query($mysqli, $sql);

    if ($sql_run) {
        $_SESSION['success'] = "Data updated";
        //printf("Affected rows: %d\n", $mysqli->affected_rows); //to t
        header('Location: reportTable.php');
    } else {
        $_SESSION['status'] = "Not updated";
        header('Location: reportTable.php');
    }
}
```

Figure 4.36: Update the new data into the database

Last feature for the view report table will be the delete function where it allows the admin users to delete the report data by clicking the delete button with a trashcan icon. Figure 4.37 shows the backend part of the delete function where the specific row with the id will be deleted from the database followed by the session message that notify users about the deletion status.

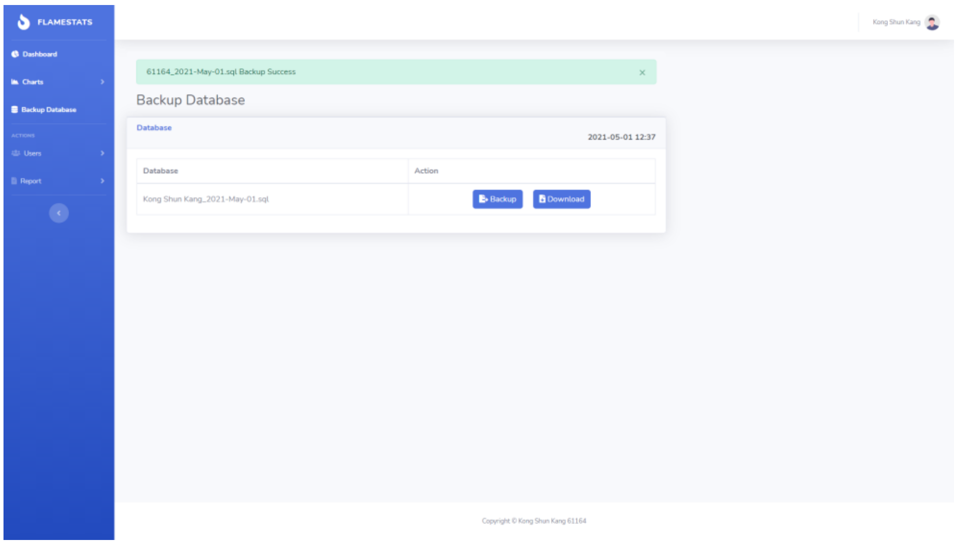


```php
// delete report section
if (isset($_POST['deleteReportBtn'])) {
    $id = $_POST['delete_id'];

    $query = "DELETE FROM report WHERE id='$id' "; // add * to

    $query_run = mysqli_query($mysqli, $query);

