

Nama : La Ode Muhammad Gazali
NIM : 222212696
Kelas : 2KS2

PRAKTIKUM PEMROGRAMAN BERBASIS WEB

(Form Handling & Koneksi Database)

Penugasan

1. Apa perbedaan \$_POST dan \$_GET? Ilustrasikan dengan contoh.
2. Apa perbedaan PDO::EXEC dan PDO::QUERY? Apa pertimbangan memilih salah satunya dibanding yang lain?
3. Apa saja perbedaan antara penggunaan ekstensi MySQLi dengan PDO?
4. Jelaskan dengan contoh perbedaan XSS (Cross-Site Scripting) dan SQL Injection! Ilustrasikan dengan contoh.
5. Kumpulkan hasil pengerjaan kelompok Anda dan hasil dari praktikum Modul 9

Penyelesaian

1. Perbedaan \$_POST dan \$_GET?

\$_POST :

- Digunakan untuk mengirim data form ke server menggunakan metode HTTP POST.
- Data yang dikirim tidak terlihat di URL.
- Biasanya digunakan untuk mengirim data sensitif seperti password atau data yang panjang.

Contoh: Memproses data login dengan method POST karena termasuk data yang sensitif

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Memproses data username dan password dengan post

    echo "Welcome, $username! Your password is: $password";
} else {
    echo "Invalid request method!";
}
?>
```

\$_GET?

- Digunakan untuk mengirim data form ke server menggunakan metode HTTP GET.
- Data yang dikirim ditampilkan di URL.
- Biasanya digunakan untuk mengirim data yang bersifat non-sensitive dan bisa dibagikan.

Contoh: Menangani proses searching yang tidak termasuk data sensitive

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    if (isset($_GET['search'])) {
        $search_query = $_GET['search'];
        // Process the search query, for example, perform a database
        search

        echo "You searched for: $search_query";
    } else {
        echo "No search query provided!";
    }
} else {
    echo "Invalid request method!";
}
?>
```

2. Perbedaan PDO::EXEC dan PDO::QUERY

PDO::EXEC:

- Digunakan untuk mengeksekusi perintah SQL yang tidak mengembalikan data, seperti perintah INSERT, UPDATE, DELETE, atau perintah SQL yang tidak menghasilkan hasil kueri (query).
- Metode ini mengembalikan jumlah baris yang terpengaruh oleh eksekusi perintah SQL. Ini bisa menjadi jumlah baris yang diubah, dihapus, atau ditambahkan oleh perintah SQL yang dieksekusi.
- Jika ingin menggunakan PDO::EXEC maka pertimbangkan bahwa anda hanya ingin mengeksekusi perintah SQL yang memodifikasi database (misalnya, INSERT, UPDATE, DELETE). Dan tidak ada keperluan untuk mengakses data yang dihasilkan oleh perintah SQL

PDO::QUERY:

- PDO::query digunakan untuk mengeksekusi perintah SQL yang menghasilkan data, seperti SELECT.

- Metode ini mengembalikan objek PDOStatement, yang bisa digunakan untuk mengakses data yang dikembalikan oleh perintah SELECT, misalnya untuk mengambil baris-baris hasil kueri.
- Jika ingin menggunakan PDO::QUERY maka pertimbangkan bahwa anda ingin mengeksekusi perintah SQL yang menghasilkan data (misalnya, SELECT). Dan terdapat keperluan untuk mengakses dan memanipulasi hasil kueri, seperti mengambil data dari tabel atau melakukan operasi pada data yang dikembalikan.

3. Perbedaan Mysqli dan PDO

- **API dan Objek:**

MySQLi: Menggunakan fungsi-fungsi prosedural atau objek-oriented. MySQLi memberikan API yang spesifik untuk MySQL, yang memungkinkan lebih banyak kontrol langsung terhadap koneksi dan query.

PDO: Menggunakan objek-oriented. PDO menyediakan antarmuka abstrak untuk berinteraksi dengan berbagai jenis database, bukan hanya MySQL. Ini membuat kode menjadi lebih portabel antar basis data.

- **Dukungan untuk Basis Data Lain:**

MySQLi: Dirancang khusus untuk MySQL, sehingga hanya mendukung MySQL.

PDO: Mendukung berbagai jenis database seperti MySQL, PostgreSQL, SQLite, SQL Server, dan lain-lain. Ini membuat aplikasi lebih fleksibel jika Anda berencana untuk berpindah basis data di masa depan.

- **Prepared Statements:**

MySQLi: Mendukung prepared statements menggunakan fungsi mysqli_prepare() dan binding parameter.

