

Laporan Praktikum CRUD Dan Login PHP Native



DIBUAT OLEH:

- 1.AMEL ISNA CAHYANINGSIH(3)
- 2.PUPUT DUMAI(26)
- 3.RADHITTIYA VALENT.O(27)
- 4.AKBAR DENI SAPUTRA(36)

REKAYASA PERANGKAT LUNAK

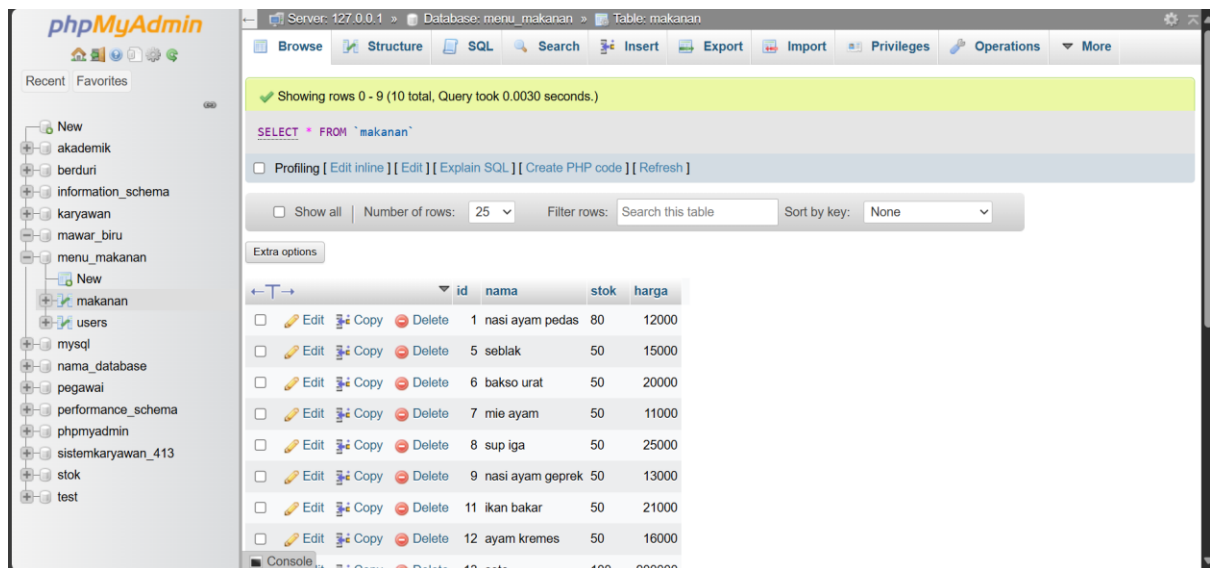
SMK TEXMACO SEMARANG

2024

TAHAPAN

1. Makanan

A. Membuat Database dan Tabel



Pertama buat database dengan nama oop3 dengan tabel didalamnya dengan nama makanan yang berisi ID, Nama, Stok, dan Harga lalu isi 1 data kedalam tabel tersebut.

B. Membuat koneksi dengan nama file database.php

```
<?php
class Database{
    private $host="localhost";
    private $db_name="menu_makanan";
    private $username="root";
    private $password="";
    public $conn;

    public function getConnection(){
        $this->conn=null;
        try {
            $this->conn=new PDO("mysql:host=".$this->host."&dbname=".$this->db_name,
            $this->username,$this->password);

            $this->conn->exec("set names utf8");

        } catch(PDOException $exception) {
            echo"Connection error: ".$exception->getMessage();
        }
    }
}
```

```

    }
    return $this->conn;
}
}
?>

```

- `<?php`: Ini adalah tag pembuka untuk kode PHP. Semua instruksi PHP ditulis di antara tag ini.
- **class Database**: Mendeklarasikan kelas baru bernama Database. Kelas ini akan mengelola koneksi ke database.
- **private \$host**: Menyimpan nama host database (di sini, localhost).
- **private \$db_name**: Nama database yang akan digunakan (dalam hal ini, menu_makanan).
- **private \$username**: Nama pengguna untuk koneksi database (di sini, root).
- **private \$password**: Kata sandi untuk pengguna database (kosong dalam contoh ini).
- **public \$conn**: Variabel publik untuk menyimpan koneksi PDO, yang dapat diakses dari luar kelas.
- **public function getConnection()**: Mendefinisikan metode publik yang akan digunakan untuk mendapatkan koneksi ke database.
- **\$this->conn = null;**: Menginisialisasi variabel koneksi dengan nilai null, memastikan bahwa tidak ada koneksi sebelumnya yang tersisa.
- **try { ... }**: Blok kode yang digunakan untuk menangani pengecualian. Jika terjadi kesalahan saat mencoba menghubungkan, kode dalam blok catch akan dieksekusi.
- **\$this->conn = new PDO(...)**: Menciptakan objek PDO baru untuk menghubungkan ke database MySQL dengan parameter berikut:
- **"mysql:host=" . \$this->host**: Mengatur host database.
- **"dbname=" . \$this->db_name**: Mengatur nama database.
- **\$this->username, \$this->password**: Menggunakan username dan password yang telah didefinisikan.
- **\$this->conn->exec("set names utf8");**: Menetapkan karakter set koneksi menjadi UTF-8, yang memungkinkan penyimpanan dan pengambilan karakter dalam berbagai bahasa.
- **catch(PDOException \$exception) { ... }**: Blok ini akan menangkap kesalahan yang terjadi saat mencoba membuat koneksi.
- **echo "Connection error:" . \$exception->getMessage();**: Jika terjadi kesalahan, pesan kesalahan akan dicetak ke layar. `getMessage()` memberikan deskripsi kesalahan yang terjadi.
- **return \$this->conn;**: Mengembalikan objek koneksi ke database (atau null jika koneksi gagal) sehingga dapat digunakan oleh bagian lain dari aplikasi.
- **}**: Menutup definisi kelas Database.
- **?>**: Tag penutup PHP, menandakan akhir dari kode PHP.

C. Menampilkan Data

```

<?php
include_once 'database/database.php';

```

```

include_once 'objects/makanan.php';

$databse=new Database();
$db=$databse->getConnection();
session_start();
if (!isset($_SESSION['username'])) {
    echo "<script>alert('Anda belum login.
    Silahkan login terlebih dahulu.');"; window.location =
    'index.php';</script>";
    exit();
}
$makanan=new makanan($db);

$stmt=$makanan->read();
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Data Makanan</title>
        <link rel="stylesheet" type="text/css" href="style.css">
        <link rel="stylesheet" type="text/css" href="navbar.css">
    </head>
    <body>
    <nav class="navbar">
        <div class="container">
            <h1 class="logo">RESTORAN PARD</h1>
            <ul class="nav-links">
                <?php
                    $menuItems = [
                        ['name' => 'Beranda', 'link' =>
'objects/home.php'],
                        ['name' => 'Menu', 'link' => 'dashboard.php'],
                        ['name' => 'Logout', 'link' => 'logout.php']
                    ];

                    foreach ($menuItems as $item) {
                        echo '<li><a href="' . $item['link'] . '">' .
$item['name'] . '</a></li>';
                    }
                <?>
            </ul>
        </div>
    </nav>
    <h1>Menu Makanan</h1>
    <a href="create.php">Tambah Makanan</a>
    &ensp;
    <table border="1">

