FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

BIC 21203
WEB DEVELOPMENT
SEMESTER 1 SESSION 2023/2024

PROJECT REPORT

ONLINE CREWS SCHEDULE SYSTEM FOR RAILWAY TRANSPORTATION WITH PASSWORD HASHING, CAPTCHA AND TWO FACTOR AUTHENTICATION (STUDY CASE : KERETA TANAH API MELAYU (KTM))

GROUP 7

GROUP MEMBERS	MATRIC NO.
MUHAMMAD KHAIRUL IKHWAN BIN IBRAHIM	CI210014
YAMUNAH A/P K.RAGUBATHI	DI210001
AIN ZULAIKHA BINTI AHMAD NADZRI	AI210276
MUHAMMAD SYAHMI IRFAN BIN SAZAKI	AI210157
AFIF BIN ARIS	AI210187
BUNGA LAELATUL MUNA	JI230003

Abstract

The project, titled "Online Crews Schedule System for Railway Transportation with Enhanced Security Features," introduces a comprehensive solution to the intricate scheduling demands within railway transportation. Acknowledging the vital significance of proficient staff scheduling for peak railway performance, the system incorporates cutting-edge security features such as role-based access, password hashing, and CAPTCHA. This effort fills a common hole in the systems that are currently in place, where security is frequently inadequate and might endanger critical data. The main goals of the research are real-time data integration, automated scheduling algorithms, and user authentication in order to create a system that is both secure and easy to use. Using best practices in web development, the multifaceted approach takes scalability and responsive design into account. The main takeaway from the project is that improved operational efficiency may be achieved by creating a dependable Online Crews Schedule System that is customized for railway problems. The effective integration of security measures and the beneficial effects of automated algorithms and real-time data on staff scheduling efficiency in railway transportation are highlighted by key findings. By resolving security flaws in current systems and setting the stage for future developments in safe scheduling within the larger transportation industry, this research contributes significantly to the area overall.

1.0 Introduction

Railway transportation plays a pivotal role in modern society, connecting people and goods across vast distances (Smith, 2018). As we increasingly rely on digital technologies to manage these transportation systems efficiently, the need for robust cybersecurity measures becomes imperative.

KTM Intercity is the brand name for a group of diesel-hauled intercity train services in Peninsular Malaysia, Southern Region, Singapore and Thailand operated by Keretapi Tanah Melayu Berhad (KTMB). KTM itself has an information system that displays general information about train departures.

This research, the author wants to create a sechedule management system to monitor the train crew schedule. And the system will certainly require a security component to protect company and crew data. This proposal outlines the development of a comprehensive management system for KTM railway transportation, incorporating essential cybersecurity components to safeguard against potential cyber threats.

1.1 Objectives

The primary objective of this project is to design and implement a sophisticated railway transportation management system that enhances operational efficiency and ensures the security of critical data and infrastructure. The specific objectives include:

1. Developing a user-friendly interface for efficient management of railway operations.
2. Implementing a role-based access control system (UserRole), reCAPTCHA, and password hashing as cybersecurity components.
3. Mitigating the risk of cyberattacks to ensure the integrity, confidentiality, and availability of railway system data.

1.2 Problem Statement

The railway transportation sector faces an escalating threat landscape of cyberattacks, ranging from unauthorized access to sensitive information to potential disruptions of critical infrastructure. Traditional management systems may not be adequately equipped to handle these threats, necessitating the development of a modernized system with robust cybersecurity features.

1.3 Scope

The scope of this project encompasses the design and implementation of a comprehensive railway transportation management system with a focus on cybersecurity. The system will cover aspects such as train scheduling, maintenance, passenger management, and cargo tracking. The cybersecurity components, including UserRole, reCAPTCHA, and password hashing, will be integrated to fortify the system against various cyber threats.

1.4 Significance of Developing the System

The development of this railway transportation management system is significant for several reasons:

- Enhanced Operational Efficiency: The system will streamline railway operations, leading to improved efficiency and reliability in transportation services.
- Cybersecurity Assurance: By incorporating UserRole, reCAPTCHA, and password hashing, the system will provide a robust defense against cyber threats, ensuring the integrity and confidentiality of sensitive data.
- Infrastructure Resilience: The implementation of advanced cybersecurity measures will enhance the resilience of critical railway infrastructure, safeguarding it against potential disruptions.
- User Experience: The user-friendly interface will contribute to a positive user experience for railway operators and stakeholders, promoting ease of use and adoption.

In subsequent chapters, we will delve into the detailed design and implementation of the railway transportation management system, emphasizing the integration of cybersecurity components, particularly the role-based access control system (UserRole), to fortify the system against cyber threats.

2.0 Literature Review

The railway crew system plays a critical role in ensuring the safe and efficient operation of railways. As such, understanding the various aspects of the railway crew system is crucial for optimizing its performance. In this literature review, we aim to explore the existing research and literature related to railway crew systems, with a focus on factors such as existing attacks on railway system, security implementation for the system, system comparison and requirements. This literature review will serve as a foundation for our report, providing a comprehensive understanding of the key issues and opportunities in this important area of railway operations.

2.1 Cybersecurity Attack in Railway System

The increasing reliance on digital technology in railway systems has brought upon significant advancements in efficiency and safety. However, this digitalization has also made railway networks vulnerable to cyber threats. Cyberattacks on railway systems pose serious risks to passenger safety, operational continuity, and the overall integrity of transportation infrastructure as public spaces are considered fragile and prone to get attacked.

The railway industry is still in the process of development, with limited awareness of emerging risks, and these risks are not given significant consideration due to the perceived high level of safety within the railway domain. There are multiple ways for attackers to get their hands on the system as stated in the research by (Thaduri et al., 2019). Table 2.1 listed multiple attacks starting from 2008 to 2017.

Table 2.1: Attacks on Cybersecurity in Railway System

YEAR	**ATTACKS**
2008	Using a TV remote, an individual derailed four tram trains in Lodz, Poland.
2011	In 2011, a band of pirates launched an assault on remote computers, causing a two-day halt of the train signaling system in the Northwestern United States.
2013	The Belgian national railway (NMBS) accidentally disclosed personal information of multiple customers.
2014	The release of data on several million journeys taken by New York taxis in a year was triggered by an anonymous information request.
2015	There was a suspected pirating of subway system in Seoul, Korea.
2016	Ticketing system at San Fransisco got attacked by ransomware.

2017	The Swedish Transport Agency exposed various driver information to Eastern Europe as a result of unauthorized data access within IBM systems.

2.2 Railway System Comparison

As the world becomes more interconnected, the potential for cyberattacks on railway systems has become a critical concern for governments, transportation authorities, and cybersecurity experts. Understanding the landscape of cyber threats facing railway systems is essential for developing effective mitigation strategies and safeguarding the integrity of critical transportation infrastructure. Thus, is why a few have come up with great and splendid systems with secure security features to ensure the safety of railway systems that will be used by many.

When comparing different railway systems, it's important to consider various factors such as security of the system, user friendly aspect and user interface design. Understanding the strengths and weaknesses of each system can provide valuable insights for policymakers, investors, and travelers as they will be the ones using this system. This comparison aims to analyze and contrast the features of prominent railway systems across different regions, highlighting the security features implemented and how their system works. By delving into these details, we can gain a comprehensive understanding of how various railway systems function.

The compared system will be between MyRapid(*Official Website for Rapid KL, Rapid Penang and Rapid Kuantan by Prasarana Malaysia Berhad*, n.d.), MyMRT(*Home - MRT Corp*, n.d.) and Railway Online Booking System by (Mohammed Moseeur Rahman Assistant Professor, n.d.). All these three systems in Table 2.2 are related to the train or railway management as to why we choose them to compare them to our system. MyRapid and MyMRT are websites where users can view the train’s schedule and the destinations of the trains, meanwhile Railway Online Booking System is a booking services for the railway where users can book their tickets. Not just for services, MyRapid also informs users on any events that MyRapid is having, such as collaboration events or free ride days. It is no surprise that MyRapid is well-received by the citizens of Malaysia.

Table 2.2 System Comparison

SYSTEM	User	Good User	SECURITY IMPLEMENTATION

	Friendly	Interface Design	Password Hashing	Captcha	2 Factor Authentication
MyRapid	YES	YES	-NOT STATED-	YES	NO
MyMRT	YES	YES	-NOT STATED-	NO	NO
Railway Online Booking System	YES	NO	-NOT STATED-	NO	NO

2.3 Technical Details

The crews' schedules system for railway transportation technical specification cover a wide range of elements and factors. Using the common web technologies like HTML, CSS, JavaScript and PHP used for database management. The crews' schedules system for railway transportation system is made to be cross-browser friendly, supporting well-known web browsers like Chrome, Firefox, Safari, and Edge to ensure broad compatibility and responsiveness. It uses responsive design principles to adapt fluidly to various screen sizes and resolutions, providing an excellent viewing experience for visitors accessing the site from PCs, laptops, tablets, and mobile phones.

User Requirement

The requirements for the railway online booking system cover user expectations, system functionality, and the required hardware and software elements. According to the user requirements, the main objective is to develop a user-friendly interface for efficient management of railway operations. The Railway Crew Schedule system managing jobs and assigning them to employees in the form of schedules, to ensure that a job is done on a specific day and within a specific period. Crews have a consolidated view of their schedule for the entire day which in turn helps them to gain more clarity and focus on their schedule. Only the administrator has permission to add or modify previous schedules.

System Requirement

Efficient Server-Side Processing	Server-side processing on the website effectively manages user requests and communications with the database. Programming languages and frameworks designed to manage dynamic server-side functionality.
Responsiveness and Adaptability	The website will be responsively developed, which means it will adjust to various screen sizes and resolutions. This makes sure that the website works and shows properly across a range of gadgets, including desktops, laptop, tablet, and mobile phones.
Popular web browser compatibility	The website be compatible with Chrome, Firefox, Safari, and Edge, among others. This guarantees that users, irrespective of their favorite browser, may view and engage with the website without any issues.

2.4 Hardware Requirement & Software Requirement

- Database Management System (DBMS)

  Data management and storage for the website require a DBMS. MySQL, PostgreSQL, MongoDB, or SQLite are popular options. Selection criteria include data structure needs, scalability, and particular features each DBMS offers.

- Web Server

  A web server is necessary to host and provide users with access to the website. Popular web servers include Microsoft IIS, Nginx, and Apache. The choice is made based on the operating system, performance specifications, and compatibility of the web server with other software elements.

2.5 Language

- Hypertext Markup Language (HTML) for structuring and presenting content.
- Cascading Style Sheets (CSS) for styling and designing the layout of web pages.
- Server-side scripting language like PHP for dynamic server-side functionality.

2.6 Database

Databases are used to collect, store, and manage a group of data in a structured way. In a relational structured database, there are tables that store data. Database used to store and manage huge volume of data.

3.0 Methodology

3.1 Requirement Gathering

SDLC Model represents the process of developing software. Our group used the agile SDLC model, and the title of our project is Online Crews Schedules System for Railway Transportation. This system will display the crew's schedules based on their departments. In this phase, all the relevant information is collected from the crews to develop the system based on their expectations and needs. We need to assess the technical feasibility study. We need to identify the challenges that may affect the project's success. Next, study projected costs and returns on investment for the project. We must consider factors like expected revenue, budgets, etc. For example, the admin wants a system involving crew schedules. In this case, the requirements must be clear like the crew's name, schedules, and their departments. Once the requirement gathering is done, it will go to the next phase which is the analysis phase.

3.2 Analysis

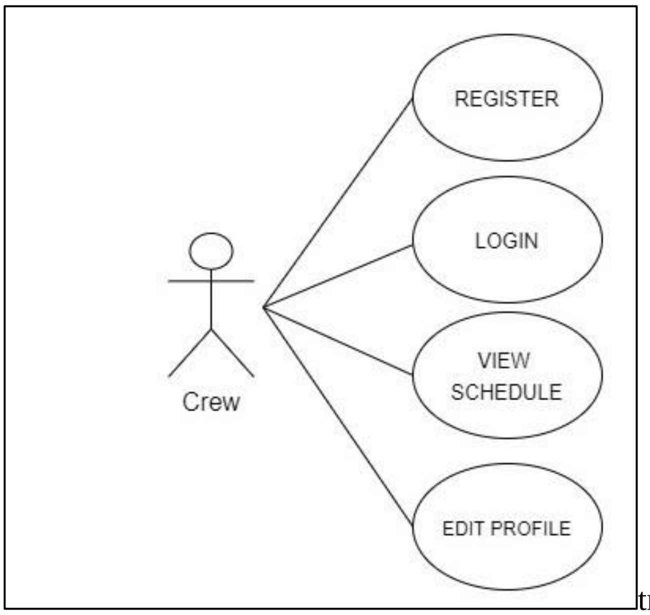
The Agile Software Development Life Cycle (SDLC) for our project, the creation of an online crew scheduling system for railway transportation, places a lot of emphasis on the requirement gathering phase. This stage is crucial since it establishes the overall project's direction. To match the software with the expectations and requirements of the crews, and the system's end users, the team gathers all pertinent data from them during this phase. The requirements must be detailed and explicit in order for the project to be successful. The crew's timetables based on their separate departments are the main necessities in this situation. These specifications provide vital information that is necessary for the system to operate well, such as crew names, timetables, and departments.

3.3 Unified Modeling Language (UML)

Unified Modelling Language (UML) is a diagram that specifies, visualizes, and constructs plans for model of system through a picture (*What Is Unified Modeling Language (UML)?*, n.d.). An example of UML diagram is using case diagram and sequence diagram. The use case diagram will show the functional requirements of the system. It is particularly useful in visualizing the functional requirements of a system, showing how different users interact with the system to achieve specific goals.

3.3.1 Use Case Diagram

A use case diagram is used to explain more about requirements of a system from an external point of view. It is particularly useful in visualizing the functional requirements of a system, showing how different users interact with the system to achieve specific goals. The Use-Case diagram also identifies any internal or external factors that may influence the system and should be taken into consideration (*Use Case Diagrams - Use Case Diagrams Online, Examples, and Tools*, n.d.).



tr

Figure 3.1: Crew Use-Case Diagram

Based on Figure 3.1 Shows the crew use case diagram that consists of four functions. Users can register their account by entering the crew ID, full name, email, username, password, department and user role. Before completing the registration, the system enhances security by user must complete the captcha. User must log in to access the system by enter their crew ID, password and user role on the login page. User can view their schedule and edit the profile in the system.

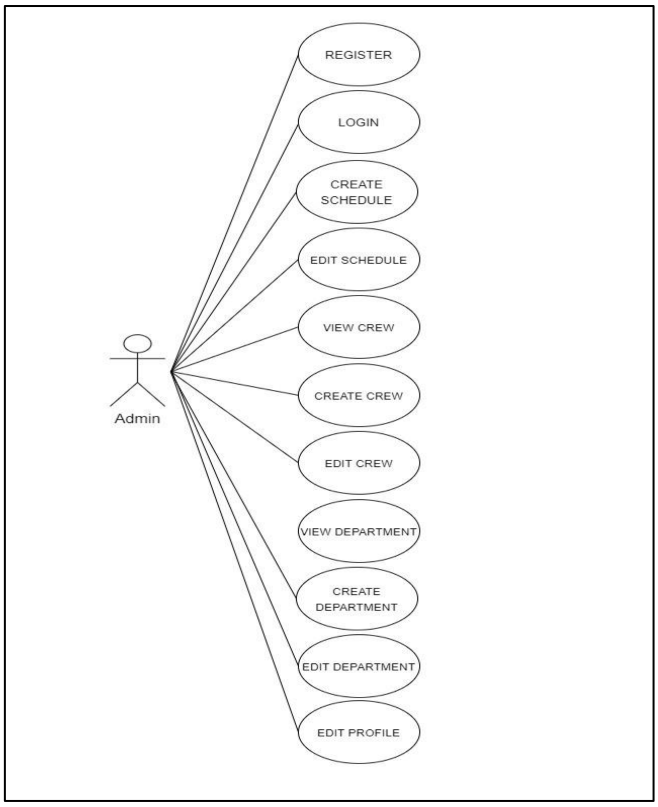


Figure 3.2: Admin Use-Case Diagram

Based on Figure 3.2 Shows the admin use case diagram. Similar to crew, and can register their account and must login to access the administrative functionalities of the system. Admin have the authority to create and edit schedule . Next is admin has access to the crew section such as view , create and edit the crew in the system. Admin also can view, create and edit the department part in the system. The admin can edit their profile in the system.

3.3.2 Sequence Diagram

A sequence diagram is a Unified Modelling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. Sequence diagram includes a collection of objects that are represented by lifelines, and the messages they exchange during the interaction (*Sequence Diagrams - IBM Documentation*, n.d.).

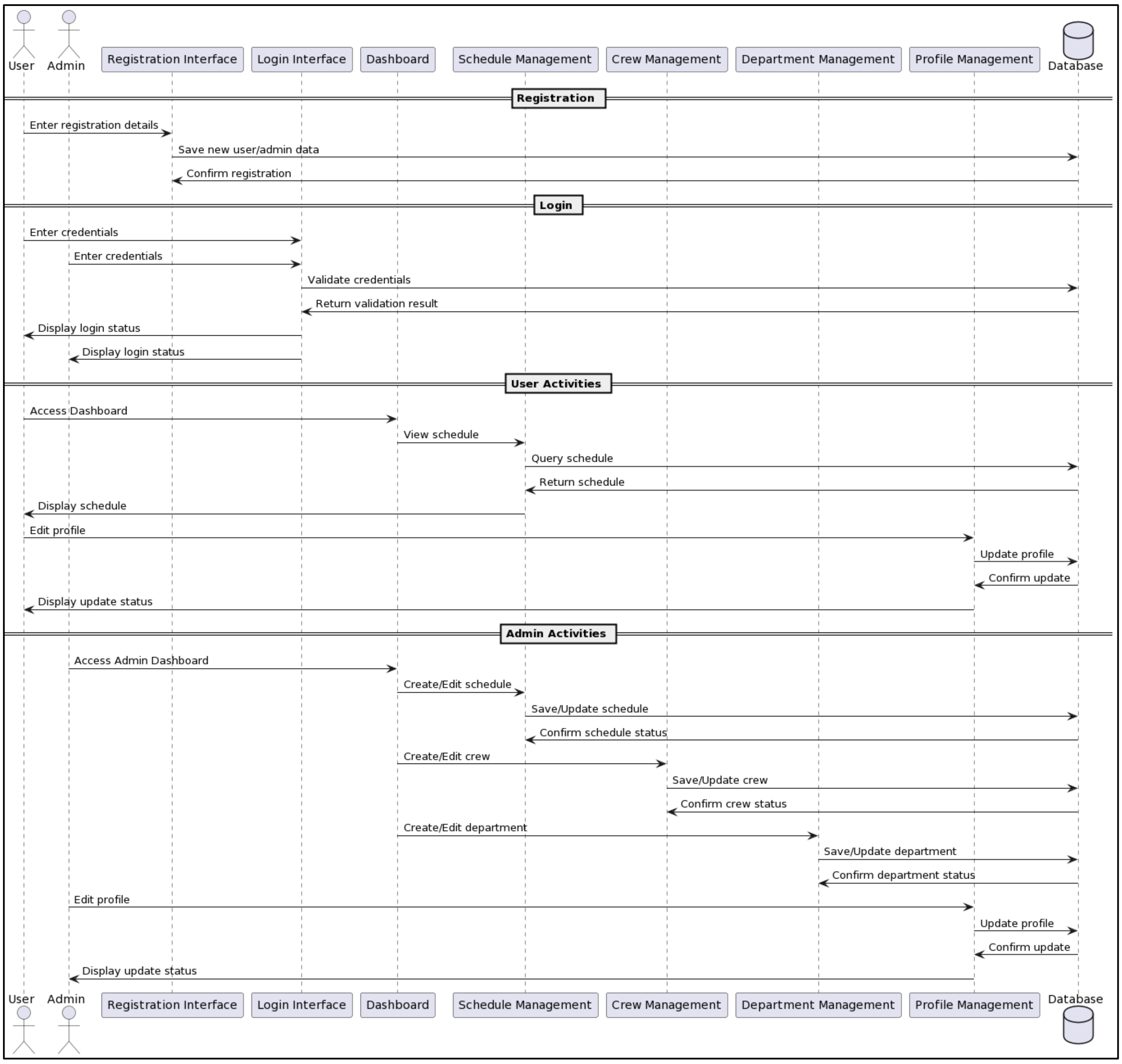


Figure 3.3: Sequence Diagram

Figure 3.3 illustrates the sequence diagram for a comprehensive system, showcasing the detailed interaction flow between users, admins, and the system's various interfaces and database. The process is divided into distinct segments: registration, login, user activities, and admin activities. In registration phase, both crew and admin input their registration details, which the system then processes and stores in the database. The database confirms the successful registration back to the registration interface, which in turn notifies the users or admins of their successful registration. In the login phase, both user and admin credentials are authenticated by the database via the login interface, granting system access upon successful verification. Once logged in, users have access to the dashboard. Users can view their schedules through the schedule management interface, which fetches the schedule data from the database and displays it to the user. Users can also edit their profiles via the profile management interface. For admin after logging in, admin can access the admin dashboard, which serves as a control center for multiple management tasks. They can create or edit schedules through the schedule management interface, with all changes being saved to the database and confirmations sent back. In the crew management section, admins can create or update crew details, which are similarly processed by the database. The same process applies to department management section, where department details can be added or modified.

3.4 Database Design

After analyzing the ideas and needs, we then created the database design and user interface (UI) design for the system. We make the design according to the analysis. The design stage is the stage of depicting the database and interface (UI) of the system.

3.4.1 Entity Relationship Diagram (ERD)

Entity Relationship Diagram is a graphic illustration of the relationship between the entities and attributes used in database of the system. A database usually will have a few entities and each entity has been assigned to a few attributes.

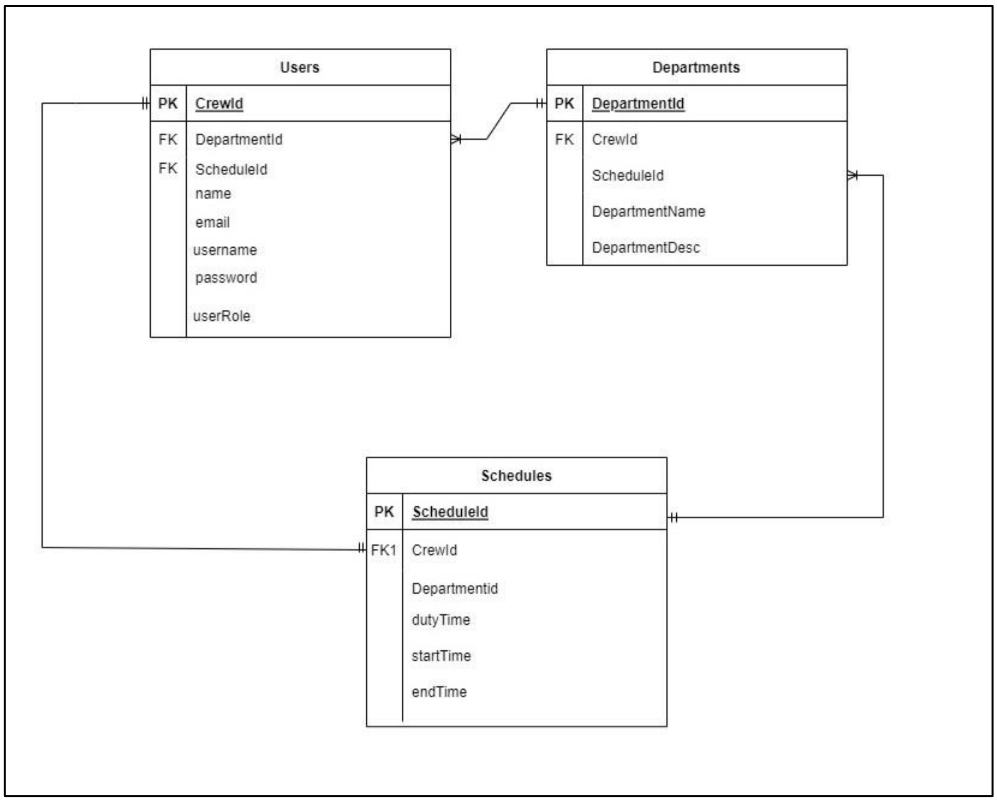


Figure 3.4: Entity relationship diagram (ERD)

Figure 3.4 Shows the entity relationship diagram of the system. Entities that are involved in the system are users, departments and schedules. Users' entity link to the department's entity by many to one relationship. Many users can be in one department. Next

the department entity links to the schedule entity by many to one. Many departments can have one schedule.

