BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Sebagai Mahasiswa Magang di PT. PLN Nusantara Power Unit Pembangkit Muara Karang, penempatan kerja dilakukan di bawah divisi *Assistant Manager System Owner* sebagai *Fullstack Developer*, dengan pembimbing lapangan Bapak Qaasim selaku mentor, serta pengawasan langsung dari Bapak Bernandez Noverson Lupy selaku *Assistant Manager System Owner*. Komunikasi dan koordinasi antar sesama *developer* dilakukan secara langsung di lingkungan kerja tanpa menggunakan aplikasi pesan seperti *WhatsApp*. Untuk pengelolaan kode dan kolaborasi dalam proses pengembangan proyek, digunakan platform *GitHub* sebagai sarana koordinasi teknis. Melalui *GitHub*, anggota tim dapat mengunggah, meninjau, dan mengintegrasikan kode, serta memantau perubahan yang terjadi secara dalam repositori proyek yang telah disepakati bersama.

3.2 Tugas yang Dilakukan

Sebagai mahasiswa magang yang memiliki peran *Fullstack Developer* pada PT. PLN Nusantara Power Unit Pembangkit Muara Karang, tugas yang dilakukan selama proses magang difokuskan pada proyek pengembangan sistem pengelolaan Asset IT berbasis web. Dalam pelaksanaan proyek ini, tanggung jawab utama mencakup pengembangan lima halaman utama pada website, yaitu halaman *Landing Page*, *Dashboard*, *CRUD Management*, *User Management* dan *Request Management*.

3.3 Uraian Pelaksanaan Magang

Rincian pekerjaan mingguan selama kegiatan magang di PT. PLN Nusantara Power Unit Pembangkit Muara Karang disajikan pada Tabel 3.1 berikut.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama magang

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Perkenalan lingkungan kerja magang di PT. PLN Nusantara Power Muara Karang. Dan mengikuti acara bulan K3 yang diikuti oleh seluruh karyawan PT. PLN Nusantara Power Unit Pembangkit Muara Karang.
2	Melakukan diskusi bersama mentor dan supervisor terkait permasalahan yang dihadapi oleh tim IT, sebagai dasar dalam merancang solusi melalui sistem yang akan dikembangkan.
3	Mempelajari ulang framework Laravel dan Bootstrap sebagai persiapan untuk pengembangan website yang akan dikerjakan.
4	Menerima desain halaman *Landing Page* dari *UI/UX Developer* dan mulai mengerjakannya.
5	Memperbaiki beberapa *bug* pada halaman *Landing Page* serta menyelesaikan pengembangannya. Melakukan diskusi dan menerima masukan dari tim IT, serta meeting dengan mentor membahas progres proyek.
6	Mengikuti presentasi produk Realwear Navigator 520. Mengembangkan komponen *navbar*, *footer*, dan *sidebar*, serta memperbaiki bug pada *footer*. Membantu persiapan seminar ergonomi tulang dan berdiskusi dengan developer mengenai rancangan basis data. Menyampaikan progres mingguan kepada mentor.
7	Menyelesaikan dan memperbaiki *bug* pada *sidebar*, serta mulai mengembangkan halaman *dashboard*. Diskusi dengan tim IT mengenai identitas pengguna untuk basis data, serta memperbaiki bug pada tabel ”Asset to be Purchased”. Mengikuti acara penutupan K3 Nasional.
8	Menyelesaikan halaman *dashboard* dan mulai mengerjakan halaman CRUD. Melakukan diskusi mingguan bersama mentor dan membahas fitur tambahan pada halaman *request*.
	Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
9	Memperbaiki tampilan halaman CRUD dan berdiskusi dengan UI/UX designer mengenai fitur *document request*. Mulai membuat modal untuk unduh Excel dan mempelajari Laravel Excel. Membantu persiapan rapat dan menyampaikan progres mingguan kepada mentor.
10	Membantu tim IT dalam menyiapkan acara atau rapat, serta menyelesaikan tampilan modal untuk fitur unduh Excel dan memperbaiki bug terkait heading yang tidak muncul pada file yang diunduh. Selain itu, mulai mengembangkan halaman User Management yang akan digunakan oleh Super Admin, dan menyampaikan laporan mingguan kepada mentor.
11	Membuat tampilan tabel pada halaman user management yang akan dilihat oleh User serta berdiskusi dengan UI/UX designer
12	Membantu tim IT dalam menyiapkan dan mengikuti acara Halal Bihalal PT. PLN UP Nusantara Power yang dihadiri oleh seluruh unit pembangkit. Membantu merapikan data Excel divisi System Owner, menambahkan fitur *Add New Account* pada halaman *User Management*, serta membuat tampilan tabel dapat digulir secara horizontal pada layar berukuran kecil.
13	Membantu tim IT dalam menyiapkan acara atau rapat. Memastikan halaman *Dashboard* bebas dari *bug* dan membuat tampilan tabel pada halaman tersebut dapat digulir secara horizontal. Selain itu, merapikan tampilan halaman *Landing Page* dan menyelesaikan penyesuaian agar halaman tersebut menjadi responsif pada berbagai ukuran layar.
	Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	**Pekerjaan yang dilakukan**
14	Membantu tim IT merapikan data Excel divisi *System Owner* serta menyiapkan acara atau rapat. Mengubah logika pada halaman *CRUD Editor* agar penamaan *type asset* tidak terpengaruh oleh huruf kapital atau kecil. Selain itu, menyusun tampilan *dashboard* agar responsif dan berdiskusi dengan UI/UX designer terkait penerapan *responsive design*. Di akhir minggu, melaporkan progres kepada mentor dan berdiskusi bersama developer mengenai kekurangan dalam proyek yang sedang dikembangkan.
15	Memperbaiki desain halaman *dashboard* agar sesuai dengan tampilan terbaru dari Figma, serta memperbaiki *bug* pada halaman *User Management*. Selain itu, melakukan laporan mingguan kepada mentor dan berdiskusi dengan *developer* terkait kekurangan yang masih terdapat pada proyek yang sedang dikembangkan.
16	Merapikan desain halaman *User Management* dan *Assets Editor* agar sesuai dengan desain terbaru dari Figma. Mengubah tampilan sidebar menjadi hamburger menu pada tampilan layar kecil serta menghapus footer pada halaman setelah login. Selain itu, membantu dan mengikuti acara donor darah, serta melakukan laporan mingguan kepada mentor.
17	melanjutkan perbaikan design halaman Assets Editor sesuai dengan design figma serta membantu tim IT menyiapkan acara atau rapat. Selain itu melakukan Laporan mingguan dengan mentor serta diskusi bersama dengan ui/ux membahas kembali mengenai kekurangan projek yang sedang di buat
	Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
18	Memperbaiki tampilan modal pada fitur *document request* sesuai desain terbaru dari Figma. Membantu dan mengikuti acara Kebangkitan Nasional serta membantu tim IT dalam menyiapkan acara seleksi penghargaan karya inovasi PT. PLN Nusantara Power. Melakukan laporan mingguan kepada mentor dan berdiskusi bersama UI/UX designer mengenai kekurangan proyek yang sedang dikembangkan.
19	Mengembangkan tampilan halaman *Assets Editor* dan *User Management* agar responsif sesuai dengan desain terbaru dari Figma.
20	Memastikan Tidak ada bug pada website, serta memastikan tampilan sudah sesuai dengan design di figma. Serta membantu tim IT dalam menyiapkan acara atau rapat.
21	Melakukan presentasi *project* kepada *Supervisor* dan Mentor magang. Dan menambahkan fitur baru berupa *Request Management*.

3.4 Perancangan dan Implementasi

Pada bagian ini, akan dijelaskan bagaimana rancangan sistem dikembangkan serta bagaimana implementasinya dilakukan berdasarkan kebutuhan pengelolaan aset IT di PT. PLN Nusantara Power Unit Pembangkit Muara Karang.

3.4.1 Flowchart Website Pegelolaan Aset IT PT. PLN NP UP Muara Karang

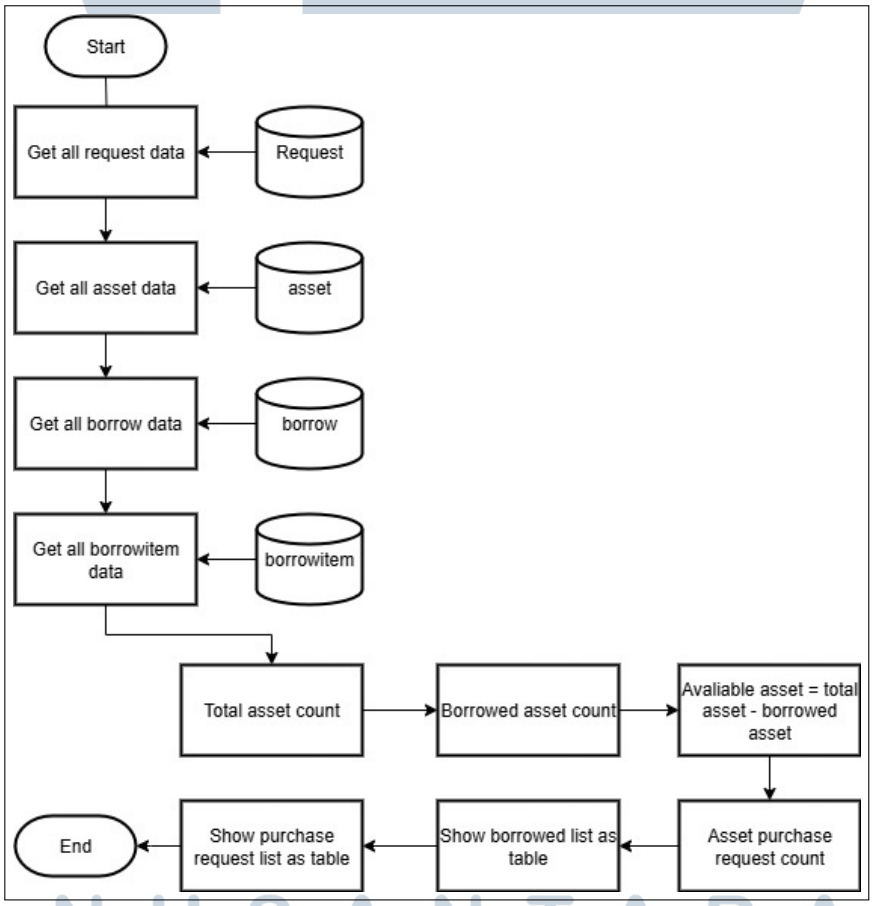
Dibawah ini merupakan *flowchart* yang akan berfungsi sebagai dasar alur website yang akan dibuat.