```

```

        <tr>
            <th>id</th>
            <th>nama</th>
            <th>stok</th>
            <th>harga</th>
            <th>aksi</th>
        </tr>
    <?php
    while ($row=$stmt->fetch(PDO::FETCH_ASSOC)) {
        extract($row);
        echo"<tr>";
        echo"<td>{$id}</td>";
        echo"<td>{$nama}</td>";
        echo"<td>{$stok}</td>";
        echo"<td>{$harga}</td>";

        echo"<td><a href='update.php?id={$id}'>Ubah</a><a
href='delete.php?id={$id}'>Hapus</a></td>";
        echo"</tr>";
    }
    ?>
</table>
</body>
</html>

```

- **Fungsi include once:** Memasukkan file PHP lain ke dalam skrip saat ini. `include_once` memastikan bahwa file hanya dimasukkan satu kali, mencegah kesalahan jika file yang sama dipanggil beberapa kali.
- `database/database.php:` Berisi kelas Database untuk mengatur koneksi ke database.
- `objects/makanan.php:` Berisi kelas makanan yang mengelola operasi terkait data makanan (misalnya, membaca, menambah, mengupdate, dan menghapus makanan).
- **Instansiasi Kelas Database:** Membuat objek Database untuk mengelola koneksi database.
- **Mendapatkan Koneksi:** Memanggil metode `getConnection()` untuk mendapatkan koneksi database dan menyimpannya dalam variabel `$db`.
- **`session_start()`:** Memulai sesi PHP, memungkinkan penyimpanan dan akses data sesi.
- **Menutup Tabel:** Mengakhiri elemen tabel.
- **Penutupan HTML:** Menutup elemen `<body>` dan `<html>`, menandakan akhir dari dokumen HTML.
- **Memeriksa Sesi:** Mengecek apakah username ada dalam sesi. Jika tidak ada, maka pengguna akan diberi peringatan melalui alert dan dialihkan ke halaman login (`index.php`). `exit()` digunakan untuk menghentikan eksekusi skrip lebih lanjut setelah pengalihan.
- **Instansiasi Kelas makanan:** Membuat objek makanan dengan memberikan koneksi database `$db`.
- **Mengambil Data:** Memanggil metode `read()` pada objek makanan untuk mengambil data makanan dari database, hasilnya disimpan dalam variabel `$stmt`.

- **HTML Document:** Memulai dokumen HTML dengan deklarasi `<!DOCTYPE html>`, yang menunjukkan bahwa ini adalah dokumen HTML5.
- **Bagian `<head>`:** Memasukkan judul halaman "Data Makanan" dan menghubungkan dua file CSS untuk styling: `style.css` dan `navbar.css`.
- **Navigasi:** Membuat elemen navigasi dengan logo dan daftar tautan.
- **Daftar Menu:** Array `$menuItems` berisi nama dan tautan untuk menu. `foreach` digunakan untuk mencetak setiap item menu ke dalam elemen ``.
- **Judul Halaman:** Menampilkan judul "Menu Makanan".
- **Tautan Tambah Makanan:** Menyediakan tautan untuk membuka halaman penambahan makanan (`create.php`).
- **Tabel HTML:** Membuat tabel dengan header untuk ID, nama, stok, harga, dan aksi (edit/hapus).
- **Loop Melalui Data:** Menggunakan `while` untuk mengambil setiap baris dari hasil query `$stmt` hingga tidak ada lagi data yang tersedia.
- **Menggunakan `extract()`:** Mengambil kolom dari `$row` menjadi variabel dengan nama kolomnya (misalnya, `$id`, `$nama`).
- **Mencetak Data ke Tabel:** Setiap kolom data dicetak dalam elemen `<td>` dan menambahkan tautan untuk mengedit (Ubah) dan menghapus (Hapus) makanan berdasarkan ID.
- **Menutup Tabel:** Mengakhiri elemen tabel.
- **Penutupan HTML:** Menutup elemen `<body>` dan `<html>`, menandakan akhir dari dokumen HTML.

RESTORAN PARD				
			Beranda	Menu Logout
Menu Makanan				
Tambah Makanan				
id	nama	stok	harga	aksi
1	nasi ayam pedas	80	12000	Ubah Hapus
5	seblak	50	15000	Ubah Hapus
6	bakso urat	50	20000	Ubah Hapus
7	mie ayam	50	11000	Ubah Hapus
8	sup iga	50	25000	Ubah Hapus
9	nasi ayam geprek	50	13000	Ubah Hapus
11	ikan bakar	50	21000	Ubah Hapus
12	sup ikan	50	16000	Ubah Hapus

D. Tambah Data

-Create.php

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Tambah Makanan</title>

```

```

<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #F5F2F0;
    margin: 0;
    padding: 20px;
  }
  h1 {
    color: #333;
  }
  form {
    background-color: #BAD0E7;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    padding: 20px;
    max-width: 400px;
    margin: auto;
  }
  label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
  }
  input[type="text"],
  input[type="number"] {
    width: 95%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }
  input[type="submit"] {
    background-color: #0099FF;
    color: white;
    padding: 10px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    width: 100%;
  }
  input[type="submit"]:hover {
    background-color: #003399;
  }
</style>
</head>
<body>
  <h1 align="center">Tambah Makanan</h1>
  <form action="create_action.php" method="POST">

```

```

<label for="Nama">Nama:</label>
<input type="text" name="nama" id="nama" required>

<label for="stok">Stok:</label>
<input type="text" name="stok" id="stok" required>

<label for="harga">Harga:</label>
<input type="number" name="harga" id="harga" required>

<input type="submit" value="Simpan">
</form>
</body>
</html>

```

- **<!DOCTYPE html>**: Menunjukkan bahwa ini adalah dokumen HTML5, memberi tahu browser untuk merender halaman dengan standar HTML5.
- **<html lang="en">**: Membuka elemen HTML dan menetapkan atribut lang ke "en" (bahasa Inggris), yang membantu mesin pencari dan perangkat pembaca layar memahami bahasa konten halaman.
- **<head>**: Bagian ini berisi metadata dan informasi tentang dokumen.
- **<meta charset="UTF-8">**: Menentukan karakter encoding untuk dokumen, memastikan bahwa semua karakter ditampilkan dengan benar.
- **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Mengatur viewport untuk memastikan halaman responsif dan tampak baik di berbagai perangkat (mobile dan desktop).
- **<title>Tambah Makanan</title>**: Menentukan judul halaman yang muncul di tab browser.
- **<style>**: Memasukkan CSS untuk mengatur tampilan dan nuansa halaman.
- **body**: Mengatur font, warna latar belakang, margin, dan padding untuk seluruh halaman.
- **h1**: Mengatur warna teks untuk elemen heading.
- **form**: Mengatur gaya form dengan warna latar belakang, border-radius, shadow, padding, dan ukuran maksimum. margin: auto digunakan untuk memusatkan form.
- **label**: Menampilkan label sebagai block element dengan margin dan ketebalan font yang lebih tebal.
- **input[type="text"], input[type="number"]**: Mengatur input text dan number untuk memiliki lebar, padding, margin, border, dan border-radius tertentu.
- **input[type="submit"]**: Mengatur gaya tombol submit, termasuk warna latar belakang, warna teks, padding, border, border-radius, dan efek pointer saat hover.
- **input[type="submit"]:hover**: Mengubah warna latar belakang tombol saat kursor berada di atasnya.
- **<body>**: Membuka elemen body yang berisi konten utama halaman.
- **<h1 align="center">Tambah Makanan</h1>**: Menampilkan judul "Tambah Makanan" di tengah halaman. Atribut align="center" digunakan untuk memusatkan teks (meskipun penggunaan CSS lebih disarankan). dengan dua atribut:

- **Type number action="create_action.php"**: Menentukan file PHP yang akan diproses saat form.
- **<form action="create_action.php" method="POST">**: Membuka form dikirimkan.
- **method="POST"**: Menentukan metode pengiriman data, yaitu POST, yang lebih aman untuk data sensitif.
- **<label>**: Menyediakan teks untuk masing-masing input field. Atribut for mengaitkan label dengan input yang sesuai menggunakan ID input.
- **<input>**: Menyediakan elemen input untuk pengguna:
- **Type text** untuk nama dan stok, yang meminta input string. untuk harga, yang membatasi input ke angka.
- **required**: Atribut yang memastikan pengguna tidak dapat mengirimkan form tanpa mengisi field tersebut.
- **Tombol Submit**: Menyediakan tombol untuk mengirimkan form. value="Simpan" menentukan teks yang ditampilkan pada tombol.
- **</form>**: Menutup elemen form.
- **</body>**: Menutup elemen body.
- **</html>**: Menutup elemen html, menandakan akhir dokumen HTML.

-create_action.php

