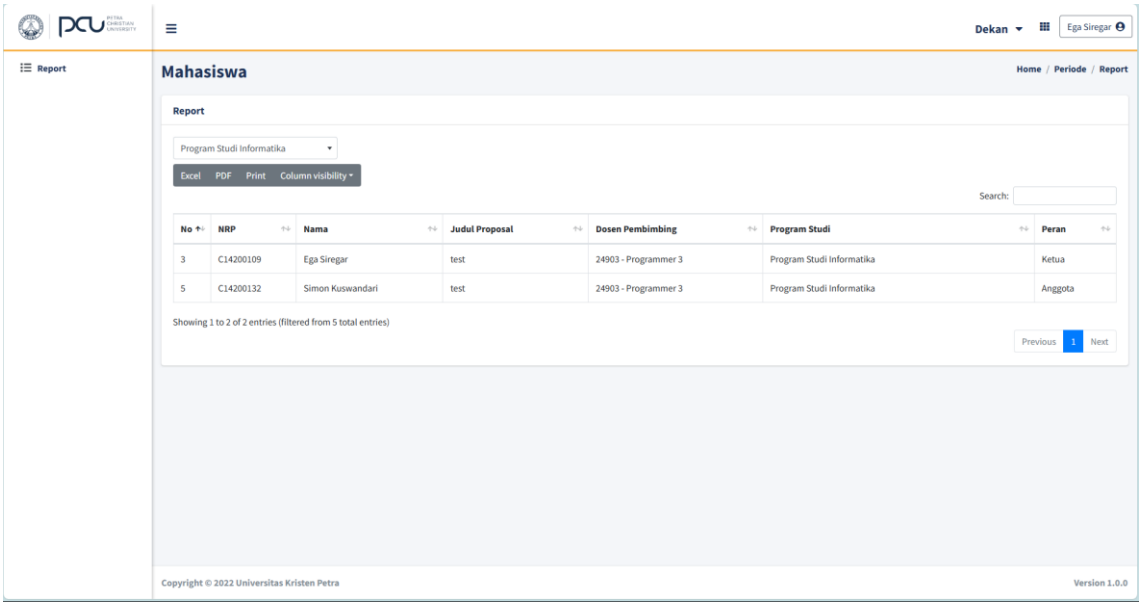
3.5.24. Tampilan Halaman Report Pimpinan Program Studi



Gambar 3.33 Tampilan Halaman Report Pimpinan Program Studi

4. IMPLEMENTASI SISTEM

4.1. Implementasi Awal

Aplikasi yang dibuat yaitu merupakan sistem manajemen proposal PKM untuk Biro Administrasi Kemahasiswaan dan Alumni UK Petra dengan menggunakan bahasa pemrograman *PHP framework laravel*, dan *database* menggunakan *postgresql*. Terdapat beberapa tools yang digunakan antara lain Laragon dan Visual Studio Code.

4.2. Konfigurasi Database

Agar aplikasi dapat mengakses database diperlukan file .env yang mengandung variabel yang diperlukan untuk koneksi ke database.

Segmen Program 4.1 Konfigurasi Database

```
DB_CONNECTION=pgsql
DB_HOST=192.168.181.200
DB_PORT=5432
DB_DATABASE=neosim
DB_USERNAME=pkm
DB_PASSWORD=
DB_SCHEMA=pkm
```

4.3. Konfigurasi Cloud Storage

Akan terjadi banyak upload file di aplikasi maka dibutuhkan cloud storage agar server tidak terbebani. Variabel yang diperlukan untuk menghubungkan aplikasi dengan cloud storage juga terletak pada file .env.

Segmen Program 4.2 Konfigurasi Cloud Storage

```
AWS_ACCESS_KEY_ID=pkm-dev
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=pkm-dev
AWS_URL=https://m1.petra.ac.id
AWS_ENDPOINT=https://m1.petra.ac.id
AWS_USE_PATH_STYLE_ENDPOINT=true
```

4.4. Pengaturan Reviewer

Dalam aplikasi ini, BAKA harus melakukan pengaturan reviewer untuk menentukan siapa yang mendapatkan role Ketua Reviewer, Anggota Reviewer, Ghostwriter agar proses review dapat berjalan.

4.4.1. Menambah Reviewer

BAKA dapat menambah reviewer dengan cara memilih tenaga didik dari menu dropdown, memilih bidang dari reviewer, dan menentukan jabatan (Ketua, Anggota, Ghostwriter).

Segmen Program 4.3 Menambah Reviewer

```
    public function store(ReviewerCreateRequest $request){
        $reviewer = new Reviewer();
        $selectedOption = explode(',', $request->nama);
        $idreviewer = decrypt($selectedOption[0]);
        $reviewer->id = $idreviewer;
        $reviewer->nip = $selectedOption[2];
        $reviewer->nama = $selectedOption[1];
        $reviewer->email = $selectedOption[3];
        $reviewer->status = $request->jabatan;
        $reviewer->save();
        $bidangvalue = $request->bidang;
        if(count($bidangvalue) !== count(array_unique($bi-
dangvalue))){
            return back()->with('error','Terdapat bidang yang sa-
ma');
        }
        else{
        foreach ($bidangvalue as $bidang) {
            $bidangreviewer = new BidangReviewer();
            $bidangid= decrypt($bidang);
            $bidangreviewer->reviewer_id = $idreviewer;
            $bidangreviewer->bidang_id = $bidangid;
            $bidangreviewer->save();
        }
        return redirect()->back()->with('success', 'Reviewer telah
ditambahkan.');
    }
    }
```

4.4.2. Edit Reviewer

BAKA dapat mengatur ulang bidang yang jadi tanggung jawab reviewer di menu edit.