A Flowchart Halaman Dashboard

Gambar 3.1 menggambarkan alur kerja sistem *dashboard* untuk menampilkan informasi pengelolaan aset IT berdasarkan data yang diambil dari beberapa tabel utama dalam *database*. Proses dimulai dengan pengambilan seluruh data permintaan dari tabel *Request*, dilanjutkan dengan pengambilan data

aset dari tabel *asset*, data peminjaman dari tabel *borrow*, dan data detail *item* yang dipinjam dari tabel *borrowitem*. Setelah semua data berhasil dikumpulkan, sistem menghitung total aset yang dimiliki dan jumlah aset yang sedang dipinjam. Kemudian, dilakukan perhitungan jumlah aset yang masih tersedia dengan mengurangkan *borrowed asset count* dari *total aset* selain itu sistem juga akan melakukan perhitungan jumlah permintaan pembelian dihitung pada proses *Asset purchase request count*.

Selain melakukan perhitungan, sistem juga menampilkan data dalam bentuk tabel. Daftar permintaan pembelian aset ditampilkan sebagai tabel melalui proses Show purchase request list as table, daftar aset yang sedang dipinjam ditampilkan melalui Show borrowed list as table. Seluruh informasi ini ditampilkan dalam dashboard sebagai hasil akhir dari proses, sebelum sistem mencapai End.



Gambar 3.1. Flowchart halaman Dashboard

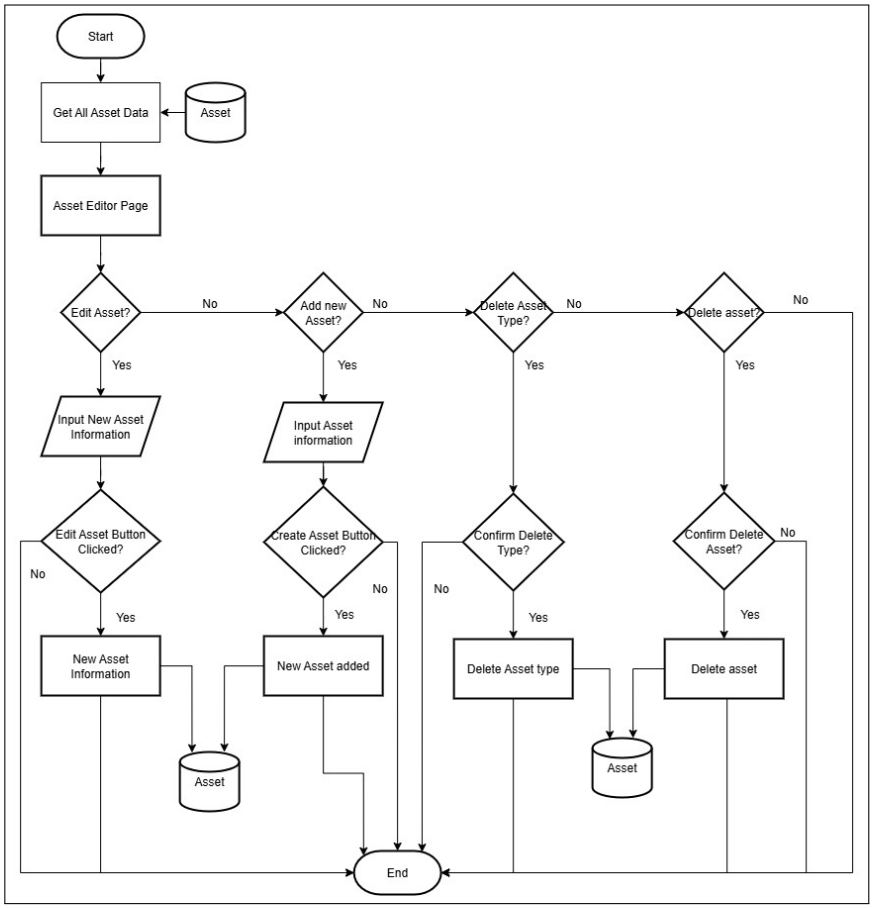
B Flowchart Halaman Asset Editor

Gambar 3.2 menunjukkan alur ketika pengguna mengakses halaman *Asset Editor*. Sistem terlebih dahulu mengambil seluruh data aset dari *database*, kemudian menampilkannya pada halaman *Asset Editor*. Di halaman ini, pengguna dapat melakukan berbagai pengelolaan data aset, seperti mengedit informasi aset, menambahkan aset baru, menghapus jenis aset, maupun menghapus aset tertentu.

Jika pengguna memilih untuk mengedit data aset, sistem akan meminta pengguna untuk mengisi informasi baru pada *form* yang tersedia. Setelah itu, sistem akan memeriksa apakah tombol *Edit Asset* diklik. Jika ya, maka data baru akan disimpan ke dalam *database* jika tidak, maka perubahan dibatalkan.

Pada proses penambahan aset, pengguna harus mengisi informasi aset terlebih dahulu. Setelah selesai, pengguna dapat menekan tombol *Create Asset*. Jika tombol ini ditekan, aset baru akan ditambahkan ke dalam *database* sebaliknya, jika tidak ditekan, maka proses penambahan dibatalkan.

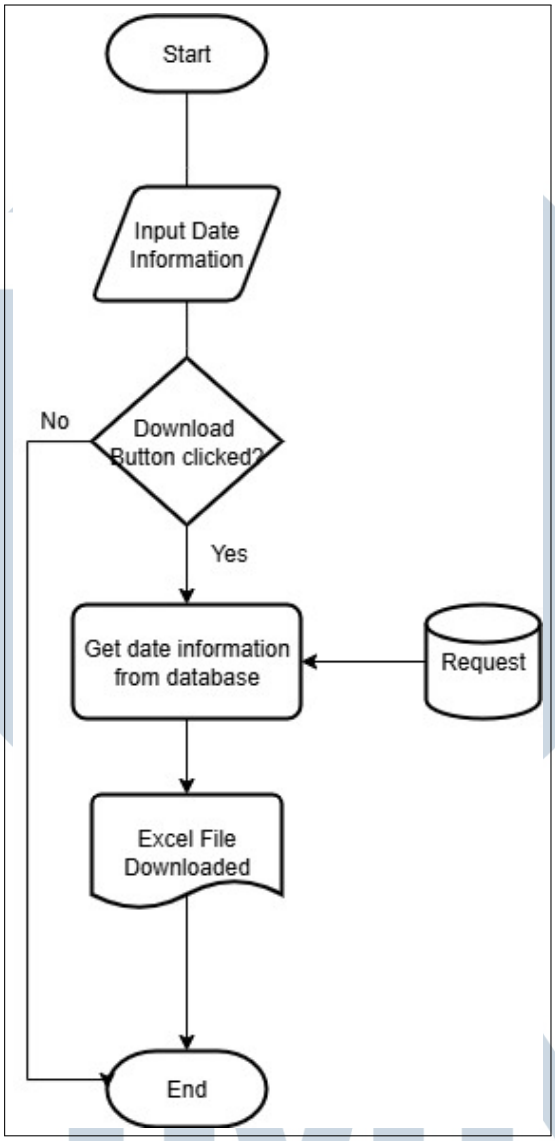
Untuk proses penghapusan, sistem membedakan antara penghapusan jenis aset dan penghapusan aset tertentu. Pada kedua jenis penghapusan tersebut, sistem akan menampilkan konfirmasi terlebih dahulu. Penghapusan hanya akan diproses jika pengguna menyetujui konfirmasi melalui tombol *Confirm Delete*. Jika tidak, sistem akan membatalkan proses tersebut dan tidak ada perubahan data yang dilakukan.



Gambar 3.2. Flowchart halaman Asset Editor

C Flowchart Modal Document Request

Gambar 3.3 merupakan sistem yang akan menampilkan fitur pengunduhan data permintaan (*request*) dalam format file *Excel*. Pengguna diminta untuk memasukkan informasi tanggal sebagai dasar pemfilteran data. Setelah informasi tanggal dimasukkan, pengguna memiliki opsi untuk menekan tombol *Download*. Jika tombol tersebut ditekan, sistem akan mengambil data permintaan dari *database* berdasarkan tanggal yang telah dimasukkan, lalu secara otomatis menghasilkan dan mengunduh file *Excel* yang memuat data tersebut. Sebaliknya, jika tombol *Download* tidak ditekan, maka proses akan dibatalkan dan tidak ada file yang dihasilkan.



