Laporan Praktikum CRUD

Dan Login PHP Navite

Disusun oleh:

1. Elza Widyastuti
2. Fadhillah Krismawati
3. Sevi Azalia Ramadhani
4. Sylvia Elvira Sarma

Kelas: XI RPL3

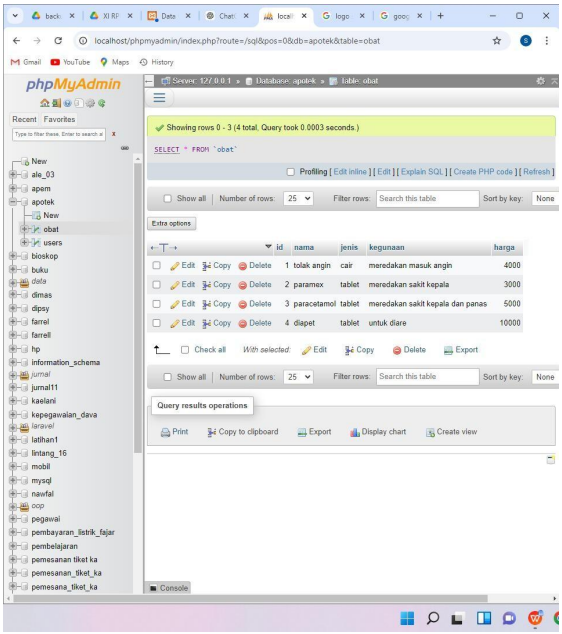
SMK Texmaco Semarang

Semarang 30/10/2024

Tahap:

1. Obat

A. Membuat database dan tabel



Penjelasan:

Langkah 1: Masuk ke phpMyAdmin

1. Buka phpMyAdmin melalui browser Anda, biasanya dapat diakses melalui `http://localhost/phpmyadmin` jika menggunakan server lokal (XAMPP atau WAMP).

Langkah 2: Membuat Database `apotek`

1. Setelah berhasil masuk, pada bagian atas klik tab "Database".

2. Pada kolom "Create database" (atau "Buat basis data" jika menggunakan bahasa Indonesia), masukkan nama database: `apotek`.

3. Klik tombol "Create" (atau "Buat").

Database `apotek` sekarang sudah berhasil dibuat.

Langkah 3: Membuat Tabel `obat` di Database `apotek`

1. Klik nama database `apotek` di sebelah kiri layar untuk membuka database.

2. Setelah masuk ke database `apotek`, klik "New" atau "Baru" untuk membuat tabel baru.

3. Masukkan nama tabel, yaitu `obat`.

4. Pada kolom jumlah kolom (jumlah field), masukkan angka `5` untuk kolom yang akan kita buat: `id`, `nama`, `jenis`, `kegunaan`, dan `harga`.

5. Klik "Go"atau "Jalankan" untuk melanjutkan.

Langkah 4: Menentukan Struktur Kolom untuk Tabel `obat`

1. id:
   - Name: `id`
   - Type: `INT`
   - Length/Values: 11
   - Index: Pilih `PRIMARY` untuk menetapkan sebagai primary key.

- A_I (Auto Increment): Centang untuk menambahkan fitur auto-increment.

2. nama:
   - Name: `nama`
   - Type: `VARCHAR`
   - Length/Values : 100 (Anda bisa mengatur panjang karakter sesuai kebutuhan, di sini kita menggunakan 100).
3. jenis:
   - Name: `jenis`
   - Type: `VARCHAR`
   - Length/Values: 50
4. kegunaan:
   - Name: `kegunaan`
   - Type: `TEXT`
   - Length/Values: (Tidak perlu mengisi karena tipe TEXT tidak membutuhkan panjang)
5. harga:
   - Name: `harga`
   - Type: `DECIMAL`
   - Length/Values: `10,2` (10 total digit, dengan 2 angka desimal untuk harga)

Langkah 5: Menyimpan Tabel `obat`

Setelah selesai mengisi semua kolom, scroll ke bawah dan klik tombol "Save" (atau "Simpan"). Sekarang tabel `obat` telah berhasil dibuat dalam database `apotek`.

Struktur Tabel `obat` yang Telah Dibuat:

- id: Kolom INT, Primary Key, Auto Increment
- nama: Kolom VARCHAR dengan panjang 100 karakter
- jenis: Kolom VARCHAR dengan panjang 50 karakter
- kegunaan: Kolom VARCHAR dengan panjang 100 karakter
- harga: Kolom DECIMAL dengan format 10,2

Anda sekarang telah berhasil membuat database `apotek` dengan tabel `obat` di phpMyAdmin. Tabel ini siap untuk diisi data mengenai obat-obatan yang diperlukan.

B. Membuat koneksi dengan database

```php
<?php
class Database{
    private $host="localhost";
    private $db_name="apotek";

      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;
  }
}
?>
```

Penjelasan:

Kode di atas adalah kelas PHP bernama `Database` yang digunakan untuk membuat koneksi ke database MySQL. Berikut adalah penjelasan detail dari setiap bagian kode:

1. Membuat Kelas `Database`
```
<?php
class Database {
    private $host = "localhost";
    private $db_name = "apotek";
    private $username = "root";
    private $password = "";
    public $conn;
```

- `$host`: Menyimpan alamat host database, yaitu `localhost`. Biasanya `localhost` digunakan saat pengembangan di server lokal.
- `$db_name`: Menyimpan nama database yang ingin diakses, dalam contoh ini adalah `apotek`.
- `$username`: Menyimpan username untuk mengakses database, yaitu `root`.
- `$password`: Menyimpan password untuk username `root`. Dalam contoh ini, password dikosongkan karena default MySQL pada XAMPP atau WAMP biasanya tidak memiliki password.
- `$conn`: Properti publik yang akan menyimpan koneksi database. Properti ini bisa diakses dari luar kelas, dan akan diisi objek koneksi (`PDO`) jika berhasil terhubung ke database.

2. Membuat Fungsi `getConnection`
```
<?php
public function getConnection() {
    $this->conn = null;
```
```

- `getConnection`: Fungsi ini berfungsi untuk membuat koneksi ke database. Saat fungsi dipanggil, properti `$conn` akan diinisialisasi menjadi `null`.

3. Mencoba Membuat Koneksi dengan `try-catch`
```php
```
```

```
try {
    $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this->username, $this->password);
    $this->conn->exec("set names utf8");
```
- `try`: Blok `try` ini digunakan untuk mencoba kode yang berpotensi menimbulkan error.
- `new PDO(...)`: Membuat koneksi ke database menggunakan **PDO (PHP Data Objects). Konstruksi ini mengatur:
  - `"mysql:host=" . $this->host . ";dbname=" . $this->db_name` untuk menentukan host dan nama database.
  - `$this->username` dan `$this->password` sebagai informasi login database.
- `set names utf8`: Mengatur karakter encoding menjadi UTF-8 untuk memastikan karakter yang diambil atau disimpan dalam database ditampilkan dengan benar.

4. Menangani Error Koneksi dengan `catch`
```php
} catch(PDOException $exception) {
    echo "Connection error:" . $exception->getMessage();
}
```
- `catch`: Jika terjadi error saat mencoba koneksi, blok `catch` ini akan menangkap error tersebut.
- `$exception->getMessage()`: Menampilkan pesan error yang lebih detail yang dihasilkan oleh PDO.

5. Mengembalikan Koneksi
```php
return $this->conn;
```
- Kode ini mengembalikan nilai `$conn`, yaitu objek koneksi PDO yang sudah berhasil dihubungkan ke database atau `null` jika koneksi gagal.

Penggunaan Kelas `Database`
Kelas ini bisa digunakan untuk koneksi database dengan langkah berikut:
1. Inisialisasi objek `Database`: `$database = new Database();`
2. Memanggil fungsi `getConnection` untuk membuat koneksi: `$conn = $database->getConnection();`

Contoh Penggunaan
```php
<?php
$database = new Database();
```

C. Menampilkan Data

```php
<?php

include_once 'database/database.php';
include_once 'objects/obat.php';
```

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

```php
$obat = new obat($db);
```

```
$stmt = $obat->read();
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Data Obat</title>
    <style>
                /* style.css */
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
```

```
        h1 {
            color: #333;
        }
```

```
        a {
            text-decoration: none;
            color: #4B5390;
        }
```

```
        a:hover {
            text-decoration: underline;
        }
```

```
        table {
```

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

```
th, td {
    padding: 12px;
    text-align: left;
        border-bottom: 1px solid #ddd;
        }
```

```
        th {
            background-color: #2F4156;
            color: white;
        }
```

```
        tr:hover {
            background-color: #f1f1f1;
        }
```

```
        td a {
            margin-right: 10px;
            color: #dc3545; /* Bootstrap danger color */
        }
```

```
        td a:hover {
            color: #c82333; /* Darker shade on hover */
        }
```

```
        .button-container
            { margin-top:
            20px;
```

```
        .btn {
            display: inline-block;
            padding: 10px 15px;
            margin-right: 10px;
            color: white;
      background-color: #007bff; /* Bootstrap primary color */
            border: none;
            border-radius: 5px;
            text-align: center;
            text-decoration: none;
            cursor: pointer;
        }
```

```
        .btn-logout {
            background-color: #dc3545; /* Bootstrap danger
color */
        }
```

```
        .btn:hover {
            opacity: 0.8;
        }
    </style>
</head>
<body>
    <h1>Data Obat</h1>
    <div class="button-container">
        <a href="create.php" class="btn">Tambah Obat</a>
        <a href="logout.php" class="btn btn-logout">Logout</a>
    </div>
    <table>
        <tr>
            <th>ID</th>
```

```
                        <th>Nama</th>
        <th>Jenis</th>
        <th>Kegunaan</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>{$jenis}</td>";
        echo "<td>{$kegunaan}</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>
```

Penjelasan :

Kode di atas adalah aplikasi web PHP yang menampilkan data obat dari database apotek menggunakan antarmuka HTML dengan PHP. Berikut adalah penjelasan detail dari setiap bagian kode.

