BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Peran dalam Kerja Magang

Sebagai mahasiswa dalam program kerja praktik magang di **PT. Prodia Widyusaha**, mahasiswa berada di bawah naungan **Departemen Transformasi Digital & IT**. Departemen ini bertanggung jawab secara strategis dalam melakukan transformasi digital terhadap berbagai proses internal perusahaan, khususnya dalam pengembangan sistem informasi yang mendukung layanan kesehatan dan laboratorium klinik.

Selama masa magang, mahasiswa berperan sebagai seorang *Fullstack Developer* yang bertugas merancang, membangun, dan mengembangkan aplikasi berbasis web yang digunakan secara internal. Proyek yang dikerjakan merupakan sistem internal perusahaan yang bertujuan untuk mempermudah pengelolaan *file lampiran hasil pemeriksaan laboratorium pasien*, yang sebelumnya masih ditangani secara manual dan tersebar dalam berbagai folder lokal di server FTP.

Sistem yang dibangun oleh mahasiswa bersifat *standalone* dan belum terintegrasi langsung dengan sistem manajemen informasi laboratorium utama perusahaan (*Laboratory Information System* / LIS). Tujuan utama dari aplikasi ini adalah untuk memfasilitasi proses pengumpulan, penamaan otomatis, unggah file ke FTP, dan pencatatan metadata file ke dalam database lokal menggunakan *MySQL*. Selain itu, sistem juga dilengkapi dengan proses autentikasi berbasis *LDAP Active Directory* dan fitur pencarian data pasien melalui koneksi *SOAP Web Service* ke database Oracle.

Mahasiswa mengembangkan seluruh komponen sistem secara end-to-end, mulai dari desain antarmuka menggunakan *Bootstrap*, implementasi logika bisnis di sisi server dengan menggunakan *CodeIgniter 3*, hingga konfigurasi sistem unggah ke FTP serta validasi data berdasarkan struktur dan aturan yang telah ditetapkan. Mahasiswa juga mengimplementasikan berbagai fitur tambahan, seperti penamaan file otomatis, perlindungan struktur file, validasi tipe file, serta metode kontrol akses berbasis sesi login pengguna cabang.

Peran ini memberikan mahasiswa kesempatan untuk menerapkan kemampuan teknis di bidang pengembangan web serta memahami secara langsung bagaimana proses digitalisasi dijalankan di lingkungan kerja profesional.

Selain itu, mahasiswa juga terlibat dalam proses diskusi, evaluasi, dan revisi sistem bersama supervisor, yang menjadi bagian penting dalam siklus pengembangan perangkat lunak di perusahaan.

3.1.1 Pembimbing dan Struktur Supervisi

Selama kerja praktik, mahasiswa mendapatkan bimbingan langsung dari dua instruktur: **Mahesa Supriyono** sebagai asisten manajer pengembangan IT dan **WUnsel Arto Negoro** sebagai penanggung jawab pengawasan umum proyek. Kedua memiliki tanggung jawab penting sebagai pembimbing teknis dan evaluator hasil kerja. Komunikasi personal dan sesi evaluasi mingguan memungkinkan pembagian tugas, evaluasi progres, dan pemberian revisi secara langsung. Penulis memperoleh pemahaman yang lebih baik tentang proses kerja profesional di lingkungan industri teknologi kesehatan berkat keterlibatan aktif supervisor di setiap tahap pengembangan proyek.

3.1.2 Peran dan Tanggung Jawab Mahasiswa

Selama kerja praktik berlangsung, mahasiswa memiliki tanggung jawab sebagai pengembang utama sistem, yang meliputi beberapa aspek pengembangan *front-end*, *back-end*. Tugas-tugas utama yang dijalankan antara lain:

1. **Perancangan dan Implementasi UI/UX menggunakan Figma**
   Mendesain dan mengimplementasikan tampilan halaman Login, Dashboard, dan Upload. Termasuk di dalamnya pengaturan layout, navigasi sidebar, struktur halaman, serta memastikan *user experience* yang mudah dipahami dan digunakan.

2. **Pengembangan Fitur Unggah File dengan Automasi**
   Membangun modul unggah file yang terintegrasi dengan *progress bar*, *preview* file, serta fitur *auto rename*. Sistem ini juga dilengkapi dengan pengelompokan folder berdasarkan status file: ”Upload”, ”Complete”, dan ”Complete Pioneer”.

3. **Integrasi Sistem dengan Oracle melalui SOAP**
   Menghubungkan aplikasi dengan database Oracle perusahaan menggunakan *Web Service (SOAP)*. Ini mencakup pengambilan data hasil lab berdasarkan

REG_NO dan PATIENT_ID, serta penampilannya dalam bentuk tabel dan *modal pop-up*.

4. **Konversi Otomatis Gambar ke PDF**
   Implementasi fitur yang secara otomatis mengubah file JPG/PNG menjadi PDF untuk menjaga standar file *attachment* yang digunakan dalam sistem.

5. **Pengembangan Fitur Setting dan Tipe File**
   Membuat modul pengaturan yang memungkinkan pengelolaan jenis file yang dapat diunggah, lengkap dengan sistem *CRUD*, validasi input, dan integrasi ke *form* upload.

6. **Refactoring dan Perbaikan Kode**
   Melakukan perbaikan struktur kode agar lebih modular dan *scalable*, serta mengatasi berbagai *bug* seperti error saat *binding* data dari *backend* ke tampilan *modal*.

3.1.3 Alur Kerja dan Mekanisme Pengembangan

Pengembangan proyek dilakukan secara bertahap sesuai dengan alur kerja yang ditetapkan oleh manajer proyek. Mahasiswa melakukan pengembangan langsung pada struktur folder lokal yang telah disiapkan, tanpa menggunakan sistem kontrol versi seperti *Git*. Alur kerja yang diterapkan adalah sebagai berikut:

- **Distribusi Tugas**: Tugas diberikan secara langsung oleh supervisor, disertai dengan arahan teknis dan desain kasar (gambar sketsa) yang menggambarkan struktur alur dan tampilan sistem.
- **Pengerjaan dan Implementasi**: Tugas dikerjakan oleh mahasiswa secara mandiri sesuai dengan tenggat waktu (*deadline*) yang telah ditentukan. Penyesuaian terhadap kebutuhan sistem dilakukan melalui konsultasi secara berkala dengan supervisor.
- **Revisi dan Evaluasi**: Setiap fitur yang telah selesai diuji dan dikonsultasikan kembali. Supervisor memberikan evaluasi serta permintaan revisi bila terdapat kekurangan atau ketidaksesuaian dengan kebutuhan sistem.
- **Pengujian Sistem**: Pengujian dilakukan secara manual oleh mahasiswa dan supervisor dengan menjalankan sistem secara langsung dan memeriksa integritas data serta fungsionalitas fitur.

- **Finalisasi dan Integrasi**: Setelah dinyatakan valid dan stabil, fitur dianggap selesai dan menjadi bagian dari sistem utama. Setelah itu, pengembangan berlanjut ke modul atau fitur berikutnya.

3.1.4 Koordinasi dan Komunikasi Kerja

Proses koordinasi dilakukan dengan metode sederhana namun efektif, mengingat proyek bersifat internal dan tim pengembang terbatas:

- **Komunikasi Harian**: Menggunakan platform **Google Chat** untuk diskusi ringan, pelaporan progres, serta klarifikasi tugas teknis.
- **Diskusi Mingguan**: Dilakukan secara langsung dalam bentuk diskusi informal di kantor, di mana supervisor memberikan evaluasi hasil kerja mingguan dan mengarahkan rencana pengembangan selanjutnya.

Meskipun mahasiswa mengerjakan proyek ini secara individual, mahasiswa mendapatkan pengalaman yang sangat berarti untuk paham akan kerja profesional.

3.2 Tugas yang Dilakukan

Selama periode magang di **PT. Prodia Widyusaha**, mahasiswa diberi tugas secara langsung oleh supervisor dari Divisi *Digital Transformation & IT*. Tugas ini disesuaikan dengan peran mahasiswa sebagai *Fullstack Developer* dan mengacu pada kebutuhan pengembangan proyek yang sedang berjalan. Sistem internal yang dikerjakan masih berada dalam tahap pengembangan dan belum dapat diakses oleh publik.

Fokus utama proyek selama pelaksanaan magang adalah pembuatan aplikasi web internal bernama **Attachment Data Hasil Lab Pasien**. Aplikasi ini dirancang untuk mengurangi proses manual, meminimalkan kesalahan penamaan file, serta memungkinkan cabang untuk mengunggah dan memantau lampiran file hasil pemeriksaan laboratorium pasien ke dalam direktori FTP perusahaan secara lebih efisien dan terstruktur.

Beberapa tugas utama yang dilakukan mahasiswa dalam proyek ini meliputi:

- Mendesain dan membangun halaman *dashboard* dan halaman unggah file menggunakan *Bootstrap* untuk menampilkan form pencarian berdasarkan `REG_NO` dan `PATIENT_ID`.

- Mengembangkan fitur upload berbasis baris (*row-based upload*) yang memungkinkan pengguna memilih tipe file, mengunggah file ke FTP, dan menyimpan metadata ke dalam database lokal `MySQL`.
- Mengimplementasikan sistem penamaan file otomatis dengan pola standar `OUTLETID-REGNO-DOCTORID-XX.pdf`, untuk menghindari kesalahan manual dalam penamaan file.
- Menyambungkan sistem ke database Oracle menggunakan SOAP Web Service untuk mengambil detail data pasien berdasarkan input pencarian.
- Menerapkan sistem autentikasi login menggunakan `LDAP Active Directory`, serta menyesuaikan alur login berdasarkan `group ID` untuk membedakan akun admin dan cabang.
- Membuat validasi dan fitur manajemen *master type file*, termasuk fungsi tambah, edit, dan hapus dengan pengecekan duplikasi serta penggunaan flash message sebagai notifikasi sistem.
- Melakukan debugging dan pengujian sistem secara menyeluruh untuk memastikan fitur berjalan sesuai ekspektasi dan tanpa error saat upload ke FTP.

Mahasiswa juga diberikan tugas tambahan selain proyek utama tersebut. Mereka harus membuat dan menyampaikan presentasi internal tentang topik teknologi terbaru, **Vertex AI** dari Google Cloud. Tujuan presentasi ini adalah untuk memperkenalkan kemampuan Vertex AI dalam mendukung pengembangan dan penerapan model *machine learning* end-to-end di lingkungan cloud. Materi yang disampaikan adalah sebagai berikut:

- Penjelasan alur kerja pengembangan ML di Vertex AI, mulai dari pelatihan hingga deployment.
- Fitur *AutoML* yang memungkinkan pelatihan model tanpa coding untuk berbagai jenis data.
- Penggunaan *Vertex Pipelines* dan *MLOps* untuk otomatisasi workflow dan penerapan *best practices* dalam pengelolaan model.
- Fungsi *Model Registry* dan *Dataset Manager* untuk menyimpan, melacak, dan memversikan model serta dataset.

- *Endpoint Deployment* untuk prediksi secara real-time atau batch, lengkap dengan auto-scaling dan monitoring performa.
- Fitur *Monitoring* dan *Explainable AI* yang mendukung deteksi bias dan memberikan transparansi terhadap hasil prediksi.
- Integrasi dengan layanan Google Cloud lainnya seperti GCS, BigQuery, Colab, Vertex Notebooks, dan Vertex Pipelines.

3.3 Uraian Pelaksanaan Magang

Mahasiswa ditempatkan di Divisi Transformasi Digital dan IT **PT. Prodia Widyusaha** selama masa magang, yang berlangsung dari 10 Maret 2025 hingga 30 Juni 2025. Mahasiswa mengikuti beberapa kegiatan orientasi dan pengarahan umum pada minggu pertama magang, yang difasilitasi oleh tim sumber daya manusia dan supervisor dari departemen terkait. Beberapa hal penting berikut termasuk dalam kegiatan orientasi:

- **Pengarahan Umum Mengenai Kebijakan Perusahaan:**
  mahasiswa diberikan pemahaman mengenai tata tertib kerja, jam kerja, budaya perusahaan, serta prosedur yang berlaku untuk pegawai maupun peserta magang.
- **Pengenalan terhadap Lingkungan Kerja dan Infrastruktur Teknologi:**
  Mahassiswa diperkenalkan dengan struktur organisasi Divisi Digital Transformation & IT, serta sistem dan perangkat lunak internal yang digunakan dalam operasional sehari-hari, termasuk penggunaan VPN, FTP, database Oracle, dan sistem autentikasi LDAP.
- **Penyesuaian dengan Workflow dan Tools Pengembangan:**
  Mahasiswwa mulai menyesuaikan diri dengan alur kerja internal, termasuk cara berkomunikasi dengan supervisor melalui Google Chat, pembagian tugas mingguan, serta penggunaan tools pengembangan seperti *Visual Studio Code*, dan phpMyAdmin untuk pengujian lokal.

Pada bulan pertama magang, mahasiswa dilatih untuk mempelajari sistem hasil online yang sudah berjalan, mengevaluasi kebutuhan fitur yang akan dibuat, dan mempelajari sistem upload FTP dan database Oracle perusahaan. Mereka juga

diajarkan tentang SOAP Web Service, yang berfungsi sebagai penghubung antara sistem aplikasi dan database pusat.

Pada bulan kedua dan ketiga, mahasiswa mulai membangun fitur-fitur utama proyek **Attachment Data Hasil Lab Pasien**. Ini termasuk halaman login LDAP, halaman dashboard, form pencarian data pasien, dan kemampuan untuk mengupload file hasil per baris dengan integrasi FTP. Selain itu, mahasiswa juga membuat metode penamaan file otomatis dan validasi input sesuai dengan standar yang telah ditetapkan.

Pada bulan terakhir pelaksanaan magang, sistem kerja berubah menjadi skema *hybrid*, di mana mahasiswa bekerja dua hari secara langsung di kantor (*WFO*) dan tiga hari dari rumah (*WFH*). Selama periode ini, mahasiswa menyelesaikan tahapan pengujian sistem, melakukan debugging, serta menyiapkan dokumentasi teknis dan presentasi hasil kerja kepada pembimbing.

Proses pelaksanaan magang secara keseluruhan berjalan dengan baik, dengan dukungan komunikasi terbuka dari supervisor serta arahan yang diberikan secara rutin setiap minggu. Melalui pelaksanaan magang ini, mahasiswa tidak hanya memperoleh pengalaman praktis dalam pengembangan sistem informasi, tetapi juga memahami pentingnya kolaborasi, dokumentasi, serta ketepatan dalam pengelolaan data sensitif di lingkungan profesional.

