Pada Antarmuka Cari Pekerjaan ini akan diimplementasikan beberapa *widget* Flutter sebagai berikut :

a. *Widget* ShowBottomSheet digunakan untuk memunculkan form sunting profi
b. *Widget* TextFormField digunakan untuk menghandle form sunting profil
c. *Widget* Text untuk menghandle kata ”Edit Pro*file*”
d. *Widget* OutlineButton untuk menghandle tombol ”Save”
e. *Widget* FlatButton untuk menghandle tombol ”Cancel”

BAB IV
IMPLEMENTASI DAN PENGUJIAN

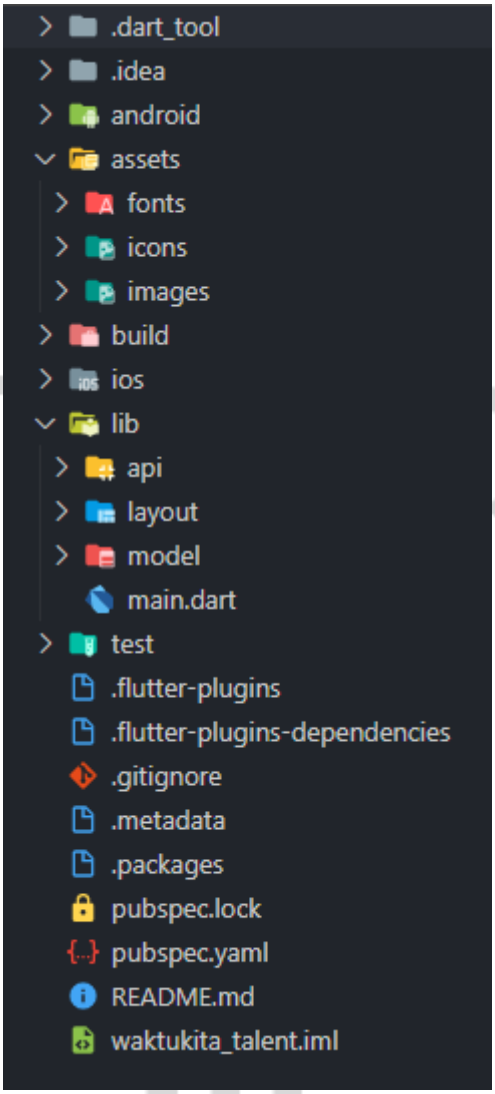
4.1 Implementasi

Pada tahap ini penulis akan menjelaskan terkait pembuatan aplikasi berdasarkan analisis kebutuhan dan perancangan. Dalam implementasi ini akan dibahas mengenai pemodelan data, pembuatan presenter dan view dengan konteks pembuatan daftar kelas saja, kode program yang lain akan penulis cantumkan dalam lampiran.

4.1.1 Penambahan *Package, Assets Images, Assets Icon* dan *Font*

Dalam perancangan yang sudah ada, penulis perlu menambahkan beberapa package seperti *http, shared_preferences, flutter_svg, url_launcher, toast* dan *image picker*. Penulis menambahkan *folder* api untuk menghandle *file* presenter, layout untuk menghandle *file* view dan model untuk menghandle *file* pemodelan data. Penulis juga menambahkan *folder assets* images untuk menampung *file* gambar, *icon*s untuk menampung *file icon* dan *font*s untuk menampung *file font* yg akan digunakan dalam aplikasi. Struktur direktori setelah penambahan beberapa *folder* oleh penulis dapat dilihat pada Gambar 4.1.

.



Gambar 4.1 Struktur Direktori setelah penambahan *folder* dari *user*

File-*file* yang sudah ditambahkan ke dalam project flutter oleh penulis bukan berarti langsung dapat dibaca oleh flutter. Package *dependencies* dan *assets* perlu ditambahkan ke dalam *file* pubspec.yaml agar dapat terbaca oleh flutter. Penambahan package *dependencies* dan *assets* dapat dilihat pada Gambar 4.2.

```yaml
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2
  shared_preferences: any
  http: any
  flutter_svg: ^0.14.1
  url_launcher: ^5.2.2
  toast: ^0.1.5
  image_picker: ^0.6.2+1

dev_dependencies:
  flutter_test:
    sdk: flutter


flutter:
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/images/wk-logo.svg
    - assets/images/wk-icon.svg
    - assets/images/bootcamp-cs.svg
    - assets/images/verif.svg
    - assets/images/companyicon.png
    - assets/icons/home.svg
    - assets/icons/jobs.svg
    - assets/icons/applied.svg
    - assets/icons/profile.svg
    - assets/icons/search.svg
    - assets/icons/logout.svg
    - assets/icons/edit.svg

  fonts:
    - family: Nunito
      fonts:
        - asset: assets/fonts/Nunito-Regular.ttf
        - asset: assets/fonts/Nunito-Bold.ttf
          weight: 700
```

Gambar 4.2 Penambahan Package *Dependencies* dan Assets dalam *file* pubspec.yaml

4.1.2 Pemodelan Data

Pemodelan data dilakukan untuk menerjemahkan data *JSON* ke dalam model yang dapat dimengerti oleh bahasa Dart.

4.1.2.1 List Endpoint

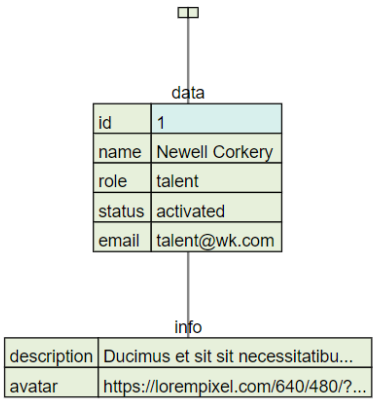
Endpoint merupakan URL yang digunakan sebagai pertukaran data dengan menggunakan metode http. Tabel 4.1 merupakan rancangan endpoint *API* yang akan digunakan dalam system.

Tabel 4.1 List Endpoint *API*

URL	Method	Params
$baseUrl/auth/register/talent	POST	
$baseUrl/auth/login	POST	
$baseUrl/auth/me	GET	
$baseUrl/talent/bootcamps	GET	- filter[name]
$baseUrl/talent/jobs	GET	- filter[q] - filter[status] - filter[position]
$baseUrl/jobs/$id	GET	
$baseUrl/jobs/$id/apply	POST	

4.1.2.2 Data *User*

Data *user* merupakan data yang akan digunakan untuk data *user* pro*file* yang berhubungan dengan anatarmuka login, antarmuka register dan antarmuka pro*file*. Gambar 4.3 merupakan diagram pohon *JSON* data *user*.

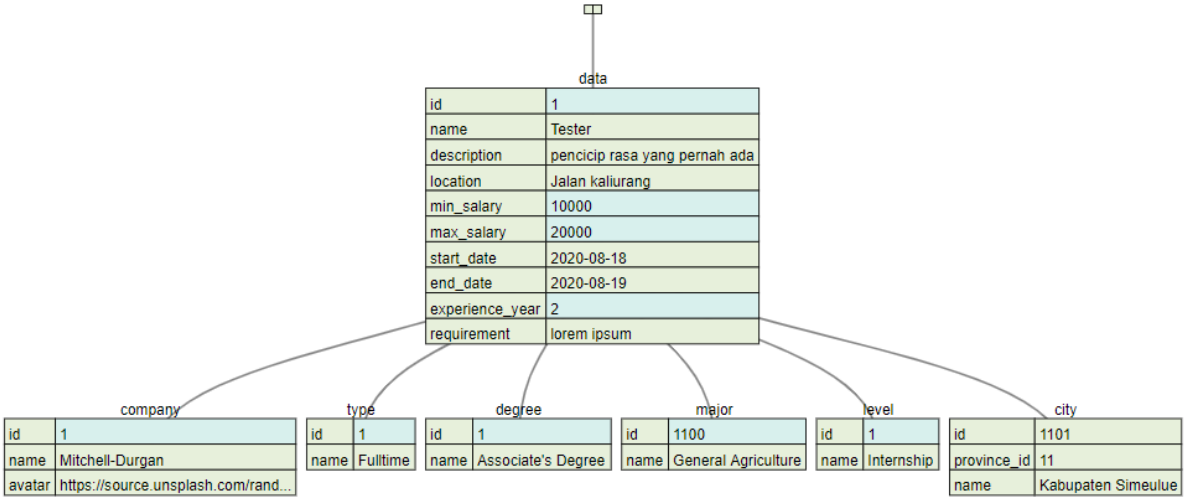


Gambar 4.3 Data *User*

4.1.2.3 Data Pekerjaan

Data pekerjaan merupakan data yang akan digunakan untuk data pekerjaan dan pekerjaan terlamar yang berhubungan dengan antarmuka daftar pekerjaan, detail pekerjaan, daftar

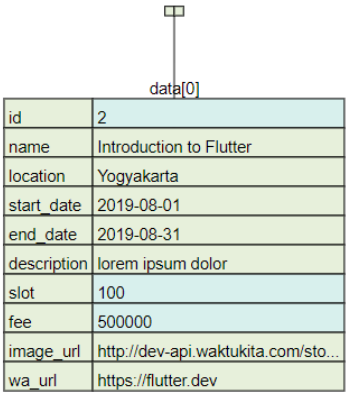
pekerjaan terlamar dan detail pekerjaan terlamar. Gambar 4.4 merupakan diagram pohon *JSON* data *user*.