1.

```
php
include_once 'database/database.php';include_once
'objects/obat.php';
```

- include_once digunakan untuk memasukkan file database.php dan obat.php.
- `database.php`: File ini berisi kelas Database untuk menghubungkan aplikasi dengan database.
- `obat.php`: File ini berisi kelas obat yang menyediakan fungsi seperti membaca, mengupdate, dan menghapus data obat dari database.

2. Membuat Koneksi Database

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

- Membuat objek Database dan memanggil metode getConnection() untuk mendapatkan koneksi database.
- Objek koneksi disimpan dalam variabel $db, yang akan digunakan untuk melakukan operasi database.

3. Membuat Objek obat dan Memanggil Fungsi read

```
php
$obat = new obat($db);$stmt = $obat->read();
```

- Membuat objek obat dengan mengirimkan $db sebagai parameter. Ini memungkinkan objek obat untuk berinteraksi dengan database.
- Memanggil metode read() pada objek obat untuk mengambil semua data obat. Hasil query disimpan dalam variabel $stmt.

4. HTML Struktur Utama

```
html
<!DOCTYPE html><html
lang="en"><head>...</head><body>...</body></html>
```

- Menyusun struktur HTML dasar untuk menampilkan data obat.

5. Style CSS

```
css
Copy code
<style>
    body { font-family: Arial, sans-serif; ... }
</style>
```

- Kode CSS ini mengatur tampilan halaman, seperti warna, ukuran font, latar belakang, dan hover untuk tabel dan tombol.

6. Judul dan Tombol Aksi

```
html
<h1>Data Obat</h1><div class="button-container">
    <a href="create.php" class="btn">Tambah Obat</a>
    <a href="logout.php" class="btn btn-logout">Logout</a></div>
```

- Menampilkan judul halaman **"Data Obat"**.
- Menyediakan dua tombol aksi:
    - **Tambah Obat**: Mengarahkan pengguna ke halaman create.php untuk menambah data obat baru.
    - **Logout**: Mengarahkan pengguna ke `logout.php` untuk keluar dari aplikasi.

7. Tabel untuk Menampilkan Data Obat

```
html
<table>
    <tr>
        <th>ID</th>
```

```
        <th>Nama</th>
        <th>Jenis</th>
        <th>Kegunaan</th>
        <th>Harga</th>
        <th>Aksi</th>
    </tr>
```

- Mendefinisikan tabel HTML dengan header kolom:
    - **ID**: ID unik obat (dari primary key).
    - **Nama**: Nama obat.
    - **Jenis**: Jenis obat.
    - **Kegunaan**: Kegunaan atau manfaat obat.
    - **Harga**: Harga obat.
    - **Aksi**: Kolom untuk tautan Ubah dan Hapus.

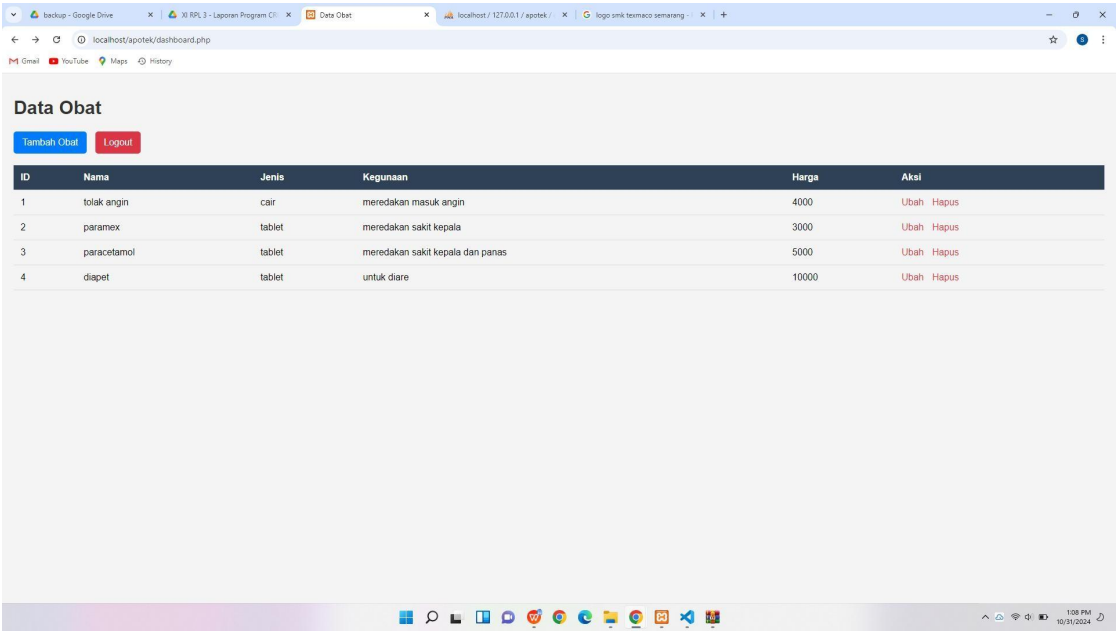
8. PHP untuk Menampilkan Data Obat dalam Tabel

php

```
<?phpwhile ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    extract($row);
    echo "<tr>";
    echo "<td>{$id}</td>";
    echo "<td>{$nama}</td>";
    echo "<td>{$jenis}</td>";
    echo "<td>{$kegunaan}</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>";
}?>
```

- `while ($row = $stmt->fetch(PDO::FETCH_ASSOC))`: Melakukan iterasi untuk setiap baris hasil dari query read().
    - `$stmt->fetch(PDO::FETCH_ASSOC)` mengambil satu baris hasil sebagai array asosiatif.
- `extract($row);`: Mengambil nilai kolom dari baris yang sedang diproses dan membuatnya sebagai variabel, misalnya `$id, $nama, $jenis,` dll.
- `echo "<tr>...</tr>";`: Menampilkan baris tabel (<tr>) untuk setiap obat.
    - `{$id}, {$nama},` **dll.**: Menyisipkan data ke dalam kolom tabel.
    - `<a href='update.php?id={$id}'>Ubah</a>`: Tautan untuk mengedit obat berdasarkan id.
    - `<a href='delete.php?id={$id}'>Hapus</a>`: Tautan untuk menghapus obat berdasarkan id.

- hasilnya



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 Obat</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 Obat</h1>
        <form action="create_action.php"
method="post">
            <label for="nama">Nama:</label>
            <input type="text" name="nama"
id="nama" required>

            <label for="jenis">Jenis:</label>
            <input type="text" name="jenis"
id="jenis" required>
```

```
            <label
for="kegunaan">Kegunaan:</label>
            <input type="text"
name="kegunaan" id="kegunaan" required>
```

```
            <label for="harga">Harga:</label>
```

```
        <input type="number" name="harga"
id="harga" required>
```

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

Penjelasan:

Kode HTML di atas menampilkan halaman form untuk menambahkan data obat baru ke database. Berikut adalah penjelasan detail dari setiap bagian kode:

1. Struktur HTML Utama

html

```
<!DOCTYPE html><html
lang="en"><head>...</head><body>...</body></html>
```

- `<!DOCTYPE html>`: Menentukan bahwa dokumen ini adalah dokumen HTML5.
- `<html lang="en">`: Tag <html> dengan atribut `lang="en"` mengindikasikan bahasa utama dokumen ini adalah bahasa Inggris.
- `<head>...</head>`: Bagian <head> berisi metadata, judul halaman, dan gaya CSS untuk halaman ini.
- `<body>...</body>`: Bagian <body> berisi konten yang akan ditampilkan kepada pengguna, yaitu form untuk menambahkan data obat.

2. Metadata dan Judul

html

```
<meta charset="UTF-8"><meta name="viewport"
content="width=device-width, initial-scale=1.0"><title>Tambah
Obat</title>
```

- `<meta charset="UTF-8">`: Menentukan bahwa dokumen ini menggunakan karakter encoding UTF-8.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Mengatur tampilan agar responsive pada berbagai ukuran layar.
- `<title>Tambah Obat</title>`: Menampilkan "Tambah Obat" sebagai judul halaman di tab browser.

3. CSS Inline untuk Styling

css