Segmen Program 4.4 Edit Reviewer

```
public function update(Request $request, $id){
        $model=Reviewer::find(decrypt($id));
        $bidangvalue = $request->bidang;
        if(count($bidangvalue) !==
count(array_unique($bidangvalue))){
            return back()->with('error','Terdapat bidang yang
sama');
        }
        foreach ($bidangvalue as $b){
            $b = explode(',', $b);
            if($b[0] != 'null'){
                $bidangreviewer = BidangReviewer::where('id',
decrypt($b[0]))->first();
            }else{
                $bidangreviewer = null;
            }
            if($bidangreviewer == null){
                $bidangreviewer = new BidangReviewer();
                $bidangreviewer->reviewer_id = decrypt($id);
                $bidangreviewer->bidang_id = decrypt($b[1]);
                $bidangreviewer->save();
            }
            elseif($bidangreviewer != null){
                $bidangreviewer->bidang_id = decrypt($b[1]);
                $bidangreviewer->save();
            }

        }
        return redirect()->back()->with('success', 'Bidang Reviewer
telah diubah.');
    }
```

4.4.3. Assign Proposal Reviewer

BAKA atau Ketua Reviewer dapat menentukan proposal untuk reviewer. Setelah ditentukan reviewer akan mendapatkan pemberitahuan melalui email bahwa ada proposal yang diberikan untuk direview.

Segmen Program 4.5 Assign Proposal Reviewer

```
public function assignReviewer($id, Request $request){
        $id = decrypt($id);
        $proposal = $request->proposal;
        if(count($proposal) !== count(array_unique($proposal))){
            return back()->with('error','Terdapat proposal yang
sama');
        }

        foreach($proposal as $p){
            $p = decrypt($p);
            $proposal = Proposal::find($p);
            if($proposal->status == 'Pemilihan Reviewer'){
                $proposal->reviewer_id = $id;
                $proposal->status = 'Review BAKA';
                $proposal->save();
            }
            elseif($proposal->status == 'Pemilihan Ghostwriter'){
                $proposal->ghostwriter_id = $id;
                $proposal->status = 'Review Ghostwriter';
                $proposal->save();
            }
        }
        $revieweremail = Reviewer::find($id)
                        ->select('email', 'nama', 'status')
                        ->first();
        if($revieweremail->status == 'Ketua' || $revieweremail-
>status == 'Anggota'){
        $this->sendEmail(
            $revieweremail->email,
            'Reviewer untuk proposal',
            "Halo $revieweremail->nama, anda telah ditetapkan
sebagai reviewer untuk proposal $proposal->judul. Silahkan cek
proposal tersebut di website PKM PETRA.");
        }
        elseif($revieweremail->status == 'Ghostwriter'){
        $this->sendEmail(
            $revieweremail->email,
            'Reviewer untuk proposal',
            "Halo $revieweremail->nama, anda telah ditetapkan
sebagai ghostwriter untuk proposal $proposal->judul. Silahkan cek
proposal tersebut di website PKM PETRA.");
        }
        return back()->with('success','Reviewer berhasil
ditetapkan');

    }
```

4.5. Pengaturan Bidang PKM

Dalam aplikasi ini, BAKA dapat melakukan pengaturan bidang PKM yaitu menambahkan, mengatur rubrik penilaian, dan mengganti nama bidang.

4.5.1. Menambah Bidang

BAKA dapat menambah bidang dengan menginput nama bidang.

Segmen Program 4.6 Menambah Bidang

```
public function store(Request $request){
        $model = new Bidang();
        $model->bidang = e($request->bidang);
        $model->save();
        return redirect('bidang');
    }
```

4.5.2. Edit Bidang

BAKA dapat melakukan penggantian nama bidang di menu edit bidang.

Segmen Program 4.7 Edit Bidang

```
public function update(Request $request, $id){
        $model = Bidang::find(decrypt($id));
        $model->bidang = e($request->bidang);
        $model->save();

        return back()->with('success', 'Data berhasil diubah');
    }
```

4.5.3. Pengaturan Rubrik Penilaian Bidang

BAKA dapat mengatur Rubrik Penilaian pada Bidang dengan kriteria dan bobotnya masing-masing.

Segmen Program 4.8 Pengaturan Rubrik Penilaian Bidang

```
public function setRubrik($id, Request $request){
        $id = decrypt($id);
        $kriterias = $request->rubrik;
        $bobots = $request->bobot;
        $sumBobot = array_sum($bobots);
        if($sumBobot != 100){
            return back()->with('error', 'Total bobot harus 100');
        }
        else{
        if(is_array($kriterias) && is_array($bobots)) {
            foreach($kriterias as $key => $kriteria){
                $model = new Rubrik();
                $model->id_bidang = $id;
                $model->kriteria = $kriteria;
                $model->bobot = $bobots[$key];
                $model->save();
            }
        }
        return back()->with('success', 'Rubrik berhasil
ditambahkan');
    }
    }
```

4.5.4. Review Proposal

Dalam aplikasi ini, terdapat alur review proposal yang dimulai dari pembukaan periode, upload proposal, review dari dosen pembimbing, reviewer, dan ghostwriter.

4.5.5. List Proposal

BAKA, dosen pembimbing, reviewer, ghostwriter, dan mahasiswa dapat memilih periode PKM yang kemudian diarahkan ke daftar proposal untuk periode tersebut. Tampilan list proposal menyesuaikan role akun yang digunakan untuk mengakses.

