

**Penerapan Sistem Create, Read, Update and Delete (CRUD) pada Website  
Donasi KampusPeduliUPNVJ**

**Laporan Implementasi dan Progress Final Project KampusPeduliUPNVJ**



**Dosen: I Wayan Rangga Pinastawa, S.Kom., M.Kom.**

**Disusun Oleh:**

<b>Fendi Permadi</b>	<b>2310512093</b>
<b>Vega Setiawan</b>	<b>2310512108</b>
<b>Anisa Bella Kayla</b>	<b>2310512110</b>
<b>Ashja Radithya Lesmana</b>	<b>2310512113</b>
<b>Jihan Aulia Rahmawati</b>	<b>2310512117</b>

**UNIVERSITAS PEMBANGUNAN NASIONAL VETERAN JAKARTA  
PROGRAM STUDI SISTEM INFORMASI PROGRAM SARJANA  
TAHUN 2024/2025**

## Daftar Isi

<b>Daftar Isi.....</b>	<b>ii</b>
<b>Daftar Gambar.....</b>	<b>iii</b>
<b>Implementasi CRUD.....</b>	<b>1</b>
1. Arsitektur Sistem.....	1
2. Implementasi Front-End.....	2
3. Endpoint API dan Interaksi Database.....	10
<b>Progress Final Project.....</b>	<b>17</b>
1. Back-End.....	17
2. Front-End.....	17
3. Skema Database.....	25
4. Langkah Selanjutnya.....	27

## Daftar Gambar

Gambar 2.2.1 Halaman Utama Kampus Peduli UPNVJ.....	18
Gambar 2.2.2 Halaman Kontak Kami Kampus Peduli UPNVJ.....	19
Gambar 2.2.3 Halaman About Us Kampus Peduli UPNVJ.....	20
Gambar 2.2.4 Halaman List Donasi Kampus Peduli UPNVJ.....	21
Gambar 2.2.5 Halaman Login Kampus Peduli UPNVJ.....	22
Gambar 2.2.6 Halaman Donasi Kampus Peduli UPNVJ.....	23
Gambar 2.2.7 Halaman Dashboard Kampus Peduli UPNVJ.....	24
Gambar 2.2.8 Halaman Register Kampus Peduli UPNVJ.....	25
Gambar 2.2.9 Kolom Tabel Users.....	26
Gambar 2.2.10 Query Pembuatan Tabel Users.....	26
Gambar 2.2.11 Kolom Tabel Donations.....	26
Gambar 2.2.12 Query Pembuatan Tabel Donations.....	26
Gambar 2.2.13 Kolom Tabel Reports.....	27
Gambar 2.2.14 Query Pembuatan Tabel Reports.....	27

# Implementasi CRUD

## 1. Arsitektur Sistem

### a. Pendahuluan REST API

REST API (*Representational State Transfer*) adalah arsitektur yang memungkinkan interaksi antara klien (frontend) dan server (backend) melalui HTTP dengan pertukaran data dalam format JSON atau XML.

Fungsi REST API dalam Sistem: REST API memungkinkan frontend untuk melakukan operasi CRUD (Create, Read, Update, Delete) dengan backend melalui permintaan HTTP ke URL endpoint tertentu.

### b. Koneksi Backend PHP dengan MySQL

#### 1) Komponen Backend

Backend dibangun menggunakan plain PHP tanpa *framework* tambahan. Setiap *endpoint* REST API direpresentasikan dalam file PHP dalam folder yang spesifik untuk setiap operasi, seperti `login/index.php`, `register/index.php`, `my_user/index.php`, dan `donation/index.php`.

#### 2) Koneksi Database MySQL

PHP berkomunikasi dengan MySQL untuk melakukan operasi penyimpanan, pengambilan, pembaruan, dan penghapusan data. Koneksi ke MySQL dilakukan dengan menggunakan MySQLi. File konfigurasi khusus bernama `connection.php` digunakan untuk menyimpan detail koneksi ke database dan memudahkan pengaturan koneksi dari file lain.

#### 3) Proses Koneksi

Pada setiap file PHP yang berfungsi sebagai endpoint, PHP menghubungkan ke MySQL dan menjalankan query SQL sesuai dengan permintaan yang diterima. Lalu hasil query dikonversi menjadi format JSON agar dapat diakses oleh frontend.

### c. Koneksi Frontend dengan AJAX

#### 1) Penggunaan AJAX untuk Mengakses REST API

Di frontend, *AJAX (Asynchronous JavaScript and XML)* digunakan untuk mengirim permintaan HTTP ke file PHP endpoint tanpa harus memuat ulang halaman. Ini membuat aplikasi lebih responsif.

## 2) Contoh Penggunaan AJAX

Ketika pengguna menekan tombol "Register", fungsi AJAX mengirim permintaan POST ke file PHP (register/index.php) yang sesuai. Setelah itu server dapat mengembalikan status pendaftaran dalam bentuk JSON, tanpa harus ada perpindahan halaman. Setelah itu pengguna bisa langsung dibawa ke halaman Login.

### d. Alur Kerja Sistem REST API dengan Plain PHP

#### 1) Frontend

AJAX dengan Fetch API digunakan untuk mengirim dan menerima data melalui REST API selanjutnya data yang diterima dari backend ditampilkan secara dinamis pada UI.

#### 2) Backend

Setiap file PHP berfungsi sebagai endpoint REST API untuk operasi CRUD lalu PHP mengirim query SQL ke MySQL dan hasilnya dikembalikan sebagai JSON.

#### 3) Database MySQL

Database MySQL berfungsi untuk menyimpan data dan merespons query dari backend, bertindak sebagai penyimpanan data utama yang diakses melalui REST API.

## 2. Implementasi Front-End

### a. Pembuatan Akun

Implementasi front-end dalam pembuatan akun secara keseluruhan berperan sebagai antarmuka bagi pengguna untuk mendaftar akun baru. Fungsi dari implementasi front-end pembuatan akun ini meliputi:

#### 1) Mengambil Nilai Input

```
43         function register() {  
44             const name = document.getElementById("name").value  
45             const email = document.getElementById("email").value  
46             const password = document.getElementById("password").value  
47             const phone = document.getElementById("phone").value  
48             const passwordRepeat = document.getElementById("password-repeat").value  
49             const registerError = document.getElementById("register-error")
```

Setiap baris tersebut mengambil nilai dari elemen input (HTML <input>) yang memiliki id tertentu (name, email, password, dll.). Nilai yang diperoleh kemudian disimpan ke variabel masing-masing.

## 2) Validasi *Password*

```
50         registerError.innerText = ""
51         console.log(password)
52         console.log(passwordRepeat)
53         if(password !== passwordRepeat){
54             registerError.innerText = "Password Tidak Sesuai"
55             return
56         }
```

Selanjutnya, kode tersebut mengosongkan pesan error terlebih dahulu (registerError.innerText = ""), kemudian memeriksa apakah password sesuai dengan passwordRepeat, dan jika tidak sesuai, menampilkan pesan "Password Tidak Sesuai" pada elemen dengan id register-error serta menghentikan fungsi dengan return.

## 3) Mempersiapkan Data untuk Dikirim

```
57         const data = new URLSearchParams();
58         data.append("nama", name);
59         data.append("email", email);
60         data.append("password", password);
61         data.append("phone", phone);
```

Kemudian, kode tersebut membuat objek URLSearchParams untuk menyimpan data yang akan dikirim, dan menggunakan append untuk menambahkan data name, email, password, dan phone ke objek data.

## 4) Mengirim Data ke Server Menggunakan *Fetch*

```
64         fetch("/api/register/", {
65             method: "POST",
66             body: data
67         }).then(x=>x.json())
68         .then(x=>{
69             if(x.status === "success"){
70                 successPopup.classList.remove('hidden');
71                 console.log(successPopup.classList);
72             }else{
73                 registerError.innerText = "Gagal melakukan registrasi"
74             }
75         })
76
77     }
```

Kode tersebut mengirim permintaan POST ke /api/register/ dengan data sebagai body, kemudian mengubah respons menjadi objek JavaScript

dengan `.then(x => x.json())`, dan mengevaluasi apakah status respons adalah "success"; jika berhasil, menampilkan elemen `successPopup` dengan menghapus kelas `hidden`, dan jika gagal, menampilkan pesan "Gagal melakukan registrasi" di elemen `registerError`.