```
<style>
    body { font-family: Arial, sans-serif; background-color:
#F5F2F0; ... }
    h1 { color: #333; }
    form { background-color: #BAD0E7; border-radius: 5px; ... }
    label { display: block; margin-bottom: 8px; font-weight:
bold; }
    input[type="text"], input[type="number"] { width: 95%;
padding: 10px; ... }
    input[type="submit"] { background-color: #0099FF; color:
white; ... }
</style>
```

- body: Mengatur font, latar belakang, margin, dan padding.
- h1: Mengatur warna teks judul halaman.
- form: Memberikan warna latar, border-radius, dan bayangan pada form agar terlihat lebih rapi.
- label: Mengatur label form dengan display: block untuk menampilkan satu per baris, dan menambah margin-bottom agar ada spasi antara label dan input.
- `input[type="text"]` **dan** `input[type="number"]`: Mengatur tampilan input teks dan angka, lebar, padding, dan border-radius.
- `input[type="submit"]`: Mengatur tombol submit dengan warna latar, warna teks putih, padding, dan hover effect.

4. Judul Halaman

html
Copy code
```
<h1 align="center">Tambah Obat</h1>
```

- Menampilkan judul "Tambah Obat" di bagian atas halaman dan menengahkan dengan align="center".

5. Form Tambah Obat

html
```
<form action="create_action.php" method="post">
```

- `<form>`: Tag form untuk mengumpulkan data input pengguna.
- `action="create_action.php"`: Mengarahkan data form ke file `create_action.php` untuk diproses ketika pengguna menekan tombol submit.
- `method="post"`: Mengirim data melalui metode POST, yang lebih aman dibanding GET.

5.1 Input Nama Obat

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

- `<label for="nama">Nama:</label>`: Label untuk input nama obat.
- `<input type="text" name="nama" id="nama" required>`: Input teks untuk mengisi nama obat. Atribut required membuat input ini wajib diisi.

5.2 Input Jenis Obat

html
```
<label for="jenis">Jenis:</label><input type="text" name="jenis"
id="jenis" required>
```

- `<label for="jenis">Jenis:</label>`: Label untuk input jenis obat.
- `<input type="text" name="jenis" id="jenis" required>`: Input teks untuk mengisi jenis obat. Atribut required membuat input ini wajib diisi.

5.3 Input Kegunaan Obat

html
```
<label for="kegunaan">Kegunaan:</label><input type="text"
name="kegunaan" id="kegunaan" required>
```

- `<label for="kegunaan">Kegunaan:</label>`: Label untuk input kegunaan obat.
- `<input type="text" name="kegunaan" id="kegunaan" required>`: Input teks untuk mengisi kegunaan obat. Atribut required membuat input ini wajib diisi.

5.4 Input Harga Obat

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

- `<label for="harga">Harga:</label>`: Label untuk input harga obat.
- `<input type="number" name="harga" id=`"harga" required>: Input angka untuk mengisi harga obat. Atribut required membuat input ini wajib diisi.

6. Tombol Submit

html
```
<input type="submit" value="Simpan">
```

- `<input type="submit">`: Tombol untuk mengirim data form ke create_action.php.
- `value="Simpan"`: Menentukan teks pada tombol submit adalah "Simpan".

- Create_action.php

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

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

```php
$obat=new obat($db);
```

```php
$obat->nama=$_POST['nama'];
$obat->jenis=$_POST['jenis'];
$obat->kegunaan=$_POST['kegunaan'];
$obat->harga=$_POST['harga'];
```

```php
if($obat->create()){
    echo "Obat berhasil ditambahkan.";
} else {
    echo"Gagal menambahkan Obat.";
}
?>
```

Penjelasan:

ode PHP di atas berfungsi untuk menyimpan data obat yang diinputkan oleh pengguna ke dalam database. Skrip ini akan dipanggil setelah pengguna mengisi form tambah obat dan mengirimkannya. Mari kita bahas bagian-bagian kodenya secara detail.

1. Memasukkan File yang Diperlukan

php
Copy code
`include_once 'database/database.php';include_once 'objects/obat.php';`

- `include_once 'database/database.php'`: Menyertakan file database.php, yang berisi kelas Database. Kelas ini bertanggung jawab untuk mengatur koneksi ke database.
- `include_once 'objects/obat.php'`: Menyertakan file obat.php, yang berisi kelas obat untuk mengelola data obat dan operasinya, termasuk operasi tambah, edit, hapus, dan baca data dari database.

2. Membuat Koneksi Database

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

- `$database = new Database();`: Membuat objek baru dari kelas Database.
- `$db = $database->getConnection();`: Memanggil metode getConnection() dari kelas Database untuk mendapatkan koneksi ke database. Koneksi ini disimpan di variabel $db dan nantinya akan digunakan oleh objek obat.

3. Membuat Objek obat

php
```
$obat = new obat($db);
```

- Membuat objek obat dengan mengirimkan $db sebagai parameter. Dengan ini, objek obat dapat mengakses koneksi database dan melakukan berbagai operasi, seperti menambah, membaca, mengedit, atau menghapus data obat.

4. Mengambil Data dari Form

php
```
$obat->nama = $_POST['nama'];$obat->jenis =
$_POST['jenis'];$obat->kegunaan = $_POST['kegunaan'];$obat->harga
= $_POST['harga'];
```

- `$_POST`: Variabel superglobal yang menyimpan data yang dikirimkan melalui form menggunakan metode POST. Setiap nilai input dari form diambil dan disimpan ke dalam properti pada objek obat.
- `$obat->nama = $_POST['nama'];`: Mengambil nilai nama yang dikirim dari form, dan menyimpannya ke properti nama pada objek obat.
- **Proses serupa dilakukan untuk** jenis, kegunaan, dan harga.

5. Menyimpan Data ke Database

php
```
if($obat->create()){
 echo "Obat berhasil ditambahkan.";
} else {
 echo "Gagal menambahkan Obat.";
}
```

- `if($obat->create())`: Memanggil metode create() dari objek obat untuk menyimpan data obat yang baru ke dalam database.
    - Jika metode create() mengembalikan nilai true, berarti data obat berhasil disimpan, dan pesan "Obat berhasil ditambahkan." akan ditampilkan.
    - Jika metode create() mengembalikan false, berarti terjadi kesalahan dalam proses penyimpanan, dan pesan "Gagal menambahkan Obat." akan muncul.

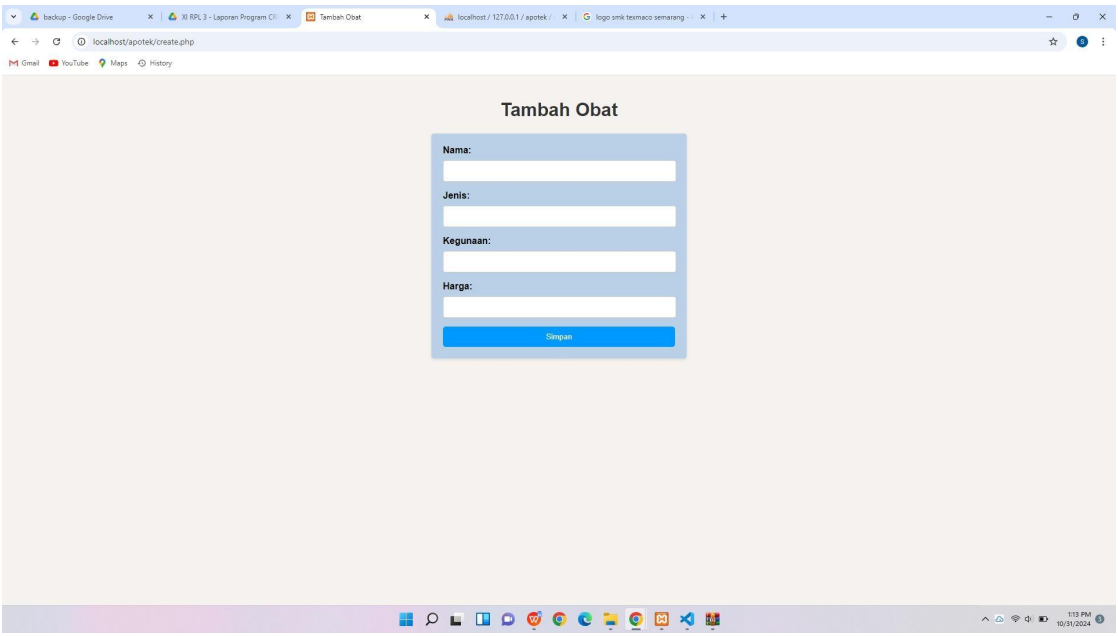
Penjelasan Tambahan tentang Metode create() pada Kelas obat

Agar skrip ini berfungsi dengan baik, kelas obat pada file obat.php harus memiliki metode create(). Metode ini biasanya berisi:

- **Query SQL**: Query INSERT untuk menambahkan data baru ke tabel obat.

- **Eksekusi Query**: Menggunakan koneksi database ($db) untuk menjalankan query tersebut.
- **Pengembalian Nilai**: Metode create() mengembalikan nilai true jika data berhasil disimpan, dan false jika terjadi kesalahan.

- tampilan tambah data



E. Edit Data

- Update.php

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

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

```php
$obat = new obat($db);
```

```
// Check if ID is provided
```

```php
if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $stmt = $obat->readOne($id);
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

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

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

```php
// Handle form submission
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    $obat->id = $id;
    $obat->nama = $_POST['nama'];
    $obat->jenis = $_POST['jenis'];
    $obat->kegunaan = $_POST['kegunaan'];
    $obat->harga = $_POST['harga'];
```

```php
    if ($obat->update()) {
```

```
            header("Location: index.php");
        } else {
            echo "Unable to update obat.";
        }
}
?>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Obat</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 Obat</h1>
```

```
        <form method="post">
                <label>Nama:</label>
                <input type="text" name="nama"
value="<?php echo $nama; ?>" required>

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

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

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

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

```
</html>
```

Penjelasan:

Skrip PHP ini memungkinkan pengguna untuk mengedit data obat yang sudah ada di database. Berikut penjelasan detailnya:

1. Menyertakan File yang Diperlukan

php
```
include_once 'database/database.php';include_once
'objects/obat.php';
```

- `database.php`: File yang mengatur koneksi database melalui kelas Database.
- `obat.php`: File yang mengandung kelas obat, yang menangani operasi CRUD untuk tabel obat.

2. Membuat Koneksi Database dan Objek obat

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

- Membuat objek Database dan memanggil metode getConnection() untuk membuka koneksi ke database.
- Membuat objek obat, yang memungkinkan kita mengakses metode CRUD untuk tabel obat menggunakan koneksi $db.

3. Mengecek ID Obat yang Akan Diedit

php
```
if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $stmt = $obat->readOne($id);
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

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

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

- `if (isset($_GET['id']))`: Mengecek apakah id obat sudah disediakan melalui parameter URL (`$_GET['id']`).
- **Mengambil Data Obat**:
    - Menggunakan metode readOne($id) dari kelas obat untuk mengambil data obat dengan id yang diberikan.
    - Data yang diambil disimpan di variabel `$row` dan diekstrak menggunakan fungsi extract() untuk membuat variabel individual (`$nama, $jenis, $kegunaan, $harga`) yang akan diisi sebagai nilai default pada form.