Segmen Program 4.9 List Proposal Dosen

```
        if ($role == "DOSEN"){
            $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
                ->join('kelompok', 'proposal.id_kelompok', '=',
'kelompok.id_kelompok')
                ->join('pegawai.pegawai', 'proposal.pembimbing_id',
'=', 'pegawai.id')
                ->join('pegawai.biodata', 'pegawai.biodata_id', '=',
'biodata.id')
                ->join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                ->join('ref.unit', 'mahasiswa.program_studi_id',
'=', 'unit.id')
                ->selectRaw("
                STRING_AGG(DISTINCT concat(mahasiswa.nrp, ' - ',
kelompok.nama, ' - ', kelompok.status), ' <br> ') as kelompok,
                concat(pegawai.nip, ' - ', biodata.gelar_depan, ' ',
biodata.nama, ' ', biodata.gelar_belakang) as dosen_pembimbing,
                proposal.created_at,
                proposal.updated_at,
                proposal.id,
                proposal.judul,
                proposal.pembimbing_id,
                proposal.status,
                proposal.nilai,
                bidang_pkm.bidang as bidang,
                (SELECT STRING_AGG(unit.nama, ' <br> ' ORDER BY
mahasiswa.nrp) FROM akademik.mahasiswa JOIN ref.unit ON
mahasiswa.program_studi_id = unit.id JOIN kelompok ON mahasiswa.id =
kelompok.id_mahasiswa WHERE kelompok.id_kelompok =
proposal.id_kelompok) as prodi")
                ->where('pegawai.nip', $checkUser->nrpnip)
                ->whereIn('proposal.status', ['Menunggu Persetujuan
Dosen Pembimbing','Review Dosen Pembimbing', 'Revisi Dosen
Pembimbing', 'Review BAKA', 'Revisi BAKA', 'Review Ghostwriter',
'Revisi Ghostwriter', 'Lulus BAKA', 'Tidak Lulus BAKA', 'Lulus
Belmawa', 'Tidak Lulus Belmawa'])
                ->where('proposal.periode_id', $periodeid)
                ->groupBy('proposal.id', 'kelompok.id_kelompok',
'pegawai.nip', 'biodata.gelar_depan', 'biodata.nama',
'biodata.gelar_belakang', 'proposal.created_at',
'proposal.updated_at', 'proposal.judul', 'proposal.pembimbing_id',
'proposal.status', 'bidang_pkm.bidang')
                ->get();

                return view('proposal.index',['model'=>$model,
'periodeid'=>encrypt($periodeid)]);
        }
```

Segmen Program 4.10 List Proposal Mahasiswa

```
elseif ($role == "MAHASISWA"){
            $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
            ->join('kelompok', 'proposal.id_kelompok', '=',
'kelompok.id_kelompok')
            ->join('pegawai.pegawai', 'proposal.pembimbing_id', '=',
'pegawai.id')
            ->join('pegawai.biodata', 'pegawai.biodata_id', '=',
'biodata.id')
            ->join('akademik.mahasiswa', 'kelompok.id_mahasiswa',
'=', 'mahasiswa.id')
            ->selectRaw("
            concat(pegawai.nip, ' - ', biodata.gelar_depan, ' ',
biodata.nama, ' ', biodata.gelar_belakang) as dosen_pembimbing,
            proposal.created_at,
            proposal.updated_at,
            proposal.id,
            proposal.judul,
            proposal.pembimbing_id,
            proposal.status,
            bidang_pkm.bidang as bidang,
            kelompok.status as peran")
            ->where('mahasiswa.nrp', $checkUser->nrpnip)
            ->where('proposal.periode_id', $periodeid)
            ->get();

            return view('proposal.index',['model'=>$model,
'periodeid'=>encrypt($periodeid)]);
        }
```

Segmen Program 4.11 List Proposal Ketua Reviewer

```
if($reviewerstatus == 'Ketua'){
                $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
                            ->join('kelompok',
'proposal.id_kelompok', '=', 'kelompok.id_kelompok')
                            ->join('pegawai.pegawai',
'proposal.pembimbing_id', '=', 'pegawai.id')
                            ->join('pegawai.biodata',
'pegawai.biodata_id', '=', 'biodata.id')
                            ->join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('ref.unit',
'mahasiswa.program_studi_id', '=', 'unit.id')
                            ->leftJoin('reviewer',
'proposal.reviewer_id', '=', 'reviewer.id')
                            ->selectRaw("
                            STRING_AGG(DISTINCT
concat(mahasiswa.nrp, ' - ', kelompok.nama, ' - ', kelompok.status),
' <br> ') as kelompok,
                            concat(pegawai.nip, ' - ',
biodata.gelar_depan, ' ', biodata.nama, ' ', biodata.gelar_belakang)
as dosen_pembimbing,
                            proposal.created_at,
                            proposal.updated_at,
                            proposal.id,
                            proposal.judul,
                            proposal.reviewer_id,
                            proposal.status,
                            proposal.nilai,
                            bidang_pkm.bidang as bidang,
                            (SELECT STRING_AGG(unit.nama, ' <br> '
ORDER BY mahasiswa.nrp) FROM akademik.mahasiswa JOIN ref.unit ON
mahasiswa.program_studi_id = unit.id JOIN kelompok ON mahasiswa.id =
kelompok.id_mahasiswa WHERE kelompok.id_kelompok =
proposal.id_kelompok) as prodi")
                            ->where(function($q) use ($checkUser){
                                $q->where('proposal.status',
'Pemilihan Reviewer')
                                ->orWhere(function($q) use
($checkUser){
                                    $q->whereIn('proposal.status',
['Review BAKA', 'Revisi BAKA', 'Lulus BAKA', 'Tidak Lulus BAKA',
'Lulus Belmawa', 'Tidak Lulus Belmawa'])
                                    ->where('reviewer.nip',
$checkUser->nrpnip);
                                });
                            })
```

