CRITERION C: DEVELOPMENT

TECHNIQUES USED

1. Exception handling to ensure the proper inputs are given

2. HTML and CSS for designing and styling the user interface

3. JavaScript to develop interactive front-end interfaces

4. Dynamic HTML content manipulation

5. MySQL database through xampp to store and manipulate data

6. PHP for backend scripting to facilitate interaction with the database

7. PHPMailer module to send emails

8. Bcrypt algorithm to hash passwords

9. Chart.js to dynamically generate and stylize bar graphs

10. Google Fonts for access to specific typographies

11. TCPDF to generate PDF documents

AREAS OF COMPLEXITY

1. account password is stored in hash format

2. Automatic email notifications will be sent to inform users of any changes in their library account.

3. Use of parameterized queries to sanitize user input to protect against SQL injection attacks on input data when accessing the account database

4. Setting a time-limited 6-digit OTP linked to the IP address of the user's device

5. Storing variables to be accessible across multiple pages

6. Enabling a configurable tabbed interface on the librarian page to facilitate the

simultaneous and independent operation of two modules

7. Implemented a book search feature that is capable of handling partial search requests

8. Allows dates to be entered in the order date range

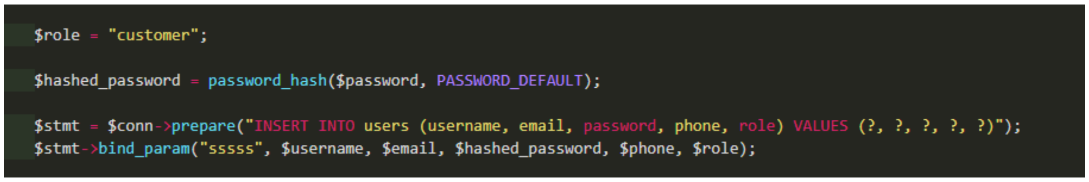
9. Export statistical data as PDF documents

10. Scheduled time for the program to automatically update room data if it is checkout time for

EXPLANATION OF USE OF COMPLEX TECHNIQUES

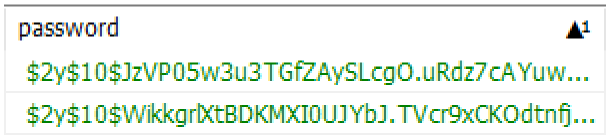
1. Account passwords are stored in a hashed format

To ensure the storage of each account's email, username, password and email, when registration is performed, the system uses the Brcypt function to generate a fixed-size character string based on the entered password. The hash is then stored in the database, while the original password is discarded. This ensures that even though the database an attacker cannot retrieve the original password because hashing is a one-way process.



```
$role = "customer";

$hashed_password = password_hash($password, PASSWORD_DEFAULT);

$stmt = $conn->prepare("INSERT INTO users (username, email, password, phone, role) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param("sssss", $username, $email, $hashed_password, $phone, $role);
```

Fig.1 1Code Snipset Password hashed and Insert data

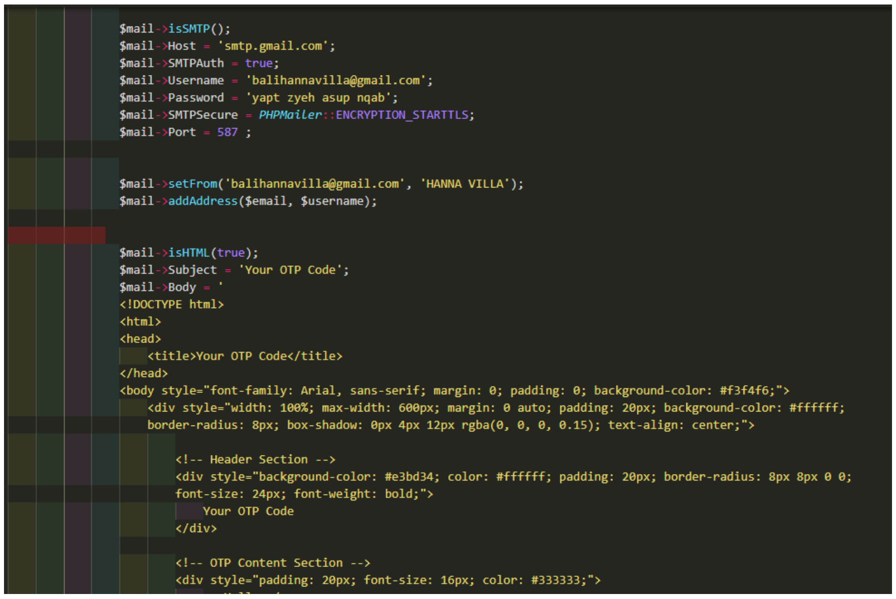


password
$2y$10$JzVP05w3u3TGfZAySLcgO.uRdz7cAYuw...
$2y$10$WikkgrlXtBDKMXI0UJYbJ.TVcr9xCKOdtnfj...

Fig.1 2 Password hashed

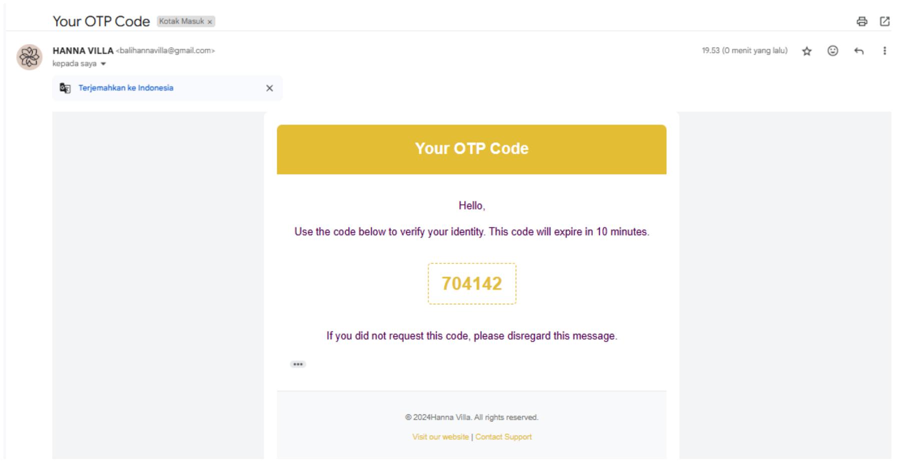
2. An automated email notification will be sent to notify users of account otp verification for signup

To verify their account, PHPMailer is used to send an emailevery time an account is registered as well as when the account password is changed. This is confirmation helps ensure that users are aware of the status of their accounts, which increases security as users are can identify any unauthorized changes to their account, so that they can take action, such as resetting their password or contacting the school admin. password



```
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'balihannavilla@gmail.com';
$mail->Password = 'yapt zyeh asup nqab';
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port = 587 ;


$mail->setFrom('balihannavilla@gmail.com', 'HANNA VILLA');
$mail->addAddress($email, $username);


$mail->isHTML(true);
$mail->Subject = 'Your OTP Code';
$mail->Body = '
<!DOCTYPE html>
<html>
<head>
    <title>Your OTP Code</title>
</head>
<body style="font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f3f4f6;">
    <div style="width: 100%; max-width: 600px; margin: 0 auto; padding: 20px; background-color: #ffffff;
    border-radius: 8px; box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.15); text-align: center;">

        <!-- Header Section -->
        <div style="background-color: #e3bd34; color: #ffffff; padding: 20px; border-radius: 8px 8px 0 0;
        font-size: 24px; font-weight: bold;">
            Your OTP Code
        </div>

        <!-- OTP Content Section -->
        <div style="padding: 20px; font-size: 16px; color: #333333;">
```

Fig.2 1 Code Snippet Send OTP to Email



Your OTP Code

HANNA VILLA <balihannavilla@gmail.com>
kepada saya

19.53 (0 menit yang lalu)

Your OTP Code

Hello,

Use the code below to verify your identity. This code will expire in 10 minutes.