3.5 User interface (UI) design

Create an intuitive user interface design, considering user needs. Making UI Dashboard, UI Create Schedule, UI User Management and Profile. To Make a UI design, we are using Figma Software. We create elements and color palettes for the system.

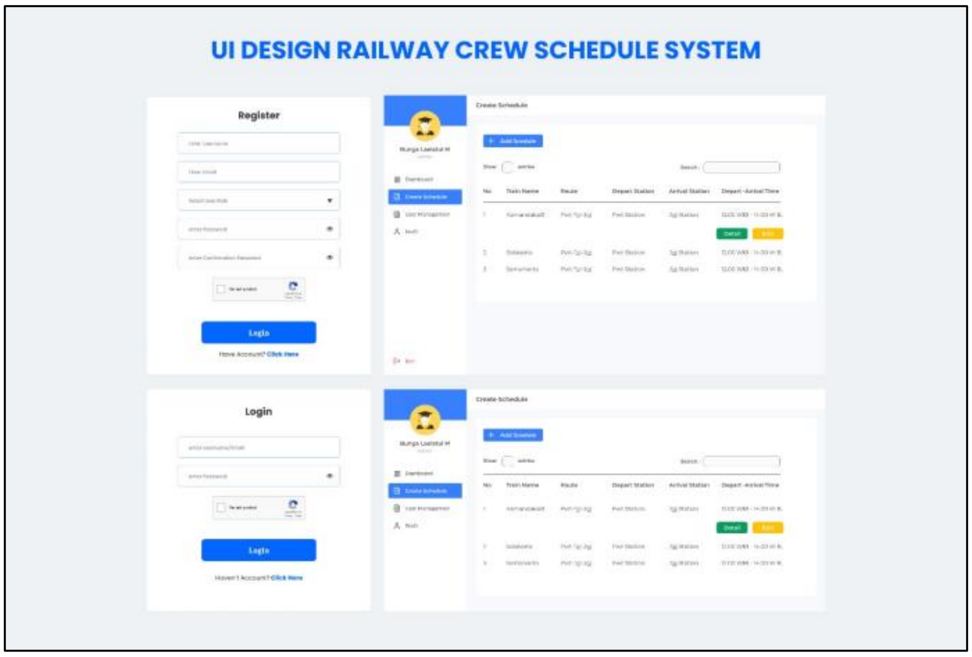


Figure 3.5: UI Design

3.6 Coding

In this coding phase, we are going to transform from concept design into a functional system based on the requirements gathered and the design specifications. We going to use various programming languages to pursue this project such as HTML language for creating the structure and layout for our web-based system, SQL language for creating and managing the database, CSS language for styling the user interface, JavaScript language for implementing client-side functionalities, and many more. Next is for User Interface (UI), we are going to implement the UI from the design phase into the actual user interface. For security features, we are going to add various methods to strengthen our system such as password encryption, google captcha, input validation, login attempts, unique and strong password combinations, and more. Lastly, for the database part, we are going to use MySQL which is an open-source RDMS that is widely used for web development.

3.7 Testing

The testing phase in the agile methodology is a continuous and iterative process that happens throughout the development of an online crew schedules system for a railway transportation project. This phase allows us to investigate several factors that have been developed for the project such as the design that needs to be user-friendly and the functionality that must archive the user requirements. In agile methodology, we can run the testing phase several times until we reach the goal of project development.

3.8 Maintenance

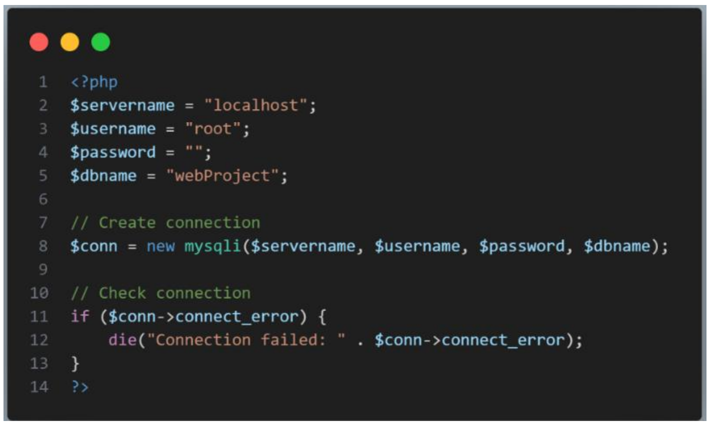
For this phase, the system will already be fully distributed to the users after the whole process of development. During this phase, we as a developer will be able to:

i. Provide ongoing support to keep the system running and ensure that it can run for a long time since this system is essential for the users and staff.
ii. Monitor the system for any errors and bugs, evaluating, and modifying the system to increase the likeliness and making it more desirable to the users.
iii. Improve the system because we will know what to update if the system has been used by the real users. Every patch of the update will include the patch notes so it will be easier for the users to adapt and understand the new features of the system.

4.0 Implementation

4.1 Home Page

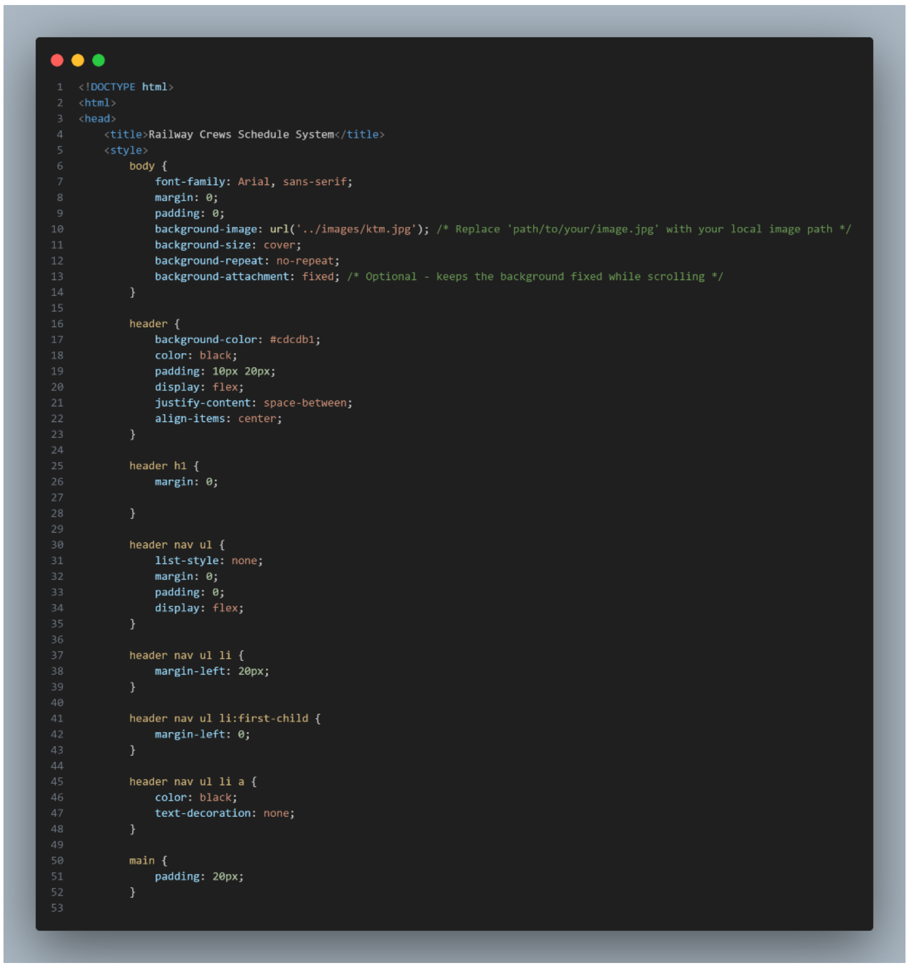
Connection.php



```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "webProject";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

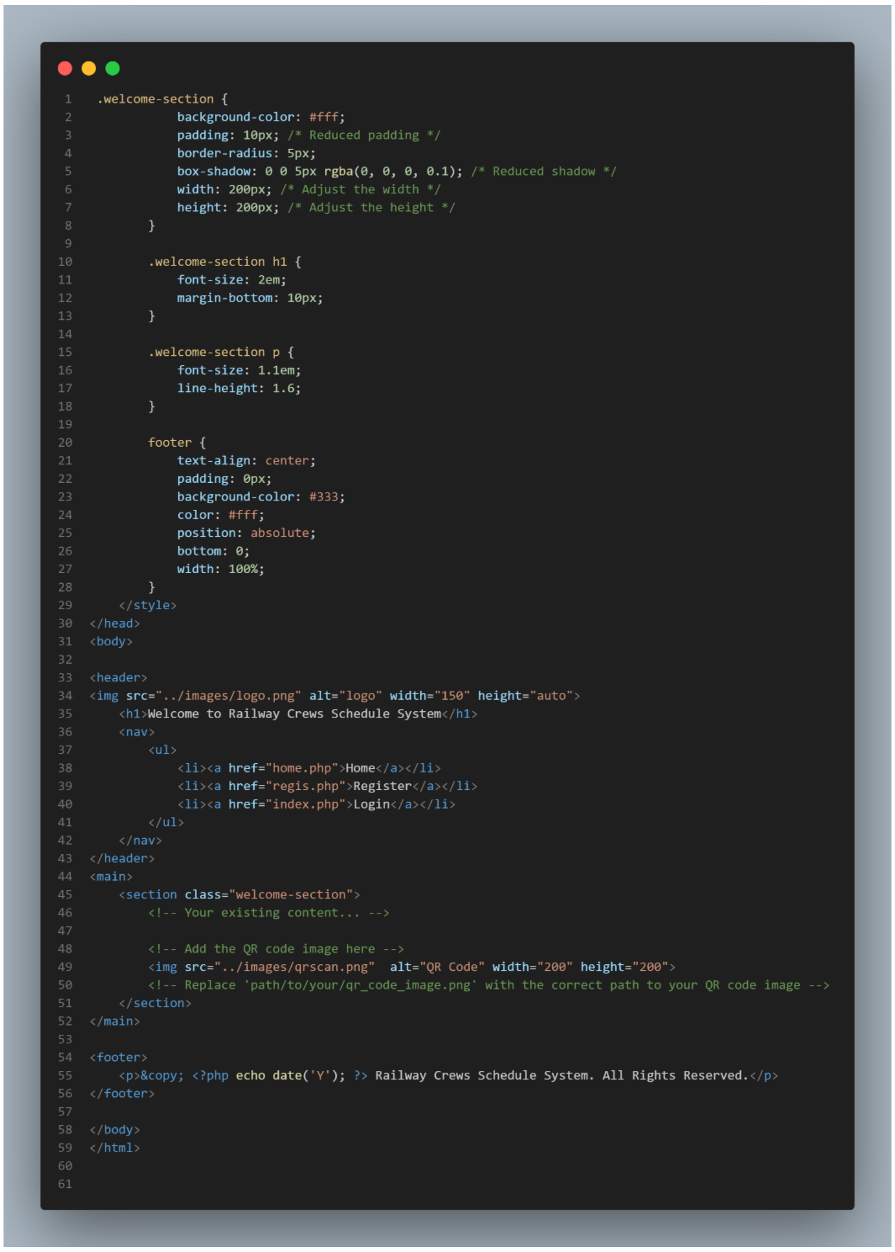
Figure 4.1: Create and Check Connection

home.php



```html
<!DOCTYPE html>
<html>
<head>
    <title>Railway Crews Schedule System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('../images/ktm.jpg'); /* Replace 'path/to/your/image.jpg' with your local image path */
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed; /* Optional - keeps the background fixed while scrolling */
        }

        header {
            background-color: #cdcdb1;
            color: black;
            padding: 10px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }

        header h1 {
            margin: 0;

        }

        header nav ul {
            list-style: none;
            margin: 0;
            padding: 0;
            display: flex;
        }

        header nav ul li {
            margin-left: 20px;
        }

        header nav ul li:first-child {
            margin-left: 0;
        }

        header nav ul li a {
            color: black;
            text-decoration: none;
        }

        main {
            padding: 20px;
        }

```

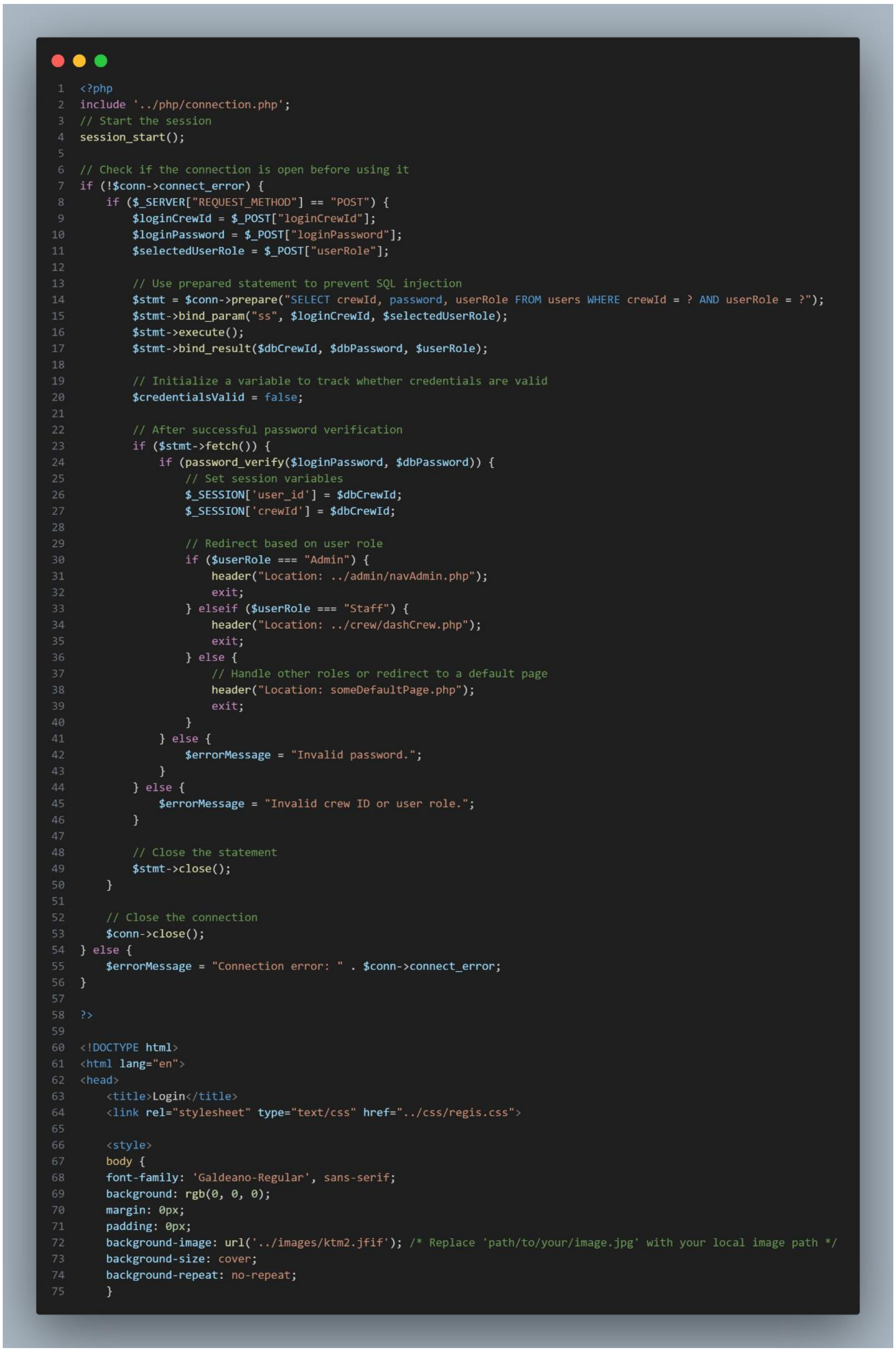
Figure 4.2: Homepage Style (1)



```
 .welcome-section {
            background-color: #fff;
            padding: 10px; /* Reduced padding */
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.1); /* Reduced shadow */
            width: 200px; /* Adjust the width */
            height: 200px; /* Adjust the height */
        }

        .welcome-section h1 {
            font-size: 2em;
            margin-bottom: 10px;
        }

        .welcome-section p {
            font-size: 1.1em;
            line-height: 1.6;
        }

        footer {
            text-align: center;
            padding: 0px;
            background-color: #333;
            color: #fff;
            position: absolute;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>

<header>
<img src="../images/logo.png" alt="logo" width="150" height="auto">
    <h1>Welcome to Railway Crews Schedule System</h1>
    <nav>
        <ul>
            <li><a href="home.php">Home</a></li>
            <li><a href="regis.php">Register</a></li>
            <li><a href="index.php">Login</a></li>
        </ul>
    </nav>
</header>
<main>
    <section class="welcome-section">
        <!-- Your existing content... -->

        <!-- Add the QR code image here -->
        <img src="../images/qrscan.png"  alt="QR Code" width="200" height="200">
        <!-- Replace 'path/to/your/qr_code_image.png' with the correct path to your QR code image -->
    </section>
</main>

<footer>
    <p>&copy; <?php echo date('Y'); ?> Railway Crews Schedule System. All Rights Reserved.</p>
</footer>

</body>
</html>
```

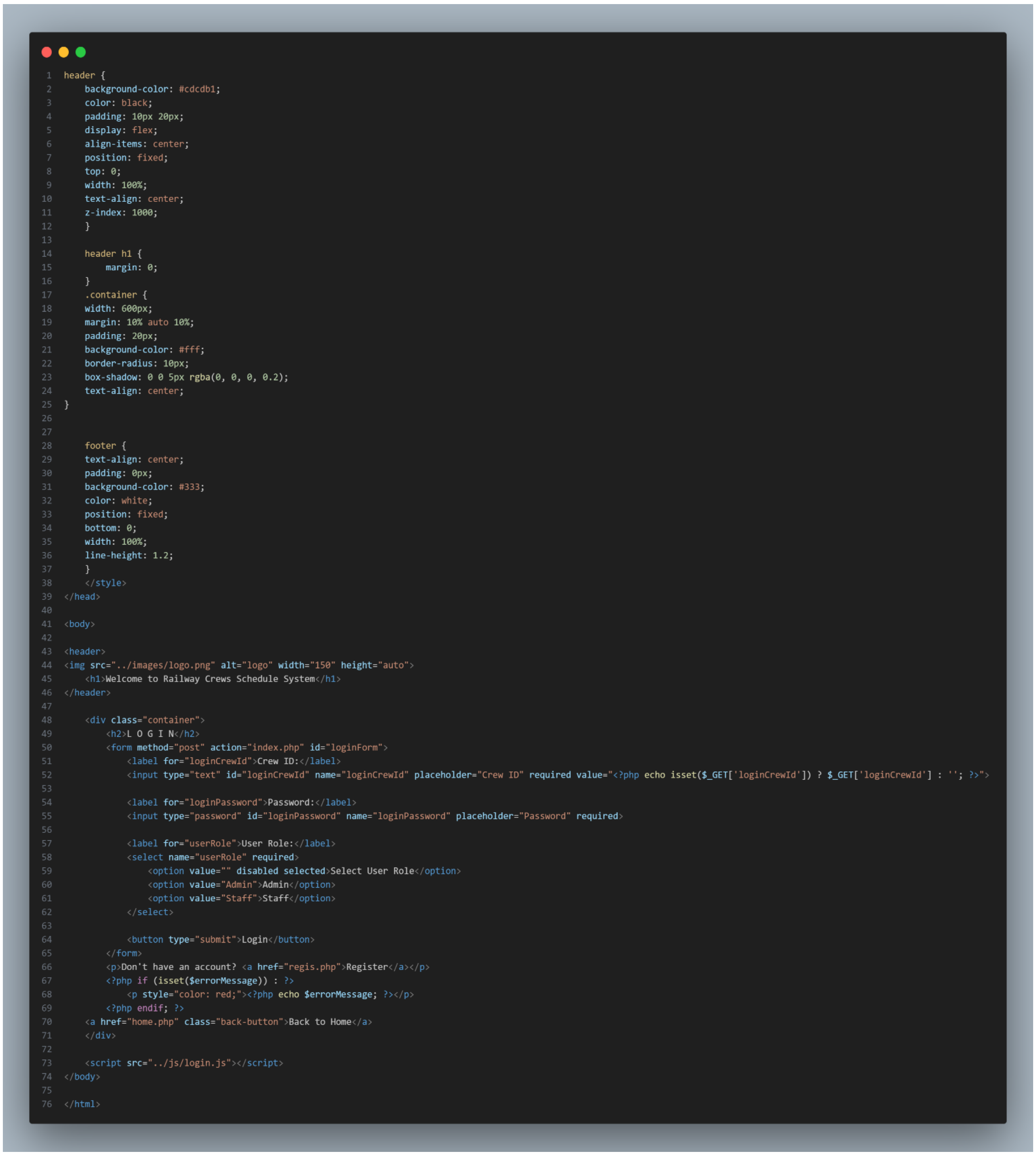
Figure 4.3: Homepage Style (2)

index.php



```php
<?php
include '../php/connection.php';
// Start the session
session_start();

// Check if the connection is open before using it
if (!$conn->connect_error) {
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $loginCrewId = $_POST["loginCrewId"];
        $loginPassword = $_POST["loginPassword"];
        $selectedUserRole = $_POST["userRole"];

        // Use prepared statement to prevent SQL injection
        $stmt = $conn->prepare("SELECT crewId, password, userRole FROM users WHERE crewId = ? AND userRole = ?");
        $stmt->bind_param("ss", $loginCrewId, $selectedUserRole);
        $stmt->execute();
        $stmt->bind_result($dbCrewId, $dbPassword, $userRole);

        // Initialize a variable to track whether credentials are valid
        $credentialsValid = false;

        // After successful password verification
        if ($stmt->fetch()) {
            if (password_verify($loginPassword, $dbPassword)) {
                // Set session variables
                $_SESSION['user_id'] = $dbCrewId;
                $_SESSION['crewId'] = $dbCrewId;

                // Redirect based on user role
                if ($userRole === "Admin") {
                    header("Location: ../admin/navAdmin.php");
                    exit;
                } elseif ($userRole === "Staff") {
                    header("Location: ../crew/dashCrew.php");
                    exit;
                } else {
                    // Handle other roles or redirect to a default page
                    header("Location: someDefaultPage.php");
                    exit;
                }
            } else {
                $errorMessage = "Invalid password.";
            }
        } else {
            $errorMessage = "Invalid crew ID or user role.";
        }

        // Close the statement
        $stmt->close();
    }

    // Close the connection
    $conn->close();
} else {
    $errorMessage = "Connection error: " . $conn->connect_error;
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login</title>
    <link rel="stylesheet" type="text/css" href="../css/regis.css">

