4. IMPLEMENTASI SISTEM

4.1. *Tools* dan *Library* yang digunakan untuk implementasi

Aplikasi *Web* Sistem TA ini dibuat dengan menggunakan *Framework* Laravel 8, Bahasa pemrograman HTML, CSS dan PHP. Dalam Penggunaannya HTML digunakan untuk penyusunan Aplikasi, *Back End* akan menggunakan PHP yang dimana digunakan untuk mengatur data antara *database* dan program yang tersedia, JavaScript dengan *library* JQuery akan digunakan sebagai desain dalam pengaturan dinamis *input* yang ada pada program, *middleware* akan digunakan untuk proses autentifikasi *login* yang disertai dengan *role*, yang dimana ada 3 *role* berbeda yaitu admin(koordinator), dosen, dan mahasiswa.

4.2. Langkah - langkah proses implementasi

Langkah-langkah untuk proses implementasi akan dijabarkan pada *point-point* berikut ini:

1. Meng*install* composer dari *website* https://getcomposer.org/
2. Membuat *project* laravel dengan menggunakan GIT

```
composer create-project laravel/laravel website_ta_elektro_laravel_8
```

3. Meng*import library* ui/auth dari laravel untuk login menggunakan GIT

```
composer require laravel ui/auth
```

4. Meng*install* CSS *Scaffolding* dan Javascript untuk ui/auth menggunakan GIT

```
php artisan ui bootstrap –auth
```

5. Menjalankan *server* menggunakan *command* pada terminal

```
php artisan serve
```

Untuk mengimplementasi *library* lainnya yang digunakan pada aplikasi ini sebagian *library* yang digunakan di *download* dari *website* penyedia *library* ada juga yang menggunakan *link* cdn untuk memanggil *library* nya yang ada kemudian dimasukkan ke dalam folder *assets* yang nantinya akan dipanggil seperti pada segmen program di bawah ini.

4.2.1. Koneksi Ke Database

Agar terhubung dengan sistem di *database* MySQL. Maka diperlukan konfigurasi sistem di bagian koneksinya. Maka pengaturan programnya akan sebagai berikut:

Segmen Program 4. 1 *Source Code* Koneksi ke Database

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=web_ta_elektro
DB_USERNAME=root
DB_PASSWORD=
```

4.3. Hubungan antara Desain dan Implementasi

Pada bab sebelumnya, terdapat desain sistem yang menjelaskan fitur dan proses apa saja yang terdapat di dalam aplikasi. Kemudian desain sistem tersebut akan diimplementasikan kedalam pembuatan program dalam bentuk beberapa segmen. Pembagian segmen program akan dipetakan pada tabel 4.1.

Tabel 4.1 Tabel keterangan hubungan segmen program dan *activity diagram*

Flowchart/activity diagram	Segmen Program	Proses
-	4.1 Import library	Import library
-	4.2 Koneksi ke Database	Koneksi ke database
Gambar 3.15	4.3 Login	Login
Gambar 3.16	4.3.6.1Pengumuman Controller>index	Menampilkan Pengumuman
Gambar 3.17	4.3.6.2Pengumuman Controller>store	Menambah Pengumuman
Gambar 3.18	4.3.1.1.Jadwal Sidang	Menampilkan Mahasiswa

	Controller>index	dengan SubTopik
-	4.3.1.1.Jadwal Sidang Controller>store	Store Topik
Gambar 3.19		
Gambar 3.20	4.3.5.1.Mahasiswa Controller >show	Menampilkan List Mahasiswa Proposal
Gambar 3.21	4.3.5.1.Mahasiswa Controller >show_akhir	Menampilkan List Mahasiswa Tugas Akhir
Gambar 3.21	4.3.14.1 Topik Controller >index	Menampilkan Form Topik
-	4.3.14.1 Topik Controller >Store	Store Topik
Gambar 3.22	4.3.7.1 Format Penilaian> index	Menampilkan Format Penilaian
-	4.3.7.3 Format Penilaian> store	Store Format Penilaian
Gambar 3.22	4.3.16.1 Format Penilaian Akhir> index	Menampilkan Format Penilaian Akhir
	4.3.16.3 Format Penilaian Akhir > store	Store Format Penilaian Akhir
Gambar 3.22	4.3.3.1 Penilaian Controller> index	Menampilkan Penilaian untuk Dosen
	4.3.3.2 Penilaian Controller>	Store Penilaian Ke Database

	store	
Gambar 3.22	4.3.17.1 Penilaian Akhir Controller> index	Menampilkan Penilaian untuk Dosen
-	4.3.17.2 Penilaian Akhir Controller> store	Store Penilaian Ke Database
-	4.3.3.3 Penilaian Controller> show	Show Penilaian Ke Admin (Koordinator)
-	4.3.17.3 Penilaian Akhir Controller> show	Show Penilaian Akhir Ke Admin (Koordinator)
Gambar 3.23	4.3.4.1. Jadwal Sidang Berhalangan Controller >index	Menampilkan Form untuk Upload Sidang Berhalangan
	4.3.4.3 Jadwal Sidang Berhalangan Controller>store	Menstore Jadwal Berhalangan Dosen
Gambar 3.24	4.3.1.1.Jadwal Sidang> index	Menampilkan Penjadwalan Sidang
-	4.3.1.2 Jadwal Sidang> store	Store Penjadwalan Sidang
-	4.3.1.3 Jadwal Sidang> show_mahasiswa_store	Menampilkan Penjadwalan Sidang di Dosen
-	4.3.18.2 Jadwal Sidang> store	Store Penjadwalan Sidang Akhir
-	4.3.18.3 Jadwal Sidang> show_mahasiswa_store	Menampilkan Penjadwalan Sidang di Dosen
-	4.3.14.4 Topik Controller > Show	Menampilkan Topik untuk Mahasiswa
-	4.3.14.5 Topik Controller >	Menampilkan Detail Topik

	show_detail	yang akan dipilih untuk Mahasiswa yang bisa memilih Topik
-	4.3.8.3 Mahasiswa Controller > destroy	Men cancel Topik yang di pilih
-	4.3.14.3Topik Controller>show_auth	Menampilkan Detail Topik yang akan dipilih.untuk Dosen
-	4.3.8.2 Riwayat Mahasiswa >store	MenStore Request Topik dari mahasiswa

4.3.1. Jadwal Sidang *Controller*

Bagian ini ditujukan untuk mengontrol sistem jadwal sidang yang digunakan oleh admin, dosen dan mahasiswa. Di dalam sistem Administrasi (yang dikelola oleh Koordinator), semua jadwal sidang akan terlihat, sedangkan di dalam sistem Dosen, jadwal sidang hanya akan ditampilkan jika Dosen tersebut berperan sebagai Pembimbing atau Penguji. Sementara itu, di dalam sistem Mahasiswa, jadwal sidang akan muncul sesuai dengan Judul Tugas Akhir (TA) yang telah diajukan kepada Dosen.

4.3.1.1. index

Segmen Program 4.2 JadwalSidangController > index

```
  public function index(string $id)
    {
        $skripsi=Skripsi::with('riwayat_mahasiswa')-
>find($id);
        $dosen=User::all()->where('role','=','2');

$dosen_pembimbing_tambahan=DosenPembimbingTambahan::
all()->where('id_skripsi','=',$skripsi->id);

        $merge_berhalangan=[];
        //
$merge_berhalangan=$dosen_pembimbing_tambahan-
>toBase()->
```

```
        $tmp=$skripsi->riwayat_mahasiswa->topik-
>user->jadwal_sidang_berhalangan;
        $merge_berhalangan = $tmp->toBase()-
>merge($merge_berhalangan);

        if($dosen_pembimbing_tambahan)
        foreach($dosen_pembimbing_tambahan as
$item){

            // dd($item->user-
>jadwal_sidang_berhalangan);
        $merge_berhalangan = $item->user-
>jadwal_sidang_berhalangan->toBase()-
>merge($merge_berhalangan);