PDO: Juga mendukung prepared statements tetapi dengan sintaksis yang sedikit berbeda. Prepared statements di PDO lebih sering digunakan dalam kode karena sintaksisnya yang lebih ringkas dan mudah dibaca.

- **Kode Lebih Bersih:**

PDO: Kode sering kali lebih bersih dan lebih mudah dibaca karena menggunakan objek dan metode yang konsisten untuk semua operasi database.

MySQLi: Kode dapat menjadi sedikit lebih berserakan karena perlu memilih antara mode prosedural atau objek-oriented, serta memiliki fungsi-fungsi yang lebih spesifik untuk MySQL.

- **Error Handling:**

MySQLi: Memiliki cara yang lebih tradisional untuk menangani kesalahan, seperti penggunaan fungsi mysqli_error() untuk menampilkan pesan kesalahan.

PDO: Menyediakan error handling yang lebih konsisten dan terstruktur dengan penggunaan pengecualian (exception), yang memudahkan penanganan kesalahan dalam kode.

- **Dukungan Transaksi:**

MySQLi: Dukungan transaksi yang solid dengan fungsi-fungsi seperti `mysqli_begin_transaction()`, `mysqli_commit()`, dan `mysqli_rollback()`.

PDO: Juga mendukung transaksi, namun dengan sintaksis yang sedikit berbeda. PDO menyediakan metode `beginTransaction()`, `commit()`, dan `rollBack()` untuk mengelola transaksi.

4. Perbedaan XSS dan SQL Injection

XSS (Cross Site-Scripting)

- XSS menargetkan kelemahan dalam cara browser menginterpretasikan konten web
- XSS terjadi ketika penyerang menyisipkan skrip berbahaya ke dalam halaman web yang kemudian dieksekusi oleh browser pengguna.
- Serangan XSS umumnya terjadi pada bagian input yang diizinkan dalam formulir, komentar, atau parameter URL.
- Tujuan dari serangan XSS adalah untuk mencuri informasi pengguna atau untuk melakukan tindakan tertentu atas nama pengguna yang terinfeksi.

Contoh:

```
<input type="text" name="comment">
<script>alert('XSS Attack!')</script>
```

Jika penyerang memasukkan input berikut sebagai komentar:

```
<script>alert('XSS Attack!')</script>
```

Ketika halaman yang memuat komentar tersebut ditampilkan kepada pengguna lain, skrip `<script>` akan dieksekusi oleh browser pengguna

SQL Injection

- SQL Injection menargetkan kelemahan dalam cara aplikasi web berinteraksi dengan basis data
- SQL Injection terjadi ketika penyerang menyisipkan kode SQL berbahaya ke dalam query SQL yang dieksekusi oleh aplikasi web.
- Serangan SQL Injection umumnya terjadi pada formulir login, pencarian, atau filter yang memproses input pengguna tanpa sanitasi yang tepat.
- Tujuan dari serangan SQL Injection adalah untuk mengakses atau mengubah data dalam basis data atau bahkan mengambil alih kontrol dari sistem.

Contoh

```
<?php
// Misalnya, query login pada aplikasi web
$username = $_POST['username'];
$password = $_POST['password'];
```

```
$sql = "SELECT * FROM users WHERE username='$username' AND  
password='$password'";  
?>
```

Jika penyerang memasukkan input berikut sebagai password:

```
' OR '1'='1
```

Maka query yang dieksekusi akan menjadi:

```
$sql = SELECT * FROM users WHERE username='' OR '1'='1' AND password=''
```

Yang akan mengembalikan hasil yang benar-benar sesuai karena '1'='1' akan selalu bernilai TRUE.

5. Praktikum modul 9

Note : Program berikut merupakan program yang telah dimodifikasi sesuai petunjuk pada modul.

PHP09A.php

```
<!DOCTYPE html>  
<html lang='en-GB'>  
<head>  
<title>PHP09 A</title>  
</head>  
<body>  
<?php  
    echo '  
        <form action="php09B.php" method="post">  
        <label>Item: <input type="text" name="item"></label>  
        </form>  
    ';  
?>  
</body>  
</html>
```

Output:

Item:

PHP09B.php

```
<!DOCTYPE html>  
<html lang='en-GB'>  
<head>  
<title>PHP09 B</title>  
</head>
```

```

<body>
<?php
    // Simpan nilai item dalam input tersembunyi
    echo '
        <form action="php09C.php" method="post">
            <input type="hidden" name="item" value="' . $_REQUEST['item'] .
        "'>

            <label>Item: ', $_REQUEST['item'], '</label><br>
            <label>Address: <input type="text" name="address"></label><br>
        </form>';
    ?>
</body>
</html>