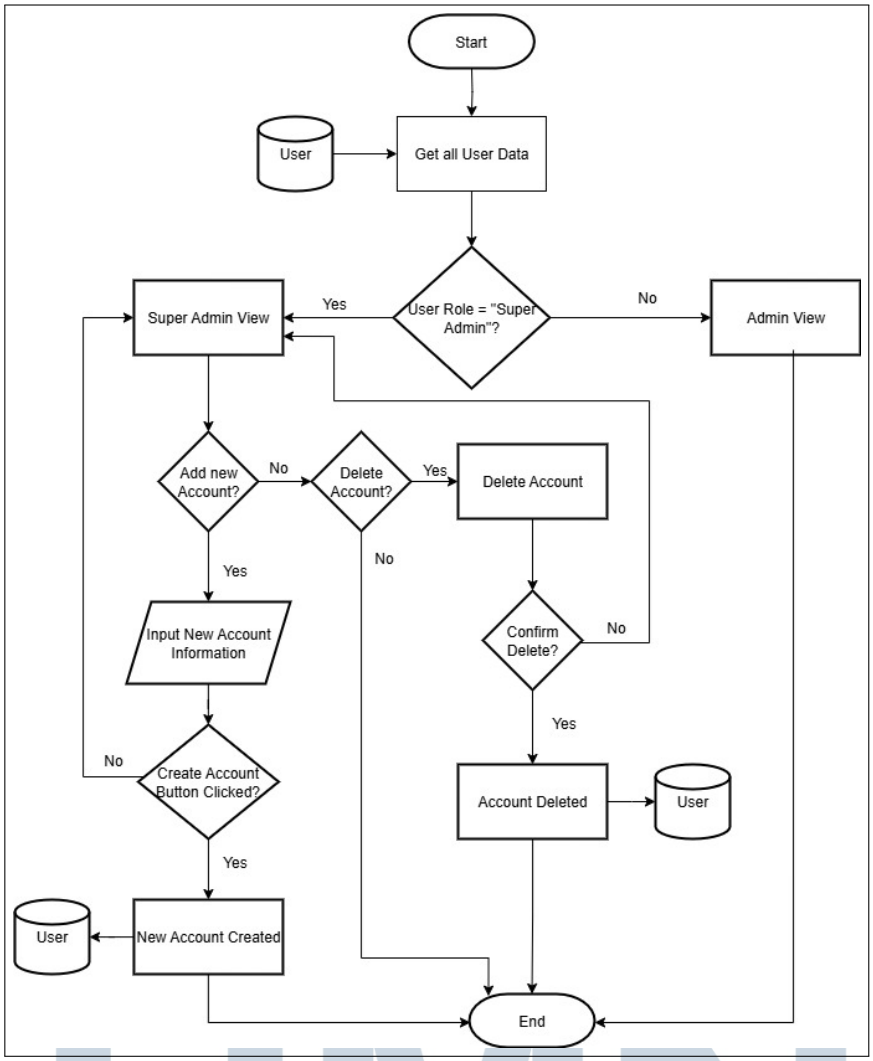
Gambar 3.3. Flowchart modal Document Request

D Flowchart Halaman User Management

Pada Gambar 3.4 ditunjukkan alur kerja dari halaman *User Management* yang berfungsi untuk mengelola data akun pengguna dalam sistem. Setelah sistem dijalankan, proses pertama yang dilakukan adalah mengambil seluruh data pengguna dari *database*. Selanjutnya, sistem akan melakukan pengecekan terhadap peran pengguna (*user role*). Jika pengguna yang mengakses halaman tersebut memiliki peran sebagai *Super Admin*, maka sistem akan menampilkan *Super Admin View* yang menyediakan fitur penuh terhadap manajemen akun. Sebaliknya, jika

pengguna bukan merupakan *Super Admin*, maka yang ditampilkan adalah *Admin View* dengan fitur terbatas.

Pada tampilan *Super Admin View*, tersedia dua fitur utama yang dapat diakses oleh pengguna dengan peran *Super Admin*, yaitu *Add New Account* dan *Delete Account*. Fitur *Add New Account* memungkinkan *Super Admin* untuk menambahkan akun baru ke dalam sistem. Ketika fitur ini dipilih, sistem akan menampilkan formulir input yang harus diisi dengan informasi akun yang akan dibuat. Setelah seluruh data diinput, pengguna dapat memilih tombol *Create Account* untuk menyimpan akun baru tersebut ke dalam *Database*. Namun, apabila pengguna tidak menekan tombol tersebut, maka proses pembuatan akun akan dibatalkan dan sistem akan kembali ke tampilan utama *Super Admin View*. Di sisi lain, apabila *Super Admin* memilih untuk menghapus akun, sistem akan menampilkan permintaan konfirmasi penghapusan terlebih dahulu. Jika tombol *Delete* di tekan, maka akun yang dipilih akan dihapus.



Gambar 3.4. Flowchart halaman User Management

E Flowchart Halaman Request Management

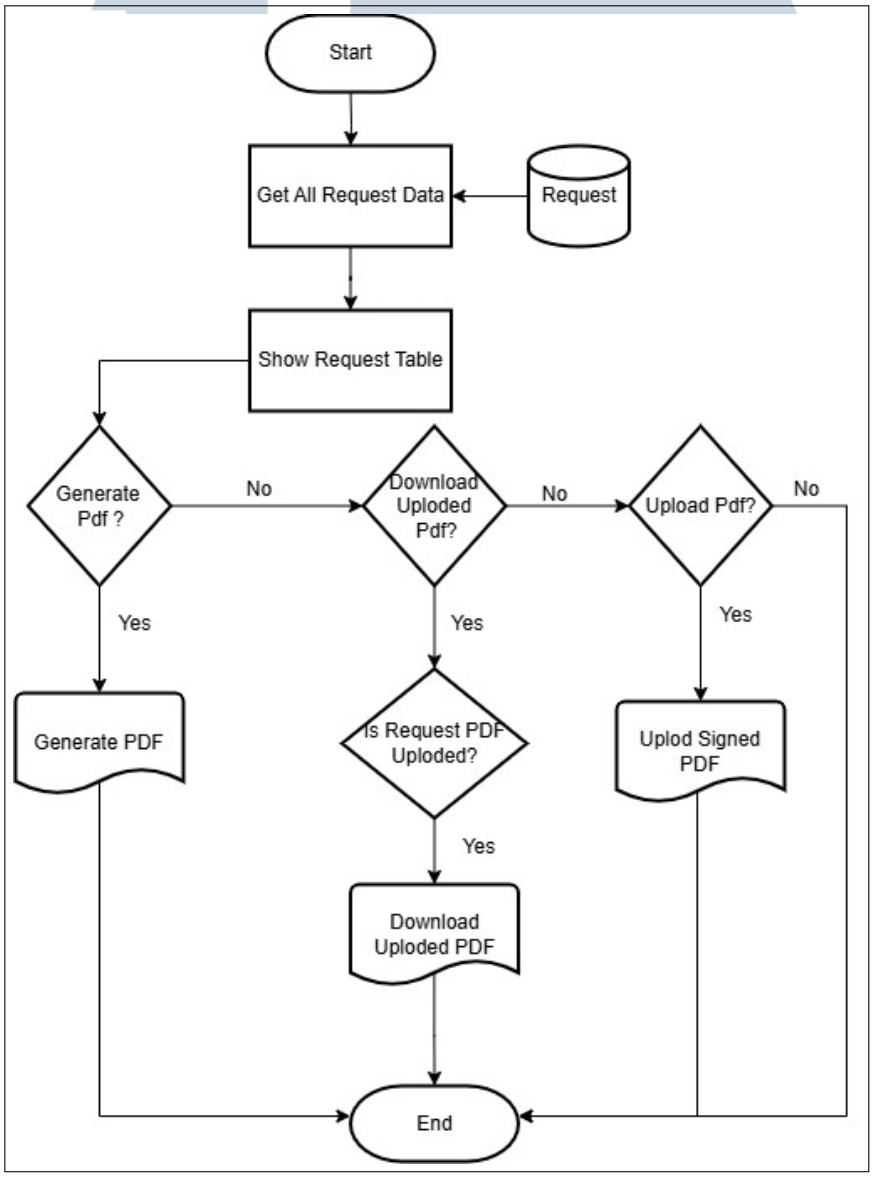
Gambar 3.5 menunjukkan alur kerja halaman *Request Management* yang berfungsi untuk mengelola seluruh data permintaan barang. Pada tahap awal, sistem mengambil data dari tabel *Request* dan menampilkannya dalam bentuk tabel pada halaman tersebut. Terdapat tiga fitur utama dalam halaman ini, yaitu *Download Generated PDF*, *Upload PDF*, dan *Download Uploaded PDF*.

Fitur *Download Generated PDF* digunakan untuk mengunduh dokumen *Request* dalam format PDF yang secara otomatis dibuat oleh sistem. Dokumen ini kemudian ditandatangani oleh pihak pemohon secara manual.

Selanjutnya, *fitur Upload PDF* memungkinkan pengguna mengunggah

kembali dokumen yang telah ditandatangani. Setelah berhasil diunggah dan disimpan, dokumen akan tampil dalam daftar *Request*.

Fitur terakhir adalah *Download Uploaded PDF*, yang memungkinkan pengguna mengunduh kembali dokumen yang telah ditandatangani. Namun, fitur ini hanya berfungsi jika dokumen yang di tanda tangani sudah diunggah. Jika belum ada unggahan, maka fitur tidak dapat digunakan