- **Validasi Obat Ditemukan**: Jika readOne tidak menemukan data obat, halaman akan menampilkan pesan "Obat not found" dan proses dihentikan.

4. Memproses Form Update

php
```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $obat->id = $id;
    $obat->nama = $_POST['nama'];
    $obat->jenis = $_POST['jenis'];
    $obat->kegunaan = $_POST['kegunaan'];
    $obat->harga = $_POST['harga'];

    if ($obat->update()) {
        header("Location: index.php");
    } else {
        echo "Unable to update obat.";
    }
}
```

- `if ($_SERVER["REQUEST_METHOD"] == "POST")`: Mengecek apakah form telah disubmit dengan metode POST.
- **Mengambil Input dari Form**:
    - Mengisi properti id, nama, jenis, kegunaan, dan harga dari objek obat dengan data yang diambil dari form `($_POST)`.
- **Memperbarui Data Obat**:
    - Memanggil metode update() dari kelas obat. Jika data berhasil diperbarui, pengguna diarahkan ke halaman index.php menggunakan fungsi header().
    - Jika pembaruan gagal, pesan "Unable to update obat." akan ditampilkan.

5. Tampilan HTML Form Edit

html
```
<!DOCTYPE html><html><head>
    <title>Edit Obat</title>
    <style>
        /* Gaya CSS untuk tampilan */
    </style></head><body>
    <h1>Edit Obat</h1>
    <form method="post">
        <label>Nama:</label>
        <input type="text" name="nama" value="<?php echo
$nama; ?>" required>

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

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

        <label>Harga:</label>
```

```
        <input type="number" name="harga" value="<?php echo
$harga; ?>" required>

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

- **Form Edit**:
    - Setiap elemen form (input) memiliki nilai value yang diambil dari variabel PHP `($nama, $jenis, $kegunaan, $harga)` yang telah diekstrak dari hasil query database. Ini menampilkan data awal dari obat yang akan diedit.

- **Atribut** required: Memastikan semua input form wajib diisi sebelum form dikirim.
- **Input** `type="submit"`: Tombol untuk mengirimkan form. Jika ditekan, data yang telah diedit akan diproses pada blok if `($_SERVER["REQUEST_METHOD"] == "POST")` di atas.

F. Hapus data

- Delete.php

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


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

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

```php
    if ($obat->delete())
        { header("Location:
```

```
        exit();
    } else {
        echo "Unable to delete obat.";
    }
} 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.";
}
?>
```

Penjelasan::

Kode PHP di atas menangani penghapusan data obat dari database. Proses penghapusan dilakukan setelah adanya konfirmasi dari pengguna untuk memastikan tindakan tersebut. Berikut ini adalah penjelasan rinci dari setiap bagian kode:

1. Menyertakan File yang Diperlukan

php
```
include_once 'database/database.php';include_once
'objects/obat.php';
```

- database.php: File ini memuat kelas Database yang bertugas untuk mengatur koneksi ke database.
- obat.php: File ini memuat kelas obat, yang berisi fungsi CRUD (Create, Read, Update, Delete) untuk tabel obat.

2. Membuat Koneksi Database dan Objek obat

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

- `$database = new Database();`: Membuat objek Database untuk membuka koneksi database.
- `$db = $database->getConnection();`: Mengambil koneksi yang aktif melalui metode `getConnection()`.
- `$obat = new obat($db);`: Membuat objek obat yang menggunakan koneksi $db. Objek ini nantinya dipakai untuk melakukan operasi penghapusan data dari database.

3. Memproses Permintaan Penghapusan

Kode selanjutnya mengecek apakah terdapat ID obat yang ingin dihapus dan apakah pengguna telah mengonfirmasi tindakan tersebut.

3.1 Mengecek ID dan Konfirmasi Penghapusan

```
php
if (isset($_GET['id']) && isset($_GET['confirm']) &&
$_GET['confirm'] == 'yes') {
    $obat->id = $_GET['id'];
```

- `if (isset($_GET['id']) && isset($_GET['confirm']) && $_GET['confirm'] == 'yes')`: Memastikan bahwa terdapat parameter id dalam URL (`$_GET['id']`) dan parameter confirm dengan nilai yes (`$_GET['confirm'] == 'yes'`). Jika kondisi ini terpenuhi, artinya pengguna sudah mengonfirmasi bahwa data obat dengan ID tertentu akan dihapus.

3.2 Menghapus Data Obat dari Database

```
php
if ($obat->delete()) {
    header("Location: index.php");
    exit();
} else {
    echo "Unable to delete obat.";
}
```

- `$obat->id = $_GET['id'];`: Mengatur ID obat yang ingin dihapus berdasarkan nilai id dari URL (`$_GET['id']`).
- `if ($obat->delete())`: Memanggil metode delete() dari kelas obat untuk menghapus data obat berdasarkan ID yang ditentukan. Jika penghapusan berhasil, halaman akan diarahkan kembali ke index.php.
- header(`"Location: index.php"); exit();`: Jika penghapusan berhasil, pengguna akan dialihkan kembali ke halaman utama (`index.php`). `exit()`; menghentikan eksekusi skrip setelah pengalihan.
- echo `"Unable to delete obat."`;: Menampilkan pesan kesalahan jika penghapusan data gagal dilakukan.

3.3 Menampilkan Prompt Konfirmasi

```
php
} elseif (isset($_GET['id'])) {
    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>';
```

- `elseif (isset($_GET['id']))`: Memastikan bahwa parameter id tersedia tetapi confirm belum diatur. Kondisi ini menunjukkan bahwa pengguna belum mengonfirmasi penghapusan.
- **Prompt Konfirmasi Penghapusan**:
    - **JavaScript** confirm(): Memunculkan dialog konfirmasi yang menanyakan kepada pengguna apakah mereka yakin ingin menghapus data obat.
        - `if (confirm("Yakin mau menghapus?"))`: Jika pengguna menekan "OK", maka pengguna akan diarahkan ulang (window.location.href) dengan menambahkan confirm=yes ke URL untuk mengonfirmasi penghapusan.
        - else: Jika pengguna memilih "Cancel", pengguna akan diarahkan kembali ke index.php tanpa melakukan tindakan apa pun.

3.4 Penanganan Jika ID Tidak Valid

