4. IMPLEMENTASI SISTEM

4.1 Aplikasi Pemrograman

Aplikasi pemrograman digunakan untuk menunjang proses implementasi desain sistem dan pembuatan *website* Sistem Informasi Manajemen Tugas Akhir adalah sebagai berikut:

i. Aplikasi Visual Studio Code versi 1.90.0 digunakan sebagai *code editor* dengan berbagai ekstensi untuk mendukung proses pemrograman *website* menggunakan *framework* Laravel. Beberapa ekstensi yang digunakan antara lain Laravel Artisan, Laravel Blade Formatter, Laravel Blade Snippets, Laravel Blade Wrapper, Laravel Extra Intellisense, Laravel Snippets, dan Laravel Goto View.

ii. Aplikasi Google Chrome digunakan sebagai media untuk menjalankan kode program untuk membuat *website* serta melakukan *debugging* program menggunakan fitur *inspect*.

iii. Aplikasi Github Desktop digunakan untuk mempermudah dalam melakukan *commit* perubahan program, serta melakukan *pull* dan *push* perubahan pada proyek yang terhubung di GitLab.

iv. Aplikasi Laragon digunakan untuk membantu dalam menjalankan proyek Laravel secara lokal seperti menyediakan *web server*, *database server* dan lainnya.

v. Aplikasi EasyConnect digunakan untuk koneksi dengan VPN yang disediakan oleh PPSI agar dapat mengakses proyek pada GitLab, menjalankan program di *browser* dan mengakses *database*.

4.2 Implementasi Program

Pada sub bab ini akan dijelaskan implementasi program dari desain sistem yang telah dibuat untuk mengembangkan *website* Sistem Informasi Manajemen Tugas Akhir dengan *framework* Laravel. Implementasi ini mencakup berbagai langkah teknis yang diambil untuk merealisasikan desain menjadi *website* yang fungsional. Tabel 4.1 menunjukkan daftar segmen program digunakan untuk implementasi program beserta pemetaan terhadap desain sistem yang dibuat.

Tabel 4.1

Pemetaan Segmen Program

Segmen Program	**Implementasi**	**Desain Sistem**
4.1	Pembagian Menu *Sidebar* Berdasarkan Hak Akses	-
4.2		Gambar 3.6
4.3		Gambar 3.7
4.4		Gambar 3.8
4.5		Gambar 3.5
4.6	Pembuatan Menu	Semua *Activity Diagram* (Gambar 3.12 - Gambar 3.31)
4.7		
4.8		
4.9	*FormHelper*	
4.10	Pembuatan Model Laravel	Gambar 3.9, Gambar 3.10, dan Gambar 3.11
4.11	Metode *pageProperties* pada *Class Controller*	Semua *Activity Diagram* (Gambar 3.12 - Gambar 3.31)
4.12	Metode *moduleGlobal* pada *Class GlobalVariable*	
4.13	Inisialisasi Variabel-variabel pada *Controller*	
4.14	Metode *Index* pada *Controller*	Semua *Activity Diagram* (Gambar 3.12- Gambar 3.31)
4.15	Metode *Create* dan *Store* pada *Controller*	Gambar 3.53
4.16	Metode *Show* pada *Controller*	Gambar 3.50
4.17	Metode *Edit* dan *Update* pada *Controller*	Gambar 3.53
4.18	Metode *Destroy* pada *Controller*	-
4.19	Metode *Filter* pada *Controller*	Gambar 3.26
4.20	Metode *berkasUpload* pada *FormBerkasDataController*	Gambar 3.20

4.21	Metode *fileTugasAkhirUpload* pada *FormBerkasDataController*	
4.22 4.23	Metode *downloadBerkas* dan *downloadFileTugasAkhir* pada *FormBerkasDataController*	Gambar 3.51
4.24 4.25	Metode *ajukanMajuSidang* dan *ajukanPerpanjang* pada *TugasAkhirController*	Gambar 3.14
4.26 4.27 4.28	Metode *batalPeserta, batalPembimbing,* dan *batalKoordinator* pada *TugasAkhirController*	-
4.29	Metode *approvePengajuan* pada *TugasAkhirController*	-
4.30 4.31	Metode *updatePembimbing* dan *updatePeserta* pada *TugasAkhirController*	Gambar 3.24
4.32	Metode *getUnitIdAttribute* pada Model *Unit*	-
4.33	Metode *generatePDF*	Gambar 3.26 dan Gambar 3.27

4.2.1 Pembagian Menu *Sidebar* Berdasarkan Hak Akses

Pada *website* ini terdapat empat hak akses yang bisa digunakan oleh pengguna, yaitu PIC, Koordinator, Dosen, dan Mahasiswa. Setiap hak akses memiliki daftar menu di *sidebar* yang berbeda-beda. Segmen Program 4.1 menunjukkan bahwa untuk mendapatkan hak akses pengguna saat ini, nilai diambil dari *session current_role* kemudian disimpan ke variabel *$role*. Variabel tersebut kemudian digunakan untuk menentukan tampilan menu mana yang harus ditampilkan pada *sidebar*. Laravel *@include* digunakan untuk menyisipkan tampilan *(view)* ke dalam tampilan yang sedang diproses, dalam hal ini adalah tampilan *sidebar*. Pemisahan tampilan menu menggunakan Laravel *@include* mempermudah pembuatan dan pengelolaan menu untuk setiap hak akses.

Segmen Program 4.1 Menu pada *Sidebar* Secara Keseluruhan

```
<!-- Main Sidebar Container -->
```

```
<aside class="main-sidebar elevation-2 sidebar-light-primary">
    <!-- Brand Logo -->
    <a href="{{ route('home') }}" class="brand-link" style="text-
align:center;border-bottom:3px #f8ad3d solid;">
        <img class="dr" src="{{ asset('img/logo.png') }}"
alt="{{ config('app.name') }}" style="width: 90%;">
        <img class="dr d-none" src="{{ asset('img/logo-
light.png') }}" alt="{{ config('app.name') }}" style="width: 90%;">
    </a>

    @php
        $role = strtolower(Session::get('current_role'));
    @endphp

    <!-- Sidebar -->
    <div class="sidebar">
        <!-- Sidebar Menu -->
        <nav class="mt-2 sidebar-petra">
            @if ($role === 'mahasiswa')
            @include('layouts.partials.menus.menu_mahasiswa')
            @elseif ($role === 'koordinator')
            @include('layouts.partials.menus.menu_koordinator')
            @elseif ($role === 'dosen')
            @include('layouts.partials.menus.menu_dosen')
            @elseif ($role === 'pic')
            @include('layouts.partials.menus.menu_pic')
            @endif
        </nav>
        <!-- /.sidebar-menu -->
    </div>
    <!-- /.sidebar -->
</aside>
```

Segmen Program 4.2 mendefinisikan struktur menu dari hak akses koordinator. Menu utama mencakup tautan langsung ke halaman beranda dan pengumuman. Selain itu, terdapat menu *dropdown* untuk "Referensi" yang menyediakan akses ke form berkas, bidang studi, dan jenis sidang. Menu *dropdown* lainnya untuk "Tugas Akhir" memberikan akses ke halaman periode tugas akhir, pengajuan tugas akhir, vakasi sidang, SKS bimbingan, dan laporan.

Segmen Program 4.2 Menu *Sidebar* Hak Akses Koordinator

```
<ul class="nav nav-pills nav-sidebar flex-column" data-
widget="treeview" role="menu" data-accordion="false">
    <li class="nav-item">
        <a href="{{ route('beranda.index') }}" class="nav-link">
            <i class="fas fa-home nav-icon"></i>
            <p>Beranda</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('pengumuman.index') }}" class="nav-link">
            <i class="fas fa-bullhorn nav-icon"></i>
            <p>Pengumuman</p>
        </a>
```

```
    </li>
    <li class="nav-item has-treeview">
        <a href="#" class="nav-link">
            <i class="nav-icon fas fa-plus-square"></i>
            <p>
                Referensi
                <i class="right fas fa-angle-left"></i>
            </p>
        </a>
        <ul class="nav nav-treeview ml-2">
            <li class="nav-item">
                <a href="{{ route('form-berkas.index') }}"
class="nav-link">
                    <i class="nav-icon fas fa-folder-open"></i>
                    <p>Form Berkas</p>
                </a>
            </li>
            <li class="nav-item">
                <a href="{{ route('bidang-studi.index') }}"
class="nav-link">
                    <i class="nav-icon fas fa-school"></i>
                    <p>Bidang Studi</p>
                </a>
            </li>
            <li class="nav-item">
                <a href="{{ route('jenis-sidang.index') }}"
class="nav-link">
                    <i class="nav-icon fas fa-chalkboard-
teacher"></i>
                    <p>Jenis Sidang</p>
                </a>
            </li>
        </ul>
    </li>
    <li class="nav-item has-treeview">
        <a href="#" class="nav-link">
            <i class="nav-icon fas fa-book-open"></i>
            <p>
                Tugas Akhir
                <i class="right fas fa-angle-left"></i>
            </p>
        </a>
        <ul class="nav nav-treeview ml-2">
            <li class="nav-item">
                <a href="{{ route('periode.index') }}" class="nav-
link">
                    <i class="nav-icon fas fa-calendar"></i>
                    <p>Periode Tugas Akhir</p>
                </a>
            </li>
            <li class="nav-item">
                <a href="{{ route('pengajuan-tugas-akhir.index') }}"
class="nav-link">
                    <i class="nav-icon fas fa-book"></i>
                    <p>Pengajuan Tugas Akhir</p>
                </a>
            </li>
            <li class="nav-item">
```

```
                <a href="{{ route('vakasi.index') }}" class="nav-
link">
                    <i class="nav-icon fas fa-money-bill"></i>
                    <p>Vakasi Sidang</p>
                </a>
            </li>
            <li class="nav-item">
                <a href="{{ route('sks-bimbingan.index') }}"
class="nav-link">
                    <i class="nav-icon fas fa-users"></i>
                    <p>SKS Bimbingan</p>
                </a>
            </li>
            <li class="nav-item">
                <a href="{{ route('laporan.index') }}" class="nav-
link">
                    <i class="fas fa-clipboard nav-icon"></i>
                    <p>Laporan</p>
                </a>
            </li>
        </ul>
    </li>
</ul>
```

Segmen Program 4.3 mendefinisikan struktur menu dari hak akses dosen. Menu-menu yang dapat diakses dari *sidebar* yakni halaman beranda, pengumuman, pengajuan bimbingan tugas akhir, daftar mahasiswa yang sedang dalam bimbingan, dan daftar mahasiswa yang akan mengikuti ujian sidang.

