Introduction to HTTP Methods

- GET and POST requests
- Differences between GET and POST requests

Form Handling with PHP

- Creating a simple HTML form for GET requests
- Creating a simple HTML form for POST requests
- Handling GET requests with PHP
- Handling POST requests with PHP

Database Interactions with PHP

- Setting up a MySQL database and table
- Connecting to a MySQL database using PDO
- Inserting data into the database using PHP
- Retrieving data from the database using PHP

User Authentication System

- Creating a user registration form
- Handling user registration with PHP and storing data in the database
- Creating a user login form
- Handling user login with PHP and validating credentials
- Implementing sessions for user authentication

User Profile Management

- **Creating a user profile page to display user information**
- **Creating a profile edit form**
- **Handling profile updates with PHP and updating data in the database**

Password Reset Functionality

- **Creating a password reset form**
- **Handling password reset requests with PHP**
- **Updating user passwords in the database**

Final Touches

- **Adding validation to forms**
- **Ensuring proper styling of forms and pages**

Version Control with Git and GitHub

- **Introduction to Git and GitHub**
- **Installing and configuring Git**
- **Creating a GitHub repository**
- **Initializing a local Git repository**
- **Committing and pushing code to GitHub**

Day 1: Introduction to GET and POST Requests

Objective: Understand the basics of GET and POST requests and their differences.

1. Introduction to HTTP Methods

- **GET** requests data from a server (should only retrieve data and have no other effect).
- **POST** submits data to be processed to a specified resource.

2. Basic Form Handling

Create a simple HTML form to demonstrate GET requests.
html
Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>GET Request Example</title>
</head>
<body>
    <form action="get_example.php" method="get">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

-

Create a simple HTML form to demonstrate POST requests.
html
Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>POST Request Example</title>
</head>
<body>
```

```
    <form action="post_example.php" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

-

3. PHP Handling of GET and POST Requests

Handle GET requests in `get_example.php`.

php

Copy code

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $name = htmlspecialchars($_GET['name']);
    echo "Hello, $name!";
}
?>
```

-

Handle POST requests in `post_example.php`.

php

Copy code

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = htmlspecialchars($_POST['name']);
    echo "Hello, $name!";
}
?>
```

-

4. Exercise:

- Create a contact form using POST and display the submitted data on a new page.

Day 2: Database Interactions with PHP

Objective: Learn to interact with a MySQL database using PHP.

1. Setting Up the Database

Use phpMyAdmin or command line to create a database and a `contacts` table.

sql

Copy code

```sql
CREATE DATABASE my_database;
USE my_database;
CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

-

2. Connecting to the Database

Write a PHP script to connect to the MySQL database using PDO.

php

Copy code

```php
<?php
$host = '127.0.0.1';
$db = 'my_database';
$user = 'root';
$pass = '';
$charset = 'utf8mb4';

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];

try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
```

```
}
?>
```

-

3. Inserting Data into the Database

Extend the contact form to save submitted data into the database.

html

Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contact Form</title>
</head>
<body>
    <form action="save_contact.php" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <label for="message">Message:</label>
        <textarea id="message" name="message" required></textarea>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

php

Copy code

```
<?php
require 'config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = htmlspecialchars($_POST['name']);
    $email = htmlspecialchars($_POST['email']);
    $message = htmlspecialchars($_POST['message']);

    $stmt = $pdo->prepare('INSERT INTO contacts (name, email, message)
VALUES (?, ?, ?)');
    $stmt->execute([$name, $email, $message]);
```

```
    echo "Contact saved successfully!";
}
?>
```

-

4. Exercise:

- Modify the contact form to save the submitted data into a `contacts` table.

Day 3: User Registration and Login System

Objective: Create a simple user registration and login system.

1. User Registration Form

Create an HTML form for user registration (first name, last name, email, password).
html
Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register</title>
</head>
<body>
    <form action="/actions/register_action.php" method="post">
        <label for="firstname">First Name:</label>
        <input type="text" id="firstname" name="firstname" required>
        <label for="lastname">Last Name:</label>
        <input type="text" id="lastname" name="lastname" required>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

-

2. Handling User Registration

Write a PHP script to handle form data and insert it into the database.

php

Copy code

```php
<?php
session_start();
require '../config/config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $firstname = htmlspecialchars($_POST['firstname']);
    $lastname = htmlspecialchars($_POST['lastname']);
    $email = htmlspecialchars($_POST['email']);
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

    $stmt = $pdo->prepare('INSERT INTO users (firstname, lastname, email, password) VALUES (?, ?, ?, ?)');
    $stmt->execute([$firstname, $lastname, $email, $password]);

    header('Location: /pages/login.php');
    exit();
}
?>
```

-

3. User Login Form

Create an HTML form for user login (email, password).

html

Copy code

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <form action="/actions/login_action.php" method="post">
        <label for="email">Email:</label>
```

```
        <input type="email" id="email" name="email" required>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

-

4. Handling User Login

Write a PHP script to verify user credentials and use sessions to keep users logged in.

php

Copy code

```
<?php
session_start();
require '../config/config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = htmlspecialchars($_POST['email']);
    $password = htmlspecialchars($_POST['password']);