#### b. Masuk Akun

Implementasi front-end dalam masuk akun menangani proses login pengguna dengan mengambil data dari form, mengirimkan data ke server untuk otentikasi, dan memberikan umpan balik berupa pesan error atau mengarahkan pengguna ke halaman beranda jika login berhasil. Fungsi dari implementasi front-end masuk akun ini meliputi:

##### 1) Mengambil Data dari Input Pengguna

```
28         function login() {  
29             const email = document.getElementById("email").value  
30             const password = document.getElementById("password").value  
31             const loginError = document.getElementById("login-error")  
32             loginError.innerText = ""
```

Kode tersebut mengambil nilai yang dimasukkan oleh pengguna ke dalam kolom email dan password pada form login, menyimpannya ke dalam variabel email dan password, mendapatkan elemen HTML dengan `id="login-error"` untuk menampilkan pesan error jika login gagal, dan mengosongkan pesan error sebelumnya, jika ada.

##### 2) Mempersiapkan Data untuk Dikirim

```
34             const data = new URLSearchParams();  
35             data.append("email", email);  
36             data.append("password", password);
```

Selanjutnya, kode tersebut membuat objek `URLSearchParams` untuk menampung data yang akan dikirim ke server dan menggunakan `.append()` untuk menambahkan email dan password ke objek data yang akan dikirim dalam permintaan POST.

##### 3) Mengirim Data ke Server Menggunakan *Fetch*

```

38         fetch("/api/login/", {
39             method: "POST",
40             body: data
41         }).then(x => x.json())
42         .then(x => {
43             if (x.status === "success") {
44                 window.location.href = "/"
45             } else {
46                 loginError.innerText = "Gagal melakukan login, email / password salah."
47             }
48         })
49     |
50 }

```

Kemudian, kode tersebut mengirim permintaan POST ke endpoint `/api/login/` dengan data email dan password sebagai body, mengubah respons server menjadi format JSON, dan memeriksa status respons; jika statusnya "success", halaman akan diarahkan ulang ke beranda, dan jika tidak, pesan error "Gagal melakukan login, email / password salah." akan ditampilkan pada elemen login-error.

#### c. Update Akun

Implementasi front-end dalam Update akun menangani pembaruan data akun pengguna, dengan mengambil data dari form, mengirimnya ke server untuk di-update, dan memberikan umpan balik berupa pesan error atau memuat ulang halaman jika pembaruan berhasil. Fungsi dari implementasi front-end update akun ini meliputi:

##### 1) Mengambil Data dari Input Pengguna

```

62     function update() {
63         const name = document.getElementById("name").value
64         const email = document.getElementById("email").value
65         const phone = document.getElementById("phone").value
66         const updateError = document.getElementById("update-error")

```

Kode tersebut mengambil nilai dari elemen input HTML dengan `id="name"`, `id="email"`, dan `id="phone"` lalu menyimpannya dalam variabel `name`, `email`, dan `phone`, serta mendapatkan elemen HTML `updateError` dengan `id="update-error"` untuk menampilkan pesan error jika pembaruan data gagal.

##### 2) Mempersiapkan Data untuk Dikirim

```

67         const data = new URLSearchParams();
68         data.append("nama", name);
69         data.append("email", email);
70         data.append("phone", phone);

```

Selanjutnya, kode tersebut membuat objek `URLSearchParams` untuk menyimpan data yang akan dikirim ke server dan menambahkan `name`, `email`, serta `phone` ke objek data menggunakan `.append()`.

##### 3) Mengirim Data ke Server Menggunakan *Fetch*



```

73         fetch("/api/my_user/", {
74             method: "POST",
75             body: data,
76             credentials: "include"
77         }).then(x => x.json())
78         .then(x => {
79             if (x.status === "success") {
80                 location.reload()
81             } else {
82                 updateError.innerText = "Gagal melakukan update"
83             }
84         })
85     }
86 }

```

Kemudian, kode tersebut mengirimkan permintaan POST ke endpoint `/api/my_user/` dengan data sebagai body dan menyertakan `credentials: "include"` agar permintaan membawa informasi sesi seperti cookie untuk otentikasi. Setelah respons diterima dan diubah menjadi objek JSON, jika status respon adalah "success", halaman akan dimuat ulang dengan `location.reload()` untuk mencerminkan data yang baru diperbarui, sedangkan jika pembaruan gagal, pesan "Gagal melakukan update" akan ditampilkan pada elemen `updateError`.

#### d. Hapus Akun

Implementasi front-end menangani penghapusan akun dengan cara menampilkan pop-up konfirmasi, mengirim permintaan ke server untuk menghapus akun, dan memberikan respons kepada pengguna. Tujuan dari implementasi ini adalah untuk memastikan bahwa pengguna dapat melakukan penghapusan akun dengan aman dan sadar akan konsekuensinya. Fungsi dari implementasi ini antara lain:

##### 1) Mengirimkan permintaan ke server untuk menghapus akun pengguna.

```

87     function deleteAccount(){
88         fetch("/api/my_user/", {
89             method: "DELETE",
90             credentials:"include"
91         })

```

- Mengirimkan permintaan HTTP DELETE ke endpoint /api/my\_user/. Endpoint ini bertugas menangani logika penghapusan akun pada sisi server.
- credentials: "include" memastikan bahwa permintaan dikirim bersama kredensial yang relevan (seperti cookie atau token sesi) jika diperlukan untuk autentikasi.

##### 2) Menangani Response

```

92         .then(x=>x.json())
93         .then(x=>{
94             if(x.status === "success"){
95                 window.location.href = "/"
96             }
97         })
98     }

```

- .then(x => x.json()): Mengubah respons dari server menjadi objek JSON.
- Jika x.status === "success", artinya penghapusan akun berhasil. Dalam hal ini, pengguna akan diarahkan ke halaman beranda (window.location.href = "/"), yang menunjukkan bahwa akun telah dihapus dan pengguna diarahkan keluar dari halaman.

### 3) Menampilkan elemen pop-up konfirmasi penghapusan akun

```

99         function deletePopUp(){
100             deletePrompt.classList.remove("hidden")
101         }

```

deletePrompt.classList.remove("hidden"): Menghapus kelas hidden dari elemen deletePrompt, sehingga pop-up yang berisi konfirmasi penghapusan ditampilkan kepada pengguna.

### 4) Event Listener untuk Tombol Tutup Pop-Up

```

102         closePopupButton.addEventListener('click', function () {
103             deleteAccount()
104         });

```

Ketika tombol tutup pada pop-up diklik, fungsi deleteAccount() dipanggil untuk memulai proses penghapusan akun. Implementasi ini menunjukkan bahwa pengguna harus mengonfirmasi tindakan mereka dengan menekan tombol, yang membantu menghindari penghapusan akun yang tidak disengaja.

### 5) Menutup Pop-Up dengan Klik di Luar Area

```

106         window.addEventListener('click', function (e) {
107             if (e.target === deletePrompt) {
108                 deletePrompt.classList.add('hidden');
109             }
110         });

```

- window.addEventListener('click', function (e) { ... }) menangani klik di mana saja di jendela.
- Jika pengguna mengklik di luar area elemen deletePrompt, kelas hidden akan ditambahkan kembali ke elemen deletePrompt, menyembunyikan pop-up.

e. Penambahan Donasi

Implementasi front-end penambahan donasi ini mengelola pengumpulan data donasi, pengiriman data ke server, serta penanganan tampilan pop-up untuk konfirmasi pembayaran. Berikut penjelasannya:

1) Mengambil Data dari Form Input

```
241         function sendMessage() {  
242             const name = document.getElementById("name").value  
243             const email = document.getElementById("email").value  
244             const phone = document.getElementById("telp").value  
245             const msg = document.getElementById("message").value
```

name, email, phone, dan msg diperoleh dari elemen HTML dengan document.getElementById().

2) Membuat Data dengan URLSearchParams

```
const data = new URLSearchParams();  
data.append("name", name);  
data.append("email", email);  
data.append("phone", phone);  
data.append("msg", msg);  
data.append("nominal", activeNominal);  
data.append("payment", activePayment);
```

Data pengguna diisi ke dalam objek URLSearchParams agar dapat dikirim ke server sebagai data POST.

3) Mengirim Data ke Endpoint /api/donation/

```
253         fetch("/api/donation/", {  
254             method: "POST",  
255             credentials: "include",  
256             body: data  
257         })
```

- Fungsi fetch() dengan metode POST mengirimkan data donasi ke server melalui endpoint /api/donation/.
- credentials: "include" memungkinkan pengiriman informasi autentikasi jika diperlukan.

f. Menampilkan Donasi

Bagian ini bekerja dengan mengambil data donasi dari server melalui API, lalu menampilkannya secara dinamis dalam bentuk tabel pada halaman web, penjelasan lengkap mengenai bagian ini antara lain:

1) Fetch Data dari API

```

111         fetch("/api/donation/")
112         .then(x => x.json())
113         .then(donations => {

```

Fungsi `fetch("/api/donation/")` akan mengirim permintaan HTTP GET ke endpoint `/api/donation/`. Setelah itu, responnya diubah menjadi objek JSON menggunakan `.then(x => x.json())`. Hasil dari konversi JSON ini berupa array berisi data donasi, yang diteruskan ke fungsi berikutnya dengan parameter `donations`.