Segmen Program 4.3 Menu *Sidebar* Hak Akses Dosen

```
<ul class="nav nav-pills nav-sidebar flex-column" data-
widget="treeview" role="menu" data-accordion="false">
    <li class="nav-item">
        <a href="/" class="nav-link">
            <i class="fas fa-home nav-icon"></i>
            <p>Beranda</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('pengumuman.index') }}" class="nav-link">
            <i class="fas fa-bullhorn nav-icon"></i>
            <p>Pengumuman</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('pengajuan-tugas-akhir.index') }}"
class="nav-link">
            <i class="fas fa-book nav-icon"></i>
            <p>Pengajuan Bimbingan</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('mahasiswa-bimbingan.index') }}"
class="nav-link">
```

```
            <i class="fas fa-users nav-icon"></i>
            <p>Mahasiswa Bimbingan</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('mahasiswa-ujian-sidang.index') }}"
class="nav-link">
            <i class="fas fa-calendar nav-icon"></i>
            <p>Mahasiswa Ujian Sidang</p>
        </a>
    </li>
</ul>
```

Segmen Program 4.4 mendefinisikan struktur menu dari hak akses mahasiswa. Menu-menu yang dapat diakses dari *sidebar* yakni halaman beranda, pengumuman, dan pengajuan tugas akhir.

Segmen Program 4.4 Menu *Sidebar* Hak Akses Mahasiswa

```
<ul class="nav nav-pills nav-sidebar flex-column" data-
widget="treeview" role="menu" data-accordion="false">
    <li class="nav-item">
        <a href="/" class="nav-link">
            <i class="fas fa-home nav-icon"></i>
            <p>Beranda</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('pengumuman.index') }}" class="nav-link">
            <i class="fas fa-bullhorn nav-icon"></i>
            <p>Pengumuman</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('pengajuan-tugas-akhir.index') }}"
class="nav-link">
            <i class="fas fa-book nav-icon"></i>
            <p>Pengajuan Tugas Akhir</p>
        </a>
    </li>
</ul>
```

Segmen Program 4.5 mendefinisikan struktur menu dari hak akses PIC. Menu-menu yang dapat diakses dari *sidebar* yakni halaman beranda, koordinator tugas akhir, model tugas akhir dan status tugas akhir.

Segmen Program 4.5 Menu *Sidebar* Hak Akses PIC

```
<ul class="nav nav-pills nav-sidebar flex-column" data-
widget="treeview" role="menu"
data-accordion="false">
    <li class="nav-item">
        <a href="/" class="nav-link">
```

```
            <i class="fas fa-home nav-icon"></i>
            <p>Beranda</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('koordinator.index') }}" class="nav-link">
            <i class="fas fa-user-tie nav-icon"></i>
            <p>Daftar Koordinator</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('model-tugas-akhir.index') }}" class="nav-
link">
            <i class="fas fa-book nav-icon"></i>
            <p>Model Tugas Akhir</p>
        </a>
    </li>
    <li class="nav-item">
        <a href="{{ route('status-tugas-akhir.index') }}"
class="nav-link">
            <i class="fas fa-clipboard-list nav-icon"></i>
            <p>Status Tugas Akhir</p>
        </a>
    </li>
</ul>
```

4.2.2 Pembuatan Menu

Pada dasarnya, sebuah menu memiliki dua jenis halaman utama, yaitu halaman indeks dan halaman form. Halaman indeks digunakan untuk menampilkan informasi entri data yang disimpan dalam sebuah *datatable*. Di beberapa menu, halaman indeks juga memiliki fitur filter yang dibuat menggunakan Bootstrap *accordion* untuk menampung elemen-elemen input yang digunakan pada filter. Sementara itu, halaman form digunakan oleh pengguna untuk menambahkan, melihat, dan mengedit data. Dalam satu halaman menu terdapat dua bagian utama, yaitu *header* halaman dan konten. *Header* halaman memuat informasi nama menu atau halaman yang sedang diakses serta *breadcrumb* atau jejak navigasi yang memudahkan pengguna mengakses halaman sebelumnya. Bagian konten menggunakan Bootstrap *card* sebagai *container* untuk informasi data. Pada halaman indeks, konten berisi *datatable* dan tombol tambah untuk menambahkan data. Sedangkan pada halaman form, konten berisi elemen-elemen input formulir data.

Segmen Program 4.6 merupakan contoh halaman indeks dari menu Bidang Studi di koordinator. Pada halaman indeks tersebut terdapat dua bagian yakni *header* halaman dan konten seperti yang sudah dijelaskan sebelumnya.

Segmen Program 4.6 Halaman Indeks Menu

```
{{-- parent layout --}}
@extends('layouts.1')

{{-- header page dan breadcrumb --}}
@section('page_header')
    <div class="content-header">
        <div class="container-fluid">
            <div class="row mb-2">
                <div class="col-sm-6">
                    <h1 class="m-0">Bidang Studi</h1>
                </div><!-- /.col -->
                <div class="col-sm-6">
                    <ol class="breadcrumb float-sm-right">
                        <li class="breadcrumb-item active"><a
href="{{ route('home') }}">Home</a></li>
                        <li class="breadcrumb-item active"><a
href="#">Bidang Studi</a></li>
                    </ol>
                </div><!-- /.col -->
            </div><!-- /.row -->
        </div><!-- /.container-fluid -->
    </div>
    <!-- /.content-header -->
@endsection

{{-- main content --}}
@section('content')
    <div class="card">
        <div class="card-header">
            <h3 class="card-title mt-1">Daftar Bidang Studi</h3>
        </div>

        <div class="card-body">
            <div class="mb-4">
                <a href="{{ route($route.'.create') }}"
role="button" class="btn btn-outline-primary">
                <i class="fa fa-plus-circle mr-1"></i>Tambah
Data
                </a>
            </div>
            <div>
                <!-- tambah table class dt untuk pakai datatable -->
                <table id="table" class="table table-bordered table-
stripped table-hover dt">
                    <thead>
                        <tr>
                            <th width="5">No</th>
                            <th>Nama Bidang</th>
                            <th>Kepala Bidang</th>
                            <th width="150" data-
orderable="false">Aksi</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach ($listData as $row)
                            <tr>
                                <td>{{ $loop->iteration }}</td>
                                <td>{{ $row->nama }}</td>
```

```
                                <td>{{ $row->kepala_bidang->biodata->namaWithGelar() }}</td>
                                <td>
                                    <div class="table-action-wrapper">
                                        <a href="{{ route($route . '.show', [$param => encrypt($row->id)]) }}"
                                            class="table-action-btn btn btn-sm mr-1 mt-1" data-toggle="tooltip"
                                            data-placement="bottom" title="View" style="background-color: #d7f5fc; color: #03c3ec;">
                                            <i class="fas fa-eye"></i>
                                        </a>
                                        <a href="{{ route($route . '.edit', [$param => encrypt($row->id)]) }}"
                                            class="table-action-btn btn btn-sm mr-1 mt-1" data-toggle="tooltip"
                                            data-placement="bottom" title="Edit" style="background-color: #fff2d6; color: #ffab00;">
                                            <i class="fas fa-edit"></i>
                                        </a>
                                        <a class="table-action-btn btn btn-sm mr-1 mt-1"

onclick="confirmDel('{{ route($route . '.destroy', [$param => encrypt($row->id)]) }}', `Apakah anda yakin menghapus bidang studi <strong>&amp;quot;{{ $row->nama }}&amp;quot;</strong> ?`)"
                                            data-toggle="tooltip" data-placement="bottom" title="Delete" style="background-color: #ffe0db; color: #ff3e1d;">
                                            <i class="fas fa-trash"></i>
                                        </a>
                                    </div>
                                </td>
                            </tr>
                        @endforeach
                    </tbody>
                </table>
            </div>
        </div>
    </div>
@endsection
```

Segmen Program 4.7 merupakan contoh halaman form dari menu Bidang Studi di koordinator. Pada halaman form tersebut terdapat dua bagian yakni header halaman dan konten seperti yang sudah dijelaskan sebelumnya. Bagian konten berisi elemen-elemen input yang menggunakan fungsi *render* pada *FormHelper* untuk menampilkan elemen input secara dinamis sesuai dengan parameter yang ditambahkan pada fungsi tersebut. Elemen-elemen input diletakkan di dalam elemen form. Atribut *action* pada form dibuat dinamis yang

bergantung pada mode (*add*, *edit*, atau *view*) saat ini. Jika *mode* bukan *add* atau tambah, maka form akan mengirimkan metode *PUT*, sebaliknya form akan mengirimkan metode *POST*.