Segmen Program 4.12 List Proposal Anggota Reviewer

```
elseif($reviewerstatus == 'Anggota'){
                    $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
                            ->join('kelompok',
'proposal.id_kelompok', '=', 'kelompok.id_kelompok')
                            ->join('pegawai.pegawai',
'proposal.pembimbing_id', '=', 'pegawai.id')
                            ->join('pegawai.biodata',
'pegawai.biodata_id', '=', 'biodata.id')
                            ->join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('ref.unit',
'mahasiswa.program_studi_id', '=', 'unit.id')
                            ->leftJoin('reviewer',
'proposal.reviewer_id', '=', 'reviewer.id')
                            ->selectRaw("
                            STRING_AGG(DISTINCT
concat(mahasiswa.nrp, ' - ', kelompok.nama, ' - ', kelompok.status),
' <br> ') as kelompok,
                            concat(pegawai.nip, ' - ',
biodata.gelar_depan, ' ', biodata.nama, ' ', biodata.gelar_belakang)
as dosen_pembimbing,
                            proposal.created_at,
                            proposal.updated_at,
                            proposal.id,
                            proposal.judul,
                            proposal.reviewer_id,
                            proposal.status,
                            proposal.nilai,
                            bidang_pkm.bidang as bidang,
                            (SELECT STRING_AGG(unit.nama, ' <br> '
ORDER BY mahasiswa.nrp) FROM akademik.mahasiswa JOIN ref.unit ON
mahasiswa.program_studi_id = unit.id JOIN kelompok ON mahasiswa.id =
kelompok.id_mahasiswa WHERE kelompok.id_kelompok =
proposal.id_kelompok) as prodi")
                            ->where('reviewer.nip', $checkUser-
>nrpnip)
                            ->whereIn('proposal.status', ['Review
BAKA', 'Revisi BAKA', 'Lulus BAKA', 'Tidak Lulus BAKA', 'Lulus
Belmawa', 'Tidak Lulus Belmawa'])
                            ->where('proposal.periode_id',
$periodeid)
                            ->groupBy('proposal.id',
'kelompok.id_kelompok', 'pegawai.nip', 'biodata.gelar_depan',
'biodata.nama', 'biodata.gelar_belakang', 'proposal.created_at',
'proposal.updated_at', 'proposal.judul', 'proposal.pembimbing_id',
'proposal.status', 'bidang_pkm.bidang')
                            ->get();
```

Segmen Program 4.13 List Proposal Ghostwriter

```
elseif($reviewerstatus == 'Ghostwriter'){
                    $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
                    ->join('kelompok', 'proposal.id_kelompok', '=',
'kelompok.id_kelompok')
                    ->join('pegawai.pegawai',
'proposal.pembimbing_id', '=', 'pegawai.id')
                    ->join('pegawai.biodata', 'pegawai.biodata_id',
'=', 'biodata.id')
                    ->join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                    ->join('ref.unit', 'mahasiswa.program_studi_id',
'=', 'unit.id')
                    ->leftJoin('reviewer',
'proposal.ghostwriter_id', '=', 'reviewer.id')
                    ->selectRaw("
                    STRING_AGG(DISTINCT concat(mahasiswa.nrp, ' - ',
kelompok.nama, ' - ', kelompok.status), ' <br> ') as kelompok,
                    concat(pegawai.nip, ' - ', biodata.gelar_depan,
' ', biodata.nama, ' ', biodata.gelar_belakang) as dosen_pembimbing,
                    proposal.created_at,
                    proposal.updated_at,
                    proposal.id,
                    proposal.judul,
                    proposal.reviewer_id,
                    proposal.status,
                    proposal.nilai,
                    bidang_pkm.bidang as bidang,
                    (SELECT STRING_AGG(unit.nama, ' <br> ' ORDER BY
mahasiswa.nrp) FROM akademik.mahasiswa JOIN ref.unit ON
mahasiswa.program_studi_id = unit.id JOIN kelompok ON mahasiswa.id =
kelompok.id_mahasiswa WHERE kelompok.id_kelompok =
proposal.id_kelompok) as prodi")
                    ->where('reviewer.nip', $checkUser->nrpnip)
                    ->whereIn('proposal.status', ['Review
Ghostwriter', 'Revisi Ghostwriter', 'Lulus BAKA', 'Tidak Lulus
BAKA', 'Lulus Belmawa', 'Tidak Lulus Belmawa'])
                    ->where('proposal.periode_id', $periodeid)
                    ->groupBy('proposal.id', 'kelompok.id_kelompok',
'pegawai.nip', 'biodata.gelar_depan', 'biodata.nama',
'biodata.gelar_belakang', 'proposal.created_at',
'proposal.updated_at', 'proposal.judul', 'proposal.pembimbing_id',
'proposal.status', 'bidang_pkm.bidang')
                    ->get();
                    return view('proposal.index',['model'=>$model,
'reviewerstatus'=>$reviewerstatus,
'periodeid'=>encrypt($periodeid)]);
                }
```

Segmen Program 4.14 List Proposal PIC

```
elseif ($role == "PIC"){
            $model = Proposal::join('bidang_pkm',
'proposal.id_bidang', '=', 'bidang_pkm.id')
            ->join('kelompok', 'proposal.id_kelompok', '=',
'kelompok.id_kelompok')
            ->join('pegawai.pegawai', 'proposal.pembimbing_id', '=',
'pegawai.id')
            ->join('pegawai.biodata', 'pegawai.biodata_id', '=',
'biodata.id')
            ->join('akademik.mahasiswa', 'kelompok.id_mahasiswa',
'=', 'mahasiswa.id')
            ->join('ref.unit', 'mahasiswa.program_studi_id', '=',
'unit.id')
            ->selectRaw("
            STRING_AGG(DISTINCT concat(mahasiswa.nrp, ' - ',
kelompok.nama, ' - ', kelompok.status), ' <br> ') as kelompok,
            concat(pegawai.nip, ' - ', biodata.gelar_depan, ' ',
biodata.nama, ' ', biodata.gelar_belakang) as dosen_pembimbing,
            proposal.created_at,
            proposal.updated_at,
            proposal.id,
            proposal.judul,
            proposal.pembimbing_id,
            proposal.status,
            proposal.nilai,
            bidang_pkm.bidang as bidang,
            proposal.nomor_wa_ketua,
            proposal.jumlah_bimbingan,
            proposal.path_laporan_kemajuan,
            proposal.path_laporan_akhir,
            (SELECT STRING_AGG(unit.nama, ' <br> ' ORDER BY
mahasiswa.nrp) FROM akademik.mahasiswa JOIN ref.unit ON
mahasiswa.program_studi_id = unit.id JOIN kelompok ON mahasiswa.id =
kelompok.id_mahasiswa WHERE kelompok.id_kelompok =
proposal.id_kelompok) as prodi")
            ->where('proposal.periode_id', $periodeid)
            ->groupBy('proposal.id', 'kelompok.id_kelompok',
'pegawai.nip', 'biodata.gelar_depan', 'biodata.nama',
'biodata.gelar_belakang', 'proposal.created_at',
'proposal.updated_at', 'proposal.judul', 'proposal.pembimbing_id',
'proposal.status', 'bidang_pkm.bidang')
            ->get();

            return view('proposal.index',['model'=>$model,
'periodeid'=>encrypt($periodeid)]);
        }
```

4.5.6. Upload Proposal

Mahasiswa dapat melakukan upload proposal dengan memasukkan judul, bidang PKM, dosen pembimbing, ketua kelompok, anggota kelompok, nomor whatsapp ketua, dan file proposal. Dosen pembimbing yang dipilih akan mendapatkan pemberitahuan melalui email setelah proposal di upload.