    if ($query_run) {
        $_SESSION['success'] = "Selected Report is deleted";
        header('Location: reportTable.php');
    } else {
        $_SESSION['status'] = "Failed to delete Report";
        header('Location: reportTable.php');
    }
}
```

Figure 4.37: Delete report data from database

4.3.7 View of the database backup page

The database backup function will be available for admin users to backup and download the current database and the records in it. The database naming will be following the user registered name append to the system date during user backup the database. In the backup database interface, users are required to click the backup button first before proceeding to download the database. Figure 4.38 shows the interface of the database backup with the alert message when a database is backed up successfully.



61164_2021-May-01.sql Backup Success

Backup Database

Database

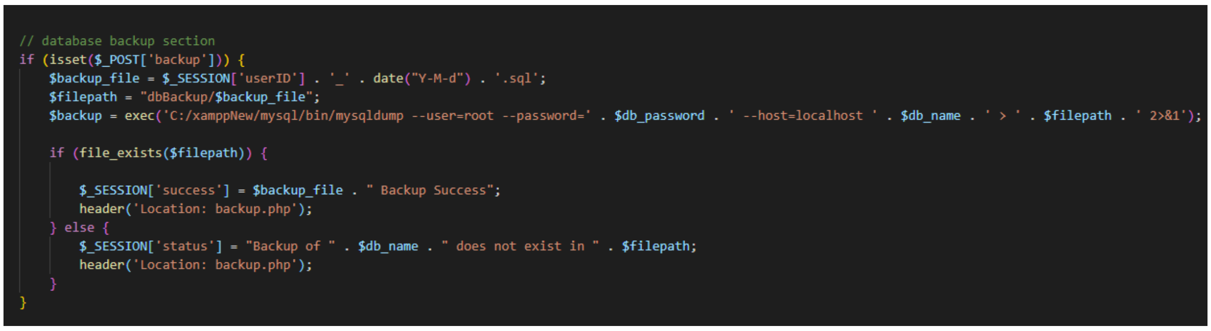
2021-05-01 12:37

Database	Action
Kong Shun Kang_2021-May-01.sql	Backup Download

Copyright © Kong Shun Kang 61164

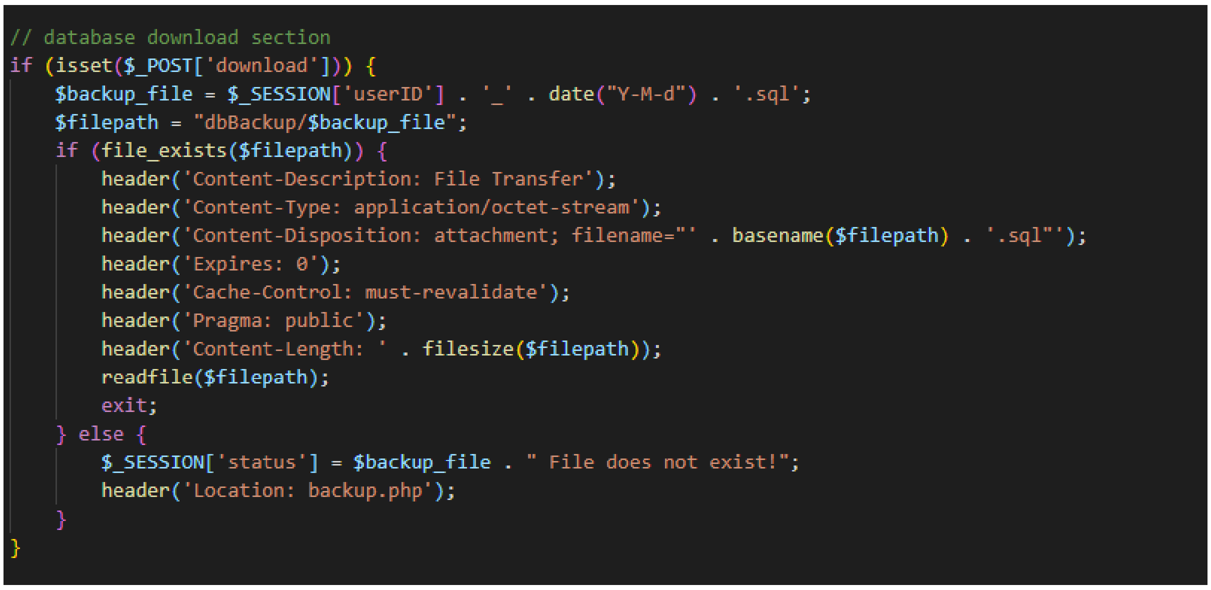
Figure 4.38:Interface of database backup page after successfully backup

The database backup function is implemented by using the mysqldump function that comes with PHP. The backup sql file will be stored in a folder named ‘dbBackup’ as the path is declared during the implementation of the database backup function. Figure 4.39 shows the implementation of the backup function where the path of the backup folder and file naming are set. Then, Figure 4.40 shows the download function that check whether the backup sql file is exist in the declared file path, if exist, the download will be proceeded. Figure 4.41 shows the sql file created from the mysqldump function. The sql file can be imported using phpMyAdmin to create a backup database that stores the same tables and records.



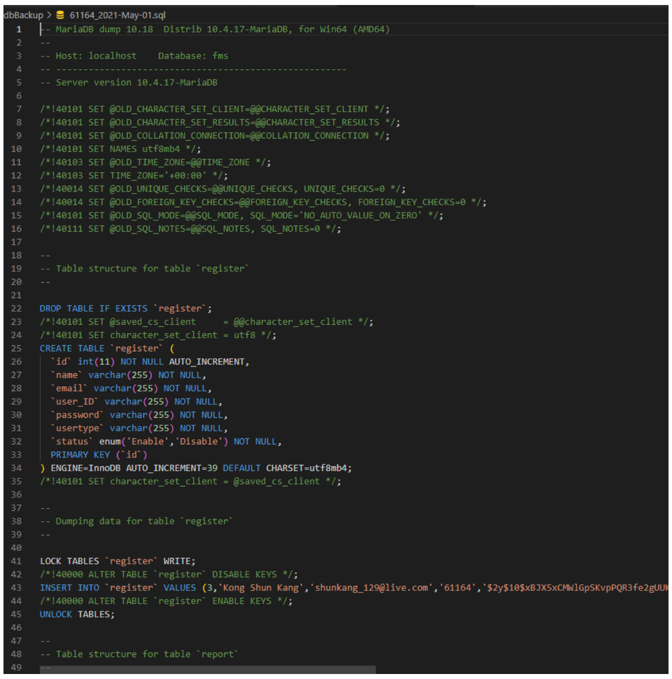
```php
// database backup section
if (isset($_POST['backup'])) {
    $backup_file = $_SESSION['userID'] . '_' . date("Y-M-d") . '.sql';
    $filepath = "dbBackup/$backup_file";
    $backup = exec('C:/xamppNew/mysql/bin/mysqldump --user=root --password=' . $db_password . ' --host=localhost ' . $db_name . ' > ' . $filepath . ' 2>&1');