Gambar 3.5. Flowchart halaman Request Management

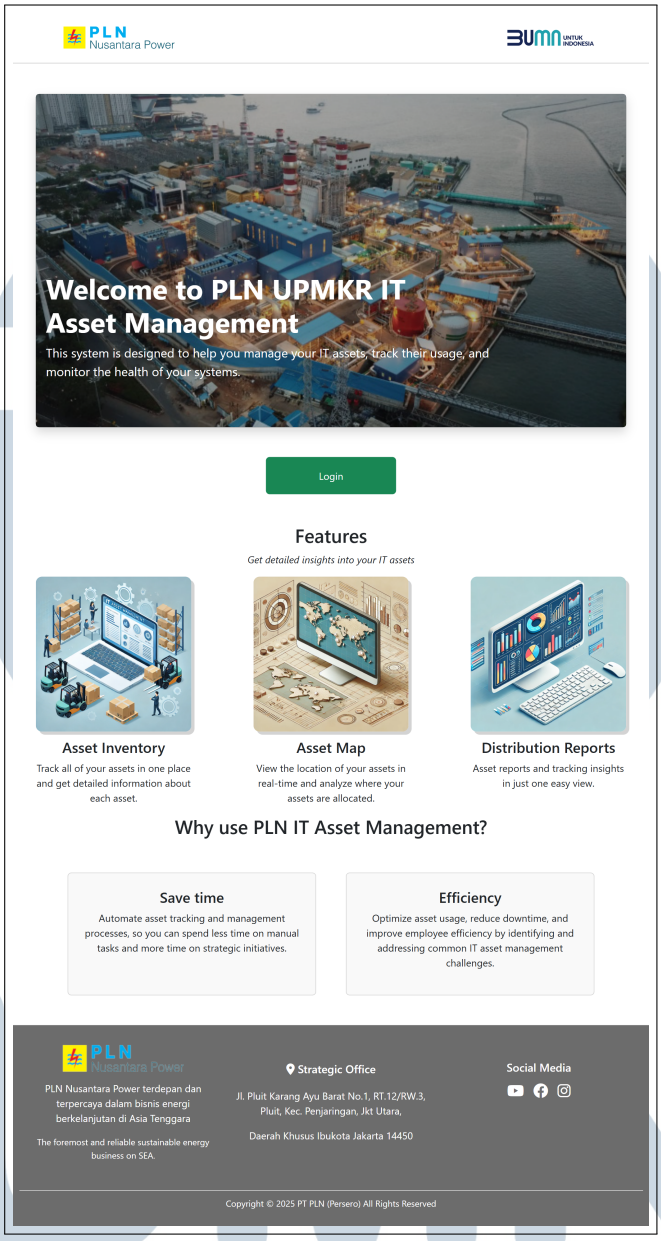
3.4.2 Implementasi Website

Tampilan hasil implementasi *website* beserta cuplikan kode program yang mendukung fungsionalitasnya disajikan dalam bagian ini. Setiap tampilan yang ditampilkan akan dijelaskan secara singkat fungsinya, dan beberapa potongan kode yang relevan akan ditampilkan untuk menggambarkan proses kerja di balik antarmuka tersebut.

A Tampilan Halaman Landing page

Gambar 3.6 menunjukkan tampilan halaman *landing page* dari website yang dikembangkan. Halaman ini berfungsi sebagai pengenalan awal terhadap sistem, yang menampilkan informasi mengenai fitur-fitur utama serta manfaat dari penggunaan website. Pada bagian tengah halaman, terdapat elemen *card* yang menjelaskan secara ringkas kegunaan sistem bagi pengguna.

Selain itu, terdapat tombol *Login* yang dapat digunakan oleh pengguna untuk masuk ke dalam sistem. Ketika tombol tersebut ditekan, sistem akan menampilkan *modal login* berupa *form* autentikasi. Jika proses *login* berhasil, maka pengguna akan diarahkan secara otomatis menuju halaman *Dashboard* sebagai halaman utama sistem setelah autentikasi.



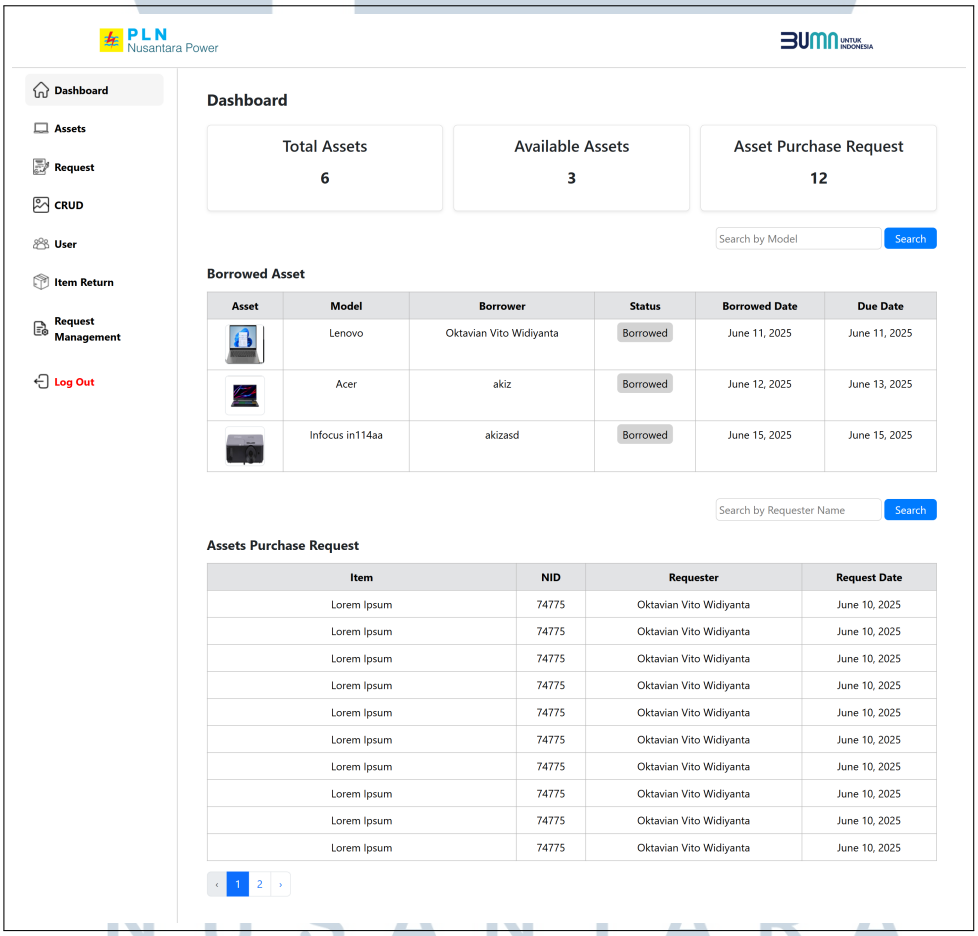
Gambar 3.6. Tampilan halaman Landing Page

B Tampilan Halaman Dashboard

Gambar 3.7 menunjukkan tampilan halaman *Dashboard* yang berfungsi sebagai pusat informasi utama dalam sistem. Pada bagian atas halaman, terdapat tiga buah *card* yang masing-masing menampilkan informasi mengenai *total aset* yang tercatat dalam *database*, jumlah aset yang masih tersedia (belum dipinjam), serta jumlah permintaan barang yang masuk.

Di bawah ketiga *card* tersebut, terdapat tabel *Borrowed Asset* yang menyajikan informasi detail mengenai aset-aset yang sedang dipinjam. Tabel ini dilengkapi dengan fitur *search bar* yang memungkinkan pengguna untuk mencari aset yang sedang dipinjam berdasarkan nama model. Apabila jumlah data melebihi 10 baris, sistem akan menampilkan fitur *pagination* untuk memudahkan navigasi antar tabel.

Selanjutnya, pada bagian bawah halaman, terdapat tabel *Assets Purchase Request* yang digunakan untuk menampilkan data permintaan pembelian aset secara lengkap. Data pada tabel ini diurutkan berdasarkan tanggal permintaan terbaru. Selain itu, pengguna juga dapat memfilter isi tabel berdasarkan *requester name*. Sama seperti tabel sebelumnya, apabila jumlah data melebihi 10 baris, sistem secara otomatis akan menampilkan fitur *pagination*.



Gambar 3.7. Tampilan halaman Dashboard

Potongan kode yang digunakan untuk menjalankan *query request list* pada

halaman *dashboard* ditunjukkan pada Kode 3.1.

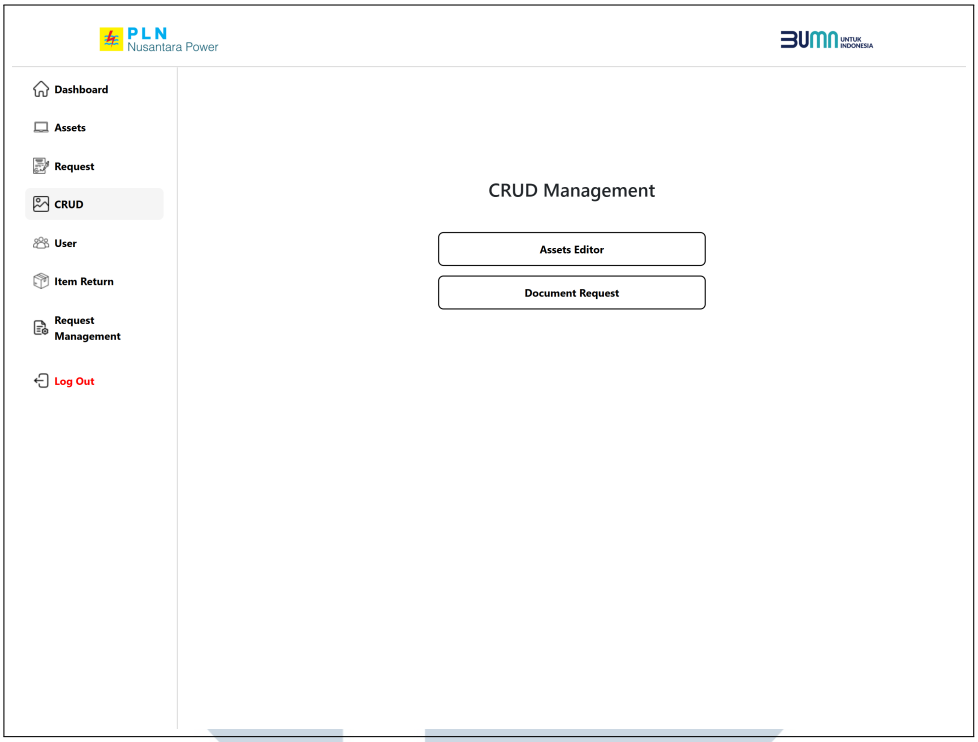
```
$purchaseRequestList = \App\Models\Request::select('RequestID', '
    Sender', 'NID', 'RequestDate', 'RequestDesc')
    ->when($searchRequester, function ($query) use (
    $searchRequester) {
        return $query->where('Sender', 'like', "%$searchRequester%
    ");
    })
    ->orderBy('RequestDate', 'desc')
    ->paginate(10, ['*'], 'request_page');
```

Kode 3.1: Kode query request list

Kode 3.1 merupakan potongan kode yang berfungsi untuk mengambil data permintaan pembelian aset dari tabel *request* dengan memilih kolom-kolom tertentu, antara lain *RequestID*, *Sender*, NID, *RequestDate*, dan *RequestDesc*. Apabila terdapat input pencarian berupa nama pengirim (disimpan dalam variabel *searchRequester*), maka sistem akan melakukan pemfilteran data berdasarkan kolom *Sender* yang mengandung kata kunci tersebut. Selanjutnya, data hasil *query* akan diurutkan berdasarkan tanggal permintaan (*RequestDate*) secara menurun, sehingga permintaan terbaru akan muncul terlebih dahulu. Data yang ditampilkan dibatasi sebanyak 10 baris per halaman, dan proses paginasi ini menggunakan nama halaman khusus, yaitu *request_page*.