Segmen Program 4.15 Upload Proposal

```
public function store(Request $request){
        if ($request->hasFile('proposal') && $request-
>file('proposal')->isValid()){
            $proposal = $request->file('proposal');
            $periode = $request->periode;
            $judul = $request->judul;
            $ketua = $request->ketua;
            $wa = $request->wa;
            $pembimbing = $request->pembimbing;
            $pembimbing = decrypt($pembimbing);
            $email = Pegawai::join('pegawai.biodata',
'pegawai.biodata_id', '=', 'biodata.id')
                            ->select('pegawai.email',
'biodata.nama')
                            ->where('pegawai.id', $pembimbing)
                            ->first();
            $bidang_id = $request->bidang;
            $bidang_id = decrypt($bidang_id);
            $fileName = time() . '_' . $proposal-
>getClientOriginalName();
            Storage::disk('s3')->put('uploads/proposal/' .
$fileName, file_get_contents($proposal));
            $proposal = new Proposal();
            $kelompok = new Kelompok();
            $idkelompok = Str::uuid();
            $ketuavalue = explode(',', $ketua);
            $anggotavalue = $request->anggota;
            if(count($anggotavalue) !==
count(array_unique($anggotavalue))){
                return back()->with('error','Terdapat anggota yang
sama');
            }
            $proposal->judul = $judul;
            $proposal->periode_id = decrypt($periode);
            $proposal->id_kelompok = $idkelompok;
            $kelompok->id_kelompok = $idkelompok;
```

```
            $kelompok->nama = $ketuavalue[1];
            $kelompok->status = 'Ketua';
            $kelompok->id_mahasiswa = decrypt($ketuavalue[0]);
            $proposal->pembimbing_id = $pembimbing;
            $proposal->id_bidang = $bidang_id;
            $proposal->status = 'Menunggu Persetujuan Dosen
Pembimbing';
            $proposal->path = $fileName;
            $proposal->nomor_wa_ketua = $wa;
            $proposal->save();
            $kelompok->save();
            foreach ($anggotavalue as $anggota) {
                $anggota = explode(',', $anggota);
                $kelompok = new Kelompok();
                $kelompok->id_kelompok = $idkelompok;
                $kelompok->id_mahasiswa = decrypt($anggota[0]);
                $kelompok->nama = $anggota[1];
$kelompok->status = 'Anggota';
                $kelompok->save();
            }

            $this->sendEmail(
                $email->email,
                'Proposal Baru',
                "Halo $email->nama, Proposal baru telah diajukan dan
anda telah dipilih sebagai dosen pembimbing. Silahkan cek proposal
tersebut di website PKM PETRA.");
            return redirect()->back()->with('success', 'Upload
proposal berhasil.');
        }
        else{
            return back()->with('error', 'Upload proposal gagal.');
        }
    }
```

4.5.7. Download Proposal

Pengguna aplikasi yang berkaitan dengan sebuah proposal dapat mendownload proposal tersebut. Fungsi download dilakukan pengecekan terlebih dahulu jika proposal sudah direvisi maka akan di download versi paling terakhir.

Segmen Program 4.16 Download Proposal

```
public function download($id){
    $id = decrypt($id);
    $proposal = Proposal::findOrFail($id);
    $filePath = $proposal->path;
    $latestPath = $proposal->latest_path;
    if($latestPath != null){
        $filePath = $latestPath;
        if (!Storage::disk('s3')->exists('uploads/revisi/'.$filePath)) {
            abort(404, 'File not found');
        }
        $stream = Storage::disk('s3')->readStream('uploads/re-
visi/'.$filePath);
        return response()->stream(function() use ($stream) {
            fpassthru($stream);
        }, 200, [
            'Content-Type' => 'application/pdf',
            'Content-Disposition' => 'inline; filename="' . base-
name($filePath) . '"'
        ]);
    }
    if (!Storage::disk('s3')->exists('uploads/proposal/'.$filePath)) {
        abort(404, 'File not found');
    }

    $stream = Storage::disk('s3')->readStream('uploads/proposal/'.$filePath);
    return response()->stream(function() use ($stream) {
        fpassthru($stream);
    }, 200, [
        'Content-Type' => 'application/pdf',
        'Content-Disposition' => 'inline; filename="' . basename($filePath) .
'"'
    ]);
    }
```

4.5.8. Persetujuan Dosen Pembimbing

Ketika dosen pembimbing dipilih menjadi pembimbing, dosen harus menyetujui terlebih dahulu agar dapat memberikan revisi.