Gambar 4.4 Data Pekerjaan

4.1.2.4 Data Kelas

Data kelas merupakan data yang akan digunakan untuk data daftar kelas dan detail yang berhubungan dengan antarmuka daftar kelas dan antarmuka detail kelas. Gambar 4.5 merupakan diagram pohon *JSON* data kelas.



Gambar 4.5 Data Kelas

Pada class model bootcamp untuk data kelas, kita akan membuat constructor dengan parameter *id, name, location, startDate, endDate, description, slot, fee, imageUrl* dan *waUrl*.

Kode program untuk membuat constructor dengan parameter tersebut dapat dilihat pada Gambar 4.6.

```
class Data {
int id;
String name;
String location;
String startDate;
String endDate;
String description;
int slot;
int fee;
String imageUrl;
String waUrl;

  Data({
    this.id,
    this.name,
    this.location,
    this.startDate,
    this.endDate,
    this.description,
    this.slot,
    this.fee,
    this.imageUrl,
    this.waUrl
  });

}
```

Gambar 4.6 *Script Constructor Data Bootcamp*

Setelah membuat constructor dengan paramaeter tersebut, kita perlu melakukan konversi dari Map ke Class Model dengan cara membuat Named Constructor menggunakan factory, agar ketika named constructor dipanggil, tidak akan membuat objek baru. Kode pembuatan Named Constructor dapat dilihat pada Gambar 4.7.

```
 factory Data.fromJson(Map<String, dynamic> map) {
    return Data(
      id: map["id"],
      name: map["name"],
      location: map["location"],
      startDate: map["start_date"],
      endDate: map["end_date"],
      description: map["description"],
      slot: map["slot"],
      fee: map["fee"],
      imageUrl: map["image_url"],
      waUrl: map["wa_url"],
    );
  }
```

Gambar 4.7 *Script Named Constructor Data Bootcamp*

Karena kelas Data berbentuk list dan berada di dalam *JSON* Object “data”, maka diperlukan constructor dan named constructor agar dapat membaca *JSON* Object “data” yang dapat dilihat pada Gambar 4.8.

```
 class Bootcamps {
  List<Data> data;

  Bootcamps ({
    this.data
  });

  factory Bootcamps.fromJson(Map<String, dynamic> map) {
    return Bootcamps (
      data : List<Data>.from(map["data"].map((data) {
        return Data.fromJson(data);
      })),
    );
  }
}
```

Gambar 4.8 *Script Constructor* dan *Named Constructor Bootcamp*

Setelah membuat semua method konversi dari respon *API* ke dalam model, kemudian kita perlu membuat fungsi untuk melakukan konversi dari class model ke dalam *JSON* format dalam bentuk string. Fungsi tersebut dapat dilihat pada Gamabar 4.9.

```
Bootcamps bootcampsFromJson(String jsonData) {
  final data = json.decode(jsonData);
  return Bootcamps.fromJson(data);
}
```

Gambar 4.9 *Script* Konversi *Class Model* ke *JSON* String

4.1.3 Presenter

Presenter merupakan script untuk menghubungkan komunikasi antara Model dan View. Interaksi yang dilakukan akan menggunakan presenter untuk memproses dan mengakses model lalu mengembalikan respon kepada View. Contoh script presenter untuk menghubungkan model kelas dan view daftar kelas seperti pada Gambar 4.10.

```
Future<Bootcamps> getBootcamps(String name) async {
```

```
    SharedPreferences          sharedPreferences          =          await
SharedPreferences.getInstance();
    final response = await client.get(
      "$baseUrl/talent/bootcamps?filter[name]="+name,
      headers: {
        HttpHeaders.authorizationHeader:
            "Bearer" + sharedPreferences.getString("token"),
      },
    );
    if (response.statusCode == 200) {
      return bootcampsFromJson(response.body);
    } else {
      return null;
    }
  }
```

Gambar 4.10 *Script* Presenter Bootcamp (Kelas)

Pada Gambar 4.10 getBootcamps menggunakan metode GET untuk mengambil data dari url $baseUrl/talent/bootcamps dan menggunakan header authorization bearer token. Jika statuscode dari hasil response adalah 200 (Status OK), maka program akan memanggil fungsi bootcampsFromJson yang berasal dari model Bootcamps, jika response menghasilkan statuscode selain 200, maka program akan memunculkan nilai null.

4.1.4 View

View merupakan script untuk menampilkan *user* interface dan data dari presenter. View dalam flutter berupa kumpulan *widget*. Contoh implementasi script Daftar Kelas seperti pada Gambar 4.11.

```
class BootcampPage extends Stateful Widget {
  @override
  _BootcampPageState createState() => _BootcampPageState();
}

class _BootcampPageState extends State<BootcampPage> {
  Future<Bootcamps> bootcamp;
  var bootcampName = "";

  @override
  void initState() {
    super.initState();
    bootcamp = ApiService().getBootcamps(bootcampName);
  }

  final TextEditingController searchController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
```

```
      body: FutureBuilder<Bootcamps>(
        future: bootcamp,
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(
              child: Text(
                  "Something          wrong          with          message:
${snapshot.error.toString()}"),
              );
          } else if (snapshot.connectionState == ConnectionState.done) {
            Bootcamps bootcamps = snapshot.data;
            if (bootcamps.data.length == 0) {
              return Center(
                child: Text("Bootcamp is empty"),
              );
            } else {
              // print(bootcamps.data);
              return listBootcamps(bootcamps);
            }
          } else {
            return Center(
              child: CircularProgressIndicator(),
            );
          }
        },
      ),
    );
  }

  Widget listBootcamps(bootcamps) {
    return CustomScrollView(
      slivers: <Widget>[
        SliverAppBar(
          backgroundColor: Colors.white,
          elevation: 0,
          pinned: true,
          snap: false,
          floating: false,
          expandedHeight: 100.0,
          flexibleSpace: FlexibleSpaceBar(
            title: Text(
              "Discover",
              style: TextStyle(
                  color:          Colors.blueGrey[800],          fontWeight:
FontWeight.bold),
            ),
            titlePadding: EdgeInsetsDirectional.only(start: 20, bottom: 16),
          ),
          actions: <Widget>[
            IconButton(
              onPressed: () {
                showBottomSheet<String>(
                  context: context,
                  builder: (BuildContext context) => Container(
                    decoration: BoxDecoration(
                      color: Colors.white,
                      border:          Border(top:          BorderSide(color:
Colors.teal[800])),
                    ),
                    child: ListView(
```

```
                    padding: EdgeInsets.fromLTRB(40, 5, 40, 5),
                    shrinkWrap: true,
                    primary: false,
                    children: <Widget>[
                      Padding(
                        padding: EdgeInsets.fromLTRB(0, 10, 0, 10),
                        child: Text("Search Class",
                            style: TextStyle(
                                fontSize: 20, color: Colors.teal[800])),
                      ),
                      TextFormField(
                        controller: searchController,
                        cursorColor: Colors.amber[600],
                        keyboardType: TextInputType.text,
                        decoration: InputDecoration(
                          hintText: 'Class name',
                          labelText: 'Search',
                        ),
                      ),
                      ButtonTheme.bar(
                        child: ButtonBar(
                          children: <Widget>[
                            FlatButton(
                              child: Text('Cancel'),
                              onPressed: () {
                                Navigator.pop(context);
                              },
                            ),
                            OutlineButton(
                              child: Text('Search',
                                  style: TextStyle(
                                      fontWeight: FontWeight.bold,
                                      color: Colors.teal[800])),
                              onPressed: () {
                                setState(() {
                                  bootcampName = searchController.text;
                                  bootcamp =
ApiService().getBootcamps(bootcampName);
                                });
                              },
                            ),
                          ],
                        ),
                      ),
                    ],
                  ),
                ),
              );
            },
            icon: SvgPicture.asset('assets/icons/search.svg',
                color: Colors.blueGrey[800]),
          )
        ],
      ),
      SliverList(
        delegate:
            SliverChildBuilderDelegate((BuildContext context, int index) {
          return Container(
              padding: EdgeInsets.fromLTRB(20, 0, 20, 15),
```

```
                child: InkWell(
                  onTap: () {
                    Navigator.push(context,
                        MaterialPageRoute(builder: (context) {
                      return DetailBootcampPage(id: index);
                    }));
                  },
                  child: Card(
                    elevation: 0.0,
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: <Widget>[
                        SizedBox(
                          height: 120,
                          width: double.infinity,
                          child: ClipRRect(
                              borderRadius: new BorderRadius.circular(5.0),
                              child: Image.network(
                                  bootcamps.data[index].imageUrl,
                                  fit: BoxFit.cover)),
                        ),
                        Container(
                          padding: const EdgeInsets.only(top: 5),
                          child: Text(
                            bootcamps.data[index].location,
                            style: TextStyle(
                                fontSize: 12, color: Colors.blueGrey[300]),
                          ),
                        ),
                        Container(
                          padding: const EdgeInsets.only(top: 5),
                          child: Text(
                            bootcamps.data[index].name,
                            maxLines: 2,
                            style: TextStyle(
                                fontWeight: FontWeight.bold,
                                color: Colors.blueGrey[800]),
                          ),
                        ),
                      ],
                    ),
                  ),
                ));
          }, childCount: bootcamps.data.length),
        ),
      ],
    );
  }
}
```