    <style>
    body {
    font-family: 'Galdeano-Regular', sans-serif;
    background: rgb(0, 0, 0);
    margin: 0px;
    padding: 0px;
    background-image: url('../images/ktm2.jfif'); /* Replace 'path/to/your/image.jpg' with your local image path */
    background-size: cover;
    background-repeat: no-repeat;
    }
```

Figure 4.4: Login Page (1)



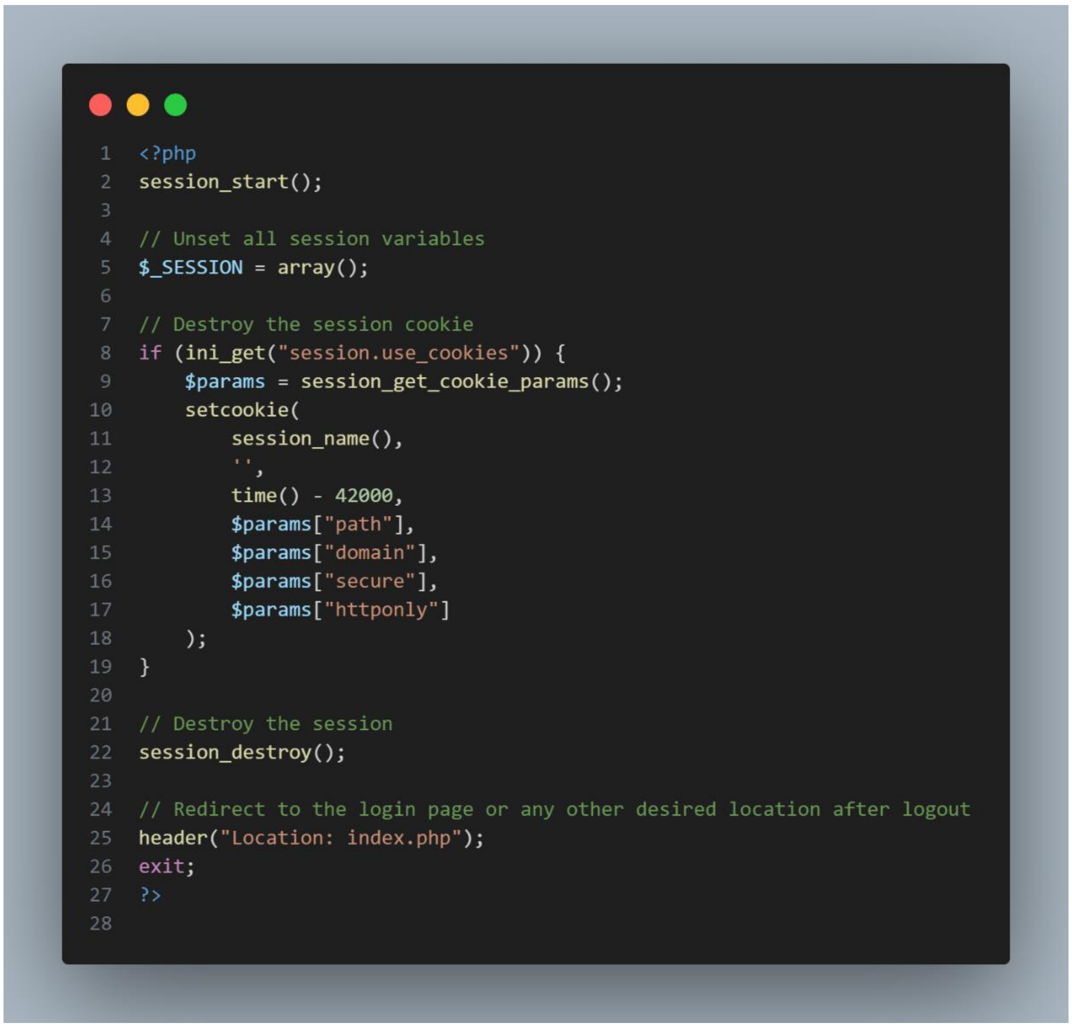
```
header {
    background-color: #cdcdb1;
    color: black;
    padding: 10px 20px;
    display: flex;
    align-items: center;
    position: fixed;
    top: 0;
    width: 100%;
    text-align: center;
    z-index: 1000;
    }

    header h1 {
        margin: 0;
    }
    .container {
    width: 600px;
    margin: 10% auto 10%;
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
    text-align: center;
}


    footer {
    text-align: center;
    padding: 0px;
    background-color: #333;
    color: white;
    position: fixed;
    bottom: 0;
    width: 100%;
    line-height: 1.2;
    }
    </style>
</head>

<body>

<header>
<img src="../images/logo.png" alt="logo" width="150" height="auto">
    <h1>Welcome to Railway Crews Schedule System</h1>
</header>

    <div class="container">
        <h2>L O G I N</h2>
        <form method="post" action="index.php" id="loginForm">
            <label for="loginCrewId">Crew ID:</label>
            <input type="text" id="loginCrewId" name="loginCrewId" placeholder="Crew ID" required value="<?php echo isset($_GET['loginCrewId']) ? $_GET['loginCrewId'] : ''; ?>">

            <label for="loginPassword">Password:</label>
            <input type="password" id="loginPassword" name="loginPassword" placeholder="Password" required>

            <label for="userRole">User Role:</label>
            <select name="userRole" required>
                <option value="" disabled selected>Select User Role</option>
                <option value="Admin">Admin</option>
                <option value="Staff">Staff</option>
            </select>

            <button type="submit">Login</button>
        </form>
        <p>Don't have an account? <a href="regis.php">Register</a></p>
        <?php if (isset($errorMessage)) : ?>
            <p style="color: red;"><?php echo $errorMessage; ?></p>
        <?php endif; ?>
    <a href="home.php" class="back-button">Back to Home</a>
    </div>

    <script src="../js/login.js"></script>
</body>

</html>
```

Figure 4.5: Login Page (2)

The structure of index.php code consists of three parts which are PHP, HTML, and CSS code. Each part has distinct functions that support the Railway Crews Scheduling System. The PHP code in this file is involved in including the connection.php file that will check the connection into the database, initialize a variable to track whether credentials are valid, and handle redirects based on user role. The HTML codes are to create a whole login page interface that requires the user to enter the crewId, password and select the user role. The CSS codes that are used in this file are three types which are inline, internal, and external.

The inline CSS type is used to make a change in the paragraph. The Internal and external CSS type is used to change the body, header, footer, and others in this file.

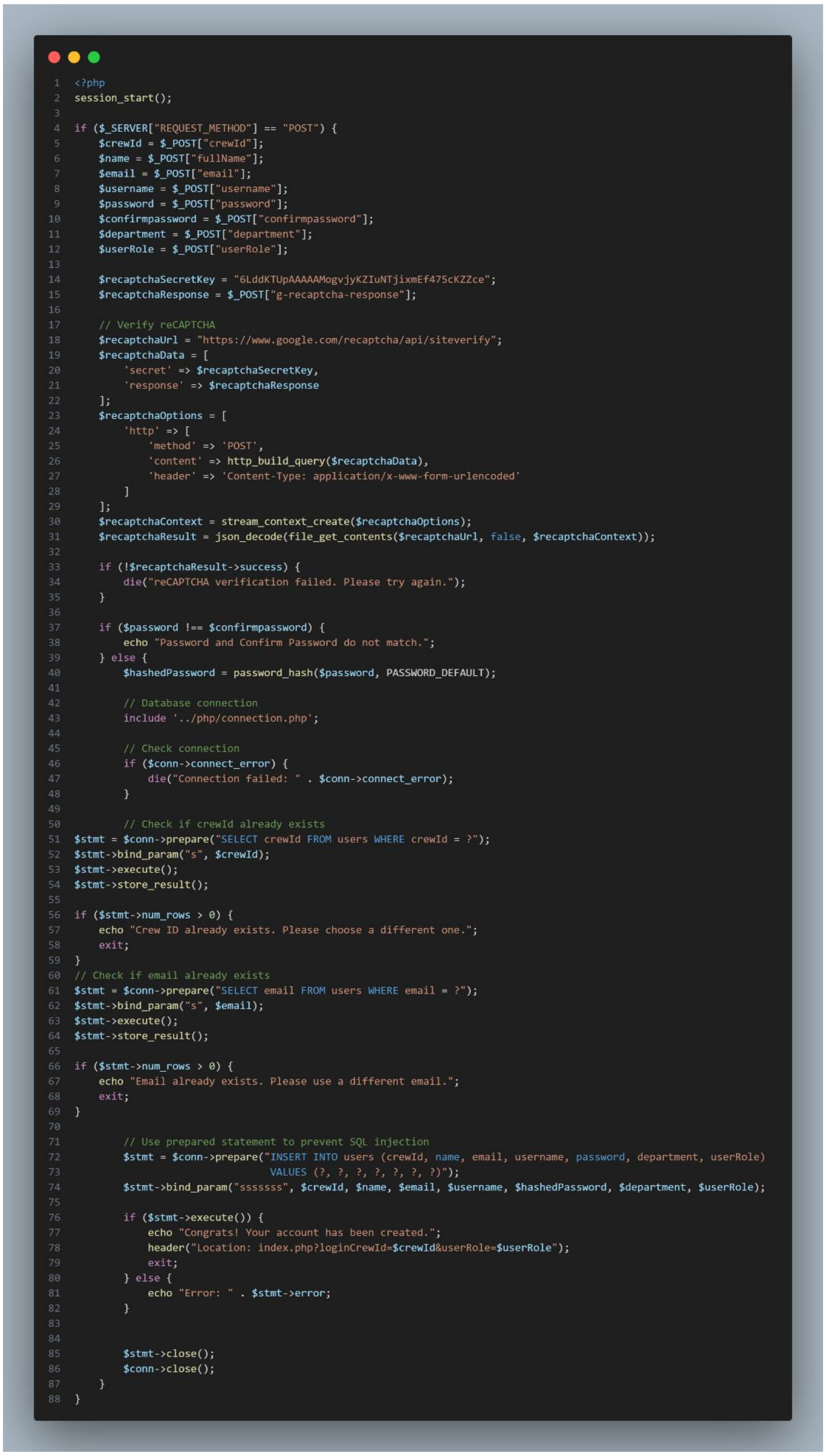
logout.php



```php
<?php
session_start();

// Unset all session variables
$_SESSION = array();

// Destroy the session cookie
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(
        session_name(),
        '',
        time() - 42000,
        $params["path"],
        $params["domain"],
        $params["secure"],
        $params["httponly"]
    );
}

// Destroy the session
session_destroy();

// Redirect to the login page or any other desired location after logout
header("Location: index.php");
exit;
?>

```

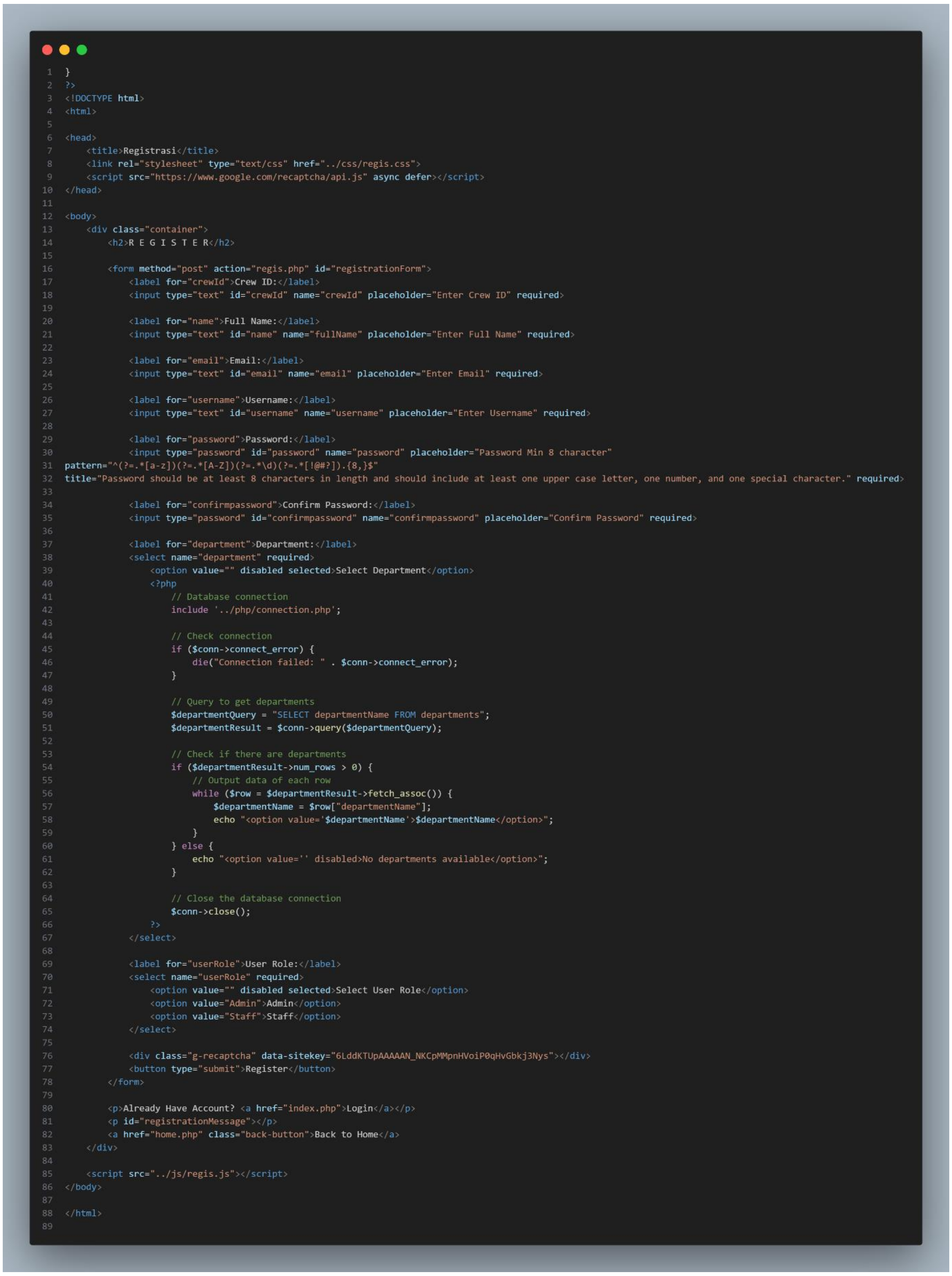
Figure 4.6: Log Out

regis.php



```php
<?php
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $crewId = $_POST["crewId"];
    $name = $_POST["fullName"];
    $email = $_POST["email"];
    $username = $_POST["username"];
    $password = $_POST["password"];
    $confirmpassword = $_POST["confirmpassword"];
    $department = $_POST["department"];
    $userRole = $_POST["userRole"];

    $recaptchaSecretKey = "6LddKTUpAAAAAMogvjyKZIuNTjixmEf475cKZZce";
    $recaptchaResponse = $_POST["g-recaptcha-response"];

    // Verify reCAPTCHA
    $recaptchaUrl = "https://www.google.com/recaptcha/api/siteverify";
    $recaptchaData = [
        'secret' => $recaptchaSecretKey,
        'response' => $recaptchaResponse
    ];
    $recaptchaOptions = [
        'http' => [
            'method' => 'POST',
            'content' => http_build_query($recaptchaData),
            'header' => 'Content-Type: application/x-www-form-urlencoded'
        ]
    ];
    $recaptchaContext = stream_context_create($recaptchaOptions);
    $recaptchaResult = json_decode(file_get_contents($recaptchaUrl, false, $recaptchaContext));

    if (!$recaptchaResult->success) {
        die("reCAPTCHA verification failed. Please try again.");
    }

    if ($password !== $confirmpassword) {
        echo "Password and Confirm Password do not match.";
    } else {
        $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

        // Database connection
        include '../php/connection.php';

        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        // Check if crewId already exists
$stmt = $conn->prepare("SELECT crewId FROM users WHERE crewId = ?");
$stmt->bind_param("s", $crewId);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    echo "Crew ID already exists. Please choose a different one.";
    exit;
}
// Check if email already exists
$stmt = $conn->prepare("SELECT email FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows > 0) {
    echo "Email already exists. Please use a different email.";
    exit;
}

        // Use prepared statement to prevent SQL injection
        $stmt = $conn->prepare("INSERT INTO users (crewId, name, email, username, password, department, userRole)
                                VALUES (?, ?, ?, ?, ?, ?, ?)");
        $stmt->bind_param("sssssss", $crewId, $name, $email, $username, $hashedPassword, $department, $userRole);

        if ($stmt->execute()) {
            echo "Congrats! Your account has been created.";
            header("Location: index.php?loginCrewId=$crewId&userRole=$userRole");
            exit;
        } else {
            echo "Error: " . $stmt->error;
        }


        $stmt->close();
        $conn->close();
    }
}
```

Figure 4.7: Registration(1)



```
}
?>
<!DOCTYPE html>
<html>

<head>
    <title>Registrasi</title>
    <link rel="stylesheet" type="text/css" href="../css/regis.css">
    <script src="https://www.google.com/recaptcha/api.js" async defer></script>
</head>

<body>
    <div class="container">
        <h2>R E G I S T E R</h2>

        <form method="post" action="regis.php" id="registrationForm">
            <label for="crewId">Crew ID:</label>
            <input type="text" id="crewId" name="crewId" placeholder="Enter Crew ID" required>

            <label for="name">Full Name:</label>
            <input type="text" id="name" name="fullName" placeholder="Enter Full Name" required>

            <label for="email">Email:</label>
            <input type="text" id="email" name="email" placeholder="Enter Email" required>

            <label for="username">Username:</label>
            <input type="text" id="username" name="username" placeholder="Enter Username" required>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password" placeholder="Password Min 8 character"
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#?]).{8,}$"
title="Password should be at least 8 characters in length and should include at least one upper case letter, one number, and one special character." required>

            <label for="confirmpassword">Confirm Password:</label>
            <input type="password" id="confirmpassword" name="confirmpassword" placeholder="Confirm Password" required>

            <label for="department">Department:</label>
            <select name="department" required>
                <option value="" disabled selected>Select Department</option>
                <?php
                    // Database connection
                    include '../php/connection.php';

                    // Check connection
                    if ($conn->connect_error) {
                        die("Connection failed: " . $conn->connect_error);
                    }

                    // Query to get departments
                    $departmentQuery = "SELECT departmentName FROM departments";
                    $departmentResult = $conn->query($departmentQuery);

                    // Check if there are departments
                    if ($departmentResult->num_rows > 0) {
                        // Output data of each row
                        while ($row = $departmentResult->fetch_assoc()) {
                            $departmentName = $row["departmentName"];
                            echo "<option value='$departmentName'>$departmentName</option>";
                        }
                    } else {
                        echo "<option value='' disabled>No departments available</option>";
                    }

                    // Close the database connection
                    $conn->close();
                ?>
            </select>

            <label for="userRole">User Role:</label>
            <select name="userRole" required>
                <option value="" disabled selected>Select User Role</option>
                <option value="Admin">Admin</option>
                <option value="Staff">Staff</option>
            </select>

            <div class="g-recaptcha" data-sitekey="6LddKTUpAAAAAN_NKCpMMpnHVoiP0qHvGbkj3Nys"></div>
            <button type="submit">Register</button>
        </form>

        <p>Already Have Account? <a href="index.php">Login</a></p>
        <p id="registrationMessage"></p>
        <a href="home.php" class="back-button">Back to Home</a>
    </div>

    <script src="../js/regis.js"></script>
</body>

</html>

```

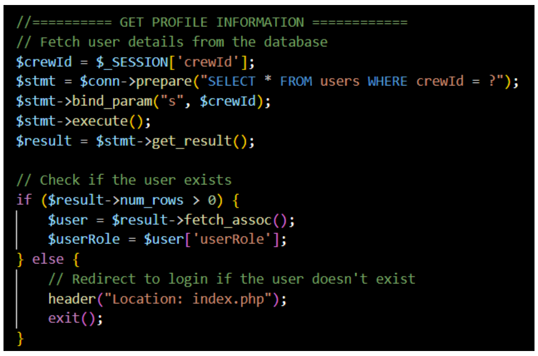
Figure 4.8: Registration (2)

For the regis.php code is divided into three sections: PHP, HTML, and CSS code. Each component has a different purpose in assisting the Railway Crews Scheduling System. This file's PHP code is involved in the connection.php file, which will verify the database connection and initialize the registration process. The HTML codes will be used to construct a whole registration page interface that will ask the user to input the crewId, full name, email, username, password, confirm password, department and user role to make a new account.

The code also provided a security measures like password validation that requires user to put a strong password for their password with condition of minimum 8 length of password, must include one uppercase, one number and one special character in the password. In the code also put the email validation that make sure the user did not use the same email to make new account to make sure the account is secure and other user cannot use other people email to make an account. Captcha is also included in the code to make sure the user is registering the account and not a bot. User also cannot use the same crewID to make an account to avoid a duplicate crewID for each user.

4.2 Admin Side

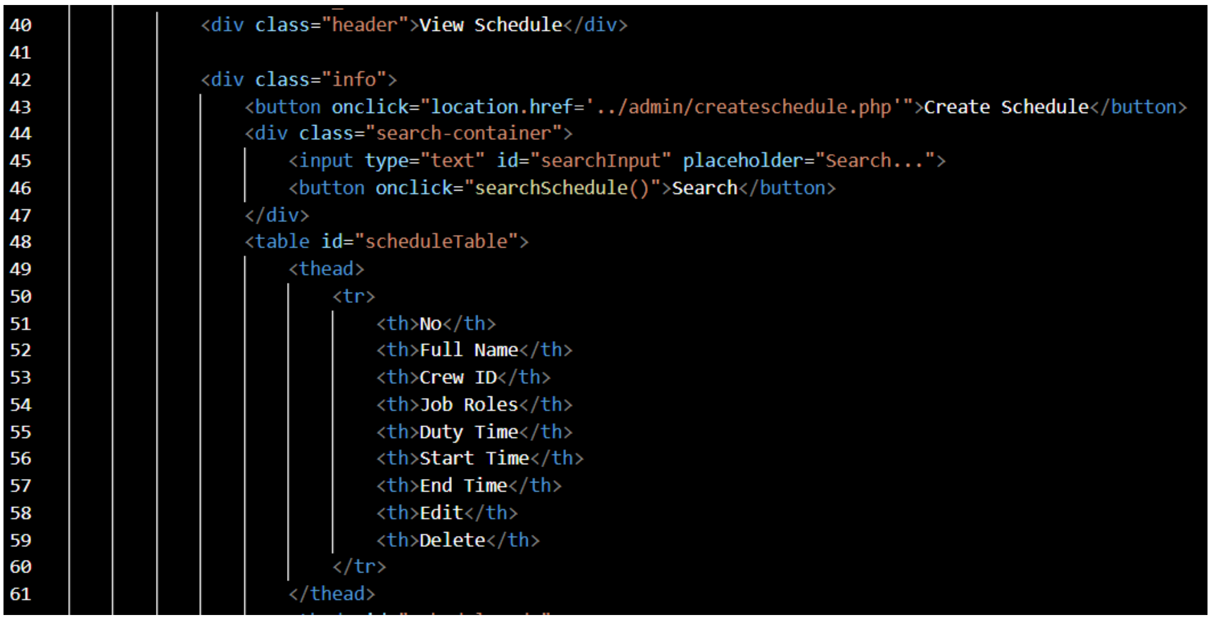
viewSchedule.php



```
//========== GET PROFILE INFORMATION ============
// Fetch user details from the database
$crewId = $_SESSION['crewId'];
$stmt = $conn->prepare("SELECT * FROM users WHERE crewId = ?");
$stmt->bind_param("s", $crewId);
$stmt->execute();
$result = $stmt->get_result();

// Check if the user exists
if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();
    $userRole = $user['userRole'];
} else {
    // Redirect to login if the user doesn't exist
    header("Location: index.php");
    exit();
}
```

Figure 4.9: Get Profile Information

Figure 4.9 is a segment of code about initializing a session for the user. It proceeds to fetch user details from the database. User identifier, crewId, is retrieved from the session. An SQL SELECT query is prepared to fetch all columns for a user with a specific crewId from the "users" table. The query is then executed, and the result is obtained. After checking if the user exists based on the returned rows, the user's role is assigned to the variable userRole. If the user does not exist, the script redirects the user to the "index.php" page.

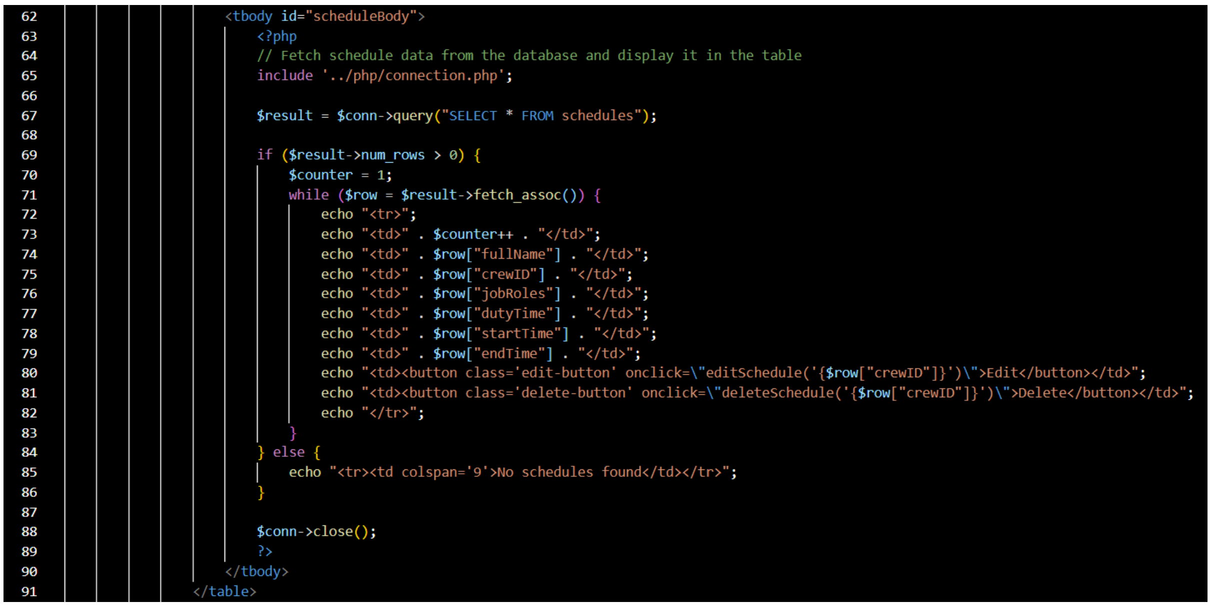