3.3.1 Deskripsi Mingguan

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Self-learning: mempelajari lingkungan kerja dan sistem internal PT. Prodia Widyusaha, memahami proyek yang akan dikerjakan, serta mendalami dasar-dasar pemrograman, SOAP Web Service, FTP, dan LDAP.
2	Setup awal project: menyiapkan environment Laravel dan MySQL, instalasi Bootstrap, menyusun struktur direktori dan database awal, serta merancang UI halaman login.
	Lanjut di halaman berikutnya

Tabel 3.1 – lanjutan dari halaman sebelumnya

Minggu Ke -	Pekerjaan yang dilakukan
3	Implementasi login LDAP dan dashboard: menghubungkan login ke Active Directory, membuat halaman dashboard awal dengan navigasi sidebar dan tampilan data summary sederhana.
4	Pengembangan form pencarian data pasien berdasarkan REG_NO dan PATIENT_ID; integrasi SOAP Web Service untuk mengambil detail data pasien dari database Oracle.
5	Implementasi modal upload dan tampilan hasil pencarian: menampilkan data pasien ke dalam modal upload per baris; mulai merancang fitur preview file dan validasi input sebelum upload.
6	Pengembangan sistem unggah file per baris ke FTP, menambahkan dropdown pemilihan tipe file (sementara manual), dan menyusun logika awal penamaan file otomatis.
7	Penyempurnaan logika auto rename file (OUTLETID-REGNO-DOCTORID-XX.pdf), pengujian upload ke FTP, serta penanganan konflik nama file dengan mekanisme increment.
8	Pengembangan fitur konversi otomatis JPG/PNG menjadi PDF sebelum dikirim ke FTP, serta pengujian pipeline upload secara utuh dari input hingga penyimpanan.
9	Mendesain ulang halaman Upload dan Dashboard agar lebih intuitif dan rapi secara UI/UX; menyusun ulang struktur kode agar lebih modular dan scalable.
10	Pengembangan fitur validasi input REG_NO dan PATIENT_ID sebelum upload; menambahkan alert dan notifikasi jika data pasien tidak ditemukan dari Oracle.
11	Refactoring fungsi upload dan integrasi modal dengan logika backend; mulai memisahkan dropdown tipe file menjadi dinamis berdasarkan database.
	Lanjut di halaman berikutnya

Tabel 3.1 – lanjutan dari halaman sebelumnya

Minggu Ke -	Pekerjaan yang dilakukan
12	Menyusun halaman **Setting** yang hanya bisa diakses oleh admin: berisi konfigurasi daftar tipe file (CRUD), halaman ini menggunakan session group_id untuk membatasi akses.
13	Menambahkan fitur **pengelolaan tipe file** (master_type_file) yang disimpan di database MySQL: membuat form tambah/edit/hapus dengan validasi duplikat dan flash message.
14	Mengintegrasikan dropdown tipe file di modal UploadBox agar mengambil data langsung dari database MySQL; memverifikasi dependensi antar form upload dan page Setting.
15	Menambahkan fitur **download lampiran** hasil yang telah diunggah; menambahkan login akun admin dari Oracle; menyimpan metadata hasil upload ke database MySQL, dengan format penamaan yang telah otomatis dan file telah dikonversi ke PDF.

3.3.2 Proyek HPSL Attachment

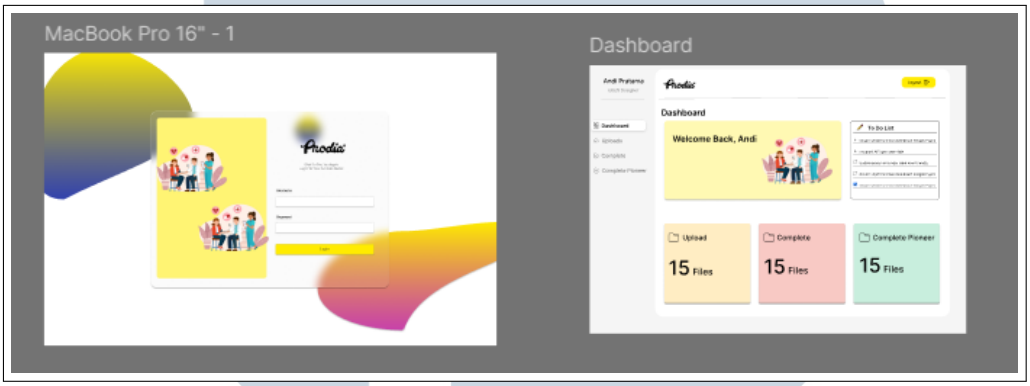
A Perancangan Antarmuka Menggunakan Figma

Sebelum proses pengembangan sistem dimulai, mahasiswa menggunakan platform desain Figma untuk membuat antarmuka pengguna *(user interface)*. Tujuan utama dari proses perancangan ini adalah untuk menunjukkan alur interaksi pengguna dengan sistem dan memastikan bahwa tampilan aplikasi mudah digunakan, mudah dipahami, dan sesuai dengan pengguna internal perusahaan, seperti admin pusat dan karyawan cabang. Seluruh halaman penting yang ada dalam sistem termasuk dalam perancangan ini, termasuk:

- Halaman **Login**.
- Halaman **Dashboard**, menampilkan jumlah file yang telah diunggah ke masing-masing folder berdasarkan status: *Upload*, *Complete*, dan *Complete Pioneer*.
- Halaman **Upload File**, yang menjadi inti dari sistem ini.
- Komponen **Modal UploadBox**, yaitu jendela pop-up untuk mengunggah file.

- Halaman **Setting**, yang hanya dapat diakses oleh pengguna dengan hak akses admin.

Selain estetika visual, desain Figma juga mempertimbangkan elemen UI/UX seperti konsistensi navigasi, keterbacaan teks, penempatan tombol dan responsif terhadap ukuran layar. Pada Gambar 3.1 dan Gambar 3.2 ini adalah salah satu contoh tampilan rancangan halaman unggah file yang dibuat menggunakan Figma:



Gambar 3.1. Dasar Desain Login dan Dashboard

Gambar 3.2. Dasar Desain Uploads dan Setting

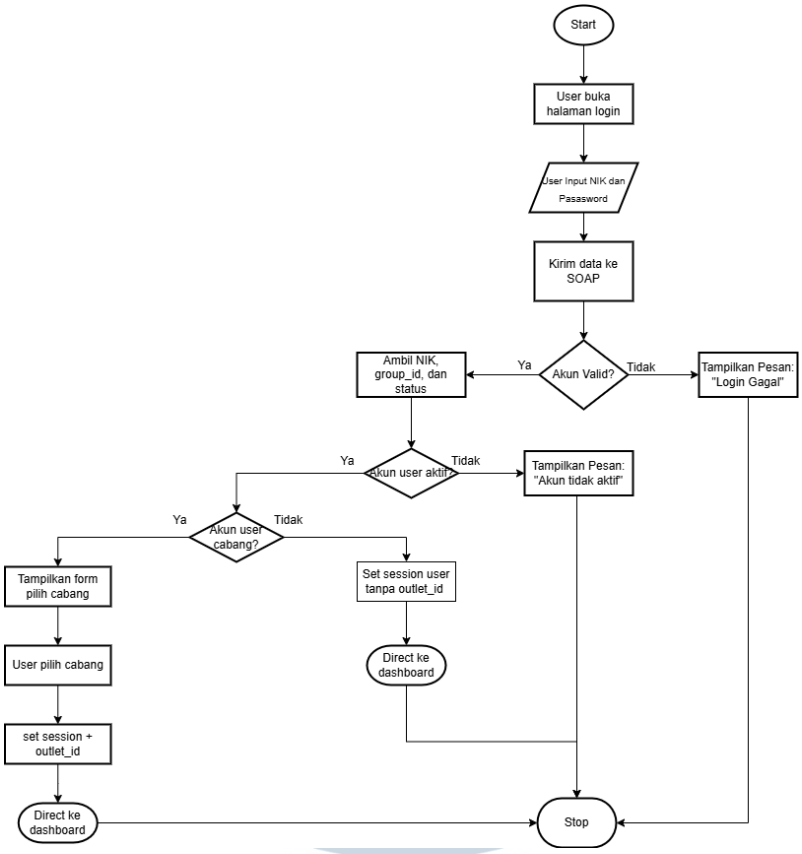
B Login Aplikasi HPSL Attachment

Login pada aplikasi **HPSL Attachment** dapat diakses oleh dua tipe akun *Active Directory (AD)* yaitu akun Admin dan Cabang. Akun yang disimpan pada database Oracle dapat diakses lewat *Web Service SOAP* (Potongan code untuk cek WS dan cek user dapat dilihat pada Kode 3.1 dan 3.2). Setiap akun juga akan memiliki akses halaman yang berbeda. Akses halaman tersebut dapat diatur pada database Oracle.

Secara umum, alur login pada sistem ini adalah sebagai berikut:

1. Pengguna membuka halaman login dan menginput `NIK` dan `Password`.
2. Sistem mengirimkan data tersebut ke layanan *Web Service SOAP* untuk dilakukan verifikasi terhadap data pengguna yang tersimpan di database Oracle.
3. Jika akun ditemukan dan data login valid, sistem akan mengambil informasi akun seperti `group_id` dan status keaktifan akun.
4. Jika akun berstatus aktif, sistem akan memeriksa apakah pengguna merupakan user cabang atau admin:
    - **User Cabang:** sistem akan menampilkan pilihan cabang yang harus dipilih oleh pengguna terlebih dahulu sebelum masuk ke halaman utama.
    - **User Admin:** sistem langsung mengarahkan pengguna ke halaman utama tanpa harus memilih cabang.
5. Setelah berhasil login, informasi pengguna (termasuk `outlet_id` jika ada) akan disimpan dalam session.
6. Berdasarkan `group_id`, pengguna diarahkan ke halaman yang sesuai:
    - **Admin:** dapat mengakses halaman dashboard, manajemen file, pengaturan tipe file, dan statistik.
    - **Cabang:** hanya memiliki akses ke halaman dashboard dan upload lampiran hasil pemeriksaan pasien.

Flow login aplikasi ini dapat dilihat pada Gambar 3.3.



Gambar 3.3. Flowchart Login Aplikasi HPSL Attachment

Gambar 3.3 menunjukan proses dimulai saat pengguna membuka halaman login dan menginputkan NIK serta Password. Data ini kemudian dikirim ke layanan SOAP untuk verifikasi. Jika kredensial tidak valid, sistem akan menampilkan pesan ”Login Gagal” dan proses berhenti.

Apabila akun valid, sistem akan mengambil informasi NIK, `group_id`, dan status akun. Kemudian, dilakukan pengecekan apakah akun tersebut aktif. Jika tidak aktif, pesan ”Akun tidak aktif” akan ditampilkan dan proses berhenti. Untuk akun yang aktif, sistem akan mengidentifikasi apakah pengguna adalah ”Akun user cabang”.

- Jika ya, sistem akan menampilkan form untuk memilih cabang. Setelah pengguna memilih cabang, informasi sesi (termasuk `outlet_id`) akan diatur, dan pengguna akan diarahkan ke halaman dashboard.

- Jika bukan akun user cabang (kemungkinan admin), sesi pengguna akan diatur tanpa `outlet_id` spesifik, dan pengguna langsung diarahkan ke halaman dashboard.

Seluruh alur ini memastikan setiap pengguna mendapatkan akses yang sesuai dengan tipe akun dan status keaktifan mereka.

```
class AttachmentModel extends CI_Model
{
  public function __construct()
  {
    parent::__construct();
        if($this->wscheck($this->config->item('wsall'))){
            $this->wsall = new SoapClient($this->config->item('
   wsall'), array('exceptions' => true));
        }
  }

  function wscheck($ws_url){
        $handle = curl_init($ws_url);
        curl_setopt($handle, CURLOPT_RETURNTRANSFER, TRUE);
        $response = curl_exec($handle);
        $http_code = curl_getinfo($handle, CURLINFO_HTTP_CODE);

        if ($http_code == 200) {
            return true;
        }
        elseif ($http_code == 404) {
            die($http_code . '<hr />Webservice (' . $ws_url . ')
   tidak ditemukan! Silahkan coba beberapa saat lagi.');
        }
        elseif ($http_code == 500) {
            die($http_code . '<hr />Terjadi masalah pada server
   Webservice (' . $ws_url . ')! Silahkan coba beberapa saat lagi.
   ');
        }
        else {
            die($http_code . '<hr />Terjadi masalah yang tidak
   teridentifikasi! Silahkan coba beberapa saat lagi.');
        }
    }

  function array_purifier($data_array)
  {
    $count = 0;
    $cek = is_object($data_array);
    $len = count($data_array);
```

```
36
37    if ($cek) {
38      if ($len > 0) {
39        $data['CHECK']['STATUS'] = true;
40        $data['CHECK']['MESSAGE'] = 'Data Berhasil di-load!';
41        foreach ($data_array as $d) {
42          foreach ($d as $key => $value) {
43            $data['DATA'][$count][$key] = $value->__toString();
44          }
45          $count++;
46        }
47      } else {
48        $data['CHECK']['STATUS'] = false;
49        $data['CHECK']['MESSAGE'] = 'Tidak ada data ditemukan!';
50      }
51    } else {
52      $data['CHECK']['STATUS'] = false;
53      $data['CHECK']['MESSAGE'] = 'Data tidak dikenal!';
54    }
55    return $data;
56  }
57
58  function array_purifier_multicursor($data_array) {
59        $count = 0;
60
61        $cek = is_object($data_array);
62        $len = count($data_array);
63
64        if ($cek) {
65            if ($len > 0) {
66                $data['CHECK']['STATUS'] = true;
67                $data['CHECK']['MESSAGE'] = 'Data Berhasil di-load
   !';
68                foreach ($data_array as $d) {
69                    foreach ($d as $key => $value) {
70                        $data['DATA'][$count][$key] = $value->
   __toString();
71                    }
72                    $count++;
73                }
74            } else {
75                $data['CHECK']['STATUS'] = false;
76                $data['CHECK']['MESSAGE'] = 'Tidak ada data
```

```
    ditemukan!';
77              }
78          } else {
79              $data['CHECK']['STATUS'] = false;
80              $data['CHECK']['MESSAGE'] = 'Data tidak dikenal!';
81          }
82
83          return $data;
84      }
```

Kode 3.1: Fungsi wsCheck() pada AttachmentModel.php

Fungsi `wsCheck()` pada Kode 3.1 bertanggung jawab untuk memverifikasi ketersediaan dan responsivitas *Web Service SOAP* dengan melakukan pengecekan status HTTP. Fungsi ini memastikan bahwa aplikasi dapat terhubung ke *web service* yang ditentukan (`ws_url`) sebelum mencoba operasi lebih lanjut. Jika *web service* tidak dapat diakses (misalnya kode HTTP 404 atau 500), fungsi akan menampilkan pesan kesalahan yang informatif kepada pengguna dan menghentikan eksekusi. Ini penting untuk menjaga stabilitas dan penanganan kesalahan koneksi pada aplikasi.