        }
        // dd($merge_berhalangan);

        return
view('features.admin.post_penjadwalan_sidang_akhir',
[
            "title"=>"Penjadwalan Sidang Akhir",
            "skripsi"=>$skripsi,
            "dosen"=>$dosen,
            "dosen_pembimbing_tambahan" =>
$dosen_pembimbing_tambahan,
            "merge_berhalangan" =>
$merge_berhalangan
        ]);
    }
```

4.3.1.2 store

Segmen Program 4.3 JadwalSidangController > store

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



        // dd($request->all());
        $request->validate([
            "judul_ta" => 'required',
            "id_skripsi"=>'required',
            "nama_mahasiswa" => 'required',
            "nip_mahasiswa" => 'required',
            "ipk" => "required",
```

```
            "dosen_penguji.*"=>"required",
            "hari_pelaksanaan" => 'required',
            "jam_pelaksanaan" => 'required',
            "jam_selesai" => 'required',
        ]);


        // dd($request->all());
        $jadwal_sidang= new JadwalSidang();

        $jadwal_sidang->id_skripsi=$request-
>id_skripsi;
        $jadwal_sidang->hari_pelaksanaan=$request-
>hari_pelaksanaan;
        $jadwal_sidang->jam_pelaksanaan=$request-
>jam_pelaksanaan;
        $jadwal_sidang->jam_selesai=$request-
>jam_selesai;
        $jadwal_sidang->save();


        $penguji=$request->dosen_penguji;
        //peguji di jadwalkan
        foreach($penguji as $pj){

            $dosen_sidang= new DosenSidang();
            $dosen_sidang-
>id_jadwal_sidang=$jadwal_sidang->id;
            $dosen_sidang->id_dosen=strtok($pj,' -
');
            $dosen_sidang->save();

        }

        //sesi penambahan jam berhalangan
```

```
        $jbs= explode(":",$request->jam_selesai);
        $jbm= explode(":",$request-
>jam_pelaksanaan);


        //Penguji Berhalangan di tambahkan
         foreach($penguji as $pj){


$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
            while (date("H:i",$timem) <=
date("H:i",$times)){

                $jsb=new JadwalSidangBerhalangan();
                $jsb->hari=$request-
>hari_pelaksanaan;
                $jsb->jam=date("H:i",$timem);
                $jsb->user_nip=strtok($pj,' -');
                $jsb->save();
                $timem=strtotime('+30
minutes',$timem);

            }
         }
         //Pembimbing tambahan berhalangan di
tambahkan
         foreach($request-
>dosen_pembimbing_tambahan as $pj){


$times=strtotime("$jbs[0]".":"."$jbs[1]");
```

```
$timem=strtotime("$jbm[0]".":"."$jbm[1]");
            while (date("H:i",$timem) <=
date("H:i",$times)){

                $jsb=new JadwalSidangBerhalangan();
                $jsb->hari=$request-
>hari_pelaksanaan;
                $jsb->jam=date("H:i",$timem);
                $jsb->user_nip=$pj;
         $jsb->save();
                $timem=strtotime('+30
minutes',$timem);

            }
         }
            //Pembimnbing berhalangan di tambahkan




$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
                while (date("H:i",$timem) <=
date("H:i",$times)){

                    $jsb=new
JadwalSidangBerhalangan();
                    $jsb->hari=$request-
>hari_pelaksanaan;
                    $jsb->jam=date("H:i",$timem);
                    $jsb->user_nip=$request-
>dosen_pembimbing_id;
                    $jsb->save();
```

```
                    $timem=strtotime('+30
minutes',$timem);

                }

         //

        return redirect('/get_skripsi_mahasiswa')-
>with("success","Topik $request->judul_ta Berhasil
Dijadwalkan");
        // dd($request->all());

    }
```

4.3.2. Penilaian *Controller*

Segmen ini gunakan untuk mengelola bagian penilaian dalam program. Fitur penilaian ini bertujuan untuk melakukan evaluasi terhadap mahasiswa dalam proses sidang proposal. Evaluasi ini melibatkan beberapa kriteria dan *sub*kriteria yang memiliki bobot yang berbeda di setiap kriteria.