## 2) Menyiapkan Tabel untuk Menampilkan Data

```

114         const tbody = document.getElementById("table-body")

```

Variabel `tbody` mengacu pada elemen HTML dengan `id="table-body"`. Elemen ini akan menjadi wadah untuk baris-baris tabel yang berisi data donasi.

## 3) Mengisi Konten Tabel dengan Data Donasi

```

115         tbody.innerHTML = donations.map(donation => {

```

Fungsi `.map()` digunakan untuk memproses setiap objek `donation` dalam array `donations`. Hasil pemrosesan ini kemudian akan diubah menjadi baris-baris HTML yang akan ditambahkan ke elemen `tbody`.

## 4) Memformat Tanggal

```

116         const date = new Date(donation.created_at)
117         const year = date.getFullYear();
118         const month = date.getMonth() + 1; // Months are zero-indexed (0-11), so add 1 to display correctly
119         const day = date.getDate();
120         const formattedDate = `${year}-${month.toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`;

```

Untuk menampilkan tanggal donasi, `created_at` yang berformat tanggal diubah menjadi objek `Date`. Variabel `year`, `month`, dan `day` digunakan untuk mendapatkan bagian tahun, bulan, dan hari dari tanggal tersebut. Fungsi `padStart(2, '0')` memastikan bahwa bulan dan hari selalu memiliki dua digit (misalnya, 2023-05-07).

## 5) Menyusun Baris Tabel untuk Setiap Donasi

```

121         return `
122             <tr class="border-t hover:bg-gray-100">
123                 <td class="px-4 py-2 text-md text-gray-900">${donation.id}</td>
124                 <td class="px-4 py-2 text-md text-gray-900">${donation.donor_name}</td>
125                 <td class="px-4 py-2 text-md text-gray-900">${formatRupiah(donation.total_donations)}</td>
126                 <td class="px-4 py-2 text-md text-gray-900">${formattedDate}</td>
127                 <td class="px-4 py-2 text-md text-gray-900">${donation.payment}</td>
128                 <td class="px-4 py-2 text-md text-gray-900">${donation.donation_message}</td>
129             </tr>

```

Kode ini menghasilkan HTML untuk satu baris tabel (`<tr>`) yang berisi data donasi. Setiap kolom (`<td>`) diisi dengan informasi dari objek `donation`, yaitu `id`, `donor_name`, `total_donations` (diformat dengan fungsi `formatRupiah()`), `formattedDate`, `payment`, dan `donation_message`.

## 6) Fungsi *formatRupiah*

Fungsi `formatRupiah(donation.total_donations)` diharapkan untuk memformat jumlah donasi menjadi format mata uang Rupiah

### 3. Endpoint API dan Interaksi Database

#### a. Login (/api/login/index.php)

Endpoint API login ini secara keseluruhan berfungsi sebagai *API autentikasi login* untuk memverifikasi apakah pengguna memiliki kredensial yang valid agar dapat mengakses sistem. Fungsi dari endpoint ini antara lain:

##### 1) Melakukan Koneksi Database dengan *connection.php*

```
1      <?php
2
3      include "../..connection.php";
```

Baris kode tersebut berfungsi untuk memasukan file “connection.php” untuk menghubungkan website dengan database yang sudah dibuat sebelumnya. Setelah web berhasil terhubung, koneksi ke database (`$conn`) sudah bisa digunakan di seluruh bagian script ini.

##### 2) Memeriksa Input Login

API ini memulai dengan memeriksa apakah ada data `email` dan password yang dikirimkan melalui metode POST. Jika ada, maka langkah autentikasi akan dilanjutkan; jika tidak, API akan memberikan respon “unauthorized”.

##### 3) Memulai Sesi

Setelah input yang diberikan diverifikasi, `session_start()` dijalankan untuk memulai sesi PHP. Bagian ini akan menyimpan informasi sesi bagi pengguna jika autentikasi berhasil.

##### 4) Query untuk Memeriksa Email di Database

Dengan menggunakan variabel `$conn` (yang berasal dari *connection.php*), API melakukan query SQL untuk mencari data pengguna berdasarkan email yang diberikan. Untuk mencegah serangan SQL injection, query ini menggunakan *mysqli\_prepare* dengan *placeholder ?*, dan kemudian variabel `$email` diikat ke query dengan *mysqli\_stmt\_bind\_param*.

##### 5) Eksekusi Query dan Validasi Password

```
15      $stmt = mysqli_prepare($conn, $sql);
16      mysqli_stmt_bind_param($stmt, "s", $email);
17      mysqli_stmt_execute($stmt);
18      $result = mysqli_stmt_get_result($stmt);
```

Setelah query dieksekusi dengan `mysqli_stmt_execute($stmt)`, hasilnya diambil dengan `mysqli_stmt_get_result($stmt)`. Jika hasil yang dikembalikan ada, maka berarti email pengguna ditemukan di database. Selanjutnya, password yang diberikan akan di-hash menggunakan

algoritma sha256 dan dibandingkan dengan password yang disimpan di database.

#### 6) Menyimpan Informasi Pengguna di Sesi

```
if (mysqli_num_rows($result) > 0) {  
    $row = mysqli_fetch_assoc($result);  
    if ($row["password"] === hash('sha256', $password)) {  
        $_SESSION['user_id'] = $row["id"];  
        $_SESSION['user_name'] = $row["name"];  
        $_SESSION['user_email'] = $row["email"];  
        $_SESSION['user_phone'] = $row["phone"];  
        echo json_encode(["status" => "success"]);  
    }
```

Jika password yang diberikan cocok, selanjutnya data pengguna seperti user\_id, user\_name, user\_email, dan user\_phone disimpan dalam variabel sesi (\$\_SESSION). Kemudian, respons JSON dikirimkan dengan status success, yang menunjukkan bahwa autentikasi berhasil.

#### 7) Respon Kesalahan

Jika email tidak ditemukan atau password salah, maka API akan mengembalikan respons JSON dengan status failed dan pesan kesalahan Wrong Credentials.

#### 8) Penutupan Koneksi Database

```
35     mysqli_stmt_close($stmt);  
36     mysqli_close($conn);  
37 } else {  
38     echo json_encode(["status" => "unauthorized"]);  
39 }
```

Pada bagian akhir, statement SQL ditutup dengan *mysqli\_stmt\_close(\$stmt)*, dan koneksi database ditutup dengan *mysqli\_close(\$conn)* untuk membersihkan resources.

2

#### b. Register (/api/register/index.php)

Endpoint API register adalah endpoint untuk mendaftarkan pengguna baru (register) dengan cara menyimpan data pengguna ke dalam database. Cara kerja API ini antara lain:

##### 1) Melakukan koneksi dengan database

```
1     <?php  
2     include "../..connection.php";
```

Pada baris include "../..connection.php";, kode ini memasukkan file *connection.php*, yang berisi konfigurasi koneksi ke database. File ini memungkinkan API untuk berkomunikasi dengan database yang telah ditentukan

##### 2) Melakukan cek input data

Kode tersebut memeriksa apakah variabel nama, email, password, dan phone telah dikirim melalui metode POST. Jika ada variabel yang tidak

tersedia, maka API akan mengembalikan respons dengan status "unauthorized".

### 3) Mengamankan Data Password

```
10     $name = $_POST['nama'];|
11     $email = $_POST['email'];
12     $password = hash('sha256', $_POST['password']);
13     $phone = $_POST['phone'];
```

### 4) Mempersiapkan Query SQL untuk Insert Data

Pada bagian ‘\$sql = "INSERT INTO users(name, email, password, phone) VALUES(?, ?, ?, ?)";’ kode menyiapkan perintah SQL untuk menambahkan data pengguna ke tabel users

Pada query diatas menggunakan tanda ‘?’ sebagai placeholder untuk data yang akan dimasukkan, yang bertujuan mencegah serangan SQL injection.