```
php
} else {
    echo "Invalid ID.";
}
```

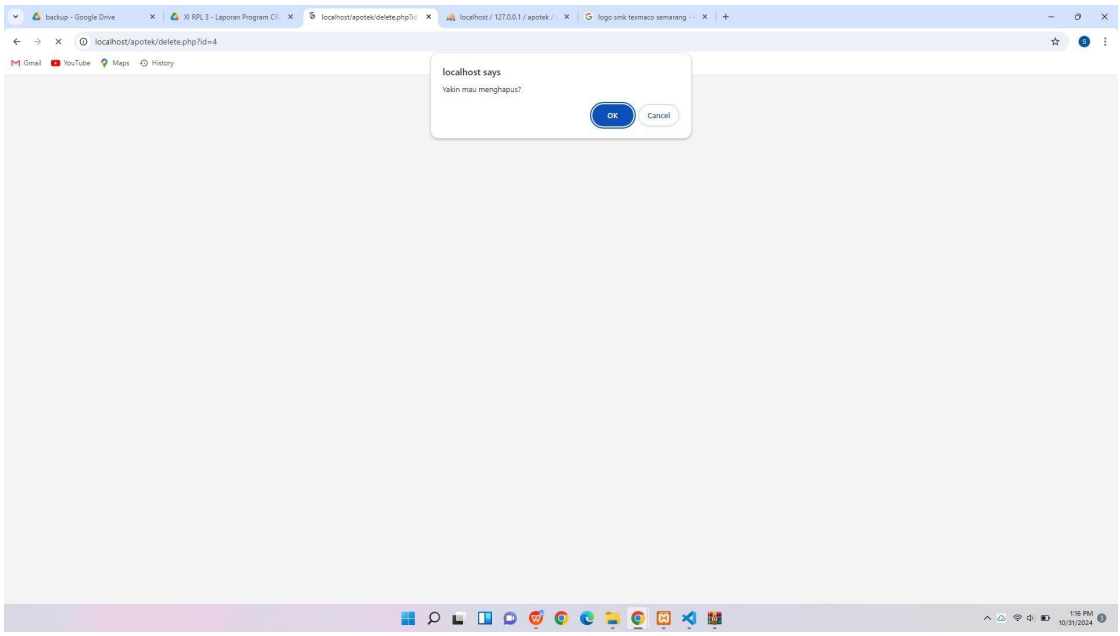
- else: Jika parameter id tidak tersedia dalam URL, pesan `"Invalid ID."` akan ditampilkan. Hal ini menangani skenario di mana pengguna mungkin mengakses skrip ini tanpa ID yang valid atau tanpa melakukan tindakan yang tepat.

Penjelasan Tambahan tentang Metode delete() pada Kelas obat

Metode delete() di kelas obat bertugas menghapus data berdasarkan id obat. Biasanya, metode ini:

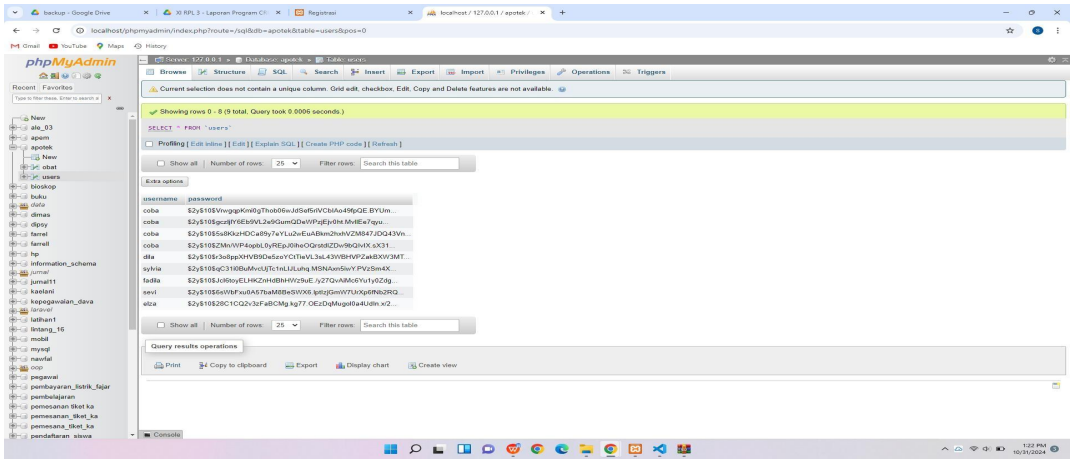
1. Menyusun query SQL DELETE FROM untuk tabel obat.
2. Mengeksekusi query untuk menghapus data sesuai ID yang diberikan.
3. Mengembalikan true jika penghapusan berhasil, atau false jika terjadi kesalahan.

- tampilan menghapus data



II. Login

A. Membuat tabel baru dengan nama users



Penjelasan:

Langkah-lanhgkahnya berikut:

1. Masuk ke phpMyAdmin: Buka browser dan akses phpMyAdmin (biasanya di `http://localhost/phpmyadmin`).

2.Buat Database

- Klik pada tab Databases di bagian atas.

- Di kolom Create database, masukkan nama database, misalnya `users`.

- Pilih jenis collation (biasanya `utf8_general_ci`), lalu klik Create

3. Buat Tabel

- Setelah database dibuat, pilih database `user` dari daftar di sebelah kiri.

- Di bagian Create table, masukkan nama tabel `users`.

- Tentukan jumlah kolom (misalnya 4) dan klik go

4. Definisikan Kolom:

- Untuk setiap kolom, masukkan nama dan jenis data:

- id INT, panjang 11, centang A_I (Auto Increment), dan set sebagai Primary Key.

- username: VARCHAR, panjang , set NOT NULL

- password: VARCHAR, panjang 100, set NOT NULL

- Setelah selesai, klik Save

5. Tabel Siap Digunakan: Tabel `users` sekarang sudah ada dalam database `user`, dan Anda dapat mulai menambahkan data atau mengelola tabel tersebut.

B. Buat file koneksi dengan nama database.php

```php
<?php
class Database{
    private $host="localhost";
    private $db_name="apotek";
    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;
    }
}
?>
```

Penjelasan:

Kode di atas adalah kelas PHP bernama `Database` yang digunakan untuk membuat koneksi ke database MySQL. Berikut adalah penjelasan detail dari setiap bagian kode:

1. Membuat Kelas `Database`

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

- `$host`: Menyimpan alamat host database, yaitu `localhost`. Biasanya `localhost` digunakan saat pengembangan di server lokal.
- `$db_name`: Menyimpan nama database yang ingin diakses, dalam contoh ini adalah `apotek`.
- `$username`: Menyimpan username untuk mengakses database, yaitu `root`.
- `$password`: Menyimpan password untuk username `root`. Dalam contoh ini, password dikosongkan karena default MySQL pada XAMPP atau WAMP biasanya tidak memiliki password.
- `$conn`: Properti publik yang akan menyimpan koneksi database. Properti ini bisa diakses dari luar kelas, dan akan diisi objek koneksi (`PDO`) jika berhasil terhubung ke database.

2. Membuat Fungsi `getConnection`

```php
<?php
public function getConnection() {
    $this->conn = null;
```

- `getConnection`: Fungsi ini berfungsi untuk membuat koneksi ke database. Saat fungsi dipanggil, properti `$conn` akan diinisialisasi menjadi `null`.

3. Mencoba Membuat Koneksi dengan `try-catch`

```php
try {
    $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name, $this->username, $this->password);
    $this->conn->exec("set names utf8");
```

- `try`: Blok `try` ini digunakan untuk mencoba kode yang berpotensi menimbulkan error.
- `new PDO(...)`: Membuat koneksi ke database menggunakan **PDO (PHP Data Objects). Konstruksi ini mengatur:
  - `"mysql:host=" . $this->host . ";dbname=" . $this->db_name` untuk menentukan host dan nama database.
  - `$this->username` dan `$this->password` sebagai informasi login database.
- `set names utf8`: Mengatur karakter encoding menjadi UTF-8 untuk memastikan karakter yang diambil atau disimpan dalam database ditampilkan dengan benar.

4. Menangani Error Koneksi dengan `catch`

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

- `catch`: Jika terjadi error saat mencoba koneksi, blok `catch` ini akan menangkap error tersebut.
- `$exception->getMessage()`: Menampilkan pesan error yang lebih detail yang dihasilkan oleh PDO.

5. Mengembalikan Koneksi

```php
return $this->conn;
```

- Kode ini mengembalikan nilai `$conn`, yaitu objek koneksi PDO yang sudah berhasil dihubungkan ke database atau `null` jika koneksi gagal.

Penggunaan Kelas `Database`

Kelas ini bisa digunakan untuk koneksi database dengan langkah berikut:
1. Inisialisasi objek `Database`: `$database = new Database();`
2. Memanggil fungsi `getConnection` untuk membuat koneksi: `$conn = $database->getConnection();`

Contoh Penggunaan

```php
<?php
$database = new Database();
```

C. Buat file dengan nama dashboard.php

```php
<?php
```

```php
include_once 'database/database.php';
include_once 'objects/obat.php';

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

$obat = new obat($db);

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

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>Data Obat</title>
    <style>
        /* style.css */
        body {
            font-family: Arial, sans-
serif;
```

```
        background-color: #f4f4f4;
        margin: 0;
        padding: 20px;
    }
```

```
    h1 {
        color: #333;
    }
```

```
    a {
        text-decoration: none;
        color: #4B5390;
    }
```

```
    a:hover {
        text-decoration: underline;
    }
```

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

```
        th, td {
            padding: 12px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }
```

```
        th {
            background-color: #2F4156;
            color: white;
        }
```

```
        tr:hover {
            background-color: #f1f1f1;
        }
```

```
        td a {
            margin-right: 10px;
            color: #dc3545; /* Bootstrap
danger color */
        }
```

```
        td a:hover {
            color: #c82333; /* Darker
shade on hover */
```

```
        }
```

```
        .button-container
            { margin-top:
            20px;
```

```
        .btn {
            display: inline-block;
            padding: 10px 15px;
            margin-right: 10px;
            color: white;
            background-color: #007bff; /*
Bootstrap primary color */
            border: none;
            border-radius: 5px;
            text-align: center;
            text-decoration: none;
            cursor: pointer;
        }
```

```
        .btn-logout {
            background-color: #dc3545; /*
Bootstrap danger color */
```

```
        }

        .btn:hover {
            opacity: 0.8;
        }
    </style>
</head>
<body>
    <h1>Data Obat</h1>
    <div class="button-container">
        <a href="create.php"
class="btn">Tambah Obat</a>
        <a href="logout.php" class="btn
btn-logout">Logout</a>
    </div>
    <table>
        <tr>
            <th>ID</th>
            <th>Nama</th>
            <th>Jenis</th>
            <th>Kegunaan</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>{$jenis}</td>";
                    echo "<td>{$kegunaan}</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>
```

Pernjelasan:

Kode PHP di atas menampilkan halaman "Data Obat" yang berisi daftar obat yang tersimpan dalam database. Halaman ini menampilkan tabel dengan data obat dan menyediakan opsi untuk mengubah atau menghapus data. Berikut ini penjelasan kode secara detail:

1. Menyertakan File yang Diperlukan

php
```
include_once 'database/database.php';include_once
'objects/obat.php';
```

- database.php: File ini berisi kelas Database yang mengelola koneksi ke database.
- obat.php: File ini berisi kelas obat, yang menyediakan metode CRUD (Create, Read, Update, Delete) untuk tabel obat.

2. Membuat Koneksi Database dan Objek obat

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

- `$database = new Database();`: Membuat objek Database untuk memulai koneksi ke database.
- `$db = $database->getConnection();`: Memanggil metode getConnection() dari kelas Database untuk mendapatkan koneksi aktif ke database.
- `$obat = new obat($db);`: Membuat objek obat yang menggunakan koneksi $db. Objek ini akan digunakan untuk mengakses data dari tabel obat.

3. Membaca Data Obat

php
```
Copy code
$stmt = $obat->read();
```

- `$stmt = $obat->read();:` Memanggil metode read() dari kelas obat, yang mengembalikan semua data dari tabel obat. Hasilnya disimpan dalam variabel $stmt, yang merupakan objek PDOStatement.

4. Struktur HTML dan CSS

Bagian ini menampilkan struktur HTML untuk halaman "Data Obat" serta gaya CSS untuk tampilan.

HTML Dasar

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