4.3.2.1 index

Segmen Program 4.4 PenilaianController > index

```
 public function index(string $id)
    {


        $skripsi =  Skripsi::all()->find($id);

$format_penilaian=FormatPenilaian::with('format_sub
_penilaian')->get();
        return
view('features.dosen.post_penilaian_dosen',[
            'title' => 'Penilaian',
```

```
            'skripsi'=>$skripsi,
            'format_penilaian' => $format_penilaian
        ]);

    }
```

4.3.2.2 store_nilai_proposal

Segmen Program 4.5 PenilaianController > store_nilai_proposal

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

        // $format_penilaian
        // dd($request->all());

        foreach($request->id_nilai as $base =>
$key){
            foreach($request->id_sub_nilai[$base]
as $base1 =>$key1){
                $penilaian= new Penilaian();
                $penilaian->user_id=Auth::user()-
>nip;
                $penilaian->skripsi_id=$request-
>id;
                $penilaian-
>format_penilaian_id=$key;
                $penilaian-
>format_sub_penilaian_id=$key1;
                $penilaian->point=$request-
>point[$base][$base1];
                $penilaian->sidang_proposal=1;
                $penilaian->save();
            }
        }
        return
```

```
redirect('/get_jadwal_sidang_mahasiswa_dosen');
    }
```

4.3.3 Jadwal Sidang Berhalang *Controller*

Bagian ini bermanfaat untuk mengendalikan segmen jadwal sidang berhalang. Pengajuan jadwal berhalangan dalam skripsi ditujukan kepada dosen-dosen yang mengalami keterbatasan waktu atau kendala lainnya. Sistem ini dirancang untuk membantu koordinator dalam mengatur jadwal yang melibatkan dosen-dosen yang sedang berhalangan.

4.3.3.1 index

Segmen Program 4.6 JadwalSidangBerhalangController > index

```
public function index()
    {
        $jsb=new JadwalSidangBerhalangan();



        $jadwal_sidang_berhalangan= $jsb->all()-
>where('user_nip' ,'=',Auth::user()->nip);

        return
view('features.dosen.post_jadwal_berhalangan',[
                    'title'=> 'Pengajuan Jadwal
Berhalangan',
                    'jadwal_sidang_berhalangan'=>
$jadwal_sidang_berhalangan,
        ]);


    }
```

4.3.3.2 store

Segmen Program 4.7 JadwalSidangBerhalangController > store

```
public function store(Request $request)
    {
        $request->validate([
            "hari_berhalangan" => 'required',
            "jam_berhalangan_mulai"=>'required'
        ]);

        if(($request->jam_berhalangan_selesai ?? ''
)== ''){
            $jsb=new JadwalSidangBerhalangan();
            $jsb->hari=$request->hari_berhalangan;
            $jsb->jam=$request-
>jam_berhalangan_mulai;
            $jsb->user_nip=Auth::user()->nip;
            $jsb->save();
        }
        else{
           $jbs= explode(":",$request-
>jam_berhalangan_selesai);
           $jbm= explode(":",$request-
>jam_berhalangan_mulai);


$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
            while (date("H:i",$timem) <=
date("H:i",$times)){

                $jsb=new JadwalSidangBerhalangan();
                $jsb->hari=$request-
>hari_berhalangan;
                $jsb->jam=date("H:i",$timem);
```

```
                $jsb->user_nip=Auth::user()->nip;
                $jsb->save();
                $timem=strtotime('+30
minutes',$timem);

            }
        }



        return redirect()->back();
    }
```

4.3.4 Format Penilaian *Controller*

Bagian ini bertujuan untuk mengatur format penilaian yang digunakan oleh koordinator, sesuai dengan bobot dan kriteria yang ditetapkan. Hal ini melibatkan koneksi dengan *database* format dan *sub*-format yang digunakan sebagai referensi. Dengan ini, koordinator dapat dengan mudah mengatur dan mengkonfigurasi format penilaian yang sesuai dengan kebutuhan.

4.3.4.1 index

Segmen Program 4.8 FormatPenilaianController > index