```
<?php
include_once 'database/database.php';
include_once 'objects/makanan.php';

$database=new Database();
$db=$database->getConnection();

$makanan=new makanan($db);

$makanan->nama=$_POST['nama'];
$makanan->stok=$_POST['stok'];
$makanan->harga=$_POST['harga'];

if($makanan->create()){
    header("Location: dashboard.php");
    echo "Makanan berhasil ditambahkan.";
} else {
    echo"Gagal menambahkan Makanan.";
}
?>
```

- **<?php**: Menandai awal dari skrip PHP. Segala yang ditulis di antara tag ini akan diproses sebagai kode PHP.
- **include_once**: Memasukkan file PHP yang disebutkan satu kali saja, sehingga tidak akan terjadi duplikasi. Jika file sudah dimasukkan sebelumnya, tidak akan dimasukkan lagi.

- **'database/database.php'**: Memuat file yang berisi definisi kelas untuk koneksi database, yang biasanya mencakup pengaturan koneksi dan penanganan kesalahan.
- **'objects/makanan.php'**: Memuat file yang berisi definisi kelas makanan, yang mungkin memiliki metode untuk mengelola data makanan (seperti menambahkan, mengedit, atau menghapus makanan).
- **\$database = new Database();**: Membuat instance baru dari kelas Database. Ini akan memanggil konstruktor kelas tersebut, yang biasanya mengatur pengaturan koneksi.
- **\$db = \$database->getConnection();**: Memanggil metode `getConnection()` dari objek `$database`, yang mengembalikan objek koneksi database (misalnya, menggunakan PDO) dan menyimpannya di variabel `$db`.
- **\$makanan = new makanan(\$db);**: Membuat instance baru dari kelas makanan, dengan mengoper objek koneksi database sebagai argumen. Ini memungkinkan kelas makanan untuk berinteraksi dengan database.
- **\$makanan->nama = \$_POST['nama'];**: Mengambil nilai dari field input dengan nama `nama` yang dikirimkan melalui metode POST dan menyimpannya dalam properti `nama` dari objek `$makanan`.
- **\$makanan->stok = \$_POST['stok'];**: Melakukan hal yang sama untuk `stok`, mengambil nilai dari field input `stok`.
- **\$makanan->harga = \$_POST['harga'];**: Mengambil nilai dari field input `harga`.
- **if (\$makanan->create());**: Memanggil metode `create()` dari objek `$makanan`. Metode ini biasanya berfungsi untuk menyimpan data makanan ke dalam database. Jika metode ini mengembalikan `true`, berarti data berhasil ditambahkan.
- **header("Location: dashboard.php");**: Jika data berhasil ditambahkan, mengarahkan pengguna ke halaman `dashboard.php` menggunakan header HTTP. Ini harus dilakukan sebelum output apapun dikirim ke browser.
- **echo "Makanan berhasil ditambahkan.";**: Menampilkan pesan sukses (namun tidak akan ditampilkan karena `header()` akan mengakhiri skrip).
- **else { echo "Gagal menambahkan Makanan."; }**: Jika metode `create()` mengembalikan `false`, berarti terjadi kesalahan saat menambahkan makanan, dan pesan kesalahan ditampilkan.
- **?>**: Menandai akhir dari kode PHP. Semua yang ditulis setelah tag ini akan diproses sebagai HTML jika ada, atau diabaikan jika tidak ada.

Tambah Makanan

Nama:

Stok:

Harga:

E. Edit Data

-edit.php

```
<!DOCTYPE html>
<html>
<head>
  <title>Edit Makanan</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #F5F2F0;
      margin: 0;
      padding: 20px;
    }
    h1 {
      color: #333;
      text-align: center;
    }
    form {
      background: #BAD0E7;
      padding: 20px;
      border-radius: 5px;
      box-shadow: 0 2px 10px rgba(0,0,0,0.1);
      max-width: 400px;
      margin: auto;
    }
    label {
      display: block;
      margin: 10px 0 5px;
    }
    input[type="text"],
    input[type="number"] {
      width: 95%;
      padding: 10px;
      margin: 5px 0 20px;
      border: 1px solid #ccc;
      border-radius: 4px;
      transition: border-color 0.3s;
    }
    input[type="text"]:focus,
    input[type="number"]:focus {
      border-color: #5cb85c;
      outline: none;
    }
    input[type="submit"] {
      background-color: #0099FF;
      color: white;
      padding: 10px 15px;
```

```

        border: none;
        border-radius: 4px;
        cursor: pointer;
        width: 100%;
    }
    input[type="submit"]:hover {
        background-color: #003399;
    }
</style>
</head>
<body>
    <h1>Edit Makanan</h1>
    <form method="post">
        <label>Nama:</label>
        <input type="text" name="nama" value="<?php echo $nama; ?>"
required>

        <label>Stok:</label>
        <input type="text" name="stok" value="<?php echo $stok; ?>"
required>

        <label>Harga:</label>
        <input type="number" name="harga" value="<?php echo $harga;
?>" required>

