BAB IV

HASIL DAN PEMBAHASAN

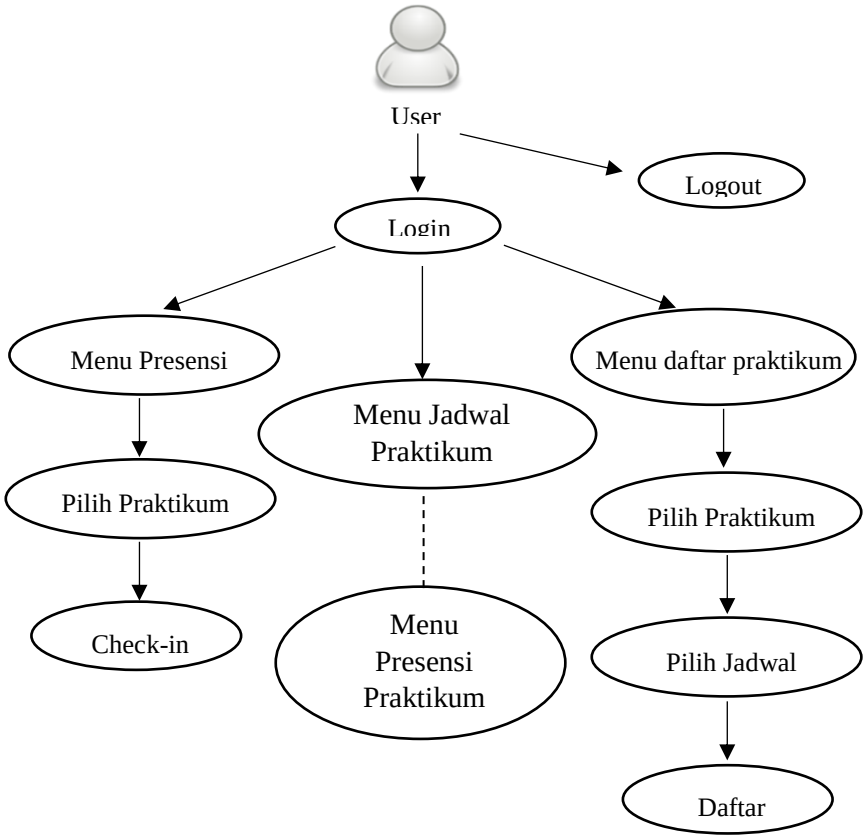
4.1 Perancangan Sistem

Perancangan sistem pada aplikasi presensi otomatis iPresence menggunakan metode *Unified Model Language* (UML). Adapun model UML yang digunakan yaitu use case diagram dan activity diagram.

4.1.1 Rancangan Proses

a. Use Case Diagram

Berdasarkan analisis kebutuhan yang telah ada maka dibuatlah use case diagram untuk membantu pembuatan logic aplikasi. Gambar use case diagram bisa dilihat pada gambar di bawah ini.



Gambar 4.1 *Use Case Diagram User*

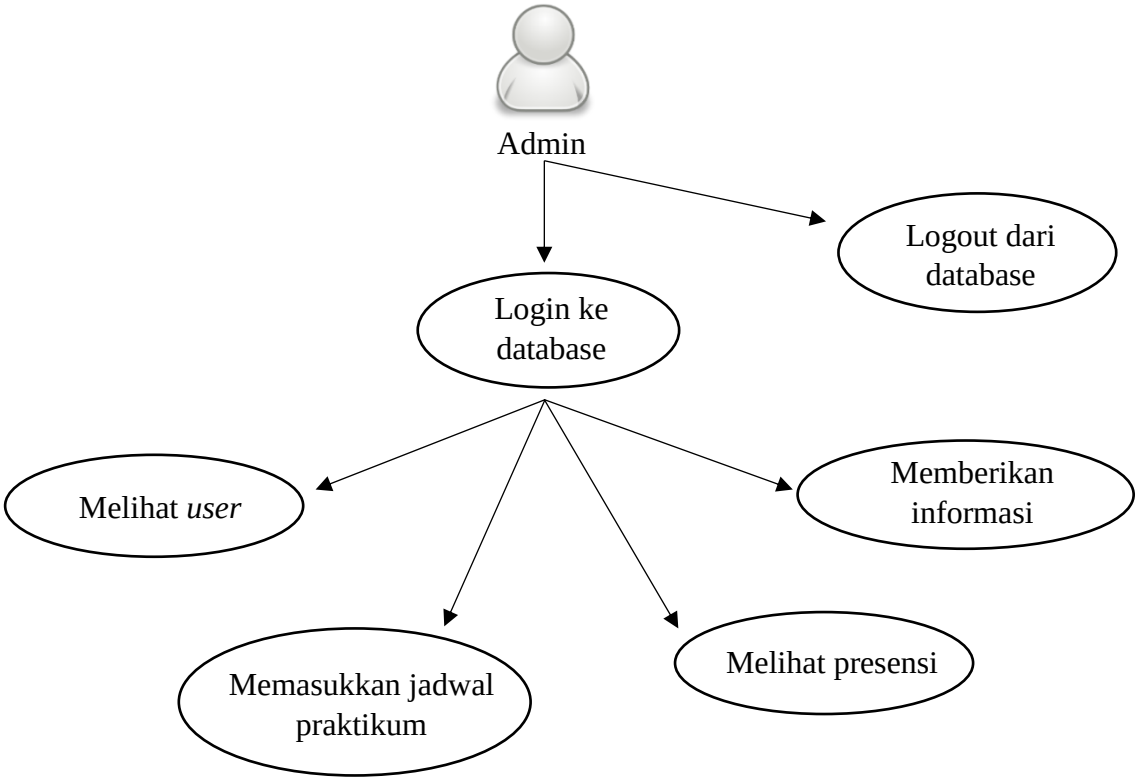
Aktor merupakan pihak yang terlibat dalam aplikasi. Pihak-pihak yang terlibat dibedakan menjadi dua yaitu admin dan pengguna (mahasiswa) yang memiliki hak akses masing-masing. Setiap aktor yang terlibat dalam sistem diasosiasikan dengan *use case* yang ada.

Dalam beberapa *use case* aktor dapat muncul dalam *use case* yang sama. *Use case diagram* untuk pengguna aplikasi bisa dilihat di gambar 4.1.

Adapun penjelasan *use case diagram user* di atas adalah sebagai berikut:

1. Login: Memungkinkan *user* untuk melakukan login atau masuk ke dalam aplikasi.
2. Melihat menu lihat praktikum: Memungkinkan *user* untuk melihat jadwal praktikum beserta lokasinya (ruangan).
3. Menu Presensi: Memungkinkan *user* untuk melakukan presensi praktikum.
4. Menu daftar: Memungkinkan *user* untuk melihat praktikum yang diselenggarakan selama satu semester oleh jurusan Teknik Elektro UMY.
5. Menu pilih praktikum: Memungkinkan *user* untuk memilih praktikum yang diambil selama satu semester.
6. Menu pilih jadwal: Memungkinkan *user* untuk memilih jadwal praktikum yang akan dilaksanakan.
7. Menu tombol daftar: Untuk menyimpan data praktikum yang diambil ke dalam *database user*.
8. Menu jadwal praktikum: Memungkinkan *user* untuk melihat jadwal praktikum yang telah diambil.
9. Menu presensi praktikum: Memungkinkan *user* untuk melihat data presensi praktikum yang telah diambil.
10. Logout: Memungkinkan *user* untuk keluar dari aplikasi.

Sedangkan *use case diagram* untuk admin adalah sebagai berikut:



Gambar 4.2 *Use Case Diagram* Admin

Adapun penjelasan *use case diagram* admin di atas adalah sebagai berikut:

1. Login ke database: Memungkinkan admin untuk masuk ke sistem database yang telah dibuat.
2. Melihat *user*: Memungkinkan admin untuk melihat user dari pengguna aplikasi yang terdapat pada menu **User** database.
3. Memasukkan jadwal praktikum: Memungkinkan admin untuk memasukkan jadwal praktikum yang telah ada ke dalam sistem database. Admin dapat memasukkan data praktikum pada menu **Bucket** database.
4. Melihat presensi: Memugkinkan admin untuk melihat presensi praktikum mahasiswa berdasarkan praktikum yang mahasiswa ambil. Admin dapat

melihat presensi praktikum mahasiswa pada menu Bucket di database sesuai dengan nama praktikum.