### 5) Mengikat dan Menjalankan Query

```
16     $stmt = mysqli_prepare($conn, $sql);
17     mysqli_stmt_bind_param($stmt, "ssss", $name, $email, $password, $phone);
18
19     if (mysqli_stmt_execute($stmt)) {
20         echo json_encode(["status" => "success"]);
21     } else {
22         echo json_encode(["status" => "failed", "error" => mysqli_stmt_error($stmt)]);
23     }
```

Query SQL yang telah disiapkan dijalankan menggunakan ‘mysqli\_prepare’ untuk memastikan query tersebut aman. Fungsi ‘mysqli\_stmt\_bind\_param’ digunakan untuk mengikat variabel *\$name*, *\$email*, *\$password*, dan *\$phone* ke masing-masing placeholder ‘?’ dalam query. Tipe parameter "ssss" menunjukkan bahwa keempat variabel adalah string. Selanjutnya kode mengeksekusi statement dengan ‘mysqli\_stmt\_execute’. Jika eksekusi berhasil, API akan mengembalikan respons JSON dengan status "success". Jika gagal, API akan mengembalikan status "failed" beserta pesan error.

### 6) Menutup Koneksi ke Database

```
25     mysqli_stmt_close($stmt);
26     mysqli_close($conn);
27 } else {
28
29     echo "{\"status\": \"unauthorized\n\"}";
```

Setelah eksekusi selesai, kode menutup statement (mysqli\_stmt\_close(\$stmt)) dan koneksi ke database (mysqli\_close(\$conn)). Ini penting untuk menghemat sumber daya server.

## c. Account (/api/my\_user/index.php)

API ini adalah endpoint yang menangani tiga fungsi utama untuk akun pengguna: mengambil data pengguna yang sedang login, mengupdate data pengguna, dan menghapus akun pengguna dari database. Penjelasan cara kerja API ini antara lain:

1) Menghubungkan ke Database

Pada baris `include ".././connection.php";`, kode ini memasukkan file `connection.php`, yang berisi pengaturan untuk koneksi ke database. Dengan memasukkan file ini, API dapat berkomunikasi dengan database sesuai dengan konfigurasi koneksi yang ada di `connection.php`.

2) Memulai Sesi Pengguna

Baris `session_start();` digunakan untuk memulai atau melanjutkan sesi PHP. Hal ini memungkinkan API untuk mengakses variabel sesi, seperti `$_SESSION['user_id']`, `$_SESSION['user_name']`, `$_SESSION['user_email']`, dan `$_SESSION['user_phone']`, yang menyimpan informasi pengguna yang sedang login.

3) Memverifikasi Sesi Aktif

Kode memeriksa apakah variabel sesi untuk pengguna (`user_id`, `user_name`, `user_email`, `user_phone`) sudah diatur. Jika sesi aktif ditemukan, API melanjutkan ke bagian selanjutnya. Jika tidak, API akan mengembalikan respons JSON dengan status "unauthorized" dan pesan "No active session found".

4) Menghapus Akun Pengguna (*DELETE Request*)

```
14          $sql = "DELETE FROM users WHERE id = $id";
```

Permintaan HTTP menggunakan metode **DELETE**, API akan menghapus akun pengguna yang sedang login berdasarkan `user_id` yang tersimpan di sesi, `mysqli_query($conn, $sql)` digunakan untuk mengeksekusi perintah SQL. Jika penghapusan berhasil, API akan mengembalikan respons "success" dalam format JSON, menghapus data sesi, dan menghancurkan sesi untuk pengguna tersebut. Jika gagal, API akan mengembalikan respons "failed" dan pesan error.

5) Mengupdate Data Pengguna (*POST Request*)

```
42          $sql = "UPDATE users SET name = ?, email = ?, phone = ? WHERE id = $id";
43          $stmt = mysqli_prepare($conn, $sql);
44          mysqli_stmt_bind_param($stmt, "sss", $name, $email, $phone);
```

Jika metode HTTP adalah POST dan data nama, email, serta phone tersedia, API akan memperbarui informasi pengguna. Data pengguna baru yaitu nama, email, dan phone akan diambil dari variabel POST, Query ini menggunakan prepared statement untuk mencegah *SQL injection*, dan `mysqli_prepare`, `mysqli_stmt_bind_param`, serta `mysqli_stmt_execute` digunakan untuk mengikat dan menjalankan query. Jika eksekusi berhasil, API akan memperbarui informasi sesi pengguna dan mengembalikan



status "success". Jika gagal, API mengembalikan status "failed" dan pesan error.

6) Mengambil Data Pengguna (GET Request)

Jika tidak ada permintaan DELETE atau POST, API akan menganggap permintaan sebagai GET (permintaan default untuk menampilkan data pengguna). API akan mengembalikan data pengguna berupa ID, nama, email, dan nomor telepon yang disimpan pada variabel sesi dalam format JSON, sehingga klien dapat menampilkan data tersebut.

7) Menutup Koneksi ke Database

```
56         mysqli_stmt_close($stmt);
57         mysqli_close($conn);
58         exit;
```

Untuk permintaan POST (update), API menutup statement dan koneksi database menggunakan `mysqli_stmt_close($stmt)` dan `mysqli_close($conn)` setelah eksekusi selesai. Ini dilakukan untuk menghemat sumber daya server.

d. Donation (/api/donation/index.php)

API ini adalah endpoint yang menangani dua fungsi utama terkait donasi membuat donasi baru dan mengambil data semua donasi dari database. API ini memungkinkan klien untuk membuat donasi baru dan mengakses semua data donasi yang tersimpan dalam database. Dengan menyertakan `connection.php`, API dapat terhubung ke database dan menjalankan operasi yang aman dengan menggunakan prepared statements untuk mencegah SQL injection. Cara kerja API ini antara lain:

1) Menghubungkan ke Database

```
1     <?php
2
3     include "../..connection.php";
```

Pada baris `include "../..connection.php";`, kode ini memasukkan file `connection.php`, yang berisi konfigurasi koneksi ke database. Ini memungkinkan API untuk terhubung ke database dan melakukan operasi CRUD.

2) Memulai Sesi Pengguna

`session_start()`; digunakan untuk memulai atau melanjutkan sesi PHP, sehingga API dapat mengakses variabel sesi yang berisi informasi pengguna yang sedang login, seperti `user_id`.

3) Memverifikasi Data Donasi (Membuat Donasi Baru)

API akan memeriksa apakah permintaan yang diterima berisi data name, phone, nominal, dan payment. Jika semua data ini tersedia, API akan menganggap permintaan sebagai permintaan untuk membuat donasi baru.

#### 4) Mengambil Data dari Input Pengguna

```
13      $email = $_POST['email'];
14      $name = $_POST['name'];
15      $phone = $_POST['phone'];
16      $nominal = $_POST['nominal'];
17      $payment = $_POST['payment'];
18      $msg = isset($_POST['msg']) ? $_POST['msg'] : '';
19      $status = "completed";
20      $id = $_SESSION['user_id'];
```

Data yang dikirimkan oleh pengguna (donor) melalui permintaan POST, seperti name, phone, nominal, payment, dan msg (pesan donasi) diambil dari variabel POST. Jika msg tidak diisi, maka nilai default akan menjadi string kosong (""). status donasi diatur sebagai "completed" secara default. id pengguna yang login diambil dari sesi aktif `$_SESSION['user_id']`.

#### 5) Menyimpan Donasi ke Database

```
22      $sql = "INSERT INTO donations(user_id, donor_name, total_donations, payment, donation_message, status) VALUES(?, ?, ?, ?, ?, ?)";
23      $stmt = mysqli_prepare($conn, $sql);
24      mysqli_stmt_bind_param($stmt, "ssssss", $id, $name, $nominal, $payment, $msg, $status);
25      if (mysqli_stmt_execute($stmt)) {
26          echo json_encode(["status" => "success"]);
27      } else {
28          echo json_encode(["status" => "failed", "error" => mysqli_stmt_error($stmt)]);
29      }
```

Query SQL `INSERT INTO donations(user_id, donor_name, total_donations, payment, donation_message, status) VALUES(?, ?, ?, ?, ?, ?)` digunakan untuk menyimpan data donasi ke dalam tabel donations.

Query ini menggunakan prepared statement untuk mencegah SQL injection, dan `mysqli_prepare`, `mysqli_stmt_bind_param`, serta `mysqli_stmt_execute` digunakan untuk mengikat dan mengeksekusi query. Jika eksekusi berhasil, API akan mengembalikan JSON dengan status "success". Jika gagal, maka API akan mengembalikan JSON dengan status "failed" serta pesan error yang diambil dari `mysqli_stmt_error($stmt)`.