        <input type="submit" value="Update">
    </form>
</body>
</html>

```

- **<!DOCTYPE html>**: Menandakan bahwa dokumen ini adalah dokumen HTML5. Ini membantu browser memahami cara merender halaman dengan benar.
- **<html>**: Menandakan awal dari dokumen HTML.
- **<head>**: Berisi metadata tentang dokumen, termasuk judul, link ke stylesheet, dan skrip.
- **<title>Edit Makanan</title>**: Menentukan judul halaman yang akan ditampilkan di tab browser.
- **<style>**: Menyediakan CSS langsung dalam dokumen untuk mengatur tampilan elemen.
- **CSS untuk body**: Mengatur font, warna latar belakang, margin, dan padding.
- **CSS untuk h1**: Mengatur warna teks dan penjumlahan.
- **CSS untuk form**: Mengatur latar belakang, padding, radius sudut, bayangan, lebar maksimum, dan margin.
- **CSS untuk label**: Mengatur tampilan label untuk input.
- **CSS untuk input[type="text"] dan input[type="number"]**: Mengatur lebar, padding, margin, border, radius sudut, dan transisi pada border saat difokuskan.

- **CSS untuk input[type="submit"]**: Mengatur warna latar belakang, warna teks, padding, border, radius sudut, kursor, dan lebar.
- **CSS untuk input[type="submit"]:hover**: Mengubah warna latar belakang saat mouse hover.
- **</head>**: Menandai akhir elemen head.
- **<body>**: Menandai awal bagian isi dari dokumen yang akan ditampilkan di browser.
- **<h1>Edit Makanan</h1>**: Menampilkan judul utama halaman dengan teks "Edit Makanan".
- **<form method="post">**: Membuat elemen form yang akan mengirimkan data menggunakan metode POST ke skrip pemroses yang ditentukan (default ke halaman ini).
- **<label>**: Menyediakan label untuk input, meningkatkan aksesibilitas.
- **<input type="text" name="nama" value="<?php echo \$nama; ?>" required>**:
 - **type="text"**: Menentukan bahwa input ini adalah teks.
 - **name="nama"**: Memberikan nama pada input yang akan digunakan untuk mengakses datanya di server.
 - **value="<?php echo \$nama; ?>"**: Mengisi input dengan nilai saat ini dari variabel PHP \$nama, yang seharusnya berisi nama makanan yang sedang diedit.
 - **required**: Menandakan bahwa field ini harus diisi sebelum form dapat disubmit.
- **Input untuk stok dan harga**: Mirip dengan input untuk nama, dengan type yang sesuai dan pengisian nilai dari variabel PHP.
- **<input type="submit" value="Update">**: Tombol untuk mengirimkan form, dengan teks "Update".
- **</body>**: Menandai akhir bagian isi dokumen.
- **</html>**: Menandai akhir dari dokumen HTML.

-edit_action

```
<?php
include_once 'database/database.php';
include_once 'objects/makanan.php';

$database = new Database();
$db = $database->getConnection();

$makanan = new makanan($db);

// Check if ID is provided
if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $stmt = $makanan->readOne($id);
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$row) {
        die("makanan not found.");
    }
}
```

```

        // Extract row data
        extract($row);
    }

    // Handle form submission
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $makanan->id = $id;
        $makanan->nama = $_POST['nama'];
        $makanan->stok = $_POST['stok'];
        $makanan->harga = $_POST['harga'];
        $makanan->aksi = $_POST['aksi'];

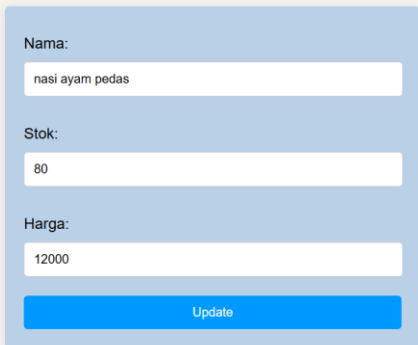
        if ($makanan->update()) {
            header("Location: dashboard.php");
        } else {
            echo "Unable to update makanan.";
        }
    }
}
?>

```

- **include_once 'database/database.php';** Mengimpor file database.php yang berisi kelas Database untuk mengelola koneksi ke database. **include_once** memastikan bahwa file hanya dimasukkan satu kali.
- **include_once 'objects/makanan.php';** Mengimpor file makanan.php yang berisi definisi kelas makanan, yang digunakan untuk mengelola operasi terkait data makanan.
- **\$database = new Database();** Membuat instansi baru dari kelas Database, memungkinkan akses ke metode yang ada di kelas tersebut.
- **\$db = \$database->getConnection();** Memanggil metode **getConnection()** dari kelas Database, yang mengembalikan objek koneksi database menggunakan PDO (PHP Data Objects).
- **\$makanan = new makanan(\$db);** Membuat instansi baru dari kelas makanan, dengan mengirimkan objek koneksi database **\$db** sebagai parameter, sehingga objek makanan dapat berinteraksi dengan database.
- **if (isset(\$_GET['id'])) {**: Memeriksa apakah parameter id tersedia dalam URL. Jika ada, itu berarti pengguna ingin mengedit item makanan tertentu.
- **\$id = \$_GET['id'];**: Mengambil nilai id dari parameter query string dan menyimpannya dalam variabel **\$id**.
- **\$stmt = \$makanan->readOne(\$id);** Memanggil metode **readOne(\$id)** dari objek makanan, yang seharusnya mengembalikan statement untuk mengambil data makanan berdasarkan ID.
- **\$row = \$stmt->fetch(PDO::FETCH_ASSOC);** Mengambil hasil dari statement yang telah dieksekusi dalam bentuk array asosiatif.
- **if (!\$row) { die("makanan not found."); }** Memeriksa apakah tidak ada hasil yang ditemukan; jika tidak ada, script berhenti dengan pesan "makanan not found."

- **extract(\$row);** Mengambil elemen dari array **\$row** dan mendeklarasikan variabel dengan nama yang sama sebagai kunci array (misalnya, jika ada kunci nama dalam **\$row**, maka **\$nama** akan dideklarasikan).
- **if (\$_SERVER["REQUEST_METHOD"] == "POST") {**: Memeriksa apakah permintaan yang diterima adalah permintaan POST, yang berarti form telah disubmit.
- **\$makanan->id = \$id;** Mengatur ID makanan yang ingin diperbarui.
- **\$makanan->nama = \$_POST['nama'];** Mengambil nilai nama dari form yang diinput pengguna dan menyimpannya dalam objek **\$makanan**.
- **\$makanan->stok = \$_POST['stok'];** Mengambil nilai stok dari form dan menyimpannya dalam objek **\$makanan**.
- **\$makanan->harga = \$_POST['harga'];** Mengambil nilai harga dari form dan menyimpannya dalam objek **\$makanan**.
- **\$makanan->aksi = \$_POST['aksi'];** Mengambil nilai aksi (jika ada) dari form, meskipun dalam konteks ini penggunaannya tidak jelas tanpa konteks lebih lanjut.
- **if (\$makanan->update()) {**: Memanggil metode **update()** dari objek **\$makanan**, yang seharusnya melakukan pembaruan data makanan di database. Jika pembaruan berhasil, maka:
- **header("Location: dashboard.php");** Mengalihkan pengguna kembali ke halaman **dashboard.php** setelah pembaruan berhasil.
- **else { echo "Unable to update makanan."; }** Jika pembaruan gagal, menampilkan pesan kesalahan.

-Tampilan update makanan



F. Delete Data

- delete.php

```
<?php
include_once 'database/database.php';
include_once 'objects/makanan.php';

$database = new Database();
```

```

$db = $database->getConnection();
$makanan = new makanan($db);

// Check if ID is provided and a delete action is confirmed
if (isset($_GET['id']) && isset($_GET['confirm']) &&
$_GET['confirm'] == 'yes') {
    $makanan->id = $_GET['id'];