```

Output:

Item: apple
Address:

PHP09C.php

```

<!DOCTYPE html>
<html lang='en-GB'>
<head>
<title>PHP09 C</title>
</head>
<body>
<?php
    echo 'Item: ', $_POST['item'], '<br>';
    echo 'Alamat: ', $_POST['address'], '<br>';
    ?>
<button onclick="goBack()">Back</button>

<script>
function goBack() {
    window.location.href = 'php09A.php';
}
</script>
</body>
</html>

```

Output:

Item: apple
Alamat: Bali

PHP09D.php

```
<!DOCTYPE html>
<html lang='en-GB'>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PHP09 D</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    h1 {
      text-align: center;
    }

    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }

    th, td {
      border: 1px solid #ddd;
      padding: 8px;
      text-align: left;
    }

    th {
      background-color: #f2f2f2;
    }
  </style>
</head>
<body>
  <h1>PHP and Databases</h1>
  <?php
    $db_hostname = "localhost"; // Ganti dengan server database Anda
    $db_database = "praktikum9"; // Ganti dengan nama database Anda
    $db_username = "root"; // Ganti dengan username database Anda
    $db_password = ""; // Ganti dengan password database Anda
    // Untuk praktikum, lebih baik jangan gunakan password asli Anda

    $db_charset = "utf8mb4"; // Opsional
```

```

$dsn =
"mysql:host=$db_hostname;dbname=$db_database;charset=$db_charset";
$opt = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false
);
try {
    $pdo = new PDO($dsn, $db_username, $db_password, $opt);
    echo "<h2>Data in meeting table (Using Table)</h2>\n";
    $stmt = $pdo->query("SELECT * FROM meetings ORDER BY slot");
    $rows = $stmt->fetchAll();

    if (count($rows) > 0) {
        echo "<table>";
        echo "<tr><th>Slot</th><th>Name</th><th>Email</th></tr>";
        foreach ($rows as $row) {
            echo "<tr>";
            echo "<td>" . $row["slot"] . "</td>";
            echo "<td>" . $row["name"] . "</td>";
            echo "<td>" . $row["email"] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
    } else {
        echo "No data found.";
    }

    $pdo = NULL;
} catch (PDOException $e) {
    exit("PDO Error: " . $e->getMessage() . "<br>");
}
?>
</body>
</html>

```

Output:

PHP and Databases		
Data in meeting table (Using Table)		
Slot	Name	Email
1	Michael North	m.north@foe.stis.ac.id
5	Jody Land	j.land@foe.stis.ac.id
7	Trish Shelby	t.shelby@foe.stis.ac.id

PHP09E.php

```
<!DOCTYPE html>
<html lang="en-GB">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Add New Meeting</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f0f0f0;
    }

    .container {
      max-width: 500px;
      margin: 50px auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h1 {
      text-align: center;
    }

    form {
      margin-top: 20px;
    }

    label {
      display: block;
      font-weight: bold;
      margin-bottom: 5px;
    }

    input[type="text"],
    input[type="email"],
    input[type="submit"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 10px;
    }
```

```

        border: 1px solid #ccc;
        border-radius: 3px;
        box-sizing: border-box;
    }

    input[type="submit"] {
        background-color: #007bff;
        color: #fff;
        cursor: pointer;
        transition: background-color 0.3s;
    }

    input[type="submit"]:hover {
        background-color: #0056b3;
    }
</style>
</head>
<body>
    <div class="container">
        <h1>Form Menambahkan Data Meeting</h1>
        <form action="php09E_action.php" method="post">
            <label for="slot">Slot:</label>
            <input type="text" id="slot" name="slot">

            <label for="name">Name:</label>
            <input type="text" id="name" name="name">

            <label for="email">Email:</label>
            <input type="email" id="email" name="email">

            <input type="submit" value="Tambah">
        </form>
    </div>

<?php
$db_hostname = "localhost"; // Ganti dengan server database Anda
$db_database = "praktikum9"; // Ganti dengan nama database Anda
$db_username = "root"; // Ganti dengan username database Anda
$db_password = ""; // Ganti dengan password database Anda
// Untuk praktikum, lebih baik jangan gunakan password asli Anda

$db_charset = "utf8mb4"; // Opsional
$dsn = "mysql:host=$db_hostname;dbname=$db_database;charset=$db_charset";
$opt = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,