Gambar 4.11 *Script* View Daftar Kelas

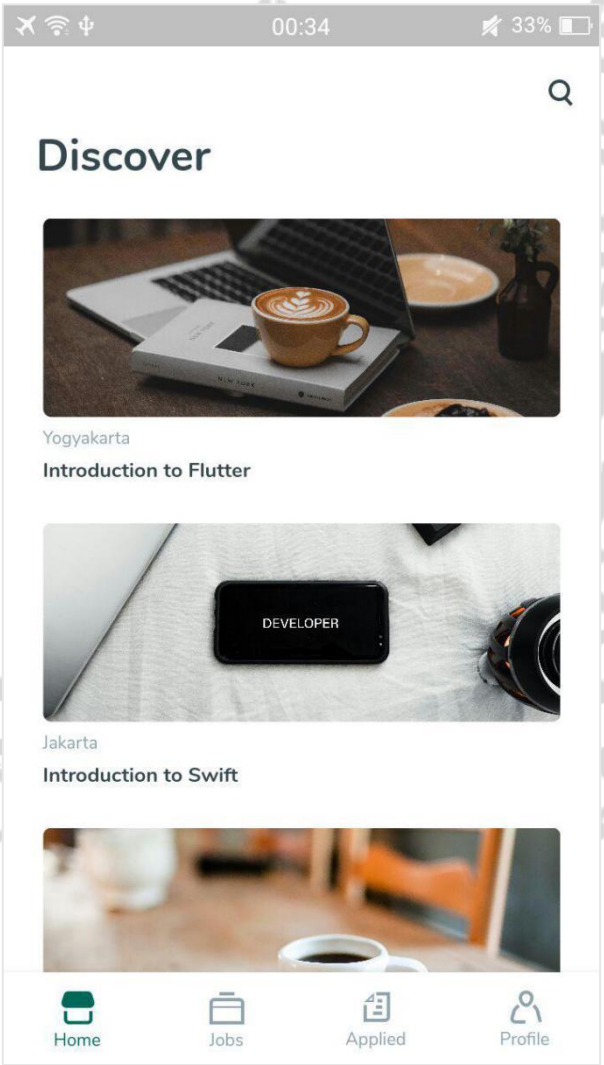
Pada Gambar 4.11 FutureBuilder digunakan untuk membuild tampilan daftar kelas, jika ada error, maka yang akan ditampilkan adalah button refresh dan jika data kelas kosong, maka akan ditampilkan “Bootcamp is Empty”.

Pada SliverAppBar, terdapat action *icon* search yang ketika ditekan maka akan muncul form pencarian yang ada dalam *Widget* BottomSheet.

Pada SliverList berisikan *Widget* Card yang menghandle *widget* gambar, text lokasi dan judul dari bootcamp kelas tersebut.

4.1.4.1 Daftar Kelas

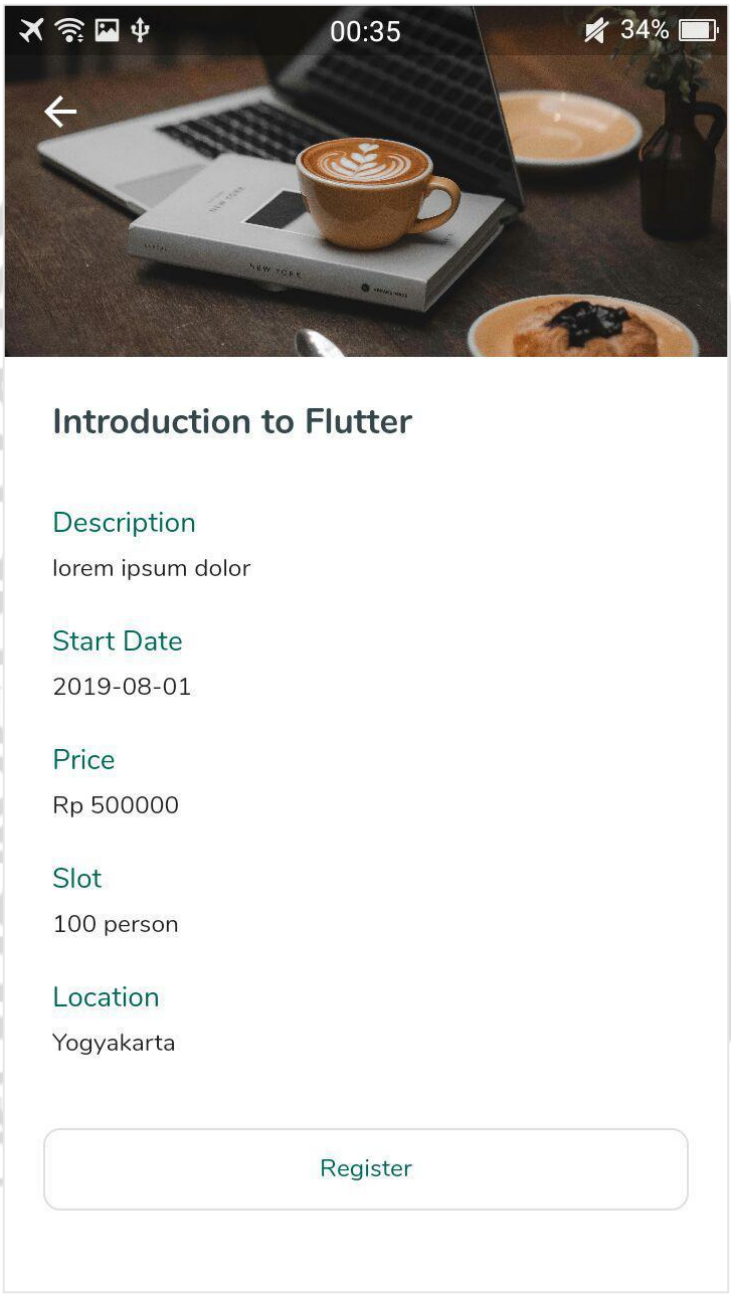
Saat pengguna pertama kali login, halaman yang akan ditampilkan adalah Daftar Kelas (Menu “Home”) pada aplikasi yang berisi kelas-kelas yang terdaftar dalam system. Tampilan Halaman Daftar Kelas seperti pada Gambar 4.12.