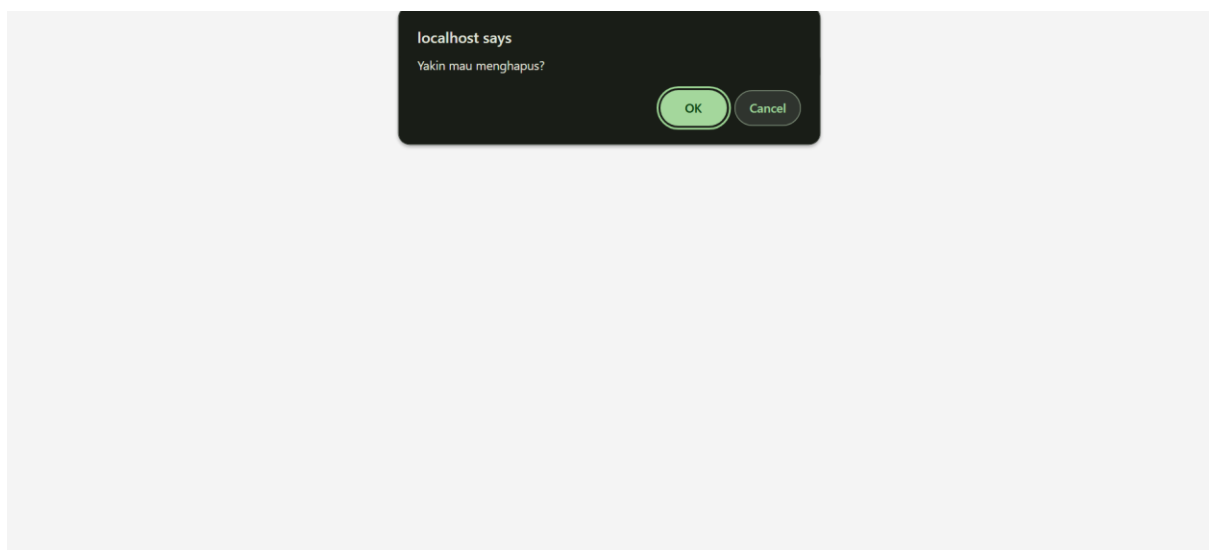
    if ($makanan->delete()) {
        header("Location: dashboard.php");
        exit();
    } else {
        echo "Unable to delete makanan.";
    }
} elseif (isset($_GET['id'])) {
    // Show confirmation prompt
    echo '<script>
        if (confirm("Yakin mau menghapus?")) {
            window.location.href = "?id=' . $_GET['id'] .
'&confirm=yes"; // Redirect with confirmation
        } else {
            window.location.href = "index.php"; // Redirect back
to index
        }
    </script>';
} else {
    echo "Invalid ID.";
}
?>

```

- **include_once 'database/database.php';** Mengimpor file database.php, yang berisi kelas Database untuk menangani koneksi ke database. **include_once** memastikan bahwa file hanya dimasukkan satu kali untuk mencegah kesalahan duplikasi.
- **include_once 'objects/makanan.php';** Mengimpor file makanan.php, yang berisi definisi kelas makanan untuk melakukan operasi CRUD pada data makanan.
- **\$database = new Database();** Membuat instansi baru dari kelas Database untuk mengelola koneksi database.
- **\$db = \$database->getConnection();** Memanggil metode **getConnection()** dari objek Database, yang mengembalikan objek koneksi database.
- **\$makanan = new makanan(\$db);** Membuat instansi baru dari kelas makanan, yang menerima objek koneksi database sebagai parameter untuk digunakan dalam operasi database.
- **if (isset(\$_GET['id']) && isset(\$_GET['confirm']) && \$_GET['confirm'] == 'yes')** : Memeriksa apakah parameter id dan confirm ada dalam URL, serta apakah nilai confirm adalah 'yes'. Ini menunjukkan bahwa pengguna telah mengonfirmasi penghapusan.

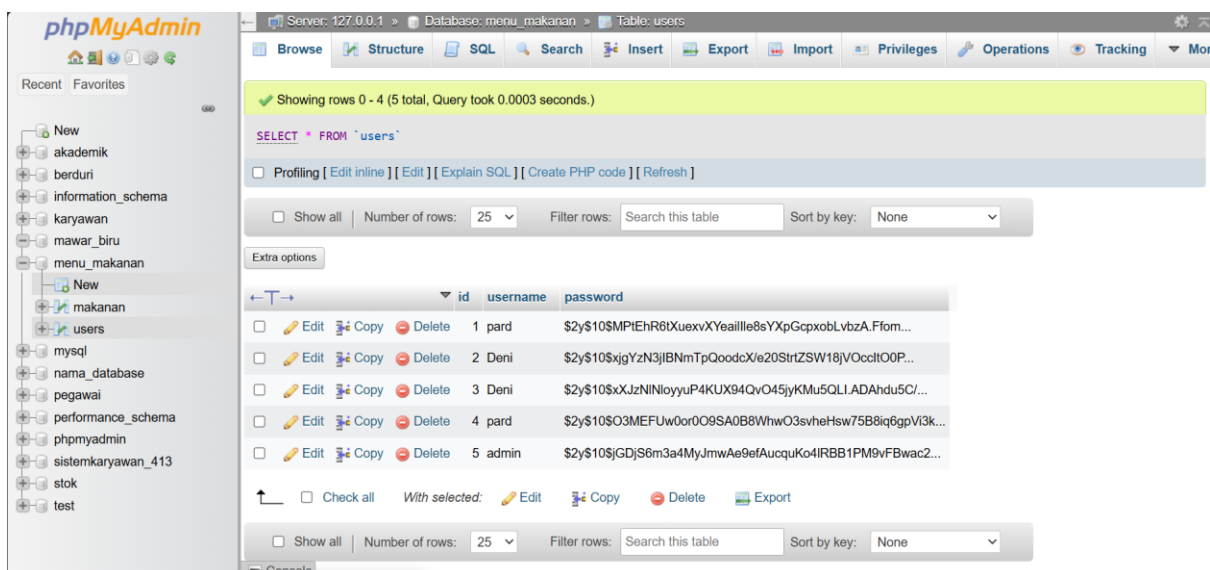
- **`$makanan->id = $_GET['id'];`**: Mengatur ID makanan yang akan dihapus dengan mengambilnya dari parameter URL id.
- **`if ($makanan->delete())`** {: Memanggil metode `delete()` dari objek makanan untuk menghapus data makanan yang sesuai dengan ID tersebut dari database.
- **Jika penghapusan berhasil:**
- **`header("Location: dashboard.php");`**: Mengalihkan pengguna ke halaman `dashboard.php` setelah penghapusan berhasil.
- **`exit();`**: Menghentikan eksekusi script lebih lanjut untuk memastikan bahwa tidak ada kode lain yang dijalankan setelah pengalihan.
- **Jika penghapusan gagal:**
- **`echo "Unable to delete makanan.";`**: Menampilkan pesan kesalahan jika proses penghapusan tidak berhasil.
- **`elseif (isset($_GET['id']))`** {: Memeriksa apakah parameter id ada dalam URL, tetapi confirm tidak ada, yang berarti pengguna perlu diminta konfirmasi untuk menghapus.
- **`echo '<script>...'`**: Menyisipkan kode JavaScript untuk menampilkan dialog konfirmasi kepada pengguna.
- **`if (confirm("Yakin mau menghapus?"))`** {: Jika pengguna mengklik "OK", URL akan diubah untuk menambahkan parameter `confirm=yes`, yang akan memicu penghapusan.
- **`window.location.href = "?id=" . $_GET['id'] . "&confirm=yes";`**: Mengarahkan pengguna kembali ke halaman ini dengan parameter konfirmasi ditambahkan ke URL.
- **`else { window.location.href = "index.php"; }`**: Jika pengguna mengklik "Cancel", pengguna akan dialihkan ke `index.php`.
- **`else { echo "Invalid ID.";`** {: Jika tidak ada parameter id dalam URL, tampilkan pesan kesalahan "Invalid ID."

-Tampilan menghapus data



II. Login

A. Membuat tabel baru dengan nama users



The screenshot shows the phpMyAdmin interface. On the left is a sidebar with a tree view of databases and tables. The main area displays the 'users' table structure and data. The table has three columns: 'id', 'username', and 'password'. There are five rows of data. The interface includes various tools like 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'More'. A console window is visible at the bottom.

	id	username	password
<input type="checkbox"/>	1	pard	\$2y\$10\$MPIEhR6IXuexvXYeaille8sYXpGcpxobLvbzA.Flom...
<input type="checkbox"/>	2	Deni	\$2y\$10\$xjgYzN3jIBNmTpQoodcX/e20StrZSW18jVOccltOOP...
<input type="checkbox"/>	3	Deni	\$2y\$10\$xXJzNlNllyyP4KUX94QvO45jyKM5QLIADAhdu5Cj...
<input type="checkbox"/>	4	pard	\$2y\$10\$O3MEFUw0or0O9SA0B8WhwO3svheHsw75B8iq6gpV3k...
<input type="checkbox"/>	5	admin	\$2y\$10\$GDjS6m3a4MyJmwAe9efAucquKo4IRBB1PM9vFBwac2...

Untuk meningkatkan keamanan dan manajemen akses dalam aplikasi, perlu dibuat tabel baru di dalam database oop3 dengan nama 'users', yang akan berfungsi untuk menyimpan data login pengguna. Tabel ini akan memiliki struktur yang terdiri dari empat kolom: "id", yang berfungsi sebagai kunci utama untuk mengidentifikasi setiap pengguna secara unik; 'username', yang menyimpan nama pengguna yang digunakan untuk login; dan 'password', yang menyimpan kata sandi pengguna dalam bentuk terenkripsi untuk melindungi informasi sensitif. Dengan adanya tabel 'users', aplikasi dapat mengelola otentikasi pengguna dengan lebih baik, memastikan hanya pengguna yang terdaftar yang

dapat mengakses sistem, serta memberikan lapisan keamanan tambahan terhadap data yang dikelola.

B. Buat file koneksi dengan nama Database.php