```
                <div class="header">View Schedule</div>

                <div class="info">
                    <button onclick="location.href='../admin/createschedule.php'">Create Schedule</button>
                    <div class="search-container">
                        <input type="text" id="searchInput" placeholder="Search...">
                        <button onclick="searchSchedule()">Search</button>
                    </div>
                    <table id="scheduleTable">
                        <thead>
                            <tr>
                                <th>No</th>
                                <th>Full Name</th>
                                <th>Crew ID</th>
                                <th>Job Roles</th>
                                <th>Duty Time</th>
                                <th>Start Time</th>
                                <th>End Time</th>
                                <th>Edit</th>
                                <th>Delete</th>
                            </tr>
                        </thead>
```

Figure 4.10: Schedule Table HTML

The schedule table is displayed using an HTML table structure with the id "scheduleTable". The table contains headers for different columns such as "No", "Full Name", "Crew ID", "Job Roles", "Duty Time", "Start Time", "End Time", "Edit", and

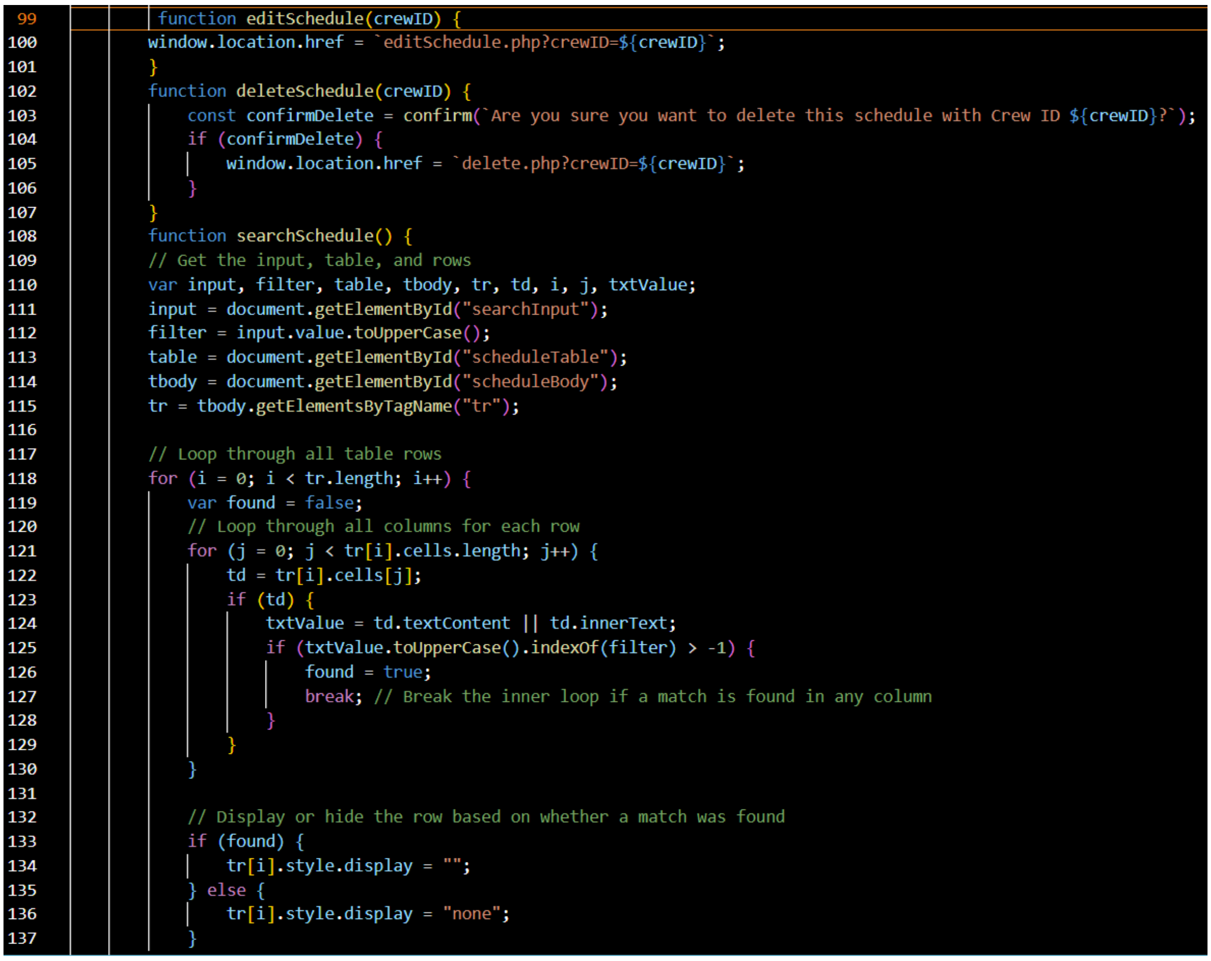
"Delete". There is a "Create Schedule" button that, when clicked, redirects the user to a page named "createschedule.php".



```php
            <tbody id="scheduleBody">
                <?php
                // Fetch schedule data from the database and display it in the table
                include '../php/connection.php';

                $result = $conn->query("SELECT * FROM schedules");

                if ($result->num_rows > 0) {
                    $counter = 1;
                    while ($row = $result->fetch_assoc()) {
                        echo "<tr>";
                        echo "<td>" . $counter++ . "</td>";
                        echo "<td>" . $row["fullName"] . "</td>";
                        echo "<td>" . $row["crewID"] . "</td>";
                        echo "<td>" . $row["jobRoles"] . "</td>";
                        echo "<td>" . $row["dutyTime"] . "</td>";
                        echo "<td>" . $row["startTime"] . "</td>";
                        echo "<td>" . $row["endTime"] . "</td>";
                        echo "<td><button class='edit-button' onclick=\"editSchedule('{$row["crewID"]}')\">Edit</button></td>";
                        echo "<td><button class='delete-button' onclick=\"deleteSchedule('{$row["crewID"]}')\">Delete</button></td>";
                        echo "</tr>";
                    }
                } else {
                    echo "<tr><td colspan='9'>No schedules found</td></tr>";
                }

                $conn->close();
                ?>
            </tbody>
        </table>
```

Figure 4.11: Schedule Table PHP

Figure 4.11 shows the segment of coding for when the schedule data is fetched from a database using PHP and displayed in the table body within the "scheduleBody" element.



```javascript
    function editSchedule(crewID) {
    window.location.href = `editSchedule.php?crewID=${crewID}`;
    }
    function deleteSchedule(crewID) {
        const confirmDelete = confirm(`Are you sure you want to delete this schedule with Crew ID ${crewID}?`);
        if (confirmDelete) {
            window.location.href = `delete.php?crewID=${crewID}`;
        }
    }
    function searchSchedule() {
    // Get the input, table, and rows
    var input, filter, table, tbody, tr, td, i, j, txtValue;
    input = document.getElementById("searchInput");
    filter = input.value.toUpperCase();
    table = document.getElementById("scheduleTable");
    tbody = document.getElementById("scheduleBody");
    tr = tbody.getElementsByTagName("tr");

    // Loop through all table rows
    for (i = 0; i < tr.length; i++) {
        var found = false;
        // Loop through all columns for each row
        for (j = 0; j < tr[i].cells.length; j++) {
            td = tr[i].cells[j];
            if (td) {
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1) {
                    found = true;
                    break; // Break the inner loop if a match is found in any column
                }
            }
        }

        // Display or hide the row based on whether a match was found
        if (found) {
            tr[i].style.display = "";
        } else {
            tr[i].style.display = "none";
        }
```

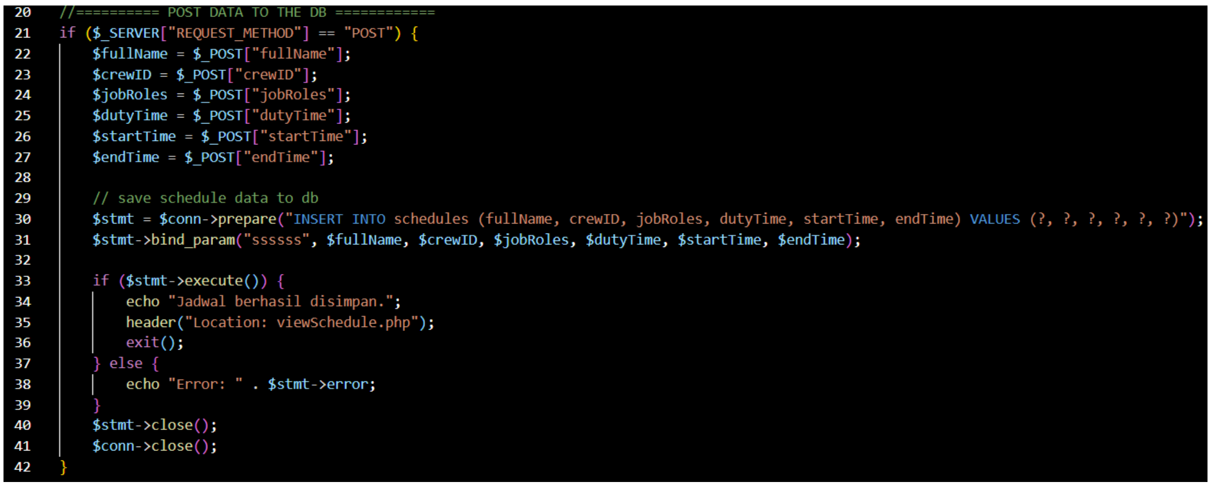
Figure 4.12: Edit,Delete,Search Schedule

Based on Figure 4.12, function editSchedule(crewID) is used for editing a schedule entry. It takes the crewID as a parameter to identify the specific schedule entry to be edited. When called, it redirects the user to the "editSchedule.php" page.

Function deleteSchedule(crewID) is responsible for deleting a schedule entry. It takes the crewID as a parameter to identify the specific schedule entry to be deleted. Upon calling this function, it first prompts the user with a confirmation message using the confirm method. If the user confirms the deletion, it redirects to the "delete.php" page with the crewID.

Function searchSchedule() is designed to handle the search functionality for the schedule table. It retrieves the search input value, the table, and its rows, and then performs a case-insensitive search through the table rows. It loops through all the table rows and within each row, it loops through the columns to check for a match with the search input. If a match is found in any column of a row, it displays the row; otherwise, it hides the row from the display.

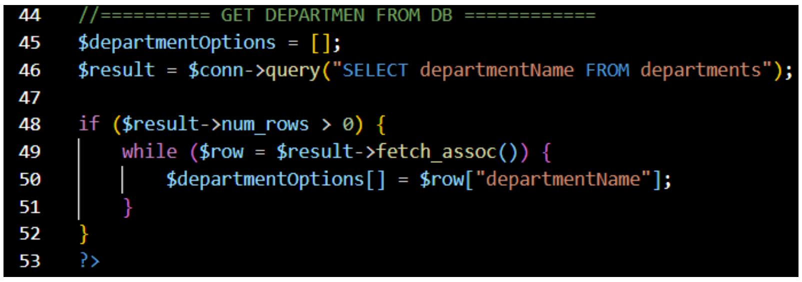
createSchedule.php



```php
//========== POST DATA TO THE DB ============
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $fullName = $_POST["fullName"];
    $crewID = $_POST["crewID"];
    $jobRoles = $_POST["jobRoles"];
    $dutyTime = $_POST["dutyTime"];
    $startTime = $_POST["startTime"];
    $endTime = $_POST["endTime"];

    // save schedule data to db
    $stmt = $conn->prepare("INSERT INTO schedules (fullName, crewID, jobRoles, dutyTime, startTime, endTime) VALUES (?, ?, ?, ?, ?, ?)");
    $stmt->bind_param("ssssss", $fullName, $crewID, $jobRoles, $dutyTime, $startTime, $endTime);

    if ($stmt->execute()) {
        echo "Jadwal berhasil disimpan.";
        header("Location: viewSchedule.php");
        exit();
    } else {
        echo "Error: " . $stmt->error;
    }
    $stmt->close();
    $conn->close();
}
```

Figure 4.13: Post Data to Database

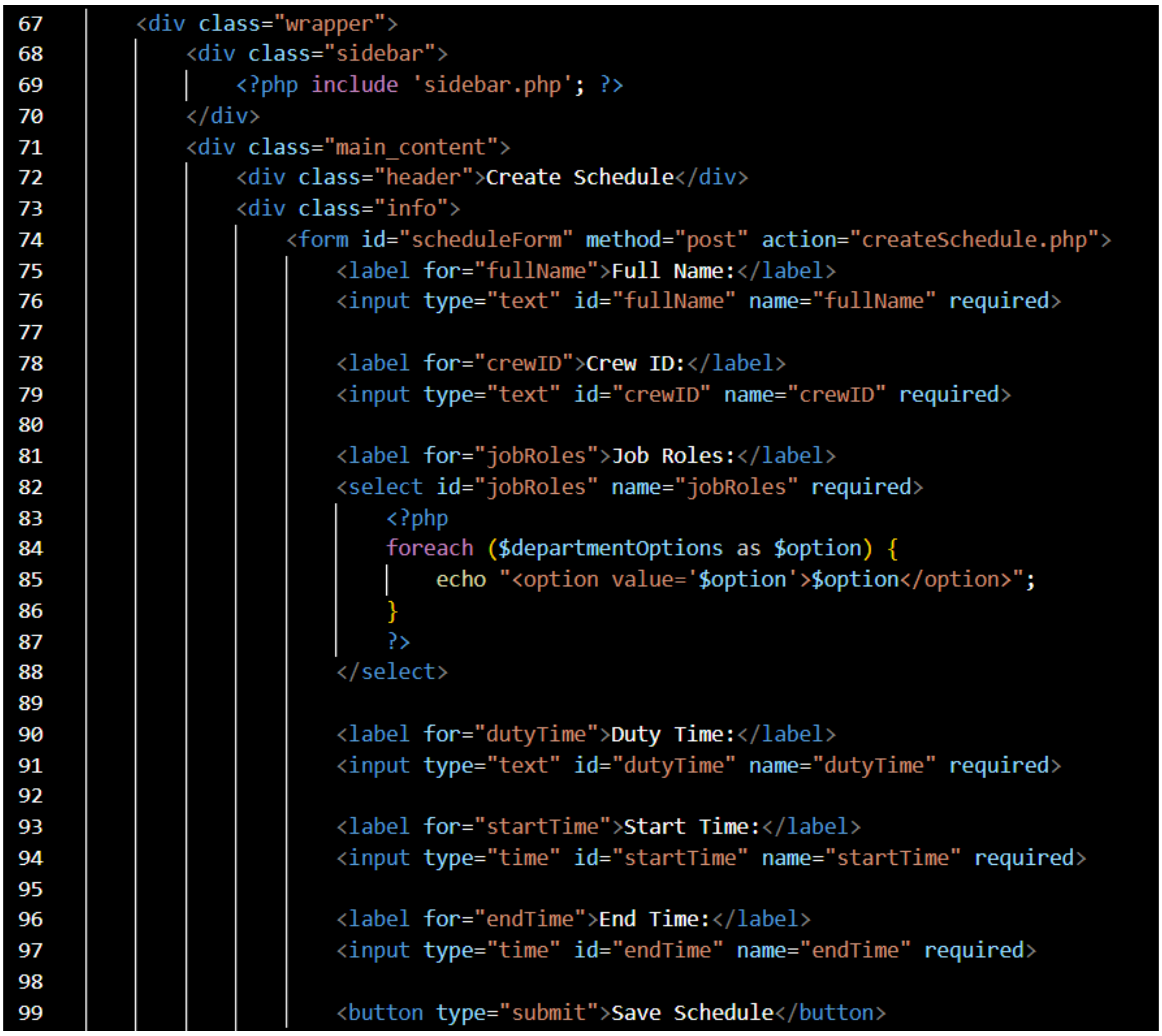
This section checks if the HTTP request method is "POST." If it is, it retrieves data from the $_POST superglobal array, including fullName, crewID, jobRoles, dutyTime, startTime, and endTime. It then prepares an SQL INSERT statement to insert this data into a table named "schedules" in the database. The bind_param method is used to bind the parameters to the SQL query to prevent SQL injection. If the insertion is successful ($stmt->execute()), a success message "Jadwal berhasil disimpan." is displayed, and the page is redirected to "viewSchedule.php." Otherwise, an error message is displayed.



```php
//========== GET DEPARTMEN FROM DB ============
$departmentOptions = [];
$result = $conn->query("SELECT departmentName FROM departments");

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $departmentOptions[] = $row["departmentName"];
    }
}
?>
```

Figure 4.14: Get Department from Database

This section retrieves department names from the database and stores them in the $departmentOptions array. It fetches the department names from a table named "departments" and populates the $departmentOptions array with the retrieved data.

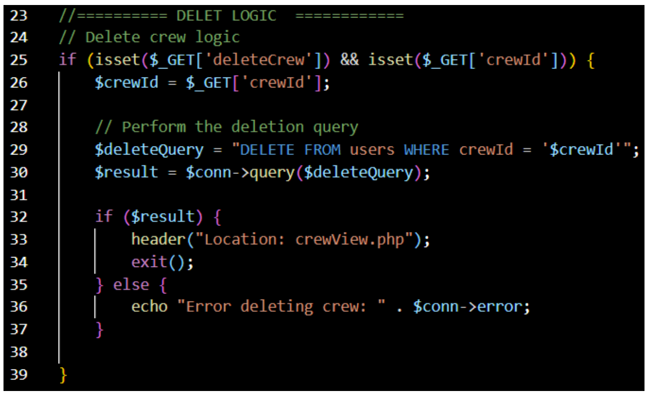


```html
    <div class="wrapper">
        <div class="sidebar">
            <?php include 'sidebar.php'; ?>
        </div>
        <div class="main_content">
            <div class="header">Create Schedule</div>
            <div class="info">
                <form id="scheduleForm" method="post" action="createSchedule.php">
                    <label for="fullName">Full Name:</label>
                    <input type="text" id="fullName" name="fullName" required>

                    <label for="crewID">Crew ID:</label>
                    <input type="text" id="crewID" name="crewID" required>

                    <label for="jobRoles">Job Roles:</label>
                    <select id="jobRoles" name="jobRoles" required>
                        <?php
                        foreach ($departmentOptions as $option) {
                            echo "<option value='$option'>$option</option>";
                        }
                        ?>
                    </select>

                    <label for="dutyTime">Duty Time:</label>
                    <input type="text" id="dutyTime" name="dutyTime" required>

                    <label for="startTime">Start Time:</label>
                    <input type="time" id="startTime" name="startTime" required>

                    <label for="endTime">End Time:</label>
                    <input type="time" id="endTime" name="endTime" required>

                    <button type="submit">Save Schedule</button>
```

Figure 4.15: Create Schedule Form HTML

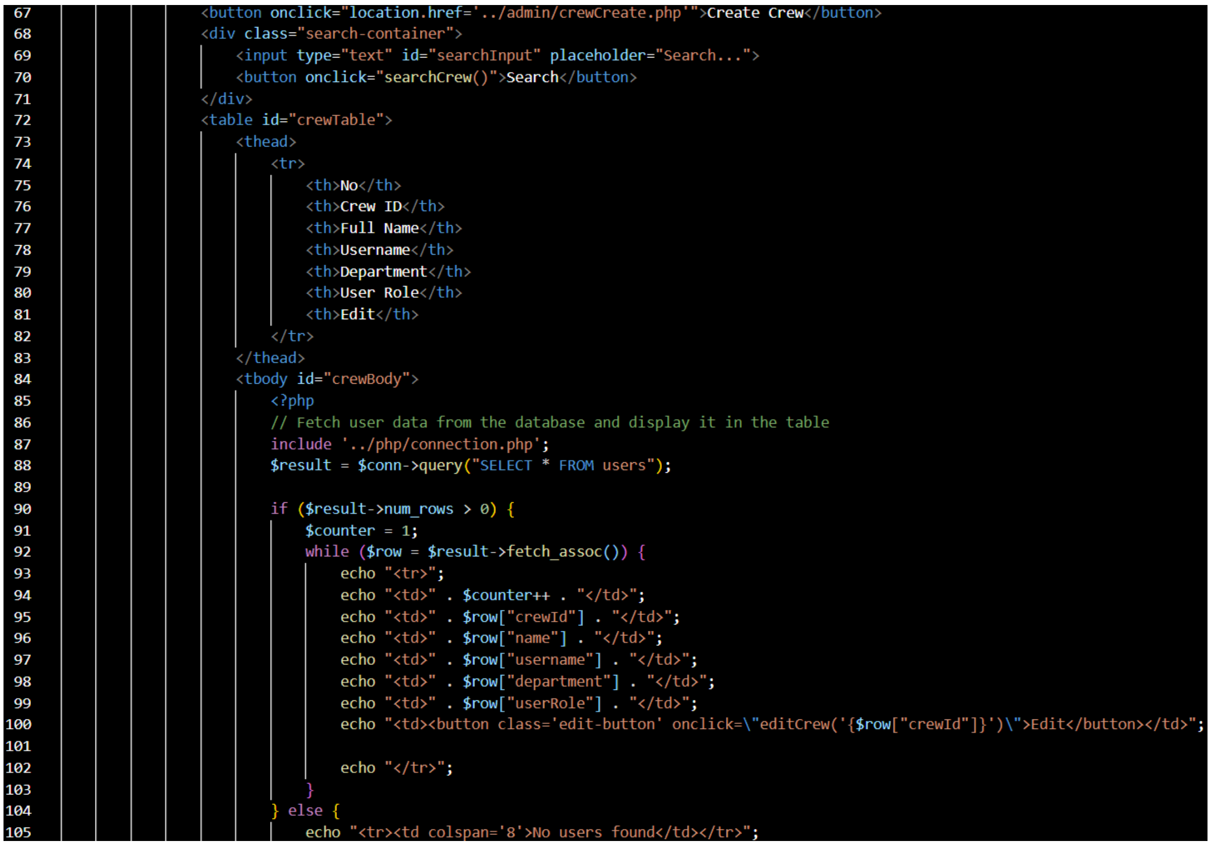
The HTML form in Figure 4.15 is used to capture input for creating a new schedule entry. It includes input fields for "Full Name," "Crew ID," "Job Roles" (select dropdown populated from database), "Duty Time," "Start Time," and "End Time."

crewView.php



```php
//========== DELET LOGIC  ============
// Delete crew logic
if (isset($_GET['deleteCrew']) && isset($_GET['crewId'])) {
    $crewId = $_GET['crewId'];

    // Perform the deletion query
    $deleteQuery = "DELETE FROM users WHERE crewId = '$crewId'";
    $result = $conn->query($deleteQuery);

    if ($result) {
        header("Location: crewView.php");
        exit();
    } else {
        echo "Error deleting crew: " . $conn->error;
    }

}
```

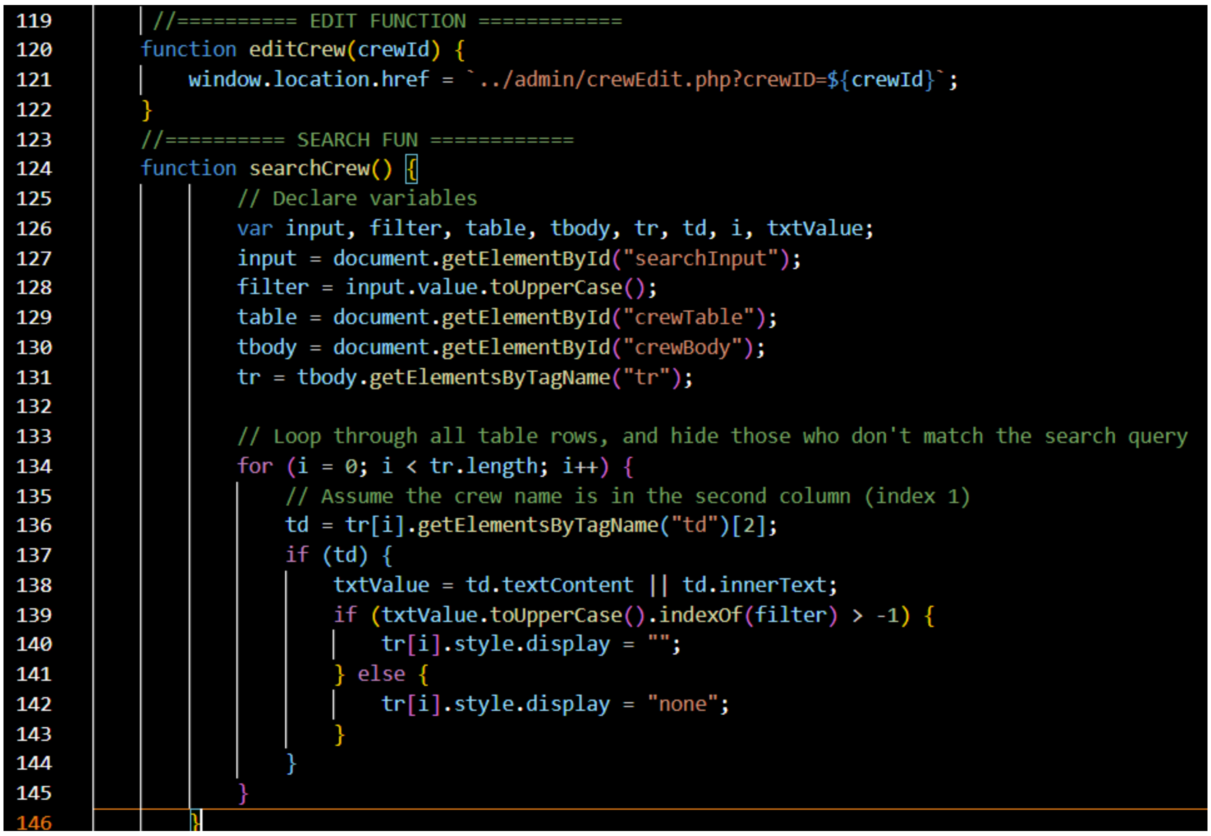
Figure 4.16: Delete Crew PHP



```php
<button onclick="location.href='../admin/crewCreate.php'">Create Crew</button>
<div class="search-container">
    <input type="text" id="searchInput" placeholder="Search...">
    <button onclick="searchCrew()">Search</button>
</div>
<table id="crewTable">
    <thead>
        <tr>
            <th>No</th>
            <th>Crew ID</th>
            <th>Full Name</th>
            <th>Username</th>
            <th>Department</th>
            <th>User Role</th>
            <th>Edit</th>
        </tr>
    </thead>
    <tbody id="crewBody">
        <?php
        // Fetch user data from the database and display it in the table
        include '../php/connection.php';
        $result = $conn->query("SELECT * FROM users");

        if ($result->num_rows > 0) {
            $counter = 1;
            while ($row = $result->fetch_assoc()) {
                echo "<tr>";
                echo "<td>" . $counter++ . "</td>";
                echo "<td>" . $row["crewId"] . "</td>";
                echo "<td>" . $row["name"] . "</td>";
                echo "<td>" . $row["username"] . "</td>";
                echo "<td>" . $row["department"] . "</td>";
                echo "<td>" . $row["userRole"] . "</td>";
                echo "<td><button class='edit-button' onclick=\"editCrew('{$row["crewId"]}')\">Edit</button></td>";

                echo "</tr>";
            }
        } else {
            echo "<tr><td colspan='8'>No users found</td></tr>";
```

Figure 4.17: Crew Table

The info section contains a button to create a new crew, a search input field with a search button, and a table to display crew details. The table has columns for "No," "Crew ID," "Full Name," "Username," "Department," "User Role," and "Edit.". PHP script included stated if there are users in the database, their details are displayed in the table rows. If no users are found, a message indicating this is displayed in a table row.