#### 6) Menutup Koneksi Database (Jika Membuat Donasi Baru)

```
31      mysqli_stmt_close($stmt);
32      mysqli_close($conn);
```

Setelah selesai mengeksekusi query untuk menyimpan donasi baru, API menutup statement dan koneksi database menggunakan `mysqli_stmt_close($stmt)` dan `mysqli_close($conn)`. Ini membantu menghemat sumber daya server.

#### 7) Mengambil Data Semua Donasi (*GET Request*)

```

34     $sql = "SELECT * FROM donations";
35     $result = mysqli_query($conn, $sql);
36
37     if ($result) {
38         $donations = [];
39         while ($row = mysqli_fetch_assoc($result)) {
40             $donations[] = $row;
41         }
42         echo json_encode($donations);
43     } else {
44         echo json_encode(["status" => "error", "message" => "Failed to fetch data"]);
45     }

```

Jika permintaan tidak berisi data yang diperlukan untuk membuat donasi baru (tidak ada name, phone, nominal, dan payment), API menganggapnya sebagai permintaan GET untuk mengambil data semua donasi.

Query SQL `SELECT * FROM donations` digunakan untuk mengambil seluruh data donasi dari tabel donations.

`mysqli_query($conn, $sql)` mengeksekusi query dan hasilnya disimpan di variabel `$result`.

Jika query berhasil, API menginisialisasi array kosong `$donations`, kemudian memasukkan setiap baris hasil query ke dalam array ini melalui `mysqli_fetch_assoc($result)`. Setelah semua data dimasukkan, API mengembalikan array dalam format JSON.

Jika query gagal, API mengembalikan JSON dengan status "error" dan pesan "Failed to fetch data".

#### 8) Menutup Koneksi Database (Jika Mengambil Data Donasi)

```

46         mysqli_close($conn);
47     }

```

Setelah menyelesaikan proses pengambilan data, API akan menutup koneksi database menggunakan `mysqli_close($conn)` untuk menghemat resources.

## Progress Final Project

### 1. Back-End

Pada pengembangan ini kita mengembangkan sistem REST API pada Back-End dimana sistem ini memungkinkan operasi CRUD pada website tanpa perlu ada perpindahan halaman yang berlebihan. Sistem REST API memang tidak umum digunakan pada lingkungan PHP dan MySQL, namun pengembangannya tergolong salah satu yang cukup mudah. Hal ini juga dikarenakan gaya pengembangan PHP yang cukup simpel dan intuitif.

Pada PHP, pengembangan REST API dilakukan dengan mengimplementasikan masing-masing rute API sebagai folder. Sebagai contoh, rute API `/api/register/` memerlukan folder `'register'` yang ditempatkan dalam folder `'api'`, dalam folder tersebut akan diisi file `index.php` yang menjadi *handler* utama dari rute API tersebut.

Pada pengembangan sebelumnya, website ini masih termasuk website statis yang tidak memiliki interaksi apapun dengan database. Sehingga tidak memungkinkan adanya fungsi seperti Login, Register, modifikasi akun, dan donasi. Maka perubahan yang dilakukan pada laporan kali ini berfokus pada merubah website statis sebelumnya menjadi website dinamis yang fungsional, meskipun masih ada beberapa fungsi yang belum diimplementasikan sepenuhnya.

Langkah pertama yang dilakukan pada pengembangan Back-End website ini adalah pembuatan database sesuai dengan skema yang telah ditetapkan. Database tersebut selanjutnya dihubungkan dengan program PHP yang berjalan sebagai Back-End website ini. Koneksi antar PHP dan MySQL dilakukan dengan metode `mysqli` dan bukan dengan metode PDO (PHP Document Object). Metode ini dipilih karena implementasinya yang tergolong simpel, dan kami merasa tidak memerlukan adanya ORM (Object Relational Mapper) yang menjadi fitur utama dari PDO.

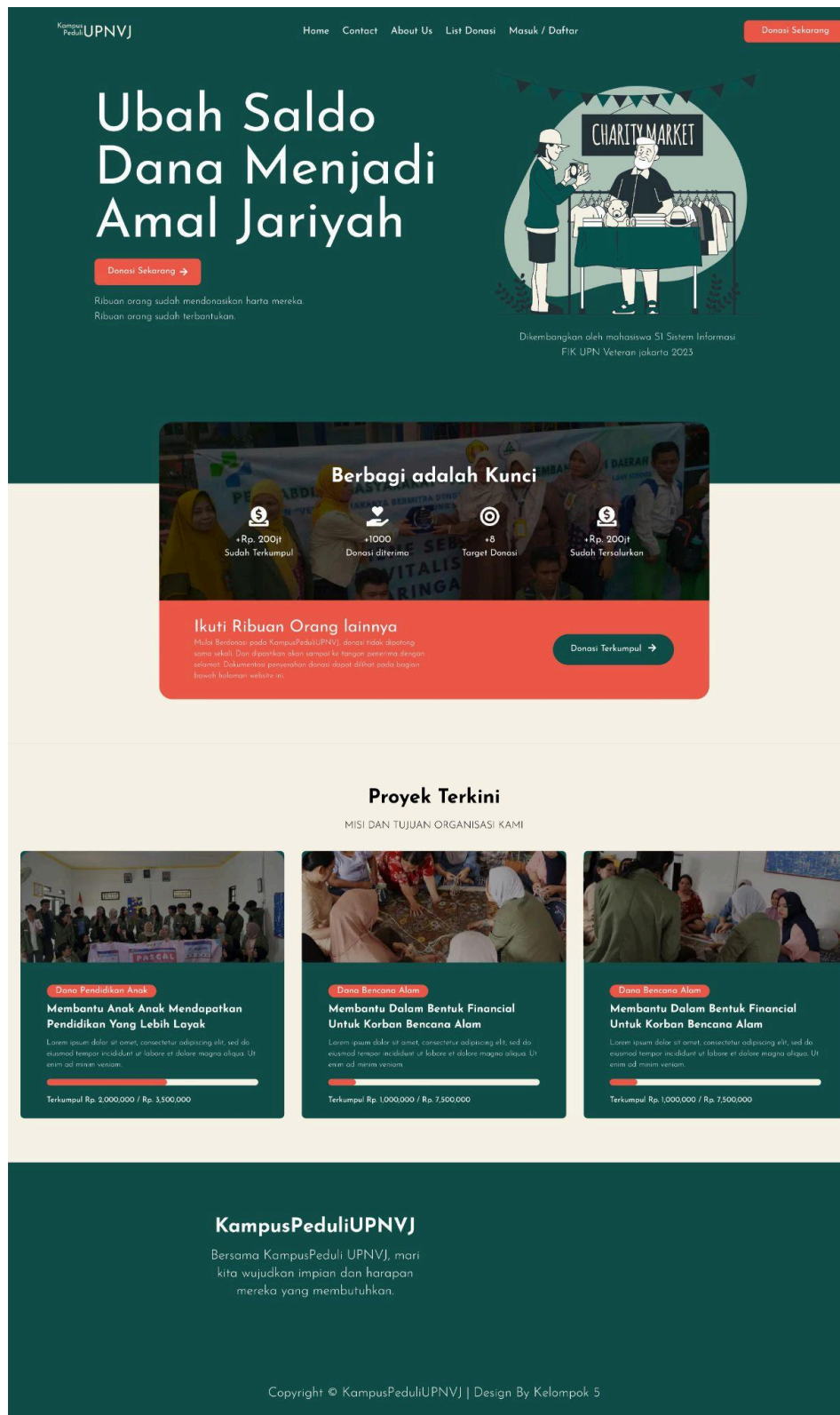
Kedua, masing-masing rute API yang diperlukan oleh website akan dibuat. Rute API tersebut masing-masing akan melakukan koneksi ke database melalui file `connection.php` dan mengeksekusi *query* yang diperlukan. Eksekusi *query* dilakukan dengan *prepared statement* sebagai langkah untuk mencegah serangan SQL Injection.

Terakhir, data yang didapat dari eksekusi *query* akan diproses oleh masing-masing rute API, sesuai dengan fungsi yang mereka jalankan. Respon dari rute API tersebut akan berupa teks JSON (Javascript Object Notation) yang mudah untuk digunakan pada Javascript bagian Front-End nantinya.