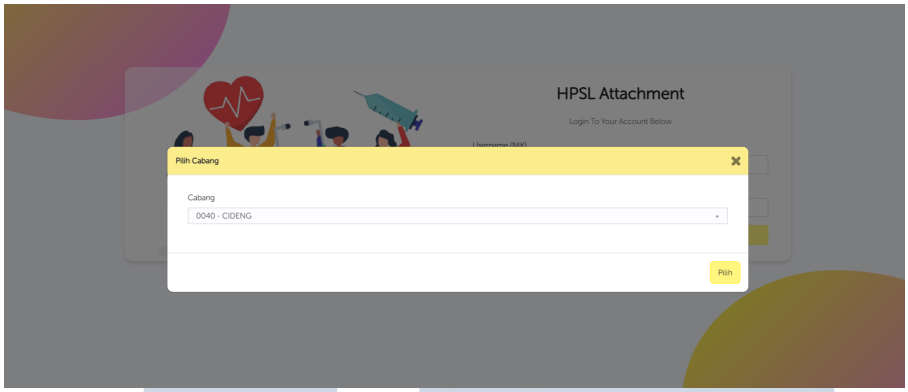
```
1  public function cek_user($username, $res)
2    {
3      $webapp_id = $this->config->item('webapp_id');
4      $nik_admin = $this->config->item('nik_admin');
5
6      // Check ke WEBAPP_USER dan WEBAPP_USER_GROUP_APP
7      $result = $this->AttachmentModel->
     PKG_WEBAPP_MENU__BRW_USER_LOGIN($username, $webapp_id);
8      if ( @$result['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
9        $this->session->set_flashdata('error', 'Maaf, Terdapat error
       pada aplikasi silahkan hubungi admin [. substr($result['DATA'
     ][0]['MESSAGE'], 0, 9) .']');
10       redirect('login');
11     }
12
13     if ($result['CHECK']['STATUS']) {
14       // Check ke WEBAPP_GROUP dan WEBAPP_USER_GROUP_APP
15       $group_user = $this->AttachmentModel->
     PKG_WEBAPP_MENU__BRW_WEBAPP_GROUP_BY_USER($username, $webapp_id
     );
16       if ( $group_user['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
17         $this->session->set_flashdata('error', 'Maaf, Terdapat
```

```
error pada aplikasi silahkan hubungi admin [’. substr($result[’
DATA’][0][’MESSAGE’], 0, 9) .’]’);
18        redirect(’login’);
19      }
20
21      if ($group_user[’DATA’][0][’ID’] == ’G10029’) {
22        $result_user_outlet = $this->AttachmentModel->
PKG_WEBAPP_MENU__BRW_USER_PER_OUTLET($username, $webapp_id);
23        if ( $result_user_outlet[’DATA’][0][’CEK_STATUS’] == ’
ERROR’ ) {
24          $this->session->set_flashdata(’error’, ’Maaf, Terdapat
error pada aplikasi silahkan hubungi admin [’. substr(
$result_user_outlet[’DATA’][0][’MESSAGE’], 0, 9) .’]’);
25          redirect(’login’);
26        } else if ( empty($result_user_outlet[’DATA’]) ) {
27          $this->session->set_flashdata(’error’, ’Maaf, Outlet
tidak ditemukan’);
28          redirect(’login’);
29        }
30        $this->session->set_flashdata(’cabang’, true);
31        $this->session->set_flashdata(’outlet’,
$result_user_outlet[’DATA’]);
32        $this->session->set_flashdata(’xsis’, ’nonxsis’);
33        redirect(’login’);
34      } else {
35        $res[’group_id’] = $group_user[’DATA’][0][’ID’];
36        $res[’group_name’] = $group_user[’DATA’][0][’NAME’];
37        $res[’outlet_id’] = ’-’;
38        $res[’outlet_name’] = ’-’;
39        $this->session->set_userdata($res);
40        $this->session->set_flashdata(’success’, ’Selamat, Anda
telah berhasil login sebagai ’.$res[’group_name’]);
41        redirect(’dashboard’);
42      }
43    } else {
44      // Check apakah user adalah super admin
45      if ($username == $nik_admin) {
46                $res[’group_id’] = "AA-1";
47                $res[’group_name’] ="Admin Attachment Apps";
48        $res[’outlet_id’] = ’-’;
49        $res[’outlet_name’] = ’-’;
50
51        $this->session->set_userdata($res);
```

```
52          $this->session->set_flashdata('success', 'Selamat, Anda
    telah berhasil login sebagai '.$res['group_name']);
53          redirect('dashboard');
54        } else {
55          $resultOutlet = $this->AttachmentModel->EREG_CMB_OUTLET(
    $username);
56          if ( $resultOutlet['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
57            $this->session->set_flashdata('error', 'Maaf, Terdapat
    error pada aplikasi silahkan hubungi admin ['. substr(
    $resultOutlet['DATA'][0]['MESSAGE'], 0, 9) .']');
58            redirect('login');
59          } else if ( empty($resultOutlet['DATA']) ) {
60            $this->session->set_flashdata('error', 'Maaf, Anda belum
     terdaftar outlet manapun');
61            redirect('login');
62          }
63
64          $this->session->set_flashdata('cabang', true);
65          $this->session->set_flashdata('outlet', $resultOutlet['
    DATA']);
66          $this->session->set_flashdata('xsis', 'xsis');
67          redirect('login');
68        }
69      }
70    }
```

Kode 3.2: Fungsi cek_user() pada Login.php

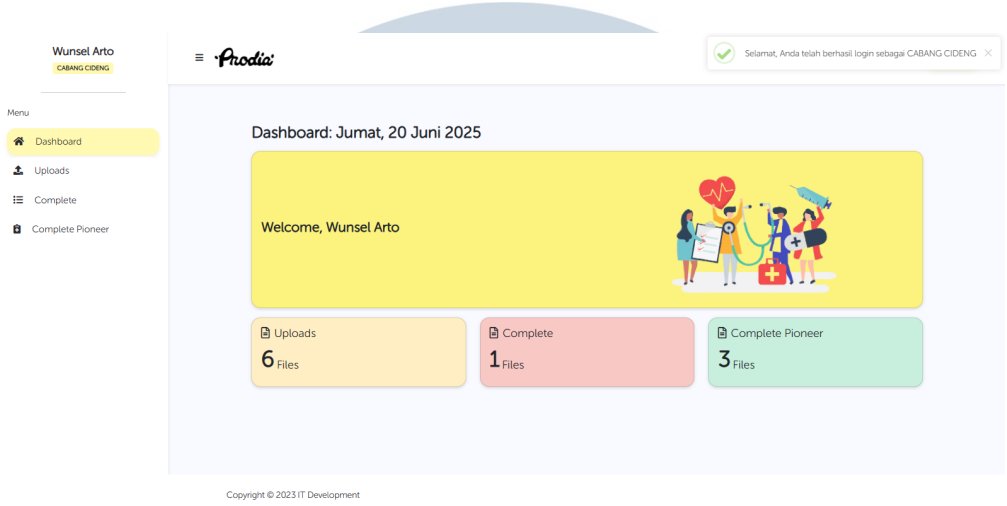
Fungsi `cek_user()` pada Kode 3.2 merupakan inti dari proses otentikasi pengguna pada aplikasi. Fungsi ini bertugas memverifikasi kredensial `NIK` dan `Password` pengguna terhadap database Oracle melalui panggilan ke *Web Service* internal (`PKG_WEBAPP_MENU__BRW_USER_LOGIN` dan `PKG_WEBAPP_MENU__BRW_WEBAPP_GROUP_BY_USER`). Selain itu, fungsi ini juga menentukan jenis akun pengguna (Admin atau Cabang) dan memuat informasi hak akses terkait (`group_id`, `outlet_id`). Hasil verifikasi ini akan memengaruhi jalur navigasi pengguna setelah berhasil login, serta data yang disimpan dalam sesi aplikasi. Fungsi ini juga menangani skenario untuk super admin dan pengguna outlet yang belum terdaftar.



Gambar 3.4. Visualisasi Login Cabang

Gambar 3.4 menunjukan setelah pengguna berhasil menginputkan `NIK` dan `Password` yang valid, sebuah *modal pop-up* dengan judul **"Pilih Cabang"** akan muncul di atas halaman login utama. *Pop-up* ini dirancang untuk memastikan bahwa pengguna tipe akun Cabang memilih lokasi operasional mereka sebelum mengakses fitur aplikasi. Di dalam *pop-up* tersebut, terdapat sebuah *dropdown menu* dengan label **Cabang**, yang berisi daftar cabang yang terkait dengan akun pengguna. Dalam contoh ini, cabang yang terlihat adalah "CIDENG". Pengguna diwajibkan untuk memilih salah satu cabang dari daftar yang tersedia. Setelah pemilihan, tombol **Pilih** di bagian bawah kanan *modal* digunakan untuk mengkonfirmasi pilihan dan melanjutkan proses masuk ke aplikasi. Desain ini bertujuan untuk mengarahkan pengguna ke halaman dan data yang relevan dengan cabang yang dipilih.

C Flow Dashboard Aplikasi HPSL Attachment



Gambar 3.5. Dashboard Aplikasi HPSL Attachment

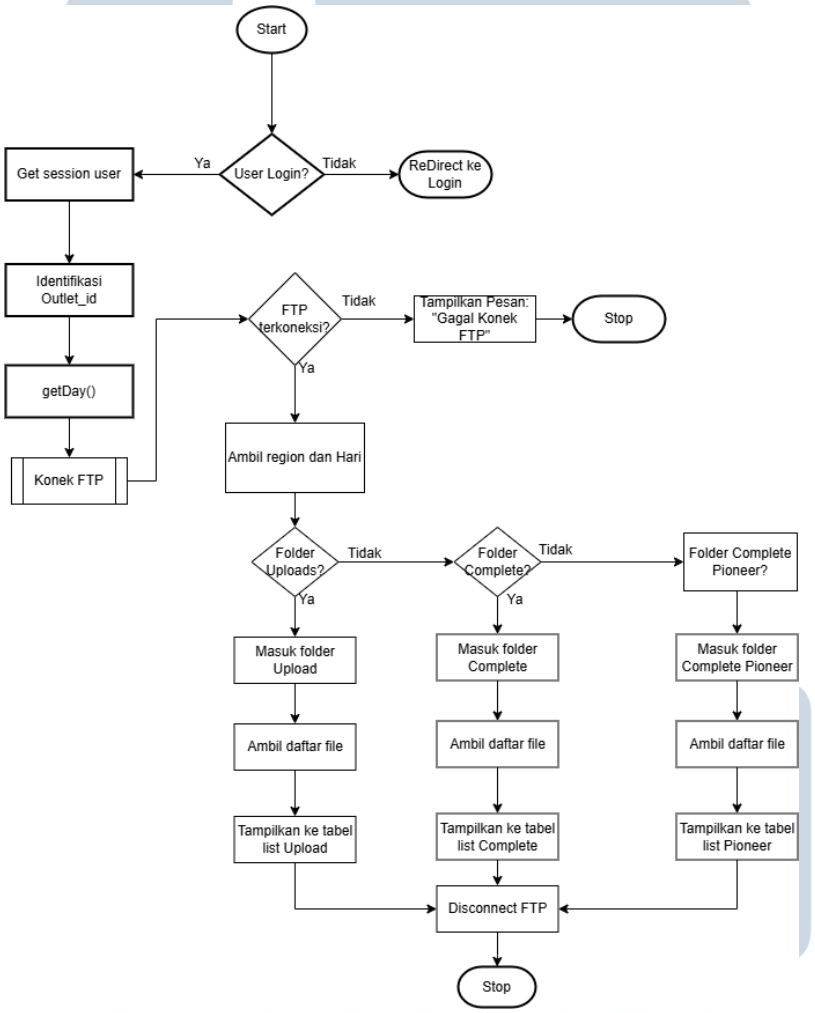
Gambar 3.5 menunjukan halaman dashboard memiliki fungsi utama untuk menampilkan total file lampiran hasil yang ada pada wilayah dan hari yang ada pada session akun tersebut. total file diambil dari server FTP. Direktori yang digunakan pada aplikasi HPSL Attachment App ada tiga direktori, yaitu `Upload`, `Complete`, dan `Complete_Pioneer`, setiap direktori memiliki subfolder bernama wilayah yang berisi wilayah dari 1 - 8, Lalu didalam folder wilayah terdapat folder hari dari senin sampai minggu. Sistem akan mendapatkan file lampiran ketika sudah akses folder tersebut.

Ketika pengguna membuka halaman dashboard, sistem akan:

1. Memverifikasi apakah pengguna telah login. Jika belum, pengguna akan diarahkan ke halaman login.
2. Mengambil data `outlet_id` dan `group_id` dari sesi login untuk menentukan wilayah pengguna.
3. Mengambil nama hari saat ini menggunakan fungsi this- custom- getDay().
4. Membuka koneksi ke server FTP.
5. Mengakses dan membaca file dari folder:
    - `Upload/WilayahX/HariIni`

- `Complete/WilayahX/HariIni`
- `Complete_Pioneer/WilayahX/HariIni`

Flow proses FTP ini dapat dilihat pada Gambar 3.6 dan kode akses FTP ini dapat dilihat pada Kode 3.3.



Gambar 3.6. Flowchart Proses FTP Aplikasi HPSL Attachment

Gambar 3.6 menunjukan alur dimulai dengan pengecekan status login pengguna. Jika pengguna belum login, sistem akan mengarahkan mereka kembali ke halaman login. Apabila sudah login, sistem akan mengambil data sesi pengguna, mengidentifikasi `outlet_id`, dan mendapatkan informasi tanggal (`getDay()`). Selanjutnya, aplikasi akan mencoba untuk melakukan koneksi ke server FTP. Jika koneksi gagal, pesan ”Gagal Koneksi FTP” akan ditampilkan dan proses berhenti.

Jika koneksi FTP berhasil, sistem akan mengambil data region (wilayah) pengguna dan hari ini. Kemudian, alur akan memeriksa keberadaan tiga folder utama secara berurutan: ”Uploads”, ”Complete”, dan ”Complete Pioneer”. Untuk setiap folder yang ada, sistem akan masuk ke dalam folder tersebut, mengambil daftar file di dalamnya, dan menampilkannya ke dalam tabel daftar yang sesuai (misalnya, ”Tabel list Upload”, ”Tabel list Complete”, ”Tabel list Pioneer”). Setelah semua proses pengambilan dan tampilan data selesai untuk ketiga folder, koneksi FTP akan diputus (Disconnect FTP) dan proses berakhir.