```
<?php
class Database{
    private $host="localhost";
    private $db_name="menu_makanan";
    private $username="root";
    private $password="";
    public $conn;

    public function getConnection(){
        $this->conn=null;

        try {
            $this->conn=new PDO("mysql:host=".$this->host."&dbname=".$this->db_name,
            $this->username,$this->password);

            $this->conn->exec("set names utf8");

        } catch(PDOException $exception) {
            echo"Connection error: ".$exception->getMessage();
        }
        return $this->conn;
    }
}
```

- **class Database:** Mendefinisikan kelas Database, yang akan digunakan untuk mengelola koneksi ke database.
- **private \$host="localhost";** Mendeklarasikan properti `$host` yang menyimpan alamat server database. Dalam hal ini, diatur ke localhost, yang berarti database berada di server yang sama dengan aplikasi.
- **private \$db_name="menu_makanan";** Mendeklarasikan properti `$db_name` yang menyimpan nama database yang ingin dihubungkan, yaitu menu_makanan.
- **private \$username="root";** Mendeklarasikan properti `$username` untuk nama pengguna yang digunakan untuk mengakses database. Di sini, diatur ke root, yang biasanya merupakan pengguna default pada instalasi MySQL.
- **private \$password="";** Mendeklarasikan properti `$password` untuk kata sandi pengguna. Di sini, tidak ada kata sandi yang diatur, yang mungkin berlaku untuk konfigurasi default MySQL.
- **public \$conn;** Mendeklarasikan properti publik `$conn`, yang akan menyimpan objek koneksi ke database.
- **public function getConnection();** Mendefinisikan metode publik `getConnection()` yang digunakan untuk menginisialisasi koneksi ke database.

- **`$this->conn = null;`**: Menetapkan nilai awal properti `$conn` menjadi null, memastikan bahwa koneksi baru akan dibuat.
- **`try { ... }`**: Memulai blok try untuk menangkap kemungkinan pengecualian (error) saat mencoba membuat koneksi.
- **`$this->conn = new PDO("mysql:host=".$this->host.";dbname=".$this->db_name, $this->username, $this->password);`**: Menciptakan objek PDO untuk menghubungkan ke database.
- **`mysql:host=`**: Menunjukkan jenis database yang digunakan (MySQL).
- **`dbname=`**: Menunjukkan nama database yang ingin dihubungkan.
- **`username` dan `password`**: Menyediakan kredensial untuk mengakses database.
- **`$this->conn->exec("set names utf8");`**: Menjalankan perintah SQL untuk mengatur karakter set koneksi ke `utf8`, yang mendukung karakter internasional dan memastikan bahwa data disimpan dengan benar dalam format UTF-8.
- **`catch(PDOException $exception) { ... }`**: Menangkap pengecualian `PDOException` jika terjadi kesalahan saat mencoba membuat koneksi.
- **`echo "Connection error: " . $exception->getMessage();`**: Menampilkan pesan kesalahan jika koneksi gagal, memberikan informasi tentang penyebab kesalahan.
- **`return $this->conn;`**: Mengembalikan objek koneksi `$conn`. Jika koneksi berhasil, ini akan menjadi objek PDO yang dapat digunakan untuk melakukan operasi database.
- **`}`**: Menandakan akhir dari definisi kelas `Database`.

D. Buat file dengan nama `index.php`

```
<?php
//index.php
require_once 'database/database.php';
require_once 'objects/user.php';

$database = new Database();
$db = $database->getConnection();
$user = new User($db);

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

    if($user->login($username, $password)) {
        header("Location: objects/home.php");
        exit();
    } else {
        $error_message = "username atau password salah!";
    }
}

?>

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

```

<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css" rel="stylesheet">
</head>
<body>
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <h3 class_exists="text-center">Login</h3>
            <?php if(isset($error_message)): ?>
                <div class="alert alert-danger">
                    <?php echo $error_message; ?>
                </div>
            <?php endif; ?>
            <form action="" method="POST">
                <div class="form-group">
                    <label>Username</label>
                    <input type="text" name="username" class="form-
control" required>
                </div>
                <div class="form-group">
                    <label>Password</label>
                    <input type="Password" name="password" class="form-
control" required>
                </div>
                <button type="submit" class="btn btn-primary btn-
block">Login</button>
                <a href="register.php" class="btn btn-warning btn-
block">Registrasi</a>
            </form>
        </div>
    </div>
</div>
</body>
</html>

```

- **require_once**: Menyertakan file PHP lain. Jika file tersebut sudah disertakan sebelumnya, tidak akan disertakan lagi.
- **'database/database.php'**: Mengimpor kelas Database yang digunakan untuk mengelola koneksi ke database.
- **'objects/user.php'**: Mengimpor kelas User yang bertanggung jawab untuk manajemen pengguna, termasuk login.
- **\$database = new Database()**; Membuat objek baru dari kelas Database untuk mengelola koneksi.
- **\$db = \$database->getConnection()**; Mengambil koneksi database dari objek Database yang telah dibuat.