```

```

        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES => false
    );

    try {
        // Membuat koneksi ke database
        $pdo = new PDO($dsn, $db_username, $db_password, $opt);

        // Mengambil nilai dari formulir
        $slot = $_POST["slot"];
        $name = $_POST["name"];
        $email = $_POST["email"];

        // Menyiapkan pernyataan SQL untuk menyisipkan data
        $stmt = $pdo->prepare("INSERT INTO meetings (slot, name, email) VALUES
(:slot, :name, :email)");

        // Mengeksekusi pernyataan SQL dengan mengikat parameter
        $stmt->execute(array(':slot' => $slot, ':name' => $name, ':email' =>
$email));

        // Menutup koneksi database
        $pdo = NULL;

        // Redirect ke halaman php09D.php setelah penyisipan berhasil
        header("Location: php09F.php");
        exit();
    } catch (PDOException $e) {
        // Menampilkan pesan error jika terjadi kesalahan
        echo "Error: " . $e->getMessage();
        die();
    }
?>

</body>
</html>

```

Output:

Form Menambahkan Data Meeting

Slot:

Name:

Email:

Tambah

PHP09F.php

```
<!DOCTYPE html>
<html lang='en-GB'>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PHP09 F</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    h1 {
      text-align: center;
    }

    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <div>
    <h2>Form Menambahkan Data Meeting</h2>
    <p><b>Slot:</b></p>
    <input type="text" value="3">
    <p><b>Name:</b></p>
    <input type="text" value="sasa">
    <p><b>Email:</b></p>
    <input type="text" value="sasaf@gmail.com">
    <p>Tambah</p>
  </div>
</body>
</html>
```

```

    th, td {
        border: 1px solid #ddd;
        padding: 8px;
        text-align: left;
    }

    th {
        background-color: #f2f2f2;
    }

    .btn-container {
        display: flex;
        justify-content: space-between;
    }

    .btn-container a {
        margin: 0 5px;
    }

    .success-message {
        background-color: #4CAF50;
        color: white;
        padding: 10px;
        margin-bottom: 20px;
        border-radius: 5px;
        text-align: center;
    }
}
</style>
</head>
<body>
    <h1>Data Meeting</h1>
    <?php
        // Tampilkan pemberitahuan jika ada parameter success di URL
        if (isset($_GET['success']) && $_GET['success'] == 1) {
            echo "<div class='success-message'>Data berhasil
dihapus.</div>";
        }
    ?>
    <?php
        // Pengaturan koneksi ke database
        $db_hostname = "localhost"; // Ganti dengan server database Anda
        $db_database = "praktikum9"; // Ganti dengan nama database Anda
        $db_username = "root"; // Ganti dengan username database Anda
        $db_password = ""; // Ganti dengan password database Anda

```

```

// Koneksi ke database
$pdo = new PDO("mysql:host=$db_hostname;dbname=$db_database",
$db_username, $db_password);
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);









// Mendapatkan data dari tabel meetings
$stmt = $pdo->query("SELECT * FROM meetings ORDER BY slot");
$rows = $stmt->fetchAll();

if (count($rows) > 0) {
    echo "<table>";
    echo
"<tr><th>Slot</th><th>Name</th><th>Email</th><th>Action</th></tr>";
    foreach ($rows as $row) {
        echo "<tr>";
        echo "<td>" . $row["slot"] . "</td>";
        echo "<td>" . $row["name"] . "</td>";
        echo "<td>" . $row["email"] . "</td>";
        echo "<td class='btn-container'>";
        echo "<a href='php09G.php?slot=" . $row["slot"] . "'><img
src='img/edit.png' style='width:30px;height:30px;' alt='Edit'
title='Edit'></a>";
        echo "<a href='php09H.php?slot=" . $row["slot"] . "'><img
src='img/remove.png' style='width:30px;height:30px;' alt='Delete'
title='Delete'></a>";
        echo "</td>";
        echo "</tr>";
    }
    echo "</table>";
} else {
    echo "No data found.";
}

// Menutup koneksi database
$pdo = null;
?>
</body>
</html>

```

Output:

Data Meeting				
Slot	Name	Email	Action	
1	Michael North	m.north@foe.stis.ac.id		
3	sasa	sasaf@gmail.com		
5	Jody Land	j.land@foe.stis.ac.id		
7	Trish Shelby	t.shelby@foe.stis.ac.id		

PHP09G.php (Edit Meeting)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Edit Meeting</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f2f2f2;
    }