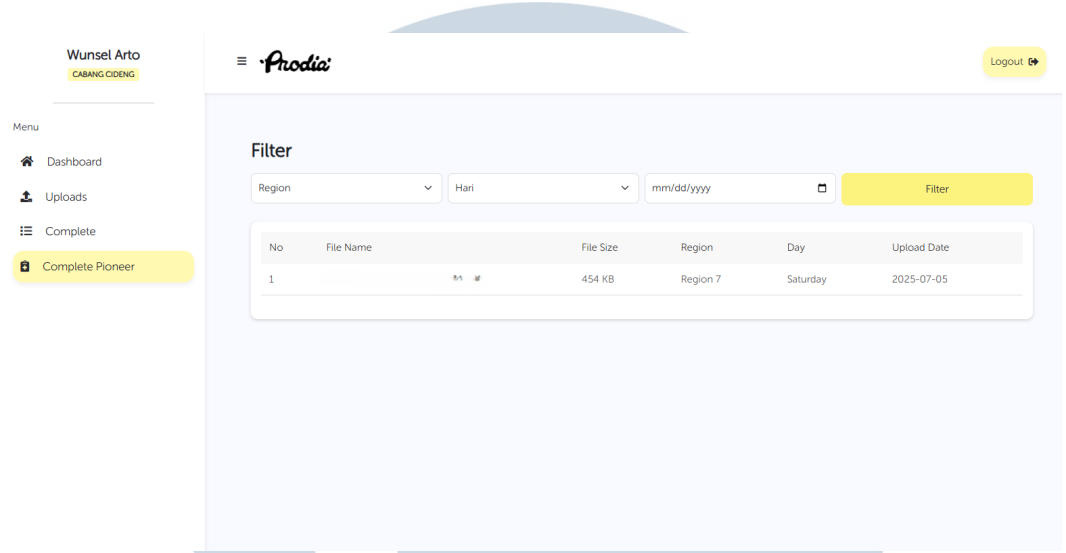
```
$result = $this->AttachmentModel->
 PKG_ATTACHMENT_APPS__BRW_REGION_BY_OUTLET($param);

 if ( @$result['DATA'][0]['CEK_STATUS'] == 'ERROR' || @$result[
'CHECK']['STATUS'] == FALSE) {
   $this->session->set_flashdata('error', 'Maaf, Terdapat error
 pada aplikasi silahkan hubungi admin [' . ($result['CHECK']['
STATUS'] == FALSE ? @$result['CHECK']['MESSAGE'] : substr(
@$result['DATA'][0]['MESSAGE'], 0, 9)) . ']');
   // Pemanggilan View
   $data['content'] = "page/dashboard";
   $data['title'] = "Dashboard";
   $this->load->view('template/page_navbar', $data);
 }

 $resultWilayah = 'WIL ' . (int)@$result['DATA'][0];

 try {
   $host = $this->config->item('sftp_host');
   $username = $this->config->item('sftp_username');
   $password = $this->config->item('sftp_password');
   $ftp = $this->custom->connectFtp($host, $username, $password
);

   // Bisa langsung pakai $ftp buat upload, download, dll
   // var_dump($_SESSION);die();
   $uploads = ftp_nlist($ftp, "/Uploads/".$resultWilayah."/".
$this->custom->getDay());
   $data['uploads'] = is_array($uploads) ? count($uploads) : 0;

   $uploads = ftp_nlist($ftp, "/Complete/".$resultWilayah."/".
$this->custom->getDay());
   $data['complete'] = is_array($uploads) ? count($uploads) :
```

```
    0;

26
27      $uploads = ftp_nlist($ftp, "/Complete_Pioneer/".
    $resultWilayah."/".$this->custom->getDay());
28      $data['complete_pioneer'] = is_array($uploads) ? count(
    $uploads) : 0;

29
30      $this->custom->closeFtp($ftp); // jangan lupa tutup koneksi
31    } catch (Exception $e) {
32      $this->session->set_flashdata('error', $e->getMessage());
33    }

34
35    // Pemanggilan View
36        $data['content'] = "page/dashboard";
37    $data['title'] = "Dashboard";
38    $this->load->view('template/page_navbar', $data);
39  }
```

Kode 3.3: Akses FTP

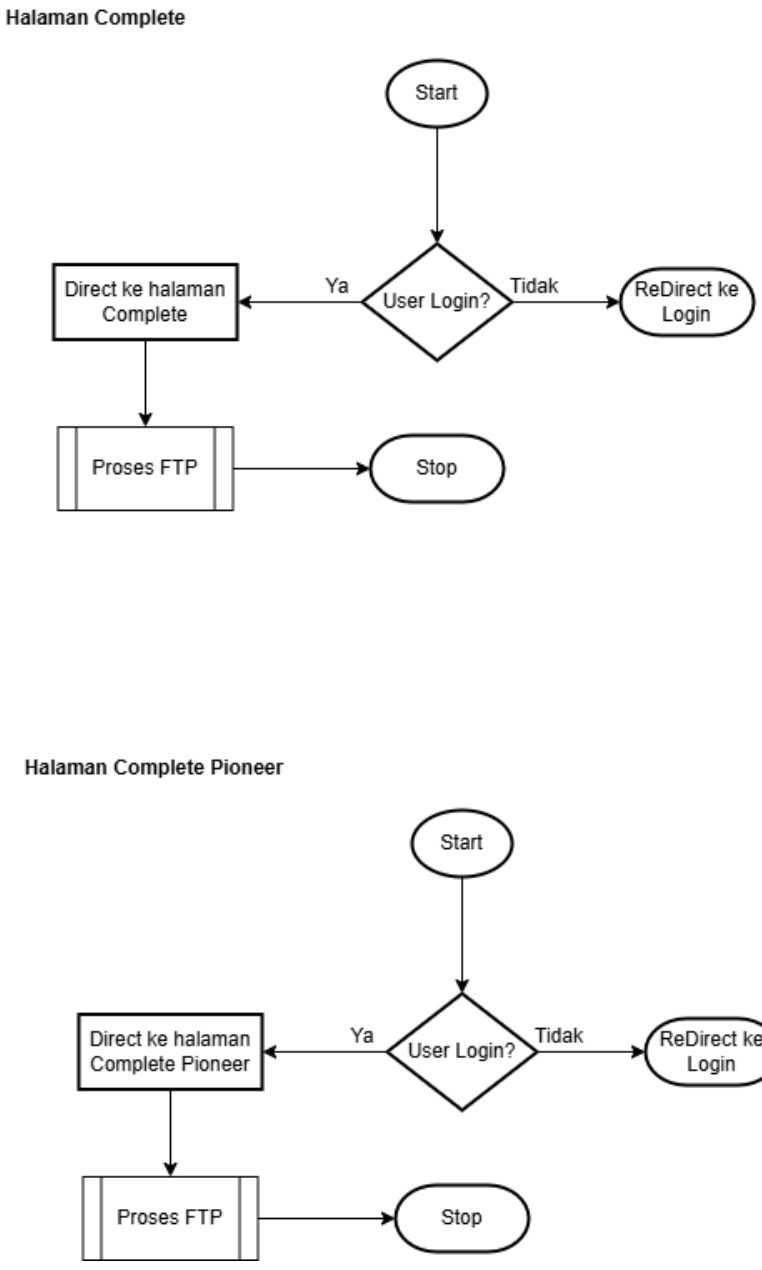
Kode 3.3 `Dashboard.php` ini berfungsi sebagai pengendali utama untuk halaman dashboard aplikasi. Tugas utamanya adalah memastikan pengguna telah login sebelum mengakses dashboard. Setelah itu, kode akan mengambil informasi wilayah pengguna dari database melalui *web service*. Informasi wilayah ini penting untuk kemudian digunakan dalam menghubungkan aplikasi ke server FTP. Setelah koneksi FTP berhasil, kode akan menghitung jumlah file yang ada di beberapa folder khusus (Uploads, Complete, dan Complete_Pioneer) di server FTP, sesuai dengan wilayah dan tanggal saat ini. Hasil perhitungan jumlah file ini akan ditampilkan di dashboard. Kode juga dilengkapi dengan penanganan kesalahan untuk koneksi FTP dan memastikan koneksi ditutup setelah selesai. Pada akhirnya, semua data yang terkumpul akan ditampilkan pada halaman dashboard.

D Halaman Complete dan Complete_Pioneer Aplikasi HPSL Attachment



Gambar 3.7. Halaman Complete dan Complete Pioneer Aplikasi HPSL Attachment

Pada Gambar 3.7 ditunjukan halaman Complete dan Complete Pioneer memiliki fungsi yang sama dan juga layout yang sama. Pada halaman ini aplikasi akan menampilkan list file lampiran yang ada pada masing-masing direktori. Halaman Complete akan menampilkan list file dari direktori `Complete`, dan `Complete_Pioneer` juga akan menampilkan list file dari direktori Complete_Pioneer. List file akan ditampilkan di tabel dengan kolom `File Name`, `File Size`, `Region`, `Day`, dan `Upload Date`. Pada Halaman Complete dan Complete Pioneer juga diberikan fungsi filter berdasarkan `REG_NO` dan `PATIENT_ID`.



Gambar 3.8. Flowchart Halaman Complete dan Complete Pioneer Aplikasi HPSL Attachment

Gambar 3.8 menunjukan kedua alur dimulai dengan validasi status login pengguna. Jika pengguna tidak terautentikasi (tidak login), mereka akan dialihkan kembali ke halaman login. Sebaliknya, jika pengguna sudah login, mereka akan langsung diarahkan ke halaman yang sesuai, yaitu ”Halaman Complete” atau ”Halaman Complete Pioneer”. Setelah berhasil dialihkan ke halaman tersebut, proses akan dilanjutkan dengan ”Proses FTP”. Kotak ”Proses FTP”

ini merepresentasikan serangkaian operasi yang melibatkan koneksi dan interaksi dengan server FTP, seperti mengambil daftar file di folder ”Complete” atau ”Complete Pioneer”. Setelah ”Proses FTP” selesai, alur untuk masing-masing halaman akan berhenti. Secara keseluruhan, flowchart ini menggarisbawahi prasyarat login dan integrasi dengan operasi FTP untuk menampilkan data relevan di kedua halaman ini.

E Halaman Upload Aplikasi HPSL Attachment

Halaman Upload menjadi antarmuka bagi pengguna cabang untuk unggah file lampiran pasien kedalam direktori FTP dan menyimpannya ke database MySQL. Proses upload memiliki standarisasi dimana tipe file yang dapat di unggah dibatasi. Tipe file yang diberi akses untuk unggah adalah file sperti, PNG, JPG, JPGEG, dan PDF. Apabila user mencoba untuk unggah file dengan tipe file lain maka aksesnya akan ditolak.

Fungsi utama dari halaman ini adalah memungkinkan pengguna untuk:

- Melakukan pencarian data hasil lab berdasarkan `REG_NO` dan `PATIENT_ID`.
- Menampilkan data pasien secara dinamis dalam bentuk *pop-up modal*.
- Mengunggah lampiran file hasil pemeriksaan ke FTP dengan format nama file yang telah distandarkan.
- Memilih tipe file lampiran yang sesuai melalui `dropdown` yang terhubung ke data master tipe file dari database MySQL.
- Menyimpan metadata file ke database lokal sebagai dokumentasi dan audit trail.

Setelah pengguna memasukkan `REG_NO` dan `PATIENT_ID`, aplikasi akan mengirim permintaan SOAP ke sistem Oracle perusahaan untuk mengambil data pemeriksaan terkait. Data yang diterima akan ditampilkan dalam modal `UploadBox`, yang berisi informasi pasien serta form untuk unggah file.

Setiap baris dalam form upload memungkinkan pengguna untuk memilih:

1. Tipe file dari `dropdown` (wajib diisi).
2. File lampiran dari perangkat lokal.

3. Melakukan proses unggah dengan tombol *Upload* per baris.

Nama file akan diubah secara otomatis menggunakan struktur berikut:

`OUTLETID-REGNO-DOCTORID-XX.pdf`

File juga akan dikonversi otomatis menjadi format PDF (jika berupa gambar JPG/PNG) sebelum dikirim ke FTP.

Setelah berhasil diunggah, metadata file (seperti `reg_no`, `patient_id`, `doctor_id`, `file_type`, `file_name`, dan `upload_date`) akan disimpan ke dalam tabel `uploaded_files` di database MySQL.

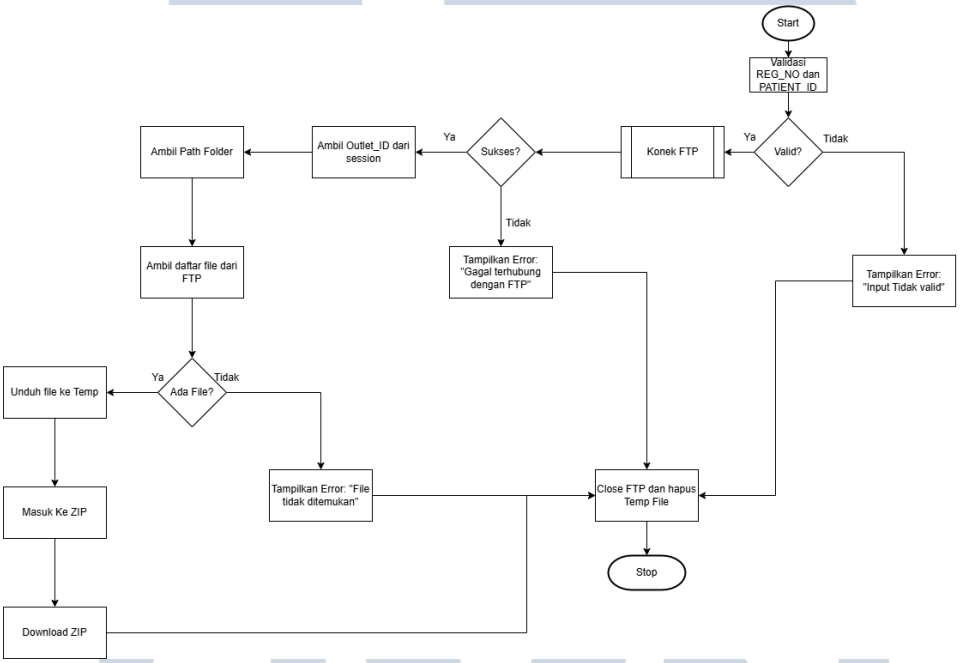
Gambar 3.9. Proses Convert PDF Aplikasi HPSL Attachment

Gambar 3.9 menunjukan proses convert PDF dimulai dengan melakukan casting variabel `$original_ext` ke tipe data string. Selanjutnya, sistem memuat library ”Pdf_converter” dan membuat objek konverter baru. Halaman baru kemudian ditambahkan ke dokumen PDF yang sedang dibuat. Langkah berikutnya adalah menentukan tipe gambar masukan. Sistem akan memeriksa apakah tipe gambar adalah JPEG, PNG, atau JPG secara berurutan.

- Jika gambar bertipe JPEG, tipe akan diset sebagai JPEG.
- Jika gambar bertipe PNG, tipe akan diset sebagai PNG.

- Jika gambar bertipe JPG, tipe akan diset sebagai JPG.

Setelah tipe gambar ditentukan, gambar tersebut akan ditambahkan ke halaman PDF. Namun, jika tipe gambar tidak didukung (bukan JPEG, PNG, atau JPG), sebuah pengecualian (*Exception*) dengan pesan ”Tipe Gambar tak didukung” akan ditampilkan. Proses konversi akan berhenti setelah gambar berhasil ditambahkan ke PDF atau jika terjadi pengecualian.