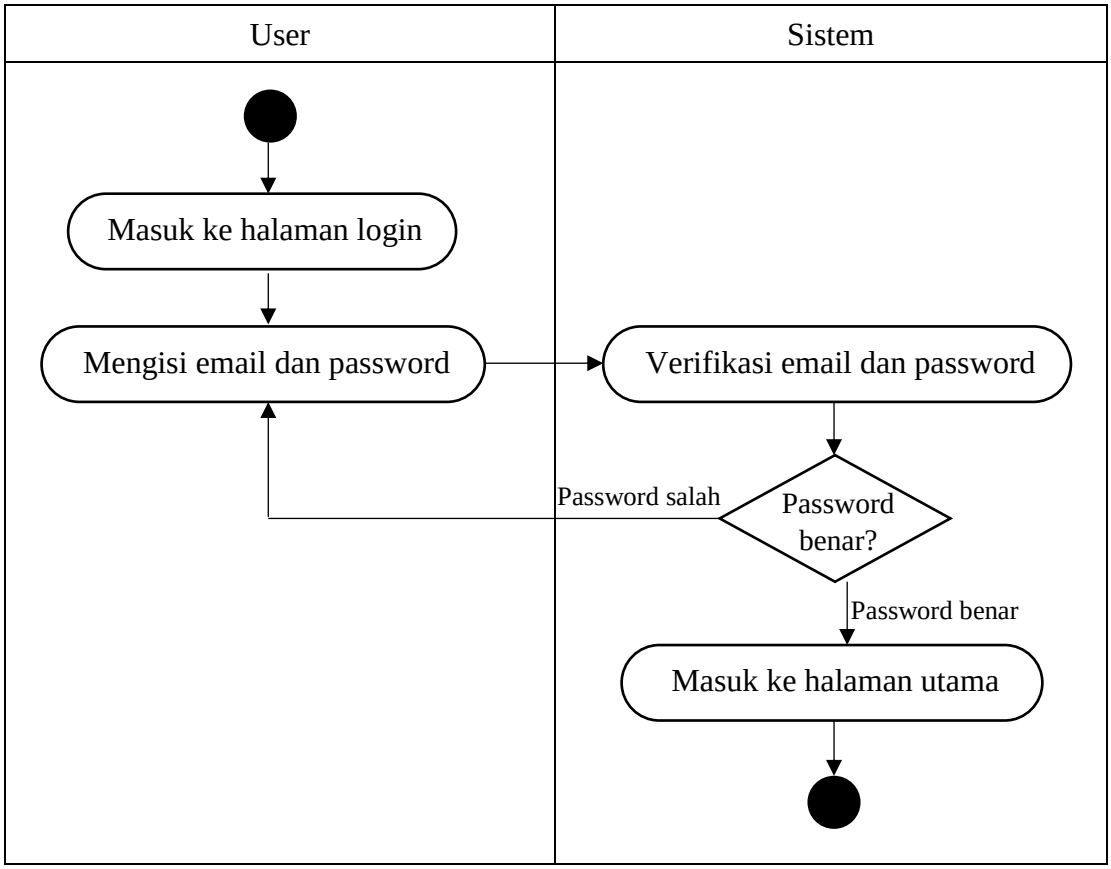
5. Memberikan informasi: Memungkinkan admin untuk memberikan informasi kepada *user* aplikasi (mahasiswa) mengenai informasi praktikum. Hal ini dapat dilakukan admin pada menu **Push Notification** pada database. Menu ini akan secara langsung memberikan informasi kepada mahasiswa.
6. Logout: Memungkinkan admin untuk keluar dari database.

b. Activity Diagram

Acitvity diagram atau diagram aktivitas menggambarkan berbagai alir aktivitas dalam sistem yang telah dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana *decision* itu berakhir. *Acitvity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi dari aplikasi.

1. *Activity Diagram* Login

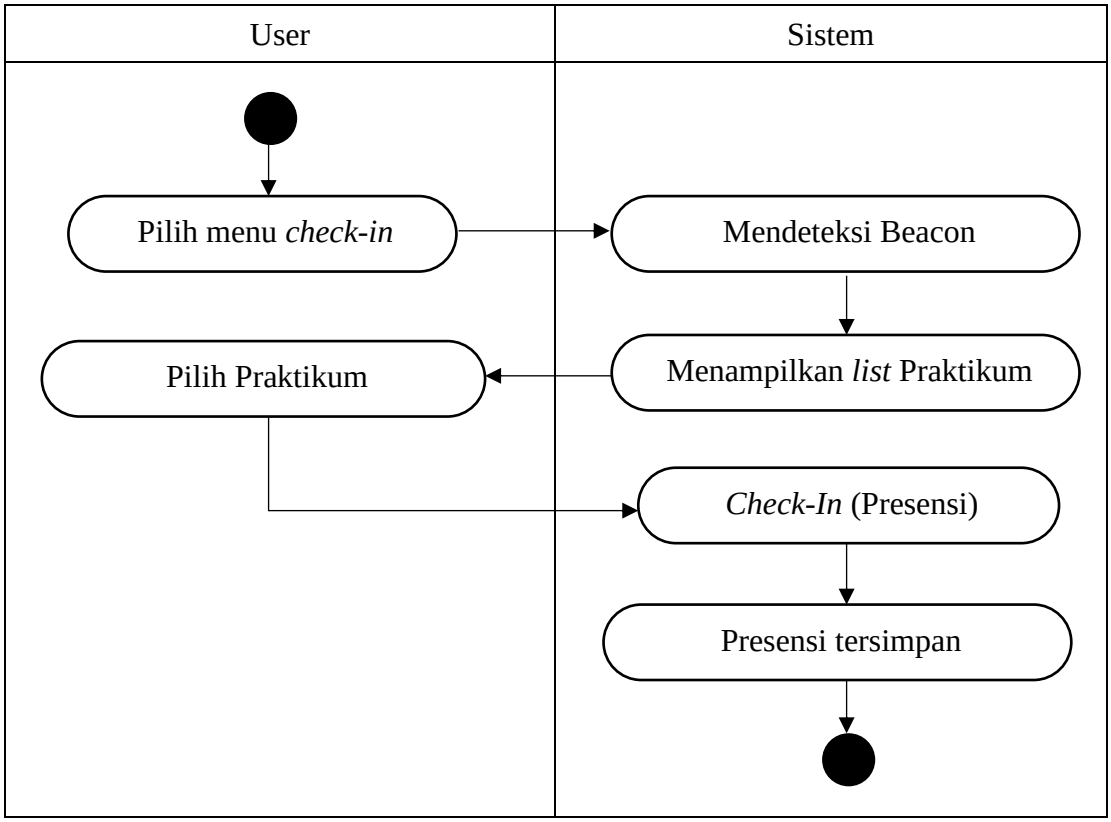
User masuk ke halaman login dan akan diminta untuk mengisi *email* dan *password*. Sistem akan melakukan veirifikasi *email* dan *password* yang telah dimasukkan *user*, jika *email* dan *password* valid maka *user* akan langsung masuk ke halaman utama (*main utama activity*) sedangkan apabila *email* dan *password* salah maka sistem akan menampilkan kembali halaman *login* dan akan ada pemberitahuan "*email* atau *password* salah". Activity diagram dapat dilihat pada gambar 4.3 di bawah ini.



Gambar 4.3 *Activity Diagram Login*

2. *Activity Diagram* Presensi

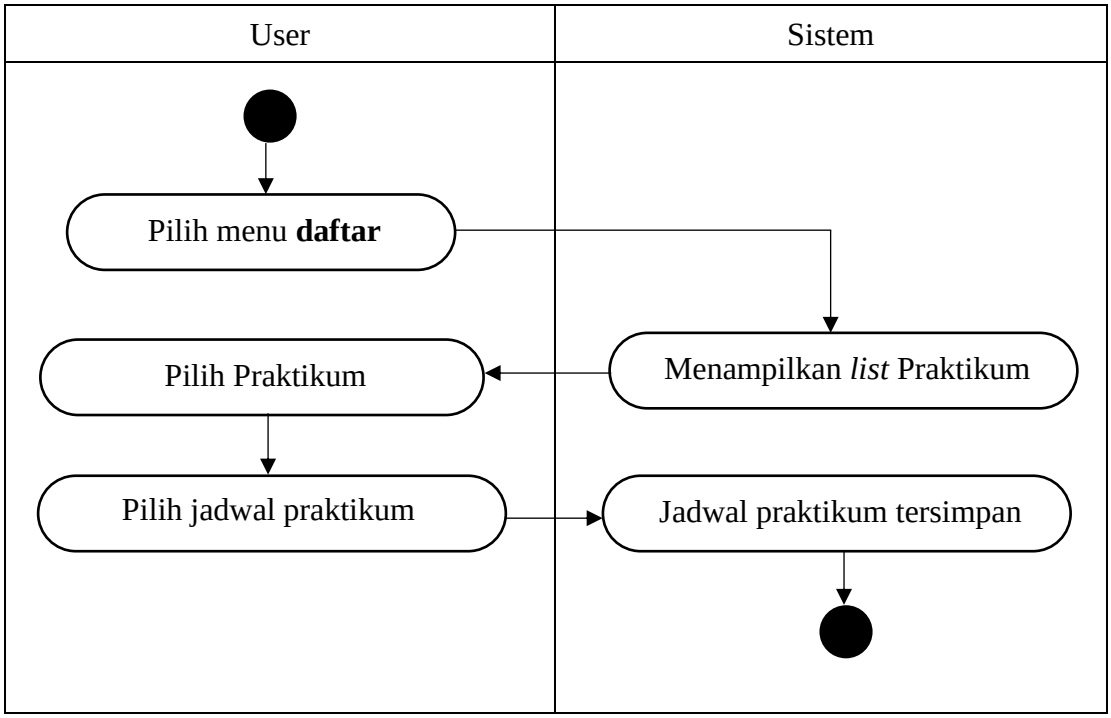
User berada pada halaman utama aplikasi (*main utama activity*). Untuk melakukan *check-in user* memilih atau mengklik menu Presensi. Setelah mengklik *user* akan diantarkan ke halaman *Ranging* (*ranging activity*) yang berisi *list* data-data praktikum beserta ruangan yang telah terdeteksi oleh *device*. *User* diharuskan untuk memilih salah satu praktikum untuk melakukan presensi sesuai dengan jadwal praktikumnya. *Activity diagram* presensi dapat dilihat pada gambar 4.4.