### 2. Front-End

#### a. Home

Berdasarkan Progress yang telah dilakukan sejauh ini, pada modul `'index.php'` di main path, terdapat tampilan utama website donasi yaitu sebagai berikut tampilannya:



Gambar 2.2.1 Halaman Utama Kampus Peduli UPNVJ

Terdapat navigasi bar yang memiliki 5 navigasi yang melanjutkan pada modul lain, kemudian terdapat beberapa tombol “Donasi Sekarang” untuk memudahkan pengguna untuk bisa langsung segera berdonasi tanpa perlu membuang waktu mencari tombol tersebut, kemudian di bagian hero’ terdapat ilustrasi dan juga slogan yang berkaitan dengan donasi, dibawahnya terdapat bagian untuk menampilkan statistik donasi dari *all time*, kemudian di dibawahnya lagi terdapat proyek atau donasi yang sedang dijalankan dan perlu memenuhi target donasi yang ditentukan, kemudian yang paling bawah terdapat footer yang berisi nama website dan slogan, juga copyright website, dan serta peta lokasi donasi yang tidak terlihat namun bisa dilihat jika mengakses tautan berikut: <https://kampuspeduliupnvj.loophole.site>

## b. Contact

Kemudian pada menu modul di path contact, seperti nama nya, menu ini memperlihatkan contact dari admin website, berikut adalah tampilanya:

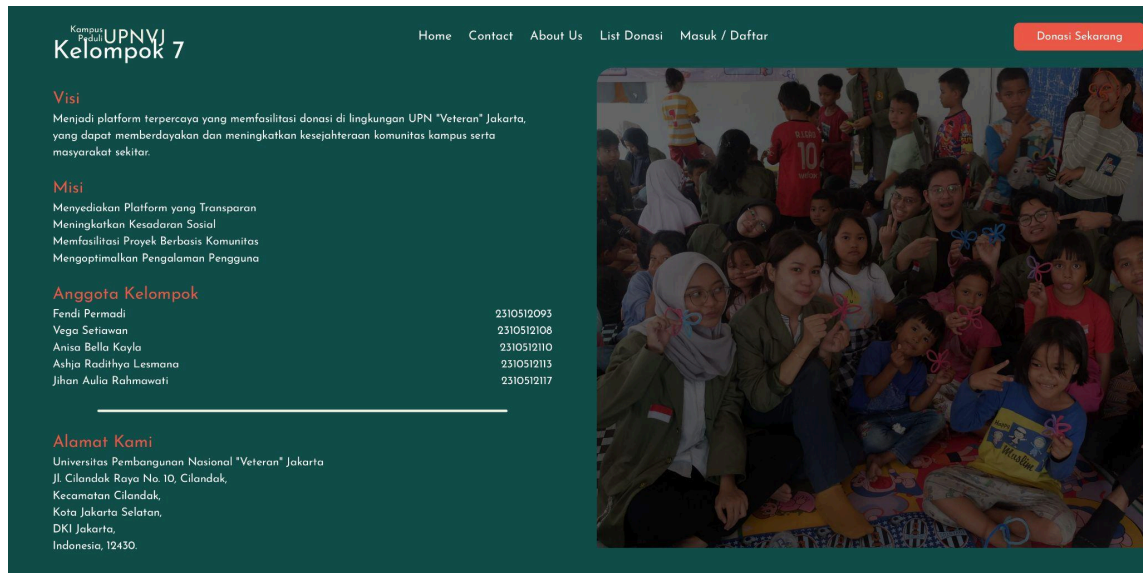
*Gambar 2.2.2 Halaman Kontak Kami Kampus Peduli UPNVJ*

Halaman "Kontak Kami" memiliki struktur yang rapi untuk memudahkan pengguna dalam menghubungi atau mencari informasi. Di bagian atas terdapat header dengan menu navigasi ke berbagai halaman, seperti Home, Contact, About Us, List Donasi, Masuk/Daftar, dan tombol "Donasi Sekarang". Bagian utama halaman menampilkan form kontak berisi kolom untuk Nama Lengkap, Email, dan Pesan, serta tombol oranye bertuliskan "Kirim" untuk mengirimkan pesan. Selain form, terdapat informasi kontak alternatif yang mencakup ikon WhatsApp, Instagram, dan email yang dapat diakses pengguna untuk menghubungi tim support melalui platform pilihan mereka. Di bagian bawah form, terdapat keterangan mengenai privasi data yang menekankan bahwa interaksi dengan tim support dilakukan secara aman. Di sisi kanan, terdapat foto dokumentasi kegiatan

yang memperlihatkan aktivitas sosial, memberikan kesan interaktif dan keterlibatan langsung dengan masyarakat. Struktur ini dirancang untuk memberikan kemudahan dan kepercayaan kepada pengguna yang ingin berinteraksi atau berkontribusi di platform Kampus Peduli UPNVJ.

### c. About\_Us

Setelah itu pada menu modul di path about\_us, seperti nama nya, menu ini memperlihatkan informasi visi, misi, anggota dan juga alamat website ini, berikut adalah tampilanya:



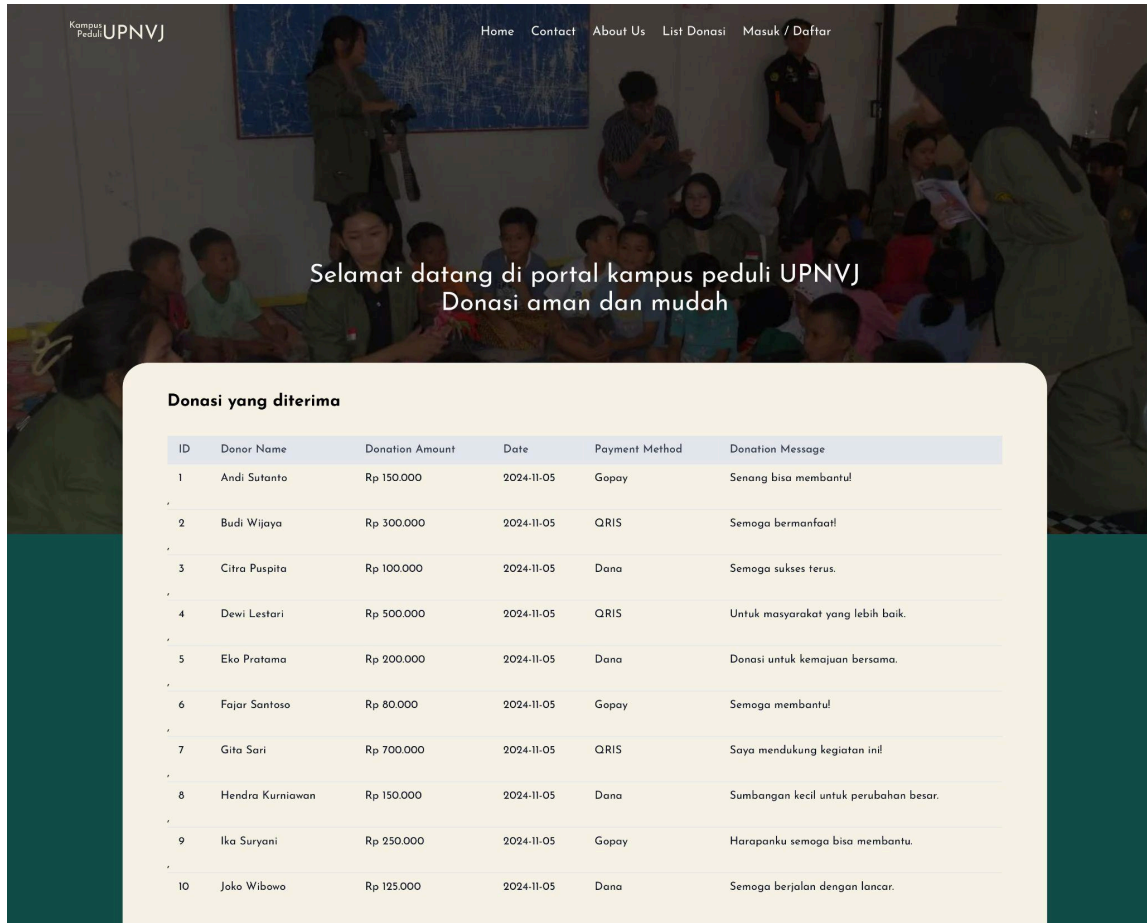
*Gambar 2.2.3 Halaman About Us Kampus Peduli UPNVJ*

Tampilan website ini adalah halaman "About Us" dari platform donasi komunitas bernama "Kampus Peduli UPNVJ Kelompok 7," yang dikembangkan oleh mahasiswa Universitas Pembangunan Nasional "Veteran" Jakarta. Halaman ini memiliki struktur yang mencakup judul, navigasi menu di bagian atas (Home, Contact, About Us, List Donasi, Masuk/Daftar, serta tombol "Donasi Sekarang"). Terdapat bagian "Visi" dan "Misi" yang menjelaskan tujuan platform ini, yaitu menjadi platform terpercaya untuk memfasilitasi donasi dalam komunitas kampus serta memberdayakan masyarakat sekitar dengan transparansi, meningkatkan kesadaran sosial, dan mengoptimalkan pengalaman pengguna. Di bawahnya terdapat daftar anggota kelompok beserta nomor identitas masing-masing, serta informasi alamat kampus sebagai tempat organisasi tersebut berada. Desainnya mengombinasikan elemen teks dengan foto anggota kelompok yang sedang bersama anak-anak, mencerminkan misi sosial mereka dalam memberdayakan masyarakat.