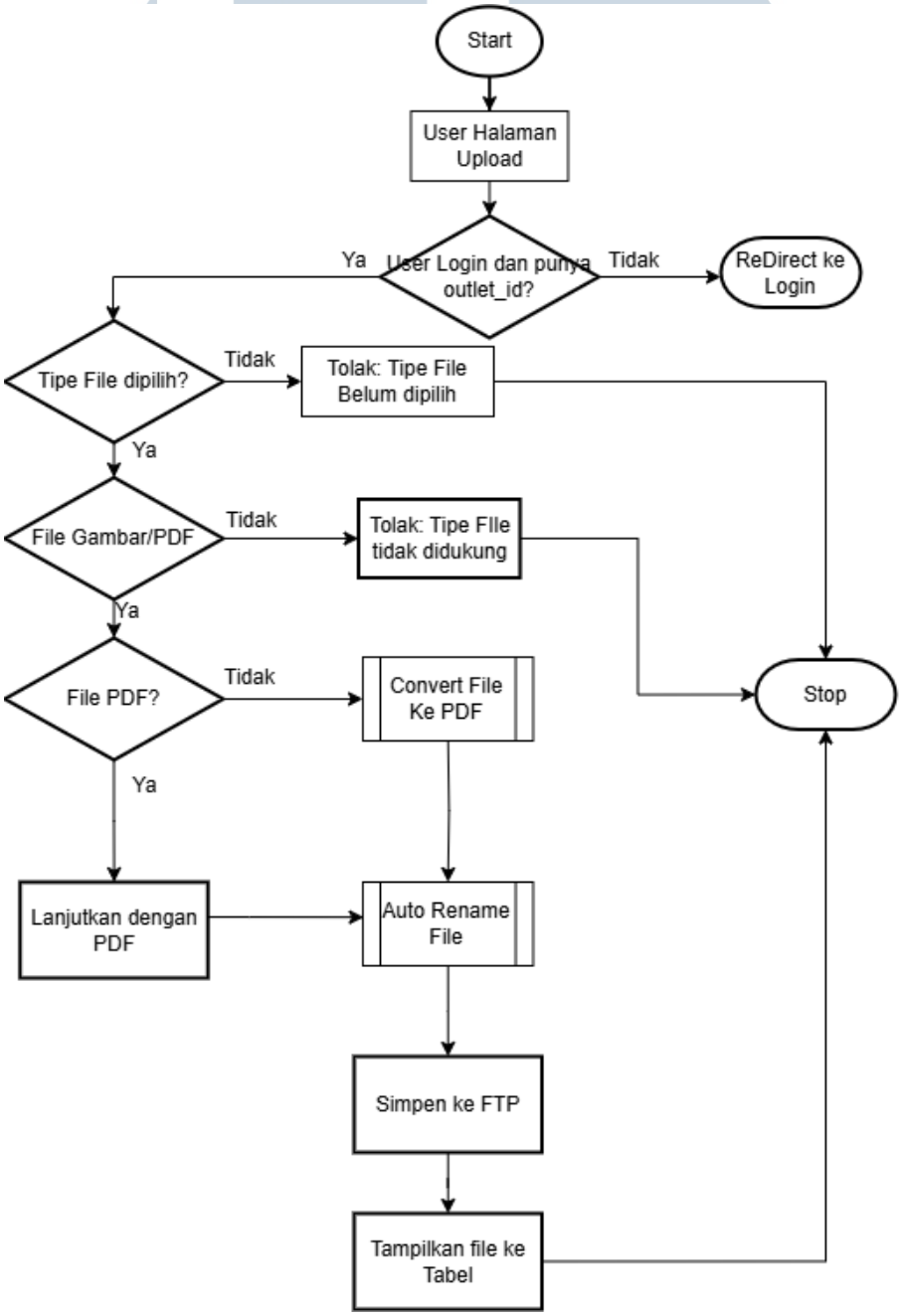
Gambar 3.10. Proses Download Attachment Aplikasi HPSL Attachment

Gambar 3.10 menunjukan proses download attachment dimulai dengan validasi input `REG_NO` dan `PATIENT_ID`. Jika input tidak valid, sistem akan menampilkan pesan error ”Input Tidak valid” dan proses berhenti setelah menutup koneksi FTP (jika ada) dan menghapus file temporer.

Jika input valid, sistem akan mencoba mengkoneksikan ke server FTP. Apabila koneksi FTP gagal, pesan error ”Gagal terhubung dengan FTP” akan ditampilkan, dan proses akan berhenti setelah pembersihan. Jika koneksi sukses, sistem akan mengambil `Outlet_ID` dari sesi pengguna dan menentukan jalur folder yang relevan di server FTP.

Selanjutnya, sistem akan mencoba mengambil daftar file dari jalur FTP tersebut. Jika tidak ada file yang ditemukan, pesan error ”File tidak ditemukan” akan ditampilkan, diikuti dengan penutupan koneksi FTP dan penghapusan file

temporer, lalu proses berhenti. Namun, jika ada file, file tersebut akan diunduh ke direktori temporer. Setelah unduhan, file-file tersebut akan dimasukkan ke dalam arsip ZIP, yang kemudian akan diunduh oleh pengguna. Terakhir, koneksi FTP akan ditutup dan file temporer dihapus sebelum proses berakhir.



Gambar 3.11. Flow Upload Aplikasi HPSL Attachment

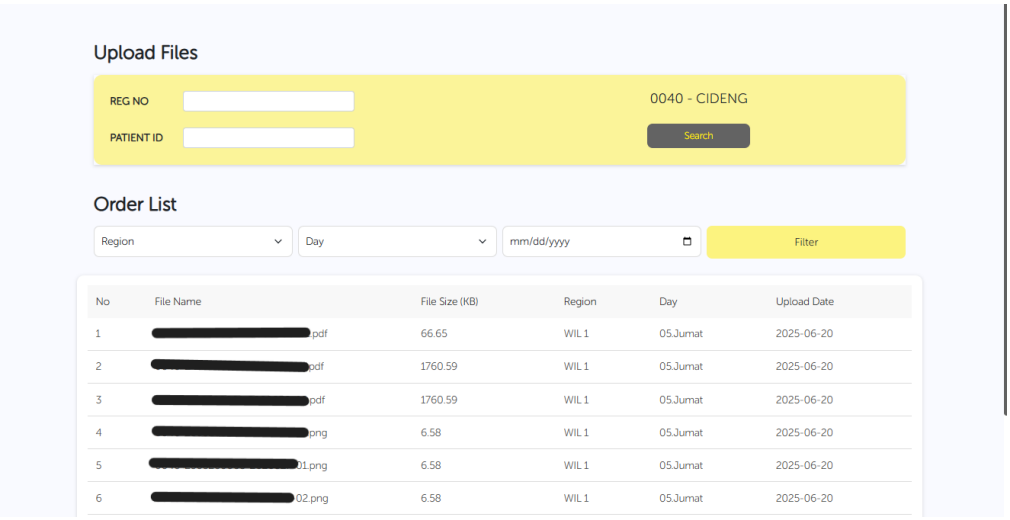
Gambar 3.11 menunjukan alur proses unggah file dimulai ketika pengguna

mengakses halaman unggah. Pertama-tama, sistem akan memeriksa apakah pengguna sudah login dan memiliki `outlet_id`. Jika tidak, pengguna akan dialihkan kembali ke halaman login, dan proses berhenti.

Jika pengguna sudah login dan memiliki `outlet_id`, sistem akan memverifikasi apakah tipe file telah dipilih oleh pengguna. Apabila belum, proses akan ditolak dengan pesan ”Tolak: Tipe File Belum dipilih”, dan proses berhenti. Selanjutnya, sistem akan mengecek apakah file yang diunggah adalah file gambar atau PDF. Jika bukan, proses akan ditolak dengan pesan ”Tolak: Tipe File tidak didukung”, dan proses berhenti. Apabila file adalah gambar atau PDF, sistem akan memeriksa apakah file tersebut adalah PDF.

- Jika file adalah PDF, proses akan langsung dilanjutkan.
- Jika file bukan PDF (berarti gambar), sistem akan melakukan konversi file gambar tersebut menjadi format PDF.

Setelah file dalam format PDF (baik asli maupun hasil konversi), file akan secara otomatis diubah namanya (auto rename file). Kemudian, file tersebut akan disimpan ke server FTP. Terakhir, informasi atau file yang diunggah akan ditampilkan ke tabel di antarmuka pengguna, dan proses selesai.

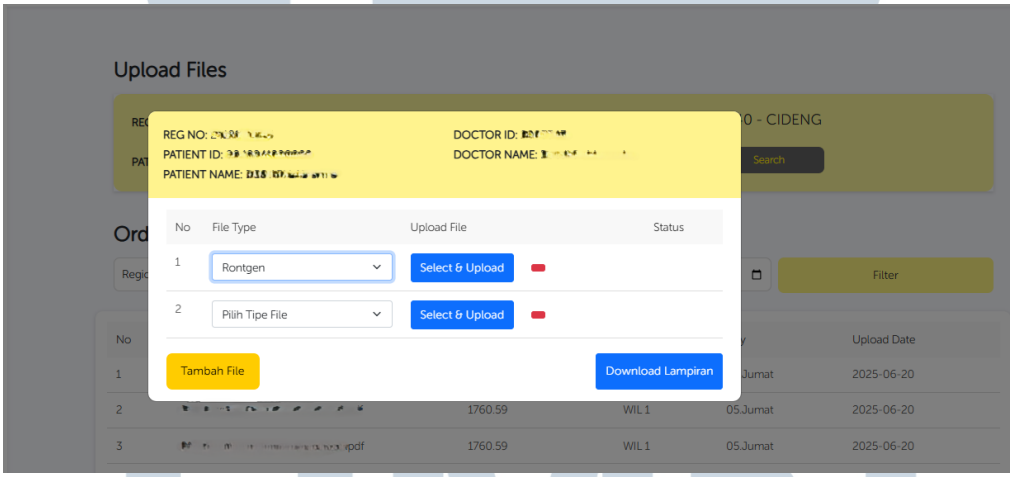


Gambar 3.12. Tampilan Halaman Upload Aplikasi HPSL Attachment

Gambar 3.12 menunjukan halaman upload aplikasi. Pada bagian atas, terdapat area untuk input data. Pengguna dapat memasukkan ”REG NO” dan ”PATIENT ID” pada kolom yang tersedia. Di sebelah kanan, terlihat

informasi cabang yang sedang aktif (”0040 - CIDENG”), mengindikasikan bahwa tampilan dan operasi akan disesuaikan dengan cabang tersebut. Tombol ”Search” kemungkinan digunakan untuk mencari file berdasarkan kriteria yang dimasukkan.

Bagian ”Order List” menampilkan daftar file yang sudah diunggah. Terdapat opsi filter berdasarkan ”Region”, ”Day”, dan rentang tanggal (”mm/dd/yyyy”) yang dilengkapi dengan ikon kalender, serta tombol ”Filter” untuk menerapkan penyaringan. Di bawahnya, sebuah tabel menyajikan detail file dengan kolom ”No”, ”File Name”, ”File Size (KB)”, ”Region”, ”Day”, dan ”Upload Date”. Daftar ini memungkinkan pengguna untuk melihat dan mengelola file-file yang telah diunggah dengan mudah, memberikan gambaran yang jelas mengenai dokumen yang tersedia.

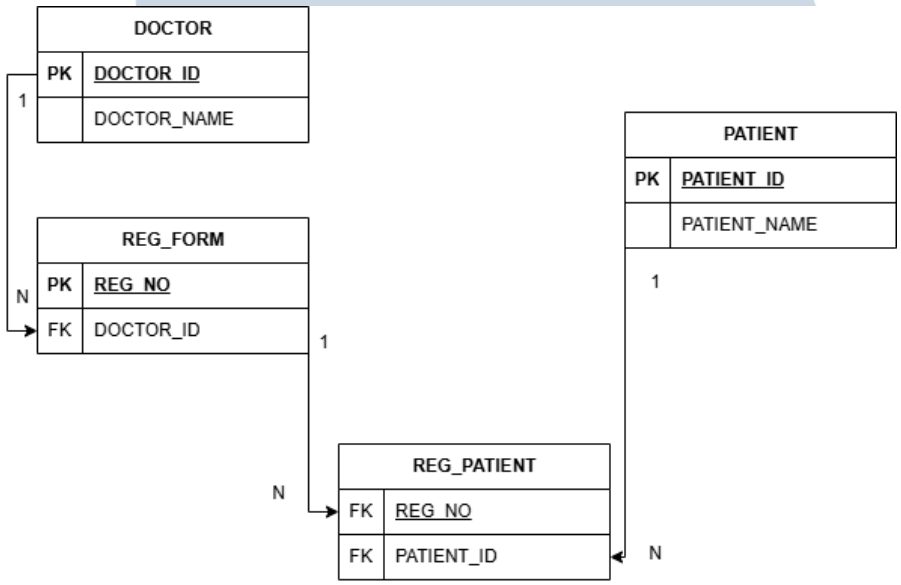


Gambar 3.13. Pop Up Halaman Upload Aplikasi HPSL Attachment

Gambar 3.13 menunjukan pop-up/modal pada halaman upload. Pada bagian atas *modal*, ditampilkan informasi kunci yang relevan dengan pasien, seperti `REG NO`, `PATIENT ID`, `PATIENT NAME`, `DOCTOR ID`, `DOCTOR NAME`. Informasi ini memastikan bahwa lampiran yang diunggah dikaitkan dengan data pasien dan dokter yang benar.

Di bawah informasi pasien, terdapat tabel dengan kolom ”No”, ”File Type”, ”Upload File”, dan ”Status”. Pada kolom ”File Type”, pengguna dapat memilih jenis dokumen melalui *dropdown menu* (contoh: ”Rontgen”). Kolom ”Upload File” menyediakan tombol ”Select & Upload” untuk memilih dan memulai proses unggah file. Tombol merah di sebelahnya kemungkinan berfungsi untuk menghapus baris input jika tidak jadi diunggah.

Tombol ”Tambah File” memungkinkan pengguna untuk menambahkan baris input baru, memungkinkan unggahan beberapa dokumen sekaligus. Terakhir, tombol ”Download Lampiran” mengindikasikan fitur untuk mengunduh lampiran yang sudah ada atau yang baru diunggah setelah proses selesai. *Modal* ini dirancang untuk mempermudah dan memandu pengguna dalam mengunggah berbagai jenis dokumen yang terkait dengan data pasien.



Gambar 3.14. ERD Search Data

Gambar 3.14 menunjukan ERD dari pencarian data order. ERD ini terdiri dari empat entitas utama:

1. **DOCTOR:** Entitas ini menyimpan informasi mengenai dokter, dengan `DOCTOR_ID` sebagai kunci utama (PK) dan atribut `DOCTOR_NAME`.
2. **PATIENT:** Entitas ini berisi data pasien, dengan `PATIENT_ID` sebagai kunci utama (PK) dan atribut `PATIENT_NAME`.
3. **REG_FORM:** Entitas ini tampaknya mewakili formulir registrasi atau data kunjungan, dengan `REG_NO` sebagai kunci utama (PK). Entitas ini juga memiliki kunci asing (FK) `DOCTOR_ID`, menunjukkan bahwa setiap formulir registrasi terkait dengan satu dokter. Hubungan antara DOCTOR dan REG_FORM adalah Satu-ke-Banyak (1 ke N), artinya satu dokter dapat memiliki banyak formulir registrasi.

4. **REG_PATIENT:** Entitas ini berfungsi sebagai tabel penghubung (junction table) antara REG_FORM dan PATIENT. Entitas ini memiliki dua kunci asing (FK), yaitu `REG_NO` yang merujuk ke REG_FORM dan `PATIENT_ID` yang merujuk ke PATIENT. Hubungan antara REG_FORM dan REG_PATIENT adalah Satu-ke-Banyak (1 ke N), yang berarti satu formulir registrasi dapat melibatkan banyak pasien (atau satu baris registrasi dihubungkan ke banyak pasien melalui tabel ini). Demikian pula, hubungan antara PATIENT dan REG_PATIENT juga Satu-ke-Banyak (1 ke N), mengindikasikan bahwa satu pasien dapat terlibat dalam banyak registrasi.