```
<style>
    /* CSS Styles */
</style></head><body>
<h1>Data Obat</h1>
<!-- Content here --></body></html>
```

- `<!DOCTYPE html>:` Menentukan tipe dokumen sebagai HTML5.
- `<title>Data Obat</title>:` Menentukan judul halaman sebagai "Data Obat".
- **Bagian** `<style>:` Berisi CSS untuk mengatur tampilan halaman.

CSS untuk Tampilan Halaman

CSS dalam tag `<style>` mengatur gaya untuk tampilan halaman. Berikut beberapa elemen penting:

- body: Mengatur gaya dasar halaman dengan font Arial dan background abu-abu muda.
- table: Mengatur tabel agar lebar penuh, dengan garis bawah dan latar belakang pada header.
- .btn **dan** .btn-logout: Mengatur tampilan tombol tambah data dan logout.

5. Tombol Tambah Obat dan Logout

```
html
<div class="button-container">
    <a href="create.php" class="btn">Tambah Obat</a>
    <a href="logout.php" class="btn btn-logout">Logout</a></div>
```

- `<a href="create.php" class="btn">Tambah Obat</a>`: Menyediakan tombol untuk menambah data obat baru. Jika diklik, akan membuka halaman create.php.
- `<a href="logout.php" class="btn btn-logout">Logout</a>`: Menyediakan tombol logout yang mengarahkan ke logout.php.

6. Tabel Data Obat

Bagian ini menampilkan data obat dalam tabel dengan header dan baris data obat yang diambil dari database.

Header Tabel

```
html
<table>
    <tr>
        <th>ID</th>
        <th>Nama</th>
        <th>Jenis</th>
        <th>Kegunaan</th>
        <th>Harga</th>
        <th>Aksi</th>
    </tr>
```

- **Kolom Header**: Tabel memiliki kolom ID, Nama, Jenis, Kegunaan, Harga, dan Aksi.

Mengambil dan Menampilkan Data Obat

```
php
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    extract($row);
    echo "<tr>";
    echo "<td>{$id}</td>";
    echo "<td>{$nama}</td>";
    echo "<td>{$jenis}</td>";
    echo "<td>{$kegunaan}</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>";
}
```

- `$row = $stmt->fetch(PDO::FETCH_ASSOC):` Mengambil setiap baris data obat sebagai array asosiatif. `fetch(PDO::FETCH_ASSOC)` memastikan bahwa data diambil sebagai array dengan nama kolom sebagai kunci.
- `extract($row);:` Menyederhanakan akses ke data, seperti `$row['id']` menjadi `$id.`
- `<tr>...</tr>`: Membuat baris tabel baru untuk setiap obat dengan kolom data obat, termasuk ID, Nama, Jenis, Kegunaan, dan Harga.
- **Kolom Aksi**: Menyediakan tautan "Ubah" dan "Hapus" untuk setiap baris obat:
    - `<a href='update.php?id={$id}'>Ubah</a>:` Mengarahkan ke halaman update.php untuk mengubah data obat dengan ID tertentu.
    - `<a href='delete.php?id={$id}'>Hapus</a>:` Mengarahkan ke delete.php untuk menghapus data obat dengan ID tertentu.

Ringkasan

Halaman ini memungkinkan pengguna melihat daftar obat yang tersimpan dalam database serta menyediakan opsi untuk menambah, mengubah, dan menghapus data obat.

D. Buat file user.php

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

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

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

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

```php
     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;
        }
}
?>
```

Penjelasan:

Kode PHP di atas merupakan implementasi kelas User yang menyediakan fungsi untuk login dan registrasi pengguna. Kelas ini berinteraksi dengan tabel users dalam database untuk memverifikasi dan menyimpan data pengguna. Berikut adalah penjelasan detail dari setiap bagian:

1. Deklarasi Kelas User

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

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

- `$conn:` Properti ini menyimpan koneksi database yang diinisialisasi melalui parameter `$db` di konstruktor kelas.
- `$table_name:` Menyimpan nama tabel (users) yang digunakan dalam database.
- `__construct($db)`: Konstruktor kelas, digunakan untuk menginisialisasi koneksi database `($conn)` dengan objek koneksi yang diterima dari luar.

2. Fungsi login

php

```
public function login($username, $password) {
    $query = "SELECT * FROM " . $this->table_name . " WHERE
username = :username LIMIT 1";
    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(":username", $username);
    $stmt->execute();
```

Fungsi ini melakukan verifikasi data login pengguna berdasarkan username dan password yang diinput:

1.

Query Pengambilan Data:

2.

- `SELECT * FROM users WHERE username = :username LIMIT 1`: Menyiapkan query SQL untuk memilih pengguna dengan username tertentu, membatasi hasilnya menjadi satu baris.
- prepare(): Mempersiapkan query SQL untuk dieksekusi, yang memberikan perlindungan terhadap SQL injection.
- bindParam(): Menghubungkan parameter :username dengan variabel $username yang diinput oleh pengguna.

3.

Eksekusi dan Verifikasi:

4.

php

```
if ($stmt->rowCount() > 0) {
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if (password_verify($password, $row['password'])) {
        session_start();
        $_SESSION['username'] = $username;
        return true;
    }
}return false;
```

- `$stmt->rowCount()`: Memeriksa apakah ada hasil dengan username yang cocok. Jika ada, lanjutkan verifikasi.
- `fetch(PDO::FETCH_ASSOC):` Mengambil data pengguna dalam bentuk array asosiatif.
- `password_verify():` Memverifikasi apakah password yang diinput pengguna cocok dengan password yang disimpan dalam database. Fungsi ini aman karena password yang disimpan dalam database telah di-hash.
- `session_start()` **dan** `$_SESSION['username']`: Jika password cocok, sesi pengguna dimulai, dan username disimpan dalam variabel sesi.
- return true **atau** return false: Mengembalikan true jika login berhasil; false jika gagal.

3. Fungsi register

```
php
public function register($username, $password) {
    $query = "INSERT INTO " . $this->table_name . " (username,
password) VALUES (:username, :password)";
    $stmt = $this->conn->prepare($query);
```

Fungsi ini menambahkan pengguna baru ke dalam database dengan meng-hash password yang diinput.

1. **Persiapan Query**:
    - `INSERT INTO users (username, password) VALUES (:username, :password):` Menyiapkan query untuk menyimpan data username dan password pengguna baru.
    - `prepare():` Mempersiapkan query SQL untuk keamanan.
2. **Enkripsi Password**:

```
php
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
```

- `password_hash()`: Fungsi PHP untuk mengenkripsi password, menggunakan algoritma `PASSWORD_DEFAULT`, yang aman terhadap serangan dan menyimpan password dalam bentuk hash.

1. **Menghubungkan Parameter dan Eksekusi Query**:

```
php
$stmt->bindParam(":username", $username);$stmt-
>bindParam(":password", $hashed_password);if ($stmt->execute()) {
    return true;
}return false;
```

- `bindParam():` Menghubungkan :username dan :password dalam query dengan variabel $username dan $hashed_password.
- `$stmt->execute():` Menyimpan data ke database. Jika berhasil, fungsi mengembalikan true, dan false jika gagal.

Kesimpulan

Kelas User ini menyediakan metode yang aman untuk:

1. **Login**: Mengecek data pengguna dengan username dan password, serta memulai sesi jika login berhasil.
2. **Registrasi**: Menyimpan data pengguna baru dalam database dengan password yang telah di-hash.

Dengan penggunaan `password_hash()` dan `password_verify()`, sistem ini menjaga keamanan data pengguna dan melindungi password dari pencurian.

E. Buat file register.php

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

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

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

```php
    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/bootst
rap@5.3.3/dist/css/bootstrap.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>
            <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>
```

Penjelasan:

kode di atas adalah implementasi halaman registrasi (register.php) untuk pengguna baru, yang memungkinkan pengguna untuk membuat akun dengan username dan password. Berikut adalah penjelasan detail setiap bagian dari kode ini:

1. Inisialisasi Koneksi dan Objek

```
php
require_once 'database/database.php';require_once
'objects/user.php';
$database = new Database();$db = $database->getConnection();$user
= new User($db);
```

- `require_once:` Menyertakan file database.php dan user.php satu kali.
  - database.php: Menyediakan kelas Database untuk menghubungkan aplikasi dengan database.
  - user.php: Menyediakan kelas User yang berisi fungsi register untuk menambahkan pengguna baru.
- **Inisialisasi Objek**:
  - Membuat objek $database dari kelas Database dan memanggil getConnection() untuk mendapatkan koneksi database ($db).
  - Membuat objek $user dari kelas User, mengoper $db sebagai parameter, sehingga objek User dapat menggunakan koneksi database.

2. Memproses Formulir Registrasi

php