Gambar 4.4 *Activity Diagram Check-In*

3. Activity Daftar

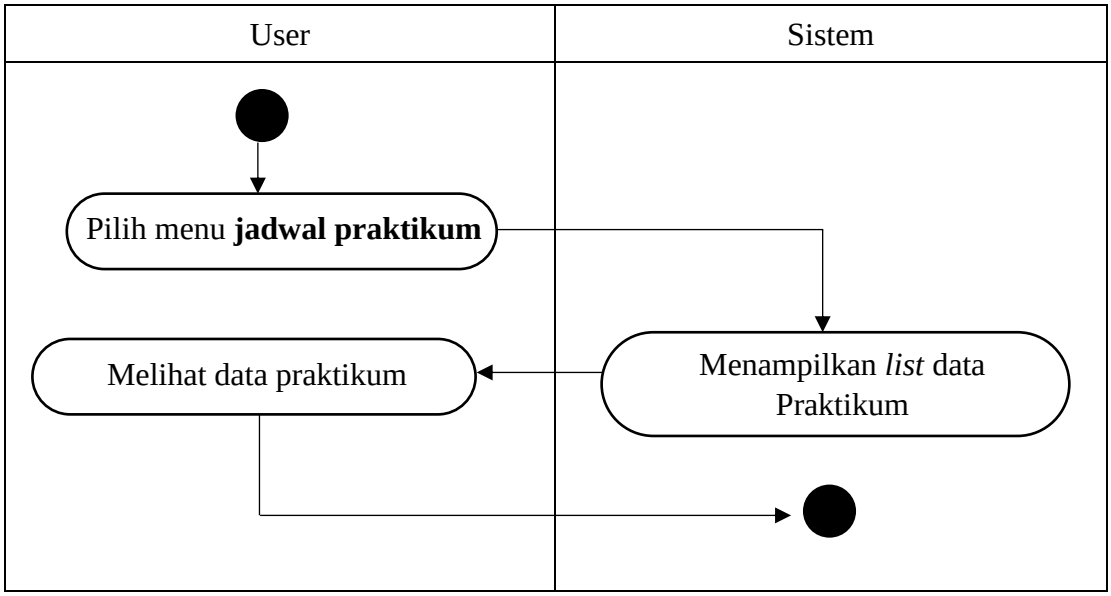
User berada pada halaman utama aplikasi. Untuk melakukan pendaftaran praktikum, user harus mengklik tombol **daftar** yang telah disediakan sehingga user akan langsung diantarkan ke halaman pendaftaran praktikum. Pada halaman ini, user akan memilih praktikum yang akan diambil dengan jadwal yang telah disediakan. *Activity diagram* daftar bisa dilihat di gambar 4.5.



Gambar 4.5 *Activity Diagram* Daftar

4. Activity Jadwal Praktikum

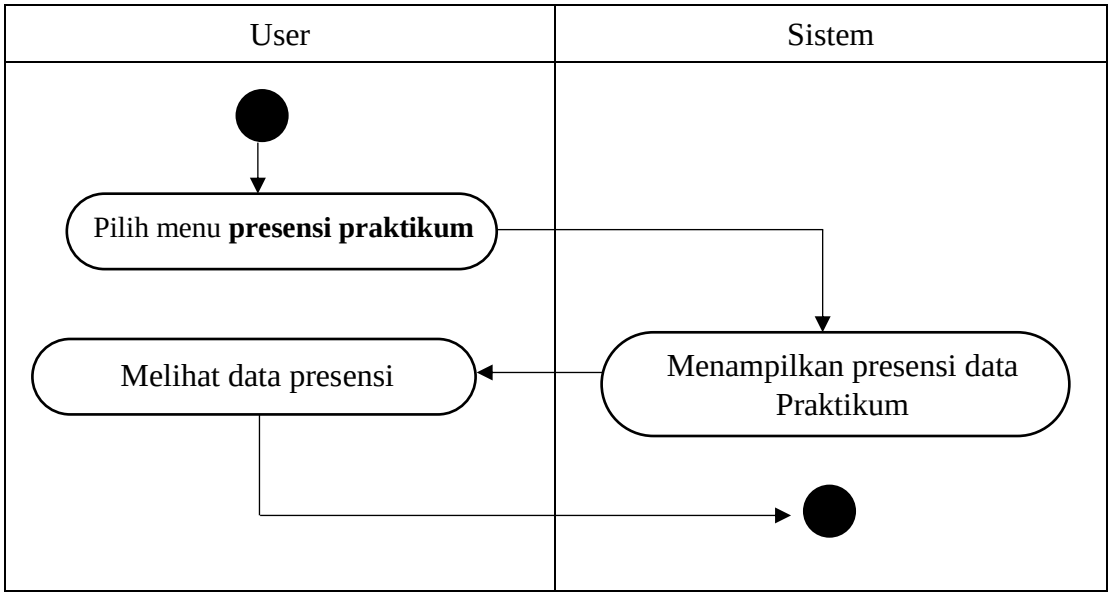
User berada pada halaman utama aplikasi. Untuk melihat jadwal, pada menu utama telah disediakan tombol ‘Jadwal Praktikum’ sehingga saat *user* menekan tombol tersebut, sistem akan langsung mengambil data user berupa praktikum dan jadwal praktikumnya. *Activity diagram* jadwal praktikum bisa dilihat pada gambar 4.6 di bawah ini.



Gambar 4.6 *Activity Diagram* Jadwal Praktikum

5. Activity Presensi Praktikum

User pada halaman utama telah disediakan tombol ‘Presensi Praktikum’. Saat *user* menekan tombol tersebut, sistem akan menampilkan data presensi *user* sesuai dengan praktikum yang ia ambil.



Gambar 4.7 *Activity Diagram* Presensi Praktikum

Data yang akan muncul berupa *email user* dan kehadiharan *user* sesuai dengan tanggal (waktu) pada saat *user* presensi. *Activity diagram* presensi praktikum dapat dilihat pada gambar 4.7.

4.1.2 Rancangan Basis Data

Untuk pengembangan aplikasi iPresence diperlukan sebuah *database* yang digunakan untuk menyimpan seluruh data. Adapun rancangan struktur tabel basis data adalah sebagai berikut:

a. Tabel *Login*

Tabel 4.1 Tabel *login*

Kolom	Tipe Data	Keterangan
Email	String	*Email* pengguna
Password	String	*Password* pengguna

Tabel *login* memberikan informasi tentang *email* dan *password* yang digunakan oleh pengguna agar dapat masuk ke aplikasi ataupun admin untuk masuk ke *database*. Rancangan tabel *login* bisa dilihat pada tabel 4.1.

b. Tabel *User*

Tabel 4.2 Tabel *user*

Kolom	Tipe Data	Keterangan
Firstname	String	Nama depan pengguna
Lastname	String	Nama akhir pengguna
Email	String	*Primary key*
Password	String	*Password* pengguna

Lanjutan **Tabel 4.2** Tabel *user*

Kolom	Tipe Data	Keterangan
Nim	String	NIM pengguna
Praktikum	String	*List* praktikum yang diambil pengguna
Jumlah	String	Jumlah praktikum yang diambil pengguna
Kodejam	String	Waktu praktikum yang diambil pengguna

Tabel *user* merupakan tabel yang berisikan daftar data-data pengguna aplikasi. Adapun rancangan tabel *user* aplikasi iPresence adalah sebagai berikut pada tabel 4.2.

c. Tabel *Beacon*

Tabel 4.3 Tabel *Beacon*

Kolom	Tipe Data	Keterangan
objectId	String	Kode beacon
Nama beacon	String	Nama pengguna
Major	String	Major beacon
Minor	String	Minor beacon
UUID	UUID	UUID beacon

Tabel *beacon* merupakan tabel yang berisi data dari suatu beacon. Tabel beacon ini sudah secara otomatis ada dalam *database* sehingga kita hanya

perlu memasukan data dari *beacon*. Adapun struktur tabel beacon dapat dilihat dari tabel 4.3.

d. Tabel Data Praktikum

Tabel 4.4 Tabel Praktikum

Kolom	Tipe Data	Keterangan
namapraktikum	String	Nama praktikum
kodepraktikum	String	Kode praktikum
Kodejam	String	Kode waktu praktikum
Major	String	Major beacon
Minor	String	Minor beacon
Ruangan	String	Lokasi praktikum
Waktu	Array	Waktu praktikum
Dosen	String	Dosen pengampu praktikum

Tabel praktikum berisi data-data praktikum yang ada. Adapun rancangan tabel data praktikum bisa dilihat pada tabel 4.4.

e. Tabel Presensi

Tabel 4.5 Tabel Kehadiran

Kolom	Tipe Data	Keterangan
Nama	String	Nama praktikan
Nim	String	NIM praktikan

Lanjutan **Tabel 4.5** Tabel Kehadiran

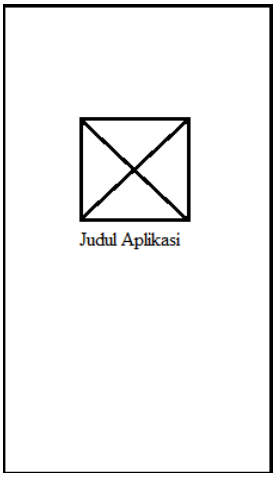
Kolom	Tipe Data	Keterangan
kodepraktikum	String	Kode praktikum
Kodejam	String	Kode jam praktikum yang diambil

Tabel kehadiran berisi data-data kehadiran praktikum mahasiswa. Adapun rancangan tabel kehadiran bisa dilihat pada tabel 4.5.

4.1.3 Rancangan Antarmuka

Perancangan antarmuka merupakan perancangan tampilan suatu aplikasi atau perangkat lunak yang memiliki fungsi sebagai media komunikasi antara pengguna dengan perangkat lunak. Perancangan ini adalah sebuah penggambaran, perencanaan dan pengaturan dari beberapa elemen komponen perangkat lunak. Perancangan antarmuka diharapkan dapat mempermudah pengguna dalam melakukan interaksi dengan sistem perangkat lunak. Berikut merupakan perancangan antarmuka yang ada pada aplikasi:

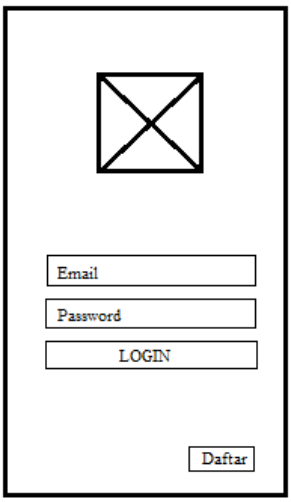
a. Perancangan *Splashscreen*



Gambar 4.8 *Splashscreen*

Perancangan *splashscreen* digunakan untuk tampilan *activity splashscreen* pada saat *user* membuka aplikasi iPresence. Perancangan splashscreen ditunjukan oleh gambar 4.8.

b. Perancangan *Activity Login*

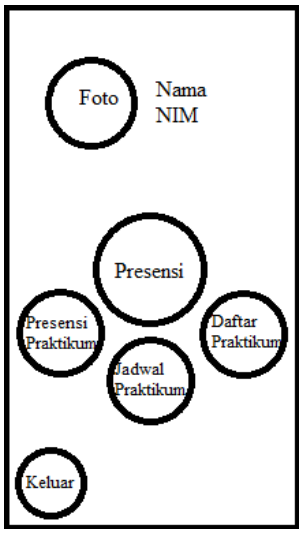


Gambar 4.9 *Login*

Perancangan *activity login* digunakan untuk tampilan *activity login* yang akan muncul setelah *activity splashscreen*. Terdapat 2 *edit text* dan 2 *button*. Perancangan *activity login* dapat dilihat pada gambar 4.9.