```
    public function index()
    {
        return
view('features.admin.post_format_penilaian',[
            'title' => 'Format Penilaian'
        ]);
    }
```

4.3.4.2 store

Segmen Program 4.9 FormatPenilaianController > store

```
 public function store(Request $request)
    {
        $request->validate([
            'judul_nilai'=> 'required',
            'bobot_nilai'=> 'required',
            'judul_sub_nilai'=> 'required'
        ]);
        // dd($request->all());
        FormatPenilaian::truncate();
        SubFormat::truncate();
        $judul_nilai=$request->judul_nilai;
        $bobot_nilai=$request->bobot_nilai;
        $judul_sub_nilai=$request->judul_sub_nilai;
        foreach($judul_nilai as $key => $item){
            $formatPenilaian=new FormatPenilaian();
            $formatPenilaian-
>judul_nilai=$judul_nilai[$key];
            $formatPenilaian-
>bobot_nilai=$bobot_nilai[$key];
            $formatPenilaian->save();
            foreach($judul_sub_nilai[$key] as $key1
=> $item1){
                $subFormat=new SubFormat();
                $subFormat-
>format_penilaian_id=$formatPenilaian->id;
                $subFormat-
>judul_sub_format=$item1;
                $subFormat->save();

            }

        }
        // dd('selesai');
```

```
        return redirect()->back()-
>with('success','Format Berhasil di Uploud');

    }
```

4.3.5 *Login Controller*

Fitur *login* digunakan untuk mengatur fungsi *login* dimana *login* dapat dilakukan dengan 3 level akses yang berbeda yakni admin, dosen, dan mahasiswa. Akses juga dapat dilakukan dengan *Google auth* sehingga login dapat dilakukan dengan akun *Google*,

4.3.5.1 redirect

Segmen Program 4.10 LoginController > redirect

```
    public function redirect()
    {
        return Socialite::driver('google')-
>redirect();
    }
```

4.3.5.2 callback

Segmen Program 4.11 LoginController > callback

```
 public function callback()
    {
    //    dd();

        // Google user object dari google
        $userFromGoogle =
Socialite::driver('google')->user();

        $userFromGoogle->getAvatar();

        // view()->composer('*',function($view) {
```

```
        //          $view->with('image',
$GLOBALS['image']);

        //     });
            // View::share('image',
$userFromGoogle->getAvatar());
            // $userFromGoogle->getEmail(),
           $email= strtok($userFromGoogle-
>getEmail(),'@');
           $domain= strtok('');
            $nama=$userFromGoogle->getName();



        // Ambil user dari database berdasarkan
google user id
        $userFromDatabase = User::where('nip',
strtok($userFromGoogle->getEmail(),'@'))->first();
        // dd($userFromDatabase);

        // Jika tidak ada user, maka buat user baru
        if (!$userFromDatabase) {
            // $newUser = new User([
            //     'nip' => $email,
            //     'name' => $userFromGoogle-
>getName(),
            //     'email' => $userFromGoogle-
>getEmail(),
            // ]);

            if($domain == 'john.petra.ac.id')
            {

                return
view('features.register_mahasiswa',[
```

```
                    'nip'=>$email,
                    'nama' => $nama,
                    'role'=>1,
                    'email' => $userFromGoogle-
>getEmail(),
                    ]);

            }
            elseif($domain == 'peter.petra.ac.id')
            {

                $dosen = new User();
                $dosen->nip=$email;
                $dosen->nama=$nama;
                $dosen->role=2;
                $dosen->email=$userFromGoogle-
>getEmail();
                $dosen-
>password=Hash::make("dosen$email");
                $dosen->save();



            }
            else{
                return redirect('/');
            }
            // $view= new View();
            // $view->with('image',$userFromGoogle-
>getAvatar());
            // view()->composer('*',function($view)
{
            //     $view->with('user', $image);