```
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.";
    }
}
```

- `$_SERVER["REQUEST_METHOD"] == "POST":` Memeriksa apakah formulir dikirim dengan metode POST.
- **Mengambil Data dari Formulir**:
  - $username **dan** $password: Menyimpan data username dan password yang diinputkan oleh pengguna melalui $_POST.
- **Mencoba Registrasi**:
  - $user->register($username, $password): Memanggil fungsi register dari objek User.
    - Jika registrasi berhasil, maka variabel $success_message berisi pesan sukses.
    - Jika gagal, variabel $error_message diisi dengan pesan kesalahan.

3. Bagian HTML

```
html
Copy code
<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/boots
trap.min.css" rel="stylesheet"></head><body><div class="container
mt-5">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <h3 class="text-center">Registrasi</h3>
```

- `<DOCTYPE html>`: Menyatakan tipe dokumen HTML5.
- **Bootstrap**: Menggunakan pustaka CSS dari Bootstrap untuk mempercantik tampilan.
- **Container**:
    - .container: Membuat area konten utama dengan margin dan padding dari Bootstrap.
    - .row dan .col-md-4: Mengatur layout konten dengan menempatkan form registrasi di tengah layar menggunakan justify-content-center.

4. Menampilkan Pesan Sukses dan Error

```
php
<?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; ?>
```

- `if(isset($error_message)) dan if(isset($success_message)):` Memeriksa apakah terdapat pesan error atau sukses setelah registrasi, dan jika ada, menampilkannya dalam bentuk alert.
- **Alert Bootstrap**:
    - alert alert-danger: Pesan error dengan tampilan berwarna merah.
    - alert alert-success: Pesan sukses dengan tampilan berwarna hijau.

5. Formulir Registrasi