Gambar 4.12 Halaman Daftar Kelas

4.1.4.2 Detail Kelas

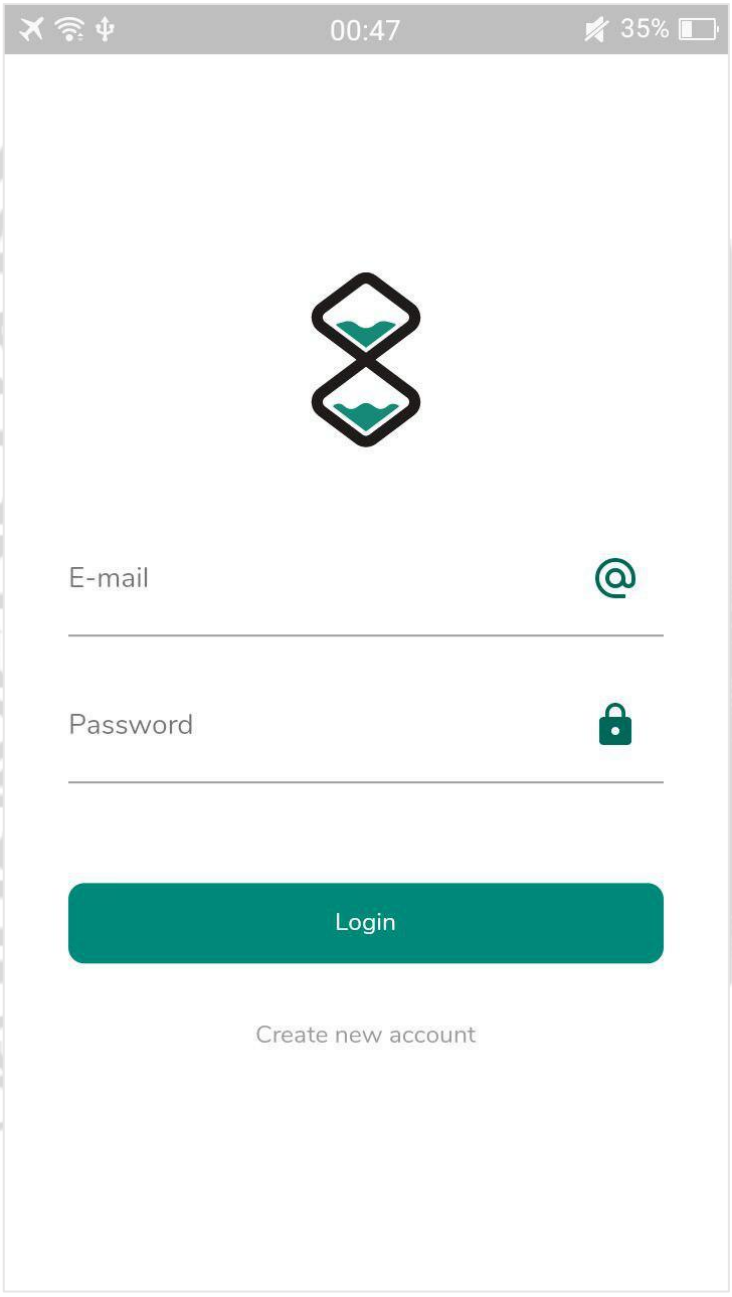
Pengguna dapat mengakses detail informasi kelas dengan cara memilih salah satu kelas yang ada dalam Daftar Kelas. Tampilan Detail Kelas seperti pada Gambar 4.13.



Gambar 4.13 Halaman Detail Kelas

4.1.4.3 Login

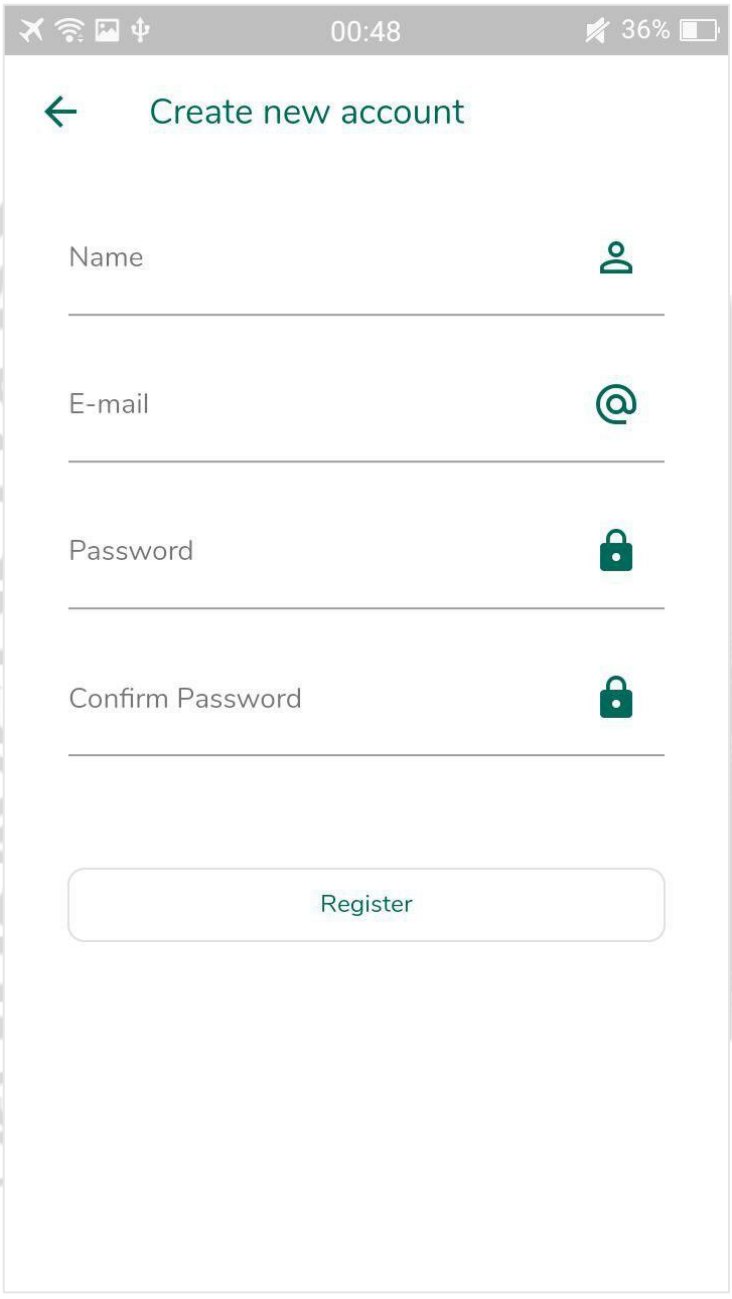
Aplikasi akan menampilkan halaman login ketika pengguna pertama kali membuka aplikasi. Terdapat form login dan tombol register untuk menuju halaman register untuk mendaftarkan akun baru. Tampilan halaman login seperti pada Gambar 4.14.