Secara keseluruhan, ERD ini menggambarkan bagaimana informasi pasien dan dokter dihubungkan melalui formulir registrasi, memungkinkan pencarian data yang terintegrasi berdasarkan nomor registrasi, ID pasien, atau ID dokter.

Berikut merupakan kumpulan code dari controller Upload.php

```
public function searchData()
    {
        header('Content-Type: application/json');

        if (!$this->custom->isLogin()) {
            echo json_encode(['status' => 'error', 'message' => '
    Silakan login terlebih dahulu.']);
            return;
        }

        $reg_no = $this->input->post('reg_no');
        $patient_id = $this->input->post('patient_id');

        if (!$reg_no || !$patient_id) {
            echo json_encode(['status' => 'error', 'message' => '
    REG NO dan PATIENT ID wajib diisi.']);
            return;
        }

        $params = [
            'reg_no' => $reg_no,
            'patient_id' => $patient_id
        ];

        $result = $this->AttachmentModel->
    PKG_ATTACHMENT_APPS__BRW_DATA_BY_PATIENT_ID($params);
```

```
24
25          if (isset($result['DATA'][0]['CEK_STATUS']) && $result['
    DATA'][0]['CEK_STATUS'] === 'ERROR') {
26              echo json_encode(['status' => 'error', 'message' =>
    $result['DATA'][0]['MESSAGE']]);
27              return;
28          }
29
30          if (isset($result['DATA'][0])) {
31              $data = $result['DATA'][0];
32              echo json_encode(['status' => 'success', 'data' =>
    $data]);
33          } else {
34              echo json_encode(['status' => 'error', 'message' => '
    Data tidak ditemukan.']);
35          }
36      }
```

Kode 3.4: Fungsi SearchData() pada Controller Upload.php

Kode 3.4 menjunjukan fungsi `searchData()` dalam `Upload.php` Controller ini bertanggung jawab untuk menangani permintaan pencarian data yang dikirimkan melalui AJAX dan mengembalikan hasilnya dalam format JSON. Berikut adalah alur kerjanya:

1. **Pengaturan Header:** Fungsi ini terlebih dahulu mengatur ‘Content-Type‘ respons menjadi ‘application/json‘ agar klien dapat menginterpretasikan data yang dikembalikan sebagai JSON.

2. **Verifikasi Login:** Sistem melakukan pengecekan apakah pengguna sudah login menggunakan `custom->isLogin()`. Jika belum, pesan error JSON akan dikembalikan, meminta pengguna untuk login.

3. **Validasi Input:** Fungsi ini mengambil nilai ‘reg_no‘ dan ‘patient_id‘ dari data POST. Jika salah satu atau kedua input tersebut kosong, pesan error JSON akan dikembalikan, mengindikasikan bahwa kedua parameter tersebut wajib diisi.

4. **Panggilan Model:** Dengan input yang valid, fungsi akan memanggil ‘PKG_ATTACHMENT_APPS__BRW_DATA_BY_PATIENT_ID‘ pada ‘AttachmentModel‘. Ini adalah panggilan ke *web service* atau fungsi model

yang bertugas mengambil data dari database berdasarkan nomor registrasi dan ID pasien.

5. **Penanganan Hasil:**

    - Jika hasil dari model mengindikasikan status ‘ERROR‘ (`CEK_STATUS`), pesan error dari model akan diteruskan kembali ke klien.
    - Jika data ditemukan (`$result['DATA'][0]` ada), data pertama akan diambil dan dikembalikan dalam format JSON dengan status ‘success‘.
    - Jika data tidak ditemukan (`$result['DATA'][0]` tidak ada), pesan error JSON ”Data tidak ditemukan.” akan dikembalikan.

Secara ringkas, fungsi ini merupakan *endpoint* API yang memfasilitasi pencarian data pasien-dokter secara dinamis, lengkap dengan validasi input dan penanganan status login serta hasil dari *web service* database.

```
private function _get_next_file_sequence_with_type_id($ftp_conn, $directory, $outlet_id, $reg_no_clean, $doctor_id_clean, $file_type_id) {
        // Pattern baru: OUTLET_ID-REG_NO-DOCTOR_ID-FILE_TYPE_ID-SEQUENCE.EXT
        $pattern = "/^{$outlet_id}-{$reg_no_clean}-{$doctor_id_clean}-{$file_type_id}-(\\d{2})\\.\\w+$/i";

        $max_sequence = 0;
        $file_list = ftp_nlist($ftp_conn, $directory);

        if ($file_list === false) {
            log_message('error', 'FTP: Failed to list files in directory: ' . $directory . ' for pattern: ' . $pattern);
            return false;
        }

        foreach ($file_list as $file_name) {
            $base_file_name = basename($file_name);

            if (preg_match($pattern, $base_file_name, $matches)) {
                $sequence = (int)$matches[1]; // Ambil nomor urut (group 1 dari regex)
                if ($sequence > $max_sequence) {
                    $max_sequence = $sequence;
```

```
                }
            }
        }

        return $max_sequence + 1; // Nomor urut berikutnya
    }
```

Kode 3.5: Logic Auto Rename

Kode 3.5 menunjukan fungsi `_get_next_file_sequence_with_type_id()` adalah fungsi internal (ditandai dengan 'private') yang bertanggung jawab untuk menentukan nomor urut (*sequence*) berikutnya untuk file yang akan diunggah, berdasarkan pola penamaan yang spesifik. Fungsi ini digunakan dalam proses *auto rename file* sebelum penyimpanan ke FTP.

1. **Pola Penamaan File:** Fungsi ini mendefinisikan pola penamaan baru untuk file: `OUTLET_ID-REG_NO-DOCTOR_ID-FILE_TYPE_ID-SEQUENCE.EXT`. Pola ini menggunakan informasi dari `outlet_id`, nomor registrasi, ID dokter, ID tipe file, dan nomor urut dua digit.

2. **Daftar File FTP:** Fungsi ini terhubung ke direktori FTP yang ditentukan dan mengambil daftar semua file yang ada di dalamnya. Jika gagal, pesan error akan dicatat.

3. **Pencarian Nomor Urut Maksimal:** Fungsi akan mengiterasi setiap file dalam daftar yang diambil dari FTP. Untuk setiap file, nama dasar file (tanpa path) akan diperiksa apakah sesuai dengan pola penamaan yang telah ditentukan menggunakan ekspresi reguler (`preg_match`).

4. **Penentuan Urutan Berikutnya:** Jika nama file cocok dengan pola, nomor urut yang ada dalam nama file (yaitu, dua digit terakhir sebelum ekstensi) akan diekstrak. Fungsi akan melacak nomor urut tertinggi (`$max_sequence`) yang ditemukan.

```
public function download_attachments() {
        $reg_no = $this->input->get('reg_no'); // Menggunakan GET karena ini link unduh
        $patient_id = $this->input->get('patient_id');

```

```
5         if (empty($reg_no) || empty($patient_id)) {
6             // Tangani error jika parameter tidak lengkap
7             show_error('Invalid request. REG_NO or PATIENT_ID is
  missing.', 400);
8             return;
9         }
10
11        $files_to_zip = $this->Upload_model->get_files_by_patient(
  $reg_no, $patient_id);
12
13        if (!empty($files_to_zip)) {
14            foreach ($files_to_zip as $file) {
15                $file_path = $file->file_path; // Pastikan ini
  path absolut atau relatif yang benar dari server
16                $file_name_in_zip = $file->original_name; // Nama
  asli di dalam ZIP
17
18                if (file_exists($file_path)) {
19                    $this->zip->read_file($file_path, FALSE,
  $file_name_in_zip); // FALSE untuk path relatif, TRUE untuk
  path absolut
20                } else {
21                    // Opsional: Log error jika file tidak
  ditemukan di server
22                    log_message('error', 'File not found for
  zipping: ' . $file_path);
23                }
24            }
25
26            // Nama file ZIP
27            $zip_file_name = 'Lampiran_' . $reg_no . '_' .
  $patient_id . '.zip';
28
29            // Download file ZIP
30            $this->zip->download($zip_file_name);
31        } else {
32            // Tidak ada file untuk diunduh
33            show_error('No attachments found for the specified
  patient.', 404);
34        }
35    }
```

Kode 3.6: Fungsi download_attachments()

Kode 3.6 menunjukan fungsi `download_attachments()` bertanggung jawab untuk mengelola proses pengunduhan lampiran file terkait dengan registrasi dan pasien tertentu. Fungsi ini dirancang untuk memungkinkan pengguna mengunduh semua lampiran yang terkait dalam satu file ZIP.

1. **Pengambilan Parameter:** Fungsi ini mengambil parameter `reg_no` (nomor registrasi) dan `patient_id` (ID pasien) dari URL menggunakan metode `GET`.
2. **Validasi Parameter:** Memastikan bahwa kedua parameter (`reg_no` dan `patient_id`) tidak kosong. Jika ada yang hilang, akan ditampilkan pesan error `Invalid request`.
3. **Pengambilan Daftar File:** Fungsi memanggil model `Upload_model` (dengan asumsi ada) untuk mendapatkan daftar file yang terkait dengan pasien berdasarkan `reg_no` dan `patient_id`.
4. **Pembuatan dan Pengunduhan ZIP:**
   - Jika ada file yang ditemukan (`!empty($files_to_zip)`), fungsi akan mengiterasi setiap file. Untuk setiap file, jalur file (`file_path`) dan nama asli di dalam ZIP (`original_name`) diambil.
   - Fungsi akan memeriksa keberadaan file fisik di server (`file_exists($file_path)`). Jika file ada, akan ditambahkan ke dalam objek ZIP menggunakan `$this->zip->read_file()`. Jika file tidak ditemukan, error akan dicatat (opsional).
   - Setelah semua file ditambahkan, sebuah nama file ZIP akan dibuat (contoh: `Lampiran_REG_NO_PATIENT_ID.zip`).
   - Terakhir, file ZIP akan diunduh ke browser pengguna menggunakan `$this->zip->download()`.
5. **Penanganan File Tidak Ditemukan:** Jika tidak ada lampiran yang ditemukan untuk pasien yang ditentukan, pesan error `No attachments found` akan ditampilkan.

Secara ringkas, fungsi ini mengotomatisasi proses pengumpulan dan pengunduhan lampiran-lampiran terkait pasien menjadi satu file ZIP yang terorganisir.

Controller `Upload.php` pada aplikasi berfungsi sebagai *endpoint* utama untuk mengelola proses unggah berkas dari sisi klien (*frontend*) ke server. Ia menangani validasi, pemrosesan, dan penyimpanan berkas.

Fungsi Utama `doUpload()`

Metode inti dalam controller ini adalah `doUpload()`, yang bekerja sebagai berikut:

1. **Pengambilan Data Masukan:** Mengambil data yang dikirim melalui metode POST, meliputi nomor registrasi (`reg_no`), ID pasien (`patient_id`), ID dokter (`doctor_id`), dan ID tipe berkas (`file_type_id`).
2. **Validasi Data Masukan & Berkas:** Dilakukan serangkaian validasi untuk memastikan kelengkapan dan keabsahan data:
    - Memastikan `reg_no`, `patient_id`, dan `file_type_id` tidak kosong.
    - Memeriksa apakah ada berkas yang diunggah.
    - Memvalidasi jenis berkas (hanya PDF, JPG, JPEG, PNG yang diizinkan) dan ukurannya tidak melebihi batas (5MB).
3. **Koneksi FTP/SFTP:** Membangun koneksi ke server FTP atau SFTP yang ditentukan dalam konfigurasi untuk tujuan penyimpanan berkas. Jika koneksi gagal, proses dihentikan.
4. **Pemrosesan Berkas:** Untuk setiap berkas yang diunggah:
    - Jika berkas adalah gambar (JPG, JPEG, PNG), ia akan secara otomatis **dikonversi menjadi format PDF**.
    - Nama berkas akan **diganti secara otomatis** mengikuti format `OUTLET_ID-REG_NO-DOCTOR_ID-FILE_TYPE_ID-SEQUENCE.EXT`.
    - Berkas kemudian diunggah ke direktori spesifik di server FTP (berdasarkan wilayah outlet dan tanggal).
5. **Penyimpanan Metadata:** Setelah berhasil diunggah ke FTP, informasi atau metadata berkas (seperti nama berkas asli, nama berkas yang disimpan, jalur, ukuran, dsb.) akan **disimpan ke dalam database MySQL**.

6. **Respons:** Setelah semua berkas diproses, controller akan mengirimkan respons dalam format JSON kembali ke *frontend*, mengindikasikan status keseluruhan (berhasil, sebagian berhasil, atau gagal) beserta pesan atau daftar error jika ada.

F Halaman Setting Master Type File dan Integrasinya dalam Proses Unggah File

Halaman Setting Master Type File merupakan salah satu fitur administratif esensial yang hanya dapat diakses oleh akun pengguna dengan hak akses administrator. Fitur ini dirancang sebagai antarmuka pengelolaan daftar tipe file atau kategori dokumen yang dapat diunggah oleh pengguna aplikasi. Tujuan utama dari halaman ini adalah untuk menyediakan fleksibilitas dalam pengelompokan dokumen serta menjaga konsistensi sistem melalui validasi yang terstandarisasi, memastikan integritas data.