```
//========== EDIT FUNCTION ============
function editCrew(crewId) {
    window.location.href = `../admin/crewEdit.php?crewID=${crewId}`;
}
//========== SEARCH FUN ============
function searchCrew() {
        // Declare variables
        var input, filter, table, tbody, tr, td, i, txtValue;
        input = document.getElementById("searchInput");
        filter = input.value.toUpperCase();
        table = document.getElementById("crewTable");
        tbody = document.getElementById("crewBody");
        tr = tbody.getElementsByTagName("tr");

        // Loop through all table rows, and hide those who don't match the search query
        for (i = 0; i < tr.length; i++) {
            // Assume the crew name is in the second column (index 1)
            td = tr[i].getElementsByTagName("td")[2];
            if (td) {
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1) {
                    tr[i].style.display = "";
                } else {
                    tr[i].style.display = "none";
                }
            }
        }
}
```

Figure 4.18: Edit and Search Crew

Function editCrew(crewId) is responsible for redirecting the user to the crew edit page with the crewId appended to the URL. If admin clicked on the edit crew, it would direct them to the ‘crewEdit.php’.

Function searchCrew() handles the search functionality for filtering crew members in the displayed table based on a search input. It first retrieves the search input element, the table, the table body, and the table rows from the document's DOM. Then, it loops through each table row and checks if the crew’s name (assumed to be in the second column, index 1) matches the search query entered by the user. If a matching crew name is found, the row is displayed; otherwise, it is hidden. The search is case-insensitive, as both the crew’s name and search query are converted to uppercase for comparison.

crewCreate.php

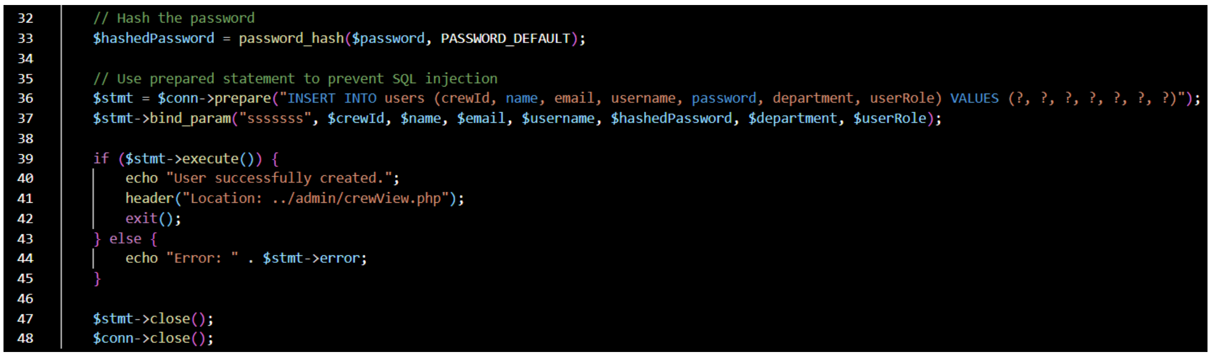

```php
// Hash the password
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

// Use prepared statement to prevent SQL injection
$stmt = $conn->prepare("INSERT INTO users (crewId, name, email, username, password, department, userRole) VALUES (?, ?, ?, ?, ?, ?, ?)");
$stmt->bind_param("sssssss", $crewId, $name, $email, $username, $hashedPassword, $department, $userRole);

if ($stmt->execute()) {
    echo "User successfully created.";
    header("Location: ../admin/crewView.php");
    exit();
} else {
    echo "Error: " . $stmt->error;
}

$stmt->close();
$conn->close();
```

Figure 4.19: Password Hashing

This segment of the coding is where we implement the password hashing.

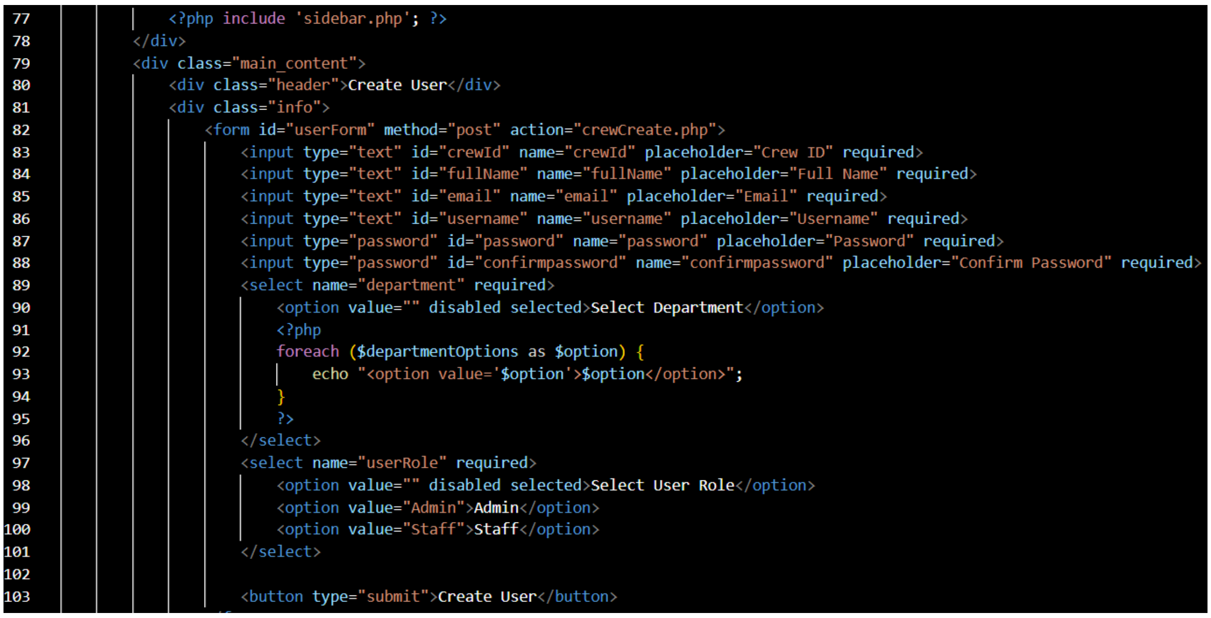
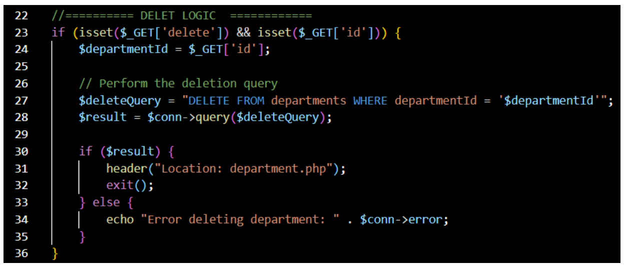

```php
        <?php include 'sidebar.php'; ?>
    </div>
    <div class="main_content">
        <div class="header">Create User</div>
        <div class="info">
            <form id="userForm" method="post" action="crewCreate.php">
                <input type="text" id="crewId" name="crewId" placeholder="Crew ID" required>
                <input type="text" id="fullName" name="fullName" placeholder="Full Name" required>
                <input type="text" id="email" name="email" placeholder="Email" required>
                <input type="text" id="username" name="username" placeholder="Username" required>
                <input type="password" id="password" name="password" placeholder="Password" required>
                <input type="password" id="confirmpassword" name="confirmpassword" placeholder="Confirm Password" required>
                <select name="department" required>
                    <option value="" disabled selected>Select Department</option>
                    <?php
                    foreach ($departmentOptions as $option) {
                        echo "<option value='$option'>$option</option>";
                    }
                    ?>
                </select>
                <select name="userRole" required>
                    <option value="" disabled selected>Select User Role</option>
                    <option value="Admin">Admin</option>
                    <option value="Staff">Staff</option>
                </select>

                <button type="submit">Create User</button>
```

Figure 4.20: Create User Form

Once admin is done with creating new user, they can press the button and the data of the users will be inserted into the database.

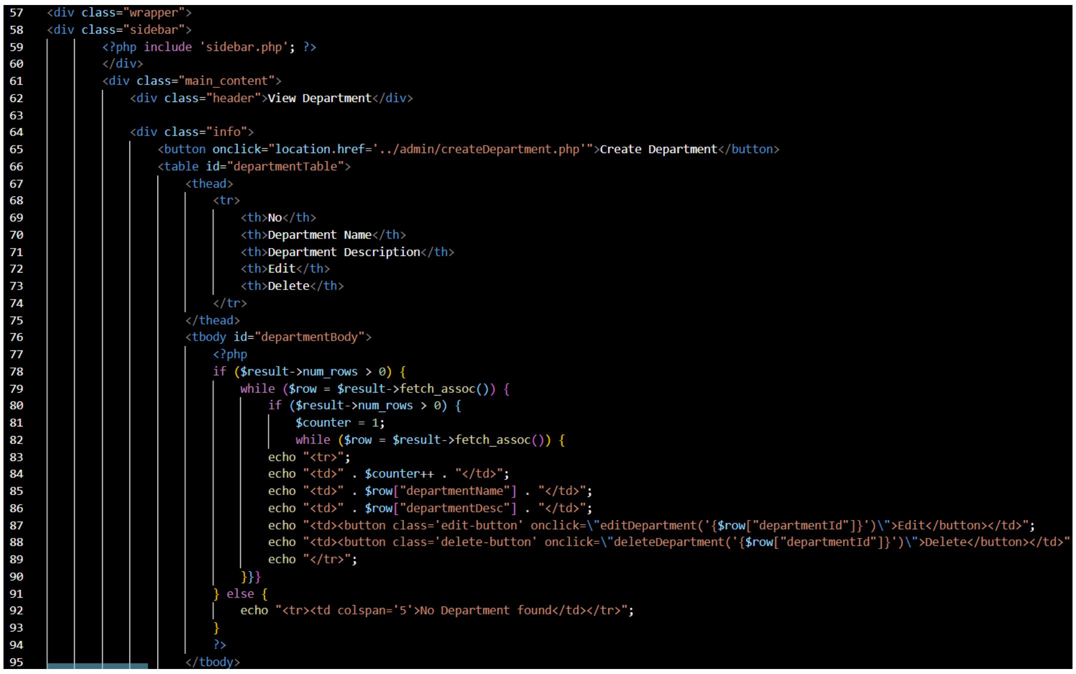
Department.php



```
//========== DELET LOGIC  ============
if (isset($_GET['delete']) && isset($_GET['id'])) {
    $departmentId = $_GET['id'];

    // Perform the deletion query
    $deleteQuery = "DELETE FROM departments WHERE departmentId = '$departmentId'";
    $result = $conn->query($deleteQuery);

    if ($result) {
        header("Location: department.php");
        exit();
    } else {
        echo "Error deleting department: " . $conn->error;
    }
}
```

Figure 4.21: Delete Department

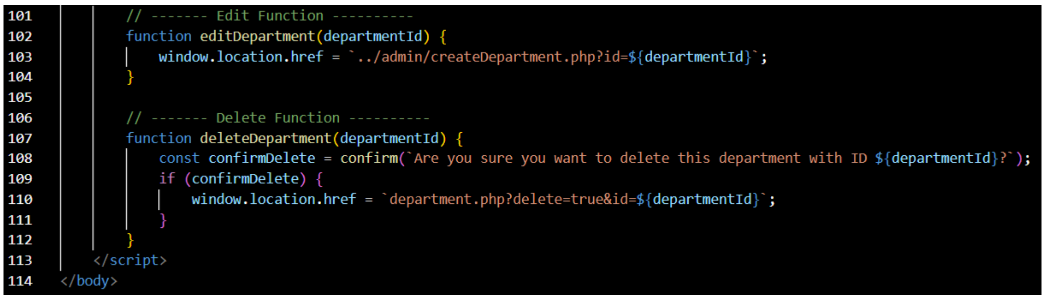
This part of the script handles the deletion query on the "departments" table using the provided department ID. If the deletion is successful, the script redirects the user to the "department.php" page. Otherwise, it outputs an error message indicating the failure to delete the department.



```
<div class="wrapper">
<div class="sidebar">
        <?php include 'sidebar.php'; ?>
        </div>
        <div class="main_content">
            <div class="header">View Department</div>

            <div class="info">
                <button onclick="location.href='../admin/createDepartment.php'">Create Department</button>
                <table id="departmentTable">
                    <thead>
                        <tr>
                            <th>No</th>
                            <th>Department Name</th>
                            <th>Department Description</th>
                            <th>Edit</th>
                            <th>Delete</th>
                        </tr>
                    </thead>
                    <tbody id="departmentBody">
                        <?php
                        if ($result->num_rows > 0) {
                            while ($row = $result->fetch_assoc()) {
                                if ($result->num_rows > 0) {
                                    $counter = 1;
                                    while ($row = $result->fetch_assoc()) {
                                echo "<tr>";
                                echo "<td>" . $counter++ . "</td>";
                                echo "<td>" . $row["departmentName"] . "</td>";
                                echo "<td>" . $row["departmentDesc"] . "</td>";
                                echo "<td><button class='edit-button' onclick=\"editDepartment('{$row["departmentId"]}')\">Edit</button></td>";
                                echo "<td><button class='delete-button' onclick=\"deleteDepartment('{$row["departmentId"]}')\">Delete</button></td>"
                                echo "</tr>";
                            }}}
                        } else {
                            echo "<tr><td colspan='5'>No Department found</td></tr>";
                        }
                        ?>
                    </tbody>
```

Figure 4.22: Department HTML

For this segment of coding, there is a "Create Department" button that, when clicked, seems to redirect to ‘createDepartment.php'. Below that, there is a table with the id "departmentTable" consist of “No", "Department Name", "Department Description", "Edit", and "Delete". The PHP code within the table body is for fetching and displaying department data from a database. It checks if there are any departments, and if so, it iterates through them to display their details and options for editing and deleting.

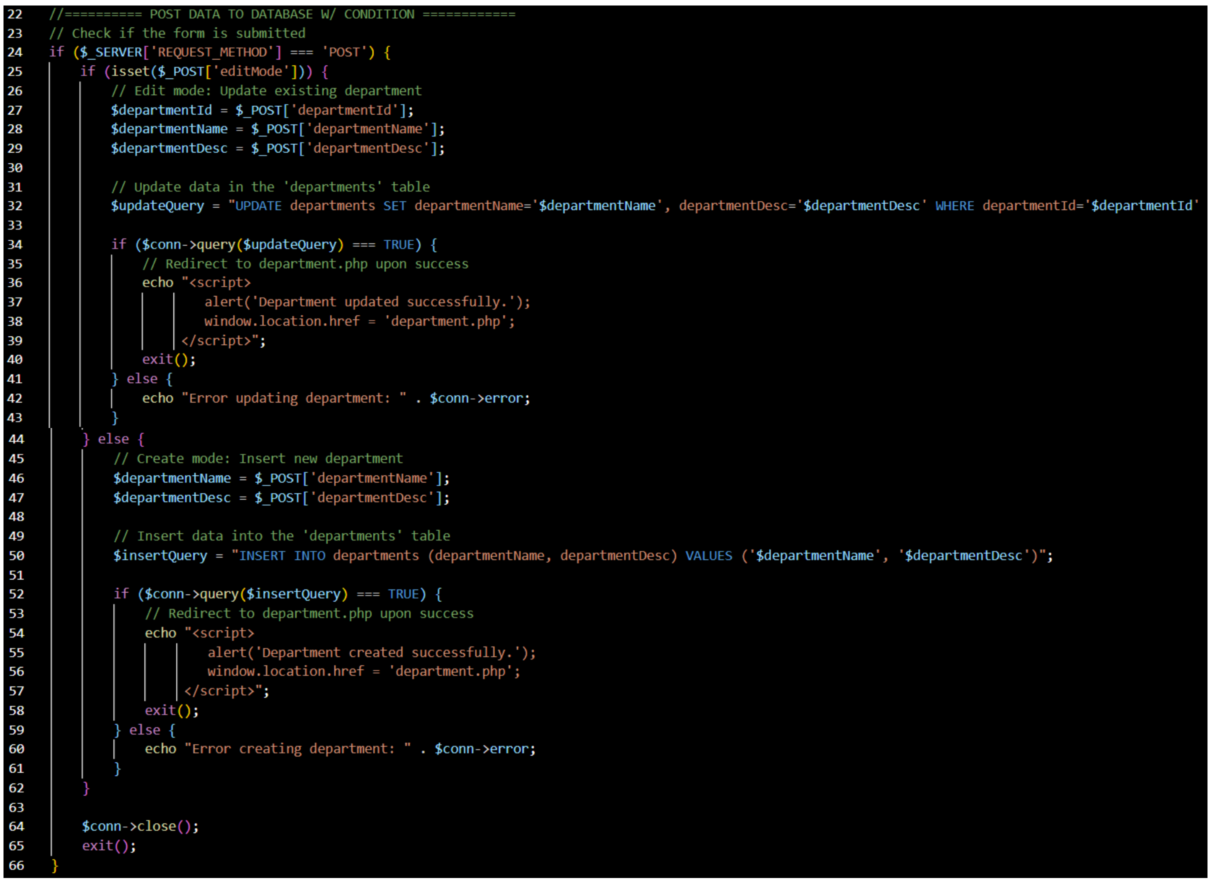


```
        // ------- Edit Function ----------
        function editDepartment(departmentId) {
            window.location.href = `../admin/createDepartment.php?id=${departmentId}`;
        }

        // ------- Delete Function ----------
        function deleteDepartment(departmentId) {
            const confirmDelete = confirm(`Are you sure you want to delete this department with ID ${departmentId}?`);
            if (confirmDelete) {
                window.location.href = `department.php?delete=true&id=${departmentId}`;
            }
        }
    </script>
</body>
```

Figure 4.23: Department HTML Edit/Delete

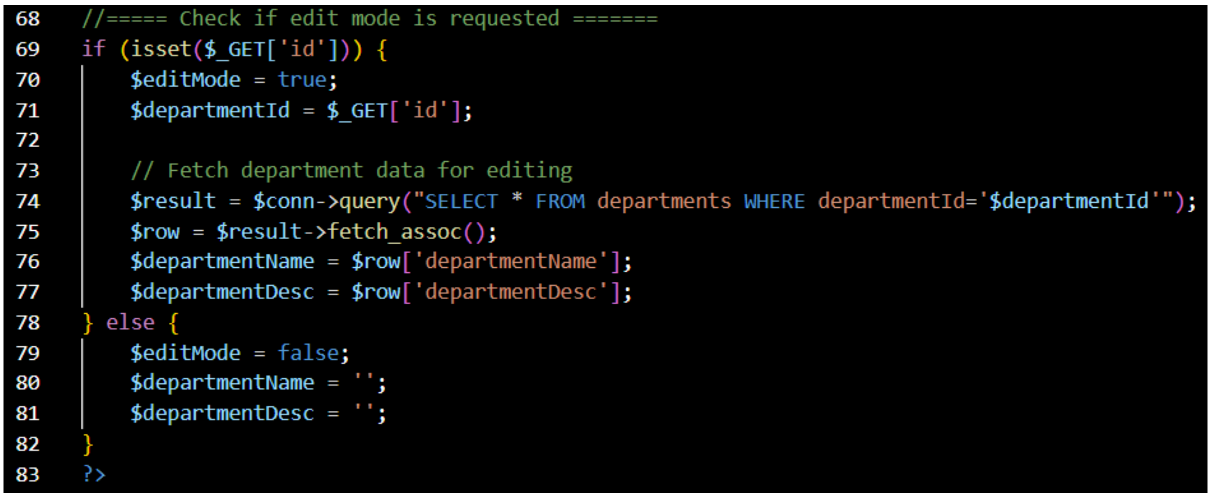
Figure 4.23 shows the edit and delete function for the website. The button Edit will direct user to the ‘createdepartment.php’ so admin can edit department. Delete button will delete the chosen department.

Createdepartment.php



```php
//========== POST DATA TO DATABASE W/ CONDITION ============
// Check if the form is submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['editMode'])) {
        // Edit mode: Update existing department
        $departmentId = $_POST['departmentId'];
        $departmentName = $_POST['departmentName'];
        $departmentDesc = $_POST['departmentDesc'];

        // Update data in the 'departments' table
        $updateQuery = "UPDATE departments SET departmentName='$departmentName', departmentDesc='$departmentDesc' WHERE departmentId='$departmentId'

        if ($conn->query($updateQuery) === TRUE) {
            // Redirect to department.php upon success
            echo "<script>
                    alert('Department updated successfully.');
                    window.location.href = 'department.php';
                </script>";
            exit();
        } else {
            echo "Error updating department: " . $conn->error;
        }
    } else {
        // Create mode: Insert new department
        $departmentName = $_POST['departmentName'];
        $departmentDesc = $_POST['departmentDesc'];

        // Insert data into the 'departments' table
        $insertQuery = "INSERT INTO departments (departmentName, departmentDesc) VALUES ('$departmentName', '$departmentDesc')";

        if ($conn->query($insertQuery) === TRUE) {
            // Redirect to department.php upon success
            echo "<script>
                    alert('Department created successfully.');
                    window.location.href = 'department.php';
                </script>";
            exit();
        } else {
            echo "Error creating department: " . $conn->error;
        }
    }