Segmen Program 4.17 Persetujuan Dosen Pembimbing

```
public function accept($id){
        $id = decrypt($id);
        $model = Proposal::find($id);
        $ketuaemail = Kelompok::join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('akademik.peserta_didik',
'mahasiswa.peserta_didik_id', '=', 'peserta_didik.id')
                            ->select('mahasiswa.nrp',
'peserta_didik.nama')
                            ->where('kelompok.id_kelompok', $model-
>id_kelompok)
                            ->where('kelompok.status', 'Ketua')
                            ->first();
        $model->status = 'Review Dosen Pembimbing';
        $model->save();
        $this->sendEmail(
            $ketuaemail->nrp.'@john.petra.ac.id',
            'Proposal PKM',
            "Halo $ketuaemail->nama, Pembimbing yang anda tunjuk
setuju untuk menjadi pembimbing. Silahkan tunggu untuk revisi yang
akan diberikan oleh pembimbing.");
        return back()->with('success','Setuju menjadi dosen
pembimbing');
    }
```

```
    public function decline($id){
        $id = decrypt($id);
        $model = Proposal::find($id);
        $ketuaemail = Kelompok::join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('akademik.peserta_didik',
'mahasiswa.peserta_didik_id', '=', 'peserta_didik.id')
                            ->select('mahasiswa.nrp',
'peserta_didik.nama')
                            ->where('kelompok.id_kelompok', $model-
>id_kelompok)
                            ->where('kelompok.status', 'Ketua')
                            ->first();
        $model->status = 'Ditolak Dosen Pembimbing';
        $model->save();
        $this->sendEmail(
            $ketuaemail->nrp.'@john.petra.ac.id',
            'Proposal PKM',
            "Halo $ketuaemail->nama, Pembimbing yang anda pilih
menolak untuk menjadi pembimbing. Silahkan upload ulang dengan
pembimbing lain.");
        return back()->with('success','Tidak setuju menjadi dosen
pembimbing');
    }
```

4.5.9. Pemberian Nilai Proposal

Reviewer yang ditentukan dapat memberikan nilai kepada proposal sesuai dengan rubrik penilaian bidang PKM dari proposal tersebut.

Segmen Program 4.18 Pemberian Nilai Proposal

```
   public function submitNilai(Request $request, $id){
        $request->validate([
            'rubrik.*' => 'required|numeric|min:1|max:7',
        ]);
        $id = decrypt($id);
        $proposal = Proposal::findorFail($id);
        $periodeid = $proposal->periode_id;
        $nilaikkm = Periode::find($periodeid)->nilai_kkm;
        $ketuaemail = Kelompok::join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('akademik.peserta_didik',
'mahasiswa.peserta_didik_id', '=', 'peserta_didik.id')
                            ->select('mahasiswa.nrp',
'peserta_didik.nama')
                            ->where('kelompok.id_kelompok',
$proposal->id_kelompok)
                            ->where('kelompok.status', 'Ketua')
                            ->first();
        $bobot = Rubrik::where('id_bidang', $proposal->id_bidang)-
>pluck('bobot', 'id')->all();
        $totalScore = 0;
        foreach ($request->rubrik as $idkriteria => $score) {
            if (isset($bobot[$idkriteria])) {
                $totalScore += $score * $bobot[$idkriteria];
            }
        }
        $proposal->nilai = $totalScore;
        if ($nilaikkm >= 300) {
            $proposal->status = 'Revisi BAKA';
            $this->sendEmail(
                $ketuaemail->nrp.'@john.petra.ac.id',
                'Proposal telah dinilai',
                "Halo $ketuaemail->nama, Nilai proposal anda
memenuhi syarat untuk maju ke tahap selanjutnya. Silahkan dipantau
untuk revisi yang akan diberikan oleh reviewer.");
        } else {
            $proposal->status = 'Tidak Lulus BAKA';
            $this->sendEmail(
                $ketuaemail->nrp.'@john.petra.ac.id',
                'Proposal telah dinilai',
                "Halo $ketuaemail->nama, Nilai proposal anda tidak
memenuhi syarat untuk maju ke tahap selanjutnya. Tetap semangat dan
coba lagi di periode selanjutnya ya!");
        }
        $proposal->save();
```

4.5.10. List Revisi Proposal

Agar proposal setiap iterasi proposal tersimpan maka dibuatkan page list revisi, dimana setiap versi dari proposal masih bisa didownload dan dilihat .

Segmen Program 4.19 List Revisi Proposal

```
public function revisiList($id){
        $periodeid = request()->query('periodeid');
        $id = decrypt($id);
        $userid = Auth::user()->id;
        $checkUser = User::join('user_kode', 'user.id', '=',
'user_kode.user_id')
        ->select('user_kode.kode as nrpnip', 'user.email as email')
        ->where('user.id', $userid)
        ->first();
        $proposal = Proposal::find($id);
        $revisi = Revision::join('pegawai.pegawai',
'revision.nip_pemberi', '=', 'pegawai.nip')
                            ->join('pegawai.biodata',
'pegawai.biodata_id', '=', 'biodata.id')
                            ->selectRaw("concat(pegawai.nip, ' - ',
biodata.gelar_depan, ' ', biodata.nama, ' ', biodata.gelar_belakang)
as pemberi,
                            revision.id, revision.revisi,
revision.status, revision.created_at, revision.file_name,
revision.nip_pemberi")
                            ->where('revision.proposal_id', $id)
                            ->get();
        return view('proposal.revisilist',['revisi'=>$revisi,
'proposal'=>$proposal, 'periodeid' => $periodeid, 'nip'=>$checkUser-
>nrpnip]);
    }
```

4.5.11. Pemberian Revisi Proposal

Dosen pembimbing, reviewer, dan ghostwriter dapat memberikan revisi kepada proposal. Mahasiswa akan mendapat pemberitahuan melalui email jika ada revisi yang masuk.