C Tampilan Halaman Assets Editor

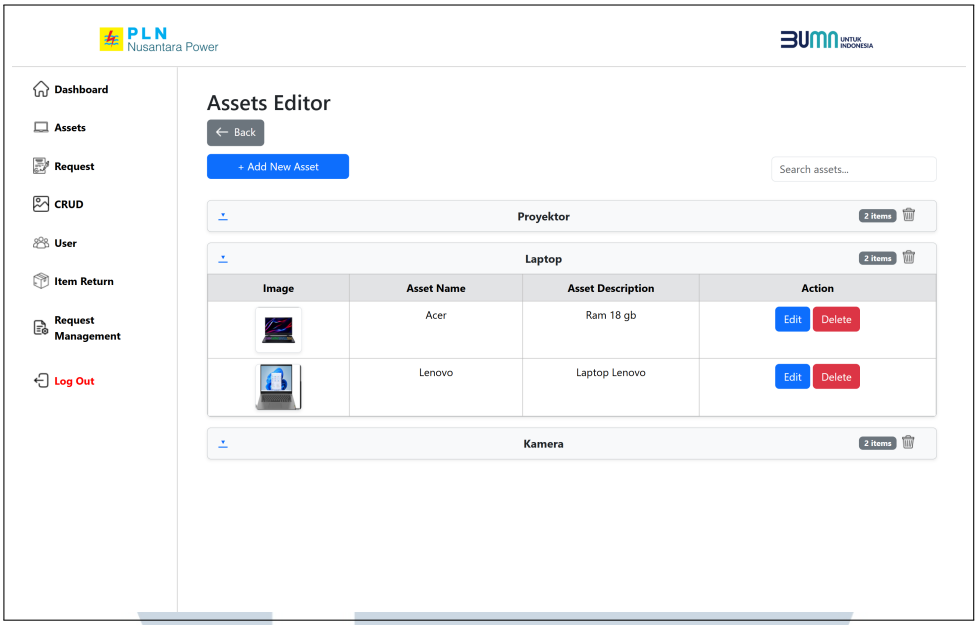
Gambar 3.8 merupakan tampilan halaman *CRUD*. Halaman *Assets Editor* bisa di akses melalui sidebar yang bertulis *CRUD* dan menekan tombol *Assets Editor*.



Gambar 3.8. Tampilan halaman CRUD

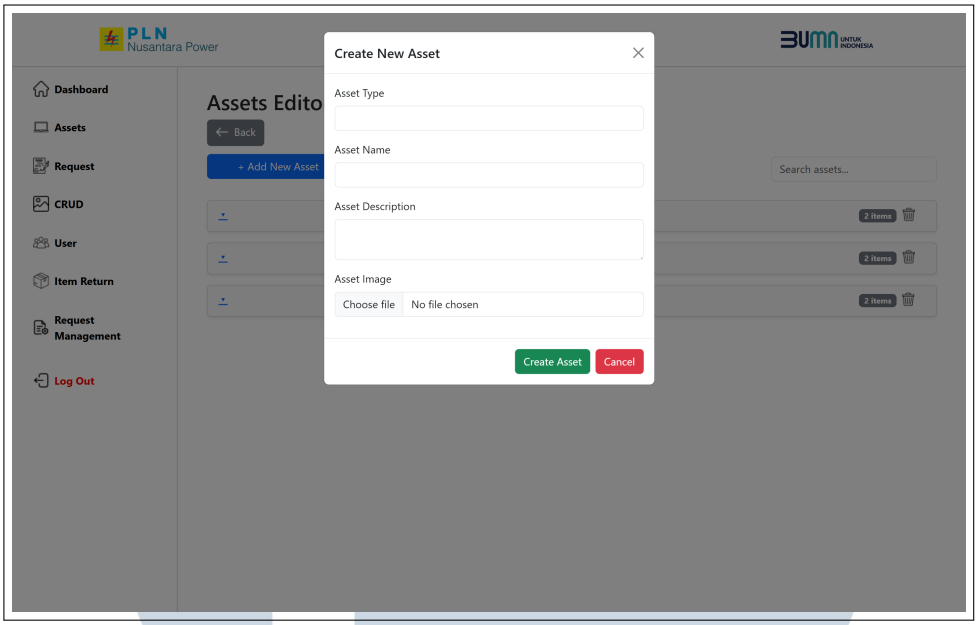
Halaman yang ditampilkan pada Gambar 3.9 merupakan tampilan dari halaman *Assets Editor*, yang berfungsi untuk mengelola data aset berdasarkan kategori atau tipe aset seperti Laptop, Proyektor, dan Kamera. Halaman ini menyediakan berbagai fitur yang dirancang untuk memudahkan pengguna dalam melakukan pengaturan data aset.

Setiap aset yang ditampilkan pada halaman ini dikelompokkan berdasarkan tipenya. Sebagai contoh, seluruh aset dengan tipe Laptop akan ditampilkan dalam satu kelompok tersendiri. Pada setiap kelompok tipe aset, terdapat ikon tong sampah di bagian kanan atas yang berfungsi untuk menghapus seluruh aset yang termasuk dalam tipe tersebut secara sekaligus. Selain itu, juga terdapat sebuah *badge* yang menunjukkan total jumlah aset berdasarkan tipenya, sehingga pengguna dapat dengan mudah mengetahui jumlah aset dalam setiap kategori.



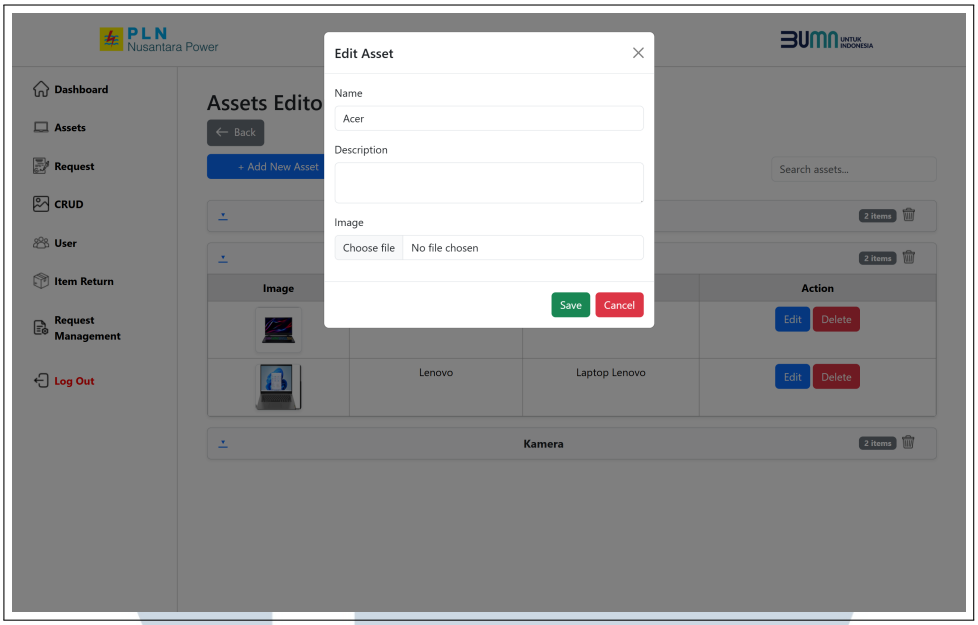
Gambar 3.9. Tampilan halaman Assets Editor

Gambar 3.10 menunjukkan tampilan modal *Create New Asset* yang muncul ketika pengguna menekan tombol *Add New Asset* yang terletak di bagian kiri atas halaman. Tombol ini berfungsi untuk menambahkan data aset baru ke dalam sistem. Dalam proses ini, pengguna diwajibkan untuk mengisi informasi penting seperti tipe aset, nama aset, dan deskripsi barang. Sementara untuk bagian gambar aset, pengguna diberi opsi untuk mengisi atau mengosongkannya sesuai kebutuhan.



Gambar 3.10. Tampilan modal Create New Asset

Untuk melakukan pengubahan data aset, pengguna dapat memilih tombol *Edit* yang akan menampilkan modal *Edit Asset*, seperti yang ditunjukan pada Gambar 3.11. Melalui modal ini, pengguna dapat mengubah gambar, nama, dan deskripsi aset. Sementara itu, tombol *Delete* digunakan untuk menghapus aset tertentu dari sistem secara permanen.