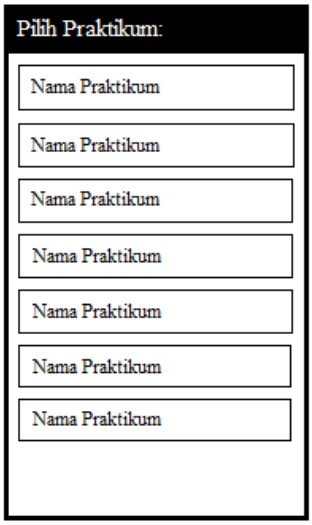
c. Perancangan *Activity Main Utama*

Perancangan *activity main utama* digunakan sebagai halaman utama dari aplikasi iPresence. Profile user akan ditampilkan berupa foto, nama dan NIM. Pada halaman ini terdapat 3 button yaitu **check-in**, **daftar**, dan **logout**. Perancangan *activity main utama* dapat dilihat dari gambar 4.10.



Gambar 4.10 *Main Utama*

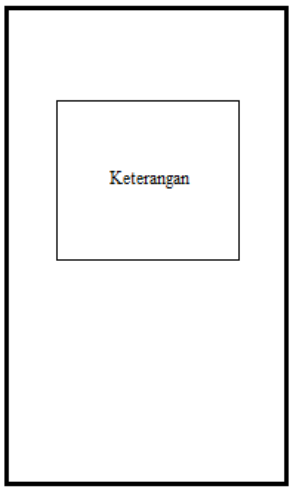
d. Perancangan *Activity Ranging*



Gambar 4.11 *Ranging*

Perancangan *activity ranging* digunakan untuk tampilan *activity ranging* yang akan memunculkan beberapa *button* sesuai dengan beacon yang terdeteksi. Berikut tampilan *activity ranging* pada gambar 4.11.

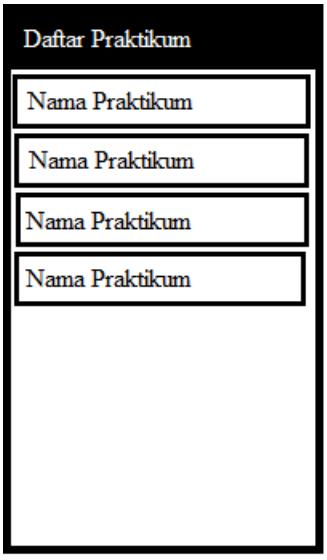
e. Perancangan *Activity Main*



Gambar 4.12 *Main*

Perancangan *activity main* digunakan untuk menampilkan pemberitahuan sukses *check-in* (presensi) dan gagal *check-in* kepada *user*. Berikut tampilan *activity main* seperti gambar 4.12 di atas.

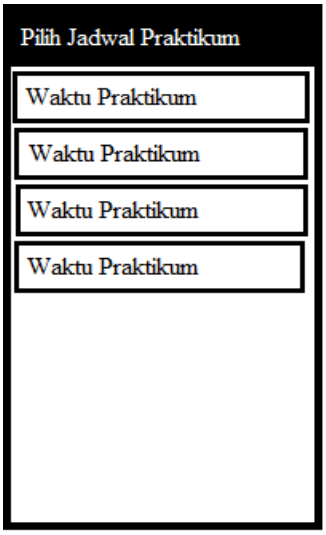
f. Perancangan *Activity* Daftar



Gambar 4.13 Daftar

Perancangan *activity* daftar digunakan untuk menampilkan *activity* daftar yang terdiri dari beberapa *listview* yang dapat diklik. Perancangan *activity* daftar dapat dilihat pada gambar 4.13.

g. Perancangan *Activity* Daftar Praktikum

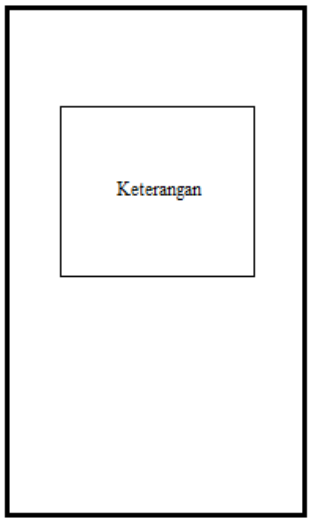


Gambar 4.14 Jadwal Praktikum

Perancangan *activity daftar praktikum* digunakan untuk menampilkan *activity daftar praktikum* yang terdiri dari beberapa *listview* waktu praktikum yang bisa diklik. Perancangan *activity* daftar dapat dilihat pada gambar 4.14.

h. Perancangan *Activity* Simpan Jadwal

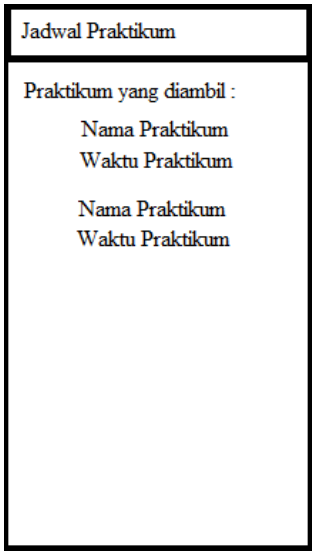
Perancangan *activity simpan jadwal* digunakan untuk menampilkan *dialog box* yang berupa konfirmasi kepada pengguna dengan jadwal yang telah dipilih. Terdapat 1 button dalam halaman ini. Berikut tampilan *activity simpan jadwal* seperti gambar 4.15.



Gambar 4.15 Simpan Jadwal Praktikum

i. Perancangan *Activity* Jadwal Praktikum

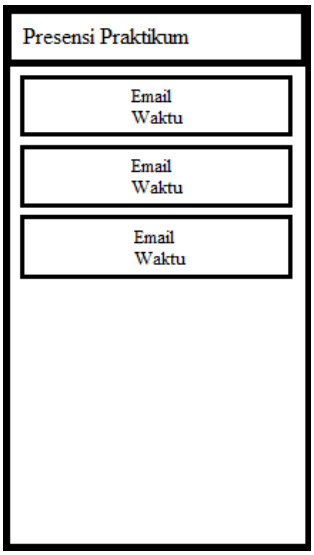
Perancangan *activity* jadwal praktikum digunakan untuk menampilkan halaman jadwal praktikum yang berisi data-data praktikum yang diambil oleh *user*. Nama praktikum dan waktu praktikum yang diambil *user* akan ditampilkan pada halaman ini. Peracangan *activity* jadwal praktikum ditunjukkan oleh gambar 4.16.



Gambar 4.16 Jadwal Praktikum

j. Perancangan *Activity* Presensi Praktikum

Perancnagan activity presensi praktikum digunakan untuk menampilkan halaman presensi praktikum yang berisi data presensi praktikum user sesuai dengan praktikum yang di ambil oleh user. Perancangan activity presensi praktikum adalah sebagai berikut pada gambar 4.17.



Gambar 4.17 Presensi Praktikum

4.2 Pengembangan Sistem

Pengembangan sistem dari aplikasi presensi otomatis "iPresence" dilakukan dengan terlebih dahulu membuat data-data yang diperlukan dalam *database* seperti data praktikum dan data *beacon*. Data-data yang ada pada *database* ini akan digunakan sebagai sumber data dalam pengoperasian aplikasi iPresence. Setelah itu membuat implementasi dari desain *user interface* aplikasi sesuai dengan kebutuhan. Pengembangan dilakukan dengan menggunakan bahasa pemrograman *java* untuk sistem aplikasi dan *xml* untuk *interface* dan media yang digunakan yaitu Android Studio.

4.3 Pembuatan *Coding*

Pembuatan *coding* pada aplikasi iPresence dilakukan sebagai *server side* atau jembatan penghubung antara Android Studio dengan *database* yaitu Mesosfer. Adapaun pembuatan *coding* pada aplikasi presensi otomatis iPresence adalah sebagai berikut:

Script koneksi Mesosfer

```
public class MesosferApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        Mesosfer.setPushNotification(true);

        // initialize Mesosfer SDK
        Mesosfer.initialize(this, "Beii00uy4o", "YSyWUTrJYM0moWVn5uBZaIxK3pSV0iGH");

        // initialize Cubeacon SDK
        Cubeacon.initialize(this);
    }
}
```

Gambar 4.18 *Script* koneksi Mesosfer

Gambar 4.18 menampilkan *coding* koneksi *database* Mesosfer dengan aplikasi yang akan dibuat. *Coding* ini merupakan penghubung antara *cloud database* dengan aplikasi dimana nantinya aplikasi dapat mengakses data-data yang ada pada *database* Mesosfer. Pada halaman ini, kode dari Mesosfer dimasukan agar aplikasi dapat terhubung ke *cloud database.*

Script *login*

```
public void handleLogin(View view) {
    String username = textUsername.getText().toString();
    String password = textPassword.getText().toString();

    if (TextUtils.isEmpty(username)) {
        Toast.makeText(this, "Alamat email belum diisi", Toast.LENGTH_LONG).show();
        textUsername.requestFocus();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        Toast.makeText(this, "Password belum diisi", Toast.LENGTH_LONG).show();
        textPassword.requestFocus();
        return;
    }

    loading.setMessage("Logging in...");
    loading.show();
    MesosferUser.logInAsync(username, password, (user, e) -> {
            loading.dismiss();
            if (e != null) {
                AlertDialog.Builder builder = new AlertDialog.Builder(LoginActivity.this);
                builder.setTitle("Login Failed");
                builder.setMessage(
                        "Maaf, email/password salah"
                );
                dialog = builder.show();
                return;
            }

            Toast.makeText(LoginActivity.this, "User logged in...", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(LoginActivity.this, MainUtamaActivity.class);
            startActivity(intent);
            finish();
    });
}
```

Gambar 4.19 Script *login*

Gambar 4.19 menampilkan *coding* dari halaman *login*. Aplikasi akan memanggil *email* dan *password* yang dimasukkan oleh *user* melalui **logInAsync** dimana aplikasi akan mengecek apakah *email* dan *password* yang dimasukkan oleh *user* terdapat pada *database* (sesuai) atau tidak. Apabila tidak, maka aplikasi akan menampilkan pesan pada *user* bahwa *email* atau *password* yang dimasukkan tidak sesuai dengan menggunakan perintah **AlertDialog.Builder**. Sedangkan apabila *email* dan *password* telah sesuai,

maka *user* akan langsung diantarkan ke halaman berikutnya yaitu halaman *main utama* dengan menggunakan perintah **Intent**.