704142

If you did not request this code, please disregard this message.

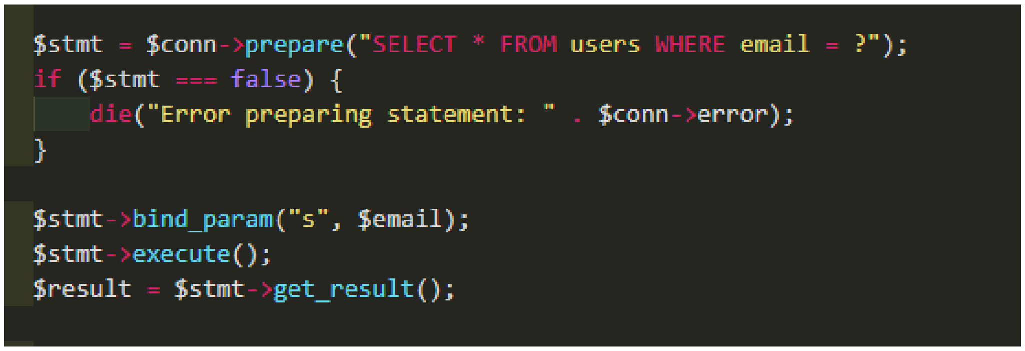
© 2024Hanna Villa. All rights reserved.

Visit our website | Contact Support

Fig.2 2 otp received via email

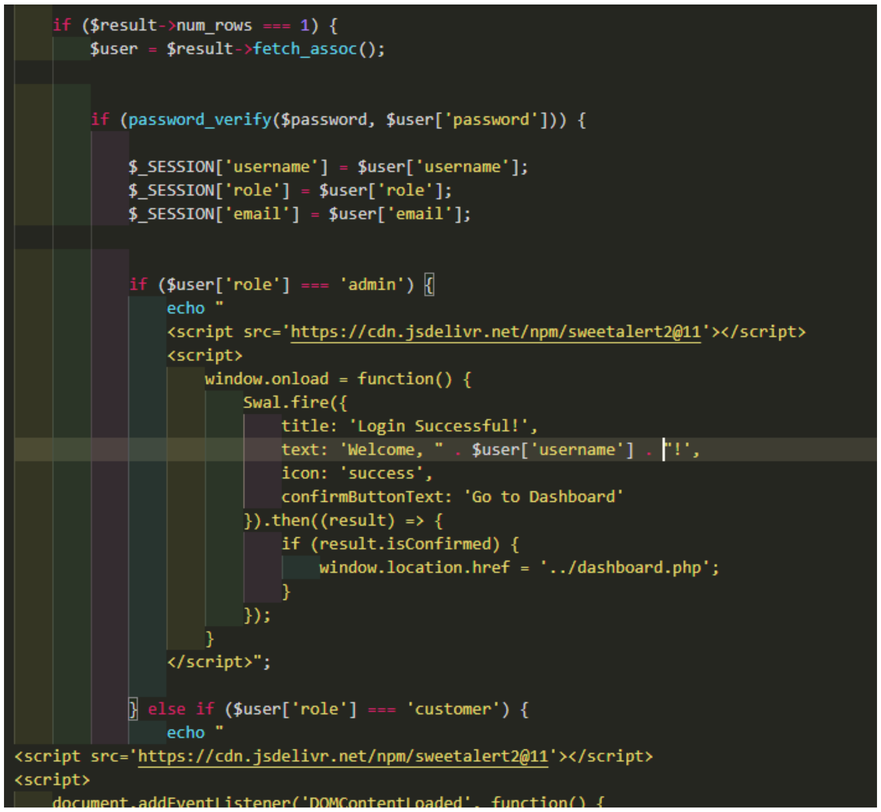
3. Use of parameterized queries to sanitize user input to protect against SQL injection attacks on input data when accessing the account database

When using SQL servers, a common security vulnerability is SQL injection, wherein the attacker can manipulate the input queries such that it allows them to gain unauthorized access to the database. Thus, if the system always accepts the user’s input as is, the attacker can then submit input(s) that contains SQL code that can execute commands to bypass authentication. Thus, sanitizing the input is needed at every stage when a user's input is taken, which would affect the account databases.



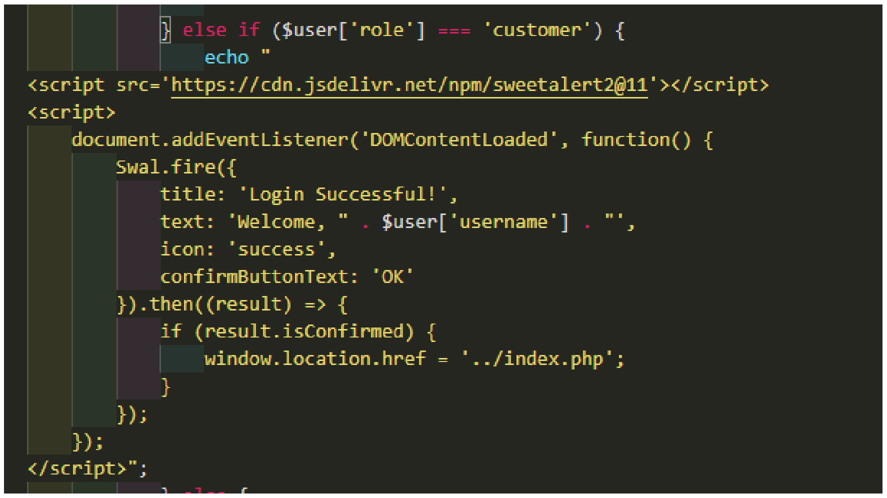
```php
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");
if ($stmt === false) {
    die("Error preparing statement: " . $conn->error);
}

$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();
```

Fig.3.1 Code Snippet check if the email is available in the database



```php
    if ($result->num_rows === 1) {
        $user = $result->fetch_assoc();

        if (password_verify($password, $user['password'])) {

            $_SESSION['username'] = $user['username'];
            $_SESSION['role'] = $user['role'];
            $_SESSION['email'] = $user['email'];

            if ($user['role'] === 'admin') {
                echo "
                <script src='https://cdn.jsdelivr.net/npm/sweetalert2@11'></script>
                <script>
                    window.onload = function() {
                        Swal.fire({
                            title: 'Login Successful!',
                            text: 'Welcome, " . $user['username'] . "!',
                            icon: 'success',
                            confirmButtonText: 'Go to Dashboard'
                        }).then((result) => {
                            if (result.isConfirmed) {
                                window.location.href = '../dashboard.php';
                            }
                        });
                    }
                </script>";

            } else if ($user['role'] === 'customer') {
                echo "
<script src='https://cdn.jsdelivr.net/npm/sweetalert2@11'></script>
<script>
    document.addEventListener('DOMContentLoaded', function() {
```