Segmen Program 4.20 Pemberian Revisi Proposal

```
public function revise(Request $request){
        $id = $request->id;
        $id = decrypt($id);
        $userid = Auth::user()->id;
        $checkUser = User::join('user_kode', 'user.id', '=',
'user_kode.user_id')
        ->select('user_kode.kode as nrpnip', 'user.email as email')
        ->where('user.id', $userid)
        ->first();
$role = session('current_role');
        $revisi = $request->revisi;
        $proposal = Proposal::find($id);
        $periodeid = $proposal->periode_id;
        $ketuaemail = Kelompok::join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                            ->join('akademik.peserta_didik',
'mahasiswa.peserta_didik_id', '=', 'peserta_didik.id')
                            ->select('mahasiswa.nrp',
'peserta_didik.nama')
                            ->where('kelompok.id_kelompok',
$proposal->id_kelompok)
                            ->where('kelompok.status', 'Ketua')
                            ->first();
        if ($role == 'DOSEN'){
            $revision = new Revision();
            $revision->proposal_id = $id;
            $revision->revisi = $revisi;
            $revision->nip_pemberi = $checkUser->nrpnip;
            $revision->save();
            $proposal->status = 'Revisi Dosen Pembimbing';
            $proposal->save();
            $this->sendEmail(
                $ketuaemail->nrp.'@john.petra.ac.id',
                'Revisi untuk proposal',
                "Halo $ketuaemail->nama, Ada revisi yang diberikan
oleh dosen pembimbing. Silahkan cek proposal anda di website PKM
PETRA.");
```

```
            return redirect()->route('proposal.listproposal', ['id'
=> encrypt($periodeid)])->with('success','Revisi diberikan');
        }
        elseif ($role == 'PEGAWAI'){
            $revision = new Revision();
            $revision->proposal_id = $id;
            $revision->revisi = $revisi;
            $revision->nip_pemberi = $checkUser->nrpnip;
            $revision->save();
            $proposal->status = 'Revisi BAKA';
            $proposal->save();
            $this->sendEmail(
                $ketuaemail->nrp.'@john.petra.ac.id',
                'Revisi untuk proposal',
                "Halo $ketuaemail->nama, Ada revisi yang diberikan
oleh reviewer BAKA. Silahkan cek proposal anda di website PKM
PETRA.");
            return back()->with('success','Revisi diberikan');
        }

    }
```

4.5.12. Upload Revisi Proposal

Ketika ada revisi yang diberikan, mahasiswa dapat melakukan upload file proposal yang sudah direvisi. Setelah melakukan upload maka status revisi akan berubah menjadi uploaded dan pemberi revisi dapat mendownload dan approve revisi.

Segmen Program 4.21 Upload Revisi Proposal

```
public function uploadRevisi(Request $request){
        $id = $request->id;
        $id = decrypt($id);
        $revision = Revision::find($id);
        $proposalid = $revision->proposal_id;
        $proposal = Proposal::find($proposalid);
        $periodeid=$proposal->periode_id;
        $emailpemberi = Pegawai::join('pegawai.biodata',
'pegawai.biodata_id', '=', 'biodata.id')
                            ->select('pegawai.email as email',
'biodata.nama')
                            ->where('pegawai.nip', $revision-
>nip_pemberi)
                            ->first();
        $file = $request->file('revisi');
        $fileName = time() . '_' . $file->getClientOriginalName();
        if($file->isValid()){
            Storage::disk('s3')->put('uploads/revisi/' . $fileName,
file_get_contents($file));
        }
        else{
            dd('File is not valid');
        }
        $revision->file_name = $fileName;
        $proposal->latest_path = $fileName;
        $revision->status = 'Uploaded';
        $revision->save();
        $this->sendEmail(
            $emailpemberi->email,
            'Revisi telah diupload',
            "Halo $emailpemberi->nama, Proposal yang telah direvisi
sudah di upload oleh mahasiswa. Silahkan cek proposal terbaru di
website PKM PETRA.");
        return redirect()->route('proposal.revisilist', ['id' =>
encrypt($proposalid), 'periodeid' => encrypt($periodeid)])-
>with('success','Revisi berhasil diupload');
    }
```

4.5.13. Upload Laporan

Jika proposal mendapatkan pendanaan dari belmawa, mahasiswa harus upload bukti bimbingan, laporan kemajuan, laporan akhir dari belmawa kepada aplikasi agar dapat dilakukan pencatatan di internal UK Petra. Laporan kemudian harus di approve oleh BAKA agar proposal mendapatkan status selesai di aplikasi.

Segmen Program 4.22 Upload Laporan

```
public function uploadLaporan(Request $request){
    $id = $request->id;
    $id = decrypt($id);
    $proposal = Proposal::find($id);
    $jumlahbimbingan = $request->jumlahbimbingan;
    $laporankemajuan = $request->file('laporankemajuan');
    $laporanakhir = $request->file('laporanakhir');
    $bimbinganfile = $request->file('bimbingan');
    $bimbinganfilename = time() . '_' . $bimbinganfile-
>getClientOriginalName();
    $laporankemajuanfile = time() . '_' . $laporankemajuan-
>getClientOriginalName();
    $laporanakhirfile = time() . '_' . $laporanakhir-
>getClientOriginalName();
    Storage::disk('s3')->put('uploads/bimbingan/' .
$bimbinganfilename, file_get_contents($bimbinganfile));
    Storage::disk('s3')->put('uploads/laporan/' .
$laporankemajuanfile, file_get_contents($laporankemajuan));
    Storage::disk('s3')->put('uploads/laporan/' . $laporanakhirfile,
file_get_contents($laporanakhir));
    $proposal->jumlah_bimbingan = $jumlahbimbingan;
    $proposal->path_bimbingan = $bimbinganfilename;
    $proposal->path_laporan_kemajuan = $laporankemajuanfile;
    $proposal->path_laporan_akhir = $laporanakhirfile;
    $proposal->status = 'Menunggu Review Laporan';
    $proposal->save();
    return back()->with('success','Laporan berhasil diupload');
}
```

4.6. Report

BAKA, dan pimpinan program studi memerlukan report per periode untuk keperluan RenStra.

4.6.1. Report BAKA

BAKA memerlukan report mahasiswa yang mengikuti PKM yang dapat difilter berdasarkan program studi. Di tabel yang ada di halaman list mahasiswa terdapat kolom nama, NRP, judul proposal, dosen pembimbing, dan peran. Tabel dapat di jadikan excel, PDF, ataupun langsung diprint.