Script Register

```
public void handleRegister(View view) {
    // get all value from input
    email = textEmail.getText().toString();
    password = textPassword.getText().toString();
    firstname = textFirstname.getText().toString();
    lastname = textLastname.getText().toString();
    nim = textNIM.getText().toString();

    // validating input values
    if (!isInputValid()) {
        // return if there is an invalid input
        return;
    }

    registerUser();
}
```

Gambar 4.20 *Script* Register

Gambar 4.20 menampilkan *script coding* dari halaman *register*. *Script* ini digunakan pada proses registrasi *user* ketika akan membuat akun pada aplikasi presensi otomatis iPresence. Registrasi *user* membutuhkan data *username* yang terdiri dari *firstname* dan *lastname*, NIM, dan *password* yang di *handle* oleh *method* **handleRegister**.

Data-data yang dimasukkan akan divalidasi oleh sistem aplikasi apakah sudah diisi atau belum oleh perintah **isInputValid()**. Apabila belum maka aplikasi akan memberikan pesan pada *user* bagian mana yang belum diisi.

Kemudian setelah *user* mengisi data dan registrasi, aplikasi akan menyimpan semua data *user* yang telah didaftarkan pada *database* Mesosfer dengan menggunakan *method* **registerUser()** yang didalamnya terdapat perintah **registerAsync**.

Script Main Utama

Script main utama berisi kode-kode pada halaman utama aplikasi iPresence. Identitas *user* seperti Nama dan NIM akan ditampilkan pada halaman ini melalui script yang ada pada gambar 4.21 di bawah ini:

```
private void panggil(){
    final MesosferUser user = MesosferUser.getCurrentUser();
    user.fetchAsync((mesosferUser, e) → {
            // hide progress dialog loading
            loading.dismiss();

    });

    String namaawal = user.getFirstName();
    String namaakhir = user.getLastName();
    String nama = namaawal+" "+namaakhir;
    String nim = user.getDataString("nim");

    setContentView(R.layout.activity_main_utama);
    TextView textView = (TextView) findViewById(R.id.text_view_id);
    textView.setText(nama);

    TextView textView3 = (TextView) findViewById(R.id.text_nik);
    textView3.setText(nim);

}
```

Gambar 4.21 *Script* menampilkan identitas pengguna

Pada halaman utama aplikasi, aplikasi akan meng-*handle* beberapa menu aplikasi seperti presensi, daftar praktikum dan keluar dari aplikasi. Untuk menu presensi di-*handle* oleh *method* **handleData** yang terhubung ke *Ranging Activity*. Script ditunjukan oleh gambar 4.22 di bawah ini:

```
public void handleData(View view) {

    MesosferUser user = MesosferUser.getCurrentUser();
    Intent intent = new Intent(this, RangingActivity.class);
    intent.putExtra("PRAKTIKUM", user.getDataString("praktikum"));
    startActivity(intent);
}
```

Gambar 4.22 *Script* menu presensi

Sedangkan untuk menu daftar praktikum di-*handle* oleh **handleDaftar** yang terhubung ke *Daftar Activity*. *Script* ditunjukan pada gambar 4.23 di bawah ini:

```
public void handleDaftar(View view) {
    Intent intent = new Intent(this, DaftarActivity.class);
    startActivity(intent);
}
```

Gambar 4.23 *Script* menu daftar praktikum

Untuk menu keluar dari aplikasi, *script* ditunjukan oleh gambar 4.24 di bawah ini:

```
public void handleLogout(View view) {
    loading.setMessage("Logging out...");
    loading.show();
    MesosferUser.logOutAsync((e) → {
            loading.dismiss();

            if (e != null) {
                // setup alert dialog builder
                AlertDialog.Builder builder = new AlertDialog.Builder(MainUtamaActivity.this);
                builder.setNegativeButton(android.R.string.ok, null);
                builder.setTitle("Log Out Error");
                builder.setMessage(
                        String.format(Locale.getDefault(), "Error code: %d\nDescription: %s",
                                e.getCode(), e.getMessage())
                );
                dialog = builder.show();
            }

            Intent intent = new Intent(MainUtamaActivity.this, LoginActivity.class);
            startActivity(intent);
            finish();
    });
}
```

Gambar 4.24 *Script* menu keluar aplikasi

Script Daftar

Script daftar berisi kode-kode untuk daftar praktikum pada halaman *Daftar Activity* yang di-*handle* oleh *method* **daftar()**. *Method* **daftar()** berfungsi untuk memanggil data praktikum yang ada pada *database* Mesosfer

sehingga dapat ditampilkan pada aplikasi yaitu dengan menggunakan perintah **map.put** seperti pada gambar 4.25 di bawah ini.

```
public void daftar() {

    MesosferQuery<MesosferData> daftar = MesosferData.getQuery("DataPraktikum");

    daftar.findAsync((list, e) -> {
            loading.dismiss();
            DaftarActivity.this.data = list;
            mapDataList.clear();
            for (MesosferData ini : list){

                waktupraktikum = ini.getDataString("praktikum");
                nama = ini.getDataString("namapraktikum");

                Map<String, String> map = new HashMap<>();
                map.put("id", nama);
                map.put("data", "Teknik Elektro UMY");
                mapDataList.add(map);
            }
            runOnUiThread(() -> { adapter.notifyDataSetChanged(); });
    });
}
```

Gambar 4.25 *Script* menampilkan daftar praktikum

User dapat mengetahui praktikum apa saja yang dilaksanakan di labolatorium pada halaman daftar dan melakukan pendaftaran jadwal praktikum. Setelah daftar praktikum muncul, user dapat memilih praktikum sesuai dengan yang ia ambil dengan cara mengklik salah satu praktikum dan aplikasi akan langsung mengantarkan *user* ke halaman selanjutnya yang berisi jadwal praktikum dengan menggunakan perintah **Intent** yang ditunjukkan oleh gambar 4.26 di bawah ini:

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    final MesosferData indah = this.data.get(position);
    Intent intent = new Intent(DaftarActivity.this, DaftarPraktikumActivity.class);
    intent.putExtra("BEACONS", indah.getDataString("praktikum"));
    startActivity(intent);
    recreate();
}
```

Gambar 4.26 *Script* ke halaman jadwal praktikum

Script Jadwal Praktikum

Script jadwal praktikum berisi *script* untuk menampilkan *list* jadwal praktikum yang telah dipilih *user*. Jadwal praktikum ditampilkan oleh aplikasi dengan menggunakan *method* **jadwalpraktikum()** yang didalamnya berisi kode untuk memanggil data jadwal sesuai dengan praktikum yang dipilih oleh *user* dengan menggungakan perintah **MesosferQuery** dan menampilkannya dengan menggunakan perintah **map.put**. *Script* dapat ditunjukan oleh gambar 4.27 di bawah ini:

```
public void jadwalpraktikum(){

    MesosferQuery<MesosferData> DataPraktikum = MesosferData.getQuery(inidia);
    DataPraktikum.findAsync((list, e) → {
            if (e != null) {
                // handle the exception
                return;
            }

            DaftarPraktikumActivity.this.dian = list;
            data.clear();

            for (MesosferData jadwalpraktikum : list ){

                hari = jadwalpraktikum.getDataString("hari");
                jam = jadwalpraktikum.getDataString("jam");
                kodepraktikum = jadwalpraktikum.getDataString("kodepraktikum");

                Map<String, String> map = new HashMap<>();

                map.put("title", hari);
                map.put("subtitle", jam);
                data.add(map);
            }
            runOnUiThread(() → { adapter.notifyDataSetChanged(); });
    });

}
```

Gambar 4.27 *Script* menampilkan jadwal praktikum

User dapat memilih jadwal praktikum sesuai dengan yang ia inginkan. Setelah user memilih jadwal praktikum, maka akan muncul kotak dialog pada user yang berisi konfirmasi jadwal praktikum. Untuk menampilkan kotak

dialog konfirmasi ini menggunakan perintah AlertDialog.Builder(). Script ditunjukan oleh gambar 4.28 di bawah ini:

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
final MesosferData datajadwal = this.listdata.get(position);

    jadwalpraktikum();
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

    // set title dialog
    alertDialogBuilder.setTitle(datajadwal.getDataString("hari")+" "+ datajadwal.getDataString("jam"));

    // set pesan dari dialog
    alertDialogBuilder
            .setMessage("Apakah anda yakin?")
            .setIcon(R.mipmap.logo)
            .setCancelable(false)
            .setPositiveButton("Ya",(dialog, id) -> {
                    Intent intent = new Intent(DaftarPraktikumActivity.this, SimpanJadwalActivity.class);
                    intent.putExtra("KODEPRAKTIKUM", datajadwal.getDataString("kodepraktikum"));
                    intent.putExtra("KODEJAM", datajadwal.getDataString("kodejam"));
                    startActivity(intent);
                    recreate();
            })
            .setNegativeButton("Tidak",(dialog, id) -> {
                    dialog.cancel();
            });

    // membuat alert dialog dari builder
    AlertDialog alertDialog = alertDialogBuilder.create();
    // menampilkan alert dialog
    alertDialog.show();
```

Gambar 4.28 *Script* konfirmasi jadwal praktikum

Pada gambar 4.28 di atas menunjukan, saat *user* mengkonfirmasi ‘Ya’ dengan jadwal yang telah dipilih, maka aplikasi akan menyimpan data praktikum yang dipilih pada data *user* yang ada pada halaman *SimpanJadwalActivity* oleh perintah **Intent**. Sedangkan apabila *user* mengkonfirmasi ‘Tidak’ maka akan kembali ke halaman jadwal praktikum.