Fig.3.2 Code Snippet Checks whether the hashed password is the same as the email and exists in the database.



```
            } else if ($user['role'] === 'customer') {
                echo "
<script src='https://cdn.jsdelivr.net/npm/sweetalert2@11'></script>
<script>
    document.addEventListener('DOMContentLoaded', function() {
        Swal.fire({
            title: 'Login Successful!',
            text: 'Welcome, " . $user['username'] . "',
            icon: 'success',
            confirmButtonText: 'OK'
        }).then((result) => {
            if (result.isConfirmed) {
                window.location.href = '../index.php';
            }
        });
    });
</script>";
```

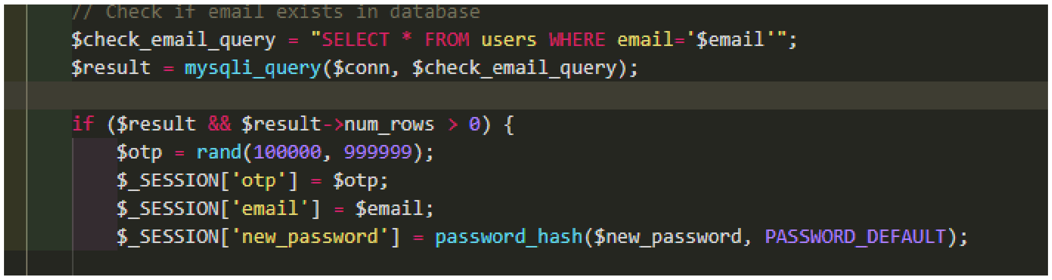
Fig.3.3 Code Snippet checks the role of the logged-in user

This code handles login authentication using PHP. Here's the step-by-step explanation:

1. **Login Data Validation:**
    - The code checks if the $result has exactly one row ($result->num_rows === 1). If true, it means the login data matches a user in the database.
2. **Password Verification:**
    - It uses password_verify($password, $user['password']) to compare the entered password with the stored (hashed) password in the database.
3. **Storing Information in $_SESSION:**
    - On successful login, user details such as username, role, and email are stored in the session.
4. **Redirection Based on Role:**
    - If the user's role is admin, they are redirected to the dashboard page (../dashboard.php) after a SweetAlert notification.
    - If the role is customer, they are redirected to the main page (../index.php) with a different SweetAlert message.
5. **Successful Login Notification:**
    - SweetAlert is used to display interactive and aesthetically pleasing messages after successful login. It uses two approaches:
        - window.onload (for admin).
        - document.addEventListener('DOMContentLoaded' (for customer).

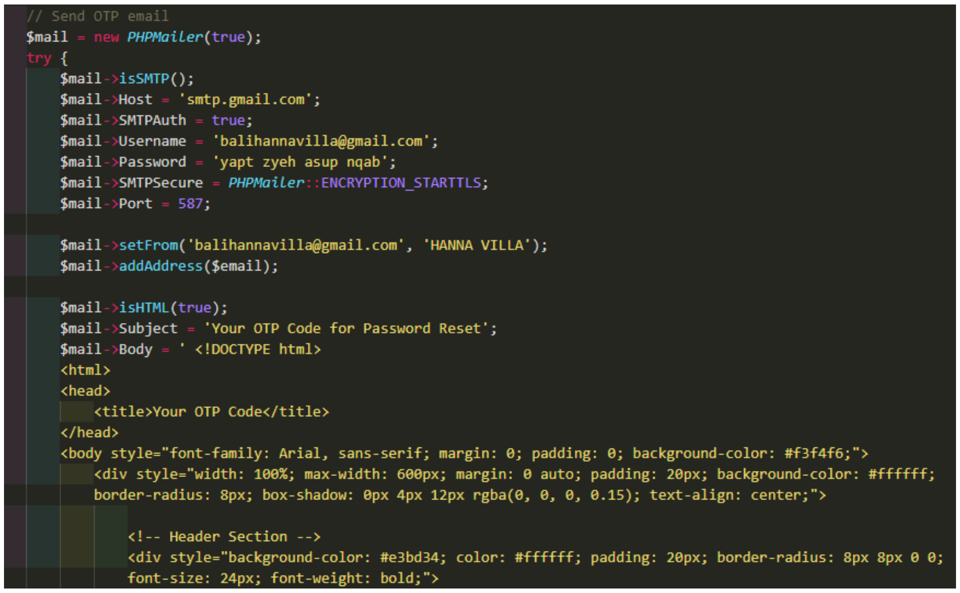
4. Checking email for forgotten password and sending otp code to registered email.

To avoid data leakage, users who forget their passwords can change or update their passwords and check the accounts registered in the database.



```php
// Check if email exists in database
$check_email_query = "SELECT * FROM users WHERE email='$email'";
$result = mysqli_query($conn, $check_email_query);

if ($result && $result->num_rows > 0) {
    $otp = rand(100000, 999999);
    $_SESSION['otp'] = $otp;
    $_SESSION['email'] = $email;
    $_SESSION['new_password'] = password_hash($new_password, PASSWORD_DEFAULT);
```

Fig.4.1 Code Snippet Checks if the email is registered and generates a random otp code.



```php
// Send OTP email
$mail = new PHPMailer(true);
try {
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'balihannavilla@gmail.com';
    $mail->Password = 'yapt zyeh asup nqab';
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port = 587;

    $mail->setFrom('balihannavilla@gmail.com', 'HANNA VILLA');
    $mail->addAddress($email);

    $mail->isHTML(true);
    $mail->Subject = 'Your OTP Code for Password Reset';
    $mail->Body = ' <!DOCTYPE html>
    <html>
    <head>
        <title>Your OTP Code</title>
    </head>
    <body style="font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f3f4f6;">
        <div style="width: 100%; max-width: 600px; margin: 0 auto; padding: 20px; background-color: #ffffff;
        border-radius: 8px; box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.15); text-align: center;">

            <!-- Header Section -->
            <div style="background-color: #e3bd34; color: #ffffff; padding: 20px; border-radius: 8px 8px 0 0;
            font-size: 24px; font-weight: bold;">
```

Fig.4 2 Code Snippet create an html tag design message to send to email

The PHP code checks if a user's email exists in the database and, if found, generates a One-Time Password (OTP) for further verification, such as password reset. The steps include:

Email Validation:

A database query checks whether the given email exists in the users table. If a match is found, the process continues.

OTP Generation:

A 6-digit random OTP is created using the rand() function, typically used for temporary authentication purposes.

Session Storage:

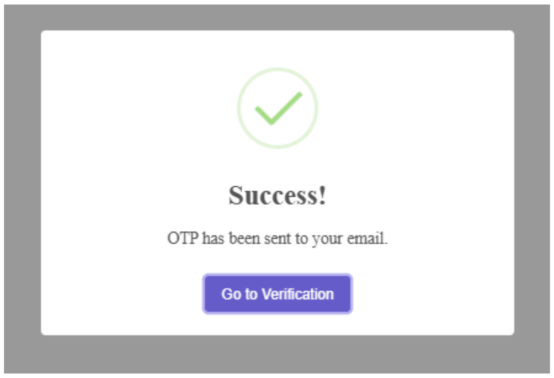
The OTP, user's email, and a hashed version of the new password are stored in the session for subsequent steps in the process.