Segmen Program 4.7 Halaman Form Menu

```
{{-- parent layout --}}
@extends('layouts.1')

{{-- header page dan breadcrumb --}}
@section('page_header')
    <div class="content-header">
        <div class="container-fluid">
            <div class="row mb-2">
                <div class="col-sm-6">
                    <h1 class="m-0">Bidang Studi</h1>
                </div><!-- /.col -->
                <div class="col-sm-6">
                    <ol class="breadcrumb float-sm-right">
                        <li class="breadcrumb-item active"><a
href="{{ route('home') }}">Home</a></li>
                        <li class="breadcrumb-item active"><a
href="{{ route($route.'.index') }}">Bidang Studi</a></li>
                    </ol>
                </div><!-- /.col -->
            </div><!-- /.row -->
        </div><!-- /.container-fluid -->
    </div>
    <!-- /.content-header -->
@endsection

{{-- main content --}}
@section('content')
    <div class="card">
        <form action="/{{ $url }}@if($mode !=
'add')/{{ encrypt($data['id']) }}@endif" method="POST">
            @if($mode != 'add')
                @method('put')
            @else
                @method('post')
            @endif
            @csrf

            <div class="card-header">
                <h3 class="card-title mt-1"><span style="text-
transform: capitalize;">{{ $mode }}</span> Bidang Studi</h3>
                <div class="card-tools" style="margin-right: 0;">
                    <a href="{{ ($mode == 'edit') ?
route($route.'.show', [$param => encrypt($data['id'])]) :
route($route.'.index') }}" role="button" class="btn btn-secondary">
                        <i class="fa fa-arrow-circle-left mr-
2"></i>Kembali
                    </a>
                    @if ($mode == 'view')
                        <a href="{{ route($route.'.edit', [$param =>
encrypt($data['id'])]) }}" role="button" class="btn btn-warning">
                            <i class="fa fa-edit mr-1"></i>Edit
                        </a>
```

```
                    @endif
                </div>
            </div>

            <div class="card-body col-12 col-xl-8">
                {{-- FORM --}}
                {!! renderInput('mb-3 row', 'col-lg-4 col-form-
label', 'col-lg-8', 'text', 'nama', 'Nama', $data['nama'] ?? '',
$mode, 'required autofocus') !!}
                {!! renderSelect('mb-3 row', 'col-lg-4 mt-2', 'col-
lg-8 mt-2 mt-sm-0', 'kepala_bidang_id', 'Kepala Bidang', 'js-data-
example-ajax w-100 select2 form-control mb-3',
$data['kepala_bidang_id'] ?? '', $selectDosen, $mode,
'required') !!}
            </div>

            @if ($mode != 'view')
            <div class="card-footer">
                <button type="submit" class="btn btn-primary"><i
class="fa fa-save mr-2"></i>Simpan</button>
            </div>
            @endif
        </form>
    </div>
@endsection
```

Segmen Program 4.8 merupakan contoh penggunaan filter pada halaman indeks di halaman Pengajuan Tugas Akhir. Sama seperti halaman form, elemen input pada filter menggunakan fungsi *render* yang didefinisikan pada *FormHelper*. Di dalam filter juga terdapat elemen form untuk menjalankan filter ketikan tombol "*Search*" ditekan.

Segmen Program 4.8 Filter pada Halaman Indeks

```
<div class="accordion" id="accordionFilter">
        <div class="card">
            <form id="filterForm" action="{{ route('pengajuan-tugas-
akhir.filter') }}" method="GET">
                <div class="card-header" id="heading">
                    <h2 class="mb-0">
                        <button class="btn btn-link btn-block text-
left px-0 position-relative" type="button"
                            data-toggle="collapse" data-
target="#collapseFilter" aria-expanded="true"
                            aria-controls="collapseFilter"
style="color: #1E3258; font-weight: bold; font-size: 1.1rem;">
                            Filter
                            <span class="collapse-icon">
                                <i class="fas fa-chevron-down"></i>
                            </span>
                        </button>
                    </h2>
                </div>

                <div id="collapseFilter" class="collapse show
container-fluid" aria-labelledby="heading"
```

```
                    data-parent="#accordionFilter">
                    <div class="card-body row d-flex align-items-
end">
                        {!! renderSelect(
                            'mb-3 row col-12 col-md-4',
                            'col-12',
                            'col-12 mt-sm-0',
                            'semester_id',
                            'Semester',
                            'js-data-example-ajax w-100 select2
form-control',
                            $currentSemester ?? '',
                            $selectSemester,
                            'add',
                            '',
                            id: 'filter-semester',
                        ) !!}
                        {!! renderSelect(
                            'mb-3 row col-12 col-md-4',
                            'col-12',
                            'col-12 mt-sm-0',
                            'unit_id',
                            'Program Studi',
                            'js-data-example-ajax w-100 select2
form-control',
                            $currentUnit ?? '',
                            $selectUnit,
                            'add',
                            '',
                            id: 'filter-unit',
                        ) !!}
                        {!! renderSelect(
                            'mb-3 row col-12 col-md-4',
                            'col-12',
                            'col-12 mt-sm-0',
                            'status_tugas_akhir_id',
                            'Status',
                            'js-data-example-ajax w-100 select2
form-control',
                            $currentStatusTugasAkhir ?? 'semua',
                            $selectStatusTugasAkhir,
                            'add',
                            '',
                            id: 'filter-status',
                            encrypt: false,
                        ) !!}
                        <div class="mb-3 row col-12 col-md-4">
                            <div class="col-12">
                                <button class="btn btn-primary"
id="btnSearch" type="submit">
                                    <span class="spinner-border
spinner-border-sm mr-1 d-none" role="status"
                                        aria-hidden="true"></span>
                                    Search
                                </button>
                            </div>
                        </div>
                    </div>
```

```
                </div>
            </form>
        </div>
    </div>
```

4.2.3 *FormHelper*

FormHelper adalah sebuah *file helper* yang digunakan untuk memuat fungsi-fungsi PHP untuk menampilkan *input field* pada sebuah form. Segmen Program 4.9 menunjukkan fungsi-fungsi dalam *FormHelper* yakni fungsi *renderInput*, *renderTextArea*, *renderSelect*, *renderSelectMultiple*, *renderCheckbox*, *renderCheckboxWithLabel*, dan *renderRadioButton*. Setiap fungsi memiliki beberapa parameter yang digunakan untuk mengatur tampilan dan perilaku elemen form yang dihasilkan. Misalnya, parameter untuk menentukan kelas CSS, tipe input, nama dan ID elemen, nilai awal, *mode* tampilan (seperti *view* atau *edit*), serta opsi tambahan lainnya seperti atribut HTML yang diperlukan. Fungsi-fungsi ini bertujuan untuk memudahkan proses pembuatan elemen input form dan memastikan konsistensi dan efisiensi dalam pembuatan form yang kompleks. Implementasi penggunaan fungsi *FormHelper* pada sebuah form dapat dilihat di Segmen Program 4.7 dan Segmen Program 4.8.

Segmen Program 4.9 Fungsi-fungsi Render *Input Field* pada *FormHelper*

```
<?php
function renderInput($div_class, $label_class, $input_div_class,
$type, $name, $label, $value, $mode, $options = '', $id = '',
$class_input = 'form-control')
{
    if ($mode == 'view') {
        if ($type == 'checkbox') {
            $options = $options . ' disabled';
        } else {
            $options = $options . ' readonly';
        }
    }

    $oldValue = old($name, $value);
    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    if ($id == '') {
        $id = $name;
    }

    $field = '<div class="' . $div_class . '">';
    if ($label != '') {
        $field .= '<label for="' . $id . '" class="' .
$label_class . ' ' . $options . '">' . $label . ' ' . $setRequired .
'</label>';
```

```
    }

    $field .= '<div class="' . $input_div_class . '">';

    $field .= '<input type="' . $type . '" class="' . $class_input .
'" id="' . $id . '" name="' . $name . '" value="' . $oldValue . '"
' . $options . '>
            </div>
        </div>';

    return $field;
}

function renderTextArea($div_class, $div_class_label,
$div_class_input, $name, $label, $rows, $value, $mode, $options, $id
= '', $class_input = 'form-control')
{
    if ($mode == 'view') {
        $options = $options . ' readonly';
    }

    $oldValue = old($name, $value);
    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    if ($id == '') {
        $id = $name;
    }

    $field = '
        <div class="' . $div_class . '">';
    $field .= '<div class="' . $div_class_label . '">';
    $field .= '<label for="' . $id . '" class="form-label" "' .
$options . '">' . $label . ' ' . $setRequired . '</label>
            </div>';
    $field .= '<div class="' . $div_class_input . '">';
    $field .= '<textarea class="' . $class_input . '" value="' .
$oldValue . '" name="' . $name . '" rows="' . $rows . '" id="' .
$id . '" ' . $options . '>' . $oldValue . '</textarea>
            </div>
        </div>';

    return $field;
};

function renderSelect($class, $div_class_label, $div_class_input,
$name, $label, $class_select, $data, $array, $mode, $options = '',
$option_value = 'id', $option_name = 'nama', $id = '', $encrypt =
true, $note = '')
{
    $disabled = '';
    if ($mode == 'view') {
        $class_select .= ' readonly-select';

        if (str_contains($class_select, 'select2')) {
            $disabled = 'disabled';
```

```
        }
    }

    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    if ($id == '') {
        $id = $name;
    }

    $field = '<div class="' . $class . '">';
    if ($label != '') {
        $field .= '<div class="' . $div_class_label . '">';
        $field .= '<label for="' . $id . '" class="form-label"
id="label_' . $id . '">' . $label . ' ' . $setRequired . '</label>';
        $field .= '</div>';
    }

    $field .= '<div class="' . $div_class_input . '">';
    $field .= '<select id="' . $id . '" class="' . $class_select .
'" name="' . $name . '" ' . $options . ' style="width: 100%;" '.
$disabled.'>';
    foreach ($array as $value) {
        $selected = old($name, $data) == $value[$option_value] ?
'selected' : '';
        if ($encrypt) {
            $field .= '<option value="' .
encrypt($value[$option_value]) . '" ' . $selected . '>' .
$value[$option_name] . '</option>';
        } else {
            $field .= '<option value="' . $value[$option_value] . '"
' . $selected . '>' . $value[$option_name] . '</option>';
        }
    }
    $field .= '</select>';
    if ($note != '') {
        $field .= '<small class="form-text text-
muted">'.$note.'</small>';
    }
    $field .= '</div>';
    $field .= '</div>';

    return $field;
};

function renderSelectMultiple($class, $div_class_label,
$div_class_input, $name, $label, $class_select, $data, $array,
$mode, $options = '', $option_value = 'id', $option_name = 'nama',
$id = '', $encrypt = true, $note = '')
{
    $disabled = '';
    if ($mode == 'view') {
        $class_select .= ' readonly-select';

        if (str_contains($class_select, 'select2')) {
            $disabled = 'disabled';
```

```
        }
    }

    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    if ($id == '') {
        $id = $name;
    }

    $field = '<div class="' . $class . '">';
    if ($label != '') {
        $field .= '<div class="' . $div_class_label . '">';
        $field .= '<label for="' . $id . '" class="form-label"
id="label_' . $id . '">' . $label . ' ' . $setRequired . '</label>';
        $field .= '</div>';
    }

    $field .= '<div class="' . $div_class_input . '">';
    $field .= '<select id="' . $id . '" class="' . $class_select .
'" name="' . $name . '[]" ' . $options . ' style="width: 100%;" '.
$disabled.' multiple="multiple">';
    foreach ($array as $value) {
        $selected = in_array($value[$option_value], old($name,
$data)) ? 'selected' : '';
        if ($encrypt) {
            $field .= '<option value="' .
encrypt($value[$option_value]) . '" ' . $selected . '>' .
$value[$option_name] . '</option>';
        } else {
            $field .= '<option value="' . $value[$option_value] . '"
' . $selected . '>' . $value[$option_name] . '</option>';
        }
    }
    $field .= '</select>';
    $field .= '<small class="form-text text-muted">Anda dapat
memilih lebih dari satu opsi.</small>';
    $field .= '</div>';
    $field .= '</div>';

    return $field;
};

function renderCheckbox($div_class, $div_input, $id, $name,
$options, $isChecked = null)
{
    $field = '
            <div class="' . $div_class . '">
                <input type="checkbox" class="' . $div_input . '"
id="' . $id . '" name="' . $name . '" ' . $options . ' ' .
($isChecked ? 'checked' : '') . '>
            </div>
        ';

    return $field;
}
```

```
function renderCheckboxWithLabel($div_class, $id, $name, $mode,
$options, $label, $isChecked = null)
{

    if ($mode == 'view') {
        $options .= ' disabled';
    }

    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    $field = '<div class="' . $div_class . '">';
    $field .= '<div class="form-check custom-checkbox mb-3">';
    $field .= '<input type="checkbox" class="form-check-input"
name="' . $name . '" id="' . $id . '" ' . $options . ' ' .
$isChecked . '>';
    $field .= '<label class="form-check-label" for="' . $id . '">' .
$label . ' ' . $setRequired . '</label>';
    $field .= '</div>';
    $field .= '</div>';

    return $field;
}

function renderRadioButton($class, $name, $id, $label, $value,
$options)
{
    $oldValue = old($name);

    $checked = ($oldValue == $value) ? 'checked' : '';

    $setRequired = '';
    if (str_contains($options, 'required')) {
        $setRequired = '<span style="color: red">*</span>';
    }

    $field = '
            <div class="' . $class . '">
                <input class="form-check-input" type="radio"
value="' . $value . '" name="' . $name . '" id="' . $id . '" ' .
$options . ' ' . $checked . '>
                <label class="form-check-label" for="' . $id . '">
                ' . $label . ' ' . $setRequired . '
                </label>
            </div>
        ';

    return $field;
}
?>
```

