BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini menghasilkan sebuah aplikasi yang dapat mendigitalisasi berbagai proses administrasi dan bimbingan tugas akhir. Dengan adanya fitur notifikasi dan pengingat, mahasiswa dan dosen dapat selalu terinformasi tentang tenggat waktu dan perkembangan terbaru, sehingga mengurangi kemungkinan terjadinya keterlambatan. Selain itu, aplikasi ini menyediakan akses yang cepat dan mudah ke informasi penting, yang dapat membantu mahasiswa pada pengerjaan tugas akhir.

Pengembangan aplikasi ini melibatkan beberapa Langkah yang meliputi perancangan antarmuka pengguna yang intuitif, pengembangan *backend* yang kuat, dan implementasi fitur-fitur kunci seperti pencatatan *logbook* harian, upload dokumen, notifikasi, dan *dashboard monitoring*. *Framework* CodeIgniter 4 digunakan untuk membangun API, membuat aplikasi web dan mengelola *database*, yang memungkinkan pengelolaan data yang efisien dan terstruktur, sementara *framework* Flutter digunakan untuk membangun aplikasi *mobile* berbasis Android.

Aplikasi ini memanfaatkan server untuk penyimpanan data *logbook* dan hasil bimbingan. Hal ini memungkinkan fleksibilitas yang lebih besar bagi mahasiswa dan dosen dalam mengakses dan memperbarui data, sambil memastikan bahwa data disimpan dengan aman di lingkungan yang terkontrol.

4.2 PEMBAHASAN

Setelah melalui proses perancangan dan implementasi berdasarkan data yang diperoleh melalui pengumpulan data sebelumnya, dihasilkan sebuah aplikasi berbasis Android, sebuah aplikasi berbasis web serta sebuah aplikasi API. Aplikasi Android dirancang khusus untuk digunakan oleh dosen dan mahasiswa, memudahkan mereka dalam membuat serta mendapatkan informasi perkembangan penelitian. Aplikasi web diperuntukkan bagi admin untuk mengelola data pengguna aplikasi Android, melakukan validasi serta mengatur jadwal seminar proposal dan

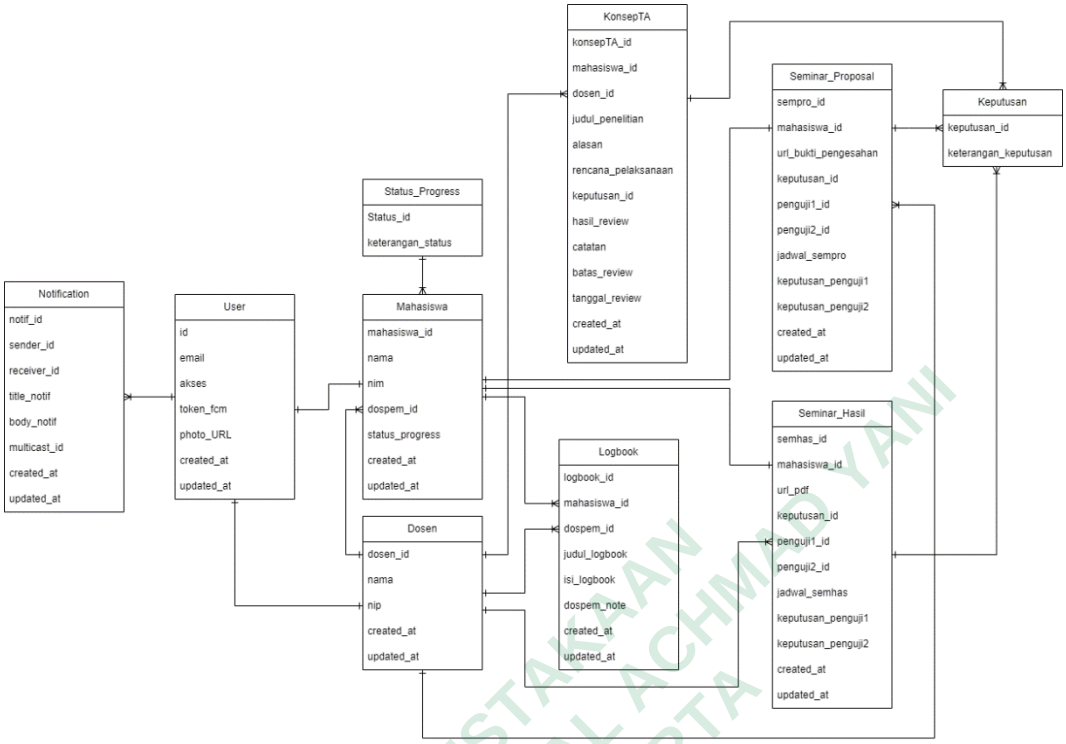
seminar hasil. Sementara itu, aplikasi API digunakan oleh aplikasi Android untuk berkomunikasi dengan *database*. Berikut pembahasan berdasarkan aplikasi:

4.2.1 BACKEND

Tiga komponen utama membentuk *backend* penelitian ini ialah server, struktur *database*, dan API. Berikut adalah penjelasan tentang masing-masing komponen

1. Struktur Database

Bagian ini mencakup desain dan manajemen database yang digunakan untuk menyimpan semua data aplikasi. Struktur *database* mencakup tabel-tabel yang diperlukan, relasi antar tabel, serta skema yang memastikan integritas dan efisiensi penyimpanan data. Penelitian ini menggunakan *database* dengan model SQL dengan *tools* XAMPP sebagai apache. Pada penelitian ini terdapat 10 tabel yang digunakan, terdiri dari *user*, mahasiswa, dosen, *logbook*, notifikasi, konsep tugas akhir, seminar proposal, seminar hasil, status *progress*, dan keputusan. Berikut merupakan gambar struktur database yang dituangkan melalui *Entity Relationship Diagram* (ERD):



Gambar 4.1 Table ERD

2. Server

Pada penelitian ini, server masih menggunakan lokal yang dikonfigurasi menggunakan Apache, bagian dari paket XAMPP, yang menyediakan lingkungan pengembangan lokal yang terdiri dari Apache sebagai server web, PHP sebagai bahasa pemrograman, dan MySQL sebagai sistem manajemen basis data. Server berfungsi sebagai pusat kendali *backend*, menangani semua permintaan dari aplikasi Android. Aplikasi backend di server menjalankan logika bisnis, memproses permintaan HTTP, dan memberikan *response* kepada klien. XAMPP yang digunakan merupakan versi V3.3.0 dan Bahasa pemrograman PHP dengan versi 7.4.22.

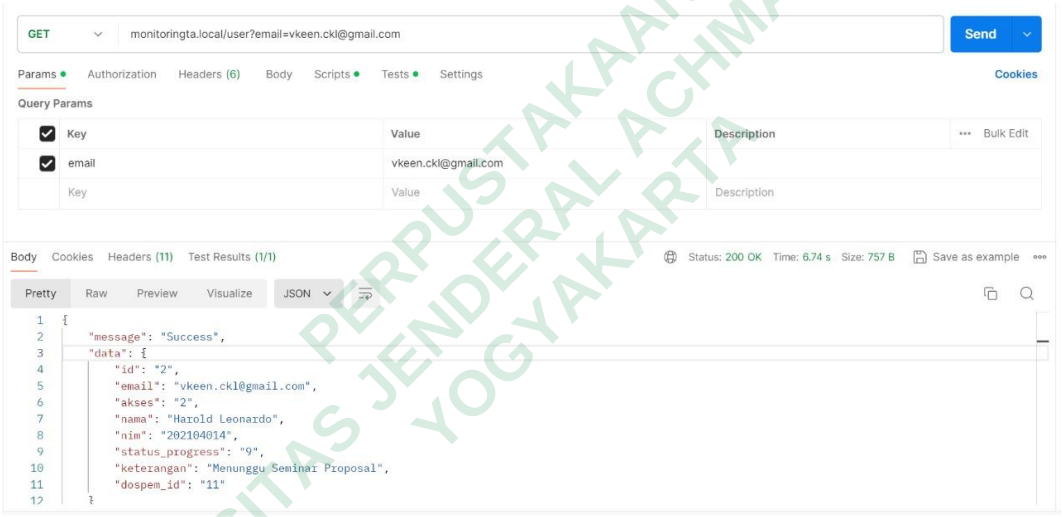
3. API

Aplikasi API digunakan untuk komunikasi antara *database* dan aplikasi Android. API ini bertindak sebagai jembatan yang memungkinkan aplikasi Android untuk mengakses dan memanipulasi data yang disimpan dalam *database*. Penelitian ini menggunakan format JSON sebagai format pertukaran data dalam komunikasi

antar aplikasi. Aplikasi API dibangun menggukana *framework* CodeIgniter 4.4.1. Fungsionalitas utama dari aplikasi API meliputi:

- Autentikasi

API menangani proses login dan validasi pengguna untuk memastikan bahwa hanya pengguna yang memenuhi persyaratan yang diizinkan untuk mengakses data dan aplikasi Android. Pada proses ini, API akan memberikan data pengguna yang sesuai dengan email yang diberikan pada saat proses login. Proses autentikasi juga menggunakan API *google sign in* sebagai metode loginnya dengan menggunakan firebase autentikasi. Berikut contoh pertukaran data antara aplikasi API dan aplikasi Android pada saat *login*:



Gambar 4.2 Data dari API saat login

- Operasi CRUD

Aplikasi API menyediakan endpoint untuk berbagai operasi CRUD pada data penelitian, dan logbook. Contohnya, mahasiswa dapat mengirim data logbook, melakukan pengajuan seminar, serta melakukan pengajuan konsep tugas akhir, sementara dosen dapat memberikan catatan kepada mahasiswa pada setiap *logbook*-nya dan memberikan keputusan terhadap hasil seminar mahasiswa. Berikut contoh kode operasi create untuk membuat logbook yang dapat dilihat pada kode program 4.1.

Kode Program 4.1 Kode Operasi Membuat Logbook

```
public function create()
    {
        $data = $this->request->getJSON();

        $insertData = [
            'id_mhs' => $data->id,
            'judul' => $data->judul,
            'isi' => $data->isi,
            'logbook_dospem_id' => $data->dosen_id,
        ];

        if($this->model->insert($insertData)){
            $sendNotif = new NotificationController;
            $mahasiswaModel = new Mahasiswa();
            $mahasiswa = $mahasiswaModel->find($data->id);

            $response = [
                'message' => 'Data has been Inputed'
            ];

            $sendNotif->send_notif('Cek perkembangan penelitian '
. $mahasiswa['nama'] . '. Serta berikan saran yang mendukung'
,$mahasiswa['nama'] .' Baru Saja Membuat Logbook', $data->id,
$data->dosen_id);

        } else {
            $response = [
                'message' => 'Data Failed Inputed'
            ];
        }
        return $this->respondCreated($response);
    }
```

Kode program 4.2 merupakan kode operasi *read* untuk mendapatkan data konsep tugas akhir yang telah diajukan.

Kode Program 4.2 Kode Operasi Membaca Konsep Tugas Akhir

```
public function showById()
    {
        $validationRules = [
            'id' => 'required',
        ];

        if (!$this->validate($validationRules)) {
            return $this->respond($this->validator->getErrors(),
400);
```

```
        }

        $id = $this->request->getVar('id');

        $data = $this->model->where('konsepTA_id_mhs', $id)
        ->select('konsepTA.*, mahasiswa.nama AS nama_mahasiswa,
dosen.nama AS nama_dosen, keputusan.keterangan_keputusan AS
status')
        ->join('mahasiswa', 'konsepTA_id_mhs =
mahasiswa.mahasiswa_id')
        ->join('keputusan', 'keputusan_kTA =
keputusan.keputusan_id')
        ->join('dosen', 'konsepTA_id_dosen = dosen.dosen_id',
'left')
        ->orderBy('keputusan_kTA', 'DESC')
        ->findAll();

        $response = [
            'message' => 'Success',
            'data' => $data,
        ];

        return $this->respond($response, 200);
    }
```

Kode program 4.3 merupakan kode operasi *update* untuk memperbaharui data yaitu catatan *logbook.*

Kode Program 4.3 Kode Operasi Membuat Catatan Logbook

```
public function update($id = null)
    {
        $data = $this->request->getJSON(true);
        if (isset($data['logbook_id']) &&
isset($data['dospem_note'])) {
            $logbook = $this->model->find($data['logbook_id']);
            $dosenModel = new Dosen();
            $dosen = $dosenModel-
>find($logbook['logbook_dospem_id']);
            $sendNotif = new NotificationController;
            $updateData = [
                'dospem_note' => $data['dospem_note'],
            ];
            if ($this->model->update($data['logbook_id'],
$updateData)) {
                $sendNotif->send_notif('Cek saran yang diberikan
oleh ' .$dosen['nama'] .' Semoga saran yang diberikan dapat
membantu', $dosen['nama'] . 'baru saja memberikan saran pada
logbook kamu', $dosen['dosen_id'], $logbook['id_mhs'] );
```

```
                return $this->respondUpdated(['message' =>
'success', 'data' => $updateData]);
            } else {
                return $this->response->setStatusCode(500)-
>setJSON(['error' => 'Failed to update logbook.']);
            }
        } else {
            // Handle the absence of 'email'
            return $this->response->setStatusCode(400)-
>setJSON(['error' => 'key Value is missing in the input data.']);
        }
    }
```

4.2.2 APLIKASI ANDROID

Pada penelitian ini, aplikasi Android dibangun menggunakan Android Stuido dengan versi 2023.2.1 (Iguana). Aplikasi yang dikembangkan digunakan oleh dua *user* yaitu mahasiswa dan dosen yang digunakan untuk melakukan kegiatan penelitian. Fungsionalitas utama aplikasi ini adalah sebagai berikut:

1. Login

Kedua user dari aplikasi ini dapat melakukan *login* menggunakan *sign in with google* dengan email yang telah didaftarkan oleh admin. Gambar 4.3 merupakan contoh tampilan *login* dari aplikasi Android.

Gambar 4.3 Tampilan *Login* Aplikasi Android

Untuk dapat melakukan login, aplikasi menjalankan *function* atau bagian kode program 4.4.

Kode Program 4.4 Melakukan *Login* pada Aplikasi

```
signInWithGoogle() async {
  // await takeToken();
  setState(() {
    _isLoading = true;
  });
  tokenFCM = (await FirebaseApi().getTokenFcm())!;
  GoogleSignInAccount? googleUser = await
GoogleSignIn().signIn();
  setState(() {
    _isLoading = false;
  });
  if(googleUser?.email != null) {
    setState(() {
      _isLoading = true;
    });
    email = googleUser!.email;
    await ApiFunction().login(email, tokenFCM);
    UserData? user = await getUserData();
    print('user : $user');
    if(user != null){
      GoogleSignInAuthentication? googleAuth = await
```

```
googleUser.authentication;
      AuthCredential credential = GoogleAuthProvider.credential(
          accessToken: googleAuth.accessToken,
          idToken: googleAuth.idToken
      );
      UserCredential userCredential = await
FirebaseAuth.instance.signInWithCredential(credential);
      photoUrlActive = userCredential.user?.photoURL;
      await savePhoto(userCredential.user?.photoURL);
      print('User Credential : ${userCredential.user}');
      if(userCredential.user != null){
        Navigator.of(context).push(MaterialPageRoute(builder:
(context) => HomeHandler(index: 0,)));
        setState(() {
          _isLoading = false;
        });
      } else {
        await GoogleSignIn().signOut();
        FirebaseAuth.instance.signOut();
        clearUserData();
        setState(() {
          _isLoading = false;
        });
      }
    } else {
      setState(() {
        _isLoading = false;
      });
    }
  };
}

Future<void> login(String email, String tokenFCM) async {

  late Map<String, dynamic> putRequestBody = {
    'email' : email,
    'tokenFcm' : tokenFCM,
  };
  await getUserData(email);
  await Api.putData('${Api.baseUrl}updateFcmToken',
      putRequestBody,
          (data) {
        print('Update FCM Token : $data');
      },
          (error) { }
  );
}
```

Selanjutnya setelah mengeksekusi kode diatas, maka kode program API *login* yang akan dijalankan adalah kode program 4.5.