    $conn->close();
    exit();
}
```

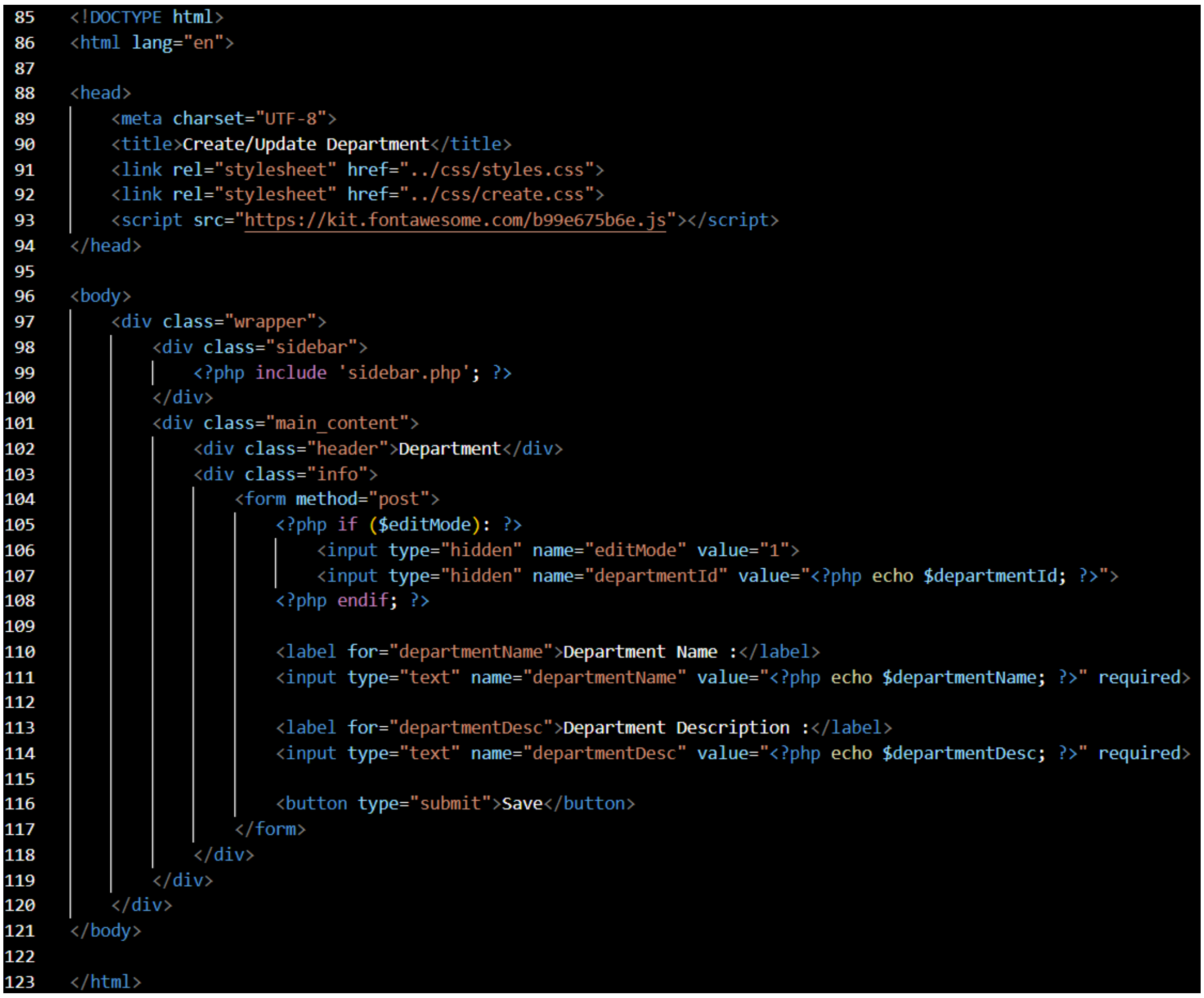
Figure 4.24: Department Form



```php
//===== Check if edit mode is requested =======
if (isset($_GET['id'])) {
    $editMode = true;
    $departmentId = $_GET['id'];

    // Fetch department data for editing
    $result = $conn->query("SELECT * FROM departments WHERE departmentId='$departmentId'");
    $row = $result->fetch_assoc();
    $departmentName = $row['departmentName'];
    $departmentDesc = $row['departmentDesc'];
} else {
    $editMode = false;
    $departmentName = '';
    $departmentDesc = '';
}
?>
```

Figure 4.25: Department Edit Request

Figure 4.24 and 4.25 shows the coding section where this code is meant to be handling user authentication, retrieving user profile information, and managing the creation/update of department data in a database based on the user's input.

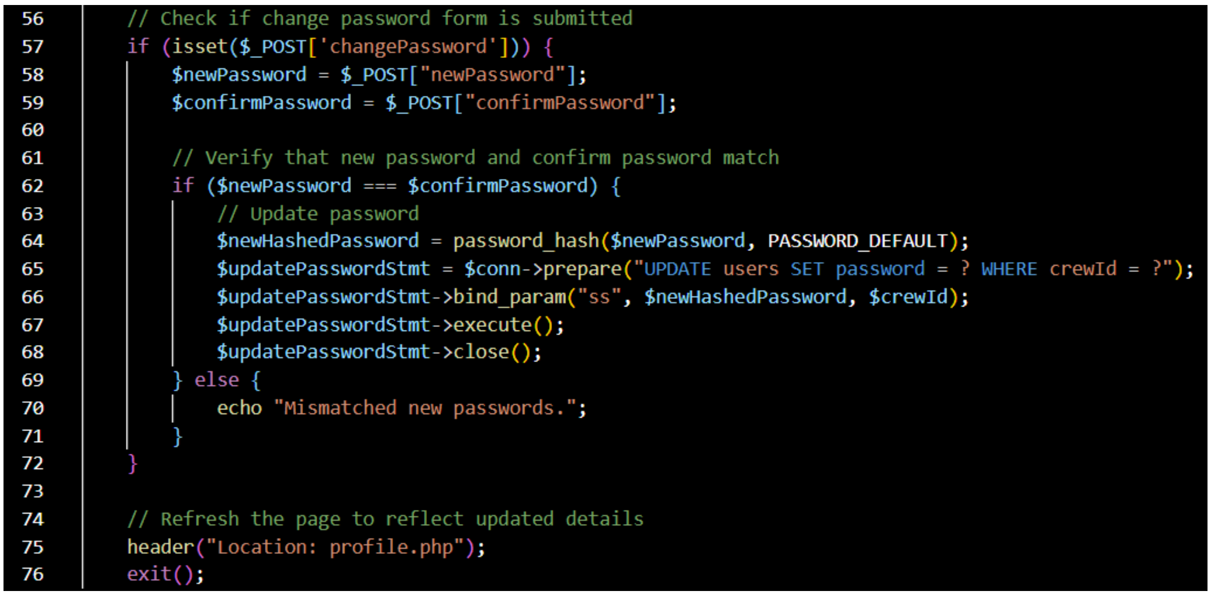


```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Create/Update Department</title>
    <link rel="stylesheet" href="../css/styles.css">
    <link rel="stylesheet" href="../css/create.css">
    <script src="https://kit.fontawesome.com/b99e675b6e.js"></script>
</head>

<body>
    <div class="wrapper">
        <div class="sidebar">
            <?php include 'sidebar.php'; ?>
        </div>
        <div class="main_content">
            <div class="header">Department</div>
            <div class="info">
                <form method="post">
                    <?php if ($editMode): ?>
                        <input type="hidden" name="editMode" value="1">
                        <input type="hidden" name="departmentId" value="<?php echo $departmentId; ?>">
                    <?php endif; ?>

                    <label for="departmentName">Department Name :</label>
                    <input type="text" name="departmentName" value="<?php echo $departmentName; ?>" required>

                    <label for="departmentDesc">Department Description :</label>
                    <input type="text" name="departmentDesc" value="<?php echo $departmentDesc; ?>" required>

                    <button type="submit">Save</button>
                </form>
            </div>
        </div>
    </div>
</body>

</html>
```

Figure 4.26: Create Department HTML

This code snippet represents a web page section for managing department details. It includes a form for adding or editing department informationThe main content area contains a header "Department" and a form for managing department information. If a condition $editMode is true, hidden input fields are added to the form to indicate the edit mode and the department ID. There is a "Save" button inside the form that, when clicked, likely submits the form data to the server for processing.

profile.php



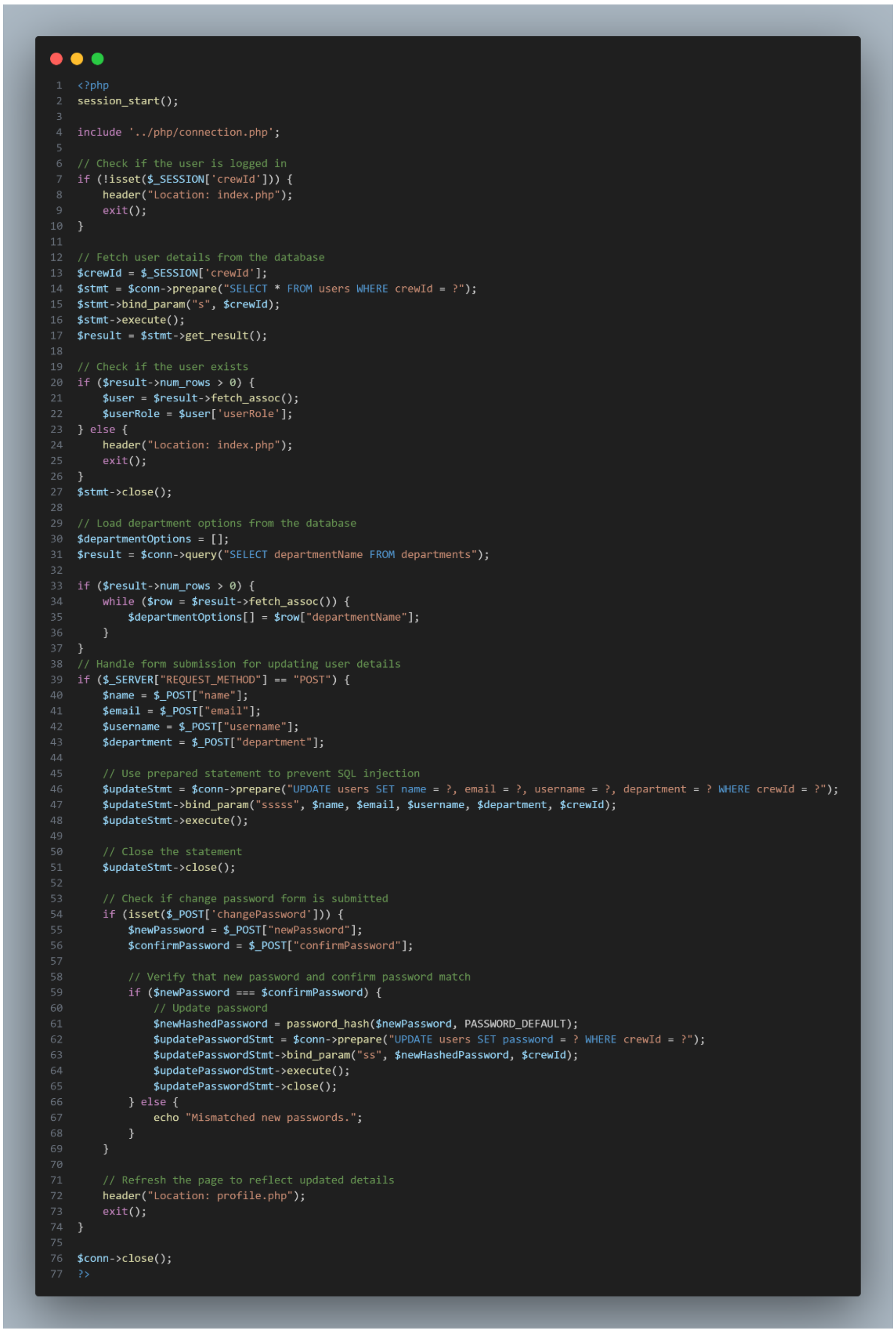
```php
    // Check if change password form is submitted
    if (isset($_POST['changePassword'])) {
        $newPassword = $_POST["newPassword"];
        $confirmPassword = $_POST["confirmPassword"];

        // Verify that new password and confirm password match
        if ($newPassword === $confirmPassword) {
            // Update password
            $newHashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);
            $updatePasswordStmt = $conn->prepare("UPDATE users SET password = ? WHERE crewId = ?");
            $updatePasswordStmt->bind_param("ss", $newHashedPassword, $crewId);
            $updatePasswordStmt->execute();
            $updatePasswordStmt->close();
        } else {
            echo "Mismatched new passwords.";
        }
    }

    // Refresh the page to reflect updated details
    header("Location: profile.php");
    exit();
```

Figure 4.27: Change Password in Admin Profile

If the change password form is submitted, it retrieves the new password and confirm password from the form using $_POST["newPassword"] and $_POST["confirmPassword"] respectively. It then verifies that the new password and confirm password match by comparing the values using an if statement. If the passwords match, the code proceeds to update the password. The new password is hashed using the password_hash() function with the PASSWORD_DEFAULT algorithm, resulting in a secure hashed password. A prepared statement is used to update the password in the database. The statement updates the "password" field in the "users" table for the user with a specific "crewId" (presumably obtained from the user's session or another source). If the new password and confirm password do not match, an error message "Mismatched new passwords." is displayed.

4.3 Crew Side

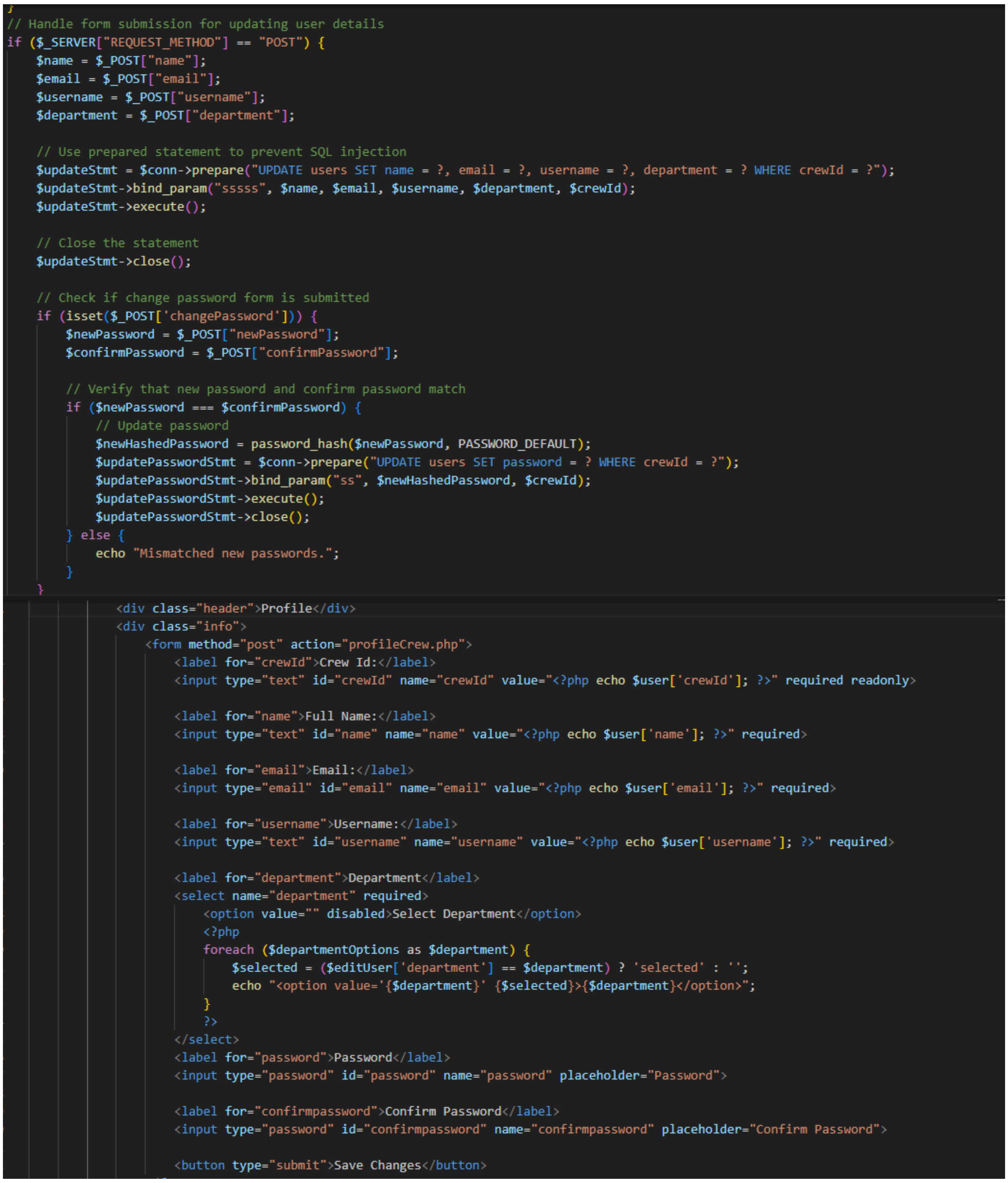
dashcrew.php



```php
<?php
session_start();

include '../php/connection.php';

// Check if the user is logged in
if (!isset($_SESSION['crewId'])) {
    header("Location: index.php");
    exit();
}

// Fetch user details from the database
$crewId = $_SESSION['crewId'];
$stmt = $conn->prepare("SELECT * FROM users WHERE crewId = ?");
$stmt->bind_param("s", $crewId);
$stmt->execute();
$result = $stmt->get_result();

// Check if the user exists
if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();
    $userRole = $user['userRole'];
} else {
    header("Location: index.php");
    exit();
}
$stmt->close();

// Load department options from the database
$departmentOptions = [];
$result = $conn->query("SELECT departmentName FROM departments");

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $departmentOptions[] = $row["departmentName"];
    }
}
// Handle form submission for updating user details
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST["name"];
    $email = $_POST["email"];
    $username = $_POST["username"];
    $department = $_POST["department"];

    // Use prepared statement to prevent SQL injection
    $updateStmt = $conn->prepare("UPDATE users SET name = ?, email = ?, username = ?, department = ? WHERE crewId = ?");
    $updateStmt->bind_param("sssss", $name, $email, $username, $department, $crewId);
    $updateStmt->execute();

    // Close the statement
    $updateStmt->close();

    // Check if change password form is submitted
    if (isset($_POST['changePassword'])) {
        $newPassword = $_POST["newPassword"];
        $confirmPassword = $_POST["confirmPassword"];

        // Verify that new password and confirm password match
        if ($newPassword === $confirmPassword) {
            // Update password
            $newHashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);
            $updatePasswordStmt = $conn->prepare("UPDATE users SET password = ? WHERE crewId = ?");
            $updatePasswordStmt->bind_param("ss", $newHashedPassword, $crewId);
            $updatePasswordStmt->execute();
            $updatePasswordStmt->close();
        } else {
            echo "Mismatched new passwords.";
        }
    }

    // Refresh the page to reflect updated details
    header("Location: profile.php");
    exit();
}

$conn->close();
?>
```

Figure 4.28: Dashboardcrew

profilecrew.php

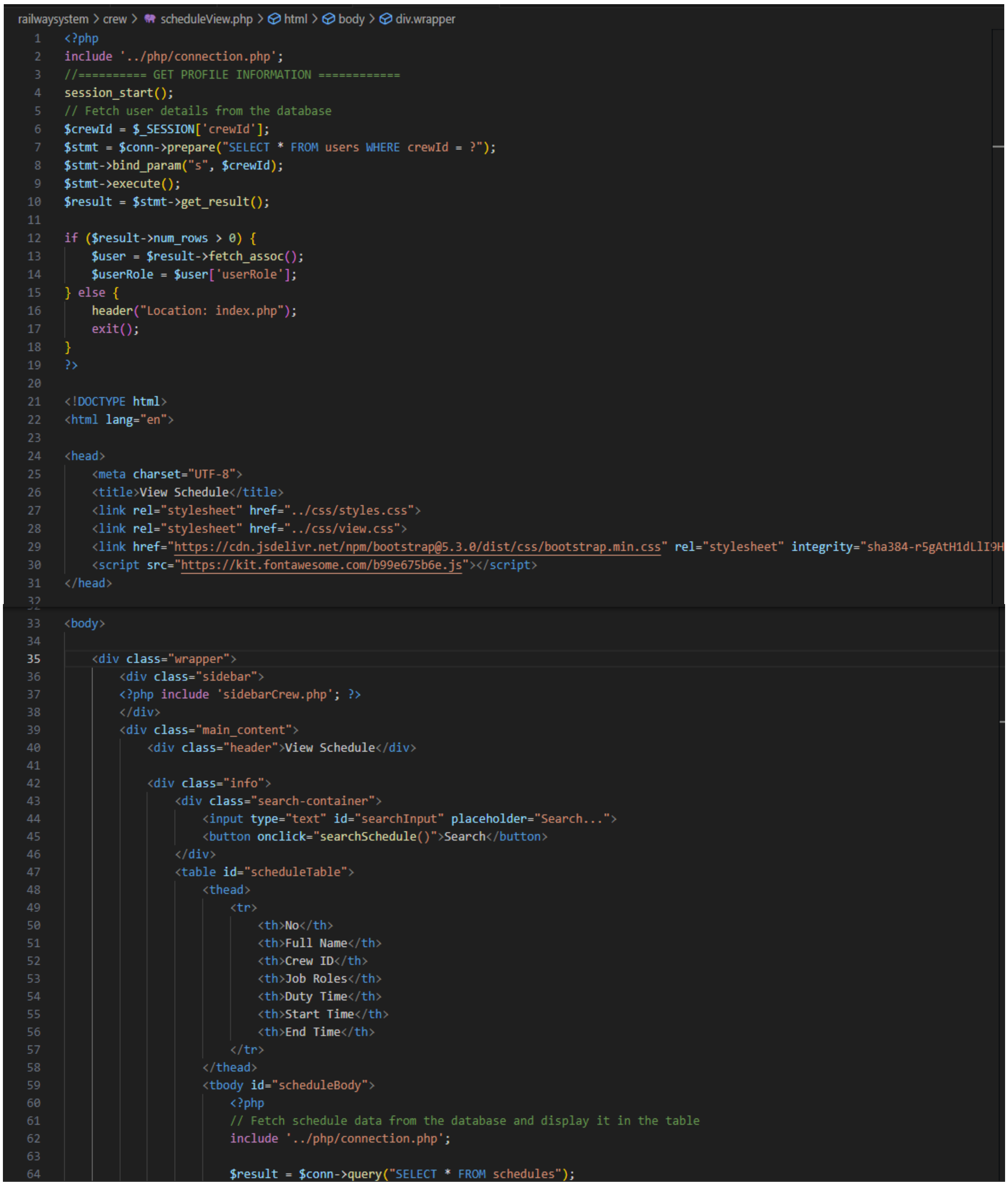


```
// Handle form submission for updating user details
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST["name"];
    $email = $_POST["email"];
    $username = $_POST["username"];
    $department = $_POST["department"];

    // Use prepared statement to prevent SQL injection
    $updateStmt = $conn->prepare("UPDATE users SET name = ?, email = ?, username = ?, department = ? WHERE crewId = ?");
    $updateStmt->bind_param("sssss", $name, $email, $username, $department, $crewId);
    $updateStmt->execute();

    // Close the statement
    $updateStmt->close();

    // Check if change password form is submitted
    if (isset($_POST['changePassword'])) {
        $newPassword = $_POST["newPassword"];
        $confirmPassword = $_POST["confirmPassword"];

        // Verify that new password and confirm password match
        if ($newPassword === $confirmPassword) {
            // Update password
            $newHashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);
            $updatePasswordStmt = $conn->prepare("UPDATE users SET password = ? WHERE crewId = ?");
            $updatePasswordStmt->bind_param("ss", $newHashedPassword, $crewId);
            $updatePasswordStmt->execute();
            $updatePasswordStmt->close();
        } else {
            echo "Mismatched new passwords.";
        }
    }
```

```
                <div class="header">Profile</div>
                <div class="info">
                    <form method="post" action="profileCrew.php">
                        <label for="crewId">Crew Id:</label>
                        <input type="text" id="crewId" name="crewId" value="<?php echo $user['crewId']; ?>" required readonly>

                        <label for="name">Full Name:</label>
                        <input type="text" id="name" name="name" value="<?php echo $user['name']; ?>" required>

                        <label for="email">Email:</label>
                        <input type="email" id="email" name="email" value="<?php echo $user['email']; ?>" required>

                        <label for="username">Username:</label>
                        <input type="text" id="username" name="username" value="<?php echo $user['username']; ?>" required>

                        <label for="department">Department</label>
                        <select name="department" required>
                            <option value="" disabled>Select Department</option>
                            <?php
                            foreach ($departmentOptions as $department) {
                                $selected = ($editUser['department'] == $department) ? 'selected' : '';
                                echo "<option value='{$department}' {$selected}>{$department}</option>";
                            }
                            ?>
                        </select>
                        <label for="password">Password</label>
                        <input type="password" id="password" name="password" placeholder="Password">

                        <label for="confirmpassword">Confirm Password</label>
                        <input type="password" id="confirmpassword" name="confirmpassword" placeholder="Confirm Password">