4.2.4 Pembuatan Model Laravel

Dalam Laravel, model adalah bagian dari arsitektur MVC (*Model-View-Controller*) yang digunakan untuk berinteraksi dengan *database*. Model merepresentasikan tabel dalam *database* dan berfungsi untuk memanipulasi data yang ada dalam tabel tersebut. Model juga menghubungkan tabel dengan relasi yang ada, seperti *one-to-many* atau *many-to-many*. Segmen Program 4.10 adalah salah satu contoh model bernama "*BidangStudi*" yang merepresentasikan tabel "*bidang_studi"* di *database*. Variable *connection* digunakan untuk mengatur koneksi *database* yang digunakan. Kemudian variabel *table* digunakan untuk mendefinisikan nama tabel dalam *database*.

Segmen Program 4.10 Pembuatan Model Laravel

```
<?php

namespace App\Models\Simta;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

use App\Models\Ref\Unit;
use App\Models\Pegawai\Pegawai;

class BidangStudi extends Model
{
    use HasFactory;

    protected $connection = 'pgsql';
    protected $table = 'bidang_studi';
    protected $guarded = ['id'];

    public function tugas_akhir()
    {
        return $this->hasMany(TugasAkhir::class, 'bidang_studi_id',
'id');
    }

    public function unit()
    {
        return $this->belongsTo(Unit::class, 'unit_id', 'id');
    }

    public function kepala_bidang()
    {
        return $this->belongsTo(Pegawai::class, 'kepala_bidang_id',
'id');
    }
}
```

4.2.5 Metode *pageProperties* pada *Class Controller*

Segmen Program 4.11 adalah sebuah metode *pageProperties()* pada class Controller. Metode ini berfungsi untuk membuat sebuah *array* yang memuat variabel-variabel yang berkaitan dengan sebuah menu seperti *url*, *route*, *param* atau parameter *route* dan *mode* (*add*, *edit*, dan *view*). Variabel-variabel tersebut digunakan pada *blade template* Laravel ketika membuat sebuah menu dan bertujuan untuk mempersingkat kodingan serta memudahkan penggunaan ulang. Sebagai contoh, variabel *url* digunakan untuk mendefinisikan URL menu yang digunakan pada atribut action pada form di setiap menu seperti pada Segmen Program 4.7 dan Segmen Program 4.8. Metode ini dipanggil pada metode *index*, *create*, *view* dan *edit* di setiap *Controller* menu.

Segmen Program 4.11 Metode *pageProperties* pada *Class Controller*

```
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

    public function pageProperties($menuUrl, $menuRoute, $menuParam,
$mode)
    {
        $pageProperties = [
            'url' => $menuUrl,
            'route' => $menuRoute,
            'param' => $menuParam,
            'mode' => $mode
        ];

        return $pageProperties;
    }
}
```

4.2.6 Metode *moduleGlobal* pada *Class GlobalVariable*

Segmen Program 4.12 adalah sebuah metode *moduleGlobal()* pada *class GlobalVariable*. Metode ini berfungsi untuk membuat variabel global yang akan digunakan di seluruh metode dalam satu *controller* menu. Nilai-nilai yang ditambahkan pada metode ini akan digunakan untuk mengisi nilai parameter pada fungsi *pageProperties()*. Nilai-nilai ini didefinisikan di dalam fungsi *constructor* dan disimpan dalam variabel bernama *globalVariable*.

Segmen Program 4.12 Metode *moduleGlobal* pada *Class GlobalVariable*

```
<?php

namespace App\Http\Controllers;

class GlobalVariable
{
    public function ModuleGlobal($module, $subModule, $menuUrl,
$menuRoute, $menuParam)
    {
        // Initialize your global properties in the constructor
        $this->module = $module;
        $this->subModule = $subModule;
        $this->menuUrl = $menuUrl;
        $this->menuRoute = $menuRoute;
        $this->menuParam = $menuParam;
    }
}
```

4.2.7 Inisialisasi Variabel-variabel pada *Controller*

Segmen Program 4.13 adalah inisialisasi awal dalam sebuah *controller*. Di bagian atas *class* terdapat *use statements* untuk mengimpor beberapa *class* yang digunakan dalam *controller* sehingga tidak perlu menyebutkan *namespace*-nya secara lengkap setiap kali digunakan. Kemudian *namespace* pada awal segmen program digunakan untuk mengelompokkan *controller* ke dalam struktur direktori yang sesuai. Ada tiga variabel dasar yang digunakan dalam *controller* yakni *$globalVariable*, *$index_file* dan *$form_file*. Variabel *$globalVariable* digunakan untuk menyimpan data yang dikirimkan oleh fungsi *ModuleGlobal* dengan parameter-parameternya. Kemudian variabel *$index_file* dan *$form_file* digunakan untuk menentukan nama *file blade* yang akan digunakan untuk halaman indeks (*list* data) dan halaman form. Variabel *$arrayDosen* adalah contoh variabel tambahan yang digunakan untuk menyimpan *list* data dosen.

Segmen Program 4.13 Inisialisasi Variabel-variabel pada *Controller*

```
<?php

namespace App\Http\Controllers\Referensi;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Crypt;

use App\Http\Controllers\Controller;
use App\Http\Controllers\GlobalVariable;

use App\Models\Ref\Unit;
use App\Models\Simta\BidangStudi;
```

```
class BidangStudiController extends Controller
{
    private $globalVariable;
    private $index_file;
    private $form_file;

    private $arrayDosen;

    public function __construct(GlobalVariable $globalVariable)
    {
        $this->globalVariable = $globalVariable;
        $this->globalVariable->ModuleGlobal(module: 'referensi',
subModule: 'bidang_studi', menuUrl: 'bidang-studi', menuRoute:
'bidang-studi', menuParam: 'bidang_studi');

        $this->index_file = 'referensi.bidang_studi.index';
        $this->form_file = 'referensi.bidang_studi.form';
    }
```

4.2.8 Metode Index pada *Controller*

Segmen Program 4.14 adalah metode *index* dalam *controller* Laravel yang digunakan untuk menampilkan halaman indeks atau *list* data. Metode *index* dimulai dengan menginisialisasi variabel *$formData* yang memuat data dari metode *pageProperties()*, seperti URL menu, *route*, parameter dan mode yang diberi nilai *"index"*. Kemudian, judul halaman ditambahkan pada variabel *$formData['title']*. Selanjutnya, variabel *$formData['listData']* digunakan untuk menyimpan data-data yang akan ditampilkan dalam menu tersebut. Terakhir, metode *index* mengembalikan *view* yang sesuai dengan nama *file blade* yang disimpan pada variabel *$index_file*, bersama dengan data yang telah dipersiapkan sebelumnya dalam variabel *$formData*.

Segmen Program 4.14 Metode *Index* pada *Controller*

```
public function index()
    {
        $formData = $this->pageProperties(menuUrl: $this-
>globalVariable->menuUrl, menuRoute: $this->globalVariable-
>menuRoute, menuParam: $this->globalVariable->menuParam, mode:
'index');
        $formData['title'] = 'Bidang Studi';

        $unit = Unit::find(session('unit_id'));
        $formData['listData'] = $unit->bidang_studi->sortBy('nama');

        return view($this->index_file, $formData);
    }
```

4.2.9 Metode *Create* dan *Store* pada *Controller*

Segmen Program 4.15 digunakan untuk membuat data baru dan melakukan penyimpanan data melalui metode *create* dan *store*. Metode *create* dimulai dengan menginisialisasi variabel *$formData* yang memuat data dari metode *pageProperties()*, seperti URL menu, *route*, parameter dan mode yang diberi nilai *"add"*. Kemudian, judul halaman ditambahkan pada variabel *$formData['title']*. Selanjutnya, variabel *$formData['selectDosen']* adalah contoh variabel yang digunakan untuk menampung opsi-opsi yang akan ditampilkan dalam elemen select pada halaman form. Hal ini berguna ketika opsi *select* memerlukan data dari *database*. Terakhir, metode *create* mengembalikan *view* yang sesuai dengan nama *file blade* yang disimpan pada variabel *$form_file*, bersama dengan data yang telah dipersiapkan sebelumnya dalam variable *$formData*.