    if (file_exists($filepath)) {

        $_SESSION['success'] = $backup_file . " Backup Success";
        header('Location: backup.php');
    } else {
        $_SESSION['status'] = "Backup of " . $db_name . " does not exist in " . $filepath;
        header('Location: backup.php');
    }
}
```

Figure 4.39: Backup function



```php
// database download section
if (isset($_POST['download'])) {
    $backup_file = $_SESSION['userID'] . '_' . date("Y-M-d") . '.sql';
    $filepath = "dbBackup/$backup_file";
    if (file_exists($filepath)) {
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename="' . basename($filepath) . '.sql"');
        header('Expires: 0');
        header('Cache-Control: must-revalidate');
        header('Pragma: public');
        header('Content-Length: ' . filesize($filepath));
        readfile($filepath);
        exit;
    } else {
        $_SESSION['status'] = $backup_file . " File does not exist!";
        header('Location: backup.php');
    }
}
```

Figure 4.40: File checking and download function



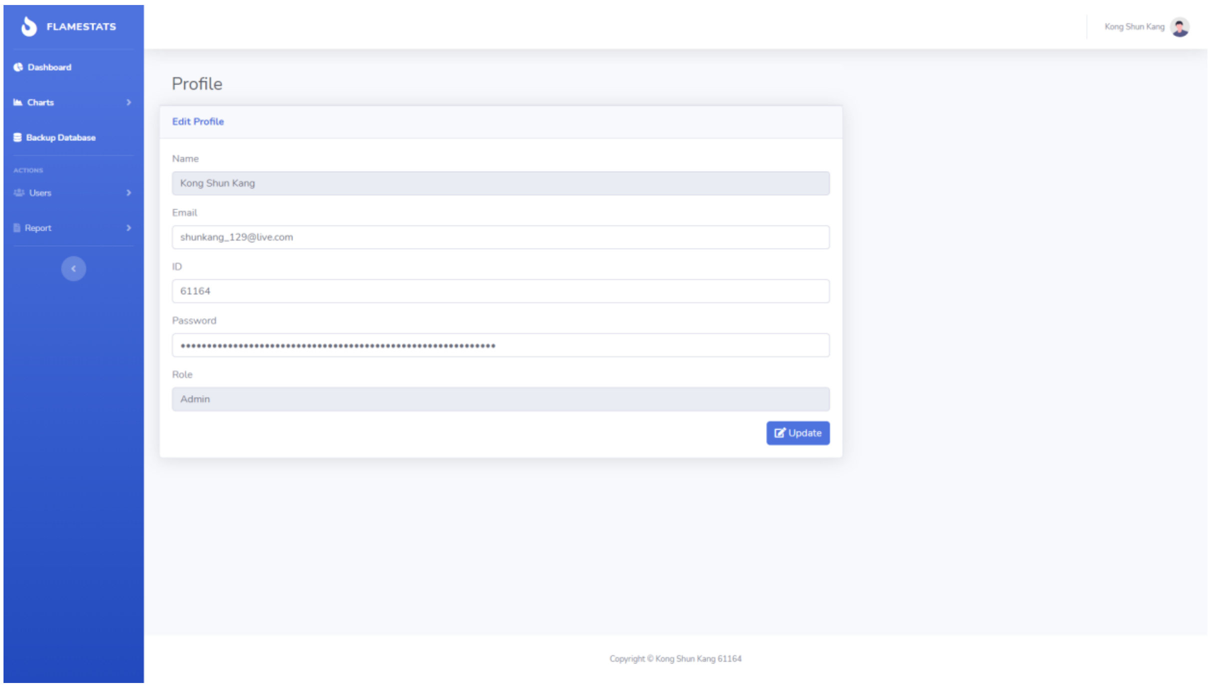
dbBackup > 61164_2021-May-01.sql