Gambar 4.14 Halaman Login

4.1.4.4 Register

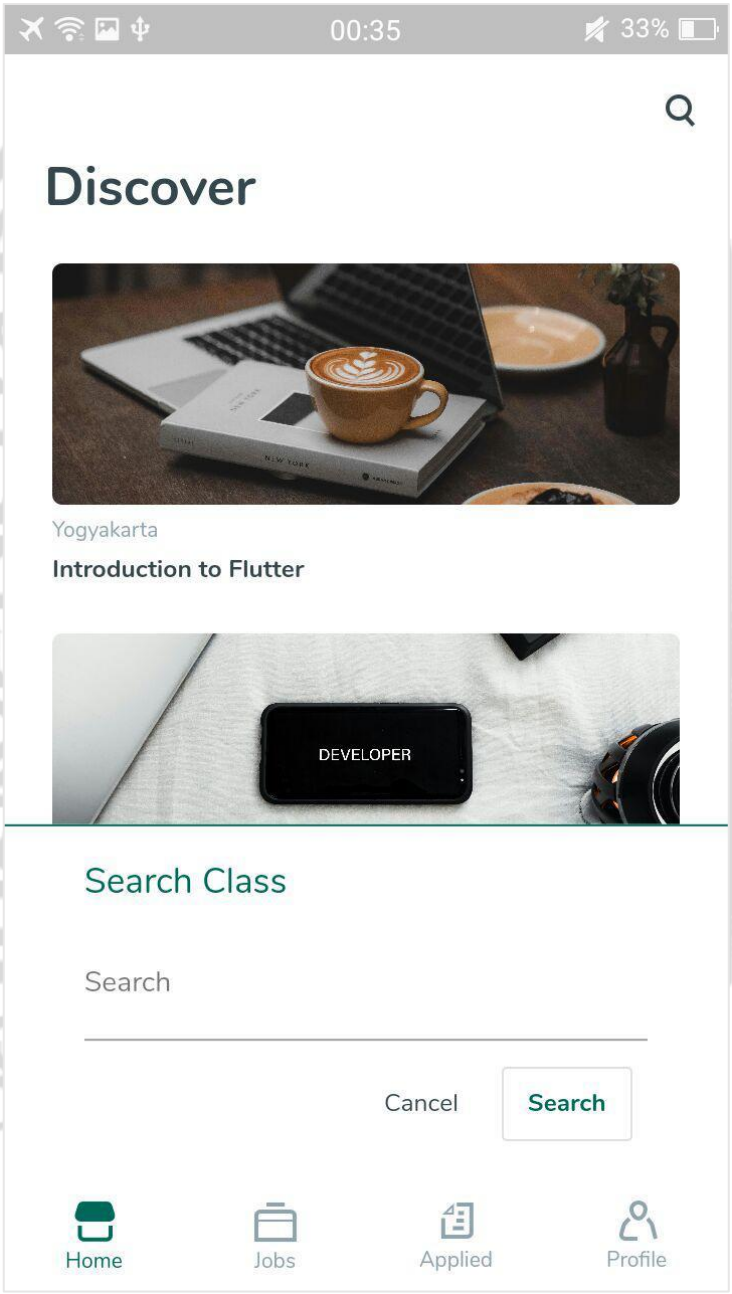
Pengguna harus mendaftarkan akun dan memverifikasi *email* melalui form register sebelum dapat login ke aplikasi. Tampilan form login seperti pada Gambar 4.13.



Gambar 4.15 Halaman Register

4.1.4.5 Cari Kelas

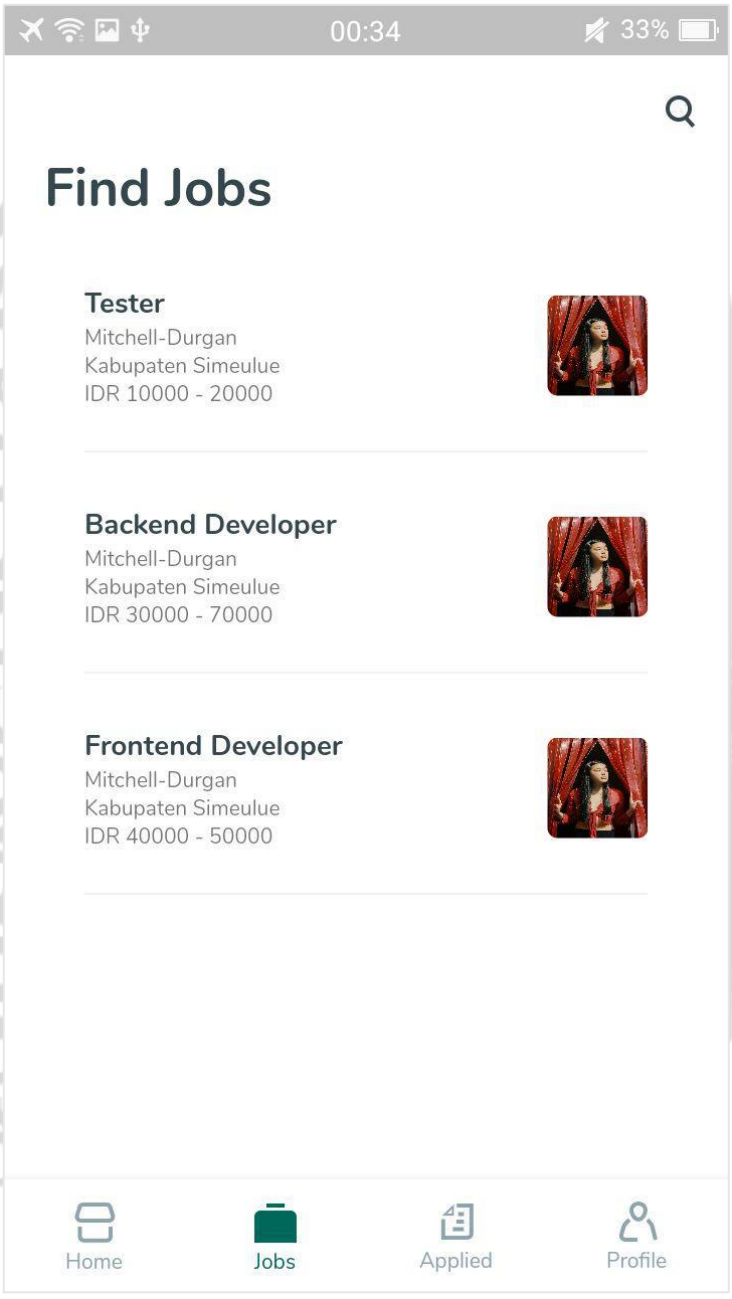
Pengguna dapat mencari kelas berdasarkan nama kelas yang diinginkan dengan cara menekan *icon* pencarian, setelah itu aplikasi akan menampilkan hasil pencarian kelas. Tampilan form pencarian kelas seperti pada Gambar 4.16.



Gambar 4.16 Form Cari Kelas

4.1.4.6 Daftar Pekerjaan

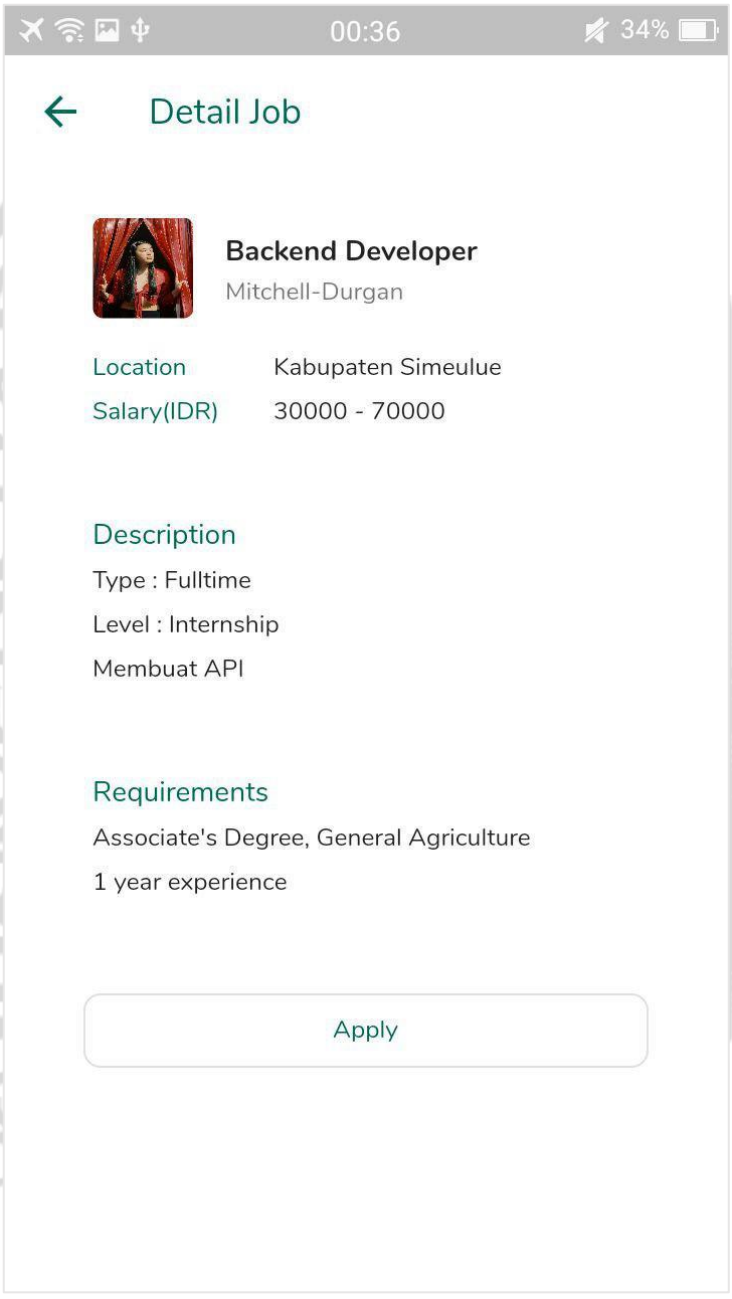
Pengguna dapat melihat daftar pekerjaan yang terdaftar dalam system dengan cara mengakses menu “Jobs” dalam aplikasi. Tampilan Daftar Pekerjaan seperti pada Gambar 4.17.