Berikutnya, terdapat metode *strore* yang dipanggil ketika pengguna melakukan penyimpanan data. Parameter pada metode ini adalah input *request* yang berisikan data yang dikirimkan pengguna saat mengisi form. Pertama, *request* tersebut melewati validasi untuk memastikan data yang dikirimkan sesuai dengan format yang diharapkan. Selanjutnya, untuk data input yang berupa ID seperti *kepala_bidang_id*, perlu dilakukan dekripsi terlebih dahulu karena data ID tersebut dienkripsi pada halaman *view* form untuk meningkatkan keamanan data. Setelah semua data divalidasi, data baru dibuat menggunakan fungsi *create* dari model Laravel. Terakhir, metode *store* akan mengarahkan pengguna ke halaman sesuai dengan variabel global *menuUrl* dan mengirimkan notifikasi sukses "Data berhasil ditambahkan".

Segmen Program 4.15 Metode *Create* dan *Store* pada *Controller*

```
public function create()
    {
        $formData = $this->pageProperties(menuUrl: $this-
>globalVariable->menuUrl, menuRoute: $this->globalVariable-
>menuRoute, menuParam: $this->globalVariable->menuParam, mode:
'add');
        $formData['title'] = 'Create Bidang Studi';

        $this->getDosen();

        $formData['selectDosen'] = $this->arrayDosen;

        return view($this->form_file, $formData);
    }

public function store(Request $request)
    {
        $request->validate([
            'nama' => 'required|max:255',
            'kepala_bidang_id' => 'required'
```

```
        ]);

        // decrypt the select option value
        $kepalaBidangId = Crypt::decrypt($request-
>kepala_bidang_id);

        // replace the encrypted value with the decrypted one
        $request->merge(['kepala_bidang_id' => $kepalaBidangId]);

        $unitId = session('unit_id'); // get unit_id from session
        $data = array_merge($request->post(), ['unit_id' =>
$unitId]);

        BidangStudi::create($data);

        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Data berhasil ditambahkan');
    }
```

4.2.10 Metode *Show* pada *Controller*

Metode *show* pada Segmen Program 4.16 digunakan untuk menampilkan data spesifik pada halaman form. Parameter yang diterima oleh metode ini adalah ID dari data yang ingin dilihat pengguna. ID ini dikirim melalui parameter pada *route show*, yang merupakan bagian dari *resource routing* bawaan Laravel. Sama seperti metode *index* dan *create*, metode *show* juga diawali dengan menginisialisasi variabel *$formData*, yang memuat data dari metode *pageProperties()*, serta menambahkan judul halaman dan menyiapkan data untuk opsi-opsi yang akan ditampilkan dalam elemen *select*. Setelah itu, ID yang dikirim melalui parameter *route* digunakan untuk mengambil data dari database berdasarkan ID tersebut. Data ini kemudian ditambahkan ke dalam *$formData* untuk digunakan dalam tampilan. Terakhir, metode *show* mengembalikan *view* yang sesuai dengan nama *file blade* yang disimpan pada variabel *$form_file*, bersama dengan data yang telah dipersiapkan sebelumnya dalam variabel *$formData*.

Segmen Program 4.16 Metode *Show* pada *Controller*

```
public function show(string $id)
    {
        $formData = $this->pageProperties(menuUrl: $this-
>globalVariable->menuUrl, menuRoute: $this->globalVariable-
>menuRoute, menuParam: $this->globalVariable->menuParam, mode:
'view');
        $formData['title'] = 'View Bidang Studi';

        $this->getDosen();

        $formData['data'] = BidangStudi::where('id', decrypt($id))-
>first();
```

```
        $formData['selectDosen'] = $this->arrayDosen;

        return view($this->form_file, $formData);
    }
```

4.2.11 Metode *Edit* dan *Update* pada *Controller*

Segmen Program 4.17 digunakan untuk melakukan perubahan data melalui metode *edit* dan *update*. Sama seperti metode *show*, metode *edit* digunakan untuk menyiapkan data yang akan ditampilkan pada form untuk diperbarui oleh pengguna. Data ini disimpan dalam variabel *$formData* yang memuat informasi dari metode *pageProperties()*, judul halaman, opsi-opsi untuk elemen *select*, dan data yang diambil dari database berdasarkan ID yang dikirim melalui parameter *route*. Setelah semua data siap, metode *edit* mengembalikan *view* yang sesuai dengan nama *file blade* yang disimpan dalam variabel *$form_file*, bersama dengan data yang telah dipersiapkan sebelumnya dalam *$formData*.

Selanjutnya, metode *update* digunakan untuk menyimpan hasil perubahan data yang dikirim melalui *request input*. Proses ini diawali dengan validasi data *request* yang masuk untuk memastikan semua data sesuai dengan format yang diharapkan. Setelah itu, ID yang terenkripsi didekripsi kembali. Setelah semua data siap, dilakukan pembaruan data menggunakan fungsi *update()* dari model Laravel. Terakhir, metode *update* mengarahkan pengguna ke halaman yang sesuai dengan variabel global *menuUrl* dan mengirimkan notifikasi sukses "Data berhasil diperbarui".

Segmen Program 4.17 Metode *Edit* dan *Update* pada *Controller*

```
public function edit(string $id)
    {
        $formData = $this->pageProperties(menuUrl: $this-
>globalVariable->menuUrl, menuRoute: $this->globalVariable-
>menuRoute, menuParam: $this->globalVariable->menuParam, mode:
'edit');
        $formData['title'] = 'Edit Bidang Studi';

        $this->getDosen();

        $formData['data'] = BidangStudi::where('id', decrypt($id))-
>first();
        $formData['selectDosen'] = $this->arrayDosen;

        return view($this->form_file, $formData);
    }

public function update(Request $request, string $id)
    {
        // decrypt the select option value
```

```
        $kepalaBidangId = Crypt::decrypt($request-
>kepala_bidang_id);

        // replace the encrypted value with the decrypted one
        $request->merge(['kepala_bidang_id' => $kepalaBidangId]);

        $data = $request->validate([
            'nama' => 'required|max:255',
            'kepala_bidang_id' => 'required'
        ]);

        BidangStudi::where('id', decrypt($id))->update($data);

        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Data berhasil diperbarui');
    }
```

4.2.12 Metode *Destroy* pada *Controller*

Segmen Program 4.18 adalah contoh metode *destroy* dalam *controller* yang digunakan untuk menghapus sebuah data berdasarkan parameter ID yang dikirimkan lewat *route*. Penghapusan data dilakukan dengan menggunakan fungsi *destroy()* dari model, setelah ID didekripsi terlebih dahulu untuk memastikan keamanan data. Setelah penghapusan data berhasil, metode *destroy* mengarahkan pengguna ke halaman yang sesuai dengan variabel global *menuUrl* dan mengirimkan notifikasi sukses "Data berhasil dihapus".

Segmen Program 4.18 Metode *Destroy* pada *Controller*

```
public function destroy(string $id)
    {
        BidangStudi::destroy(decrypt($id));

        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Data berhasil dihapus');
    }
```

4.2.13 Metode Filter pada *Controller*

Segmen Program 4.19 adalah contoh metode *filter* yang digunakan untuk memfilter daftar data pada halaman indeks di berbagai menu. Metode ini dijalankan ketika pengguna mengisi *input field* pada filter lalu menekan tombol *"Search"*. Pada dasarnya, metode ini mirip dengan metode *index* namun yang membedakan adalah pada saat mengambil daftar data, ditambahkan parameter untuk memfilter data. Parameter pada metode ini adalah input request yang berisikan data yang dikirimkan pengguna saat mengisi filter.

Metode ini dimulai dengan mendekripsi input ID dan mengambil nilai dari input request lainnya. Kemudian dilakukan pemilihan kondisi berdasarkan input filter yang diberikan

untuk memfilter data yang relevan dari *database*, sesuai dengan kriteria yang dimasukkan pengguna, seperti status tertentu atau rentang waktu tertentu. Setelah data difilter, setiap data yang relevan dapat diformat ulang atau ditambahkan informasi tambahan yang diperlukan untuk ditampilkan pada tampilan akhir. Data hasil filter, nilai filter saat ini, dan judul halaman ditambahkan ke dalam *$formData*. Terakhir, metode *filter* mengembalikan *view* yang sesuai dengan nama *file blade* yang disimpan pada variabel *$index_file*, bersama dengan data yang telah dipersiapkan sebelumnya dalam variabel *$formData*.

Segmen Program 4.19 Metode Filter pada *Controller*

```
public function filter(Request $request)
    {

        $semesterId = Crypt::decrypt($request-
>input('semester_id'));
        $status = $request->input('status');

        if ($status == 'active') {
            $pengumuman = Pengumuman::where('semester_id',
$semesterId)
                ->where('unit_id', session('unit_id'))
                ->where('is_active', true)
                ->orderBy('created_at', 'desc')
                ->get();
        } else if ($status == 'inactive') {
            $pengumuman = Pengumuman::where('semester_id',
$semesterId)
                ->where('unit_id', session('unit_id'))
                ->where('is_active', false)
                ->orderBy('created_at', 'desc')
                ->get();
        } else {
            $pengumuman = Pengumuman::where('semester_id',
$semesterId)
                ->where('unit_id', session('unit_id'))
                ->orderBy('created_at', 'desc')
                ->get();
        }

        // Change created_at date format and get semester name
        foreach ($pengumuman as $data) {
            $data->formatted_created_at = $data->created_at-
>translatedFormat('F j, Y');
            $data->semester_nama = $data->semester->nama;
        }

        $formData = $this->pageProperties(menuUrl: $this-
>globalVariable->menuUrl, menuRoute: $this->globalVariable-
>menuRoute, menuParam: $this->globalVariable->menuParam, mode:
'index');

        $this->getSemester();
```

```
        $formData['selectActive'] = $this->arrayIsActiveFilter;
        $formData['selectSemester'] = $this->arraySemester;

        $formData['listData'] = $pengumuman;

        // For filter
        $formData['currentSemester'] = $semesterId;
        $formData['currentStatus'] = $status;

        $formData['title'] = 'Pengumuman';

        return view($this->index_file, $formData);
    }
```

4.2.14 Metode *berkasUpload* dan *fileTugasAkhirUpload* pada *FormBerkasDataController*

Segmen Program 4.20 adalah metode untuk mengunggah *file* menggunakan S3 untuk mengumpulkan berkas kebutuhan sidang akhir dan *file* tugas akhir. Metode ini memiliki parameter *request* yang terdiri dari *tugas_akhir_id*, *form_berkas_id*, nama, *tipe_input*, *file*, *format_file*, dan url. Apabila pengguna hanya menggunakan input bertipe url, maka sistem hanya akan mengunggahnya ke dalam *database*. Jika *tipe_input* adalah *file*, maka *file* akan diunggah ke AWS S3 dengan nama *file* yang disusun dari waktu, nama berkas, dan kode pengguna. *File* diunggah ke AWS S3 menggunakan method *PUT*. Parameter pertama adalah *path* atau lokasi di dalam *bucket* S3 dimana *file* akan disimpan, dalam format *'berkas/<nama_unit>/<nama_file>'*. Parameter kedua adalah isi dari *file* tersebut, yang diambil menggunakan *file_get_contents($file)*.