                        <button type="submit">Save Changes</button>
```

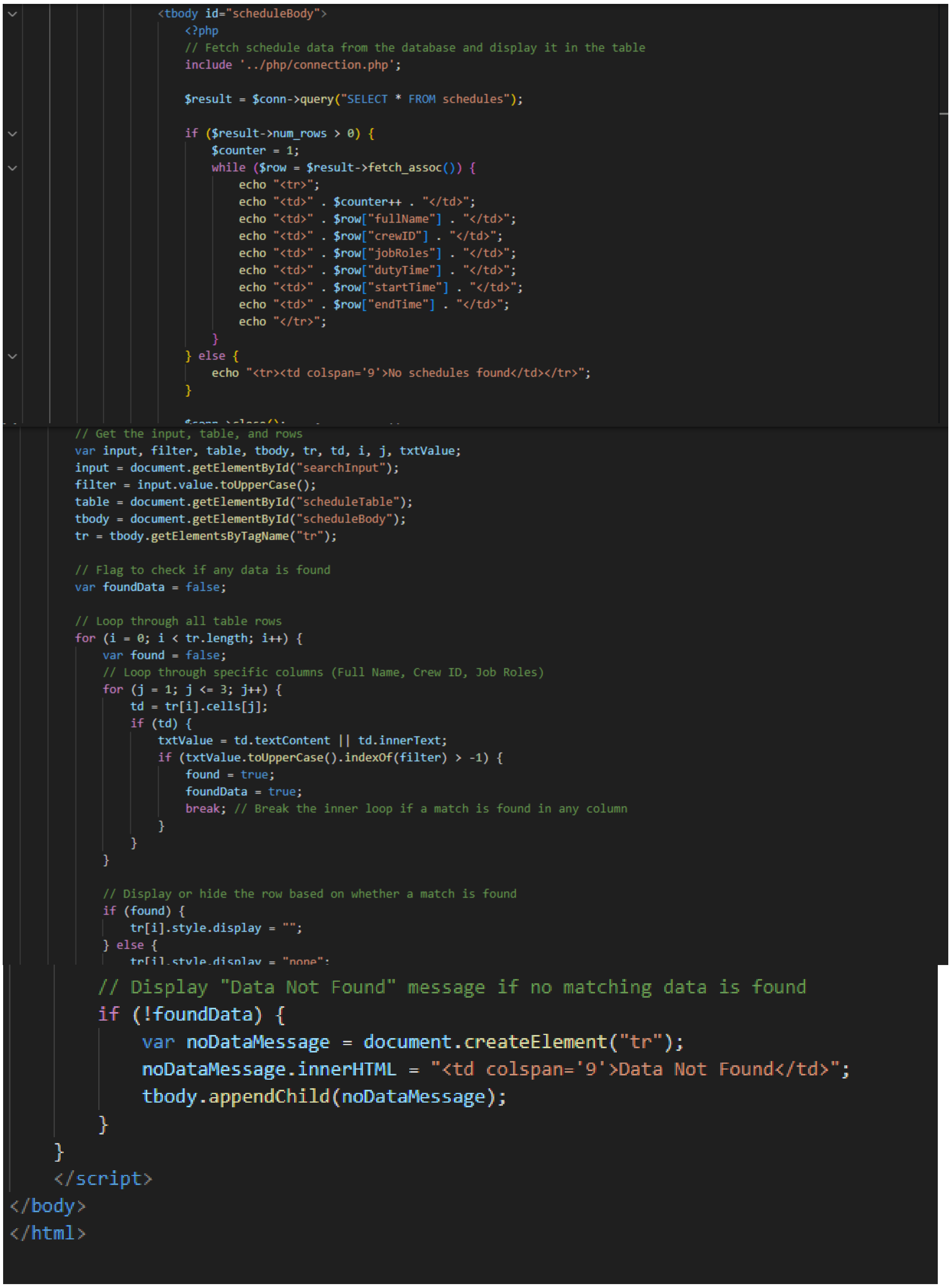
Figure 4.29: Update Profile Crew

Scheduleview.php


railwaysystem > crew > scheduleView.php > html > body > div.wrapper

```
<?php
include '../php/connection.php';
//========== GET PROFILE INFORMATION ============
session_start();
// Fetch user details from the database
$crewId = $_SESSION['crewId'];
$stmt = $conn->prepare("SELECT * FROM users WHERE crewId = ?");
$stmt->bind_param("s", $crewId);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();
    $userRole = $user['userRole'];
} else {
    header("Location: index.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>View Schedule</title>
    <link rel="stylesheet" href="../css/styles.css">
    <link rel="stylesheet" href="../css/view.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-r5gAtH1dLlI9H
    <script src="https://kit.fontawesome.com/b99e675b6e.js"></script>
</head>

<body>

    <div class="wrapper">
        <div class="sidebar">
        <?php include 'sidebarCrew.php'; ?>
        </div>
        <div class="main_content">
            <div class="header">View Schedule</div>

            <div class="info">
                <div class="search-container">
                    <input type="text" id="searchInput" placeholder="Search...">
                    <button onclick="searchSchedule()">Search</button>
                </div>
                <table id="scheduleTable">
                    <thead>
                        <tr>
                            <th>No</th>
                            <th>Full Name</th>
                            <th>Crew ID</th>
                            <th>Job Roles</th>
                            <th>Duty Time</th>
                            <th>Start Time</th>
                            <th>End Time</th>
                        </tr>
                    </thead>
                    <tbody id="scheduleBody">
                        <?php
                        // Fetch schedule data from the database and display it in the table
                        include '../php/connection.php';

                        $result = $conn->query("SELECT * FROM schedules");
```

Figure 4.30: Scheduleview(1)

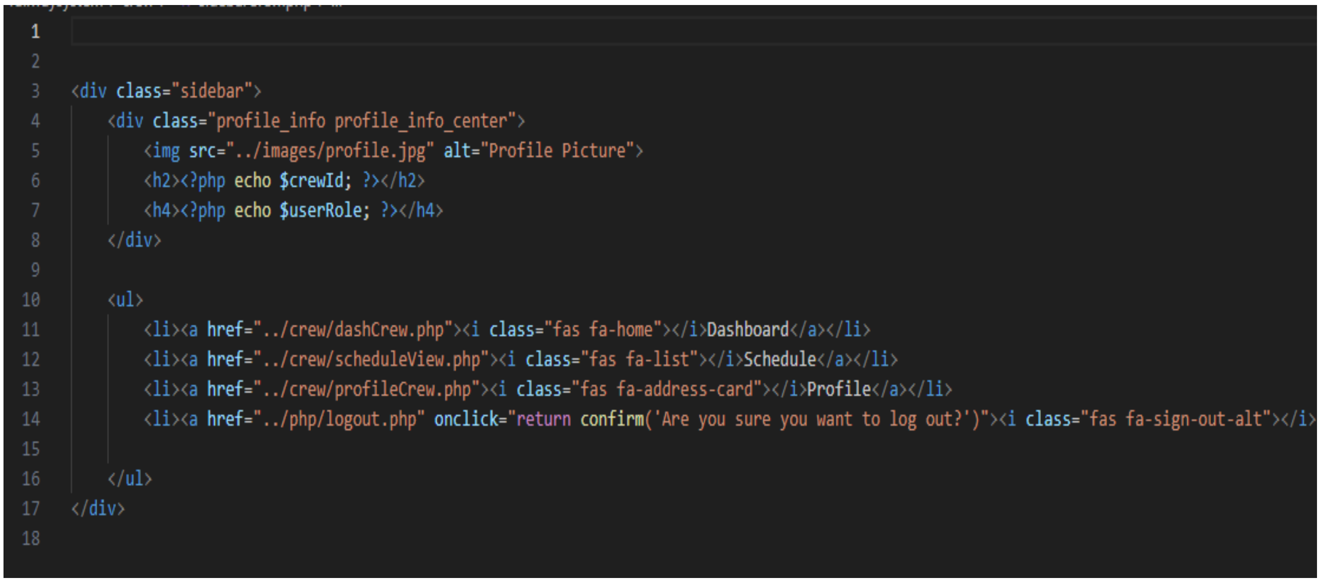


```
                <tbody id="scheduleBody">
                    <?php
                    // Fetch schedule data from the database and display it in the table
                    include '../php/connection.php';

                    $result = $conn->query("SELECT * FROM schedules");

                    if ($result->num_rows > 0) {
                        $counter = 1;
                        while ($row = $result->fetch_assoc()) {
                            echo "<tr>";
                            echo "<td>" . $counter++ . "</td>";
                            echo "<td>" . $row["fullName"] . "</td>";
                            echo "<td>" . $row["crewID"] . "</td>";
                            echo "<td>" . $row["jobRoles"] . "</td>";
                            echo "<td>" . $row["dutyTime"] . "</td>";
                            echo "<td>" . $row["startTime"] . "</td>";
                            echo "<td>" . $row["endTime"] . "</td>";
                            echo "</tr>";
                        }
                    } else {
                        echo "<tr><td colspan='9'>No schedules found</td></tr>";
                    }

                    $conn->close();
```

```
        // Get the input, table, and rows
        var input, filter, table, tbody, tr, td, i, j, txtValue;
        input = document.getElementById("searchInput");
        filter = input.value.toUpperCase();
        table = document.getElementById("scheduleTable");
        tbody = document.getElementById("scheduleBody");
        tr = tbody.getElementsByTagName("tr");

        // Flag to check if any data is found
        var foundData = false;

        // Loop through all table rows
        for (i = 0; i < tr.length; i++) {
            var found = false;
            // Loop through specific columns (Full Name, Crew ID, Job Roles)
            for (j = 1; j <= 3; j++) {
                td = tr[i].cells[j];
                if (td) {
                    txtValue = td.textContent || td.innerText;
                    if (txtValue.toUpperCase().indexOf(filter) > -1) {
                        found = true;
                        foundData = true;
                        break; // Break the inner loop if a match is found in any column
                    }
                }
            }

            // Display or hide the row based on whether a match is found
            if (found) {
                tr[i].style.display = "";
            } else {
                tr[i].style.display = "none";
```

```
        // Display "Data Not Found" message if no matching data is found
        if (!foundData) {
            var noDataMessage = document.createElement("tr");
            noDataMessage.innerHTML = "<td colspan='9'>Data Not Found</td>";
            tbody.appendChild(noDataMessage);
        }
    }
    </script>
</body>
</html>
```

Figure 4.31: Scheduleview(2)



```


<div class="sidebar">
    <div class="profile_info profile_info_center">
        <img src="../images/profile.jpg" alt="Profile Picture">
        <h2><?php echo $crewId; ?></h2>
        <h4><?php echo $userRole; ?></h4>
    </div>

    <ul>
        <li><a href="../crew/dashCrew.php"><i class="fas fa-home"></i>Dashboard</a></li>
        <li><a href="../crew/scheduleView.php"><i class="fas fa-list"></i>Schedule</a></li>
        <li><a href="../crew/profileCrew.php"><i class="fas fa-address-card"></i>Profile</a></li>
        <li><a href="../php/logout.php" onclick="return confirm('Are you sure you want to log out?')"><i class="fas fa-sign-out-alt"></i>