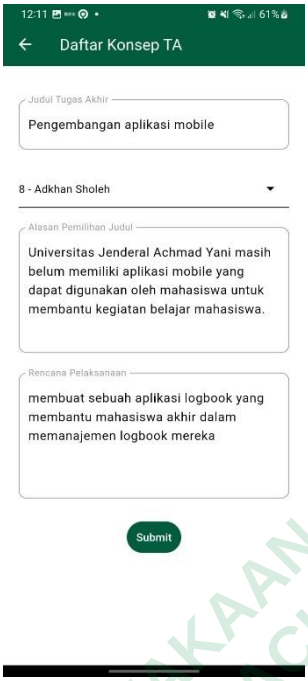
Kode Program 4.5 Kode Program API *Login* untuk Aplikasi

```
//Routes pada app/config/routes.php
$routes->get('/user', 'UserController::showByEmail');

//Controller pada app/controllers/UserController.php
public function showByEmail()
    {
        $Dosen = new Dosen();
        $Mahasiswa = new Mahasiswa();

        $validationRules = [
            'email' => 'required|valid_email',
        ];

        log_message('debug', 'Setelah rules Validation');

        if (!$this->validate($validationRules)) {
            return $this->respond($this->validator->getErrors(),
400);
        }

        log_message('debug', 'Setelah Validation');

        $email = esc($this->request->getVar('email'));

        log_message('debug', $email );


        log_message('debug', 'Memulai email terdefenisi ');

        $getUser = $this->model->like('email', $email)->first();
        log_message('debug', 'User ditemukan ');
        if(!$getUser){
        log_message('debug', 'User tidak ada');
            return $this->failNotFound('User not found');
        }
        $akses = (int)$getUser['akses'];
        if($akses === 1)
        {
            $user = $Dosen->like('dosen_id', $getUser['id'])
            ->select('user.id, user.email, user.akses,
dosen.nama, dosen.nip')
            ->join('user', 'id = dosen.dosen_id')->first();
            // $user = $Dosen->like('dosen_id', (int)$akses)-
>first();
```

```
        }
        elseif($akses === 2)
        {
            $user = $Mahasiswa->like('mahasiswa_id',
$getUser['id'])
            ->select('user.id, user.email, user.akses,
mahasiswa.nama, mahasiswa.nim, mahasiswa.status_progress,
status_progress.keterangan, mahasiswa.dospem_id')
            ->join('user', 'id = mahasiswa.mahasiswa_id')-
>join('status_progress', 'status_id =
mahasiswa.status_progress')->first();
            // $user = $Mahasiswa->like('dosen_id', (int)$akses)-
>first();
        }
        else
        {
            return $this->failNotFound('You are not allowed to
login in this App');
        }

        if (!$user) {
            return $this->failNotFound('User not found');
        }

        $data = [
            'message' => 'Success',
            'data' => $user
        ];
        log_message('debug', 'Berhasil mendapatkan data user: ' .
json_encode($data));
        return $this->respond($data, 200);
    }
```

2. Pengajuan Konsep Tugas Akhir dan Seminar

Pengajuan konsep tugas akhir dapat dilakukan dengan menuangkan ide yang ada dalam kata-kata. Mahasiswa dapat mengisi *form* yang ada yaitu judul penelitian, memilih dosen pembimbing, memberikan alasan pemilihan judul, serta rencana pelaksanaan. Gambar 4.4 merupakan tampilan membuat konsep tugas akhir pada aplikasi.



Gambar 4.4 Tampilan Pengajuan Konsep Tugas Akhir

Untuk mengirim konsep tugas akhir yang telah diisi, aplikasi menjalankan kode program 4.6.

Kode Program 4.6 Membuat Konsep Tugas Akhir

```
myButton("Submit", () async {
  if (_Judul.text.isNotEmpty && _Alasan.text.isNotEmpty &&
_Rencana.text.isNotEmpty) {
    setState(() {
      _isLoading =true;
    });
    // All fields are valid
    await ApiFunction().newKonsepTA(body);
    Navigator.pop(context);
  } else {
    // Show an error message if any field is invalid
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Please fill out the required
fields')),
    );
  }
})

Future<void> newKonsepTA(Map<String,dynamic> body) async {
  await Api.postData(
      '${Api.baseUrl}konsepTA',
```

```
      body,
          (error) {
        print(error);
          });
}
```

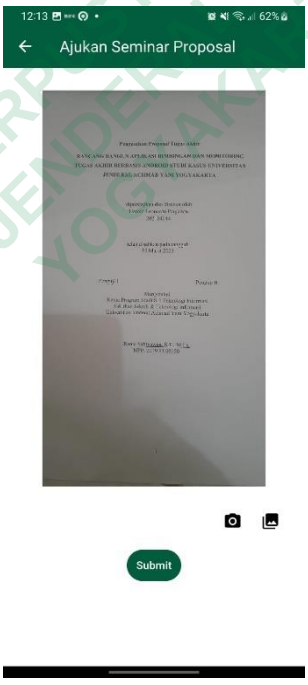
Setalah mengeksekusi kode diatas, kode API yang dijalankan setelahnya adalah kode program 4.7.

Kode Program 4.7 Kode Program API Membuat Konsep Tugas Akhir

```
//Routes pada app/config/routes.php
$routes->post('/konsepTA', 'KonsepTAController::create');

//Controller pada app/controllers/KonsepTAController.php
public function create()
    {
        $validationRules = [
            'id_mhs' => 'required|integer',
            'judul' => 'required',
            'alasan' => 'required',
            'rencana_pelaksanaan' => 'required',
        ];

        if(!$this->validate($validationRules)){
            return $this->failValidationErrors($this->validator-
>getErrors());
        }

        $mahasiswaModel = new Mahasiswa();

        $data = $this->request->getJSON();

        $currentDateTime = new \DateTime();
        $currentDateTime->add(new \DateInterval('P1W'));
        $newDateTime = $currentDateTime->format('Y-m-d H:i:s');

        $insertData = [
            'konsepTA_id_mhs' => $data->id_mhs,
            'judul_penelitian' => $data->judul,
            'alasan' => $data->alasan,
            'rencana_pelaksanaan' => $data->rencana_pelaksanaan,
            'konsepTA_id_dosen' => $data->id_dosen,
            'batas_review' => $newDateTime,
        ];

        if($this->model->insert($insertData)){
```

```
            $mahasiswaModel->update($data->id_mhs,
['status_progress' => 2]);
            $response = [
                'message' => 'Data has been Inputed'
            ];
        } else {
            $response = [
                'message' => 'Data Failed Inputed'
            ];
        }
        return $this->respondCreated($response);
    }
```

Untuk dapat mengajukan seminar proposal maupun seminar hasil, mahasiswa diwajibkan memenuhi syarat tertentu yaitu bukti pengesahan pada saat seminar proposal dan *file* tugas akhir pada saat seminar hasil. Pada aplikasi yang dibangun, mahasiswa dapat melakukan foto langsung atau memilih dari *gallery* untuk mengunggah bukti pengesahannya. Hal ini dapat dilihat pada gambar 4.5.



Gambar 4.5 Mengunggah Bukti Pengesahan

Kode program 4.8 merupakan bagian kode yang digunakan oleh aplikasi Android untuk mengunggah bukti pengesahan.