Security Measures:

Passwords are hashed using password_hash() to enhance security.

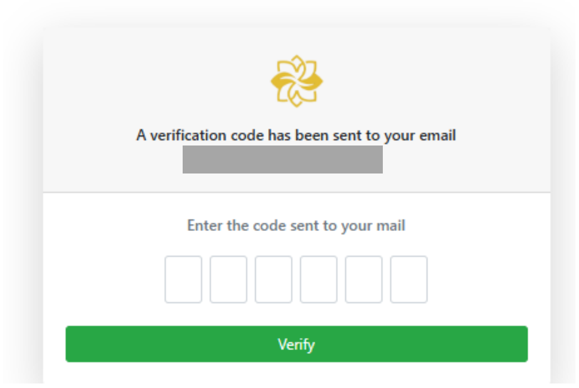
The query is vulnerable to SQL Injection and should use prepared statements for better protection.

This code is often used in password recovery workflows, ensuring that only authorized users can proceed with resetting their passwords.


Success!

OTP has been sent to your email.

Fig.4.3 if the otp code is successfully sent


A verification code has been sent to your email

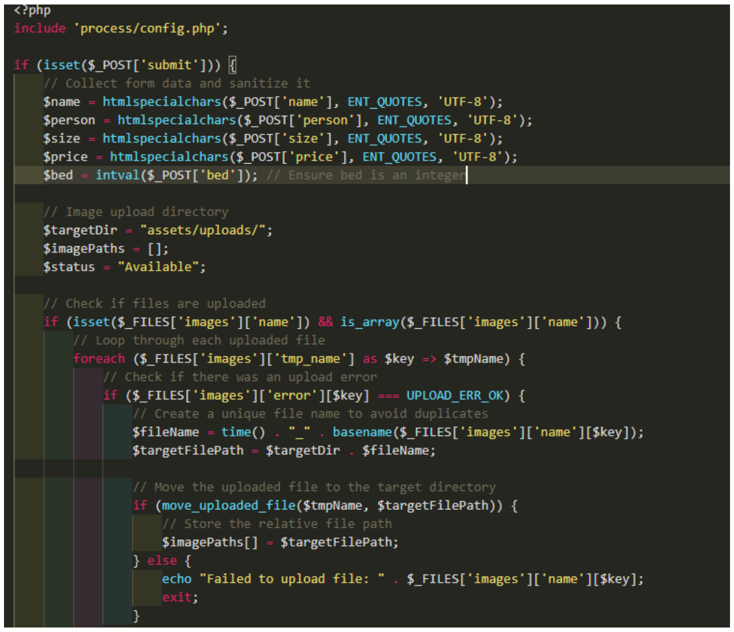
[redacted]

Enter the code sent to your mail

Fig.4.4 otp code entry page

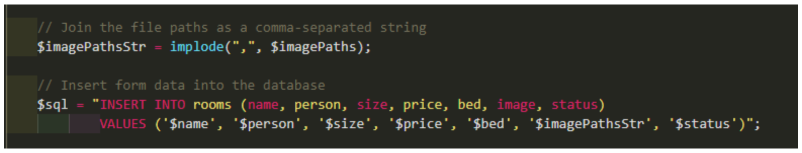
5. Adding rooms to book

Room data can be added edited and even deleted and displayed on the admin dashboard page in tabular form.



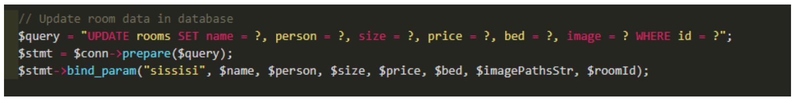
```php
<?php
include 'process/config.php';

if (isset($_POST['submit'])) {
    // Collect form data and sanitize it
    $name = htmlspecialchars($_POST['name'], ENT_QUOTES, 'UTF-8');
    $person = htmlspecialchars($_POST['person'], ENT_QUOTES, 'UTF-8');
    $size = htmlspecialchars($_POST['size'], ENT_QUOTES, 'UTF-8');
    $price = htmlspecialchars($_POST['price'], ENT_QUOTES, 'UTF-8');
    $bed = intval($_POST['bed']); // Ensure bed is an integer

    // Image upload directory
    $targetDir = "assets/uploads/";
    $imagePaths = [];
    $status = "Available";

    // Check if files are uploaded
    if (isset($_FILES['images']['name']) && is_array($_FILES['images']['name'])) {
        // Loop through each uploaded file
        foreach ($_FILES['images']['tmp_name'] as $key => $tmpName) {
            // Check if there was an upload error
            if ($_FILES['images']['error'][$key] === UPLOAD_ERR_OK) {
                // Create a unique file name to avoid duplicates
                $fileName = time() . "_" . basename($_FILES['images']['name'][$key]);
                $targetFilePath = $targetDir . $fileName;

                // Move the uploaded file to the target directory
                if (move_uploaded_file($tmpName, $targetFilePath)) {
                    // Store the relative file path
                    $imagePaths[] = $targetFilePath;
                } else {
                    echo "Failed to upload file: " . $_FILES['images']['name'][$key];
                    exit;
                }
```

Fig.5.1 Code Snippet make room additions



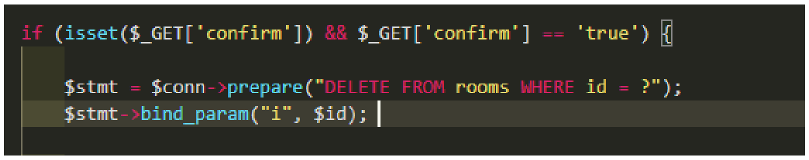
```php
// Join the file paths as a comma-separated string
$imagePathsStr = implode(",", $imagePaths);

// Insert form data into the database
$sql = "INSERT INTO rooms (name, person, size, price, bed, image, status)
        VALUES ('$name', '$person', '$size', '$price', '$bed', '$imagePathsStr', '$status')";
```

Fig.5.2 Code Snippet enter room data into the database



```php
// Update room data in database
$query = "UPDATE rooms SET name = ?, person = ?, size = ?, price = ?, bed = ?, image = ? WHERE id = ?";
$stmt = $conn->prepare($query);
$stmt->bind_param("sissisi", $name, $person, $size, $price, $bed, $imagePathsStr, $roomId);
```

Fig.5.3 Code Snippet update room data



```php
if (isset($_GET['confirm']) && $_GET['confirm'] == 'true') {

    $stmt = $conn->prepare("DELETE FROM rooms WHERE id = ?");
    $stmt->bind_param("i", $id);
```

Fig.5.4 Code Sinppet Delete room

This PHP script is designed to process a form submission for adding room details into a hotel management system. It handles both textual data (room information) and file uploads (room images). Here's a summary of the process:

Data Collection and Sanitization:

- The script collects room details from a form submission (name, person, size, price, bed).
- Sanitization is applied using htmlspecialchars() to prevent XSS attacks, and intval() ensures numeric inputs like bed are valid integers.