    </ul>
</div>

```

Figure 4.32: Sidebarcrew Form

5.0 Result

5.1 Home Page

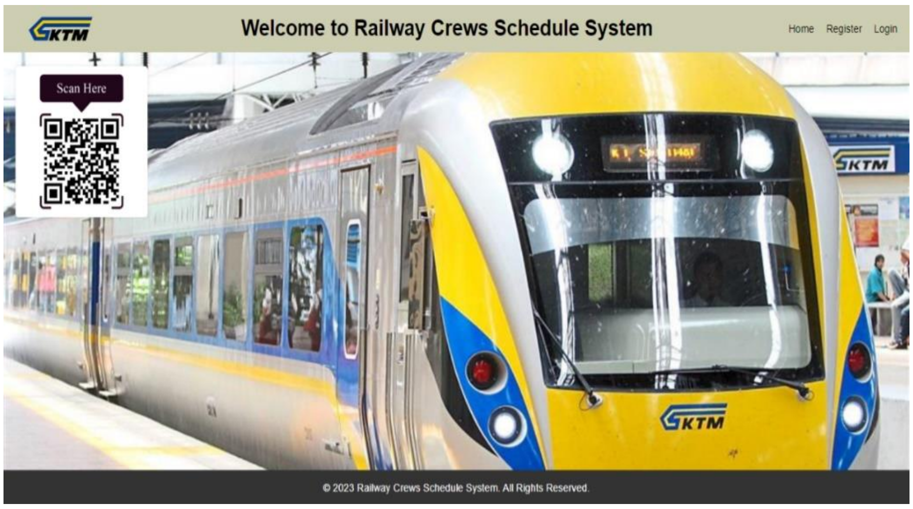
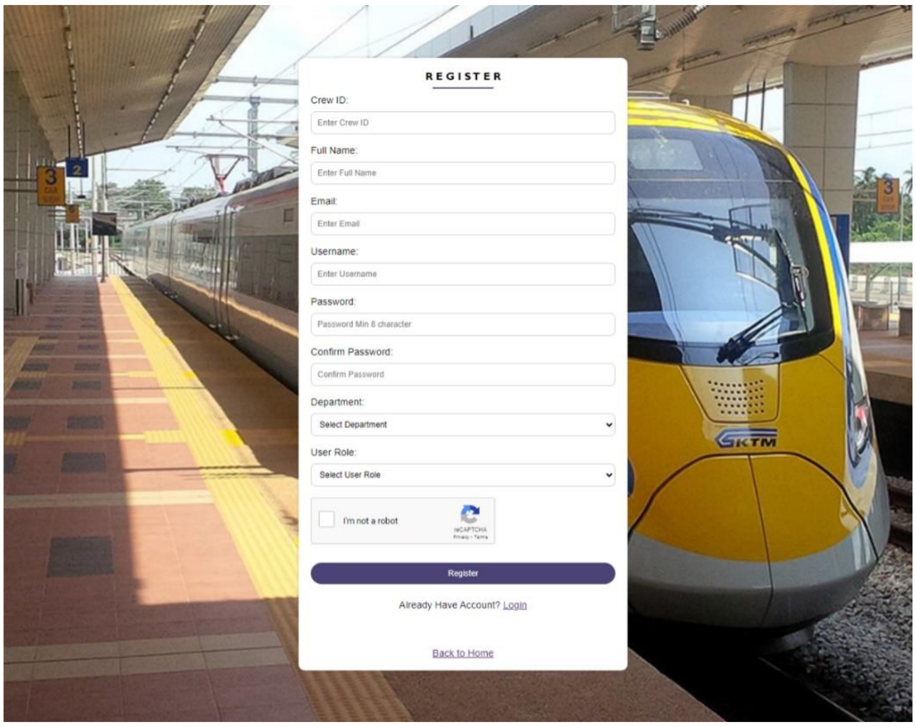

KTM
Welcome to Railway Crews Schedule System
Home
Register
Login
Scan Here
© 2023 Railway Crews Schedule System. All Rights Reserved.

Figure 5.1: Home page

This is the home page of Railway Crews Schedule System. On this page, there are a few navigations bar including Home which will redirect to this page, register which will redirect to the register page and Login which will redirect to the login page.


REGISTER

Crew ID: Enter Crew ID

Full Name: Enter Full Name

Email: Enter Email

Username: Enter Username

Password: Password Min 8 character

Confirm Password: Confirm Password

Department: Select Department

User Role: Select User Role

I'm not a robot

Register

Already Have Account? Login

Back to Home

Figure 5.2: Register page for admin and crew page

On this member registration page shown in Figure 5.2, user can fill in their details such as crew id, full name, email address, username, password, confirm password, department, and they will choose a user role. After user entered their details and set up their password, they can easily register by clicking the register button.

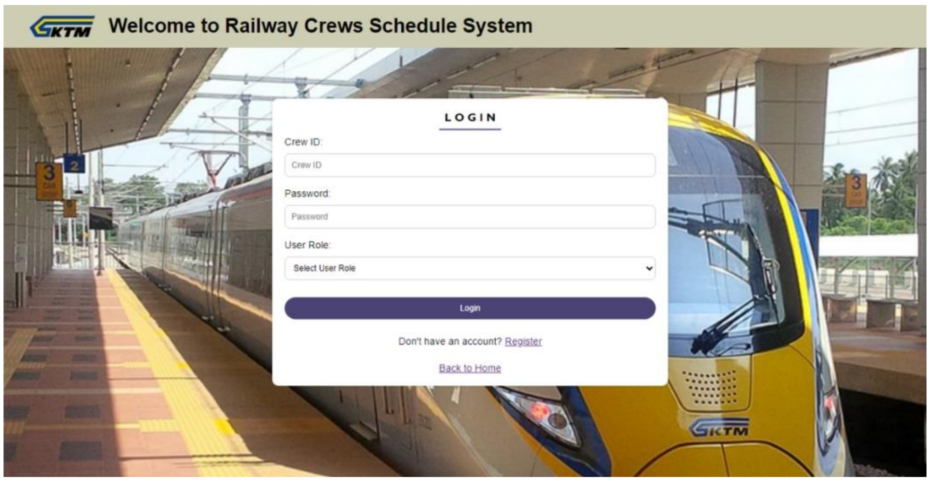


Figure 5.3: Login Page for Admin and Staff Page

Users can log in to their registered account by entering crew id, select their role and the password they set during member registration. Then, users can log in to their account by clicking the login button.

5.2 Admin Side

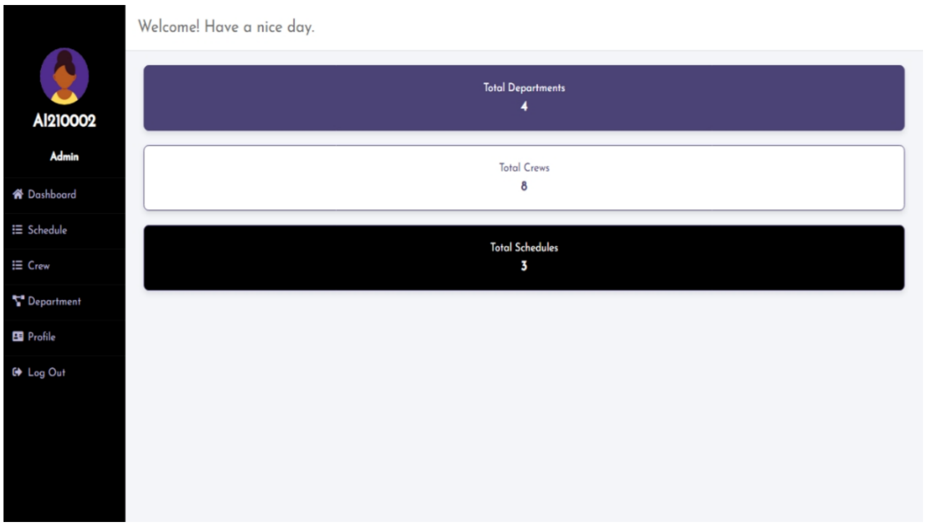
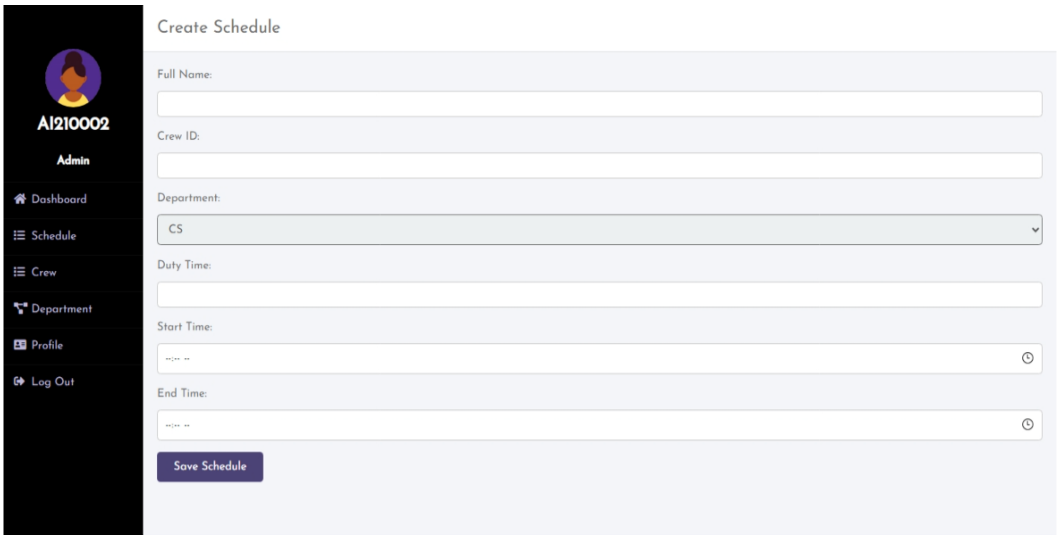


Figure 5.4: Dashboard Admin Page

This is the welcome page of administration contain total departments, total crews, and total schedules.



Create Schedule

Full Name:

Crew ID:

Department: CS

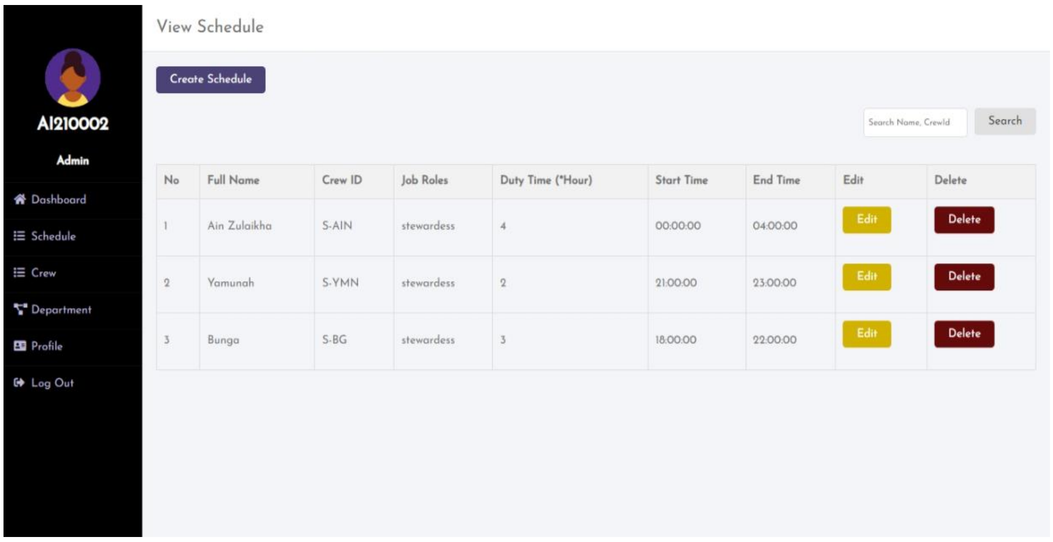
Duty Time:

Start Time: --:-- --

End Time: --:-- --

Figure 5.5: Create Schedule Page

On this page, admin can create schedule for crews.

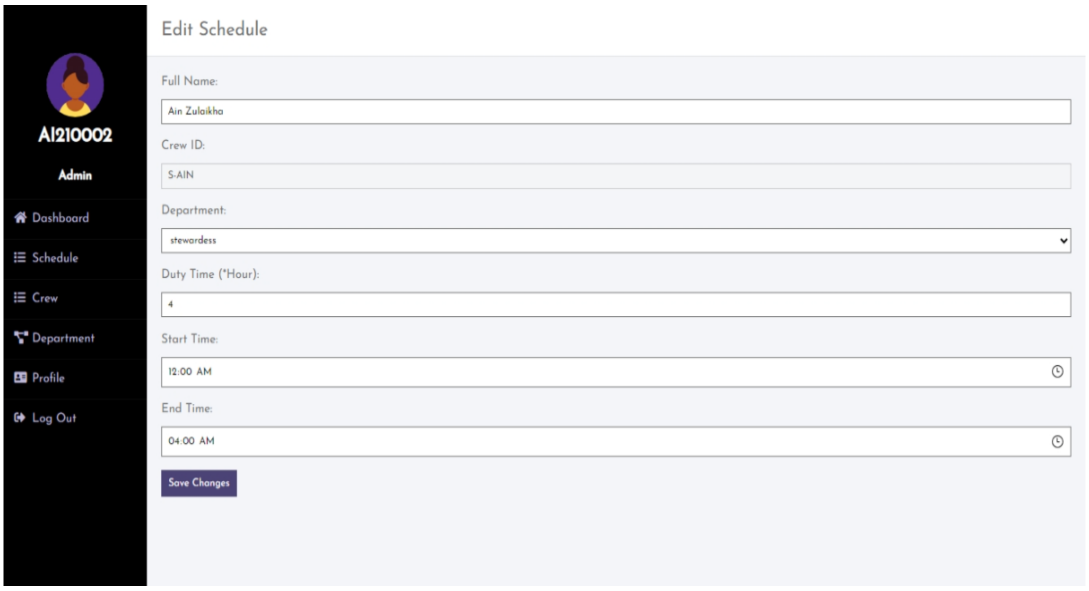


View Schedule

No	Full Name	Crew ID	Job Roles	Duty Time (*Hour)	Start Time	End Time	Edit	Delete
1	Ain Zulaikha	S-AIN	stewardess	4	00:00:00	04:00:00	Edit	Delete
2	Yamunah	S-YMN	stewardess	2	21:00:00	23:00:00	Edit	Delete
3	Bunga	S-BG	stewardess	3	18:00:00	22:00:00	Edit	Delete

Figure 5.6: View schedule page

This is a view schedule page where the admin can view the schedule that they created by themselves. They also can edit the schedule again and delete the schedule if not necessary.



Edit Schedule

Full Name: Ain Zulaikha

Crew ID: S-AIN

Department: stewardess

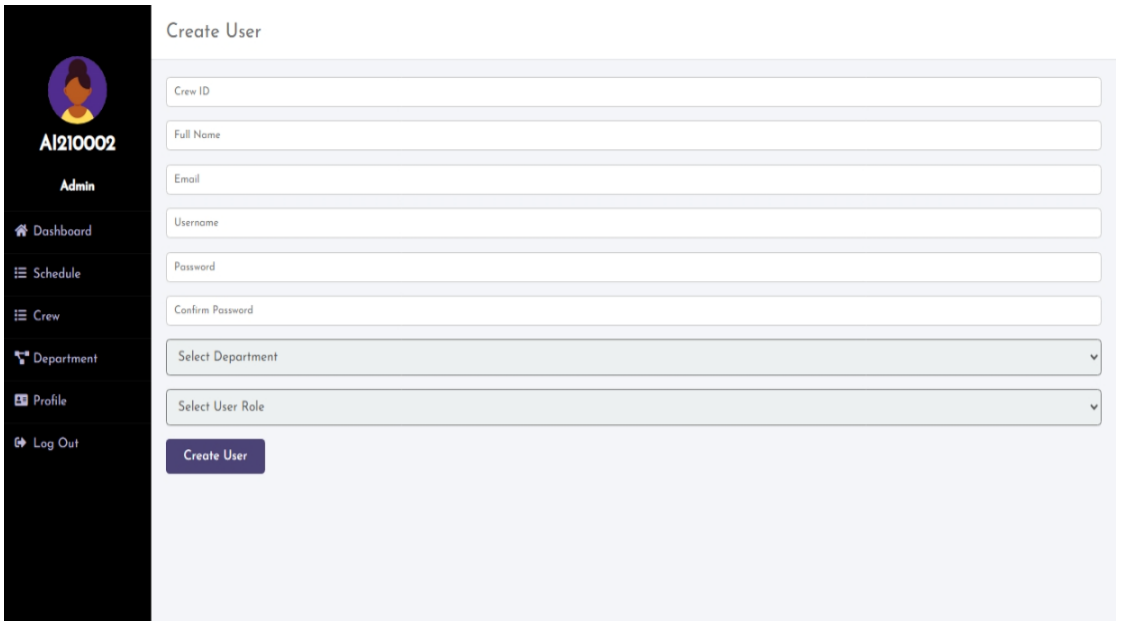
Duty Time (*Hour): 4

Start Time: 12:00 AM

End Time: 04:00 AM

Figure 5.7: Edit Schedule Page

This is an edit schedule page for admin to edit the schedule for crew if needed.



Create User

Crew ID

Full Name

Email

Username

Password

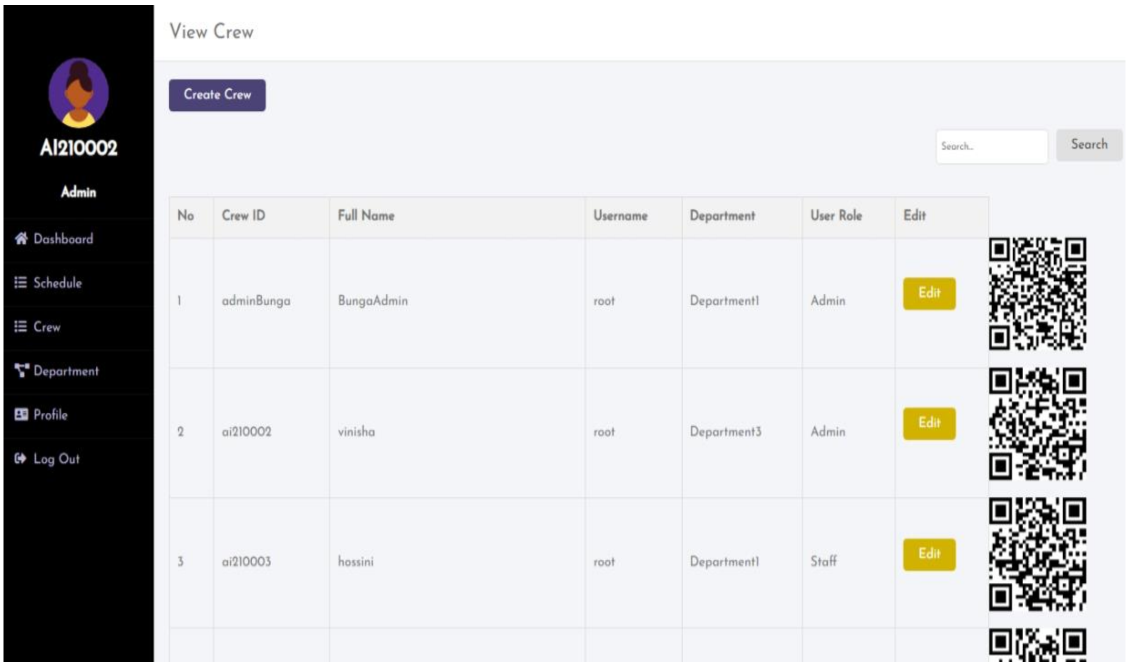
Confirm Password

Select Department

Select User Role

Figure 5.8: Create Crew Page

On this page, admin can create account for crews other than crews registered themselves.

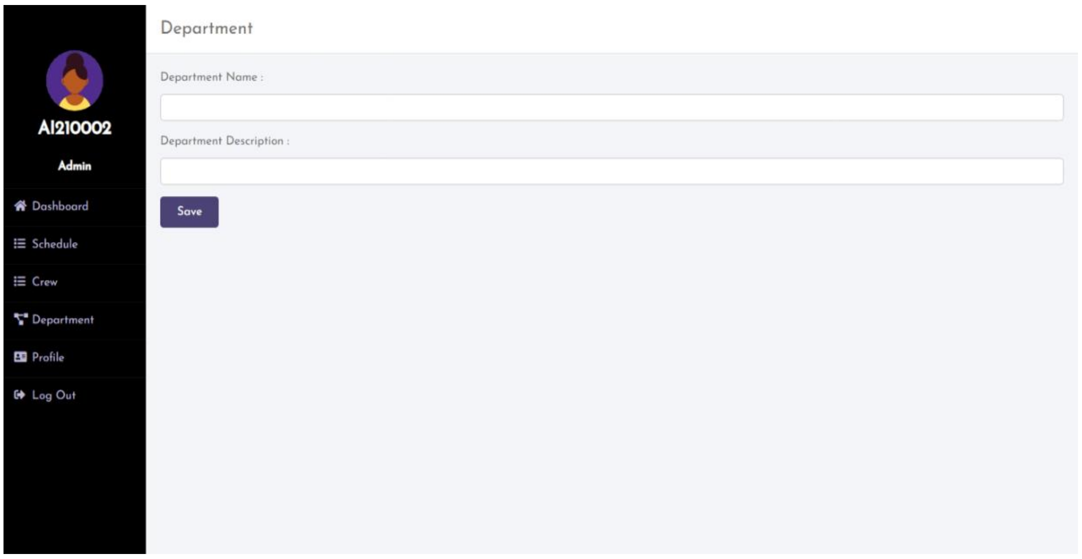


View Crew

No	Crew ID	Full Name	Username	Department	User Role	Edit
1	adminBunga	BungaAdmin	root	Department1	Admin	Edit
2	ai210002	vinisha	root	Department3	Admin	Edit
3	ai210003	hossini	root	Department1	Staff	Edit

Figure 5.9: View crew page

The admin can view the crew that registered by themselves. They also can edit the registered crew’s account.



Department

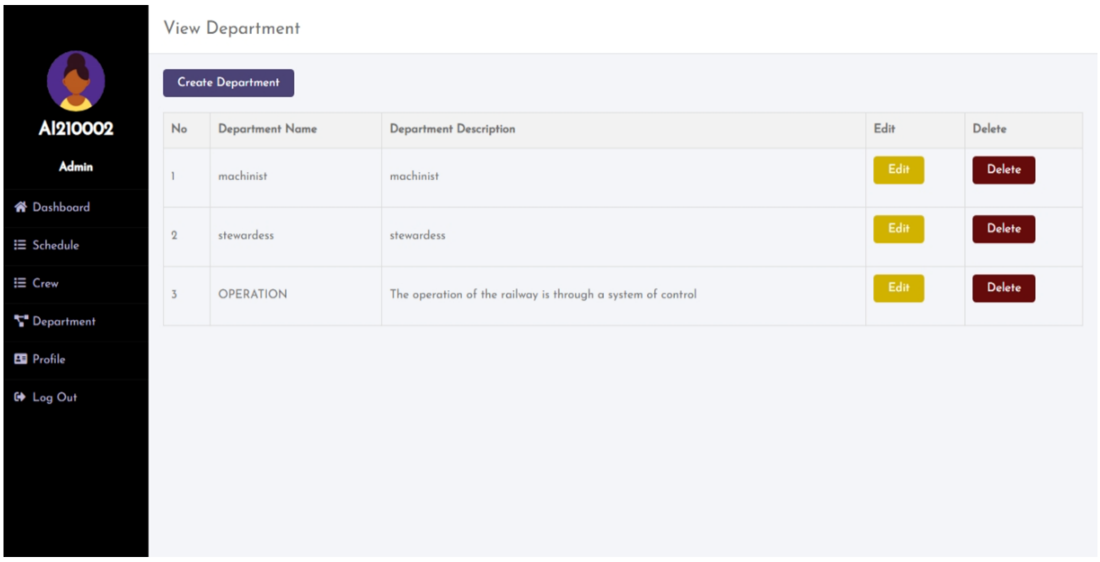
Department Name :

Department Description :

Save

Figure 5.10: Create department page

This is a create department page of our system. The admin can create the department by entering the department name and department description.

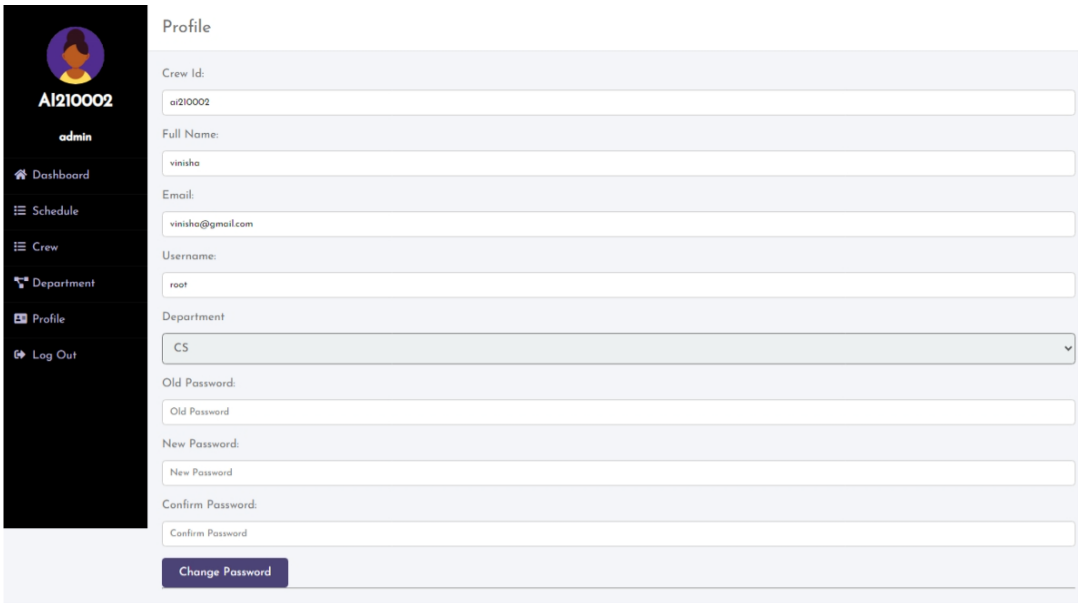


View Department

No	Department Name	Department Description	Edit	Delete
1	machinist	machinist	Edit	Delete
2	stewardess	stewardess	Edit	Delete
3	OPERATION	The operation of the railway is through a system of control	Edit	Delete

Figure 5.11: View department page

This is a view department page where the admin can view the department that they created by themselves. They also can edit the department again and delete the department if not necessary.



Profile

Crew Id: ai210002

Full Name: vinisha

Email: vinisha@gmail.com

Username: root

Department: CS

Old Password: Old Password

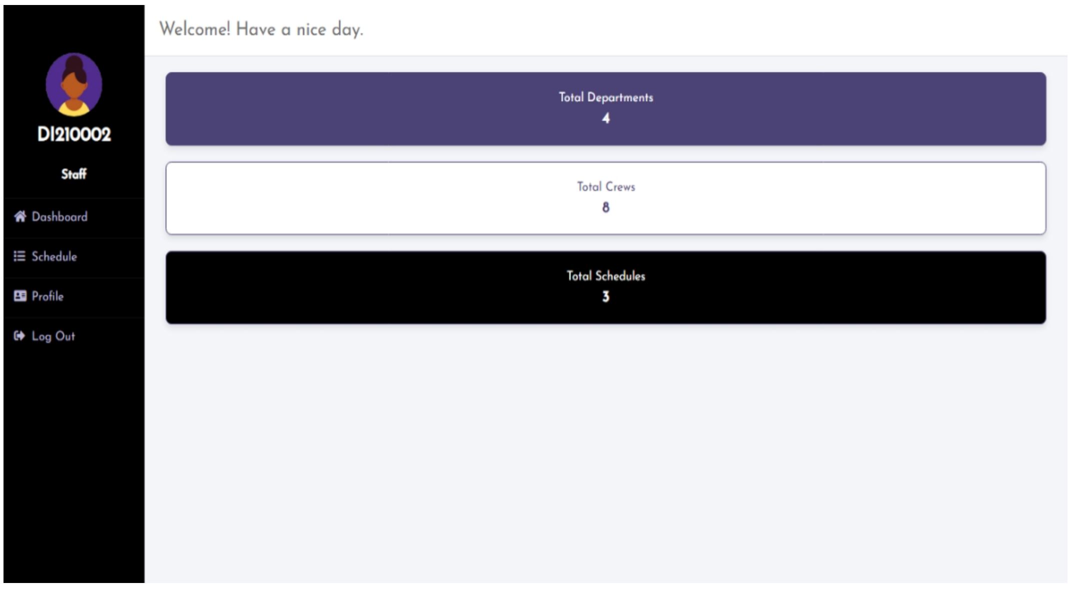
New Password: New Password

Confirm Password: Confirm Password

Figure 5.12: Profile admin

On this page, the admin can update their profile.

5.3 Crew Side



Welcome! Have a nice day.

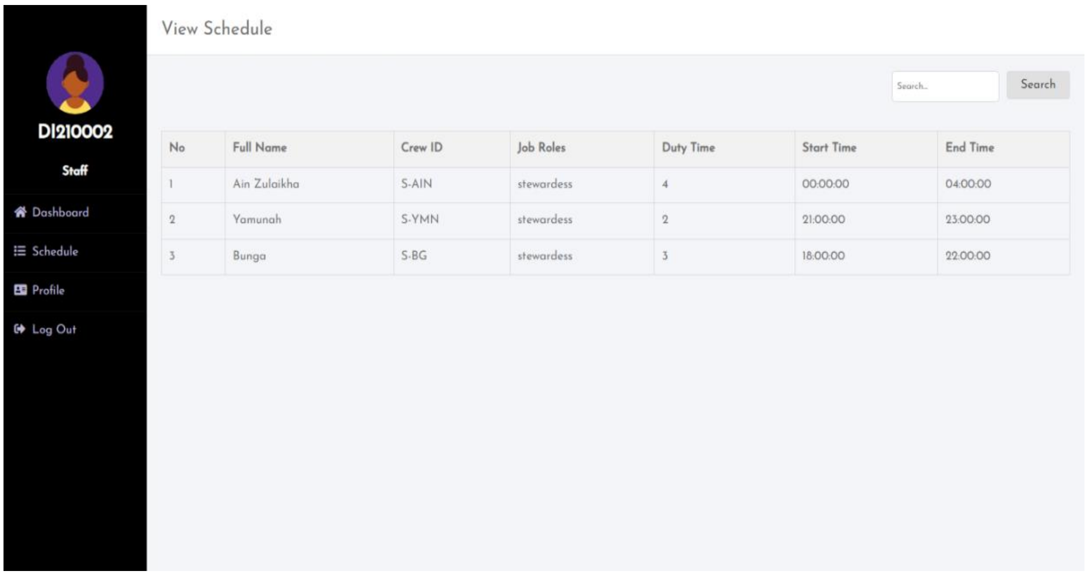
Total Departments
4

Total Crews
8

Total Schedules
3

Figure 5.13: Dashboard Crew Page

For Dashboard Crew Page, user can view total department, crews, and total schedule in the system. Users can refer to their schedule by going to the schedule page category.

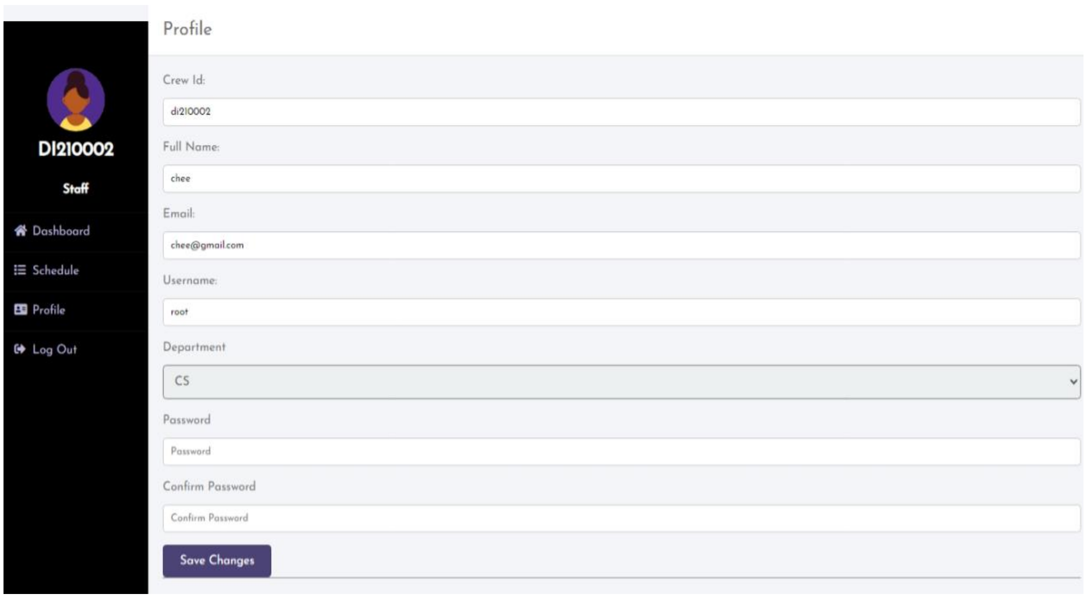


View Schedule

No	Full Name	Crew ID	Job Roles	Duty Time	Start Time	End Time
1	Ain Zulaikha	S-AIN	stewardess	4	00:00:00	04:00:00
2	Yamunah	S-YMN	stewardess	2	21:00:00	23:00:00
3	Bunga	S-BG	stewardess	3	18:00:00	22:00:00

Figure 5.14: View schedule page

On this page, users can view the schedule based on their full name and crew id.



Profile

Crew Id: di210002

Full Name: chee

Email: chee@gmail.com

Username: root

Department: CS

Password: Password

Confirm Password: Confirm Password

Figure 5.15: Profile crew

For the Profile crew page, users can edit their profile such as their full name, username, and password.

References

1) Garrisi, G., & Cervelló-Pastor, C. (2020). Train-scheduling optimization model for railway networks with multiplatform stations. Sustainability (Switzerland), 12(1), 1–25. https://doi.org/10.3390/su12010257
2) Hanczar, P., & Zandi, A. (2021). A novel model and solution algorithm to improve crew scheduling in railway transportation: A real world case study. Computers and Industrial Engineering, 154(December 2020). https://doi.org/10.1016/j.cie.2021.107132
3) Heil, J., Hoffmann, K., & Buscher, U. (2020). Railway crew scheduling: Models, methods and applications. European Journal of Operational Research, 283(2), 405–425. https://doi.org/10.1016/j.ejor.2019.06.016
4) Smith, R. (2018). The Role of Railway Transportation in Modern Society. Journal of Transportation Studies, 12(3), 221-238.
5) Mohammed Moseeur Rahman Assistant Professor, C. (n.d.). *Railway Online Booking System Design and Implementation*.
6) Thaduri, A., Aljumaili, M., Kour, R., & Karim, R. (2019). Cybersecurity for eMaintenance in railway infrastructure: risks and consequences. In *International Journal of System Assurance Engineering and Management* (Vol. 10, Issue 2, pp. 149–159). Springer. https://doi.org/10.1007/s13198-019-00778-w
7) *Home - MRT Corp*. (n.d.). Retrieved December 31, 2023, from https://www.mymrt.com.my/
8) *Official website for Rapid KL, Rapid Penang and Rapid Kuantan by Prasarana Malaysia Berhad*. (n.d.). Retrieved December 31, 2023, from https://myrapid.com.my/
9) W3Schools. (n.d.). PHP Tutorial. Retrieved from https://www.w3schools.com/php/
10) Keretapi Tanah Melayu Berhad (KTMB).Maps and Route. Retrieved from https://www.ktmb.com.my/MapsAndRoute.html
11) FreeCodeCamp. Basic SQL Commands. Retrieved from https://www.freecodecamp.org/news/basic-sql-commands/
12) Codecademy. (n.d.). SQL Commands. Retrieved from https://www.codecademy.com/article/sql-commands
13) Tutorialspoint. (n.d.). PHP MySQL Login. Retrieved from https://www.tutorialspoint.com/php/php_mysql_login.htm
14) GeeksforGeeks.PHP Database Connection. Retrieved from https://www.geeksforgeeks.org/php-database-connection/
15) Simplilearn. (n.d.). PHP Login Form. Retrieved from https://www.simplilearn.com/tutorials/php-tutorial/php-login-form

16) PHP Group. (n.d.). Filter Validation Examples. Retrieved from https://www.php.net/manual/en/filter.examples.validation.php
17) Cloudways. (n.d.). How to Connect MySQL Database with PHP. Retrieved from https://www.cloudways.com/blog/connect-mysql-with-php/
18) Position Is Everything. (n.d.). PHP Validation Email. Retrieved from https://www.positioniseverything.net/php-validation-email/
19) Tutorial Republic. (n.d.). PHP MySQL Login System. Retrieved from https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php
20) Keretapi Tanah Melayu Berhad (KTMB). Retrieved from https://www.ktmb.com.my/
21) *Home - MRT Corp*. (n.d.). Retrieved December 31, 2023, from https://www.mymrt.com.my/
22) Mohammed Moseeur Rahman Assistant Professor, C. (n.d.). *Railway Online Booking System Design and Implementation*.
23) *Official website for Rapid KL, Rapid Penang and Rapid Kuantan by Prasarana Malaysia Berhad*. (n.d.). Retrieved December 31, 2023, from https://myrapid.com.my/
24) *Sequence diagrams - IBM Documentation*. (n.d.). Retrieved December 27, 2023, from https://www.ibm.com/docs/en/rsm/7.5.0?topic=uml-sequence-diagrams
25) Thaduri, A., Aljumaili, M., Kour, R., & Karim, R. (2019). Cybersecurity for eMaintenance in railway infrastructure: risks and consequences. In *International Journal of System Assurance Engineering and Management* (Vol. 10, Issue 2, pp. 149–159). Springer. https://doi.org/10.1007/s13198-019-00778-w
26) *Use Case Diagrams - Use Case Diagrams Online, Examples, and Tools*. (n.d.). Retrieved December 27, 2023, from https://www.smartdraw.com/use-case-diagram/
27) *What is Unified Modeling Language (UML)?* (n.d.). Retrieved December 27, 2023, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/