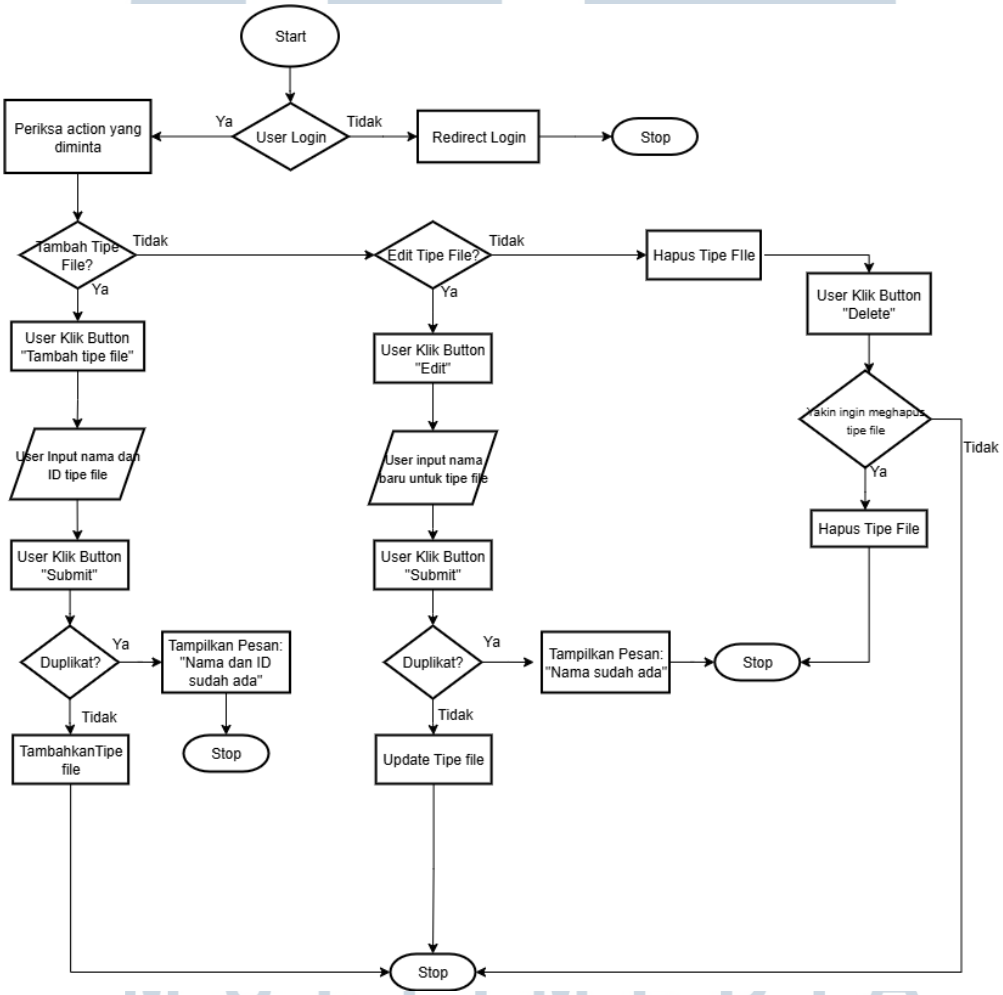
Alih-alih mendefinisikan jenis file secara statis di dalam kode program, administrator kini memiliki kemampuan untuk dengan mudah menambah, mengubah, atau menonaktifkan tipe file langsung melalui antarmuka ini. Setiap entri pada master tipe file umumnya memiliki sebuah ID unik dan nama deskriptif yang ditampilkan di seluruh sistem. Dengan pendekatan yang dinamis ini, proses pengelolaan klasifikasi file menjadi lebih efisien dan dapat disesuaikan dengan cepat sesuai kebutuhan organisasi tanpa harus memodifikasi ulang struktur program yang mendasar.

Fungsi utama yang tersedia pada halaman ini mencakup beberapa operasi penting:

- **Penambahan Tipe File Baru:** Administrator dapat menambahkan kategori dokumen baru, seperti ”Hasil Radiologi”, ”Resep Dokter”, atau ”Surat Rujukan”, yang memperkaya jenis lampiran yang bisa diunggah.
- **Pengeditan Tipe File yang Sudah Ada:** Memungkinkan perubahan pada nama tipe file yang sudah terdaftar, sehingga administrator dapat mengoreksi atau memperbarui deskripsi sesuai kebutuhan.
- **Penonaktifan Entri Tipe File:** Memberikan opsi untuk menonaktifkan tipe file tertentu tanpa harus menghapus data terkait secara permanen dari sistem, menjaga riwayat data.

- **Penampilan Daftar Tipe File:** Menyajikan seluruh daftar tipe file dalam format tabel yang terorganisir, lengkap dengan kolom ID, Nama Tipe, dan Status untuk memudahkan tinjauan dan pengelolaan.

Data dari modul ini secara otomatis digunakan untuk mengisi opsi *dropdown* pada *form* unggah file di halaman Upload. Hal ini menciptakan keterkaitan langsung antara pengaturan administratif dan pengalaman pengguna akhir, serta memastikan bahwa seluruh file yang diunggah telah dikategorikan sesuai dengan tipe yang telah ditentukan sebelumnya oleh administrator.



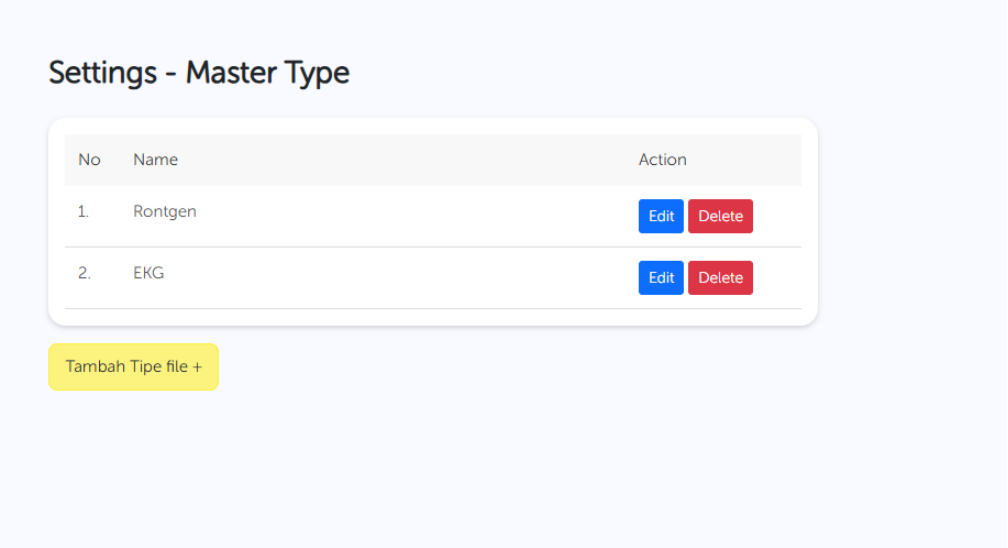
Gambar 3.15. Flow Setting Master Type File

Gambar 3.15 menunjukan alur dimulai dengan memeriksa aksi yang diminta oleh pengguna. Pertama, sistem memverifikasi status login pengguna. Jika tidak login, pengguna akan dialihkan ke halaman login dan proses berhenti.

Jika pengguna sudah login, alur akan bercabang berdasarkan aksi yang dipilih:

1. **Tambah Tipe File:**
    - Pengguna mengklik tombol ”Tambah tipe file”.
    - Pengguna menginput nama dan ID tipe file baru.
    - Pengguna mengklik tombol ”Submit”.
    - Sistem memeriksa apakah nama dan ID tipe file sudah ada (duplikat). Jika ya, pesan ”Nama dan ID Sudah ada” akan ditampilkan dan proses berhenti.
    - Jika tidak duplikat, tipe file baru akan ditambahkan, dan proses berhenti.
2. **Edit Tipe File:**
    - Pengguna mengklik tombol ”Edit”.
    - Pengguna menginput nama baru untuk tipe file yang akan diedit.
    - Pengguna mengklik tombol ”Submit”.
    - Sistem memeriksa apakah nama tipe file baru sudah ada (duplikat). Jika ya, pesan ”Nama sudah ada” akan ditampilkan dan proses berhenti.
    - Jika tidak duplikat, tipe file akan diperbarui, dan proses berhenti.
3. **Hapus Tipe File:**
    - Pengguna mengklik tombol ”Delete”.
    - Sistem akan meminta konfirmasi (”Yakin ingin menghapus tipe file?”). Jika tidak dikonfirmasi, proses akan berhenti.
    - Jika dikonfirmasi, tipe file akan dihapus, dan proses berhenti.

Flowchart ini secara komprehensif menjelaskan bagaimana aplikasi mengelola operasi CRUD (Create, Read, Update, Delete) untuk data tipe file, dengan validasi dan konfirmasi yang sesuai.

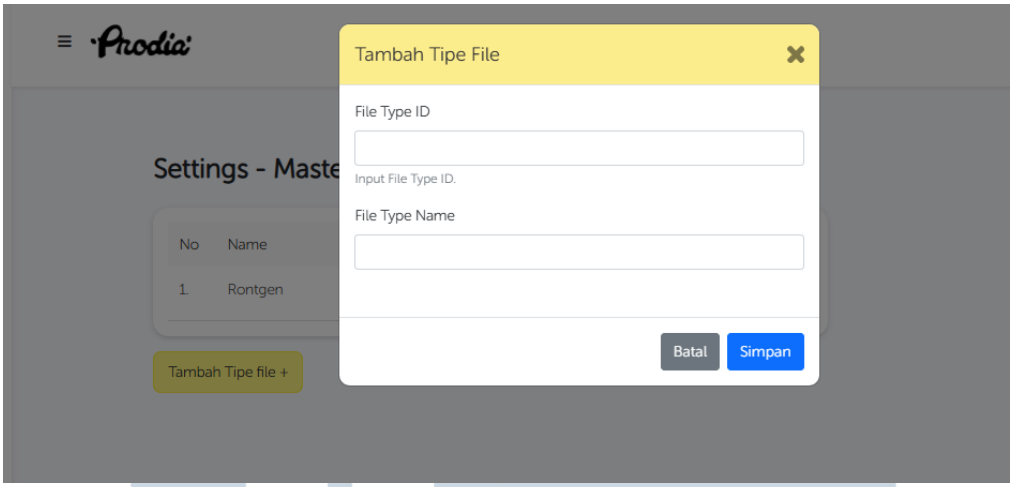


Gambar 3.16. Halaman Setting Master Type File

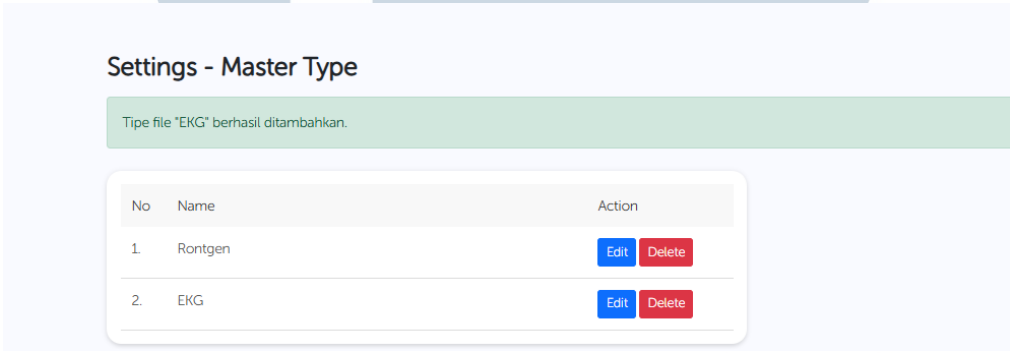
Gambar 3.16 Halaman ini menyajikan tabel dengan kolom ”Name”, dan ”Action”. Kolom ”Name” menampilkan nama-nama tipe file yang sudah terdaftar, seperti ”Rontgen” dan ”EKG”. Kolom ”Action” menyediakan dua tombol untuk setiap baris:

- Tombol **Edit** digunakan untuk mengubah nama atau detail tipe file yang sudah ada.
- Tombol **Delete** digunakan untuk menghapus tipe file dari daftar.

Terdapat tombol **Tambah Tipe file** yang berfungsi untuk menambahkan tipe file baru ke dalam sistem. Antarmuka ini dirancang untuk memberikan kemudahan bagi administrator atau pengguna yang berwenang untuk menambah, mengedit, atau menghapus tipe-tipe file yang valid dalam aplikasi.



Gambar 3.17. Pop Up untuk menambahkan tipe file



Gambar 3.18. File Berhasil Ditambahkan

Gambar 3.17 menunjukkan sebuah modal pop-up yang muncul secara dinamis setelah pengguna mengklik tombol ”Tambah Tipe File”. Antarmuka ini dirancang untuk memfasilitasi penambahan tipe file baru ke dalam sistem, menyediakan form yang intuitif bagi pengguna.

Pada pop-up tersebut, pengguna akan diminta untuk mengisi dua informasi krusial. Pertama, mereka harus memasukkan `File Type ID` yang akan berfungsi sebagai pengenal unik untuk tipe file tersebut dalam database. Kedua, pengguna perlu menginput `File Type Name` yang akan menjadi nama deskriptif untuk tipe file yang baru ditambahkan, memudahkan identifikasi di kemudian hari.

Setelah semua informasi yang diperlukan terisi, dengan menekan tombol ”Simpan”, sistem akan memproses dan mengirimkan data input tersebut untuk disimpan ke dalam database. Data yang berhasil diinput oleh pengguna akan secara resmi menjadi tipe file yang baru dan tersedia untuk digunakan di seluruh aplikasi.

Sebagai konfirmasi keberhasilan operasi, sistem akan menampilkan pesan notifikasi ”Tipe File berhasil ditambahkan” kepada pengguna, seperti yang dapat dilihat pada gambar 3.18.

F.1 Hubungan dan Integrasi Tipe File dalam Proses Unggah

Hubungan antara pengaturan master tipe file dan proses unggah file dijalankan secara menyeluruh mulai dari lapisan antarmuka pengguna (frontend), logika aplikasi (backend), hingga penyimpanan metadata di basis data. Penjelasan integrasinya adalah sebagai berikut:

- **Validasi di Sisi Klien (Frontend)** Saat pengguna memilih file dan menentukan tipe file melalui dropdown pada form upload, sistem secara otomatis akan melakukan validasi awal menggunakan JavaScript. Validasi ini mengecek apakah ekstensi file yang diunggah sesuai dengan tipe file yang dipilih. Apabila tidak sesuai, proses unggah akan dibatalkan sebelum data dikirim ke server. Hal ini tidak hanya meningkatkan kenyamanan pengguna, tetapi juga menghemat bandwidth dan waktu pemrosesan.

- **Validasi di Sisi Server (Backend)**
  Setelah data diterima oleh server, kontroler PHP akan menjalankan validasi lanjutan yang lebih ketat. Ini termasuk memastikan bahwa ID tipe file yang dikirim benar-benar valid dan terdaftar di sistem master. Selain itu, sistem juga melakukan pengecekan keamanan terhadap format dan isi file untuk mencegah unggahan berkas yang berpotensi berbahaya.

- **Konversi Otomatis File ke PDF**
  Salah satu fitur tambahan dari sistem adalah kemampuan untuk mengonversi file gambar (seperti .jpg dan .png) menjadi format PDF secara otomatis. Proses ini dijalankan berdasarkan deteksi jenis file, sehingga dokumen yang diunggah menjadi seragam dalam format dan memudahkan proses pencarian serta arsip.

- **Penamaan File Secara Otomatis**
  Setelah validasi berhasil dan sebelum file disimpan, sistem akan memberikan nama baru kepada file berdasarkan skema tertentu, seperti: `OUTLETID-REGNO-DOCTORID-FILETYPEID-XX.pdf`. Dengan skema ini, tipe

file menjadi bagian dari identitas file, yang membantu pengelolaan file di dalam server FTP dan memudahkan proses debugging jika diperlukan.

- **Penyimpanan Metadata ke Database MySQL**
  Setelah file berhasil diunggah ke FTP, sistem akan menyimpan data pendukung (metadata) ke dalam basis data lokal. Data ini mencakup informasi seperti nama asli file, nama baru setelah diubah, path direktori penyimpanan, waktu unggah, dan tipe file berdasarkan ID dari master. Dengan ini, sistem dapat menyediakan fitur pencarian atau filter berdasarkan kategori dokumen.

- **Tampilan di UI (User Interface)**
  File yang telah berhasil diunggah akan langsung muncul di daftar unggahan pada halaman pengguna. Informasi tentang tipe file juga ditampilkan di sini agar pengguna mengetahui dengan jelas jenis dokumen yang mereka unggah, memperkuat transparansi dan akuntabilitas sistem.