            // });
```

```
            $user=new User();
            $login_user=$user->all()-
>where('nip','=',$email)->first();

              if(Auth::attempt($login_user)){
               session()->regenerate();

                return redirect()-
>intended('/dashboard');
               }
               return redirect('/');

        }
        // dd('masuk');

        // Jika ada user langsung login saja
        // if(Auth::attempt($credentials)){
        //     $request->session()->regenerate();

        //     return redirect()-
>intended('/dashboard');
        //    }
            $user=new User();
            $login_user=$user->all()-
>where('nip','=',$email)->first();
            session()->regenerate();
            Auth::login($login_user);
            return redirect()-
>intended('/dashboard');

            // // dd($login_user);
            // $credentials = [
            //     'email' => $login_user['email'],
            //     'password' =>
$login_user['password'],
```

```
            // ];
            // // dd($credentials);
            //   if(Auth::attempt($credentials)){

            //     // dd($credentials);
            //    session()->regenerate();

            //     return redirect()-
>intended('/dashboard');
            //    }
            //    return redirect('/');
    }
```

4.3.5.3 store

Segmen Program 4.12 LoginController > store

```
   public function store(Request $request){
        return $request->all();
    }
```

4.3.5.4 authenticate

Segmen Program 4.13 LoginController > authenticate

```
  public function authenticate(Request $request):
RedirectResponse{


        $credentials = $request->validate([
            'email' => ['required', 'email'],
            'password' => ['required'],
        ]);


        // $user=new User();
        // $user->email=$request->email;
```

```
        // $user->password=$request->password;

        if(Auth::attempt($credentials)){
        $request->session()->regenerate();

        return redirect()->intended('/dashboard');
       }


        return back()->with('loginError','Login
Failed !');
    }
```

4.3.5.5 logout

Segmen Program 4.14 LoginController > logout

```
    public function logout(Request $request){
        Auth::logout();
        $request->session()->invalidate();
        $request->session()->regenerateToken();
        return redirect('/login');
    }
```

4.3.6 *Register Controller*

Bagian ini digunakan untuk mengontrol bagian pendaftaran bilamana terdapat dosen maupun mahasiswa yang belum terdaftar, maka melalui fitur ini dapat didaftarkan.

4.3.6.1 index

Segmen Program 4.15 RegisterController > index

```
    public function index(){
        return
view("features.admin.register_mahasiswa",[
            'title'=>'Register'
```

```
        ]);
    }
```

4.3.6.2 store

Segmen Program 4.16 RegisterController > store

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

        // dd($request->all());

        $validatedData=$request->validate([
            'nip'=>'required','unique:users',
            'nama' => 'required',
            'role'=>'required',
            'email' =>
'required|email:dns','unique:users',
            'password'=>'required',



        ]);



        $validatedData['password'] =
Hash::make($validatedData['password']);

        if ($validatedData['role']== 1){
            $mahasiswa =$request->only(['user_nip',
'konsentrasi','ipk']);
            Mahasiswa::create($mahasiswa);

        }

        //
$mahasiswa['ipk']=number_format($mahasiswa['ipk'],2
);
```

```
        User::create($validatedData);
        // $request->flash('success','Registration
Success');

        return redirect('/register')-
>with('success','Registration Success');

    }
```

4.3.7 Skripsi *Controller*

Fitur skripsi *controller* ini digunakan untuk mengontrol *database* yang ada pada skripsi untuk di *get*, *set* ataupun di *store*.