Kode Program 4.8 Mengunggah Bukti Pengesahan

```
myButton('Submit', () async {
  setState(() {
  });
  if (_image != null){
    setState(() {
      _isLoading = true;
    });
    await ApiFunction().pengajuanSeminarProposal(_image!);
  Navigator.pop(context);
  } else {
    setState(() {
      _isLoading = false;
    });
    final snackBar = SnackBar(
        backgroundColor : primaryColor,
        content: Text('Silahkan Upload Bukti Terlebih Dahulu',
style: TextStyle(color: Colors.white),));
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }

}),
Future<void> pengajuanSeminarProposal(File file) async {
  Map<String, dynamic> body = {
    "id_mhs" : userActive?.id,
  };
  await Api.postDataWithFile(
      '${Api.baseUrl}upload-file-sempro',
      body,
      file,
          (error) {
        print(error);
      });
}
```


Setelah menjalankan kode program aplikasi diatas, kode program API yang dijalankan adalah kode program 4.9.

Kode Program 4.9 Kode API Mengunggah Bukti Pengesahan

```
//Routes pada app/config/routes.php
$routes->post('/upload-file-sempro', 'SeminarProposal::create');

//Controller pada app/controllers/SeminarProposal.php
public function create()
{
    helper(['form', 'url']);
```

```
    $validationRules = [
        'data' => 'required',
    ];

    if (!$this->validate($validationRules)) {
        return $this->failValidationErrors($this->validator-
>getErrors());
    }

    $mahasiswaModel = new Mahasiswa();

    $JSONdata = $this->request->getPost('data');
    $file = $this->request->getFile('file');
    $data = json_decode($JSONdata, true);
    $existingUser = $this->model->where('id_mhs_seminar',
$data['id_mhs'])->first();

    if ($file->isValid() && !$file->hasMoved()) {
        $fileName = $file->getName();
        $file->move(FCPATH . 'sempro', $fileName);

        if ($existingUser) {
            // Update the existing record
            $updateData = [
                'url_photo_bukti_seminar' => $fileName,
                'id_keputusan_sempro' => 0,
            ];
            if ($this->model->update($data['id_mhs'],
$updateData)) {
                $response = [
                    'message' => 'Data has been updated'
                ];
            } else {
                $response = [
                    'message' => 'Data update failed'
                ];
            }
        } else {
            // Insert a new record
            $insertData = [
                'id_mhs_seminar' => $data['id_mhs'],
                'url_photo_bukti_seminar' => $fileName,
            ];
            if ($this->model->insert($insertData)) {
                $response = [
                    'message' => 'Data has been inserted'
                ];
            } else {
                $response = [
```

```
                    'message' => 'Data insertion failed'
                ];
            }
        }
        $mahasiswa = $mahasiswaModel->find($data['id_mhs']);
        $sendNotif = new NotificationController;
        $sendNotif->send_notif('Nantikan Jadwal Seminar Proposal
'. $mahasiswa['nama'], $mahasiswa['nama'].' Telah Mendaftarkan
Seminar Proposal', $mahasiswa['mahasiswa_id'],
$mahasiswa['dospem_id'] );

        $mahasiswaModel->update($data['id_mhs'],
['status_progress' => 7]);
    } else {
        $response = [
            'message' => 'No valid file found or file already
moved'
        ];
    }

    return $this->respondCreated($response);
}
```

Untuk mengunggah file tugas akhir, mahasiswa dapat memilih *file* tugas akhirnya lalu mengirimnya seperti gambar berikut

Gambar 4.6 Mengunggah File Penelitian

Bagian kode program yang digunakan oleh aplikasi Android untuk mengunggah *file* penelitian seperti kode program 4.10.

Kode Program 4.10 Mengunggah *File* Penelitian

```
myButton('Submit', () async {
  if (_file != null) {
    setState(() {
      _isLoading = true;
    });
    await ApiFunction().pengajuanSeminarHasil(_file!);
    Navigator.pop(context);
  } else {
    setState(() {
      _isLoading = false;
    });
    final snackBar = SnackBar(
      backgroundColor: primaryColor,
      content: Text(
        'Silahkan Upload File Penelitian Terlebih Dahulu',
        style: TextStyle(color: Colors.white),
      ),
    );
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }
}),

Future<void> pengajuanSeminarHasil(File file) async {
  Map<String, dynamic> body = {
    "id_mhs" : userActive?.id,
  };
  await Api.postDataWithFile(
      '${Api.baseUrl}upload-file-semhas',
      body,
      file,
          (error) {
        print(error);
      });
}
```


Setelah menjalankan kode diatas, kode API yang dijalankan setelahnya adalah kode program 4.11.

Kode Program 4.11 Kode Program API Mengunggah *File* Penelitian

```
//Routes pada app/config/routes.php
$routes->post('/upload-file-semhas', 'SeminarHasil::new');

//Controller pada app/Controller/SeminarHasilController.php
public function new()
    {
        helper(['form', 'url']);

        $validationRules = [
            'data' => 'required',
        ];

        if (!$this->validate($validationRules)) {
            return $this->failValidationErrors($this->validator-
>getErrors());
        }

        $mahasiswaModel = new Mahasiswa();

        $JSONdata = $this->request->getPost('data');
        $file = $this->request->getFile('file');
        $data = json_decode($JSONdata, true);
        $existingUser = $this->model-
>where('id_mhs_seminar_hasil', $data['id_mhs'])->first();

        log_message('debug', 'Berhasil mendapatkan data user: ' .
$file->getName());

        if ($file->isValid() && !$file->hasMoved()) {
            $fileName = $file->getName();
            $filePath = FCPATH . 'semhas/' . $fileName;

            if (file_exists($filePath)) {
                // Define the new name for the existing file
                $newExistingFileName = 'old_' . $fileName;
                $newExistingFilePath = FCPATH . 'semhas/' .
$newExistingFileName;

                // Rename the existing file
                if (!rename($filePath, $newExistingFilePath)) {
                    // Handle error if renaming fails
                    echo "Failed to rename the existing file.";
                    return;
                }
            }
            $file->move(FCPATH . 'semhas', $fileName);
```

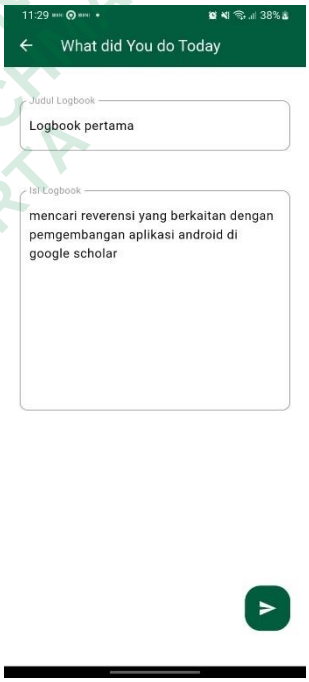
```
            if ($existingUser != null) {
                // Update the existing record
                $updateData = [
                    'url_pdf_seminar_hasil' => $fileName,
                    'id_keputusan_semhas' => 0,
                ];
                if ($this->model-
>update($existingUser['id_seminar_hasil'], $updateData)) {
                    $response = [
                        'message' => 'Data has been updated'
                    ];
                } else {
                    $response = [
                        'message' => 'Data update failed'
                    ];
                }
            } else {

                $semproModel = New SeminarProposal();
                $dataSempro = $semproModel-
>where('id_mhs_seminar', $data['id_mhs'])->first();
                $insertData = [
                    'id_mhs_seminar_hasil' => $data['id_mhs'],
                    'url_pdf_seminar_hasil' => $fileName,
                    'id_penguji_semhas_1' =>
$dataSempro['id_penguji_1'],
                    'id_penguji_semhas_2' =>
$dataSempro['id_penguji_2'],
                ];
                if ($this->model->insert($insertData)) {
                    $response = [
                        'message' => 'Data has been inserted'
                    ];
                } else {
                    $response = [
                        'message' => 'Data insertion failed'
                    ];
                }
            }
            $mahasiswa = $mahasiswaModel->find($data['id_mhs']);
            $sendNotif = new NotificationController;
            $sendNotif->send_notif('Nantikan Jadwal Seminar Hasil
'. $mahasiswa['nama'], $mahasiswa['nama'].' Telah Mendaftarkan
Seminar Hasil', $mahasiswa['mahasiswa_id'],
$mahasiswa['dospem_id'] );

            $mahasiswaModel->update($data['id_mhs'],
['status_progress' => 13]);
        } else {
```

```
            $response = [
                'message' => 'No valid file found'
            ];
        }

        return $this->respondCreated($response);
    }
```

3. Membuat logbook harian dan catatan logbook

Logbook merupakan komponen penting dalam pengerjaan tugas akhir. Pada aplikasi ini mahasiswa dapat dengan mudah membuat *logbook* hariannya, hanya dengan menekan tombol " + " (tambah) di bawah kanan seperti gambar 4.4 lalu mengisi bagian judul dan bagian isi seperti gambar 4.5

Gambar 4.7 Tampilan *Logbook*



Gambar 4.8 Pembuatan *logbook*

Kode program yang digunakan oleh aplikasi Android untuk membuat *logbook* adalah kode program 4.12.