Image Upload:

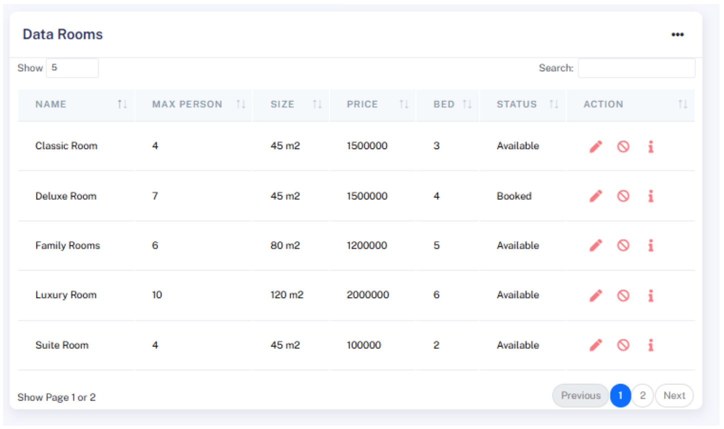
- The script allows multiple images to be uploaded for each room.
- It checks for errors in file uploads and assigns unique filenames to avoid overwriting existing files.
- Successfully uploaded images are moved to the assets/uploads/ directory, and their paths are stored in an array.

Database Insertion:

- All collected data, including a comma-separated list of image file paths, is inserted into the rooms table in the database.
- The room is also given a default status of "Available."

Error Handling:

- If no images are uploaded or a file fails to upload, the script displays an appropriate error and stops execution.

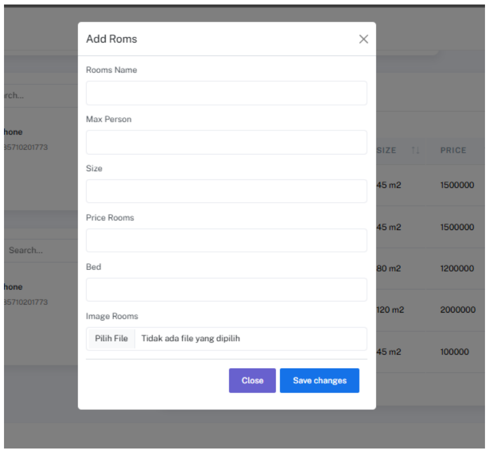


Data Rooms

Show 5 Search:

NAME	MAX PERSON	SIZE	PRICE	BED	STATUS	ACTION
Classic Room	4	45 m2	1500000	3	Available	
Deluxe Room	7	45 m2	1500000	4	Booked	
Family Rooms	6	80 m2	1200000	5	Available	
Luxury Room	10	120 m2	2000000	6	Available	
Suite Room	4	45 m2	100000	2	Available	

Show Page 1 or 2



Add Roms

Rooms Name

Max Person

Size

Price Rooms

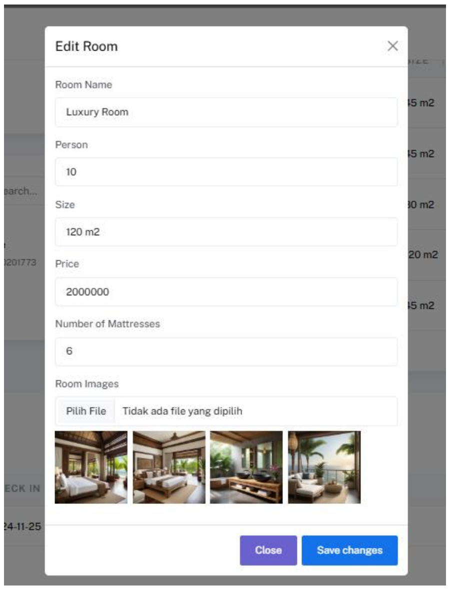
Bed

Image Rooms

Pilih File Tidak ada file yang dipilih

Fig.5.5 displays room data in the table

Fig.5.6 adding a room



Edit Room

Room Name: Luxury Room

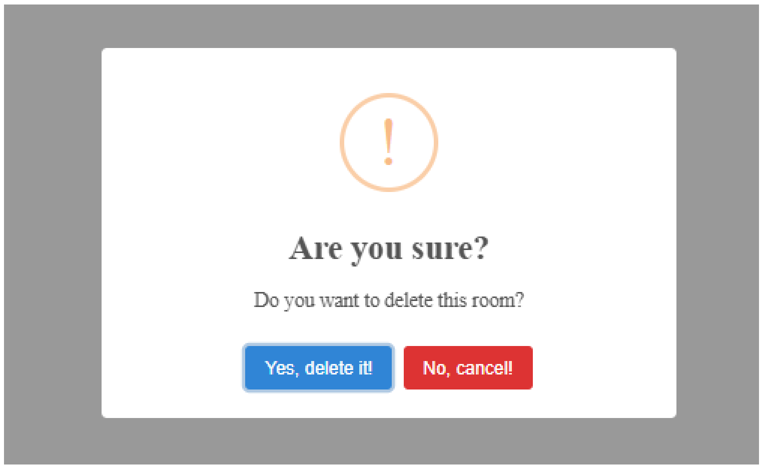
Person: 10

Size: 120 m2

Price: 2000000

Number of Mattresses: 6

Room Images: Pilih File Tidak ada file yang dipilih



Are you sure?

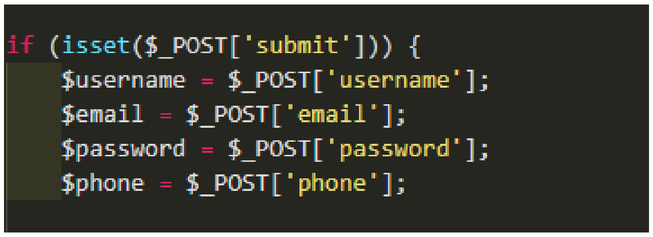
Do you want to delete this room?

Fig.5.7 edit the room

Fig.5.8 Delet room

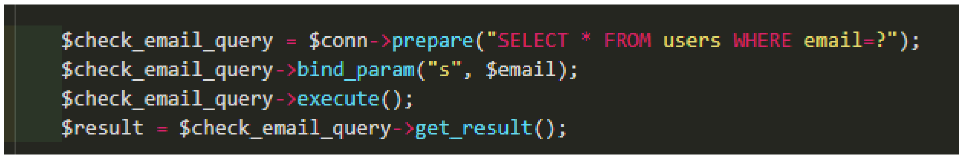
6. add users for admins who do not need to register

add an admin account to add more than one user only



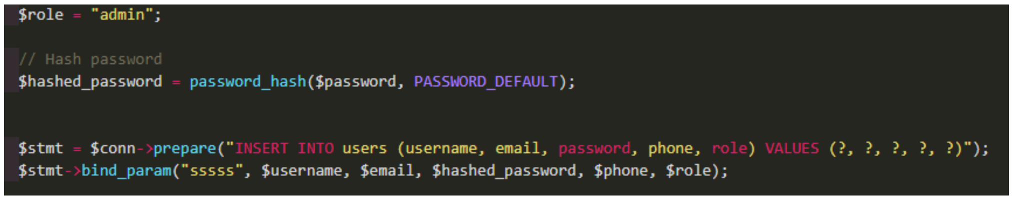
```php
if (isset($_POST['submit'])) {
    $username = $_POST['username'];
    $email = $_POST['email'];
    $password = $_POST['password'];
    $phone = $_POST['phone'];
```