    .container {
      width: 50%;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 5px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    h1 {
      text-align: center;
    }

    form {
      margin-top: 20px;
    }

    label {
```

```

        display: block;
        margin-bottom: 5px;
    }

    input[type="text"] {
        width: 100%;
        padding: 8px;
        margin-bottom: 10px;
        border: 1px solid #ccc;
        border-radius: 4px;
        box-sizing: border-box;
    }

    input[type="submit"] {
        width: 100%;
        padding: 10px;
        background-color: #007bff;
        color: #fff;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #0056b3;
    }
</style>
</head>
<body>
    <h1>Edit Meeting</h1>
    <div class="container">
        <form action="php09G.php" method="post">
            <input type="hidden" name="old_slot" value="<?php echo
isset($_GET['slot']) ? $_GET['slot'] : ''; ?>">
            <label for="new_slot">New Slot:</label>
            <input type="text" id="new_slot" name="new_slot" value="<?php
echo isset($_GET['slot']) ? $_GET['slot'] : ''; ?>"><br>
            <label for="new_name">New Name:</label>
            <input type="text" id="new_name" name="new_name" value="<?php
echo isset($_GET['name']) ? $_GET['name'] : ''; ?>"><br>
            <label for="new_email">New Email:</label>
            <input type="text" id="new_email" name="new_email" value="<?php
echo isset($_GET['email']) ? $_GET['email'] : ''; ?>"><br>
            <input type="submit" value="Save">
        </form>
    </div>
</body>
</html>

```



```

</div>

<?php
    // Menyertakan atau menetapkan nilai koneksi database
    $db_hostname = "localhost";
    $db_database = "praktikum9";
    $db_username = "root";
    $db_password = "";

    // Koneksi ke database
    $pdo = new PDO("mysql:host=$db_hostname;dbname=$db_database",
$db_username, $db_password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Memeriksa apakah metode permintaan adalah POST
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Mengambil data dari formulir edit
        $oldSlot = $_POST["old_slot"];
        $newSlot = $_POST["new_slot"];
        $newName = $_POST["new_name"];
        $newEmail = $_POST["new_email"];

        try {
            // Menyiapkan dan mengeksekusi pernyataan SQL untuk mengubah
data pertemuan
            $stmt = $pdo->prepare("UPDATE meetings SET slot = :new_slot,
name = :new_name, email = :new_email WHERE slot = :old_slot");
            $stmt->execute(array(':new_slot' => $newSlot, ':new_name' =>
$newName, ':new_email' => $newEmail, ':old_slot' => $oldSlot));

            // Redirect kembali ke halaman php09F.php setelah berhasil
mengubah data
            header("Location: php09F.php");
            exit();
        } catch(PDOException $e) {
            echo "Error: " . $e->getMessage();
        }
    }

?>
</body>
</html>

```

Edit Meeting

New Slot:

New Name:

New Email:

Save

PHP09H.php

```
<?php
// Menyertakan atau menetapkan nilai koneksi database
$db_hostname = "localhost";
$db_database = "praktikum9";
$db_username = "root";
$db_password = "";





// Koneksi ke database
$pdo = new PDO("mysql:host=$db_hostname;dbname=$db_database", $db_username,
$db_password);
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Memeriksa apakah metode permintaan adalah GET dan apakah parameter slot
diberikan
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET["slot"])) {
    // Mengambil slot dari parameter GET
    $slotToDelete = $_GET["slot"];

    // Menyiapkan dan mengeksekusi pernyataan SQL untuk menghapus data
    pertemuan
    $stmt = $pdo->prepare("DELETE FROM meetings WHERE slot = :slot");
    $stmt->execute(array(':slot' => $slotToDelete));