#### d. List\_Donations

Setelah itu pada menu modul di path list\_donasi, menu ini memperlihatkan laporan keuangan donasi yang sudah diterima dari semua donatur, berikut adalah tampilanya:



ID	Donor Name	Donation Amount	Date	Payment Method	Donation Message
1	Andi Sutanto	Rp 150.000	2024-11-05	Gopay	Senang bisa membantu
2	Budi Wijaya	Rp 300.000	2024-11-05	QRIS	Semoga bermanfaat!
3	Citra Puspita	Rp 100.000	2024-11-05	Dana	Semoga sukses terus.
4	Dewi Lestari	Rp 500.000	2024-11-05	QRIS	Untuk masyarakat yang lebih baik.
5	Eko Pratama	Rp 200.000	2024-11-05	Dana	Donasi untuk kemajuan bersama.
6	Fajar Santoso	Rp 80.000	2024-11-05	Gopay	Semoga membantu!
7	Gita Sari	Rp 700.000	2024-11-05	QRIS	Saya mendukung kegiatan ini!
8	Hendra Kurniawan	Rp 150.000	2024-11-05	Dana	Sumbangan kecil untuk perubahan besar.
9	Ika Suryani	Rp 250.000	2024-11-05	Gopay	Harapanku semoga bisa membantu.
10	Joko Wibowo	Rp 125.000	2024-11-05	Dana	Semoga berjalan dengan lancar.

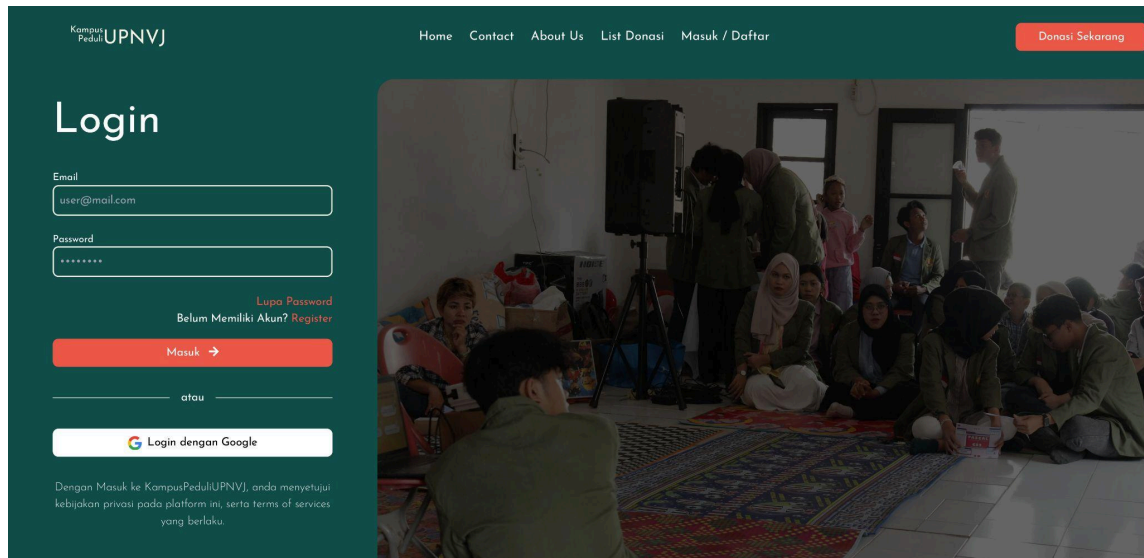
*Gambar 2.2.4 Halaman List Donasi Kampus Peduli UPNVJ*

Tampilan website ini adalah halaman “List Donasi” dari platform donasi komunitas "Kampus Peduli UPNVJ." Halaman ini memiliki struktur yang mencakup judul di bagian atas, navigasi menu (Home, Contact, About Us, List Donasi, Masuk/Daftar), serta latar belakang foto kegiatan sosial. Di bagian tengah halaman terdapat tabel dengan judul "Donasi yang diterima," yang menampilkan daftar donasi yang diterima oleh platform. Tabel ini memiliki kolom untuk ID, Nama Donatur, Jumlah Donasi, Tanggal, Metode Pembayaran, dan Pesan Donasi, yang mencatat setiap donasi dengan detail lengkap, termasuk nama donatur dan pesan motivasi mereka. Desain ini dibuat sederhana namun informatif untuk transparansi, memastikan pengguna bisa melihat riwayat donasi yang masuk beserta detail transaksinya.

#### e. Login



Setelah itu pada menu modul di path login, menu ini memperlihatkan autentikasi akun atau masuk ke akun yang sudah pernah dibuat, berikut adalah tampilanya:



*Gambar 2.2.5 Halaman Login Kampus Peduli UPNVJ*

Tampilan website ini adalah halaman “Login” dari platform donasi komunitas "Kampus Peduli UPNVJ." Halaman ini memiliki struktur yang terdiri dari navigasi menu di bagian atas (Home, Contact, About Us, List Donasi, Masuk/Daftar) dan tombol “Donasi Sekarang” yang berwarna mencolok di pojok kanan atas untuk menarik perhatian pengguna. Di bagian utama terdapat formulir login dengan kolom untuk mengisi email dan password, serta tautan “Lupa Password” dan “Register” untuk pengguna yang belum memiliki akun. Tombol “Masuk” berwarna merah mudah diakses, dan terdapat juga opsi “Login dengan Google” sebagai alternatif metode masuk. Pada bagian bawah, terdapat teks yang menjelaskan bahwa dengan masuk ke platform, pengguna menyetujui kebijakan privasi dan terms of service yang berlaku. Desain ini dilengkapi dengan latar belakang foto kegiatan sosial, menciptakan nuansa yang selaras dengan tujuan sosial dari platform ini.

#### **f. Donations**

Setelah itu pada menu modul di path donations, menu ini memperlihatkan formulir untuk pengajuan donasi, berikut adalah tampilanya:

Kampus Peduli UPNVJ

Home Contact About Us List Donasi Masuk / Daftar

Selamat datang di portal kampus peduli UPNVJ  
Donasi aman dan mudah

**Nominal**  
Pilih nominal yang tersedia

Rp 10.000 Rp 25.000 Rp 75.000 Rp 100.000

**Nominal Lainnya**

Rp.125.000

Minimum donasi Rp 10.000

**Metode Pembayaran**

▼ Gopay

**Info Donatur**  
*Masuk akun atau lengkapi data dibawah ini*

**Nama Lengkap\***

Agung Prasasto

**No Telepon\***

+62xxxxxxxxx

**Email (Opsional)**

user@mail.com

**Sembunyikan nama saya (Anonim)**

**Beri Pesan (Opsional)**

Pesan...

Dengan melanjutkan donasi, saya setuju Syarat & Ketentuan

**Donasi Sekarang**

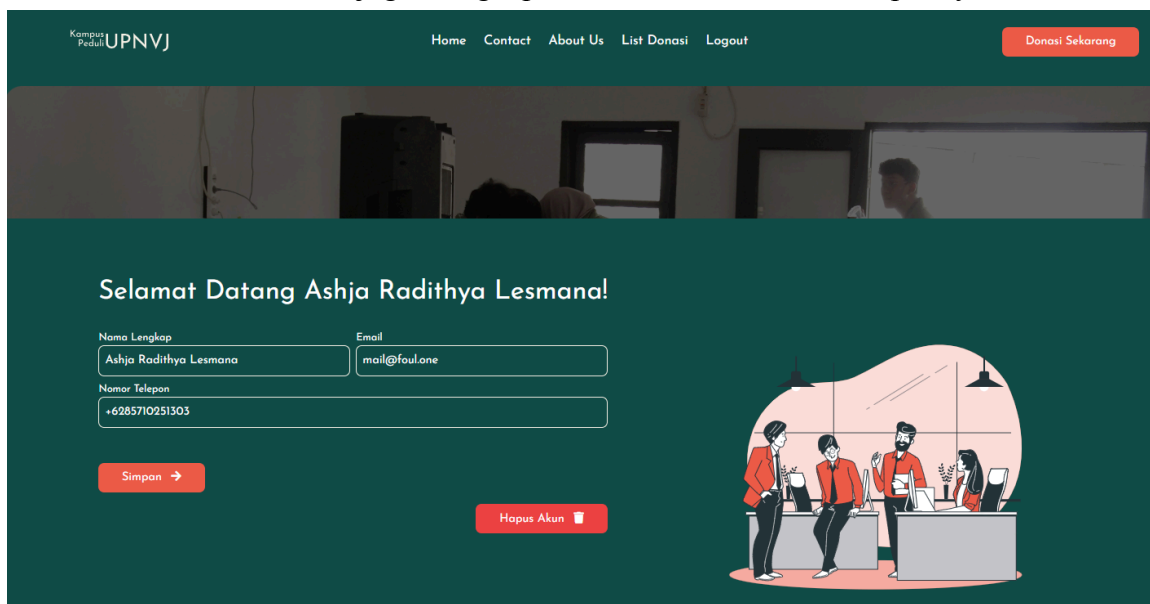
*Gambar 2.2.6 Halaman Donasi Kampus Peduli UPNVJ*