Fig 6.1 Code Snippet retrieve data from the form



```php
    $check_email_query = $conn->prepare("SELECT * FROM users WHERE email=?");
    $check_email_query->bind_param("s", $email);
    $check_email_query->execute();
    $result = $check_email_query->get_result();
```

Fig 6.2 Code Snippet Check if the previous email has been registered



```php
$role = "admin";

// Hash password
$hashed_password = password_hash($password, PASSWORD_DEFAULT);


$stmt = $conn->prepare("INSERT INTO users (username, email, password, phone, role) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param("sssss", $username, $email, $hashed_password, $phone, $role);
```

Fig 6.3 Code Snipet hashes the password and stores it in the database

This PHP script facilitates the secure and user-friendly registration of new users. It ensures that user inputs are validated, provides real-time feedback using SweetAlert2, and securely stores user data in the database. Below are the detailed steps:

A. Input Collection

- The script collects form data submitted via the POST method: Fields: username, email, password, and phone.

B. Input Validation The script performs several validation checks:

- Empty Fields: Ensures all fields are filled. If any are empty, a warning message is displayed via SweetAlert2.
- Email Format: Validates the email using filter_var() with FILTER_VALIDATE_EMAIL. If the email is invalid, an error message is shown.
- Phone Number: Checks the phone number using a regex pattern to ensure it:

- Contains only numeric characters.
- Has a length between 10-15 characters.
- If invalid, an error message is displayed.

C. Email Uniqueness Check The script checks if the email already exists in the database:

- Executes a SELECT query using prepared statements to avoid SQL injection.
- If the email is already registered, a warning is displayed to the user.

D. Password Hashing If all validations pass and the email is unique:

- The password is hashed using password_hash() with PASSWORD_DEFAULT to ensure secure storage.

E. Inserting User Data into the Database A prepared INSERT query adds the following data to the users table:

- Fields: username, email, hashed password, phone, and a hardcoded role value (admin).
- Feedback:
- On success, a success message is shown using SweetAlert2.
- On failure, an error message indicates that the user registration failed.

F. Interactive Feedback with SweetAlert2 Uses SweetAlert2 to display user-friendly feedback messages:

- Success, warning, or error messages appear dynamically.
- Includes a return action (window.history.back()) to navigate back if needed.

G. Security Measures Prepared Statements:

- Protects against SQL injection when executing database queries.
- Password Hashing: Ensures that stored passwords cannot be reversed or directly retrieved.

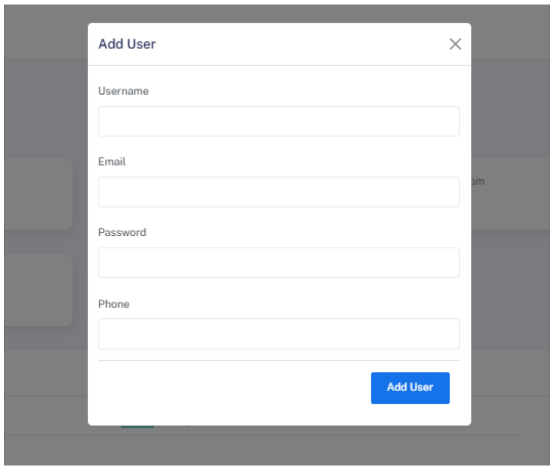
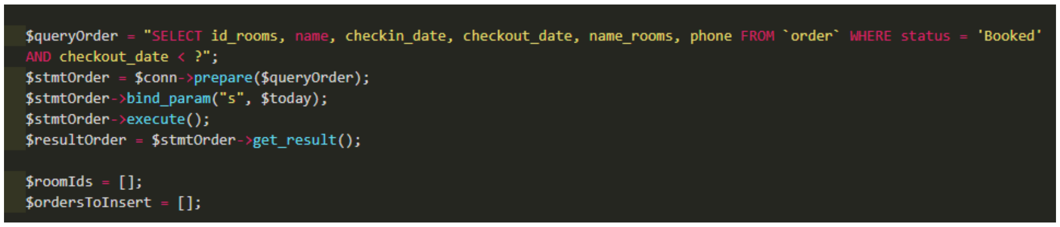


Fig 6.4 Form add user

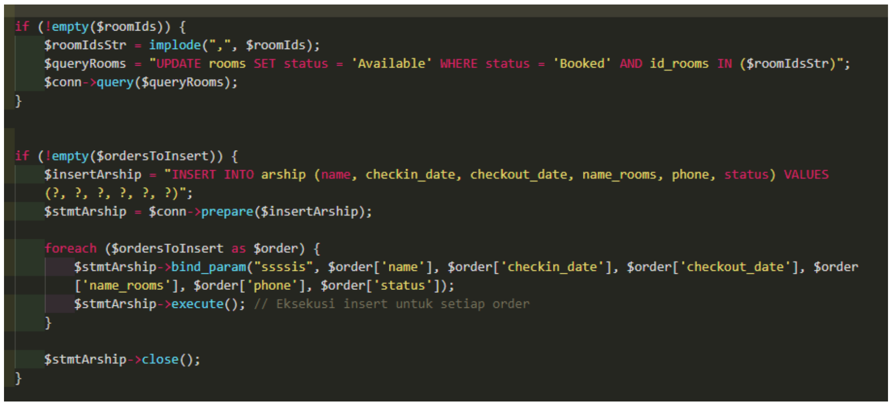
7. performs automatic status updates on rooms that are due for checkout.

This php is made to automatically update the status of the room and store order history data on the ship page.



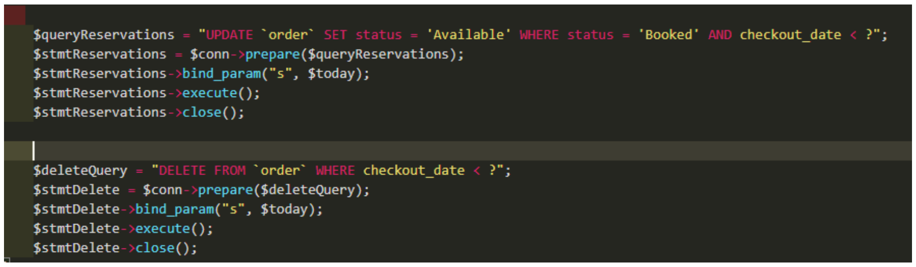
```
$queryOrder = "SELECT id_rooms, name, checkin_date, checkout_date, name_rooms, phone FROM `order` WHERE status = 'Booked'
AND checkout_date < ?";
$stmtOrder = $conn->prepare($queryOrder);
$stmtOrder->bind_param("s", $today);
$stmtOrder->execute();
$resultOrder = $stmtOrder->get_result();

$roomIds = [];
$ordersToInsert = [];
```