```
html
<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">Daftar</button></form><p class="text-center mt-3">Sudah
punya akun? <a href="index.php">Login di sini</a></p>
```

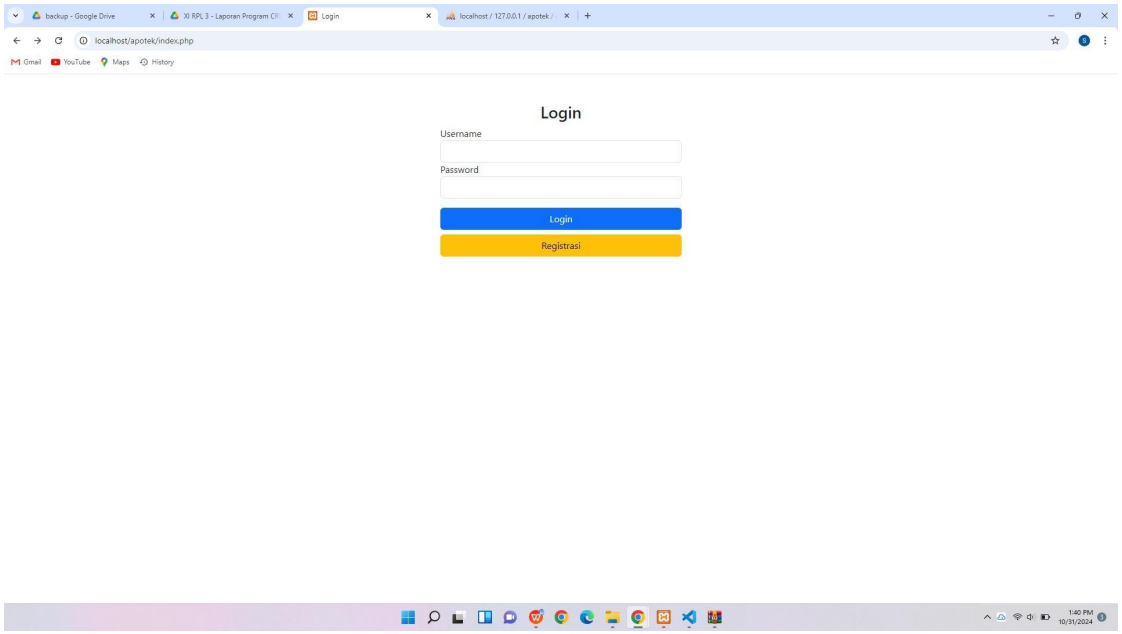
- `form action="" method="POST"`: Membuat form dengan metode POST untuk mengirim data.
- `<input type="text" name="username">:` Field input untuk username.
- `<input type="password" name="password">`: Field input untuk password.
- `<button type="submit">Daftar</button>`: Tombol untuk mengirim form.
- **Link ke halaman login**: Menampilkan tautan bagi pengguna yang sudah memiliki akun untuk login di halaman lain.

Kesimpulan

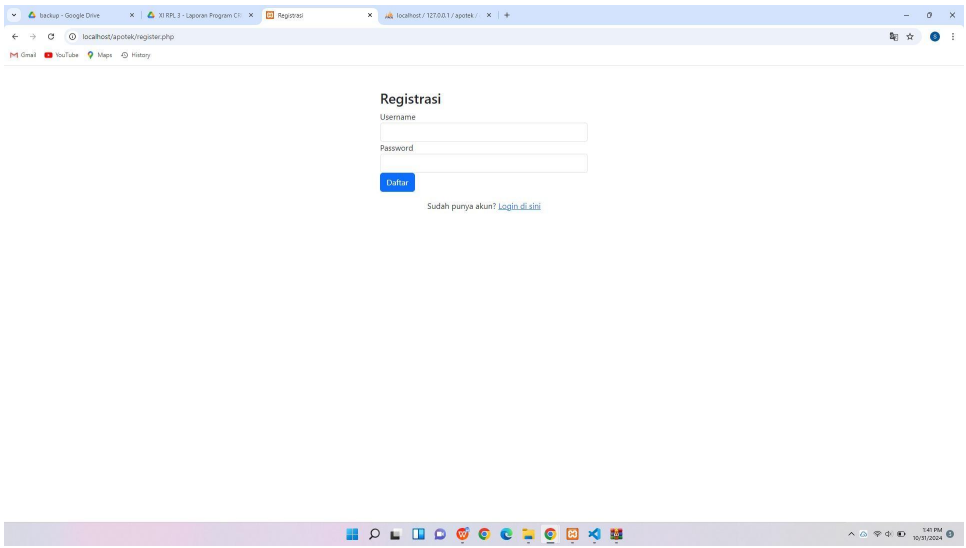
Kode register.php ini menyediakan halaman untuk pengguna baru mendaftarkan akun mereka. Jika username berhasil terdaftar, pengguna mendapatkan pesan sukses. Jika terjadi masalah (misalnya, username sudah ada), pesan error akan ditampilkan.

III. Hasil

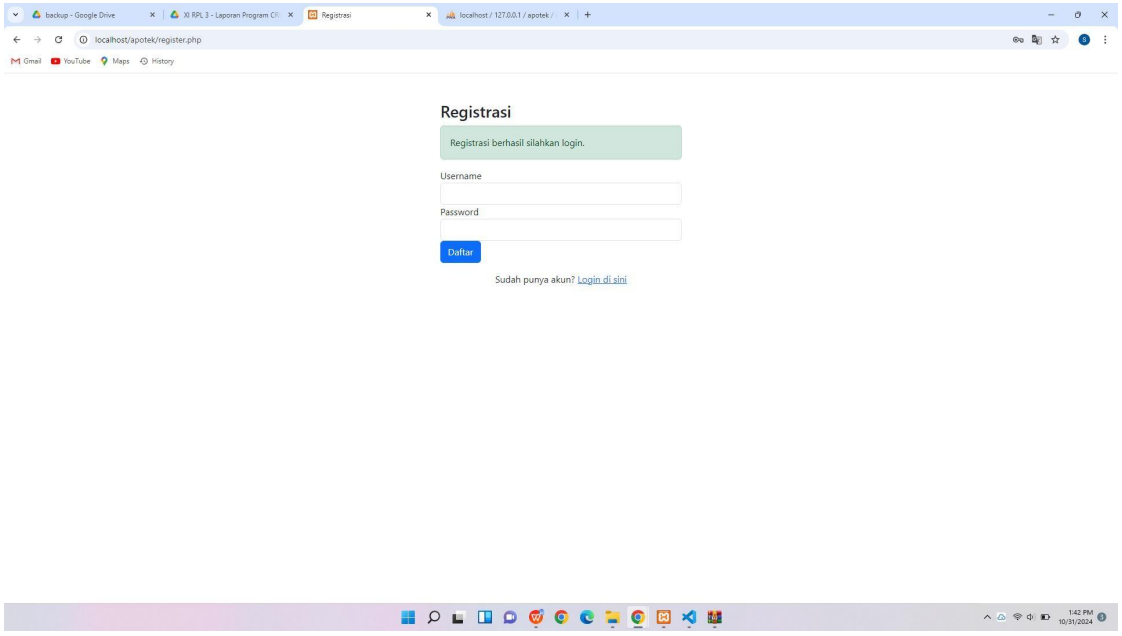
A. Dashboard



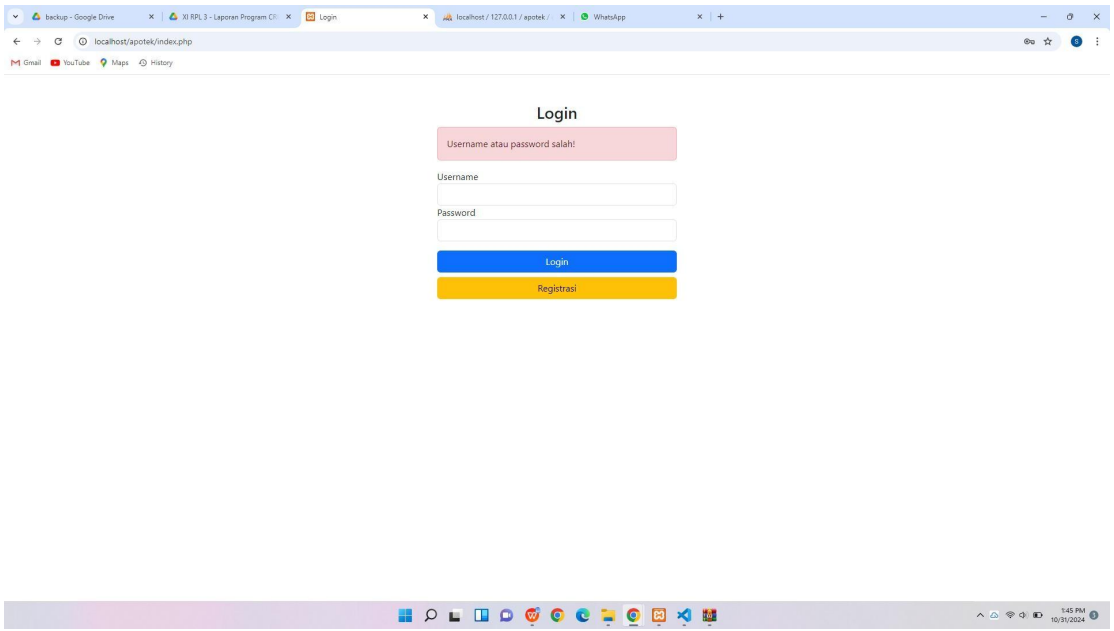
B. Register



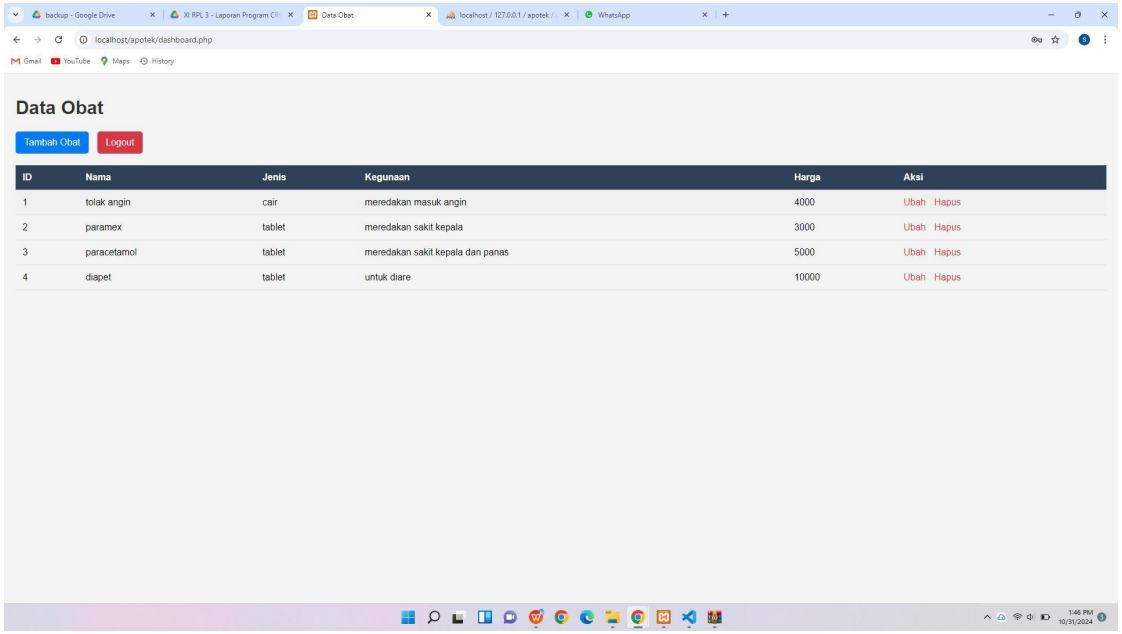
c. Registrasi berhasil



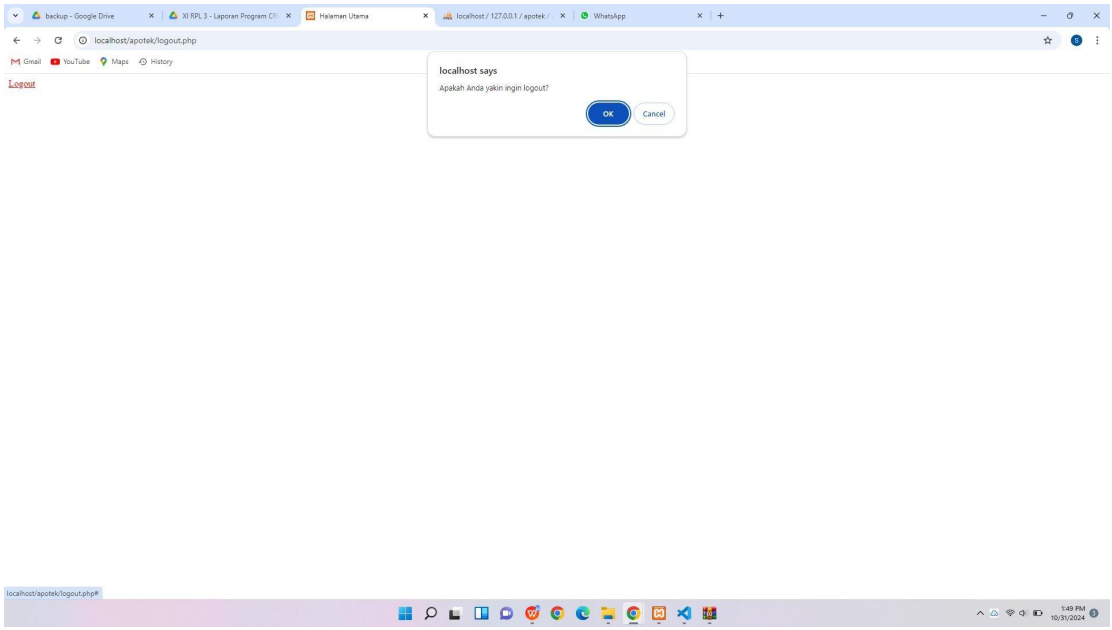
C. logini gagal

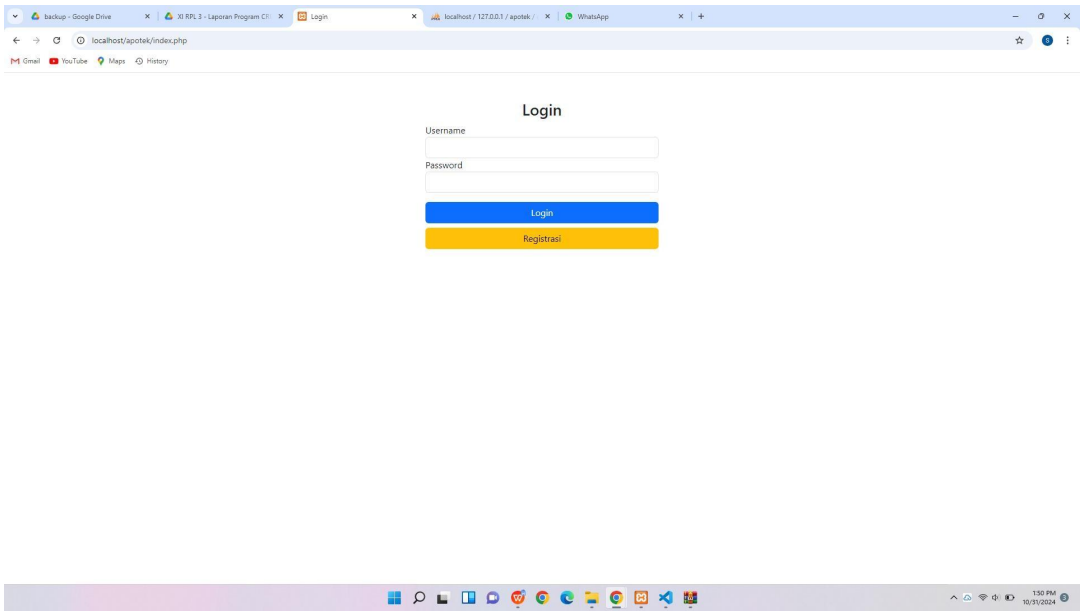


D. login berhasil



E. Log out


backup - Google Drive
XI RPL 3 - Laporan Program CR
Halaman Utama
localhost / 127.0.0.1 / apotek /
WhatsApp
localhost/apotek/logout.php
Gmail
YouTube
Maps
History
Logout
localhost says
Apakah Anda yakin ingin logout?
OK
Cancel
localhost/apotek/logout.php#
1:49 PM
10/31/2024


backup - Google Drive
XI RPL 3 - Laporan Program CR
Login
localhost / 127.0.0.1 / apotek /
WhatsApp
localhost/apotek/index.php
Gmail
YouTube
Maps
History
Login
Username
Password
Login
Registrasi
1:50 PM
10/31/2024