```sql
-- MariaDB dump 10.18  Distrib 10.4.17-MariaDB, for Win64 (AMD64)
--
-- Host: localhost    Database: fms
-- ------------------------------------------------------
-- Server version 10.4.17-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `register`
--

DROP TABLE IF EXISTS `register`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `register` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `user_ID` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `usertype` varchar(255) NOT NULL,
  `status` enum('Enable','Disable') NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=39 DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `register`
--

LOCK TABLES `register` WRITE;
/*!40000 ALTER TABLE `register` DISABLE KEYS */;
INSERT INTO `register` VALUES (3,'Kong Shun Kang','shunkang_129@live.com','61164','$2y$10$xBJX5xCMWlGpSKvpPQR3fe2gUUK
/*!40000 ALTER TABLE `register` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `report`
--
```

Figure 4.41: The backup sql file created from mysqldump

4.3.8 View of the user profile page

Users can view and update their own account info by clicking the profile button in the dropdown of their account image at the top right of the top navigation bar. In the user profile page as shown in Figure 3.42, users can check on their current data or choose to update the data by filling in new data before clicking on the update button. Users are required to fill in either the old password or new password to update the profile data. Figure 3.43 and Figure 3.44 shows the implementation of the profile update function where the data from the form will be passed using HTTP POST method from the form in Figure 4.43 to update the existing row in the database.



Profile

Edit Profile

Name
Kong Shun Kang

Email
shunkang_129@live.com

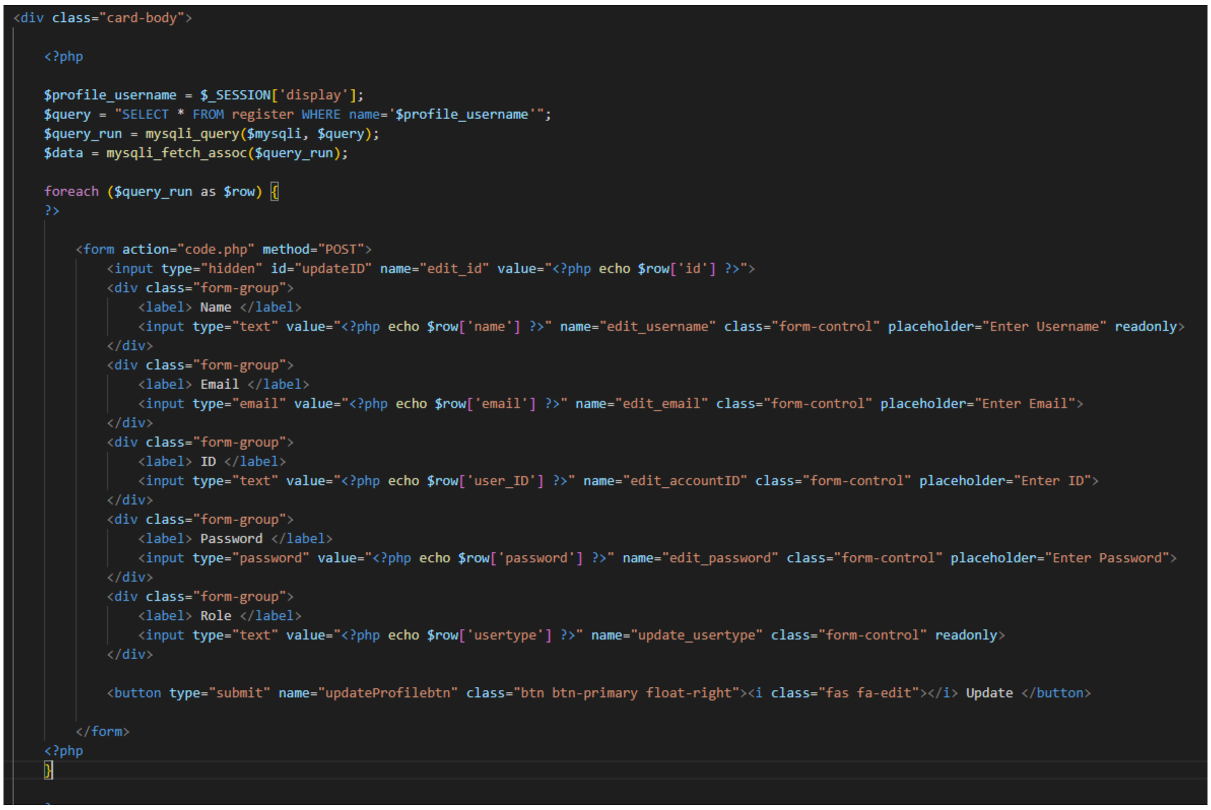
ID
61164

Password
••

Role
Admin

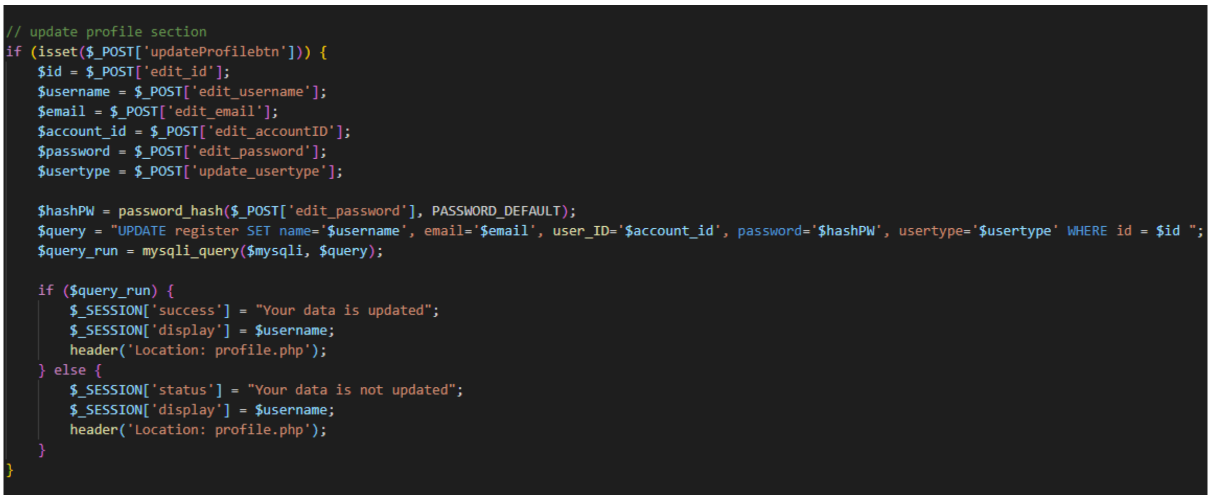
Copyright © Kong Shun Kang 61164

Figure 4.42: Interface of view and update user profile