Gambar 3.11. Tampilan modal Edit Asset

Fungsi untuk menyimpan data aset ke dalam *database* ditunjukkan pada Kode 3.2.

```
public function store(Request $request)
{
    // Validasi input dari form
    $request->validate([
        'AssetName' => 'required|string|max:100',
        'AssetType' => 'required|string|max:100',
        'AssetImage' => 'nullable|image|max:280000000',
        'AssetDesc' => 'string',
    ]);

    // Susun data dari request untuk disimpan ke database
    $data = [
        'AssetName' => $request->AssetName,
        'AssetType' => ucwords(strtolower($request->AssetType)),
    // Kapital awal kata
        'AssetDesc' => $request->AssetDesc,
    ];

    if ($request->hasFile('AssetImage')) {
        // Simpan file ke dalam storage, simpan path-nya
        $path = $request->file('AssetImage')->store('assets', '
    public');
```

```
21          $data['AssetImage'] = $path;
22      }
23
24      DB::table('asset')->insert($data);
25      return redirect()->route('crud.assets_editor')->with('success'
        , 'Asset created successfully');
26  }
```

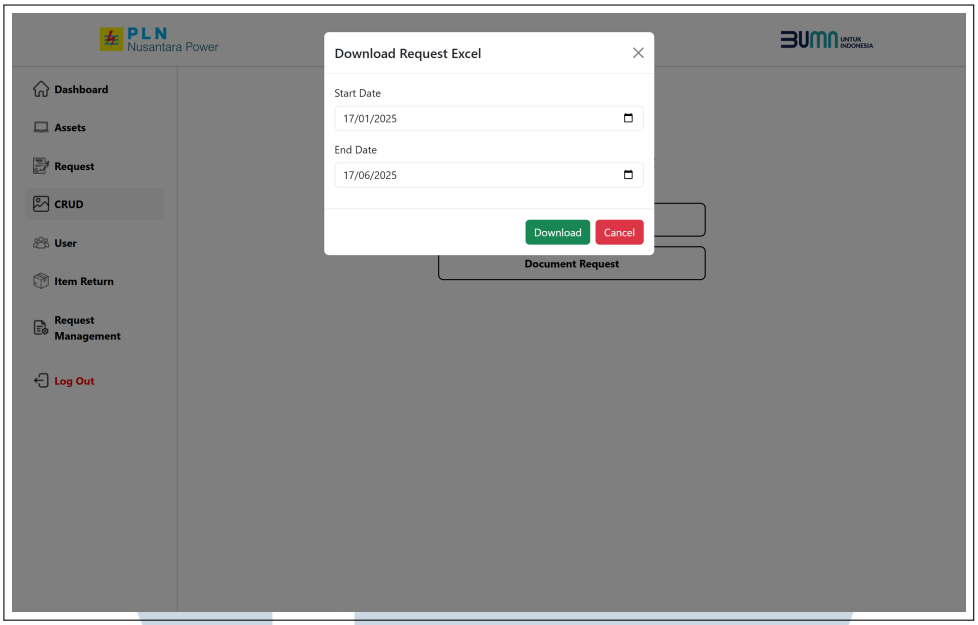
Kode 3.2: Contoh fungsi store untuk menyimpan data aset

Kode 3.2 menunjukkan potongan kode fungsi *store* pada *controller Laravel* yang menyimpan data aset baru ke *database*. Fungsi ini menerima objek *Request* berisi data dari *form* pengguna. Langkah pertama adalah validasi data. *Field AssetName, AssetType,* dan *AssetDesc* wajib diisi sebagai *string* dengan panjang maksimal 100 karakter, kecuali *AssetDesc* tanpa batasan. *Field AssetImage* bersifat opsional dan jika diisi harus berupa gambar dengan ukuran maksimal 280 MB.

Setelah melakukan validasi, data akan disusun dalam bentuk array $data. *AssetName* dan *AssetDesc* diambil langsung, sedangkan *AssetType* akan diubah dulu menjadi huruf kecil dan setiap kata dikapitalisasi agar konsisten pada penamaan tipe aset. Jika ada gambar diunggah, file disimpan di direktori *assets* pada penyimpanan publik, dan path-nya dimasukkan ke *field AssetImage* dalam $data. Terakhir, data dimasukkan ke tabel *asset* menggunakan metode *insert()*. Jika berhasil, pengguna diarahkan ke halaman *assets_editor* dengan pesan sukses.

D Tampilan Modal Document Request

Untuk mengakses *modal Document Request*, pengguna harus menekan tombol *Document Request* seperti yang terlihat pada Gambar 3.8. *Modal* ini memiliki fungsi untuk mengunduh file *Excel* yang berisi data dari tabel *request*. Sebelum proses pengunduhan dilakukan, pengguna perlu menentukan rentang waktu dengan memilih tanggal awal dan tanggal akhir. Setelah itu, pengguna dapat menekan tombol *Download* untuk memulai proses. Tampilan dari *modal Document Request* yang muncul setelah tombol ditekan ditunjukkan pada Gambar 3.12.



Gambar 3.12. Tampilan modal Document Request

Fungsi untuk mengunduh data permintaan dalam format Excel ditunjukkan pada Kode 3.3.

```
public function downloadExcel(Request $request)
{
    $start = $request->start_date;
    $end = $request->end_date;

    return Excel::download(new RequestsExport($start, $end), 'requests.xlsx');
}
```

Kode 3.3: Fungsi untuk mengunduh data permintaan dalam format Excel

Kode 3.3 merupakan fungsi *downloadExcel* dalam *controller Laravel* yang digunakan untuk mengunduh data *request* dalam format file Excel. Fungsi ini menerima input dari pengguna melalui objek *Request*, khususnya dua parameter: *start_date* dan *end_date*, yang menunjukkan rentang tanggal untuk data yang ingin diekspor. Nilai dari kedua parameter ini kemudian akan digunakan untuk membuat *instance* dari kelas *RequestsExport*, yang bertugas untuk mengambil data *request* berdasarkan rentang tanggal yang sudah dipilih. Setelah itu, fungsi ini memanggil metode *Excel::download*, yang merupakan bagian dari *package Laravel Excel*, untuk menghasilkan dan mengunduh file Excel dengan nama requests.xlsx.

Kode 3.4 menunjukkan kelas *RequestsExport* untuk mengekspor data permintaan ke Excel.

```
namespace App\Exports;

use App\Models\Request;
use Maatwebsite\Excel\Concerns\FromCollection;
use Maatwebsite\Excel\Concerns\WithHeadings;

class RequestsExport implements FromCollection, WithHeadings
{
    protected $start, $end;

    public function __construct($start, $end)
    {
        $this->start = $start;
        $this->end = $end;
    }

    public function collection()
    {
        return Request::whereBetween('RequestDate', [$this->start, $this->end])
            ->get(['RequestID', 'NID', 'Sender', 'Receiver', 'Field', 'RequestDate', 'RequestDesc']);
    }

    public function headings(): array
    {
        return ['RequestID', 'NID', 'Sender', 'Receiver', 'Field', 'RequestDate', 'RequestDesc'];
    }
}
```

Kode 3.4: Kelas RequestsExport untuk export data permintaan ke Excel

Kode 3.4 merupakan kelas dari *RequestsExport* yang digunakan untuk mengekspor data dari tabel *request* ke dalam format file *Excel* dengan memanfaatkan *package Laravel Excel* (*Maatwebsite*). Kelas ini berada dalam namespace *App\Exports* dan mengimplementasikan dua hal yaitu *FromCollection* dan *WithHeadings*.

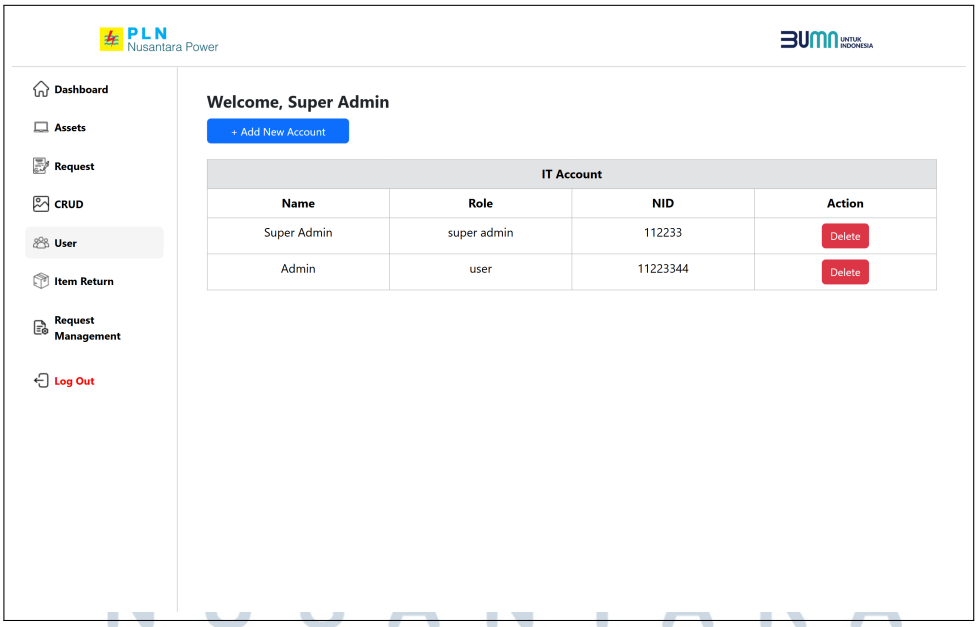
Implementasi *FromCollection* menunjukkan bahwa sumber data yang akan diekspor berasal dari *query collection*, sementara *WithHeadings* digunakan untuk menambahkan baris *header* pada file *Excel* yang akan dihasilkan. Kelas ini memiliki dua properti bersifat *protected*, yaitu *$start* dan *$end*, yang berfungsi sebagai batasan rentang tanggal data yang akan diekspor.

Metode *collection*() memiliki fungsi untuk mengambil data dari model *Request* berdasarkan filter tanggal pada kolom *RequestDate*, dengan menggunakan *whereBetween*(). Data yang diambil adalah kolom *RequestID*, NID, *Sender*, *Receiver*, *Field*, *RequestDate*, dan *RequestDesc*. Sementara itu, metode *headings*() mengembalikan *array* yang berisi nama-nama kolom sebagai baris pertama dalam file *Excel*.