Gambar 4.17 Halaman Daftar Pekerjaan

4.1.4.7 Detail Pekerjaan

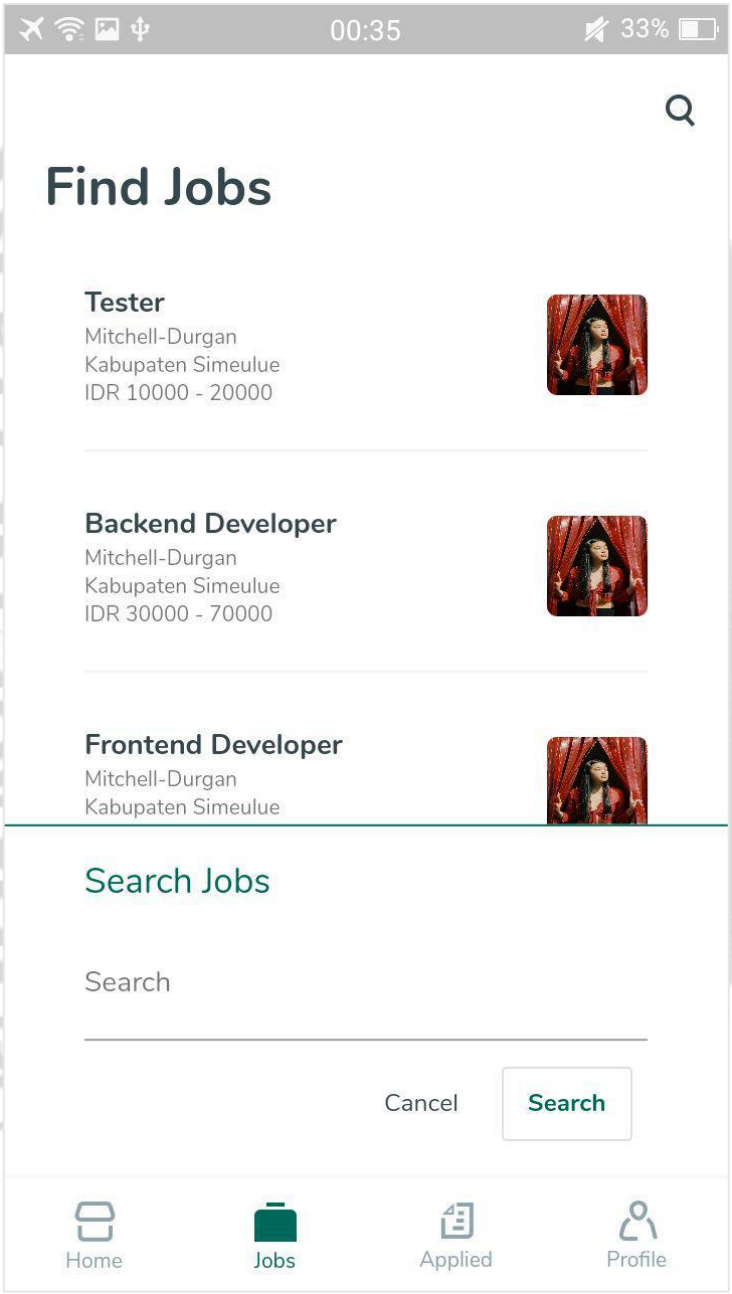
Pengguna dapat melamar pekerjaan dengan cara menekan tombol “Apply” yang terdapat dalam halaman Detail Pekerjaan seperti pada Gambar 4.18.



Gambar 4.18 Halaman Detail Pekerjaan

4.1.4.8 Cari Pekerjaan

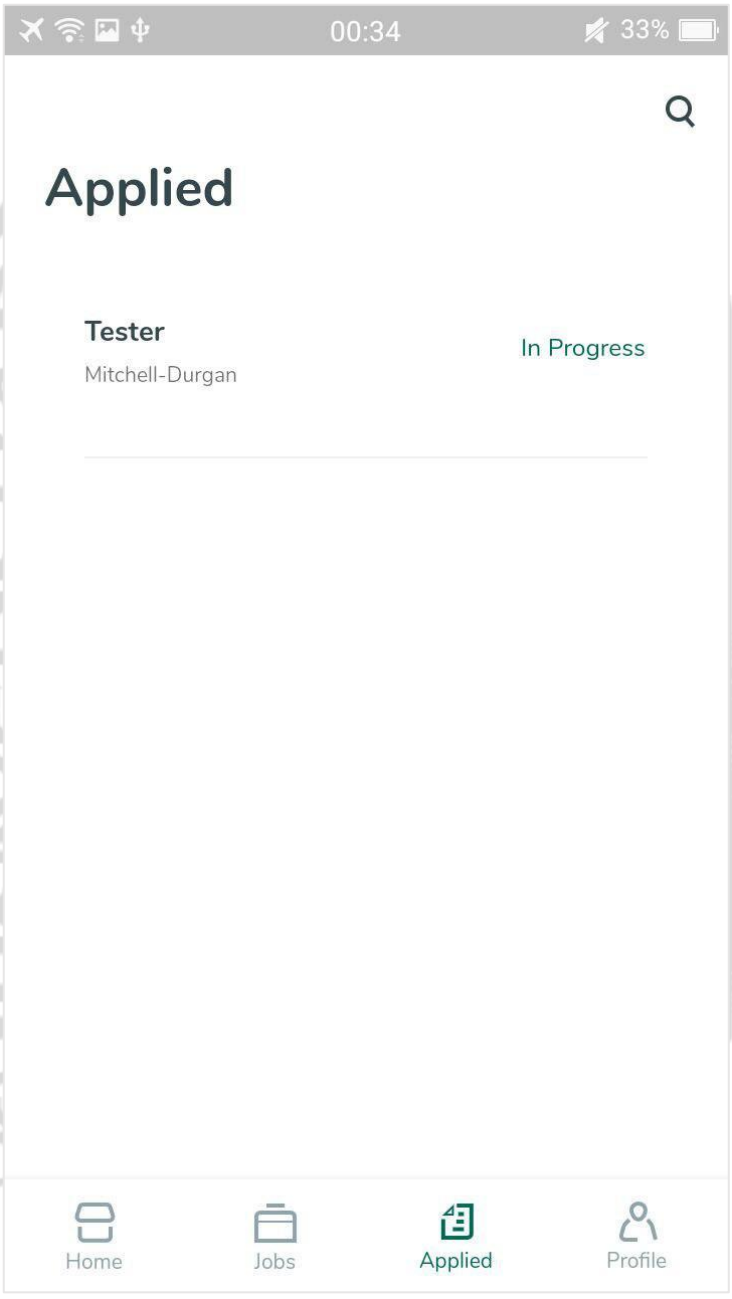
Pengguna dapat mencari pekerjaan berdasarkan nama pekerjaan yang diinginkan dengan cara menekan *icon* pencarian, setelah itu aplikasi akan menampilkan hasil pencarian pekerjaan. Tampilan form pencarian pekerjaan seperti pada Gambar 4.19.