```
<div class="card-body">

    <?php

    $profile_username = $_SESSION['display'];
    $query = "SELECT * FROM register WHERE name='$profile_username'";
    $query_run = mysqli_query($mysqli, $query);
    $data = mysqli_fetch_assoc($query_run);

    foreach ($query_run as $row) {
    ?>

        <form action="code.php" method="POST">
            <input type="hidden" id="updateID" name="edit_id" value="<?php echo $row['id'] ?>">
            <div class="form-group">
                <label> Name </label>
                <input type="text" value="<?php echo $row['name'] ?>" name="edit_username" class="form-control" placeholder="Enter Username" readonly>
            </div>
            <div class="form-group">
                <label> Email </label>
                <input type="email" value="<?php echo $row['email'] ?>" name="edit_email" class="form-control" placeholder="Enter Email">
            </div>
            <div class="form-group">
                <label> ID </label>
                <input type="text" value="<?php echo $row['user_ID'] ?>" name="edit_accountID" class="form-control" placeholder="Enter ID">
            </div>
            <div class="form-group">
                <label> Password </label>
                <input type="password" value="<?php echo $row['password'] ?>" name="edit_password" class="form-control" placeholder="Enter Password">
            </div>
            <div class="form-group">
                <label> Role </label>
                <input type="text" value="<?php echo $row['usertype'] ?>" name="update_usertype" class="form-control" readonly>
            </div>

            <button type="submit" name="updateProfilebtn" class="btn btn-primary float-right"><i class="fas fa-edit"></i> Update </button>