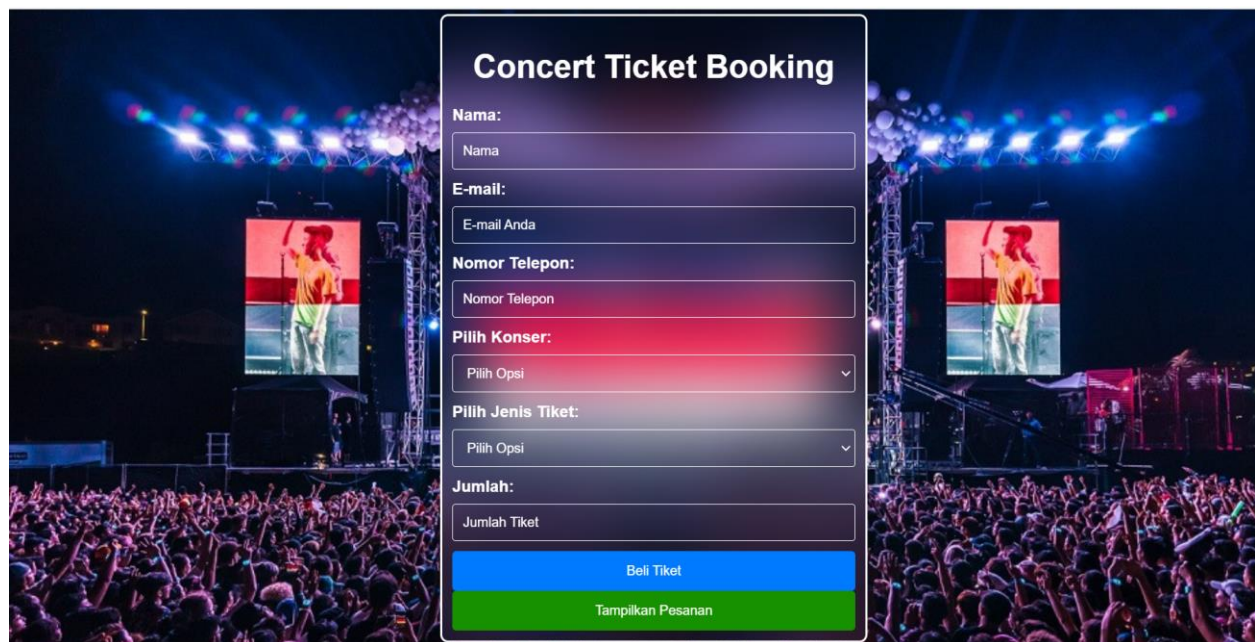
    // Redirect kembali ke halaman php09F.php setelah berhasil menghapus
    data
    header("Location: php09F.php?success=1");
    exit();
}
```