Untuk menyimpan data jadwal praktikum yang dipilih *user*, aplikasi menggunakan *method* **simpankode()** yang ada pada halaman *SimpanJadwalActivity* dengan menggunakan perintah **setData()** dan **setAsync()** yang ditunjukan pada gambar 4.29. Saat melakukan penyimpanan berhasil maka akan muncul pemberitahuan pada *user* bahwa pendaftaran

berhasil. Begitu juga saat gagal menyimpan akan muncul pemberitahuan bawah pendaftaran gagal. Aplikasi menggunakan perintah **AlertDialog.Builder** untuk melakukan hal tersebut. *Script* ditunjukan pada gambar 4.30.

```
private void simpankode() {

    MesosferUser user = MesosferUser.getCurrentUser();
    int jumlah = Integer.parseInt(user.getDataString("jumlah"));

    //String[] praktikum =  {user.getDataString("praktikum")};
    //String hana = user.getDataString(itudia);
    String[] waktu = {user.getDataString(itudia)};

    //praktikum[jumlah] = inidia;
    //final String kode = Arrays.toString(praktikum);
    final String kodejam = Arrays.toString(waktu);
    final String kode = inidia.toString();
    //final String kodejam = itudia.toString();

    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

    // set title dialog
    alertDialogBuilder.setTitle("Daftar Praktikum");

    if (jumlah == 0){
        user.setData("praktikum", kode );
        user.setData("kodejam", kodejam);

        alertDialogBuilder
                .setMessage("Daftar Praktikum Berhasil")
                .setCancelable(false)
                .setPositiveButton("OK",new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        Intent intent = new Intent(SimpanJadwalActivity.this, MainUtamaActivity.class);
                        startActivity(intent);
                        finishAffinity();
                    }
                });
        // membuat alert dialog dari builder
        AlertDialog alertDialog = alertDialogBuilder.create();
        // menampilkan alert dialog
        alertDialog.show();

    }
```

Gambar 4.29 *Script set* data dan pemberitahuan

```
    else if (jumlah>=3){

        alertDialogBuilder
                .setMessage("Daftar Praktikum Gagal")
                .setCancelable(false)
                .setPositiveButton("OK",(dialog, id) -> {

                        Intent intent = new Intent(SimpanJadwalActivity.this, MainUtamaActivity.class);
                        startActivity(intent);
                        finishAffinity();
                });
        // membuat alert dialog dari builder
        AlertDialog alertDialog = alertDialogBuilder.create();
        // menampilkan alert dialog
        alertDialog.show();

    }

    user.setData("jumlah", String.valueOf(jumlah+1));
    user.saveAsync((e) -> { //loading.dismiss(); });
}
```

Gambar 4.30 *Script* menyimpan data

Script Ranging Beacon

Script ranging beacon berisi kumpulan kode-kode dalam mendeteksi *beacon* yang ada di sekitar kita. Pada halaman ini, sistem aplikasi akan mendeteksi *beacon* yang ada yang menginterpretasikan satu praktikum. Aplikasi akan mendeteksi *beacon* dengan menggunakan *method* **didRangeBeaconsInRegion**. *Method* ini bekerja dengan cara memanggil tabel data pada *database* yaitu tabel Data Praktikum yang berisi informasi mengenai praktikum. Didalamnya terdapat *filter* dimana saat nilai *major* dan *minor beacon* yang terdeteksi sama nilainya pada tabel Data Praktikum, aplikasi akan memanggil nama *beacon* tersebut dengan perintah **createWithObjectId** sedangkan untuk *filter* menggunakan perintah **whereEqualTo**.

Setelah nama praktikum terpanggil, maka akan ditampilkan dalam aplikasi dengan menggunakan perintah **map.put**. *Script* ditunjukan oleh gambar 4.31 di bawah ini:

```
@Override
public void didRangeBeaconsInRegion(final List<CBBeacon> beacons, CBRegion region) {
    this.beaconsmes = beacons;

    Map<String, String> map;
    String title, subtitle;

    // clear data
    data.clear();

    for (CBBeacon beacon : beacons) {

        //memanggil bucket data praktikum
        final MesosferQuery<MesosferData> DataPraktikumQuery = MesosferData.getQuery("DataPraktikum");

        //persamaan yang diambil yaitu major dan minor
        DataPraktikumQuery.whereEqualTo("major", String.valueOf(beacon.getMajor()));
        DataPraktikumQuery.whereEqualTo("minor", String.valueOf(beacon.getMinor()));

        //memanggil
        DataPraktikumQuery.findAsync((list, e) -> {
                if (e != null) {
                    return;
                }

                RangingActivity.this.listdata = list;
                for(MesosferData satu : list){
                    kodepraktikum = satu.getDataString("kodepraktikum");
                    namapraktikum = satu.getDataString("namapraktikum");
                }
        });

        if (beacon.getMajor() == 1 && beacon.getMinor() == 284) {
            MesosferData DataPraktikum = MesosferData.createWithObjectId("g0frlwcRQt");
            name = DataPraktikum.getDataString("namapraktikum");
            ruang = DataPraktikum.getDataString("ruangan");
        } else if (beacon.getMajor() == 2 && beacon.getMinor() == 285) {
            MesosferData DataPraktikum = MesosferData.createWithObjectId("352ze9NkBN");
            name = DataPraktikum.getDataString("namapraktikum");
            ruang = DataPraktikum.getDataString("ruangan");
        } else if (beacon.getMajor() == 3 && beacon.getMinor() == 286 ) {
            MesosferData DataPraktikum = MesosferData.createWithObjectId("wvfrNAVVNR");
            name = DataPraktikum.getDataString("namapraktikum");
            ruang = DataPraktikum.getDataString("ruangan");
        }

        //menampilkan
        map = new HashMap<>();
        map.put("title", name);
        map.put("subtitle", "Teknik Elektro UMY");
        data.add(map);


    }
    runOnUiThread(() -> {
            adapter.notifyDataSetChanged();
            if (getSupportActionBar() != null) {
                getSupportActionBar().setSubtitle("Praktikum yang terdeteksi: " + beacons.size());
            }
    });
}
```

Gambar 4.31 *Script* mendeteksi beacon

Setelah nama-nama *beacon* (praktikum) muncul, *user* dapat memilih praktikum yang akan ia hadiri dengan cara mengklik salah satu praktikum. Aktivitas ini menggunakan *method* **onItemClick**. *Script* ditunjukan oleh gambar 4.32 di bawah ini.

```
@Override
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    final CBBeacon CBbeacon = this.beaconsmes.get(i);
    final MesosferData listdatapraktikum = this.listdata.get(i);

    //memanggil mesosfer data
    MesosferQuery<MesosferData> ambil = MesosferData.getQuery("DataPraktikum");
    ambil.whereEqualTo("major", String.valueOf(CBbeacon.getMajor()));
    ambil.findAsync((listbeacon, e) -> {

            // check if there is an exception happen
            if (e != null) {
                Log.e("SPLASH", "Error when downloading beacon: " + e);
                return;
            }

            // get the result
            if (listbeacon != null && !listbeacon.isEmpty()) {
                // populate the result into list of Beacon

                ArrayList<Beacon> beacons = new ArrayList<>();

                for (MesosferData cloudBeacon : listbeacon) {

                    Beacon beacon = new Beacon();
                    beacon.setProximityUUID(CBbeacon.getProximityUUID());
                    beacon.setMajor(CBbeacon.getMajor());
                    beacon.setMinor(CBbeacon.getMinor());
                    beacon.setIdentifier(cloudBeacon.getObjectId());

                    beacons.add(beacon);

                    Intent intent = new Intent(RangingActivity.this, RangingPraktikumActivity.class);
                    intent.putParcelableArrayListExtra("BEACONS", beacons);
                    intent.putExtra("kodesemua", listdatapraktikum.getDataString("kodepraktikum") +
                            listdatapraktikum.getDataString("namapraktikum"));
                    intent.putExtra("namapraktikum", listdatapraktikum.getDataString("namapraktikum"));
                    intent.putExtra("kode", listdatapraktikum.getDataString("kodepraktikum"));
                    startActivity(intent);
                    recreate();
                    //finishAffinity();
                }
            }
    });
}
```