Kode Program 4.12 Membuat *Logbook*

```
IconButton(
  onPressed: () async {
    setState(() {
      _isLoading = true;
    });
```

```
    await ApiFunction().newLogbook(_Judul.text, _Isi.text);
    Navigator.pushReplacement(context, MaterialPageRoute(builder:
(context) => HomeHandler(index: 2)));
  },
  icon: Icon(Icons.send, color: Colors.white,),

),

Future<void> newLogbook(String judul, String isi) async {
  Map<String, dynamic> query = {
    "id" : userActive?.id,
    "judul" : judul,
    "isi" : isi,
    "dosen_id" : userActive?.idDosen,
  };
  print('data to be send :$query');
  await Api.postData(
      '${Api.baseUrl}/new-logbook',
      query,
          (error) {
        print(error);
          }
  );
}
```

Setelah menjalankan kode program pada aplikasi diatas, kode program API yang akan dijalankan adalah kode program 4.13.

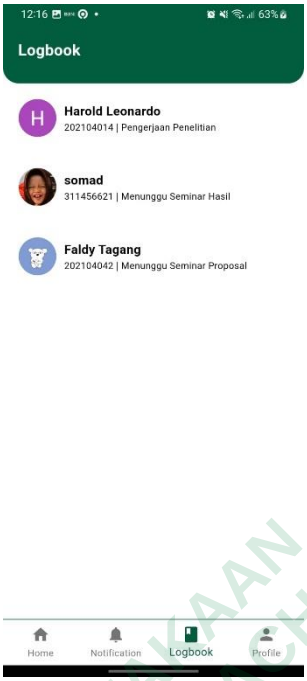
Kode Program 4.13 Kode Program API Membuat *Logbook*

```
//Routes pada app/config/routes.php
$routes->post('/new-logbook', 'LogbookController::create');

//Controller pada app/controller/logbookController.php
public function create()
    {
        $data = $this->request->getJSON();

        $insertData = [
            'id_mhs' => $data->id,
            'judul' => $data->judul,
            'isi' => $data->isi,
            'logbook_dospem_id' => $data->dosen_id,
        ];

        if($this->model->insert($insertData)){
            $sendNotif = new NotificationController;
            $mahasiswaModel = new Mahasiswa();
```

```
            $mahasiswa = $mahasiswaModel->find($data->id);

            $response = [
                'message' => 'Data has been Inputed'
            ];

            $sendNotif->send_notif('Cek perkembangan penelitian '
. $mahasiswa['nama'] . '. Serta berikan saran yang mendukung'
,$mahasiswa['nama'] . ' Baru Saja Membuat Logbook', $data->id,
$data->dosen_id);

        } else {
            $response = [
                'message' => 'Data Failed Inputed'
            ];
        }
        return $this->respondCreated($response);
    }
```

Dosen pembimbing dapat melihat *logbook* dari setiap mahasiswa bimbingannya seperti gambar 4.9 dan 4.10.

Gambar 4.9 Tampilan *Logbook* Pengguna Dosen Berdasarkan Mahasiswa



Gambar 4.10 Tampilan Daftar Mahasiswa bimbingan

Dosen pembimbing dapat memberikan catatan disetiap *logbook* mahasiswa. Tampilannya dapat dilihat pada gambar 4.11.

Gambar 4.11 Tampilan Catatan *Logbook*

Kode program 4.14 merupakan kode program yang digunakan oleh aplikasi yang untuk membuat catatan *logbook*.

Kode Program 4.14 Kode Tombol Memberikan Catatan *Logbook*

```
IconButton(
  onPressed: () async {
    setState(() {
      _isLoading =true;
    });
    await
ApiFunction().updateNotesLogbook(int.parse(widget.data['logbook_i
d']), _notes.text);
    Navigator.pop(context);
  },
  icon: Icon(Icons.send, color: Colors.white,),
),

Future<void> updateNotesLogbook(int logbookId, String notes)
async {
  late Map<String, dynamic> putRequestBody = {
    'logbook_id' : logbookId,
    'dospem_note' : notes,
  };
  await Api.putData('${Api.baseUrl}update-logbook',
      putRequestBody,
          (data) {
        print('Update Logbook : $data');
      },
          (error) { }
  );
}
```

Setelah menjalankan kode program aplikasi diatas, kode program API yang akan dijalankan adalah kode program 4.15.

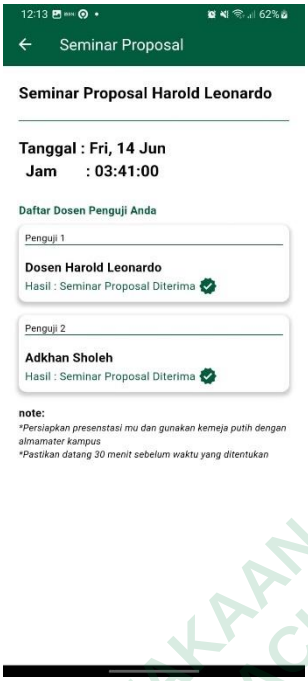
Kode Program 4.15 Kode Program API Memberikan Catatan *Logbook*

```
//Route pada app/config/routes.php
$routes->put(‘/update-logbook’, ‘LogbookController::update’);

//Controller pada app/controller/logbookController.php
public function update($id = null)
    {
        $data = $this->request->getJSON(true);
        if (isset($data[‘logbook_id’]) &&
isset($data[‘dospem_note’])) {
```

```
            $logbook = $this->model->find($data['logbook_id']);
            $dosenModel = new Dosen();
            $dosen = $dosenModel-
>find($logbook['logbook_dospem_id']);
            $sendNotif = new NotificationController;
            $updateData = [
                'dospem_note' => $data['dospem_note'],
            ];
            if ($this->model->update($data['logbook_id'],
$updateData)) {
                $sendNotif->send_notif('Cek saran yang diberikan
oleh ' .$dosen['nama'] .' Semoga saran yang diberikan dapat
membantu', $dosen['nama'] . 'baru saja memberikan saran pada
logbook kamu', $dosen['dosen_id'], $logbook['id_mhs'] );
                return $this->respondUpdated(['message' =>
'success', 'data' => $updateData]);
            } else {
                return $this->response->setStatusCode(500)-
>setJSON(['error' => 'Failed to update logbook.']);
            }
        } else {
            // Handle the absence of 'email'
            return $this->response->setStatusCode(400)-
>setJSON(['error' => 'key Value is missing in the input data.']);
        }
    }

    /**
     * Delete the designated resource object from the model
     *
     * @return mixed
     */
    public function delete($id = null)
    {
        //
    }
```

4. Melihat informasi seminar

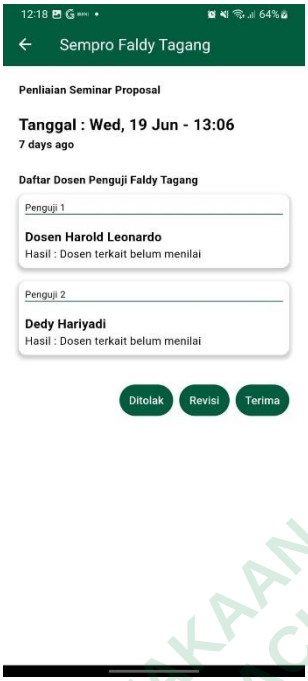
User dapat melihat informasi pada seminar proposal maupun seminar hasil yang telah ditentukan oleh admin seperti gambar 4.12.