4.3.7.1 store

Segmen Program 4.17 SkripsiController > store

```
   public function store(Request $request)
    {
        $request->validate([
            'judul_skripsi'=>'required',
            'id_riwayat_mahasiswa' => 'required',
        ]);
        // dd("helo");

        //update Riwayat_Mahasiswa

        $rm=RiwayatMahasiswa::find($request-
>id_riwayat_mahasiswa);
        $rm->acc_judul=TRUE;
        $rm->save();



        //store skripsi
```

```
        $skripsi=new Skripsi();
        // dd($skripsi->all());
        $skripsi->judul=$request->judul_skripsi;
        $skripsi->id_riwayat_mahasiswa=$request-
>id_riwayat_mahasiswa;
        $skripsi->lulus_proposal= false;
        $skripsi->lulus_skripsi= false;
        $skripsi->save();
        return redirect()->back();
    }
```

4.3.7.2 edit

Segmen Program 4.18 SkripsiController > edit

```
  public function edit(Skripsi $skripsi, Request
$request)
    {


        $skripsi->find($request->id_skripsi)-
>update(['judul'=> $request->judul_skripsi]);
        // dd($request->id_riwayat_mahasiswa);

        return redirect()->back()-
>with("success_change_$request-
>id_riwayat_mahasiswa" ,'Berhasil di Ubah');
    }
```

4.3.8 Topik *Controller*

Fitur ini akan mengendalikan proses kontrol terhadap topik-topik yang diunggah oleh dosen. Topik-topik tersebut akan dipilih oleh mahasiswa dan akan dikendalikan oleh koordinator.

4.3.8.1 index

Segmen Program 4.19 TopikController > index

```
    public function index(){
        return view('features.dosen.post_topik',[
            'title' => 'Topik'
        ]);
    }
```

4.3.8.2 store

Segmen Program 4.20 TopikController > store

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



        $request->validate([

            'inputJudulTopik' => 'required',
            'inputDeskripsiTopik'=>'required',
            'inputKuota' => 'required',
            'inputPersyaratanMahasiswa'
=>'required',
            'inputFasilitasYangDidapatkan' =>
'required',



        ]);
        $topik= new Topik();
        $topik->judul=$request-
>input('inputJudulTopik');
        $topik->deskripsi=$request-
>input('inputDeskripsiTopik');
        $topik->persyaratan_mahasiswa=$request-
>input('inputPersyaratanMahasiswa');
        $topik->fasilitas_diperoleh=$request-
>input('inputFasilitasYangDidapatkan');
```

```
        $topik->kuota=$request-
>input('inputKuota');
        $topik->user_nip=$request-
>input('inputUserNip');

        $topik->save();
        // if ($validatedData['role']== 2){
        //     $mahasiswa =$request-
>only(['user_nip', 'konsentrasi','ipk']);
        // }

        //
$mahasiswa['ipk']=number_format($mahasiswa['ipk'],2
);

        // $request->flash('success','Registration
Success');

        return redirect('/post_topik')-
>with('success','Registration Success');
    }
```

4.3.9.Jadwal Sidang Akhir

Bagian ini ditujukan untuk mengontrol sistem jadwal sidang yang digunakan oleh admin, dosen dan mahasiswa. Di dalam sistem Administrasi (yang dikelola oleh Koordinator), semua jadwal sidang akan terlihat, sedangkan di dalam sistem Dosen, jadwal sidang hanya akan ditampilkan jika Dosen tersebut berperan sebagai Pembimbing atau Penguji. Sementara itu, di dalam sistem Mahasiswa, jadwal sidang akan muncul sesuai dengan Judul Tugas Akhir (TA) yang telah diajukan kepada Dosen.

4.3.9.1. index

Segmen Program 4.21 JadwalSidangAkhir > index

```
  public function index(string $id)
```

```
    {

$skripsi=Skripsi::with('riwayat_mahasiswa')-
>find($id);
        $dosen=User::all()->where('role','=','2');

$dosen_pembimbing_tambahan=DosenPembimbingTambahan:
:all()->where('id_skripsi','=',$skripsi->id);

        $merge_berhalangan=[];
        //
$merge_berhalangan=$dosen_pembimbing_tambahan-
>toBase()->

        $tmp=$skripsi->riwayat_mahasiswa->topik-
>user->jadwal_sidang_berhalangan;
        $merge_berhalangan = $tmp->toBase()-
>merge($merge_berhalangan);

        if($dosen_pembimbing_tambahan)
        foreach($dosen_pembimbing_tambahan as
$item){