```
}  
?>
```

Data Meeting			
Data berhasil dihapus.			
Slot	Name	Email	Action
1	Michael North	m.north@foe.stis.ac.id	 
7	Trish Shelby	t.shelby@foe.stis.ac.id	 

Hasil Pengerjaan Kelompok (File terlampir dalam rar)

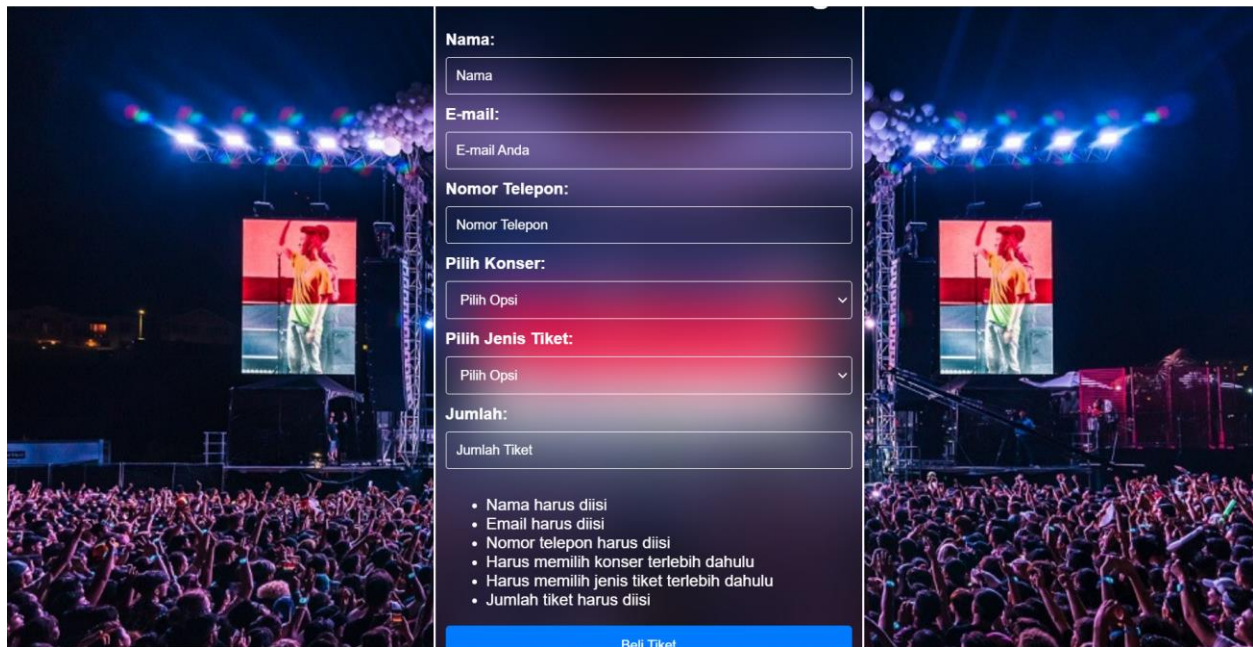
Tampilan awal



The image shows a concert ticket booking form overlaid on a background image of a concert stage with a large crowd. The form is titled "Concert Ticket Booking" and contains the following fields and buttons:

- Nama:** Nama
- E-mail:** E-mail Anda
- Nomor Telepon:** Nomor Telepon
- Pilih Konser:** Pilih Opsi
- Pilih Jenis Tiket:** Pilih Opsi
- Jumlah:** Jumlah Tiket
- Beli Tiket** (Blue button)
- Tampilkan Pesanan** (Green button)

Validasi untuk memastikan tidak ada field yang tidak diisi



The screenshot shows a mobile application interface for concert ticket booking. The background is a live concert scene with a large crowd and stage lights. The form is centered and has a dark theme. It contains the following fields and labels:

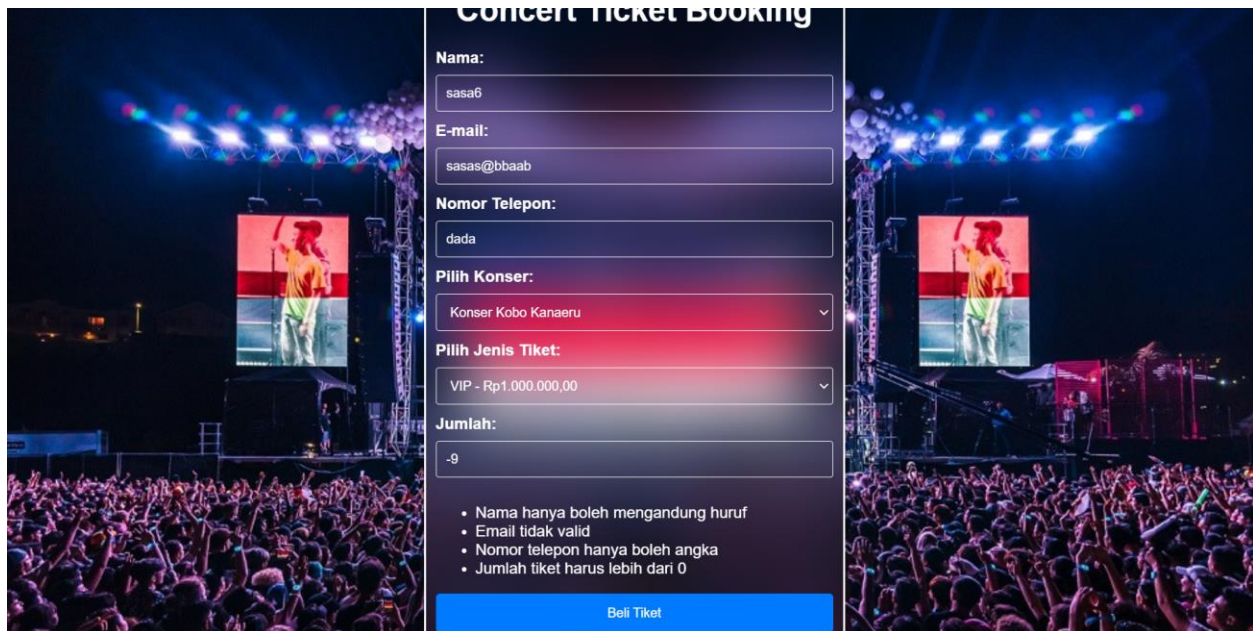
- Nama:** (Name) - Empty text input field.
- E-mail:** (Email) - Empty text input field.
- Nomor Telepon:** (Phone Number) - Empty text input field.
- Pilih Konser:** (Select Concert) - Dropdown menu with "Pilih Opsi" (Select Option) as the placeholder.
- Pilih Jenis Tiket:** (Select Ticket Type) - Dropdown menu with "Pilih Opsi" as the placeholder.
- Jumlah:** (Quantity) - Empty text input field.

Below the form, there is a list of validation messages:

- Nama harus diisi
- Email harus diisi
- Nomor telepon harus diisi
- Harus memilih konser terlebih dahulu
- Harus memilih jenis tiket terlebih dahulu
- Jumlah tiket harus diisi

At the bottom of the form is a blue button labeled "Beli Tiket" (Buy Ticket).

Validasi untuk Nama, Email, Nomor Telepon, dan jumlah



The screenshot shows the same mobile application interface, but with the form fields filled out. The background is the same concert scene. The form contains the following fields and labels:

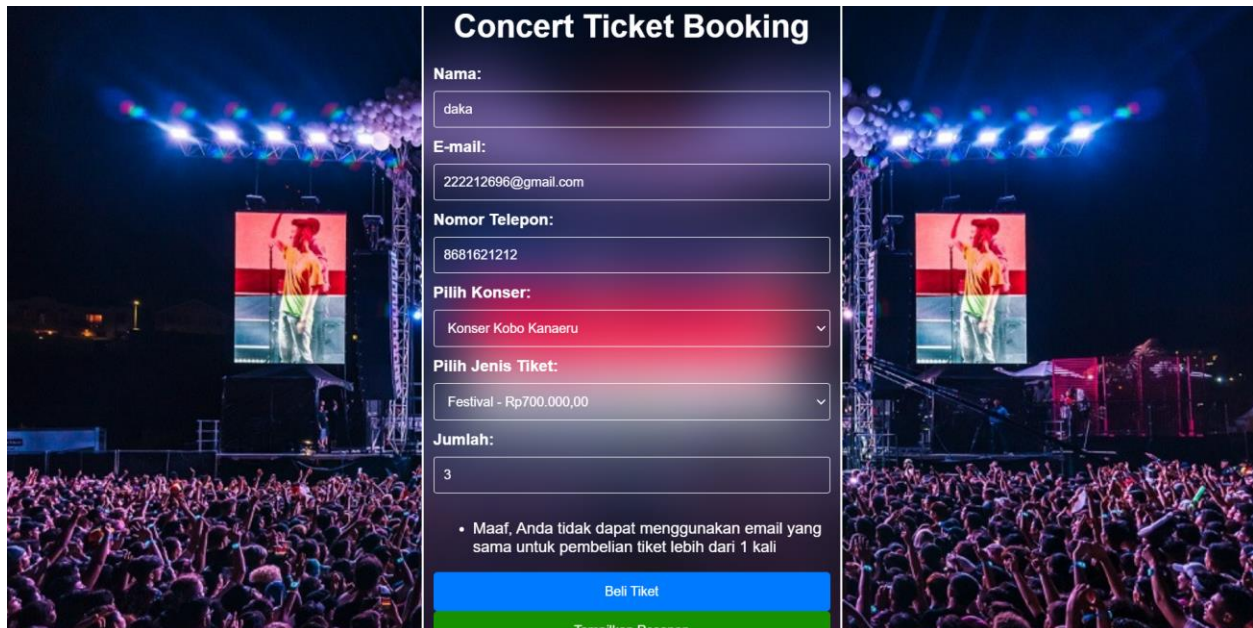