Segmen Program 4.23 Report BAKA

```
public function listmahasiswa($id){
    $periodeid = decrypt($id);
    $unit = Unit::whereIn('jenis', ['3','4'])
            ->get();
    $listmhs = Kelompok::join('proposal', 'kelompok.id_kelompok',
'=', 'proposal.id_kelompok')
            ->join('akademik.mahasiswa', 'kelompok.id_mahasiswa',
'=', 'mahasiswa.id')
            ->join('ref.unit', 'mahasiswa.program_studi_id', '=',
'unit.id')
            ->join('pegawai.pegawai', 'proposal.pembimbing_id', '=',
'pegawai.id')
            ->join('pegawai.biodata', 'pegawai.biodata_id', '=',
'biodata.id')
            ->select('proposal.judul', 'kelompok.nama as nama',
'kelompok.status as peran', 'mahasiswa.nrp', 'unit.nama as prodi')
            ->selectRaw("concat(pegawai.nip, ' - ',
biodata.gelar_depan, ' ', biodata.nama, ' ', biodata.gelar_belakang)
as dosen_pembimbing")
            ->where('proposal.periode_id', $periodeid)
            ->groupBy('kelompok.nama', 'proposal.judul',
'kelompok.status', 'mahasiswa.nrp', 'unit.nama', 'pegawai.nip',
'biodata.gelar_depan', 'biodata.nama', 'biodata.gelar_belakang')
            ->get();
    return view('report.daftarmahasiswaprodi',['listmhs'=>$listmhs,
'unit'=>$unit, 'periodeid'=>encrypt($periodeid)]);
}
```

4.6.2. Report Pimpinan Program Studi

Pimpinan program studi mendapatkan report dengan cara melakukan pengecekan pada nip akun dan dikaitkan kepada tabel program studi. Pimpinan program studi akan mendapatkan list mahasiswa yang mengikuti PKM dari program studi yang dipimpin.

Segmen Program 4.24 Report Pimpinan Program Studi

```
public function reportPimpinan($id){
        $periodeid = decrypt($id);
        $userid = Auth::user()->id;
        $checkUser = User::join('user_kode', 'user.id', '=',
'user_kode.user_id')
                    ->select('user_kode.kode as nrpnip', 'user.email
as email')
                    ->where('user.id', $userid)
                    ->first();
        $unit = Unit::join('pegawai.pegawai',
'unit.pimpinan_pegawai_id', '=', 'pegawai.id')
                    // ->where('pegawai.nip', $checkUser->nrpnip)
                    ->where('unit.id', '=', '103')
                    ->first();
        $prodi = Unit::where('info_left', '>=', $unit->info_left)
                    ->where('info_right', '<=', $unit->info_right)
                    ->whereIn('jenis', ['3','4'])
                    ->get();
        $listmhs = Kelompok::join('proposal',
'kelompok.id_kelompok', '=', 'proposal.id_kelompok')
                ->join('akademik.mahasiswa',
'kelompok.id_mahasiswa', '=', 'mahasiswa.id')
                ->join('ref.unit', 'mahasiswa.program_studi_id',
'=', 'unit.id')
                ->join('pegawai.pegawai', 'proposal.pembimbing_id',
'=', 'pegawai.id')
                ->join('pegawai.biodata', 'pegawai.biodata_id', '=',
'biodata.id')
                ->select('proposal.judul', 'kelompok.nama as nama',
'kelompok.status as peran', 'mahasiswa.nrp', 'unit.nama as prodi')
                ->selectRaw("concat(pegawai.nip, ' - ',
biodata.gelar_depan, ' ', biodata.nama, ' ', biodata.gelar_belakang)
as dosen_pembimbing")
                ->where('proposal.periode_id', $periodeid)
                ->groupBy('kelompok.nama', 'proposal.judul',
'kelompok.status', 'mahasiswa.nrp', 'unit.nama', 'pegawai.nip',
'biodata.gelar_depan', 'biodata.nama', 'biodata.gelar_belakang')
                ->get();
        return view('pimpinan.reportpimpinan',['listmhs'=>$listmhs,
'prodi'=>$prodi, 'periodeid'=>$periodeid]);
    }
```

4.7. Security

Dalam aplikasi ini, diimplementasikan fitur security untuk pencegahan *Cross Site Scripting* (XSS) dan *SQL Injection*. Karena aplikasi terhubung dengan database UK Petra dan dapat mengakses data mahasiswa dan tenaga didik UK Petra maka fitur security sangat diperlukan.

4.7.1. Pencegahan XSS

Pencegahan XSS dilakukan dengan cara mengimplementasikan *middleware Content Security Policy* (CSP)*, escaping pada output*, dan validasi serta sanitasi input. Pada segmen program 4.29 merupakan middleware untuk middleware CSP.

Segmen Program 4.25 Pencegahan XSS

```
public function handle(Request $request, Closure $next): Response
    {
        $response = $next($request);

        $csp = "default-src 'self'; "
        . "font-src 'self' data: https://fonts.gstatic.com; "
        . "script-src 'self' 'unsafe-inline' 'unsafe-eval'; "
        . "style-src 'self' 'unsafe-inline'; "
        . "img-src 'self' data:; "
        . "connect-src 'self';";

        $response->headers->set('Content-Security-Policy', $csp);

        return $response;
    }
```

4.7.2. Pencegahan SQL Injection

Pencegahan SQL Injection dilakukan dengan cara membuat *query* aplikasi dengan *Eloquent ORM*, dan *Query Builder*. Segmen program 4.30 merupakan query yang dibuat dengan Eloquent ORM.

Segmen Program 4.26 Pencegahan SQL Injection

```
        $user_query = Mahasiswa::join('peserta_didik',
'mahasiswa.peserta_didik_id', '=', 'peserta_didik.id')
        ->whereRaw("UPPER(mahasiswa.nrp) LIKE UPPER(?)",
["%{$search}%"])
        ->orWhereRaw("UPPER(peserta_didik.nama) LIKE UPPER(?)",
["%{$search}%"])
        ->selectRaw("concat(mahasiswa.nrp, ' - ',
peserta_didik.nama) as namanrp, mahasiswa.id, peserta_didik.nama as
nama")
        ->orderBy('nama')
        ->paginate($per_page);
        return response()->json([
            'results' => array_map(function ($data)
            {
                return [
                    'id' => encrypt($data->id).','.$data->nama,
                    'text' => $data->namanrp
                ];
            }, $user_query->items()),
            'pagination' => [
                'more' => $user_query->currentPage() < $user_query-
>lastPage()
            ]
        ]);
```