Fig 7.1 Code Snippet to check checkout time with room booked status



```
if (!empty($roomIds)) {
    $roomIdsStr = implode(",", $roomIds);
    $queryRooms = "UPDATE rooms SET status = 'Available' WHERE status = 'Booked' AND id_rooms IN ($roomIdsStr)";
    $conn->query($queryRooms);
}


if (!empty($ordersToInsert)) {
    $insertArship = "INSERT INTO arship (name, checkin_date, checkout_date, name_rooms, phone, status) VALUES
    (?, ?, ?, ?, ?, ?)";
    $stmtArship = $conn->prepare($insertArship);

    foreach ($ordersToInsert as $order) {
        $stmtArship->bind_param("ssssis", $order['name'], $order['checkin_date'], $order['checkout_date'], $order
        ['name_rooms'], $order['phone'], $order['status']);
        $stmtArship->execute(); // Eksekusi insert untuk setiap order
    }

    $stmtArship->close();
}
```

Fig 7.2 Code Snippet update the status of the room and enter the completed order into arship.



```
$queryReservations = "UPDATE `order` SET status = 'Available' WHERE status = 'Booked' AND checkout_date < ?";
$stmtReservations = $conn->prepare($queryReservations);
$stmtReservations->bind_param("s", $today);
$stmtReservations->execute();
$stmtReservations->close();

$deleteQuery = "DELETE FROM `order` WHERE checkout_date < ?";
$stmtDelete = $conn->prepare($deleteQuery);
$stmtDelete->bind_param("s", $today);
$stmtDelete->execute();
$stmtDelete->close();
```

Fig 7.3 Code Snippet Delete order data that has passed the checkout time

Retrieve Bookings that have expired:

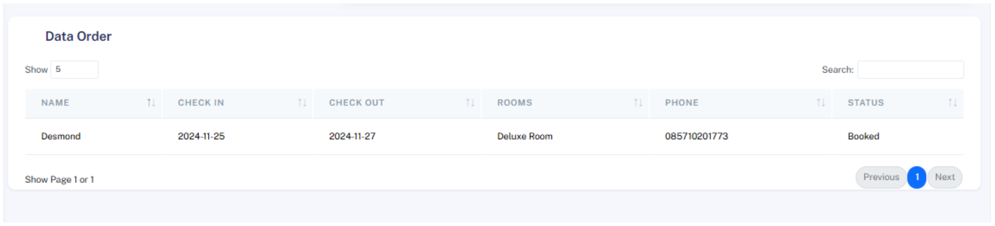
- Retrieve all bookings from the bookings table whose status is “Booked” but whose payment date has passed (checkout_date < $today).
- This function stores the retrieved data in two arrays:

- $roomIds: Contains the IDs of rooms with expired bookings.
- $ordersToInsert: Prepares data to archive expired bookings.
- Update the Room Status to “Available”:

If there are expired bookings, this will update the rooms table, setting the corresponding room status from “Booked” to “Available”.
Archive Expired Bookings:

Inserts the expired booking details into the arship table for historical record keeping.

Update Booking Status: The status of the expired booking in the booking table is updated to “Available”.



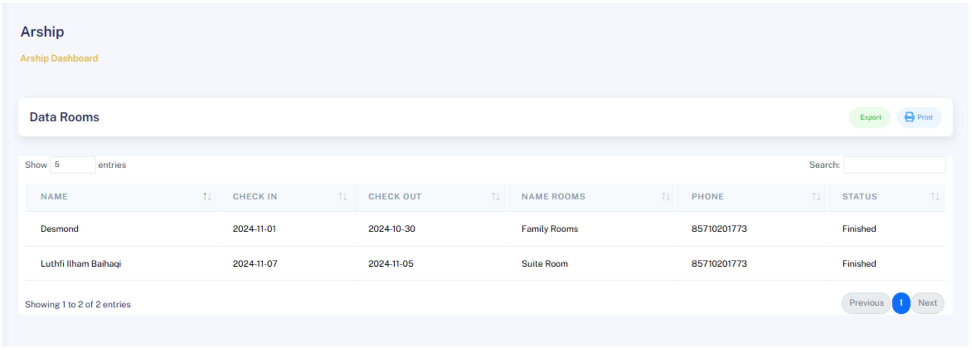
Data Order

Show 5 Search:

NAME	CHECK IN	CHECK OUT	ROOMS	PHONE	STATUS
Desmond	2024-11-25	2024-11-27	Deluxe Room	085710201773	Booked

Show Page 1 or 1

Fig 7.4 Table Data Order



Arship

Arship Dashboard

Data Rooms

Show 5 entries Search:

NAME	CHECK IN	CHECK OUT	NAME ROOMS	PHONE	STATUS
Desmond	2024-11-01	2024-10-30	Family Rooms	85710201773	Finished
Luthfi Ilham Baihaqi	2024-11-07	2024-11-05	Suite Room	85710201773	Finished

Showing 1 to 2 of 2 entries

Fig 7.5 Table Data Rooms

Delete Expired Booking: Finally, delete the expired booking from the booking table.

Purpose: This function ensures that the system stores accurate data by:

- Clearing rooms that are no longer booked.
- Archiving expired bookings for historical purposes.

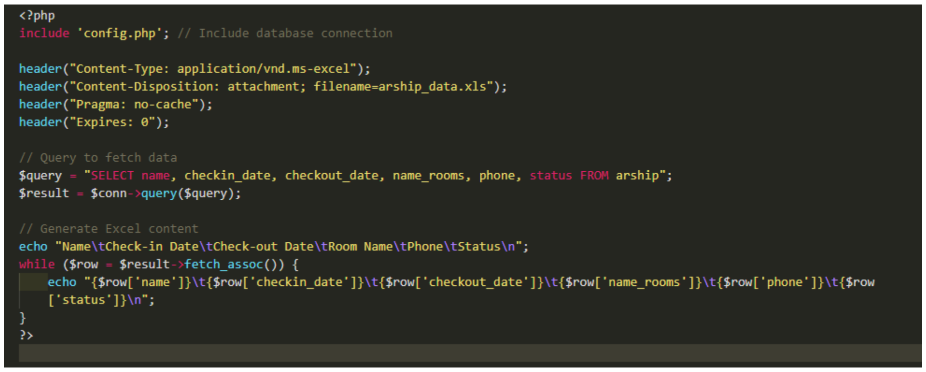
- Clearing the booking table by deleting expired entries.

Key features:

- Database Prepared Statements: Prevents SQL injection by using prepared statements for all queries.
- Efficient Updates: Handles bulk updates for room and booking status.
- Data Archiving: Save important data by storing bookings

8. convert arship data to excel or pdf

export the file and download it in excel or pdf so that the data can be processed further.



```php
<?php
include 'config.php'; // Include database connection

header("Content-Type: application/vnd.ms-excel");
header("Content-Disposition: attachment; filename=arship_data.xls");
header("Pragma: no-cache");
header("Expires: 0");

// Query to fetch data
$query = "SELECT name, checkin_date, checkout_date, name_rooms, phone, status FROM arship";
$result = $conn->query($query);

// Generate Excel content
echo "Name\tCheck-in Date\tCheck-out Date\tRoom Name\tPhone\tStatus\n";
while ($row = $result->fetch_assoc()) {
    echo "{$row['name']}\t{$row['checkin_date']}\t{$row['checkout_date']}\t{$row['name_rooms']}\t{$row['phone']}\t{$row
    ['status']}\n";
}
?>
```

Fig 8.1 Code Snippet export to excel

HTTP Headers for Excel File

The code sets specific HTTP headers to indicate that the response is an Excel file (Content-Type: application/vnd.ms-excel) with the name arship_data.xls. These headers also prevent browser caching to ensure the file is always up-to-date.