        </form>
    <?php
    }
```

Figure 4.43: Frontend of viewing profile form and display data from database

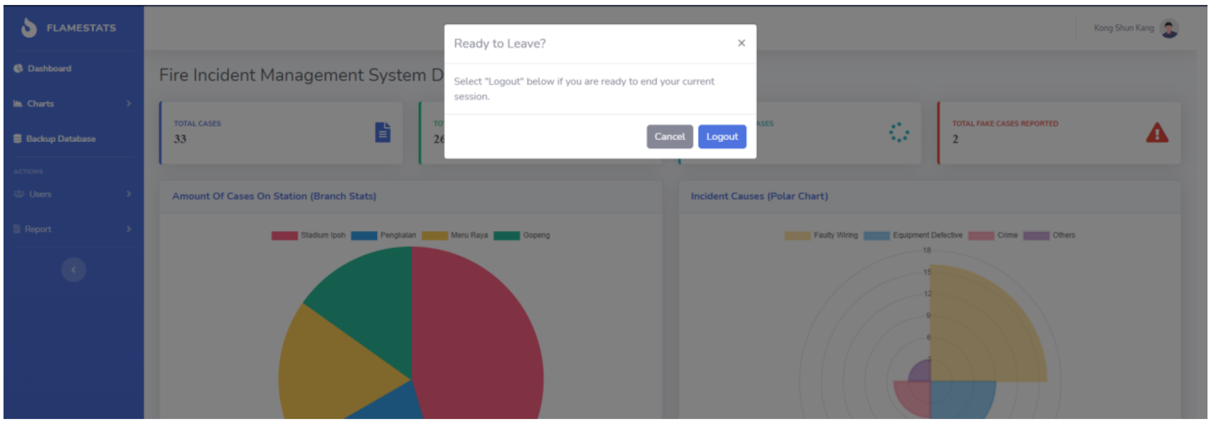


```
// update profile section
if (isset($_POST['updateProfilebtn'])) {
    $id = $_POST['edit_id'];
    $username = $_POST['edit_username'];
    $email = $_POST['edit_email'];
    $account_id = $_POST['edit_accountID'];
    $password = $_POST['edit_password'];
    $usertype = $_POST['update_usertype'];

    $hashPW = password_hash($_POST['edit_password'], PASSWORD_DEFAULT);
    $query = "UPDATE register SET name='$username', email='$email', user_ID='$account_id', password='$hashPW', usertype='$usertype' WHERE id = $id ";
    $query_run = mysqli_query($mysqli, $query);

    if ($query_run) {
        $_SESSION['success'] = "Your data is updated";
        $_SESSION['display'] = $username;
        header('Location: profile.php');
    } else {
        $_SESSION['status'] = "Your data is not updated";
        $_SESSION['display'] = $username;
        header('Location: profile.php');
    }
}
```

Figure 4.44: Update user profile function

4.3.9 View of the logout function

Lastly, users can proceed to logout after confirming the logout at the popup modal triggered as shown in Figure 4.45. Then, Figure 4.46 shows the implementation of logout function where the system sessions will be unset and destroyed before redirecting users back to the login page for the next login session.

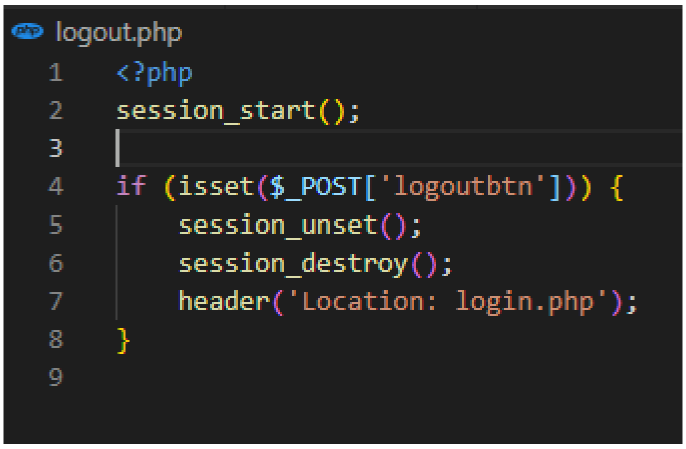


Ready to Leave?

Select "Logout" below if you are ready to end your current session.

Cancel Logout

Figure 4.45: Logout popup modal


logout.php

```php
<?php
session_start();

if (isset($_POST['logoutbtn'])) {
    session_unset();
    session_destroy();
    header('Location: login.php');
}

```

Figure 4.46: Logout session destroy and unset

4.4 Summary

The implementation and development of the proposed system is showcased and explained throughout this chapter with attachment of images of the codes and interfaces. However, the system will continue to be improved and enhanced to ensure the system's user friendliness to be high and achieving the project objectives which are to design and develop a dashboard-based management system that provides better data visualization for the fire department. After the implementation phase, the next phase of the project which is system testing and evaluation will be carried out and justified in the next chapter.