- **\$user = new User(\$db);**: Membuat objek baru dari kelas User, mengoper koneksi database yang telah dibuat untuk digunakan dalam operasi pengguna.
- **if(\$_SERVER["REQUEST_METHOD"] == "POST")**: Memeriksa apakah permintaan yang diterima adalah permintaan POST, yang biasanya digunakan untuk mengirimkan data melalui formulir.
- **\$username = \$_POST['username'];**: Mengambil nilai dari input username yang dikirim melalui metode POST.
- **\$password = \$_POST['password'];**: Mengambil nilai dari input password yang dikirim melalui metode POST.
- **if(\$user->login(\$username, \$password))**: Memanggil metode login dari objek User, mengirimkan username dan password untuk verifikasi. Jika login berhasil:
- **header("Location: objects/home.php");**: Mengalihkan pengguna ke halaman beranda (home.php).
- **exit();**: Menghentikan eksekusi script lebih lanjut setelah pengalihan.
- **else**: Jika login gagal:
- **\$error_message = "username atau password salah!";**: Menetapkan pesan kesalahan yang akan ditampilkan di formulir jika login gagal.
- **<!DOCTYPE html>**: Mendefinisikan bahwa dokumen ini adalah HTML5.
- **<html lang="en">**: Menetapkan bahasa dokumen sebagai bahasa Inggris.
- **<head>**: Bagian dari dokumen yang berisi metadata, judul, dan tautan ke file CSS.
- **<meta charset="UTF-8">**: Mengatur karakter encoding ke UTF-8, yang mendukung banyak karakter internasional.
- **<title>Login</title>**: Menetapkan judul halaman yang ditampilkan di tab browser.
- **<link href="..." rel="stylesheet">**: Mengimpor stylesheet Bootstrap untuk styling komponen halaman.
- **<div class="container mt-5">**: Menggunakan kelas Bootstrap untuk membuat wadah dengan margin atas (mt-5).
- **<div class="row justify-content-center">**: Membuat baris dan mengatur agar konten di dalamnya terpusat.
- **<div class="col-md-4">**: Menentukan kolom dengan ukuran medium 4, membatasi lebar konten login.
- **<h3 class_exists="text-center">Login</h3>**: Judul halaman login. (Catatan: `class_exists` seharusnya `class`, tampaknya ada kesalahan pengetikan).
- **<?php if(isset(\$error_message)): ?>**: Memeriksa apakah ada pesan kesalahan yang telah ditetapkan.
- **<div class="alert alert-danger">**: Membuat div untuk menampilkan pesan kesalahan dengan styling Bootstrap.
- **<?php echo \$error_message; ?>**: Menampilkan pesan kesalahan jika ada.
- **<form action="" method="POST">**: Membuat formulir yang akan mengirimkan data ke halaman yang sama menggunakan metode POST.
- **<div class="form-group">**: Mengelompokkan label dan input untuk styling.
- **<label>Username</label>**: Label untuk input username.
- **<input type="text" name="username" class="form-control" required>**: Input teks untuk username. `required` menunjukkan bahwa input ini wajib diisi.
- **<label>Password</label>**: Label untuk input password.

- `<input type="Password" name="password" class="form-control" required>`: Input untuk password.
- `<button type="submit" class="btn btn-primary btn-block">Login</button>`: Tombol untuk mengirim formulir, dengan styling Bootstrap.
- `Registrasi`: Tautan ke halaman registrasi, ditampilkan sebagai tombol.
- `</body>`: Menandakan akhir dari bagian isi halaman.
- `</html>`: Menandakan akhir dari dokumen HTML.

F.buat file user.php

```
<?php
// classes/user.php
require_once './database/database.php';

class User {
    private $conn;
    private $table_name = "users";

    public function __construct($db) {
        $this->conn = $db;
    }

    /**
     * Mengecek kredensial login
     * @param string $username
     * @param string $password
     * @return boolean
     */

    public function login($username, $password) {
        // Query untuk mengambil data user yang sesuai
        // dengan username yang diinput
        $query = "SELECT * FROM " . $this->table_name . " WHERE
username = :username LIMIT 1";
        // Mempersiapkan statement untuk query di atas
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(":username", $username);
        $stmt->execute();

        // jika ada data user yang sesuai dengan username yang diinput
        if($stmt->rowCount() > 0) {
            // Ambil data user yang sesuai dengan username yang diinput
            $row = $stmt->fetch(PDO::FETCH_ASSOC);
            // Jika password yang diinput sesuai dengan password yang
            tersimpan di database
            if(password_verify($password, $row['password'])) {
```

```

        // Mulai sesi dan simpan username
        session_start();
        $_SESSION['username'] = $username;
        // Kembali true
        return true;
    }
}
//kembalikan false jika tidak ada data user yang sesuai dengan
username yang diinput atau password yang diinput tidak sesuai dengan
return false;
}

// Fungsi untuk menambahkan user baru
public function register($username, $password) {
    $query = "INSERT INTO " . $this->table_name .
        " (username, password) VALUES (:username, :password)";
    $stmt = $this->conn->prepare($query);

    // Enkripsi password sebelum disimpan ke database
    $hashed_password = password_hash($password,
PASSWORD_DEFAULT);

    $stmt->bindParam(":username", $username);
    $stmt->bindParam(":password", $hashed_password);

    if($stmt->execute()) {
        return true;
    }

    return false;
}
}
?>

```

- **require_once**: Mengimpor file PHP lain. Jika file tersebut sudah disertakan sebelumnya, tidak akan disertakan lagi.
- **'./database/database.php'**: Mengimpor kelas Database yang akan digunakan untuk mengelola koneksi ke database.
- **class User**: Mendefinisikan kelas User yang akan mengelola semua operasi terkait pengguna.
- **private \$conn**: Menyimpan koneksi database yang akan digunakan dalam metode kelas.
- **private \$table_name = "users"**: Mendefinisikan nama tabel di database yang berisi informasi pengguna.
- **public function __construct(\$db)**: Konstruktor yang dipanggil saat objek dari kelas User dibuat. Mengharuskan parameter \$db (koneksi database) saat instansiasi.
- **\$this->conn = \$db**: Menyimpan koneksi database yang diterima ke dalam properti \$conn.

- **public function login(\$username, \$password):** Mendefinisikan metode untuk memeriksa kredensial login pengguna.
- **\$query:** Membuat query SQL untuk mengambil semua kolom dari tabel users berdasarkan username yang diberikan. LIMIT 1 memastikan hanya satu baris yang diambil.
- **\$stmt = \$this->conn->prepare(\$query);** Mempersiapkan query untuk dieksekusi.
- **\$stmt->bindParam(":username", \$username);** Mengikat parameter :username ke variabel \$username untuk mencegah SQL Injection.
- **\$stmt->execute();** Menjalankan query yang telah disiapkan.
- **if(\$stmt->rowCount() > 0):** Memeriksa apakah ada baris yang dikembalikan dari query. Jika ada, berarti username yang dimasukkan valid.
- **\$row = \$stmt->fetch(PDO::FETCH_ASSOC);** Mengambil data pengguna sebagai array asosiatif.
- **if(password_verify(\$password, \$row['password'])):** Memeriksa apakah password yang dimasukkan cocok dengan password yang tersimpan di database. Password yang tersimpan biasanya telah di-hash.
- **Session_start();** Memulai sesi PHP. Ini diperlukan untuk menyimpan informasi pengguna saat login.
- **\$_SESSION['username'] = \$username;** Menyimpan username dalam variabel sesi untuk penggunaan selanjutnya.
- **return true;** Mengembalikan true jika login berhasil.
- **public function register(\$username, \$password):** Mendefinisikan metode untuk menambahkan pengguna baru.
- **\$query:** Membuat query SQL untuk memasukkan pengguna baru ke tabel users.
- **\$stmt = \$this->conn->prepare(\$query);** Mempersiapkan query untuk dieksekusi.
- **\$hashed_password = password_hash(\$password, PASSWORD_DEFAULT);** Meng-hash password menggunakan fungsi password_hash, yang akan mengenkripsi password untuk penyimpanan yang aman.
- **\$stmt->bindParam(":username", \$username);** Mengikat parameter username.
- **\$stmt->bindParam(":password", \$hashed_password);** Mengikat parameter :password dengan password yang telah di-hash.
- **if(\$stmt->execute()):** Menjalankan query untuk menambahkan pengguna. Jika berhasil, mengembalikan true.
- **return false;** Mengembalikan false jika terjadi kesalahan dalam eksekusi query.