Segmen Program 4.20 Metode *berkasUpload* pada *FormBerkasDataController*

```
public function berkasUpload(Request $request)
    {
        $user = Auth::user();
        $formatFile = $request->format_file;
        $formatFile = str_replace(' ', '', $formatFile);
        $formatFile = str_replace('.', '', $formatFile);
        // Validate the input data
        $request->validate([
            'tugas_akhir_id' => 'required',
            'form_berkas_id' => 'required|integer',
            'nama' => 'required|string',
            'tipe_input' => 'required|string',
            'file' => $request->input('tipe_input') !== 'url' ?
'required|file|mimes:' . $formatFile :'',
            'url' => $request->input('tipe_input') === 'url' ?
'required' : '',
        ]);

        $formberkasdata = FormBerkasData::where('tugas_akhir_id',
$request->tugas_akhir_id)->where('form_berkas_id', $request-
>form_berkas_id)->first();
```

```
        $data = '';

        $unit = Unit::where('id', session('unit_id'))->first();
        // Handle file upload or URL saving logic here
        if ($request->tipe_input == 'url') {
            // Save the URL
            $data = $request->url;
        } else {
            // Save the file
            if ($request->hasFile('file')) {
                $file = $request->file('file');
                $fileName = time() . '-' . $request->nama . '-' .
$user->user_kode->kode;
                $extension = $file->getClientOriginalExtension(); //
Get the original file extension
                $data = $fileName . '.' . $extension; // Append the
extension to the filename
                Storage::disk('s3')->put('berkas/' . $unit->nama .
'/' . $data, file_get_contents($file));
            }
        }

        if ($formberkasdata) {
            $formberkasdata->update([
                'data' => $data,
                'is_approve' => 0,
                'tanggal_approve' => null,
                'tugas_akhir_id' => $request->tugas_akhir_id,
                'form_berkas_id' => $request->form_berkas_id
            ]);
        } else {
            FormBerkasData::create([
                'data' => $data,
                'is_approve' => 0,
                'tanggal_approve' => null,
                'tugas_akhir_id' => $request->tugas_akhir_id,
                'form_berkas_id' => $request->form_berkas_id
            ]);
        }

        // Redirect back or to a success page
        return redirect()->back()->with('success', 'Berkas berhasil
diupload');
    }
```

Segmen Program 4.21 adalah fungsi *fileTugasAkhirUpload*, dalam kode ini juga bertujuan untuk mengunggah *file* ke AWS S3 *Storage*, mirip dengan fungsi *berkasUpload* yang sebelumnya. Fungsi ini juga memiliki logika tambahan untuk memperbarui status tugas akhir menjadi "SIAP SIDANG PROPOSAL" jika jenis periode adalah "proposal" dan status tugas akhir saat ini adalah "PENGERJAAN PROPOSAL".

Segmen Program 4.21 Metode *fileTugasAkhirUpload* pada *FormBerkasDataController*

```
public function fileTugasAkhirUpload(Request $request, string
$tugas_akhir)
    {
        $user =
User::find(session('login_web_59ba36addc2b2f9401580f014c7f58ea4e3098
9d'));

        // Validate the input data
        $request->validate([
            'tugas_akhir_id' => 'required',
            'periode_detail_id' => 'required',
            'file' => 'required|file|mimes:pdf',
        ]);

        $fileTugasAkhir = FileTugasAkhir::where('tugas_akhir_id',
$request->tugas_akhir_id)->where('periode_detail_id', $request-
>periode_detail_id)->first();
        $periodeDetail = PeriodeDetail::find($request-
>periode_detail_id);
        $data = '';

        // Save the file
        if ($request->hasFile('file')) {
            $file = $request->file('file');
            $fileName = time() . '-' . $periodeDetail->jenis . '-' .
$user->user_kode->kode;
            $extension = $file->getClientOriginalExtension(); // Get
the original file extension
            $data = $fileName . '.' . $extension; // Append the
extension to the filename
            Storage::disk('s3')->put('file_tugas_akhir/' .
$periodeDetail->periode->unit->nama . '/' . $data,
file_get_contents($file));
        }


        if ($fileTugasAkhir) {
            $fileTugasAkhir->update([
                'nama_file' => $data,
                'tugas_akhir_id' => $request->tugas_akhir_id,
                'periode_detail_id' => $request->periode_detail_id
            ]);
        } else {
            FileTugasAkhir::create([
                'nama_file' => $data,
                'tugas_akhir_id' => $request->tugas_akhir_id,
                'periode_detail_id' => $request->periode_detail_id
            ]);
            if ($periodeDetail->jenis == 'proposal') {

                $tugas_akhir = TugasAkhir::find($request-
>tugas_akhir_id);
                $statusSiapSidangProposal =
StatusTugasAkhir::where('nama', 'SIAP SIDANG PROPOSAL')->first();

                if ($tugas_akhir->status_tugas_akhir->nama ==
'PENGERJAAN PROPOSAL') {
                    $tugas_akhir->update([
```

```
                        'status_tugas_akhir_id' =>
$statusSiapSidangProposal->id
                    ]);
                }
            }
        }

        // Redirect back or to a success page
        return redirect()->back()->with('success', 'File berhasil
diupload');
    }
```

4.2.15 Metode *downloadBerkas* dan *downloadFileTugasAkhir* pada *FormBerkasDataController*

Segmen Program 4.22 adalah implementasi dari metode *downloadBerkas()* dan *downloadFileTugasAkhir()*. Kedua metode ini digunakan untuk mengunduh berkas yang sudah diunggah ke dalam penyimpanan AWS S3. Fungsi *downloadBerkas()* menangani unduhan berkas berdasarkan *tugas_akhir_id* dan *form_berkas_id*. Fungsi ini mencari entri yang sesuai di *database* melalui model *FormBerkasData*, kemudian membangun *path file* berdasarkan nama unit dan nama *file* yang diterima dari *request*. Jika *file* tersebut ada di S3, fungsi menentukan *MIME type file*, membuka *stream* ke *file* tersebut, dan mengembalikan respon berisi konten *file* yang diunduh. Menggunakan *Storage::disk('s3')->exists($filePath)* untuk mengecek apakah *file* ada di S3. Jika *file* tidak ditemukan maka fungsi akan mengembalikan respon *"404 Not Found"*.

Segmen Program 4.22 Metode *downloadBerkas* pada *FormBerkasDataController*

```
public function downloadBerkas(Request $request)
    {
        $fileName = $request->input('fileName');
        $tugas_akhir_id = $tugas_akhir;
        $form_berkas_id = $request->input('form_berkas_id');

        $formberkasdata = FormBerkasData::where('tugas_akhir_id',
$tugas_akhir_id)->where('form_berkas_id', $form_berkas_id)->first();

        // Retrieve the path to the file
        $filePath = 'berkas/' . $formberkasdata->form_berkas->unit-
>nama . '/' . $fileName;

        // Check if the file exists
        if (Storage::disk('s3')->exists($filePath)) {
            // Determine the file's MIME type
            $mimeType = Storage::disk('s3')->mimeType($filePath);

            // Open a stream to the file
            $fileStream = Storage::disk('s3')-
>readStream($filePath);

            // Return a streamed response with the file content
```

```
            return new StreamedResponse(function () use
($fileStream) {
                fpassthru($fileStream);
            }, 200, [
                'Content-Type' => $mimeType, // Set the appropriate
Content-Type based on the file's MIME type
                'Content-Disposition' => 'inline; filename="' .
$fileName . '"',
            ]);
        }
        // If the file does not exist, return a 404 Not Found
response
        abort(404);
    }
```

Segmen Program 4.23 adalah metode *downloadFileTugasAkhir*, dalam kode ini juga bertujuan untuk download *file* dari AWS S3 *Storage*, mirip dengan fungsi *downloadBerkas* yang sebelumnya. Fungsi ini menangani unduhan *file* tugas akhir berdasarkan *tugas_akhir_id* dan *periode_detail_id*. Fungsi ini mencari entri yang sesuai di *database* melalui model *FileTugasAkhir*.

Segmen Program 4.23 Metode *downloadFileTugasAkhir* pada *FormBerkasDataController*

```
public function downloadFileTugasAkhir(Request $request)
    {
        $fileName = $request->input('fileName');
        $tugas_akhir_id = $tugas_akhir;
        $periode_detail_id = $request->input('periode_detail_id');

        $fileTugasAkhir = FileTugasAkhir::where('tugas_akhir_id',
$tugas_akhir_id)->where('periode_detail_id', $periode_detail_id)-
>first();

        // Retrieve the path to the file
        $filePath = 'file_tugas_akhir/' . $fileTugasAkhir-
>periode_detail->periode->unit->nama . '/' . $fileName;

        // Check if the file exists
        if (Storage::disk('s3')->exists($filePath)) {
            // Determine the file's MIME type
            $mimeType = Storage::disk('s3')->mimeType($filePath);

            // Open a stream to the file
            $fileStream = Storage::disk('s3')-
>readStream($filePath);

            // Return a streamed response with the file content
            return new StreamedResponse(function () use
($fileStream) {
                fpassthru($fileStream);
            }, 200, [
                'Content-Type' => $mimeType, // Set the appropriate
Content-Type based on the file's MIME type
```

```
                'Content-Disposition' => 'inline; filename="' .
$fileName . '"',
            ]);
        }

        // If the file does not exist, return a 404 Not Found
response
        abort(404);
    }
```

4.2.16 Metode *ajukanMajuSidang* dan *ajukanPerpanjang* pada *TugasAkhirController*

Segmen Program 4.24 adalah salah satu metode dari *TugasAkhirController* yang berfungsi untuk mengubah status tugas akhir mahasiswa dari “PENGERJAAN TUGAS AKHIR” menjadi “SIAP SIDANG”. Fungsi ini menerima parameter ID *tugas_akhir* dari tugas akhir yang ingin diajukan untuk siap maju sidang. Setelah data berhasil diperbarui, metode ini akan mengarahkan pengguna ke halaman yang sesuai dengan variabel global *menuUrl* dan mengirimkan notifikasi sukses "Topik berhasil diajukan untuk maju sidang akhir".

