

Universiteti i Prishtinës “Hasan Prishtina”

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike



Lënda: Programimi në Ueb 2

Tema: Ticket Booker (Faza 2)

Prof. Dr. Dhuratë Hyseni
Ass. Msc. Dalinë Vranovci

Punuan:

Lorik Agaj
Dion Gashi
Brela Shala
Blerton Ismaili
Dielli Doçi

Prishtinë 2024

1. Hyrje.....	3
2. Sqarime të kërkesave.....	3
2.1 PHP dhe MySQL.....	3
2.2 PHP (konc. e avancuara) Poinerët dhe SQL Injection.....	5
2.2.1 MySQL Injections.....	6
2.3 PHP(konc. e avancuara) Files, Error Handling dhe PHP Email.....	7
2.3.1 Error Handling.....	7
2.3.2 PHP Email.....	8

1. Hyrje

Ky dokument shërben për dokumentimin e fazës së dytë të projektit nga lënda “Programimi në Ueb 2”. Meqënëse në dokumenin e raportit të fazës së parë është sqaruar domeni i projektit, kësaj rradhe do të sqarojmë përmbushjen e kërkesave për këtë fazë.

2. Sqarime të kërkesave

2.1 PHP dhe MySQL

Gjatë fazës së dytë na është kërkuar që të funksionalizohet projekti jonë duke e lidhur atë me bazë të të dhënave. Ne kemi përdorur **phpmyadmin** për manipulime të ndryshme me bazë të të dhënave. Së pari me anë të një php file kemi bërë lidhjen me bazën e të dhënave.

```
// Database configuration
$servername = "localhost:3308";
$username = "root"; // Replace with your database username
$password = ""; // Replace with your database password
$dbname = "ticketbooker"; // Replace with your database name

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

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

// Connection successful message for testing purposes
echo "Connected successfully";
```

Pastaj me anë të një skripte të SQL kemi shkruar strukturën e databazës e cila është egzekutuar me anë të phpmyadmin.

Server: 127.0.0.1 » Database: ticketbooker

Structure SQL Search Query Export Import Operations Privileges Routines Events

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> tickets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> user_tickets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
3 tables	Sum	2	InnoDB	utf8mb4_general_ci	112.0 KiB	0 B

Check all With selected:

Po ashtu në projektin tonë kemi disa funksione të cilat shërbejnë për insertim, update dhe fshirje në bazë të të dhënave. Një shembull i këtyre funksioneve është funksioni createUser, i cili shërben për krijimin e përdoruesve të rinj në pjesën e Signup të faqes.

```
function createUser($firstName, $lastName, $username, $email, $password, $userType)
{
    global $conn; // Access the database connection object

    try {
        // Generate a random salt
        $salt = bin2hex(random_bytes(16));

        // Hash the password with the salt using bcrypt
        $hashedPassword = password_hash($password . $salt,
PASSWORD_BCRYPT);

        // Prepare the SQL statement
        $stmt = $conn->prepare("INSERT INTO users (first_name, last_name,
username, email, password_hash, salt, user_type) VALUES (?, ?, ?, ?, ?, ?,
?)");

        // Bind parameters to the statement
        $stmt->bind_param("ssssss", $firstName, $lastName, $username,
$email, $hashedPassword, $salt, $userType);

        // Execute the statement
        if ($stmt->execute()) {
            sendEmail($email, 'Welcome', 'new user', $firstName);
            return true; // User inserted successfully
        }
    }
}
```

```

    } else {
        // Print the error message
        throw new Exception("Failed to insert user");
    }
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
    return false; // Failed to insert user
}
}

```

2.2 PHP (konc. e avancuara) Poinerët dhe SQL Injection

Referencat në PHP na lejojnë që të bëhen ndryshimet në variablën e caktuar në të njejtën hapësirë memorike. Po ashtu ato na mundësojnë të përdorim aliase të ndryshme të cilat pointojnë në të njejtin lokacion memorik. Me anë të funksionit `unset()` ne largojmë referencimin e variablës së caktuar. Një shembull i përdorimit të referencave në projektin tonë është funksioni në vijim për filtrim.

```

function filterTickets(&$tickets)
{
    $filteredTickets = array();

    if (isset($_GET['find'])) {
        foreach ($tickets as &$ticket) {
            if (
                ($_GET['type'] == '' || $_GET['type'] ==
$ticket->type) &&
                ($_GET['when'] == '' || $_GET['when'] ==
$ticket->date) &&
                ($_GET['location'] == '' || $_GET['location'] ==
$ticket->location)
            ) {
                $filteredTickets[] = $ticket;
            }
            unset($ticket);
        }
        return $filteredTickets;
    } else {
        return $tickets; // Return all tickets if no filters applied
    }
}