Gambar 4.12 Tampilan Informasi Seminar

5. Menyetujui, merevisi atau menolak hasil seminar

Setelah proses seminar baik seminar proposal maupun seminar hasil, dosen penguji dapat memberikan eputusan terhadap hasil seminar yang baru dilaksanakan. Terdapat 3 (tiga) pilihan sebagai hasil eputusan yaitu menerima, merevisi, dan menolak hasil seminar. Dosen penguji dapat melakukan pemberian eputusan sesaat setelah seminar dimulai. Hal tersebut dapat dilihat pada gambar 4.13.



Gambar 4.13 Tampilan Seminar untuk Dosen Saat Seminar sedang Berjalan

Untuk memberikan eputusan hasil seminar, Dosen penguji dapat menekan salah satu tombol yang tersedia, aplikasi akan menjalankan bagian kode program 4.16, setelah itu kode program 4.17 akan dijalankan oleh API.

Kode Program 4.16 Kode Tombol Keputusan Dosen Penguji

```
Widget _buildButton(int keputusanPenguji) {
  switch (keputusanPenguji) {
    case 0:
      return Row(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          buttonRef(),
          spaceX(context, 0.02),
          buttonRev(),
          spaceX(context, 0.02),
          buttonAcc()
        ],
      );
    case 1:
      return Row(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          buttonRev(),
          spaceX(context, 0.02),
          buttonAcc()
```

```
        ],
      );
    case 2:
      return Align(
          alignment: Alignment.centerRight,
          child: buttonAcc()
      );
    default:
      return Container();
  }
}
Widget buttonRef(){
  return myButton('Ditolak', () async {
    bool authenticate = await
BiometricService.authenticate('Tolak Seminar Hasil
${widget.data['nama']}?');
    if (authenticate){
      await ApiFunction().updateSemhasKeputusanDosen('1',
semhasData['id_seminar_hasil']);
      Navigator.pop(context);
    }
  });
}
Widget buttonRev(){
  return myButton('Revisi', () async {
    bool authenticate = await
BiometricService.authenticate('Revisi Seminar Hasil
${widget.data['nama']}?');
    if (authenticate){
      await ApiFunction().updateSemhasKeputusanDosen('2',
semhasData['id_seminar_hasil']);
      Navigator.pop(context);
    }
  });
}
Widget buttonAcc(){
  return myButton('Terima', () async {
    bool authenticate = await
BiometricService.authenticate('Terima Seminar Hasil
${widget.data['nama']}?');
    if (authenticate){
      await ApiFunction().updateSemhasKeputusanDosen('3',
semhasData['id_seminar_hasil']);
      Navigator.pop(context);
    }
  });
}
```

Kode Program 4.17 Kode Program API Memberikan Keputusan

```
//Route pada app/config/routes.php
$routes->put('/seminarHasil-update-keputusan-dosen',
'SeminarHasilController::updateKeputusanDosenSemhas');

//controller pada app/controllers/seminarhasilController.php
public function updateKeputusanDosenSemhas()
    {
        $rules = [
            'keputusan' => 'required',
            'id' => 'required',
            'penguji' => 'required',
        ];

        if(!$this->validate($rules)){
            return $this->failValidationErrors($this->validator-
>getErrors());
        }

        $data = $this->request->getJSON(true);

        $semhas = $this->model->find($data['id']);
        $keputusan = (int)$data['keputusan'];

        if($semhas['id_penguji_semhas_1'] == $data['penguji']){
            if($keputusan == 1){
                $updateData = [
                    'keputusan_penguji_semhas_1' => $keputusan,
                    'id_keputusan_semhas' => 0,
                ];
            } else {
                $updateData = [
                    'keputusan_penguji_semhas_1' => $keputusan,
                ];
            }
        } else if ($semhas['id_penguji_semhas_2'] ==
$data['penguji']) {
            if($keputusan == 1){
                $updateData = [
                    'keputusan_penguji_semhas_2' => $keputusan,
                    'id_keputusan_semhas' => 0,
                ];
            } else {
                $updateData = [
                    'keputusan_penguji_semhas_2' => $keputusan,
                ];
            }
        } else{
```

```
            return $this->response->setStatusCode(500)-
>setJSON(['error' => 'Cannot Find Penguji']);
        }

        if ($this->model->update($data['id'], $updateData)) {
            $dosenModel = new Dosen();
            $dosen = $dosenModel->find($data['penguji']);
            $semhasUpdate = $this->model->find($data['id']);
            if($semhasUpdate['keputusan_penguji_semhas_1'] != 0
&& $semhasUpdate['keputusan_penguji_semhas_2'] != 0){
             if($semhasUpdate['keputusan_penguji_semhas_1'] <
$semhasUpdate['keputusan_penguji_semhas_2']){
                $keputusanUpdate =
$semhasUpdate['keputusan_penguji_semhas_1'];
             }  else
if($semhasUpdate['keputusan_penguji_semhas_1'] ==
$semhasUpdate['keputusan_penguji_semhas_2']){
                $keputusanUpdate =
$semhasUpdate['keputusan_penguji_semhas_1'];
             } else {
                $keputusanUpdate =
$semhasUpdate['keputusan_penguji_semhas_2'];
             }
            }
            if($keputusanUpdate == 1){
                $updateMahasiswa = [
                    'status_progress' => 14,
                ];
            }else if($keputusanUpdate == 2){
                $updateMahasiswa = [
                    'status_progress' => 16,
                ];
            }else if($keputusanUpdate == 3){
                $updateMahasiswa = [
                    'status_progress' => 17,
                ];
            }
            if($data['keputusan'] == 1){
                $title = 'Sayang Sekali, '.$dosen['nama'] . '
Menolak Seminar Hasil Kamu';
                $body = 'Tetap semangat dan perbaiki kembali
daripada gagal wisuda tahun ini!';
            }else if($data['keputusan'] == 2){
                $title = 'Kerja Bagus!, '.$dosen['nama'] . '
Menerima dengan Revisi';
                $body = 'Ayo perbaiki secepatnya, agar bisa
wisuda tahun ini!';
            }else if($data['keputusan'] == 3){
```

```
                $title = 'Selamat!, ' .$dosen['nama'] . '
Menerima Seminar Hasil Kamu';
                $body = 'Luar biasa!, Kamu telah mengalahkan raja
terkahir dalam dunia perkuliahan S1';
            }
            $send_notif = new NotificationController();
            $result_notif = $send_notif-
>send_notif($body,$title,$dosen['dosen_id'],$semhas['id_mhs_semin
ar_hasil']);
            $mahasiswa = new Mahasiswa();
            $mahasiswa->update($semhas['id_mhs_seminar_hasil'],
$updateMahasiswa);
            return $this->respondUpdated(['message' => 'success',
'data' => $updateData]);
        } else {
            // return $this->fail('Failed to update name.');
            return $this->response->setStatusCode(500)-
>setJSON(['error' => 'Failed to update user.']);
        }
    }
```

6. Notifikasi

Fitur notifikasi memiliki fungsi sebagai pengingat mahasiswa dalam membuat *logbook* harian, update dari dosen, serta pemberitahuan mengenai update penelitian seperti jadwal seminar dan hasil keputusan dari dosen dan admin. Sementara itu, dosen mendapatkan pemberitahuan mengenai perkembangan penelitian mahasiswanya serta jadwal seminar yang akan dilaksanakan sebagai dosen penguji. Gambar 4.14 merupakann tampilan dari notifikasi yang diterima oleh user mahasiswa.