            // dd($item->user-
>jadwal_sidang_berhalangan);
        $merge_berhalangan = $item->user-
>jadwal_sidang_berhalangan->toBase()-
>merge($merge_berhalangan);

        }
        // dd($merge_berhalangan);

        return
view('features.admin.post_penjadwalan_sidang_akhir'
,[
            "title"=>"Penjadwalan Sidang Akhir",
            "skripsi"=>$skripsi,
            "dosen"=>$dosen,
            "dosen_pembimbing_tambahan" =>
$dosen_pembimbing_tambahan,
            "merge_berhalangan" =>
$merge_berhalangan
        ]);
    }
```

4.3.9.2 store

Segmen Program 4.22 JadwalSidangAkhir > store

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

```
        // dd($request->all());
        $request->validate([
            "judul_ta" => 'required',
            "id_skripsi"=>'required',
            "nama_mahasiswa" => 'required',
            "nip_mahasiswa" => 'required',
            "ipk" => "required",
            "dosen_penguji.*"=>"required",
            "hari_pelaksanaan" => 'required',
            "jam_pelaksanaan" => 'required',
            "jam_selesai" => 'required',
        ]);


        // dd($request->all());
        $jadwal_sidang= new JadwalSidang();

        $jadwal_sidang->id_skripsi=$request-
>id_skripsi;
        $jadwal_sidang->hari_pelaksanaan=$request-
>hari_pelaksanaan;
        $jadwal_sidang->jam_pelaksanaan=$request-
>jam_pelaksanaan;
        $jadwal_sidang->jam_selesai=$request-
>jam_selesai;
        $jadwal_sidang->save();


        $penguji=$request->dosen_penguji;
        //peguji di jadwalkan
        foreach($penguji as $pj){

            $dosen_sidang= new DosenSidang();
```

```
            $dosen_sidang-
>id_jadwal_sidang=$jadwal_sidang->id;
            $dosen_sidang->id_dosen=strtok($pj,' -
');
            $dosen_sidang->save();

        }

        //sesi penambahan jam berhalangan


        $jbs= explode(":",$request->jam_selesai);
        $jbm= explode(":",$request-
>jam_pelaksanaan);


        //Penguji Berhalangan di tambahkan
         foreach($penguji as $pj){


$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
            while (date("H:i",$timem) <=
date("H:i",$times)){

                $jsb=new JadwalSidangBerhalangan();
                $jsb->hari=$request-
>hari_pelaksanaan;
                $jsb->jam=date("H:i",$timem);
                $jsb->user_nip=strtok($pj,' -');
                $jsb->save();
                $timem=strtotime('+30
minutes',$timem);
```

```
            }
         }
         //Pembimbing tambahan berhalangan di
tambahkan
         foreach($request-
>dosen_pembimbing_tambahan as $pj){


$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
            while (date("H:i",$timem) <=
date("H:i",$times)){

               $jsb=new JadwalSidangBerhalangan();
               $jsb->hari=$request-
>hari_pelaksanaan;
               $jsb->jam=date("H:i",$timem);
               $jsb->user_nip=$pj;
         $jsb->save();
               $timem=strtotime('+30
minutes',$timem);

            }
         }
            //Pembimnbing berhalangan di tambahkan




$times=strtotime("$jbs[0]".":"."$jbs[1]");

$timem=strtotime("$jbm[0]".":"."$jbm[1]");
               while (date("H:i",$timem) <=
date("H:i",$times)){
```

```
                    $jsb=new
JadwalSidangBerhalangan();
                    $jsb->hari=$request-
>hari_pelaksanaan;
                    $jsb->jam=date("H:i",$timem);
                    $jsb->user_nip=$request-
>dosen_pembimbing_id;
                    $jsb->save();
                    $timem=strtotime('+30
minutes',$timem);

                }

         //

        return redirect('/get_skripsi_mahasiswa')-
>with("success","Topik $request->judul_ta Berhasil
Dijadwalkan");
        // dd($request->all());

    }
```