G Manajemen Pengguna dan Grup (User & Group Management)

Fitur **Manajemen Pengguna dan Grup** merupakan bagian dari modul administratif dalam aplikasi HPSL Attachment yang hanya dapat diakses oleh pengguna dengan hak akses administrator. Fitur ini berfungsi untuk mengelola akun pengguna, hak akses, serta struktur grup pengguna yang digunakan dalam pengaturan otorisasi sistem.

Fungsi-fungsi utama dalam modul ini dikembangkan pada controller `User.php`, dan terhubung langsung dengan database Oracle melalui prosedur yang dipanggil di model `AttachmentModel`. Modul ini dibangun untuk mendukung pengelolaan akun berbasis LDAP Active Directory yang telah digunakan oleh perusahaan.

G.1 Pengelolaan Grup Pengguna

Halaman ini memungkinkan administrator untuk melihat, menambahkan, mengubah, dan menghapus data grup pengguna yang disimpan di sistem. Grup pengguna memiliki peran penting karena menentukan hak akses terhadap menu dan fitur di dalam aplikasi.

- **Penambahan Grup Baru:** Melalui fungsi `tambahGroup()`, administrator dapat menambahkan grup baru ke sistem dan langsung mengatur hak aksesnya ke menu tertentu.
- **Pengeditan Grup:** Fungsi `ubahGroup()` memungkinkan perubahan nama grup dan menu yang terkait. Saat melakukan perubahan, sistem akan menghapus seluruh menu lama dari grup tersebut dan menyimpan kembali hak akses terbaru.
- **Penghapusan Grup:** Fungsi `deleteGroup()` memungkinkan penghapusan grup yang tidak lagi digunakan, dengan pengecekan terlebih dahulu apakah grup masih memiliki pengguna aktif.

G.2 Manajemen Akun Pengguna

Fitur ini digunakan untuk mengelola akun pengguna sistem, baik untuk admin pusat maupun pengguna cabang. Semua proses penambahan dan pembaruan akun dilakukan dengan otentikasi dan validasi terhadap Active Directory perusahaan.

- **Pencarian Data User dari Active Directory:** Sistem memungkinkan pencarian pengguna berdasarkan NIK melalui LDAP, dan mengambil data nama, email, dan jabatan yang ditampilkan di form input.
- **Penambahan User Baru:** Fungsi `tambahUser()` menangani penyimpanan akun baru ke tabel `WEBAPP_USER`. Pengguna cabang harus memilih outlet, dan akan disimpan juga ke dalam tabel `WEBAPP_USER_OUTLET`.
- **Pengubahan Data User:** Fungsi `ubahUser()` digunakan untuk mengubah data pengguna dan relasi grup atau outlet-nya.
- **Penghapusan User:** Dengan fungsi `deleteUser()`, administrator dapat menghapus akun user dari sistem, dengan validasi tambahan untuk mencegah penghapusan akun milik sendiri.

G.3 Validasi dan Hak Akses

Setiap aksi pada modul ini telah dilengkapi dengan proses validasi yang dilakukan melalui form helper bawaan CodeIgniter. Selain itu, pengecekan

hak akses menu dilakukan melalui middleware `custom->isAccess()`, yang memastikan bahwa pengguna hanya dapat mengakses fitur sesuai peran dan grup-nya.

G.4 Integrasi dengan Fitur Lain

Modul User ini memiliki keterkaitan erat dengan fitur `Upload`, di mana hanya pengguna dari grup cabang tertentu yang diperbolehkan mengunggah file. Data outlet yang terkait dengan pengguna juga menjadi parameter penting saat menyimpan file ke folder FTP dan mencatat metadata file ke database MySQL. Modul ini juga mendukung sistem login yang dibangun dengan LDAP, serta menjamin fleksibilitas dalam pengaturan otorisasi sistem secara terpusat.

Dengan adanya fitur manajemen pengguna dan grup ini, administrator sistem memiliki kontrol penuh terhadap siapa saja yang dapat mengakses aplikasi, fitur apa yang bisa digunakan oleh masing-masing grup, serta bagaimana data operasional seperti file upload dikaitkan dengan pengguna yang berwenang.

```
1  // Potongan kode tambahUser()
2  public function tambahUser()
3    {
4      // Check login
5      if ( !$this->custom->isLogin() ) {
6        $this->session->set_flashdata('error', 'Maaf, Silahkan login
       terlebih dahulu');
7        redirect('login');
8      }
9
10     // Check menu
11     $menu_id = '91'; // menu id untuk user
12     if ( !$this->custom->isAccess($menu_id) ) {
13       // Pemanggilan View
14       $data['content'] = "page/not_found";
15       $data['title'] = "User";
16       $this->load->view('template/page_navbar', $data);
17       return;
18     }
19
20     // Check Form Validation
21     $this->form_validation->set_rules($this->rulesTambahUser());
22     if ($this->form_validation->run() == FALSE) {
23       $msg_errors = $this->custom->validationError($this->
```

```
    form_validation->error_array());
24      $this->session->set_flashdata('error', $msg_errors);
25      redirect('list-user');
26    }
27
28    // Jika memilih cabang, outlet tidak boleh kosong
29    if ( $this->input->post('group') == 'G10029' && empty($this->
    input->post('cabang')) ) {
30      $this->session->set_flashdata('error', 'Maaf, Outlet tidak
    boleh kosong');
31      redirect('list-user');
32    }
33
34    // Jika memilih cabang, maka akan insert ke WEBAPP_USER_OUTLET
35    if ( $this->input->post('group') == 'G10029' ) {
36      $param = array(
37        'action'    => 'I',
38        'app_id'    => $this->config->item('webapp_id'),
39        'group_id'    => $this->input->post('group'),
40        'user_id'   => $this->input->post('username'),
41        'nik'     => $this->session->userdata('nik')
42      );
43
44      // Insert ke tabel WEBAPP_USER_GROUP_APP
45      $result = $this->AttachmentModel->
    PKG_WEBAPP_MENU__IUD_WEBAPP_USER_GROUP_APP($param);
46      if ( @$result['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
47        $err = array(
48          'error'   => false,
49          'message' => 'Maaf, Terdapat error pada aplikasi
    silahkan hubungi admin ['. substr($result['DATA'][0]['MESSAGE'
    ], 0, 9) .']'
50        );
51      }
52
53      foreach ( $this->input->post('cabang') as $key => $value) {
54        $param3 = array(
55          'action'    => 'I',
56          'user_id'   => $this->input->post('username'),
57          'outelt_id'   => $value,
58          'user_update' => $this->session->userdata('nik'),
59          'app_id'    => $this->config->item('webapp_id')
60        );
```

```
61
62        // Insert ke tabel WEBAPP_USER_OUTLET
63        $result3 = $this->AttachmentModel->
PKG_WEBAPP_MENU__IUD_WEBAPP_USER_OUTLET($param3);
64      }
65    } else {
66      $param = array(
67        'action'    => 'I',
68        'app_id'    => $this->config->item('webapp_id'),
69        'group_id'    => $this->input->post('group'),
70        'user_id'   => $this->input->post('username'),
71        'nik'     => $this->session->userdata('nik')
72      );
73
74      // Insert ke tabel WEBAPP_USER_GROUP_APP
75      $result = $this->AttachmentModel->
PKG_WEBAPP_MENU__IUD_WEBAPP_USER_GROUP_APP($param);
76      if ( @$result['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
77        $err = array(
78          'error'   => false,
79          'message' => 'Maaf, Terdapat error pada aplikasi
silahkan hubungi admin ['. substr($result['DATA'][0]['MESSAGE'
], 0, 9) .']'
80        );
81      }
82    }
83
84    $param2 = array(
85      'action'    => 'I',
86      'user_id'   => $this->input->post('username'),
87      'name'      => $this->config->item('name'),
88      'nik'     => $this->session->userdata('nik')
89    );
90
91    // Insert ke tabel WEBAPP_USER
92    $result2 = $this->AttachmentModel->
PKG_WEBAPP_MENU__IUD_WEBAPP_USER($param2);
93    if ( $result2['DATA'][0]['CEK_STATUS'] == 'ERROR' ) {
94      $err2 = array(
95        'error'   => false,
96        'message' => 'Maaf, Terdapat error pada aplikasi silahkan
hubungi admin ['. substr($result2['DATA'][0]['MESSAGE'], 0, 9)
.']'
```

```
97        );
98      }
99
100     if ( $err['error'] === false && $err2['error'] === false ) {
101       $this->session->set_flashdata('error', $err['message']);
102       $this->session->set_flashdata('error', $err2['message']);
103       redirect('list-user');
104     }
105
106
107     $this->session->set_flashdata('success', 'Anda telah berhasil
        menambahkan user');
108     redirect('list-user');
109   }
```

Kode 3.7: Fungsi tambahUser() pada Controller User.php

Kode 3.7 menampilkan fungsi `tambahUser()` dalam `User.php` Controller bertanggung jawab untuk menambahkan pengguna baru ke dalam sistem. Alur kerjanya melibatkan beberapa tahapan validasi dan interaksi dengan model untuk penyimpanan data ke database.

1. **Verifikasi Login dan Akses Menu:** Fungsi ini pertama-tama memeriksa apakah pengguna yang sedang mengakses sudah login. Jika belum, pengguna akan dialihkan ke halaman login. Selain itu, fungsi ini juga memverifikasi hak akses pengguna ke menu ”User” (dengan `menu_id` 91); jika tidak memiliki akses, halaman ”not_found” akan ditampilkan.

2. **Validasi Form Input:** Sistem menerapkan aturan validasi form menggunakan `form_validation->set_rules()` dan `form_validation->run()`. Jika ada kesalahan validasi (misalnya, input tidak lengkap atau tidak sesuai format), pesan error akan disimpan di *flashdata* sesi, dan pengguna dialihkan kembali ke halaman ‘list-user‘.

3. **Validasi Khusus Grup Cabang:** Jika grup yang dipilih adalah ’G10029’ (identifikasi untuk grup cabang), fungsi ini memeriksa apakah input ’cabang’ kosong. Jika ya, pesan error ”Outlet tidak boleh kosong” akan ditampilkan, dan pengguna dialihkan kembali.

4. **Penyimpanan Data Pengguna:**

- **Untuk Grup Cabang ('G10029'):** Data pengguna (ID aplikasi, ID grup, ID pengguna baru, dan NIK pengguna yang melakukan input) akan dimasukkan ke tabel `WEBAPP_USER_GROUP_APP` melalui model `AttachmentModel->PKG_WEBAPP_MENU__IUD_WEBAPP_USER_GROUP_APP()`. Jika ada error, akan dicatat. Selain itu, untuk setiap cabang yang dipilih, data akan dimasukkan ke tabel `WEBAPP_USER_OUTLET` melalui `PKG_WEBAPP_MENU__IUD_WEBAPP_USER_OUTLET()`.
- **Untuk Grup Selain Cabang:** Hanya data pengguna utama yang dimasukkan ke tabel `WEBAPP_USER_GROUP_APP`. Penanganan error serupa juga diterapkan.

5. **Penyimpanan Data User Utama:** Data pengguna (ID pengguna baru, nama, dan NIK pengguna yang melakukan input) kemudian dimasukkan ke tabel `WEBAPP_USER` melalui `AttachmentModel->PKG_WEBAPP_MENU__IUD_WEBAPP_USER()`.
6. **Penanganan Error Umum dan Redirect:** Setelah semua operasi penyimpanan, fungsi akan memeriksa apakah ada error yang terjadi selama proses insert ke tabel `WEBAPP_USER_GROUP_APP` dan `WEBAPP_USER`. Jika ada, pesan error akan ditampilkan. Jika semua operasi berhasil, pesan sukses "Anda telah berhasil menambahkan user" akan ditampilkan, dan pengguna dialihkan ke halaman 'list-user'.

Secara keseluruhan, fungsi ini mengelola penambahan pengguna baru secara komprehensif, mulai dari validasi input, penanganan hak akses, hingga interaksi dengan database untuk menyimpan data pengguna beserta asosiasi grup dan cabangnya.

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala yang Ditemukan

Selama pelaksanaan kerja praktik magang di PT. Prodia Widyusaha, mahasiswa menghadapi sejumlah kendala teknis dan non-teknis dalam proses pengembangan aplikasi HPSL Attachment. Beberapa kendala utama yang ditemukan antara lain:

1. **Keterlambatan Akses Sistem dan Akun Internal**
   Pada awal magang, mahasiswa belum memiliki akun resmi untuk mengakses

infrastruktur pengembangan seperti server FTP, SOAP Web Service, dan Oracle Database, sehingga proses pengembangan aktif baru dimulai pada pertengahan April 2025.

2. **Keterbatasan Infrastruktur untuk Pengujian FTP dan SOAP**
   Mahasiswa kesulitan melakukan pengujian fitur koneksi ke FTP dan Web Service Oracle saat bekerja dari rumah karena terbatasnya akses jaringan ke server internal perusahaan.

3. **Masalah dalam Konversi Gambar ke PDF**
   Proses konversi file berformat gambar (JPG/PNG) ke PDF sempat mengalami kegagalan dalam pembacaan file dan menghasilkan file kosong.

4. **Adaptasi terhadap Framework CodeIgniter 3**
   Mahasiswa awalnya merancang sistem menggunakan Laravel, namun kemudian harus menyesuaikan dengan CodeIgniter 3, yang memiliki keterbatasan fitur dan pendekatan pengembangan yang berbeda.

5. **Minimnya Dokumentasi Sistem Sebelumnya**
   Tidak semua fitur atau prosedur sistem lama terdokumentasi dengan jelas, sehingga mahasiswa mengalami kesulitan dalam memahami alur login, validasi user, dan proses integrasi outlet.

3.4.2 Solusi

Untuk mengatasi kendala-kendala tersebut, mahasiswa menerapkan beberapa pendekatan dan solusi sebagai berikut:

1. **Pemanfaatan Waktu untuk Desain Awal**
   Sambil menunggu aktivasi akun internal, mahasiswa mulai merancang struktur halaman dan tampilan antarmuka pengguna secara offline menggunakan Figma serta Bootstrap.

2. **Simulasi Lingkungan Lokal**
   Mahasiswa membuat simulasi koneksi FTP dan endpoint SOAP lokal agar pengembangan tetap dapat dilakukan saat WFH. Pengujian sistem secara menyeluruh dilakukan kembali saat berada di kantor (WFO).

3. **Penggunaan Library Eksternal untuk Konversi PDF**
   Mahasiswa mengganti metode konversi dengan library eksternal yang lebih

stabil dan mendukung pengolahan file gambar ke format PDF secara konsisten.

4. **Refactoring Modular pada CodeIgniter 3**
   Mahasiswa menyesuaikan diri dengan pola MVC pada CodeIgniter 3 dan menulis kode secara modular untuk memastikan sistem tetap scalable dan mudah dipelihara.

5. **Diskusi Rutin dan Analisis Mandiri**
   Mahasiswa melakukan diskusi mingguan dengan supervisor untuk memahami alur prosedural sistem lama, serta membaca langsung isi database dan struktur SOAP response untuk memahami konteks integrasi sistem.