    $stmt = $pdo->prepare('SELECT * FROM users WHERE email = ?');
    $stmt->execute([$email]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['firstname'] = $user['firstname'];
        $_SESSION['lastname'] = $user['lastname'];
        header('Location: /pages/dashboard.php');
    } else {
        echo 'Invalid email or password.';
    }
}
?>
```

-

5. Exercise:

- Complete the registration and login functionality, ensuring users can register and log in successfully.

Day 4: User Profile Management

Objective: Implement user profile management features.

1. Profile Page

Create a profile page to display user information.

php

Copy code

```php
<?php
session_start();
require '../config/config.php';

if (!isset($_SESSION['user_id'])) {
    header('Location: /pages/login.php');
    exit();
}

$user_id = $_SESSION['user_id'];
$stmt = $pdo->prepare('SELECT * FROM users WHERE id = ?');
$stmt->execute([$user_id]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Profile</title>
</head>
<body>
    <h1>Profile</h1>
    <p>First Name: <?php echo htmlspecialchars($user['firstname']);
?></p>
    <p>Last Name: <?php echo htmlspecialchars($user['lastname']);
?></p>
```

```
    <p>Email: <?php echo htmlspecialchars($user['email']); ?></p>
    <a href="/pages/edit_profile.php">Edit Profile</a>
    <a href="/actions/logout.php">Logout</a>
</body>
</html>
```

-

2. Profile Edit Form

Create a form to allow users to edit their profile information.

php

Copy code

```
<?php
session_start();
require '../config/config.php';

if (!isset($_SESSION['user_id'])) {
    header('Location: /pages/login.php');
    exit();
}

$user_id = $_SESSION['user_id'];
$stmt = $pdo->prepare('SELECT * FROM users WHERE id = ?');
$stmt->execute([$user_id]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Profile</title>
</head>
<body>
    <h1>Edit Profile</h1>
    <form action="/actions/edit_profile_action.php" method="post">
        <label for="firstname">First Name:</label>
        <input type="text" id="firstname" name="firstname"
value="<?php echo htmlspecialchars($user['firstname']); ?>" required>
        <label for="lastname">Last Name:</label>
```

```
        <input type="text" id="lastname" name="lastname" value="<?php
echo htmlspecialchars($user['lastname']); ?>" required>
        <button type="submit">Save Changes</button>
    </form>
</body>
</html>
```

-

3. Handling Profile Updates

Write a PHP script to handle profile updates and save the changes to the database.

php

Copy code

```
<?php
session_start();
require '../config/config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user_id = $_SESSION['user_id'];
    $firstname = htmlspecialchars($_POST['firstname']);
    $lastname = htmlspecialchars($_POST['lastname']);

    $stmt = $pdo->prepare('UPDATE users SET firstname = ?, lastname =
? WHERE id = ?');
    $stmt->execute([$firstname, $lastname, $user_id]);

    header('Location: /pages/profile.php');
    exit();
}
?>
```

-

4. Exercise:

- Allow users to update their profile information and upload a profile picture.

Day 5: Password Reset and Final Touches

Objective: Implement password reset functionality and polish the project.

1. Password Reset Form

Create a form for users to reset their password.

html

Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Password Reset</title>
</head>
<body>
    <form action="/actions/password_reset_action.php" method="post">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <button type="submit">Reset Password</button>
    </form>
</body>
</html>
```

-

2. Handling Password Reset

Write a PHP script to handle password reset requests and update the password in the database.

php

Copy code

```
<?php
require '../config/config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = htmlspecialchars($_POST['email']);

    $stmt = $pdo->prepare('SELECT * FROM users WHERE email = ?');
    $stmt->execute([$email]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user) {
        // Generate a new password
```

```
        $new_password = bin2hex(random_bytes(4));
        $hashed_password = password_hash($new_password,
PASSWORD_DEFAULT);

        // Update the password in the database
        $stmt = $pdo->prepare('UPDATE users SET password = ? WHERE
email = ?');
        $stmt->execute([$hashed_password, $email]);

        echo "New password: $new_password";
    } else {
        echo "No account found with that email.";
    }
}
?>
```

-

3. Finishing Touches

- Ensure all forms and pages are properly styled.
- Add validation to all forms.

4. Exercise:

- Complete the password reset functionality and ensure all parts of the project work seamlessly.

Day 6: Introduction to GitHub

Objective: Learn to use GitHub for version control and project management.

1. Introduction to Git and GitHub

- Explain what Git and GitHub are and why they are important.

2. Setting Up Git

- Install Git on their systems.

Configure Git with their username and email.
bash
Copy code
```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

-

3. Creating a GitHub Repository

- Create a new repository on GitHub for their project.

4. Pushing Code to GitHub

Initialize a local Git repository in their project directory.
bash
Copy code
```
git init
git add .
git commit -m "Initial commit"
```

-

Add the remote repository and push the commits.
bash
Copy code
```
git remote add origin https://github.com/yourusername/auth_system.git
git push -u origin main
```

-

5. Exercise:

- Push their entire project to GitHub and share the repository link.