```

2.2.1 MySQL Injections

MySQL injections janë dobësi të sigurisë, ku me anë të tyre përdorues të ndyrshëm mund të manipulojnë query të ndryshëm të SQL dhe të ndryshojnë të dhënat në databazë. MySQL Injections dijnë të jenë tejet të rrezikshëm, ku ato mund të shkaktojnë qasje të pa autorizuar, manipulim të të dhënave, data dumping, ndryshime të privilegjeve, denial of service etj. Mirëpo për t'ju shmangur MySQL Injections duhet përdorur metoda të ndryshme parandaluese si prepared statements, të cilat ndajnë strukturën e query-të me të dhënat. Forma tjera të mbrojtjes janë validimi i inputit të përdoruesit, përdorimi i Object-Relational Mapping (ORM) librave, etj. Në shembullin në vijim kemi përdorur prepared statements, ku ? shërben si placeholder i të dhënave.

```
function authenticateUser($email, $password) {
    global $conn; // Assuming $conn is your database connection object

    // Retrieve hashed password from the database based on the provided email
    $query = "SELECT password_hash,salt FROM users WHERE email = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows === 1) {
        // User found, verify password
        $userData = $result->fetch_assoc();
        $hashedPassword = $userData['password_hash'];
        $salt = $userData['salt'];
        // Verify password
        if (password_verify($password . $salt, $hashedPassword)) {
            // Password is correct
            return true;
        } else {
            // Invalid password
            return false;
        }
    } else {
        // User not found
        return false;
    }

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

```
}
```

2.3 PHP(konc. e avancuara) Files, Error Handling dhe PHP Email

2.3.1 Error Handling

Error handling është përgaditur në disa funksione të ndryshme me funksione të caktuara. Një nga përdorimet është dhe me anë të bllokut try catch në funksionin në vijim.

```
function createUser($firstName, $lastName, $username, $email, $password,
$userType)
{
    global $conn; // Access the database connection object

    try {
        // Generate a random salt
        $salt = bin2hex(random_bytes(16));

        // Hash the password with the salt using bcrypt
        $hashedPassword = password_hash($password . $salt,
PASSWORD_BCRYPT);

        // Prepare the SQL statement
        $stmt = $conn->prepare("INSERT INTO users (first_name, last_name,
username, email, password_hash, salt, user_type) VALUES (?, ?, ?, ?, ?, ?,
?)");

        // Bind parameters to the statement
        $stmt->bind_param("ssssss", $firstName, $lastName, $username,
$email, $hashedPassword, $salt, $userType);

        // Execute the statement
        if ($stmt->execute()) {
            sendEmail($email, 'Welcome', 'new user', $firstName);
            return true; // User inserted successfully
        } else {
            // Print the error message
            throw new Exception("Failed to insert user");
        }
    }
}
```

```

    }
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
    return false; // Failed to insert user
}
}

```

2.3.2 PHP Email

Me anë të librarisë PHPMailer ne kemi krijuar një formë të automatizuar të dërgimit të emailave. Ku në momentin që një përdorues bëhet signup, atij i vije një email përshëndetës për ti uruar mirëseardhje. Më poshtë gjeni kodin i cili mundëson dërgimin e emailit.

```

require_once("php-mailer/PHPMailer.php");
require_once("php-mailer/SMTP.php");
require_once("php-mailer/Exception.php");

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

function sendEmail( $email, $message, $subject,$name){
$mail = new PHPMailer(true); // Enable exceptions
    $mail->isSMTP();
    $mail->Host = "smtp.gmail.com";
    $mail->SMTPAuth = true;
    //Enable less security apps in email
    //Replace your student email and password in here
    $mail->Username = 'admin@student.uni-pr.edu';
    $mail->Password = 'admin';
    $mail->Port = 587;

    $mail->setFrom('admin@student.uni-pr.edu', $name);
    $mail->addReplyTo($email);
    $mail->addAddress($email);

    $mail->isHTML(true);
    $mail->Subject = $subject;
    $mail->Body = $message;

    $mail->send();
}

```