Query to Fetch Data

An SQL query is executed to fetch specific columns from the arship table:

name: User's name.

checkin_date: Check-in date.

checkout_date: Check-out date.

name_rooms: Room name.

phone: Phone number.

status: Reservation status.

Creating the Excel Column Headers

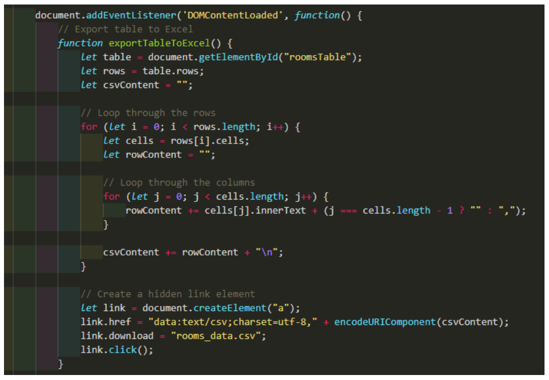
The column headers (Name, Check-in Date, Check-out Date, Room Name, Phone, Status) are written to the Excel file as the first row. Columns are separated by tabs (\t), and a new line character (\n) starts the next row.

Writing Data to the Excel File

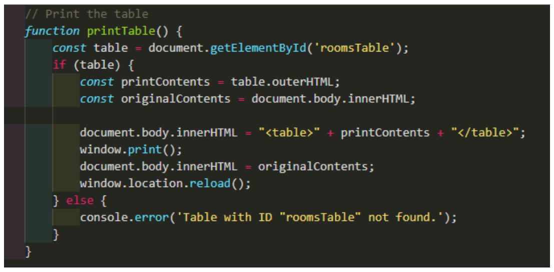
The data retrieved from the SQL query is looped through and written to the Excel file. Each column value is separated by tabs, and rows are separated by new line characters.

Downloadable Excel File

Once the data has been written, the file is automatically prompted for download by the browser with the name arship_data.xls.



```
document.addEventListener('DOMContentLoaded', function() {
    // Export table to Excel
    function exportTableToExcel() {
        let table = document.getElementById("roomsTable");
        let rows = table.rows;
        let csvContent = "";

        // Loop through the rows
        for (let i = 0; i < rows.length; i++) {
            let cells = rows[i].cells;
            let rowContent = "";

            // Loop through the columns
            for (let j = 0; j < cells.length; j++) {
                rowContent += cells[j].innerText + (j === cells.length - 1 ? "" : ",");
            }

            csvContent += rowContent + "\n";
        }

        // Create a hidden link element
        let link = document.createElement("a");
        link.href = "data:text/csv;charset=utf-8," + encodeURIComponent(csvContent);
        link.download = "rooms_data.csv";
        link.click();
    }
```



```
// Print the table
function printTable() {
    const table = document.getElementById('roomsTable');
    if (table) {
        const printContents = table.outerHTML;
        const originalContents = document.body.innerHTML;

        document.body.innerHTML = "<table>" + printContents + "</table>";
        window.print();
        document.body.innerHTML = originalContents;
        window.location.reload();
    } else {
        console.error('Table with ID "roomsTable" not found.');
    }
}
```

Fig 8.2 Code Snippet Export To Pdf

- .**Export Table to Excel (CSV Format)**

Access the Table: The table is identified using document.getElementById("roomsTable").

Iterate Through Rows: The rows property is used to loop through all the rows of the table.

For each row, the cells property is accessed to get its columns.

Each cell’s text content (innerText) is added to the rowContent, separated by commas. This process generates CSV-style formatting.

Create CSV Content: The rows are joined together with newline characters (\n) to form the full content.

Generate a Download Link:

A hidden <a> link element is created and assigned the CSV content as a data:text/csv URL.

The download attribute is set to suggest a file name (rooms_data.csv) for the download.

The link is programmatically "clicked" to trigger the download.

- **Print the Table**

The function printTable allows the user to print the content of the table:

Retrieve the Table: The function looks for a table with the ID roomsTable using document.getElementById.

Prepare Print Content: If the table exists:

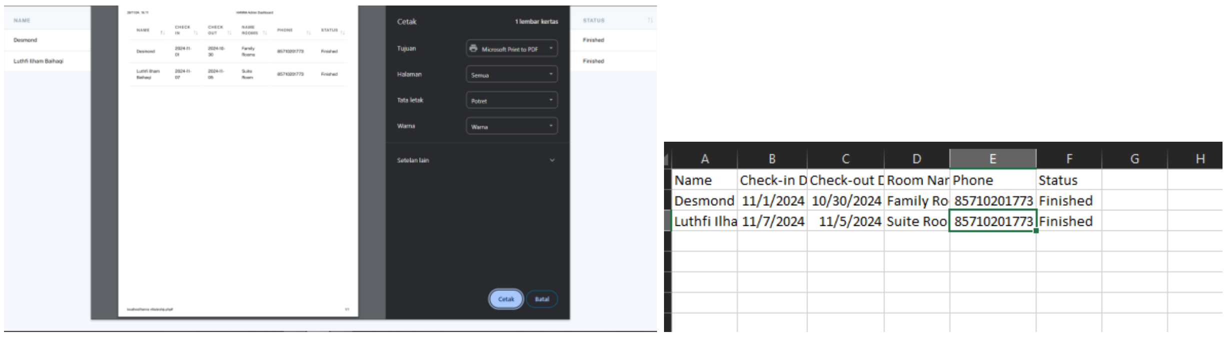
The table's HTML (outerHTML) is extracted.

The current content of the webpage is saved (originalContents) to restore it later.

The webpage content is temporarily replaced with the table’s HTML for printing.

Trigger Print Dialog: The browser’s print functionality is invoked with window.print().

Restore Original Content: After printing, the original webpage content is restored, and the page is refreshed with window.location.reload().



NAME	CHECK IN	CHECK OUT	NAME ROOMS	PHONE	STATUS
Desmond	2024-11-01	2024-10-30	Family Rooms	85710201773	Finished
Luthfi Ilham Baihaqi	2024-11-07	2024-11-05	Suite Room	85710201773	Finished

Cetak 1 lembar kertas

Tujuan: Microsoft Print to PDF

Halaman: Semua

Tata letak: Potret

Warna: Warna

Setelan lain

	A	B	C	D	E	F	G	H
	Name	Check-in D	Check-out D	Room Nar	Phone	Status		
	Desmond	11/1/2024	10/30/2024	Family Ro	85710201773	Finished		
	Luthfi Ilha	11/7/2024	11/5/2024	Suite Roo	85710201773	Finished		

Fig 8.3 Export to Pdf and Exce

SOURCES

https://chatgpt.com/

https://www.w3schools.com/php/php_sessions.asp

https://www.w3schools.com/php/php_includes.asp

https://www.w3schools.com/php/php_file_upload.asp

https://www.w3schools.com/js/