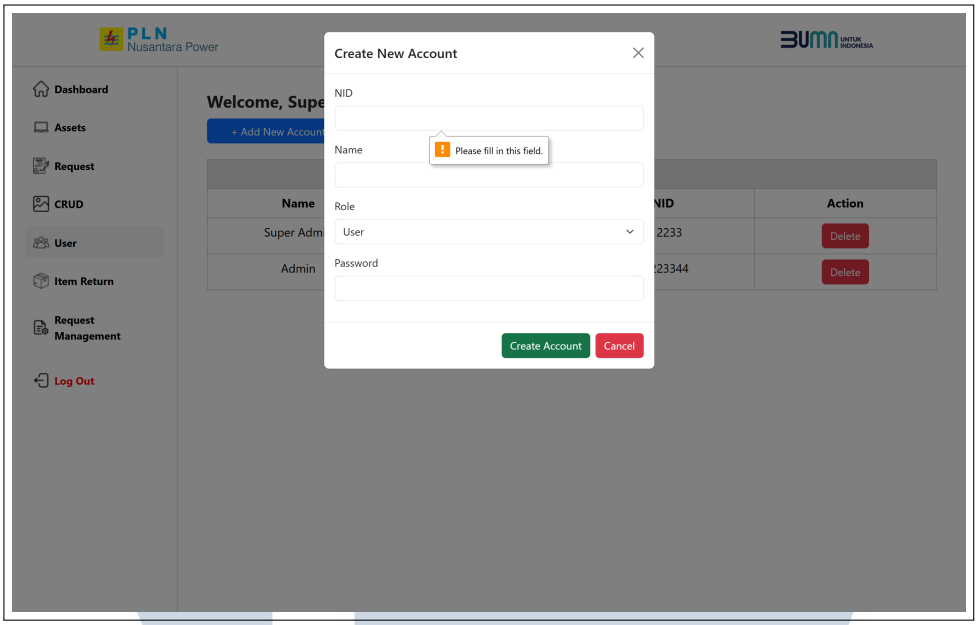
E Tampilan Halaman User Management

Pada halaman *User Management* terdapat dua tampilan berbeda yaitu tampilan *Super Admin* dan *Admin*, dua tampilan ini ditentukan berdasarkan *role* yang dimiliki akun yang sedang *login*. Gambar 3.13 merupakan tampilan yang dilihat oleh *user* dengan *role Super admin*, pada tampilan *super admin* terdapat tombol *Add New Account* yang berfungsi untuk membuat akun baru, jika tombol tersebut di tekan maka akan muncul *modal Create New Account* seperti pada Gambar 3.14, pengguna akan diwajibkan untuk memasukan beberapa informasi ke akun yang ingin dibuat, informasi tersebut termasuk NID, Nama, *role*, dan *password*. Setelah mengisi informasi tersebut maka pengguna bisa menekan tombol *create account* untuk membuat akun baru.

Setelah akun dibuat atau jika sudah terdapat akun yang terdaftar, sistem akan menampilkan tabel dengan judul *IT Account* yang memuat daftar seluruh akun dalam sistem. Tabel ini terdiri dari kolom *Name*, *Role*, NID, dan *Action*. Pada kolom *Action*, tersedia tombol *Delete* berwarna merah yang memungkinkan *super admin* untuk menghapus akun tertentu. Saat tombol tersebut ditekan, sistem akan menampilkan modal konfirmasi penghapusan untuk memastikan tindakan yang diambil oleh pengguna.

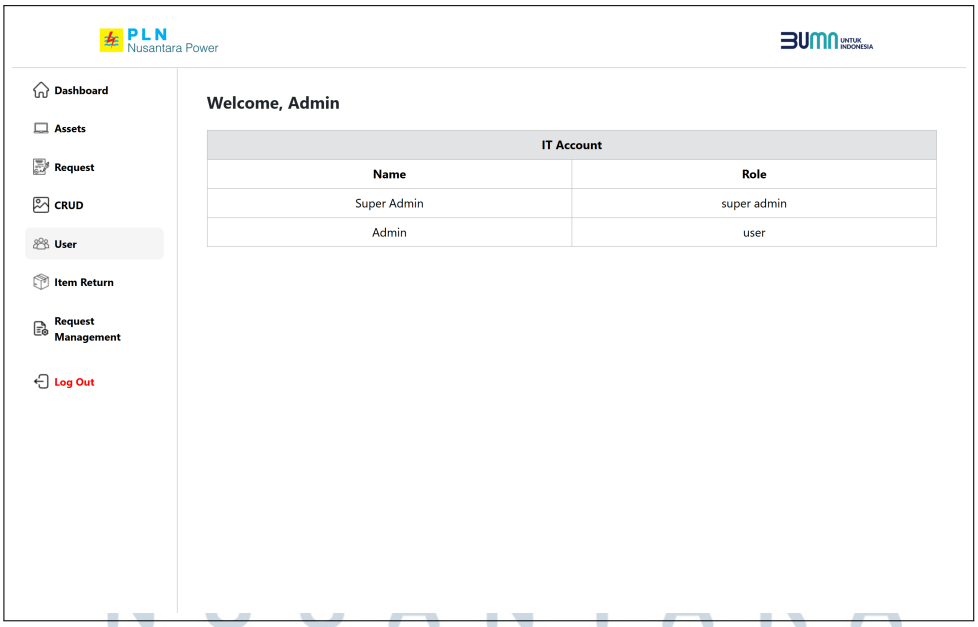


Gambar 3.13. Tampilan halaman User Management dengan role ”super admin”



Gambar 3.14. Modal Create New Account

Gambar 3.15 merupakan tampilan jika pengguna yang sedang *login* memiliki *role "User"*. pengguna yang memiliki *role "User"* hanya akan bisa melihat tabel yang berisi Nama dan *Role* saja.



Gambar 3.15. Tampilan halaman User Dengan Role "User"

Fungsi yang digunakan untuk memproses pembuatan akun baru, termasuk

validasi data dan penyimpanan ke *database*, dapat dilihat pada Kode 3.5.

```
public function createAccount(Request $request)
{
    // Validasi form
    $validated = $request->validate([
        'NID' => 'required|unique:user,NID',
        'Name' => 'required',
        'Role' => 'required',
        'Password' => 'required|min:6',
    ]);

    // Simpan data user ke database
    User::create([
        'NID' => $validated['NID'],
        'Name' => $validated['Name'],
        'Role' => $validated['Role'],
        'Password' => Hash::make($validated['Password']), // Password di-hash
    ]);

    return redirect()->route('user')->with('success', 'Account created successfully!');
}
```

Kode 3.5: Fungsi untuk membuat akun baru

Kode 3.5 merupakan fungsi dari *createAccount* yang digunakan untuk menambahkan akun pengguna baru ke dalam sistem. Fungsi ini diawali dengan proses validasi data yang dikirim melalui *form* menggunakan metode `$request->validate()`. Validasi ini bertujuan untuk memastikan bahwa kolom NID wajib diisi dan bersifat unik pada tabel *users*, sedangkan kolom Name dan *Role* juga bersifat wajib. Selain itu, kolom *Password* harus diisi dengan panjang minimal enam karakter.

Data yang sudah divalidasi kemudian disimpan ke dalam tabel *user* melalui pemanggilan model *User*. Sebelum proses penyimpanan, *Password* akan di-hash terlebih dahulu menggunakan fungsi *Hash::make*() untuk memastikan keamanan data pengguna di dalam *database*. Setelah proses penyimpanan selesai, pengguna akan diarahkan kembali ke halaman dengan *route* bernama *user*, serta akan menerima notifikasi yang menyatakan bahwa akun telah berhasil dibuat.

Fungsi yang digunakan untuk menghapus data *user*, termasuk pengecekan *user* yang dihapus adalah *user* yang sedang *login*, ditunjukkan pada Kode 3.6.

```php
public function destroy($nid)
{
    $user = User::findOrFail($nid);

    // Ambil user yang login dari session
    $currentUser = Session::get('user');

    // Cek user yang dihapus adalah user yang sedang login
    if ($currentUser && $currentUser->NID == $user->NID) {
        Session::forget('user'); // Logout manual
        $user->delete();         // Hapus user

        return redirect()->route('login.page')->with('success', 'Your account has been deleted and you have been logged out.');
    }

    // Jika bukan akun sendiri, hapus biasa
    $user->delete();

    return redirect()->route('user')->with('success', 'User deleted successfully.');
}
```