Gambar 4.32 *Script* meilih praktikum

Script Ranging Praktikum

Script ranging praktikum berfungsi sebagai *filter* aplikasi. Pada saat *user* memilih praktikum untuk presensi, aplikasi akan mengecek ke tabel *user*

apakah *user* tersebut mengambil praktikum tersebut atau tidak dengan menggunakan *method* **pilihpraktikum()**. Apabila *user* mengambil data tersebut, *user* akan langsung melakukan presensi di *MainActivity* dengan perintah **Intent** *script* ditunjukan oleh gambar 4.33. Sedangkan apabila *user* tidak mengambil praktikum, aplikasi akan mucul pemberitahuan bahwa *user* tidak mengambil praktikum tersebut dengan perintah **AlertDialog.Builder** yang ditunjukan pada gambar 4.34.

```
public void pilihpraktikum(Beacon beaconn) {

    MesosferUser user = MesosferUser.getCurrentUser();

    //kode = user.getDataString(itu);
    user.fetchAsync((mesosferUser, e) -> { loading.dismiss(); });

    p1 = user.getDataString("praktikum");
    p2 = user.getDataString("praktikum2");
    p3 = user.getDataString("praktikum3");
    ambil = kode.toString();
    ambilbeacon = beacon.toString();


    if (p1.equals(ambil)){

        Intent intent = new Intent(RangingPraktikumActivity.this, MainActivity.class);
        intent.putParcelableArrayListExtra("BEACONS", beacon);
        intent.putExtra("namapraktikum", namapraktikum);
        intent.putExtra("kodepraktikum", kode);
        intent.putExtra("kodesemua", kodesemua);
        startActivity(intent);
        recreate();
    }
```

Gambar 4.33 *Script* filter praktikum

```
    else {

        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setTitle("Gagal");
        alertDialogBuilder
                .setMessage("Anda tidak mengambil praktikum ini!")
                .setCancelable(false)
                .setPositiveButton("OK",(dialog, id) -> {

                    Intent intent = new Intent(RangingPraktikumActivity.this, MainUtamaActivity.class);
                    startActivity(intent);
                        finishAffinity();
                })
                .show();
        return;
    }
}
```

Gambar 4.34 *Script* pemberitahuan gagal presensi

Script Main Activity

Script main activity merupakan *script* aktivitas presensi dengan menggunakan *method* **didEnterRegion**. Saat *beacon* terdeteksi oleh aplikasi, maka sistem akan langsung menjalankan *method* **searchForStoryline** untuk menampilkan pemberitahuan pada *user* saat *user* memasuki area *beacon*. *Script* ditunjukan oleh gambar 4.35 dan gambar 4.36 di bawah ini:

```
@Override
public void didEnterRegion(CBRegion region) {
    Log.d("MAIN", "Entering region: " + region);
    searchForStoryline(region, MesosferBeacon.Event.ENTER);
    sendLog(region, MesosferBeacon.Event.ENTER);
    updateView(region);
}
```

Gambar 4.35 *Script* mendeteksi beacon

```
private void searchForStoryline(CBRegion region, MesosferBeacon.Event event) {
    MesosferStorylineDetail
            .getQuery()
            .whereEqualTo("beacons", region.getIdentifier())
            .whereEqualTo("event", event.name())
            .setLimit(1)
            .findAsync((list, e) -> {
                if (e != null) {
                    Log.e("MAIN", "Error happen when finding storyline: " + e);
                    return;
                }

                if (list != null && !list.isEmpty()) {
                    MesosferStorylineDetail detail = list.get(0);
                    displayStoryline(detail);
                }
            });
}
```

Gambar 4.36 *Script searchForStoryline*

Method **sendLog** untuk merekam aktivitas *user* saat memasukin area *beacon* ataupun diluar area *beacon*. *Script* ditunjukan oleh gambar 4.37 di bawah ini:

```
private void sendLog(CBRegion region, final MesosferBeacon.Event event) {
    String beaconId = region.getIdentifier();
    MesosferBeacon beacon = MesosferBeacon.createWithObjectId(beaconId);

    MesosferLog.createLog()
            .setEvent(event)
            .setBeacon(beacon)
            .setModule(MesosferBeacon.Module.PRESENCE)
            .sendAsync((e) -> {
                if (e != null) {
                    Log.e("MAIN", "Failed to send log: " + e);
                    return;
                }

                String state = event == MesosferBeacon.Event.ENTER ? "Check in" : "Check out";
                Toast.makeText(MainActivity.this, state + " sent.", Toast.LENGTH_SHORT).show();
            });
}
```

Gambar 4.37 *Script sendLog*

Lalu *method* **updateView(region)** untuk merekam *user* yang melakukan presensi. *User* yang melakukan presensi akan terekam aktivitasnya

dan akan masuk ke tabel presensi yang nantinya akan menjadi tabel kehadiran *user*. *Script* ditunjukan oleh gambar 4.38 di bawah ini:

```
private void updateView (final CBRegion region){
    final MesosferUser user = MesosferUser.getCurrentUser();
    user.fetchAsync((mesosferUser, e) → {
            // hide progress dialog loading
            loading.dismiss();
    });

    final MesosferQuery<MesosferData> data = MesosferData.getQuery(datap);
    data.findAsync(new FindCallback<MesosferData>() {
        @Override
        public void done(List<MesosferData> list, MesosferException e) {
            if (e != null) {
                // handle the exception
                return;
            }

            for (MesosferData datapresensi : list){
                namapraktikum = datapresensi.getDataString("namapraktikum");
                kodeprk = datapresensi.getDataString("kodepraktikum");
                //kodejam = datapresensi.getDataString("kodejam");
                presensi = datapresensi.getDataString("presensi");
                //kodehari = datapresensi.getDataString("kodehari");
            }

            //datapengguna
            namaawal = user.getFirstName();
            namaakhir = user.getLastName();
            nama = namaawal + " " + namaakhir;
            email = user.getEmail();
            nim = user.getDataString("nim");
            objek = user.getObjectId();
            String handuk = jam.toString();

            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
            Date date = new Date();
            String tanggal = dateFormat.format(date);

            MesosferData PenggunaKePresensi = MesosferData.createData(presensi);
            PenggunaKePresensi.setData("nama", nama);
            PenggunaKePresensi.setData("nim", nim);
            PenggunaKePresensi.setData("kodepraktikum", kodeprk);
            PenggunaKePresensi.setData("email", email);
            PenggunaKePresensi.setData("namapraktikum", namapraktikum);
            PenggunaKePresensi.setData("kodejam", handuk);
            PenggunaKePresensi.setData("tanggal", tanggal);
            PenggunaKePresensi.setData("objek", objek);
            PenggunaKePresensi.saveAsync((e) → { loading.dismiss(); });
        }
    });
}
```

Gambar 4.38 *Script updateView(region)*

Script Jadwal Praktikum

Script jadwal praktikum merupakan *script* aktivitas jadwal praktikum. *Script* ini berisi kode untuk memanggil atau mengambil data praktikum dan kode waktunya yang ada pada tabel *user*. Untuk memanggil data *user*, digunakan perintah **.getCurrentUser** lalu perintah **getDataString** untuk mengambil data pada kolom praktikum dan kodejam. *Script* ditunjukan oleh gambar 4.39.

```
MesosferUser user = MesosferUser.getCurrentUser();
user.fetchAsync(new GetCallback<MesosferUser>() {
    @Override
    public void done(MesosferUser mesosferUser, MesosferException e) {

    }
});

String p1 = user.getDataString("praktikum");
String p2 = user.getDataString("praktikum2");
String p3 = user.getDataString("praktikum3");
String k1 = user.getDataString("kodejam");
String k2 = user.getDataString("kodejam2");
String k3 = user.getDataString("kodejam3");
String jumlah = user.getDataString("jumlah");

setContentView(R.layout.activity_praktikumku);
TextView textView10 = (TextView) findViewById(R.id.text_praktikum2);
textView10.setText(jumlah);

TextView textView = (TextView) findViewById(R.id.text_praktikummm);
textView.setText(p1);

TextView textView2 = (TextView) findViewById(R.id.text_waktu);
textView2.setText(k1);

TextView textView3 = (TextView) findViewById(R.id.text_praktikummm2);
textView3.setText(p2);

TextView textView4 = (TextView) findViewById(R.id.text_waktu2);
textView4.setText(k2);

TextView textView5 = (TextView) findViewById(R.id.text_praktikummm3);
textView5.setText(p3);

TextView textView6 = (TextView) findViewById(R.id.text_waktu3);
textView6.setText(k3);
}
```

Gambar 4.39 *Script* Jadwal Praktikum

Script Presensi Praktikum

Script presensi praktikum berisi script untuk menjalankan aktivitas presensi praktikum. Pada aktivitas ini, akan ditampilkan data presensi praktikum user sesuai dengan praktikum yang diambil dengan menggunakan method datapr(). Pada method tersebut, sistem akan mengambil data user dari data presensi yang berupa email dan waktu presensi user. Apabila user tidak mengambil praktikum tersebut, maka data presensi tidak akan muncul. Script presensi praktikum dapat dilihat pada gambar 4.40.

```
private void datapr() {
    MesosferQuery<MesosferData> praktikum = MesosferData.getQuery(inidia);
    praktikum.findAsync((list, e) -> {
            if (e != null) {
                // handle the exception
                return;
            }

            for (MesosferData pre : list){
                String p = pre.getDataString("presensi");

                final MesosferUser user = MesosferUser.getCurrentUser();
                user.fetchAsync(new GetCallback<MesosferUser>() {
                    @Override
                    public void done(MesosferUser mesosferUser, MesosferException e) {

                    }
                });

                MesosferQuery<MesosferData> datapre = MesosferData.getQuery(p);
                datapre.whereEqualTo("email", user.getEmail());
                datapre.findAsync((list, e) -> {

                        loading.dismiss();
                        if (e != null) {
                            // handle the exception
                            return;
                        }
                     data.clear();
                     for (MesosferData obj : list){

                         String[] a = {obj.getDataString("tanggal")};
                         String kode = Arrays.toString(a);
                         //Toast.makeText(Presensi.this, kode, Toast.LENGTH_SHORT).show();
                         Map<String, String> map = new HashMap<>();
                         map.put("title", "email : "+ user.getEmail() );
                         map.put("subtitle", kode);
                         data.add(map);
                         //Toast.makeText(Presensi.this, em, Toast.LENGTH_SHORT).show();
                            //setContentView(R.layout.activity_praktikumku);
                         }
                         adapter.notifyDataSetChanged();
                    });

            }
    });
}
```

Gambar 4.40 *Script* Presensi Praktikum

4.5 Implementasi pada Android Studio

Coding implementasi pada aplikasi presensi otomatis ‘iPresence’ terbagi menjadi dua yaitu antarmuka (*interface*) dan fungsi sistem. Implementasi antarmuka dilakukan pada folder *layout*, *drawable*, dan *styles*. Sedangkan implementasi *coding* fungsi dilakukan pada folder *Java*. Adapun hasil implementasi yang telah dilakukan sebagai berikut:

Activity Splashscreen

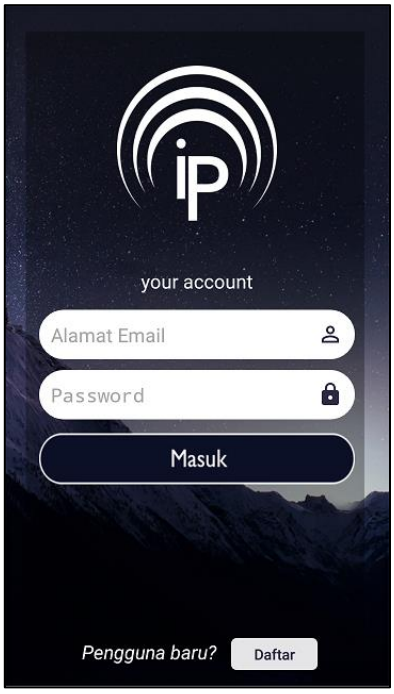
Gambar 4.41 di bawah menampilkan tampilan saat sistem sedang mengakses atau memuat aktivitas lain.