Gambar 4.19 Form Cari Pekerjaan

4.1.4.9 Daftar Pekerjaan Terlamar

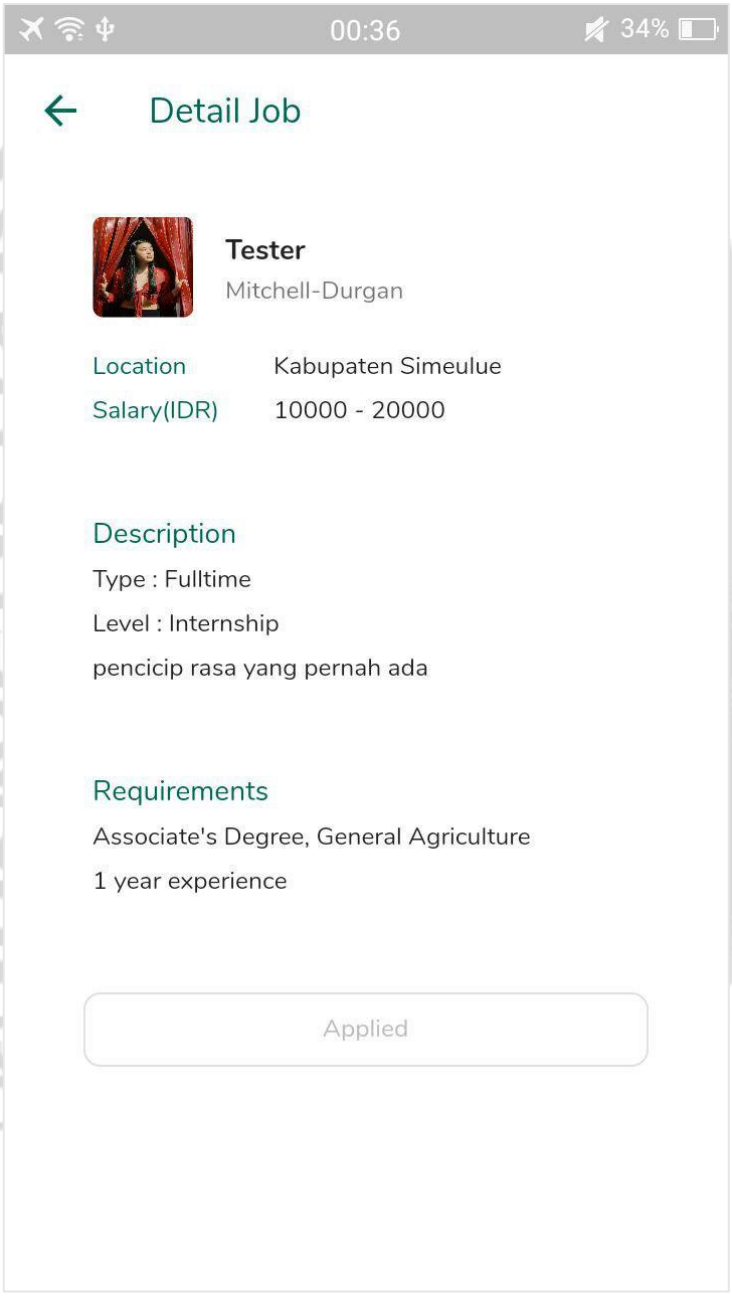
Pengguna dapat meilihat daftar pekerjaan terlamar dengan cara mengakses menu "Applied" dalam aplikasi. Tampilan Daftar Pekerjaan Terlamar seperti pada Gambar 4.20.



Gambar 4.20 Halaman Daftar Pekerjaan Terlamar

4.1.4.10 Detail Pekerjaan Terlamar

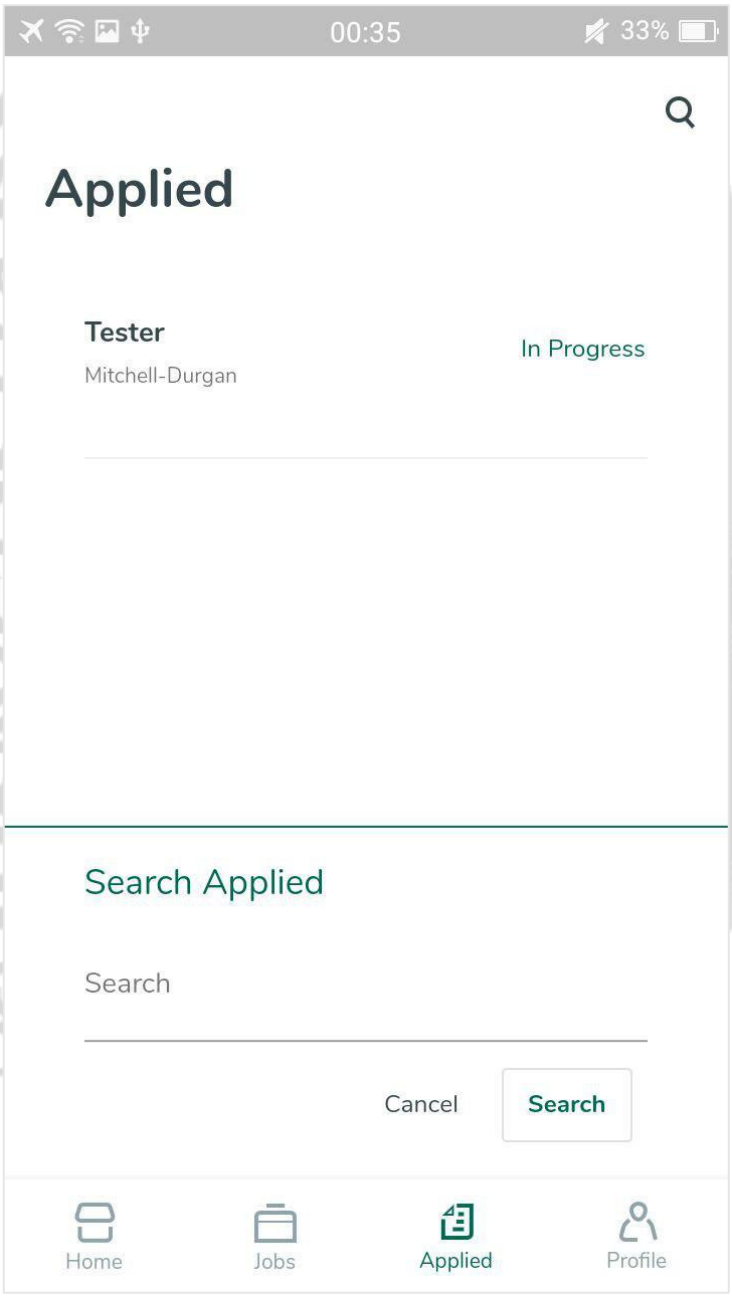
Pengguna dapat mengakses detail informasi pekerjaan terlamar dengan cara memilih salah satu pekerjaan terlamar yang ada dalam Daftar Pekerjaan Terlamar. Tampilan Detail Pekerjaan Terlamar seperti pada Gambar 4.21.



Gambar 4.21 Halaman Detail Pekerjaan Terlamar

4.1.4.11 Cari Pekerjaan Terlamar

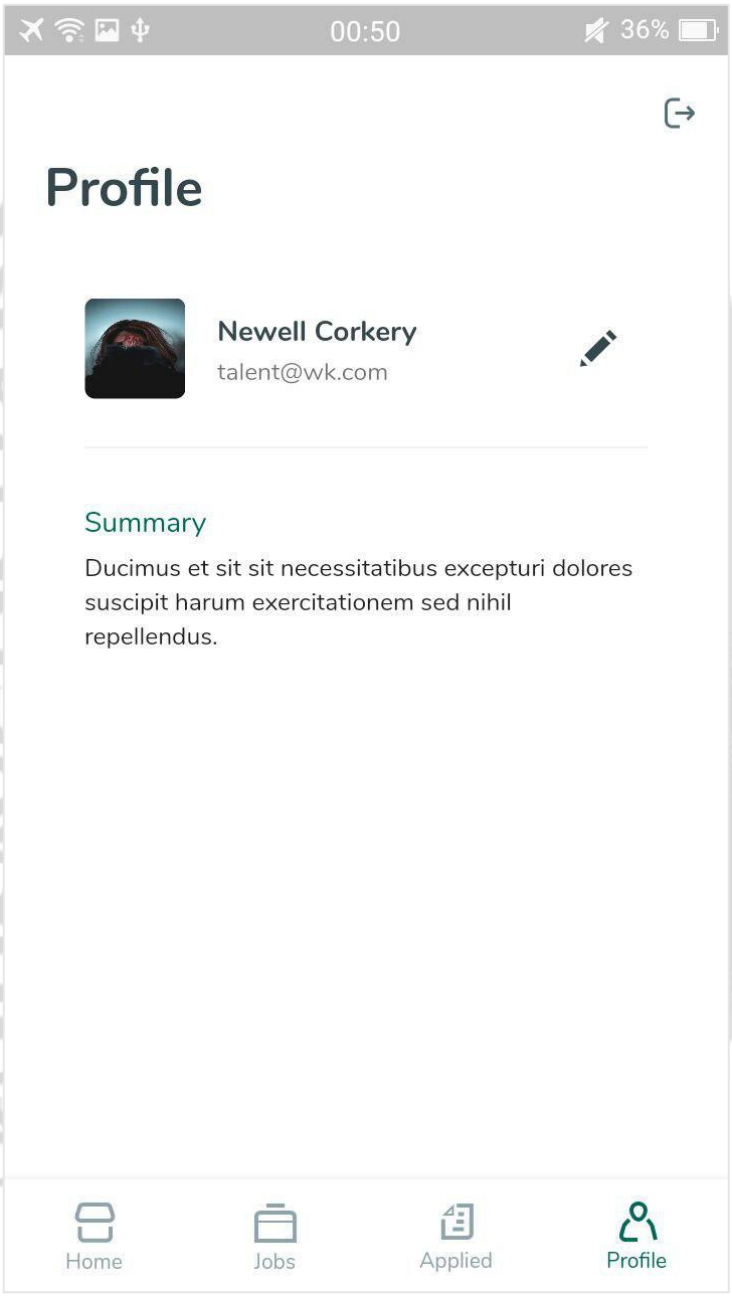
Pengguna dapat mencari pekerjaan terlamar berdasarkan nama pekerjaan yang diinginkan dengan cara menekan *icon* pencarian, setelah itu aplikasi akan menampilkan hasil pencarian pekerjaan terlamar. Tampilan form pencarian pekerjaan terlamar seperti pada Gambar 4.22.