G.buat file register.php

```
<?php
// register.php
require_once 'database/database.php';
require_once 'objects/user.php';

$databse = new Database();
$db = $databse->getConnection();
$user = new User($db);
```

```

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

    if($user->register($username, $password)) {
        $success_message = "Registrasi berhasil silahkan login.";
    } else {
        $error_message = "Register gagal. Username mungkin sudah
digunakan.";
    }
}
?>

<DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Registrasi</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
p.min.css" rel="stylesheet">
</head>
<body>
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <h3 class_exists="text-center">Registrasi</h3>
            <?php if(isset($error_message)): ?>
                <div class="alert alert-danger">
                    <?php echo $error_message; ?>
                </div>
            <?php endif; ?>
            <?php if(isset($success_message)): ?>
                <div class="alert alert-success">
                    <?php echo $success_message; ?>
                </div>
            <?php endif; ?>
            <form action="" method="POST">
                <div class="form-group">
                    <label>Username</label>
                    <input type="text" name="username" class="form-
control" required>
                </div>
                <div class="form-group">
                    <label>Password</label>
                    <input type="Password" name="password" class="form-
control" required>
                </div>
            </form>
        </div>
    </div>
</div>

```

```

        <button type="submit" class="btn btn-primary btn-
block">Daftar</button>
    </form>
    <p class="text-center mt-3">Sudah punya akun?
    <a href="index.php">Login di sini</a></p>
</div>
</div>
</div>
</body>
</html>

```

- `<?php`: Digunakan untuk mengimpor file PHP lain. Jika file tersebut sudah **require_once** disertakan sebelumnya, tidak akan disertakan lagi.
- `'database/database.php'`: Mengimpor kelas Database yang mengelola koneksi ke database.
- `'objects/user.php'`: Mengimpor kelas User, yang berisi metode untuk registrasi dan login pengguna.
- `$database = new Database();`: Membuat objek baru dari kelas Database.
- `$db = $database->getConnection();`: Mengambil koneksi database dengan memanggil metode `getConnection()` dari objek Database.
- `$user = new User($db);`: Membuat objek baru dari kelas User, mengoper koneksi database yang telah dibuat.
- `if($_SERVER["REQUEST_METHOD"] == "POST")`: Memeriksa apakah formulir telah dikirim dengan metode POST.
- `$username = $_POST['username'];`: Mengambil nilai username dari data yang dikirimkan melalui formulir.
- `$password = $_POST['password'];`: Mengambil nilai password dari data yang dikirimkan melalui formulir.
- `if($user->register($username, $password))`: Memanggil metode register dari objek User untuk mendaftar pengguna baru. Metode ini mengembalikan true jika registrasi berhasil, dan false jika gagal.
- `$success_message`: Jika registrasi berhasil, menyimpan pesan sukses.
- `$error_message`: Jika registrasi gagal (misalnya, username sudah digunakan), menyimpan pesan kesalahan.
- `<DOCTYPE html>`: Menyatakan bahwa ini adalah dokumen HTML5 (seharusnya `<!DOCTYPE html>`).
- `<html lang="en">`: Memulai elemen HTML dengan atribut lang yang menunjukkan bahasa dokumen.
- `<head>`: Bagian yang berisi metadata dokumen.
- `<meta charset="UTF-8">`: Menetapkan pengkodean karakter untuk dokumen.
- `<title>Registrasi</title>`: Menentukan judul halaman yang ditampilkan di tab browser.
- `<link>`: Menyertakan file CSS Bootstrap dari CDN untuk styling halaman.
- `<div class="container mt-5">`: Memulai kontainer dengan margin atas.
- `<div class="row justify-content-center">`: Menggunakan grid Bootstrap untuk menyusun konten agar terpusat.
- `<div class="col-md-4">`: Mengatur lebar kolom untuk formulir.

- `<?php if(isset($error_message)): ?>`: Memeriksa apakah variabel `$error_message` ada. Jika ada, tampilkan pesan kesalahan dalam kotak alert.
- `<div class="alert alert-danger">`: Menggunakan kelas Bootstrap untuk styling pesan kesalahan.
- `<?php echo $error_message; ?>`: Menampilkan pesan kesalahan.
- Sama untuk `$success_message`, tetapi dengan kelas alert `alert-success` untuk pesan sukses.
- `<form action="" method="POST">`: Membuat formulir dengan metode POST. `action=""` berarti mengirim data ke halaman yang sama.
- **Input untuk username dan password**: Menggunakan kelas `form-control` dari Bootstrap untuk styling.
- **Tombol Daftar**: Tombol untuk mengirimkan formulir.
- `<p class="text-center mt-3">Sudah punya akun? Login di sini</p>`: Menampilkan pesan untuk pengguna yang sudah memiliki akun, dengan tautan ke halaman login (`index.php`).
- `</div>, </body>, </html>`: Menutup semua elemen yang telah dibuka sebelumnya. Ini mencakup penutupan `div` kontainer dan elemen HTML.

III. Hasil

a. Dashboard

Login

Username

Password

Login

Registrasi

b. Register

Registrasi

Username

Password

Daftar

Sudah punya akun? [Login di sini](#)

c.Registrasi Berhasil

Registrasi

Registrasi berhasil silahkan login.

Username

Password

Daftar

Sudah punya akun? [Login di sini](#)

d.Login berhasil



Deni



Puput



Amel



Radhit

-Tampilan menu makanan

Menu Makanan

[Tambah Makanan](#)

id	nama	stok	harga	aksi
1	nasi ayam pedas	80	12000	Ubah Hapus
5	seblak	50	15000	Ubah Hapus
6	bakso urat	50	20000	Ubah Hapus
7	mie ayam	50	11000	Ubah Hapus
8	sup iga	50	25000	Ubah Hapus
9	nasi ayam geprek	50	13000	Ubah Hapus
11	ikan bakar	50	21000	Ubah Hapus
12	sup kornet	50	16000	Ubah Hapus

e.Login gagal

Login

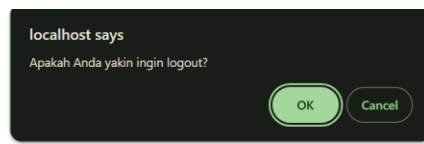
username atau password salah!

Username

Password

[Login](#)[Registrasi](#)

f.Log out



Login

Username

Password

Login

Registrasi