- Nama:** (Name) - Text input field containing "sasa6".
- E-mail:** (Email) - Text input field containing "sasas@bbaab".
- Nomor Telepon:** (Phone Number) - Text input field containing "dada".
- Pilih Konser:** (Select Concert) - Dropdown menu with "Konser Kobo Kanaeru" selected.
- Pilih Jenis Tiket:** (Select Ticket Type) - Dropdown menu with "VIP - Rp1.000.000,00" selected.
- Jumlah:** (Quantity) - Text input field containing "-9".

Below the form, there is a list of validation messages:

- Nama hanya boleh mengandung huruf
- Email tidak valid
- Nomor telepon hanya boleh angka
- Jumlah tiket harus lebih dari 0

At the bottom of the form is a blue button labeled "Beli Tiket" (Buy Ticket).

Validasi agar satu email hanya digunakan untuk satu proses pemesanan

A screenshot of a 'Concert Ticket Booking' form overlaid on a background image of a concert stage with a large crowd. The form is titled 'Concert Ticket Booking' and contains several input fields and dropdown menus. The fields are: 'Nama:' with the value 'daka'; 'E-mail:' with the value '222212696@gmail.com'; 'Nomor Telepon:' with the value '8881621212'; 'Pilih Konser:' with a dropdown menu showing 'Konser Kobo Kanaeru'; 'Pilih Jenis Tiket:' with a dropdown menu showing 'Festival - Rp700.000,00'; and 'Jumlah:' with the value '3'. Below the form, there is a blue button labeled 'Beli Tiket' and a green button labeled 'Tampilkan Pesanan'. A message below the form states: 'Maaf, Anda tidak dapat menggunakan email yang sama untuk pembelian tiket lebih dari 1 kali'.

Tampilan daftar pemesanan tiket

A screenshot of a 'DAFTAR PEMESANAN TIKET KONSER' (Concert Ticket Booking List) overlaid on a background image of a concert stage with a large crowd. The list is a table with 7 columns: Nama, E-mail, Nomor Telepon, Konser, Jenis Tiket, Jumlah Tiket, and Invoice. The table contains 9 rows of data. At the bottom of the table, there is a blue button labeled 'Kembali'.

Terdapat pilihan untuk cetak invoice yang akan menampilkan rincian dari pemesanan

Tampilan Invoice