Gambar yang ditampilkan merupakan halaman ‘donasi’. Tampilan ini terdiri dari beberapa bagian utama, yaitu header dengan navigasi menu berisi tautan seperti "Home," "Contact," "About Us," "List Donasi," dan "Masuk/Daftar." Di bawah header terdapat sambutan singkat yang menginformasikan bahwa situs ini menyediakan donasi yang aman dan mudah. Bagian utama halaman ini adalah formulir donasi, yang dimulai dengan pilihan nominal donasi mulai dari Rp 10.000 hingga Rp 100.000 serta kolom untuk

memasukkan nominal lainnya sesuai keinginan donatur. Selanjutnya, ada pilihan metode pembayaran, yang dalam gambar ini terpilih opsi "Gopay." Di bawahnya, terdapat informasi donatur yang meliputi nama lengkap, nomor telepon, dan email (opsional). Pengguna juga dapat memilih untuk menyembunyikan namanya (donasi anonim) dan memberikan pesan opsional. Di bagian bawah terdapat tombol "Donasi Sekarang" yang memungkinkan pengguna melanjutkan proses donasi setelah menyetujui syarat dan ketentuan. Struktur halaman ini dirancang agar proses donasi menjadi jelas dan mudah diikuti oleh pengguna.

#### g. Dashboard

Setelah itu pada menu modul di path dashboard, menu ini memperlihatkan menu edit akun dan juga menghapus akun, berikut adalah tampilannya:



*Gambar 2.2.7 Halaman Dashboard Kampus Peduli UPNVJ*

Gambar ini menampilkan halaman dashboard atau profil pengguna. Struktur halaman terdiri dari header dengan menu navigasi, termasuk "Home," "Contact," "About Us," "List Donasi," dan "Logout," serta tombol "Donasi Sekarang" di pojok kanan atas. Di bagian utama, terdapat ucapan sambutan personal yang menampilkan nama pengguna, diikuti dengan formulir berisi informasi pribadi, seperti nama lengkap, email, dan nomor telepon yang dapat diedit oleh pengguna. Di bawah formulir ini terdapat tombol "Simpan" untuk memperbarui informasi pengguna dan tombol "Hapus Akun" berwarna merah untuk menghapus akun. Di sebelah kanan, terdapat ilustrasi grafis yang menggambarkan aktivitas kolaboratif, menambahkan elemen visual yang menarik pada halaman. Struktur halaman ini dirancang untuk memudahkan pengguna dalam mengelola informasi profil mereka dan memberikan akses cepat untuk tindakan lebih lanjut seperti logout atau menghapus akun.

## h. Register

Setelah itu pada menu modul di path register, menu ini memperlihatkan formulir untuk pembuatan akun, berikut adalah tampilanya:

*Gambar 2.2.8 Halaman Register Kampus Peduli UPNVJ*

Gambar ini menunjukkan halaman registrasi. Pada tampilan ini, struktur halaman terdiri dari beberapa elemen, dimulai dengan header yang menyediakan navigasi utama, termasuk tautan "Home," "Contact," "About Us," "List Donasi," serta "Masuk/Daftar," dan tombol "Donasi Sekarang" di pojok kanan atas. Bagian utama halaman adalah formulir registrasi yang meminta pengguna untuk mengisi data diri, seperti nama lengkap, email, kata sandi (dan konfirmasinya), serta nomor telepon. Di bawah kolom input ini terdapat opsi untuk login jika sudah memiliki akun, serta tombol "Daftar" untuk menyelesaikan pendaftaran. Ada juga pilihan login menggunakan akun Google, memberikan alternatif akses yang lebih mudah. Pada bagian bawah, terdapat catatan bahwa dengan mendaftar, pengguna setuju dengan kebijakan privasi dan syarat layanan yang berlaku di platform ini. Struktur halaman ini dirancang untuk memudahkan proses pendaftaran pengguna baru dengan tampilan yang sederhana dan mudah diikuti.

## 3. Skema Database

Implementasi CRUD pada sistem donasi Kampus Peduli UPNVJ memerlukan sebuah database dengan tiga tabel yang sudah diterapkan pada pertemuan sebelumnya. Namun ada sedikit perubahan dari beberapa skema tabel yang digunakan. Skema ketiga tabel tersebut adalah sebagai berikut:

a. Tabel Users

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	
password	varchar(255)	YES		NULL	
created_at	datetime	YES		current_timestamp()	
updated_at	datetime	YES		current_timestamp()	on update current_timestamp()
phone	varchar(255)	YES		NULL	

7 rows in set (0.009 sec)

Gambar 2.2.9 Kolom Tabel Users

```
CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255),
  email VARCHAR(100),
  password VARCHAR(255),
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  phone VARCHAR(255)
);
```

Gambar 2.2.10 Query Pembuatan Tabel Users

b. Tabel Donations

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	YES	MUL	NULL	
donor_name	varchar(100)	NO		NULL	
total_donations	decimal(10,2)	NO		NULL	
created_at	datetime	YES		current_timestamp()	
payment	varchar(100)	NO		NULL	
donation_message	text	YES		NULL	
status	enum('completed','pending','failed')	YES		pending	

Gambar 2.2.11 Kolom Tabel Donations

```
CREATE TABLE donations (
  id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT,
  donor_name VARCHAR(100) NOT NULL,
  total_donations DECIMAL(10, 2) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  payment VARCHAR(100) NOT NULL,
  donation_message TEXT,
  status ENUM('completed', 'pending', 'failed') DEFAULT 'pending',
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Gambar 2.2.12 Query Pembuatan Tabel Donations

c. Tabel Reports

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
donation_id	int(11)	YES	MUL	NULL	
report_date	datetime	YES		current_timestamp()	
details	text	YES		NULL	
impact	text	YES		NULL	
created_at	datetime	YES		current_timestamp()	
updated_at	datetime	YES		current_timestamp()	on update current_timestamp()

Gambar 2.2.13 Kolom Tabel Reports

```
CREATE TABLE reports (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  donation_id INT,  
  report_date DATETIME DEFAULT CURRENT_TIMESTAMP,  
  details TEXT,  
  impact TEXT,  
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,  
  updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  FOREIGN KEY (donation_id) REFERENCES donations(id)  
);
```

Gambar 2.2.14 Query Pembuatan Tabel Reports

#### 4. Langkah Selanjutnya

Untuk menjadi aplikasi yang berjalan sepenuhnya, ada beberapa fitur yang harus dikembangkan dan beberapa aspek yang harus diperbaiki kembali. Fitur yang perlu diimplementasikan selanjutnya mencakup: laporan penyaluran donasi, moderasi donasi (operasi CRUD pada donasi) yang bisa dilakukan oleh pengguna website dengan akses admin, dan fungsi logout akun. Beberapa fitur tersebut membutuhkan modifikasi skema tabel berupa penambahan kolom baru, seperti kolom *role* pada tabel Users yang mendefinisikan akses seorang pengguna (admin atau pengguna biasa).

Selain itu, beberapa aspek pada website ini juga perlu diperhatikan kembali. Contohnya apakah website responsif pada setiap jenis *device*, berapa lama *load time* website, maksimum pengguna yang bisa menggunakan website secara bersamaan, serta cara untuk mengoptimalkan semuanya. Pada akhir project, diharapkan website sudah memiliki fungsi yang lengkap, responsif untuk diakses semua *device* dan memiliki *load time* yang cepat (kurang dari satu detik) bahkan pada saat digunakan oleh banyak pengguna.