Kode 3.6: Fungsi untuk menghapus user

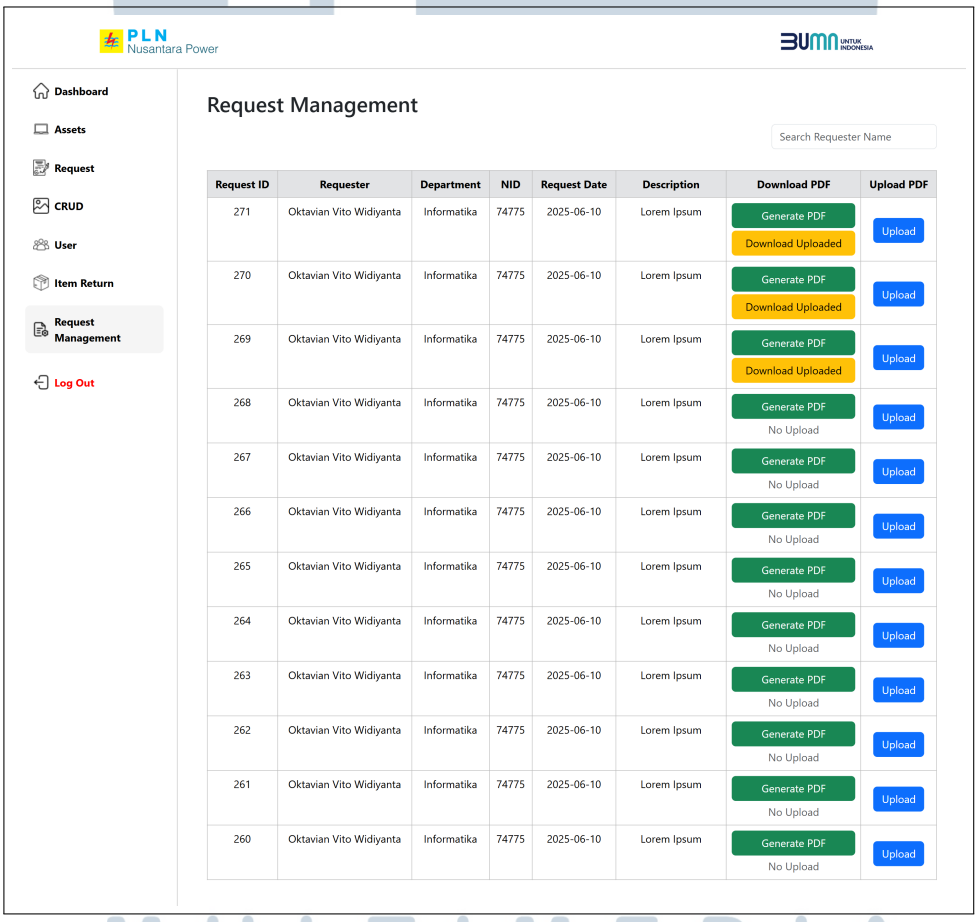
Kode 3.6 merupakan fungsi *destroy* yang menerima parameter $nid yang ingin dihapus. Pertama, sistem akan mencari *user* tersebut di *database* menggunakan *User::findOrFail($nid)* Setelah *user* ditemukan, fungsi ini akan mengambil data *user* yang sedang login dari sesi dengan *Session::get('user')*. Kemudian, dilakukan pengecekan apakah *user* yang sedang *login* adalah *user* yang ingin dihapus. Jika benar, maka sistem akan menghapus sesi *login* menggunakan *Session::forget('user')* untuk melakukan *logout*, lalu menghapus data *user* dari *database*. Setelah itu, pengguna diarahkan ke halaman *login* dengan notifikasi bahwa akunnya telah dihapus dan sudah *logout*. Jika *user* yang dihapus bukan *user* yang sedang *login*, maka data *user* tetap dihapus, tetapi tanpa menghapus sesi *login*, dan pengguna akan diarahkan ke halaman daftar user *(route('user'))* dengan pesan bahwa *user* berhasil dihapus.

F Tampilan Halaman Request Management

Gambar 3.16 merupakan tampilan Halaman *Request Management* yang berfungsi untuk mengelola dan memantau seluruh *request* dari pengguna sistem. Informasi dalam halaman ini ditampilkan dalam bentuk tabel yang memuat kolom seperti *Request ID*, *Requester*, *Department*, NID, *Request Date*, dan *Description*. Di bagian kanan atas terdapat *search bar* yang memungkinkan pengguna mencari data berdasarkan nama *requester*, sehingga mempermudah proses pencarian data permintaan tertentu.

Setiap baris dalam tabel mewakili satu permintaan, dan disediakan beberapa

fitur utama. Pertama, terdapat tombol ”*Generate PDF*” berwarna hijau yang berfungsi untuk membuat file *PDF* berdasarkan data permintaan, yang nantinya bisa dicetak lalu ditandatangani. Setelah file tersebut ditandatangani, pengguna dapat mengunggahnya kembali melalui tombol ”*Upload*” berwarna biru yang tersedia di kolom ”*Upload PDF*”. Setelah proses unggah berhasil dilakukan, akan muncul tombol tambahan yaitu ”*Download Uploaded*” berwarna kuning yang memungkinkan pengguna mengunduh kembali dokumen *PDF* yang telah ditandatangani. Namun, jika belum ada dokumen yang diunggah, maka di bagian tersebut akan muncul tulisan ”*No Upload*” sebagai penanda bahwa file belum tersedia.



Gambar 3.16. Tampilan halaman Request Management

Fungsi yang digunakan untuk menghasilkan file *PDF* berdasarkan ID permintaan dapat dilihat pada Kode 3.7.

```
1 public function generatePdf($id)
2 {
```

```
    // Ambil data request berdasarkan ID
    $request = RequestModel::findOrFail($id);

    // Buat file PDF dari view 'request_pdf' dengan data request
    yang sudah diambil
    $pdf = Pdf::loadView('request_pdf', compact('request'));

    // Unduh dengan nama file sesuai ID request
    return $pdf->download('request_' . $request->RequestID . '.pdf
    ');
}
```

Kode 3.7: Fungsi untuk generate PDF berdasarkan ID request

Kode 3.7 merupakan fungsi dari *generatePdf* yang berguna untuk menghasilkan dan mengunduh file *PDF* dari data *request* berdasarkan ID-nya. Pertama, fungsi ini mengambil data dari model *RequestModel* dengan metode *findOrFail($id)*, yang akan mencari data berdasarkan ID yang diberikan. Setelah data berhasil ditemukan, fungsi ini memuat tampilan bernama *request_pdf* dan menyisipkan data *request* ke dalamnya menggunakan fungsi *Pdf::loadView*(). Tampilan tersebut kemudian diubah menjadi file *PDF*. File PDF tersebut dikembalikan ke pengguna untuk diunduh dengan nama file yang menggunakan format *request_ID*.pdf, di mana ID adalah nilai dari *RequestID* pada data yang diambil.

Fungsi yang digunakan untuk mengunggah file *PDF* dan menyimpan *path* ke dalam *database* ditunjukkan pada Kode 3.8.

```
public function uploadPdf(Request $request, $id)
{
    // Validasi file berformat PDF, dan maksimal 2MB
    $request->validate([
        'pdf_file' => 'required|mimes:pdf|max:2048',
    ]);

    // Ambil file dari form input
    $file = $request->file('pdf_file');

    // Buat nama file unik menggunakan ID dan timestamp
    $fileName = 'uploaded_' . $id . '_' . time() . '.' . $file->
    getClientOriginalExtension();

    // Simpan ke 'public/request_pdfs'
    $path = $file->storeAs('request_pdfs', $fileName, 'public');
```

```
16
17      // Temukan data request berdasarkan ID
18      $requestModel = RequestModel::findOrFail($id);
19
20      // Simpan path file ke kolom 'uploaded_pdf' di database
21      $requestModel->uploaded_pdf = $path;
22      $requestModel->save();
23
24      return back()->with('success', 'PDF uploaded successfully!');
25  }
```

Kode 3.8: Fungsi untuk upload file PDF dan simpan path-nya

Kode 3.8 merupakan fungsi *uploadPdf* yang bertugas untuk mengunggah file *PDF* dan menyimpannya ke dalam sistem, serta mencatat lokasi file tersebut di *database*. Fungsi ini menerima dua parameter, yaitu objek *Request* dan ID dari *request* yang berkaitan. Pertama, fungsi ini melakukan validasi terhadap input file, memastikan bahwa file yang diunggah wajib berformat *PDF (mimes:pdf)*, dan tidak melebihi ukuran 2MB.

Kemudian file *PDF* diambil dari *input form* dan dibuatkan nama unik dengan format yang mencakup ID dan *timestamp* agar tidak terjadi duplikasi nama. File kemudian disimpan ke direktori *public/request_pdfs* menggunakan method *storeAs*, yang juga mengembalikan *path* file tersebut.

Selanjutnya, fungsi mencari data *RequestModel* berdasarkan ID. Setelah data ditemukan, path dari file yang diunggah disimpan ke kolom *uploaded_pdf* dalam tabel yang sesuai di database, lalu data tersebut disimpan kembali dengan *method save*(). Terakhir, fungsi menampilkan pesan sukses bahwa file *PDF* berhasil diunggah.

Fungsi yang digunakan untuk mengunduh file *PDF* yang telah diunggah sebelumnya ditunjukkan pada Kode 3.9.

```
1  public function downloadUploadedPdf($id)
2  {
3      // Ambil data request berdasarkan ID
4      $request = RequestModel::findOrFail($id);
5
6      // Ambil path lengkap file di storage
7      $filePath = storage_path('app/public/' . $request->
       uploaded_pdf);
8
9      // Cek file benar ada di storage atau tidak
10     if (!file_exists($filePath)) {
```

```
11          return back()->with('error', 'File not found at: ' .
    $filePath);
12      }
13
14      return response()->download($filePath);
15  }
```

Kode 3.9: Fungsi untuk download file PDF yang sudah diupload

Kode 3.9 merupakan fungsi *downloadUploadedPdf* yang digunakan untuk mengunduh file *PDF* yang sebelumnya telah diunggah dan disimpan pada sistem. Fungsi ini menerima parameter $id yang digunakan untuk mencari data request terkait melalui *RequestModel::findOrFail($id)*.

Setelah data berhasil ditemukan, fungsi mengambil *path* lengkap dari file *PDF* yang tersimpan di *storage* lokal, tepatnya pada direktori *storage/app/public/*, dengan menambahkan nama file dari kolom *uploaded_pdf*. Sebelum mengunduh file, fungsi ini melakukan pengecekan apakah file benar-benar ada di lokasi tersebut menggunakan *file_exists()*. Jika file tidak ditemukan, maka pengguna akan diarahkan kembali ke halaman sebelumnya dengan pesan *error* yang menyatakan bahwa file tidak ditemukan.

Jika file ada, maka fungsi akan mengembalikannya sebagai *response download* menggunakan `response()->download($filePath)`, yang secara otomatis memulai proses pengunduhan file ke perangkat pengguna.

3.5 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan magang ada beberapa kendala teknis yang memengaruhi jalannya pekerjaan.

3.5.1 Kendala

Kendala yang di alami selama kegiataan magang, yaitu:

1. Saat proses pengerjaan pernah terjadi error pada *Software* Xampp, error yang dialami adalah ERROR 2002 (HY000).
2. Saat ingin melakukan install *package* maatwebsite/excel terjadi error yang dikarena *php extension* yang belum di *enable*,

3.5.2 Solusi

Sebagai upaya mengatasi kendala yang muncul selama proses magang, diterapkan dua solusi berikut ini.

1. Menginstall ulang *Software* Xampp dengan versi yang paling baru.
2. Menyalakan *extension* yang diperlukan untuk menginstall *maatwebsite/excel* pada file php.ini.