Gambar 4.22 Form Cari Pekerjaan Terlamar

4.1.4.12 Lihat Profil

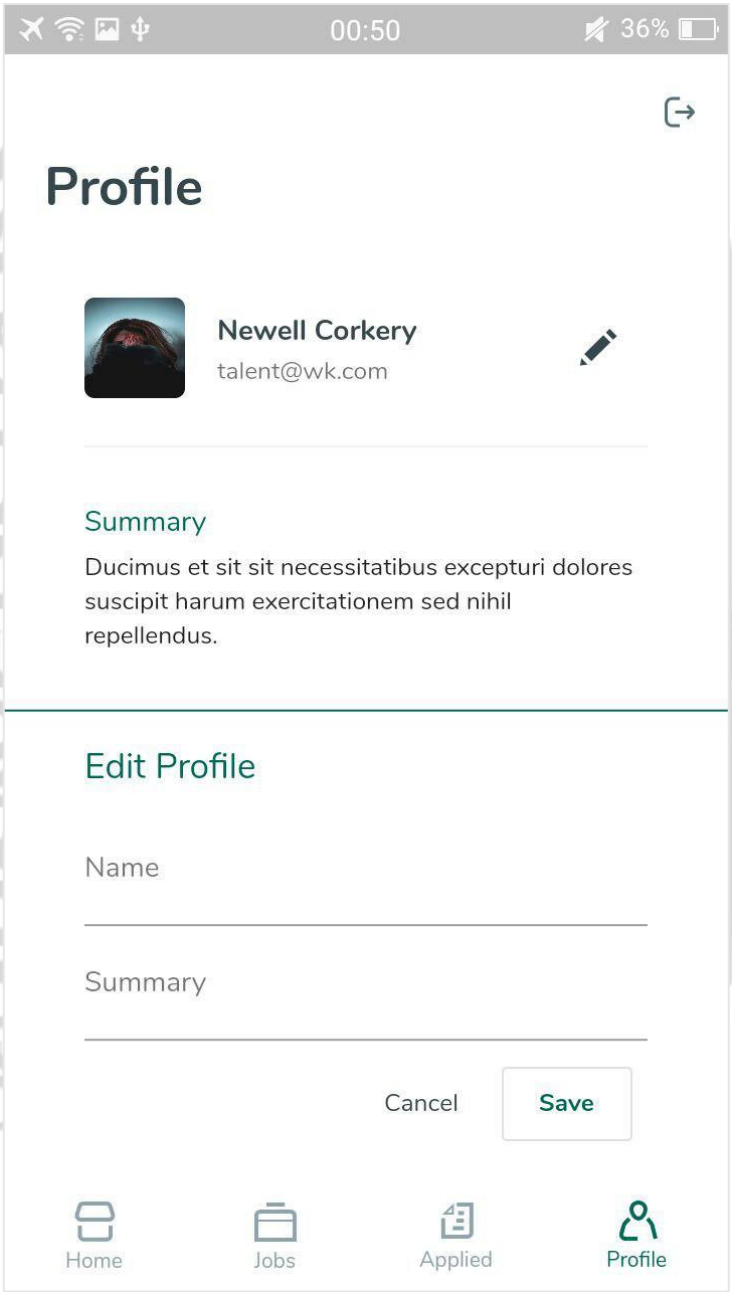
Pengguna dapat melihat profil akun dalam aplikasi dengan cara mengakses menu "Pro*file*" dalam aplikasi. Tampilan halaman Pro*file* seperti pada Gambar 4.23.



Gambar 4.23 Halaman Profil

4.1.4.13 Sunting Profil

Pengguna dapat menyunting data akunnya melalui aplikasi dengan cara menekan *icon* edit pada halaman pro*file* yang kemudian akan muncul form untuk menyunting informasi profil. Tampilan form edit profil seperti pada Gambar 4.24.



Gambar 4.24 Form Sunting Profil

4.2 Pengujian Fungsionalitas

Pengujian Fungsionalitas dilakukan untuk menguji fitur-fitur yang dirancang dan diimplementasikan dapat berjalan dengan baik.

Tabel 4.2 Pengujian Fungsionalitas

No	Aktivitas	Hasil	Kesimpulan
1	Registrasi	Proses pendaftaran berhasil	Berhasil
2	Login	Proses login berhasil	Berhasil
3	Lihat Daftar Kelas	Menampilkan daftar kelas	Berhasil
4	Lihat Detail Kelas	Menampilkan detail kelas	Berhasil
5	Cari Kelas	Menampilkan hasil pencarian kelas	Berhasil
6	Lihat Daftar Pekerjaan	Menampilkan daftar pekerjaan	Berhasil
7	Lihat Detail Pekerjaan	Menampilkan detail pekerjaan	Berhasil
8	Lamar Pekerjaan	Proses lamar pekerjaan berhasil	Berhasil
9	Cari Pekerjaan	Menampilkan hasil pencarian pekerjaan	Berhasil
10	Lihat Daftar Pekerjaan Terlamar	Menampilkan daftar pekerjaan terlamar	Berhasil
11	Lihat Detail Pekerjaan Terlamar	Menampilkan detail pekerjaan terlamar	Berhasil
12	Cari Pekerjaan Terlamar	Menampilkan hasil pencarian pekerjaan terlamar	Berhasil
13	Lihat Profil	Menampilkan halaman profil	Berhasil
14	Sunting Profil	Proses sunting berhasil	Berhasil

4.3 Pengujian Usabilitas

Pengujian Usabilitas dilakukan pada 10 responden yang menjadi target pengguna aplikasi waktukita.com.

Tabel 4.3 merupakan hasil pengujian usabilitas menggunakan kuisioner dan dihitung menggunakan skala linkert.

Tabel 4.3 Pengujian Usabilitas

No	Pertanyaan	STS	KS	C	S	SS	Persentase
		1	2	3	4	5	
1	Form Registrasi mudah digunakan dan berjalan baik		1	4	4	1	68%
2	Form Login mudah digunakan dan berjalan baik			4	4	2	76%
3	Fitur dalam menu Home mudah digunakan dan berjalan baik		1	3	5	1	72%
4	Fitur dalam menu Jobs mudah digunakan dan berjalan baik		1	3	5	1	72%
5	Fitur dalam menu applied mudah digunakan dan berjalan baik		1	3	5	1	72%
6	Fitur dalam menu pro*file* mudah digunakan dan berjalan baik		2	5	3		62%
7	Apakah *user interface* sudah baik?		1	5	3	1	68%
Rerata Presentase							**70%**

4.4 Hasil Pengujian

Berdasarkan hasil pengujian fungsionalitas, aplikasi dapat berjalan dengan baik. Dengan hasil tersebut maka dapat diharapkan untuk memudahkan startup Waktukita.com mengembangkan aplikasi *mobile*nya.

Berdasarkan hasil pengujian usabilitas, rerata persentase yang didapatkan adalah 70%, Dengan hasil tersebut diharapkan dapat dijadikan evaluasi untuk meningkatkan aspek usabilitas aplikasi.