Segmen Program 4.24 Metode *ajukanMajuSidang* pada *TugasAkhirController*

```
public function ajukanMajuSidang(string $tugas_akhir)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));

        // Decrypt data ids
        $tugas_akhir_id = decrypt($tugas_akhir);

        // Update status tugas akhir to 'SIAP SIDANG AKHIR'
        $tugas_akhir = TugasAkhir::find($tugas_akhir_id);

        $statusSiapSidangAkhir = StatusTugasAkhir::where('nama',
'SIAP SIDANG AKHIR')->first();
        $tugas_akhir->status_tugas_akhir_id =
$statusSiapSidangAkhir->id;

        $tugas_akhir->save();

        return redirect()->back()->with('success', 'Topik berhasil
diajukan untuk maju sidang akhir');
    }
```

Segmen Program 4.25 berfungsi untuk mengubah status tugas akhir mahasiswa dari “PENGERJAAN TUGAS AKHIR” menjadi “PERPANJANG”. Sama seperti fungsi *ajukanMajuSidang*, fungsi ini menerima parameter ID *tugas_akhir* dari tugas akhir yang ingin diperpanjang. Setelah data berhasil diperbarui, metode ini akan mengarahkan pengguna ke halaman yang sesuai dengan variabel global *menuUrl* dan mengirimkan notifikasi sukses "Topik berhasil diperpanjang".

Segmen Program 4.25 Metode *ajukanPerpanjang* pada *TugasAkhirController*

```
public function ajukanPerpanjang(string $tugas_akhir)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));

        // Decrypt data ids
        $tugas_akhir_id = decrypt($tugas_akhir);

        // Update status tugas akhir to ''
        $tugas_akhir = TugasAkhir::find($tugas_akhir_id);

        $statusPerpanjang = StatusTugasAkhir::where('nama',
'PERPANJANG')->first();
        $tugas_akhir->status_tugas_akhir_id = $statusPerpanjang->id;

        $tugas_akhir->save();

        return redirect()->back()->with('success', 'Topik berhasil
diperpanjang');
    }
```

4.2.17 Metode *batalPeserta*, *batalPembimbing*, dan *batalKoordinator* pada *TugasAkhirController*

Segmen Program 4.26 merupakan bagian dari *TugasAkhirController* yang mengatur proses pembatalan tugas akhir oleh mahasiswa. Sebelum melakukan perubahan pada status tugas akhir, dilakukan pengecekan terlebih dahulu terhadap status peserta. Jika peserta telah mengajukan pembatalan (*is_batal = true*), proses pembatalan akan menunggu persetujuan dari dosen pembimbing masing-masing sebelum status tugas akhir dapat diubah menjadi gagal. Persetujuan dari semua dosen pembimbing diperlukan sebelum status akhir tugas dapat berubah. Selain itu, apabila status tugas akhir masih berada dalam tahap “PENGAJUAN TOPIK”, peserta memiliki wewenang untuk membatalkan tugas akhir secara langsung tanpa perlu menunggu persetujuan dari dosen pembimbing.

Segmen Program 4.26 Metode *batalPeserta* pada *TugasAkhirController*

```
public function batalPeserta(Request $request)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));
        $pesertaList = Peserta::where('tugas_akhir_id',
decrypt($request['id']))->get();
        foreach ($pesertaList as $peserta) {
            $peserta->update([
                'is_batal' => true
            ]);
        }
```

```
        if ($pesertaList->first()->tugas_akhir->status_tugas_akhir-
>nama == 'PENGAJUAN TOPIK') {
            $this->batalKoordinator($request);
        }
        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Pembatalan berhasil diajukan... Silahkan menunggu
persetujuan dari seluruh Dosen Pembimbing');
    }
```

Metode *batalPembimbing()* pada Segmen Program 4.27 bertanggung jawab untuk memproses pembatalan tugas akhir oleh dosen pembimbing terhadap mahasiswa. Status pembimbing diperbarui menjadi *is_batal* yang bernilai "*true*", menandakan pembatalan tugas akhir oleh pembimbing tersebut. Metode ini juga melakukan pengecekan terhadap status pembimbing lain yang terkait dengan tugas akhir yang sama. Jika semua pembimbing telah membatalkan tugas akhir, metode akan melanjutkan proses pembatalan dengan memanggil metode pada Segmen Program 4.28 yang akan merubah status tugas akhir menjadi batal.

Segmen Program 4.27 Metode *batalPembimbing* pada *TugasAkhirController*

```
public function batalPembimbing(Request $request)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));
        $user = Auth::user();
        $dosen_id = $user->user_kode->pegawai->id;
        $pembimbing = Pembimbing::where('tugas_akhir_id',
decrypt($request['id']))->where('dosen_id', $dosen_id)->first();
        $pembimbing->update([
            'is_batal' => true
        ]);
        $all_batal = true;
        foreach ($pembimbing->tugas_akhir->pembimbing as
$pembimbing) {
            if ($pembimbing->is_batal == false && $pembimbing-
>status == 'internal') {
                $all_batal = false;
            }
        }
        if ($all_batal ||  isset($request['batalLangsung'])) {
            $this->batalKoordinator($request);
        }
        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Pembatalan berhasil diterima');
    }
```

Segmen Program 4.28 bertujuan untuk memproses pembatalan tugas akhir oleh koordinator terhadap mahasiswa yang terlibat. metode ini mengambil daftar peserta yang terkait dengan tugas akhir tertentu dari *database* berdasarkan ID tugas akhir yang didekripsi dari *request* yang diterima. Untuk setiap peserta dalam daftar, status pembatalan (*is_batal*) mereka

diperbarui menjadi "*true*" dan status aktif (*is_active*) mereka diubah menjadi "*false*", menandakan bahwa mereka telah membatalkan partisipasi dalam tugas akhir tersebut.

Setelah memperbarui status peserta, metode ini mengambil daftar pembimbing yang terlibat dalam tugas akhir yang sama. Jika terdapat pembimbing yang terkait, status pembatalan mereka juga diperbarui menjadi "*true*", menunjukkan bahwa mereka telah membatalkan pembimbingan terhadap tugas akhir tersebut. Selanjutnya, status tugas akhir itu sendiri diubah menjadi "BATAL" dalam *database* dengan memperbarui kolom *status_tugas_akhir_id* berdasarkan ID status yang diambil dari tabel *status_tugas_akhir*.

Segmen Program 4.28 Metode *batalKoordinator* pada *TugasAkhirController*

```
public function batalKoordinator(Request $request)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));
        $pesertaList = Peserta::where('tugas_akhir_id',
decrypt($request['id']))->get();
        foreach ($pesertaList as $peserta) {
            $peserta->update([
                'is_batal' => true,
                'is_active' => false
            ]);
        }
        $pembimbingList = Pembimbing::where('tugas_akhir_id',
decrypt($request['id']))->get();
        if (isset($pembimbingList)) {
            foreach ($pembimbingList as $pembimbing) {
                $pembimbing->update([
                    'is_batal' => true
                ]);
            }
        }
        $statusBatal = StatusTugasAkhir::where('nama', 'BATAL')-
>first();
        $tugas_akhir = TugasAkhir::find(decrypt($request['id']));
        $tugas_akhir->update([
            'status_tugas_akhir_id' => $statusBatal->id
        ]);
        return redirect('/' . $this->globalVariable->menuUrl)-
>with('success', 'Pembatalan berhasil');
    }
```

4.2.18 Metode *approvePengajuan* pada *TugasAkhirController*

Segmen Program 4.29 berfungsi mengatur proses persetujuan dari dosen pembimbing terhadap pengajuan proposal bimbingan yang diajukan oleh mahasiswa. Ketika pembimbing menyetujui pengajuan ini, status *is_approve* dari pembimbing yang bersangkutan akan diperbarui menjadi "*true*", menandakan bahwa dosen tersebut menyetujui pengajuan

bimbingan oleh mahasiswa. Selanjutnya, daftar pembimbing yang terlibat dalam tugas akhir tersebut diambil dan dilakukan pengecekan apakah semua pembimbing telah menyetujui tugas akhir tersebut. Jika semua pembimbing telah menyetujui, maka status tugas akhir diperbarui menjadi “PENGERJAAN PROPOSAL”, menandakan bahwa proposal telah disetujui oleh semua dosen pembimbing yang terlibat.

Segmen Program 4.29 Metode *approvePengajuan* pada *TugasAkhirController*

```
public function approve(string $tugas_akhir, string $pembimbing)
    {
        // Get module global for each role
        $this->getModuleGlobal(role: session('current_role'));
        // Decrypt data ids
        $tugas_akhir_id = decrypt($tugas_akhir);
        $pembimbing_id =  decrypt($pembimbing);
        // Update is_approve pembimbing
        $pembimbing = Pembimbing::find($pembimbing_id);
        $pembimbing->is_approve = true;
        $pembimbing->save();
        $listPembimbing = Pembimbing::where('tugas_akhir_id',
$tugas_akhir_id)->get();
        $allApproved = true;
        // Check if all pembimbing is approved
        foreach ($listPembimbing as $pembimbing) {
            if (!$pembimbing->is_approve) {
                $allApproved = false;
                break;
            }
        }
        // Update status tugas akhir if all pembimbing is already
approved
        if ($allApproved) {
            $tugas_akhir = TugasAkhir::find($tugas_akhir_id);
            if ($tugas_akhir && $tugas_akhir->status_tugas_akhir_id
< 2) {
                $tugas_akhir->status_tugas_akhir_id = 2;
                $tugas_akhir->save();
            }
        }
        return redirect()->back()->with('success', 'Pengajuan
bimbingan telah anda setujui');
    }
```

4.2.19 Metode *upadatePembimbing* dan *updatePeserta* pada *TugasAkhirController*

Segmen Program 4.30 adalah metode yang digunakan untuk memperbarui data pembimbing pada sebuah tugas akhir. Ada tiga jenis perubahan yang dapat terjadi pada data pembimbing: penambahan dosen pembimbing, penghapusan dosen pembimbing, dan penggantian dosen pembimbing tanpa menambah atau menghapus. Metode ini bertujuan untuk menangani ketiga kondisi tersebut. Sebelum melakukan perubahan, data pembimbing

sebelumnya disimpan dalam sebuah *array* untuk mencatat status *is_approve* dari dosen pembimbing yang sudah ada. Metode ini kemudian melakukan iterasi sebanyak jumlah *request input* pembimbing. Jika pembimbing sudah ada, maka data tersebut akan diperbarui. Jika dosen pembimbing belum ada atau ada penambahan dosen pembimbing, maka data pembimbing baru akan dibuat. Terakhir, dilakukan pengecekan untuk menghapus pembimbing yang tidak ada dalam input terbaru jika jumlahnya lebih sedikit daripada jumlah pembimbing lama.