Gambar 4.14 Tampilan Notifikasi di *smartpohne User*

Gambar 4.15 Tampilan Notifikasi di Aplikasi

Dalam menjalankan fungsi-fungsi yan telah dijelaskan, penggunaan izin (*permissions*) diperlukan untuk mengakses fitur dan data sensitif pada perangkat pengguna. Berikut adalah beberapa izin yang umum digunakan pada aplikasi Android:

- notifikasi
- storage
- camera
- biometrik
- internet

Aplikasi ini memiliki ukuran file sebesar 157.563 KB (157 MB), dan memerlukan minimal 21 SDK untuk mengakses izin seperti *biometric*.

4.2.3 APLIKASI WEB

Aplikasi web dikembangkan menggunakan CodeIgniter 4.4.1 yang digunakan oleh admin untuk mengelola data yang diluar kendali mahasiswa dan dosen seperti meninjau konsep tugas akhir mahasiswa, memberikan jadwal seminar, memasukkan data mahasiswa dan dosen serta menentukan dosen

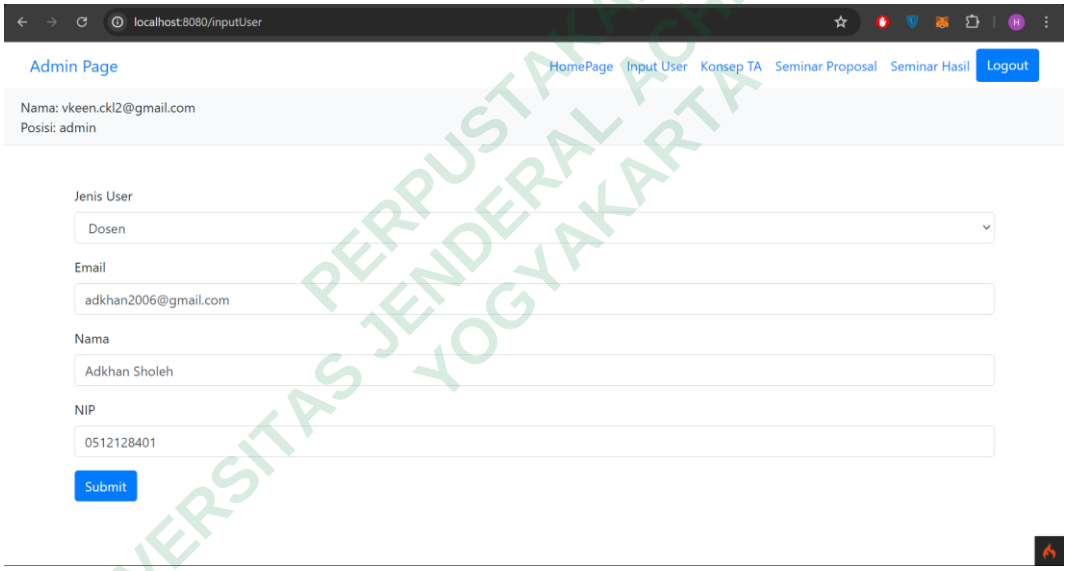
pembimbing dan dosen penguji. Fungsi yang digunakan untuk mendukung admin dalam melakukan tugasnya sebagai berikut:

1. Login

Admin dapat melakukan login menggunakan *sign in with google* dengan email yang telah terdaftar, proses ini menggunakan *Cloud Google* sebagai alat bantunya.

2. Manajemen pengguna

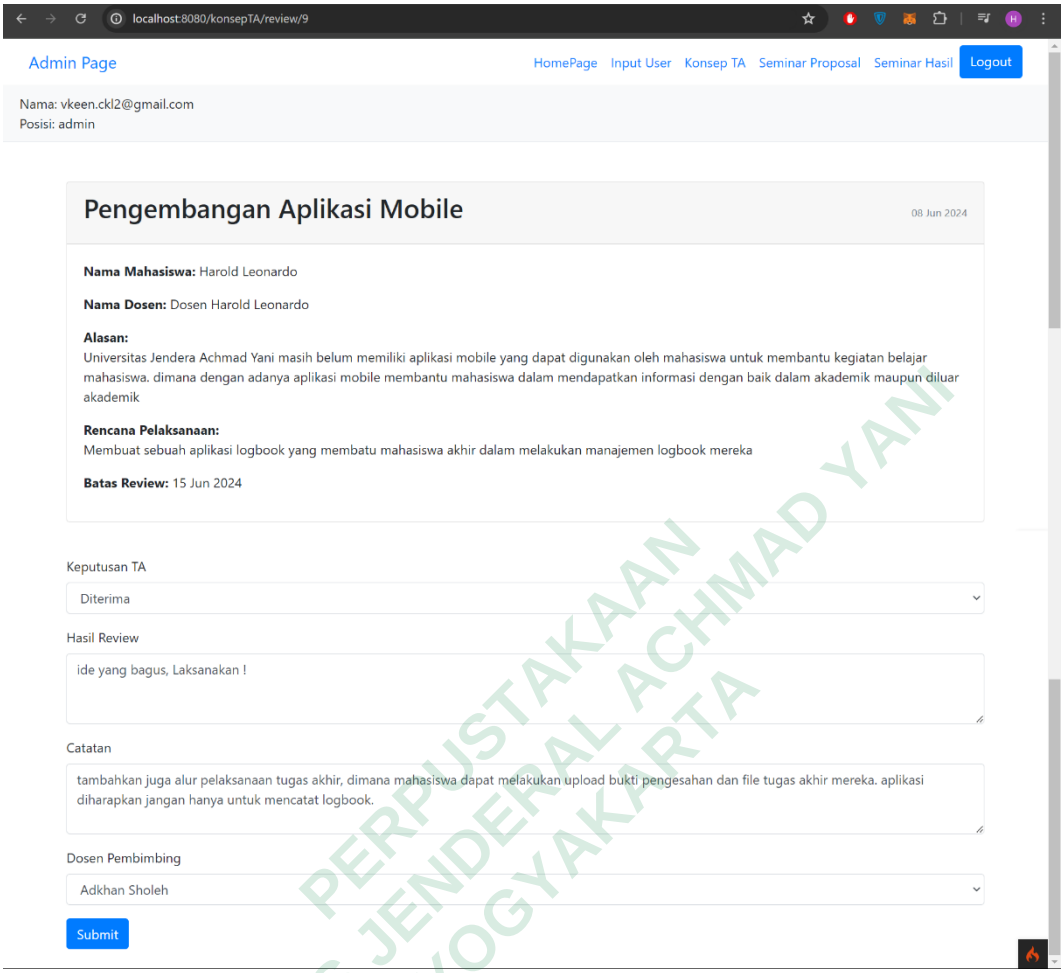
Admin dapat mendaftarkan pengguna sesuai dengan data yang telah diberikan oleh mahasiswa dan dosen. Data yang di masukkan berupa email, nama, nim (untuk mahasiswa), nip (untuk dosen) serta tipe dari user (mahasiswa atau dosen). Berikut contoh tampilannya



Gambar 4.16 Tampilan *Input User*

3. Meninjau Konsep Tugas Akhir

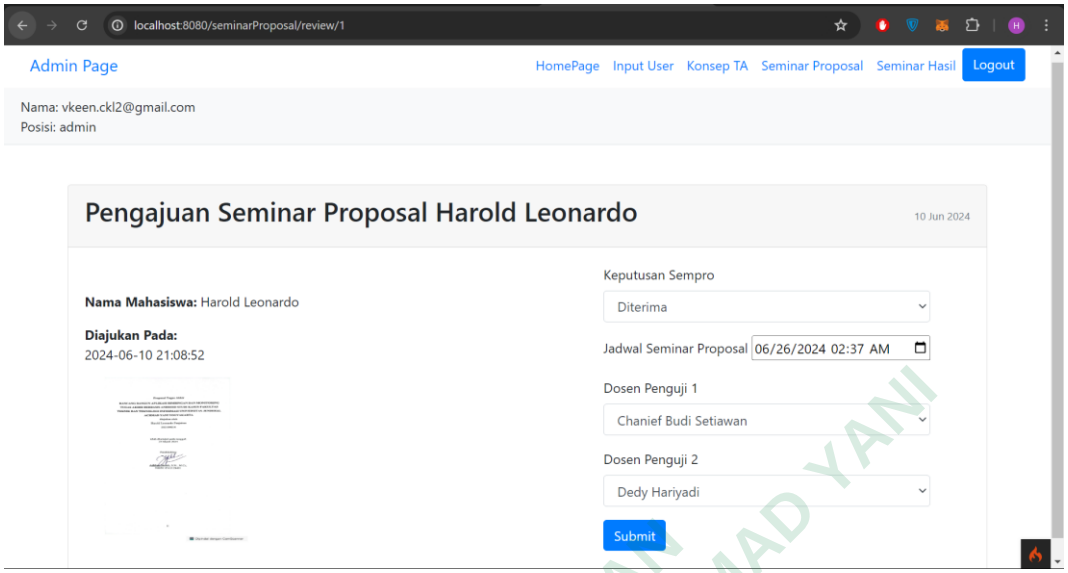
Admin dapat melakukan *review* terhadap konsep tugas akhir yang dibuat oleh mahasiswa serta menentukan dosen pembimbing terhadap mahasiswa seperti pada gambar 4.17.