Segmen Program 4.30 Metode *updatePembimbing* pada *TugasAkhirController*

```
public function updatePembimbing($data, $tugas_akhir)
    {
        $existingPembimbings = Pembimbing::where('tugas_akhir_id',
$tugas_akhir->id)->get()->keyBy('pembimbing_ke');

        // Digunakan untuk update status is_approve untuk dosen yang
sudah ada sebelumnya
        $existingPembimbingApproved = [];
        foreach ($existingPembimbings as $pembimbing) {
            // Ambil dosen_id dari pembimbing yang statusnya sudah
approve
            if ($pembimbing->is_approve && $pembimbing->status ==
'internal') {
                $existingPembimbingApproved[] = $pembimbing-
>dosen_id;
            }
        }

        // Array untuk menyimpan pembimbing_ke yang ada di input
terbaru
        $pembimbingKeInInput = [];

        // Update pembimbing atau buat baru
        foreach ($data["pembimbing"] as $index => $pembimbing) {
            $pembimbingKeInInput[] = $index;

            // Periksa apakah ada pembimbing dengan pembimbing_ke
yang sama
            $currentPembimbing = $existingPembimbings->get($index);

            if ($currentPembimbing) {
                // Update pembimbing yang ada
                if (isset($pembimbing['internal'])) {
                    $dosenId = decrypt($pembimbing['internal']);
                    $currentPembimbing->update([
                        'status' => 'internal',
                        'dosen_id' => $dosenId,
                        'nama' => null,
                        'is_approve' => in_array($dosenId,
$existingPembimbingApproved) ? true : false // Update sesuai dengan
status is_approve sebelumnya
                    ]);

                    // Update bobot sks jika bobot sks diinputkan
                    if (isset($pembimbing['sks'])) {
```

```
                        $currentPembimbing->update([
                            'bobot_sks' => $pembimbing['sks'],
                        ]);
                    }
                } elseif (isset($pembimbing['eksternal'])) {
                    $currentPembimbing->update([
                        'status' => 'eksternal',
                        'dosen_id' => null,
                        'nama' => $pembimbing['eksternal'],
                        'is_approve' => true
                    ]);
                }
            } else {
                // Buat pembimbing baru bila ada penambahan
pembimbing
                if (isset($pembimbing['internal'])) {
                    $dosenId = decrypt($pembimbing['internal']);
                    Pembimbing::create([
                        'pembimbing_ke' => $index,
                        'status' => 'internal',
                        'dosen_id' => $dosenId,
                        'bobot_sks' => isset($pembimbing['sks']) ?
$pembimbing['sks'] : 0,
                        'tugas_akhir_id' => $tugas_akhir->id,
                        'is_approve' => false
                    ]);
                } elseif (isset($pembimbing['eksternal'])) {
                    Pembimbing::create([
                        'pembimbing_ke' => $index,
                        'status' => 'eksternal',
                        'nama' => $pembimbing['eksternal'],
                        'tugas_akhir_id' => $tugas_akhir->id,
                        'is_approve' => true
                    ]);
                }
            }
        }

        // Hapus pembimbing yang tidak ada dalam input terbaru
(pengurangan pembimbing)
        foreach ($existingPembimbings as $existingPembimbing) {
            if (!in_array($existingPembimbing->pembimbing_ke,
$pembimbingKeInInput)) {
                $existingPembimbing->delete();
            }
        }
    }
```

Sama seperti metode *updatePembimbing*, Segmen Program 4.31 digunakan untuk memperbarui data peserta dari tugas akhir. Langkah pertama adalah mengambil data *mahasiswa_id* dari peserta yang terkait dengan tugas akhir dan menyimpannya ke dalam *array existingPesertas*. Kemudian, *mahasiswa_id* dari data request juga disimpan ke dalam *array mahasiswa_ids*. Selanjutnya, dilakukan iterasi untuk membandingkan data peserta yang lama dengan data baru berdasarkan *mahasiswa_id*. Jika *mahasiswa_id* belum ada sebelumnya, maka

dibuat data peserta baru. Terakhir, data peserta dengan *mahasiswa_id* yang tidak ada dalam *array mahasiswa_ids* baru akan dihapus.

Segmen Program 4.31 Metode *updatePeserta* pada *TugasAkhirController*

```
public function updatePeserta($data, $tugas_akhir)
    {
        $existingPesertas = Peserta::where('tugas_akhir_id',
$tugas_akhir->id)->get()->keyBy('mahasiswa_id');
        $mahasiswa_ids = [decrypt($data["mahasiswa_id"])];

        if (!empty($data["mahasiswa_partner"])) {
            $mahasiswa_ids[] = decrypt($data["mahasiswa_partner"]);
        }

        foreach ($mahasiswa_ids as $mahasiswa_id) {
            // Create peserta baru bila peserta belum ada sebelumnya
(ganti peserta)
            if (!$existingPesertas->has($mahasiswa_id)) {
                Peserta::create([
                    'mahasiswa_id' => $mahasiswa_id,
                    'tugas_akhir_id' => $tugas_akhir->id,
                ]);
            }
        }

        foreach ($existingPesertas as $existingPeserta) {
            if (!in_array($existingPeserta->mahasiswa_id,
$mahasiswa_ids)) {
                $existingPeserta->delete();
            }
        }
  }
```

4.2.20 Metode *getUnitIdAttribute* pada Model Unit

Segmen Program 4.32 adalah bagian dari Model Unit yang berfungsi untuk menentukan apakah suatu unit merupakan unit *parent* atau unit *child*. Metode *getUnitIdAttribute()* akan memeriksa status unit tersebut. Jika metode *isParent()* mengembalikan nilai "*true*", maka unit tersebut adalah unit *parent*, dan fungsi akan mengembalikan ID unit tersebut. Namun sebaliknya, jika unit tersebut adalah unit *child*, fungsi akan mengembalikan ID *parent* dari unit tersebut menggunakan *parent_id*. Metode ini digunakan untuk menampilkan master data sesuai dengan unit atau program studi seperti jenis sidang, komposisi nilai, bidang studi dan lain-lain. Sebagai contoh, mahasiswa jurusan Sistem Informasi Bisnis akan mengambil master data dari unit *parent* karena jurusan tersebut merupakan *child* dari Program Studi Informatika. Master data yang dibuat oleh koordinator disimpan dengan *unit_id* dari Program Studi Informatika.

Segmen Program 4.32 Metode *getUnitIdAttribute* pada Model Unit

```
public function isParent()
{
        return $this->childs()->exists();
}

// Method untuk mendapatkan parent unit jika unit ini child
public function parent()
{
        return $this->belongsTo(Unit::class, 'parent_id');
}

// Method untuk mendapatkan child units jika unit ini parent
public function childs()
{
        return $this->hasMany(Unit::class, 'parent_id');
}

// Method untuk mendapatkan id unit sesuai kebutuhan (parent_id atau
id)
public function getUnitIdAttribute()
{
        return $this->isParent() ? $this->id : $this->parent_id;
}
```

4.2.21 Metode *generatePDF*

Segmen Program 4.33 digunakan untuk membuat objek PDF menggunakan *jsPDF*, lalu mengambil tangkapan layar dari elemen "*content*" menggunakan *html2canvas*. Gambar yang dihasilkan kemudian ditambahkan ke objek PDF. Jika konten melampaui satu halaman, halaman baru ditambahkan secara otomatis. Setelah PDF selesai, skrip menghasilkan URL untuk PDF dan membukanya di *tab* baru dengan nama *file* yang ditambahkan sebelumnya. Terakhir, *file* PDF dapat disimpan dengan nama yang diambil dari elemen "*filename*".

Segmen Program 4.33 Metode JavaScript untuk *Generate* PDF

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.mi
n.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/1.4.1/html2c
anvas.min.js">
</script>
    <script>
        function generatePDF(filename) {
            const {
                jsPDF
            } = window.jspdf;
            const doc = new jsPDF();
```

```
html2canvas(document.querySelector("#content")).then(canvas => {
                const imgData = canvas.toDataURL("image/png");
                const imgWidth = 190;
                const pageHeight = 295;
                const imgHeight = canvas.height * imgWidth /
canvas.width;
                let heightLeft = imgHeight;
                let position = 10;

                doc.addImage(imgData, 'PNG', 10, position, imgWidth,
imgHeight);
                heightLeft -= pageHeight - position;

                while (heightLeft >= 0) {
                    position = heightLeft - imgHeight;
                    doc.addPage();
                    doc.addImage(imgData, 'PNG', 10, position,
imgWidth, imgHeight);
                    heightLeft -= pageHeight;
                }

                // Create a Blob from the PDF and generate a URL
                const pdfBlob = doc.output('blob');
                const pdfUrl = URL.createObjectURL(pdfBlob);

                // Open the PDF in a new tab
                const newTab = window.open(pdfUrl, '_blank');
                if (newTab) {
                    // Set the suggested filename when the user
tries to save the PDF
                    newTab.document.title = filename;
                } else {
                    console.error("Failed to open a new tab.");
                }

                // Save the PDF as a file
                doc.save(filename + '.pdf');
            });
        }


        document.addEventListener("DOMContentLoaded", function() {
            const filenameElement =
document.getElementById("filename");
            if (filenameElement) {
                const filename = filenameElement.innerText;
                generatePDF(filename); // Call generatePDF function
with filename
            } else {
                console.error("Element with ID 'filename' not
found.");
            }
            // Call window.history.back() after a slight delay
            setTimeout(function() {
                window.history.back();
            }, 500);
        });
```

```
    </script>
```