Gambar 4.17 Tampilan Saat Admin Meninjau Konsep Tugas Akhir

4. Meninjau Seminar Proposal dan Seminar Hasil

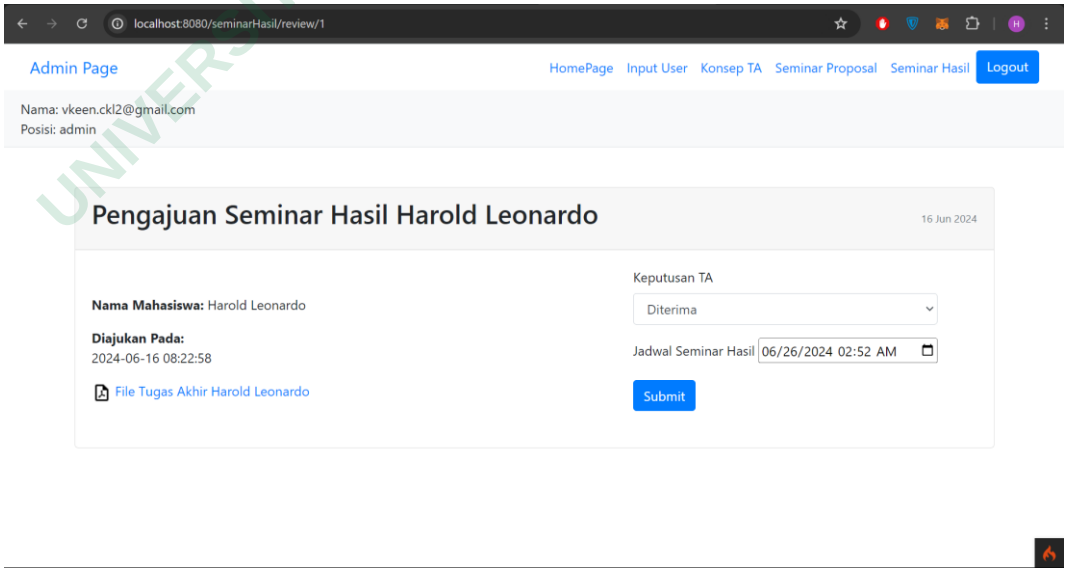
Admin dapat melihat bukti pengesahan dari dosen pembimbing untuk mengajukan seminar proposal yang telah dikirim oleh mahasiswa lalu memvalidasi bukti tersebut, setelah itu admin dapat memberikan keputusan serta menentukan dosen penguji dan jadwal seminar seperti gambar 4.18.



Gambar 4.18 Tampilan Ketika Admin Meninjau Pengajuan Seminar Proposal

Bukti pengesahan dapat dilihat secara detail dengan cara menekan foto dari bukti pengesahan. Ketika bukti pengesahan ditekan makan akan menampilkan halaman baru yang hanya akan menampilkan foto butki pengesahan tersebut.

Pada tahap seminar hasil, admin dapat meninjau *file* tugas akhir mahasiswa yang telah dikirim sebagai bentuk persyaratan pendaftaran seminar hasil. Setelah melakukan *review*, admin dapat memberikan keputusan atas pengajuan seminar hasil mahasiswa. Tampilannya dapat dilihat pada gambar 4.19



Gambar 4.19 Tampilan Admin Ketika Meninjau Seminar Hasil

4.3 HASIL PENGUJIAN

Pengujian aplikasi dilakukan dengan memberikan akses kepada beberapa mahasiswa untuk menggunakan aplikasi dan mensimulasikan pengerjaan tugas akhir seperti melakukan pengajuan konsep tugas akhir, membuat *logbook*, mengajukan seminar proposal, dan mengajukan seminar hasil. Berikut ini adalah hasil pengujian yang telah dilakukan.

4.3.1 LOGBOOK

Mahasiswa dapat mengakses dan mengisi *logbook* harian mereka tanpa ada masalah. Semua entri *logbook* tersimpan dengan benar di *database* dan dapat diakses kembali kapan saja. Dosen dapat mengakses *logbook* setiap mahasiswa dan memberikan catatan yang dapat dilihat juga oleh mahasiswanya. Tampilan *logbook* dapat dilihat pada gambar 4.9.

4.3.2 KONSEP TUGAS AKHIR

Mahasiswa dapat membuat beberapa konsep tugas akhir dan mengirimkannya untuk persetujuan admin. Seluruh data pada pengajuan konsep tugas akhir dapat diakses kembali pada menu konsep tugas akhir. Pengujian menunjukkan bahwa ketika satu konsep diterima oleh admin maka konsep tugas akhir lainnya secara otomatis berubah menjadi ditolak sesuai dengan yang diharapkan. Pada saat salah satu konsep tugas akhir diterima, maka admin harus menentukan dosen pembimbing mahasiswa tersebut. Dosen juga akan menerima notifikasi saat mendapatkan mahasiswa bimbingan baru. Tampilan konsep tugas akhir dapat dilihat pada gambar 4.4.

4.3.3 SEMINAR PROPOSAL DAN SEMINAR HASIL

Fitur ini diuji dengan beberapa mahasiswa yang mengunggah bukti pengesahan proposal mereka. Pengujian menunjukkan bahwa proses unggah *file* berjalan lancar dan *file* tersimpan dengan baik di server. Admin dapat melihat file unggahan dari mahasiswa serta melakukan validasi atau memberikan keputusan. Mahasiswa dapat melihat status proposal mereka, apakah diterima, ditolak, atau

perlu revisi, dengan notifikasi yang dikirim melalui aplikasi. Tampilan untuk mengajukan seminar proposal dan seminar hasil dapat dilihat pada gambar 4.5 dan 4.6.

4.3.4 NOTIFIKASI

Fitur notifikasi diuji dengan mengirimkan berbagai jenis notifikasi kepada mahasiswa dan dosen. Hasilnya, semua notifikasi diterima dengan baik dan ditampilkan di aplikasi secara *real-time*. Mahasiswa melaporkan bahwa notifikasi membantu dalam mendapatkan informasi terbaru tentang status tugas akhir dan seminar. Contoh tampilan notifikasi yang diterima oleh *user* terdapat pada gambar 4.14.

4.3.5 PENGGUNAAN ANTARMUKA

Selama pengujian, mahasiswa menemukan antarmuka aplikasi masih sangat sederhana, masih banyak bagian pada aplikasi yang hanya menampilkan halaman kosong. Bagian navigasi antar fitur aplikasi berjalan mulus tanpa adanya *bug* atau kesalahan.

4.3.6 KECEPATAN DAN KINERJA

Aplikasi menunjukkan performa yang cukup baik selama pengujian. Tidak ada keluhan mengenai lambatnya aplikasi atau kegagalan dalam memuat data. Tetapi aplikasi memakan penyimpanan yang cukup besar yaitu 460MB seperti pada gambar 4.20.

Gambar 4.20 Besar aplikasi e-logbook