Gambar 4.41 Tampilan *Splashscreen*

Activity Login

Gambar 4.42 di bawah menampilkan tampilan *activity* login. Pada *activity* login *user* harus mengisi *email* dan *password* yang telah terdaftar untuk masuk ke dalam aplikasi.

Pastikan *email* dan *password user* telah terdaftar sehingga dapat masuk dan mengakses menu yang ada pada aplikasi.

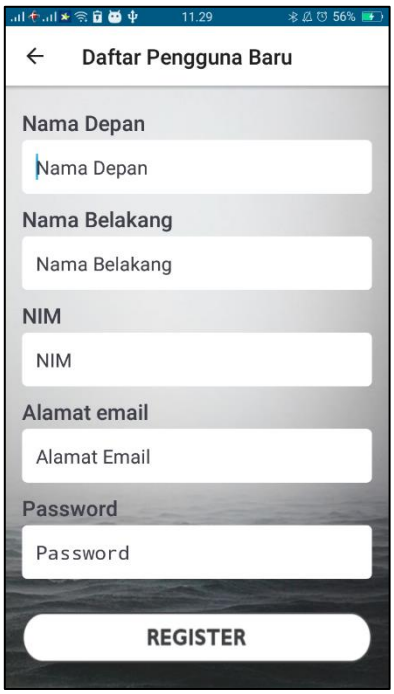


Gambar 4.42 Tampilan *Login*

Activity Register

Gambar 4.43 di bawah menampilkan tampilan dari *activity* registrasi. *User* harus mengisi kolom data yang diperlukan oleh aplikasi untuk membuat akun aplikasi ‘iPresence’ ini seperti nama awal, nama akhir, *email*, *password*, dan NIM.

Apabila ada kolom yang belum terisi, aplikasi akan otomatis menampilkan notifikasi validasi pada user kolom mana yang belum terisi.



Gambar 4.43 Tampilan Register dan Notifikasi

Activity Main Utama

Gambar 4.44 di bawah ini menampilkan tampilan dari *activity main* utama. Ada dua menu utama dari aplikasi ini yaitu **Presence** dan **Daftar**. *User* akan diarahkan ke *activity* yang dituju ketika menekan salah satu tombol menu di *activity* main utama.

Gambar 4.44 Tampilan *Main* Utama

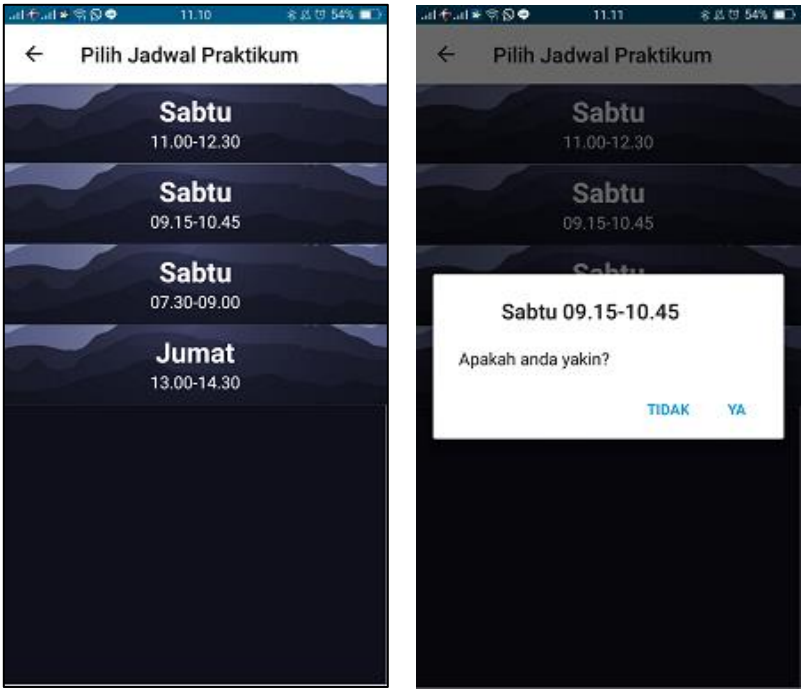
Activity Daftar

Gambar 4.45 di bawah ini menampilkan tampilan dari *activity* daftar dimana pada *activity* ini akan menampilkan *list* praktikum yang dilaksanakan selama satu semester di labolatorium. *User* akan diarahkan ke jadwal praktikum saat menekan salah satu praktikum pada *list* yang ditampilkan.

Gambar 4.45 Tampilan *List Praktikum*

Activity Daftar Praktikum

Gambar 4.46 di bawah ini menampilkan tampilan dari *activity* daftar praktikum yang berisi jadwal dari suatu praktikum. Saat *user* menekan salah satu jadwal yang ditampilkan, akan muncul notifikasi pada *user* untuk mengkonfirmasi jadwal yang telah dipilih seperti pada gambar 4.46.

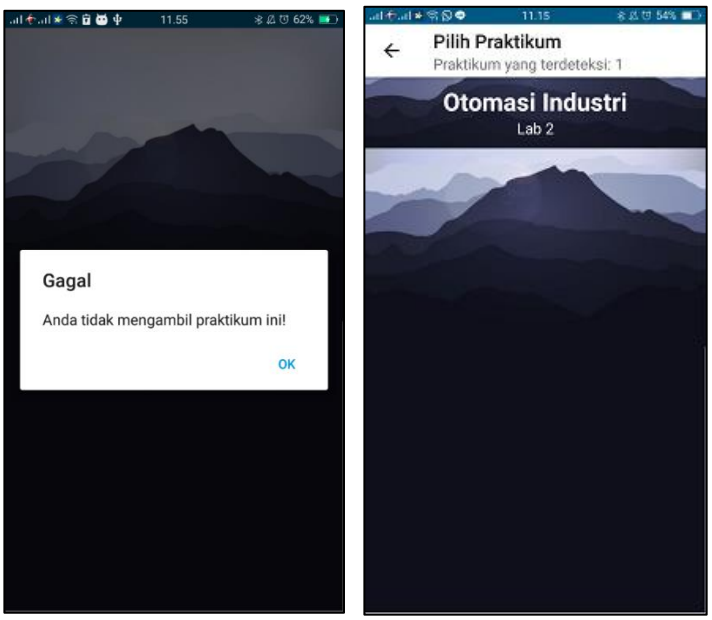


Gambar 4.46 Tampilan *Pilih Jadwal* dan Konfirmasi

Activity Ranging

Gambar 4.47 menampilkan tampilan dari *activity ranging*. Pada tampilan ini akan ada *list* dari beacon yang terdeteksi oleh aplikasi. setiap satu beacon menginterpretasikan satu praktikum.

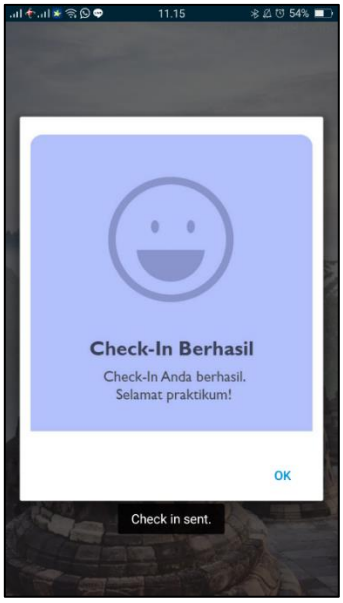
Saat *user* menekan salah satu praktikum, sistem akan melakukan pengecekan ke database *user* untuk memastikan apakah *user* mengambil praktikum yang dipilih atau tidak. Jika ada, maka *user* akan diarahkan ke *activity main*. Tapi jika tidak maka aplikasi akan menampilkan notifikasi kepada *user*.



Gambar 4.47 Tampilan Notifikasi dan *List* Praktikum

Activity Main

Gambar 4.48 di bawah menampilkan tampilan saat berhasil melakukan *check-in* atau presensi pada *activity main*.



Gambar 4.48 Tampilan berhasil *check-in*

4.6 Pengujian Sistem

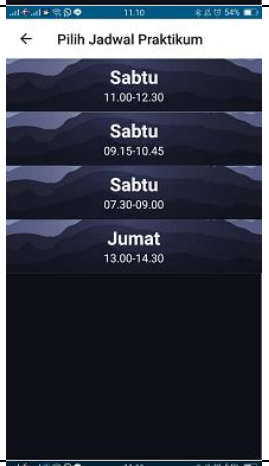
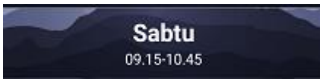
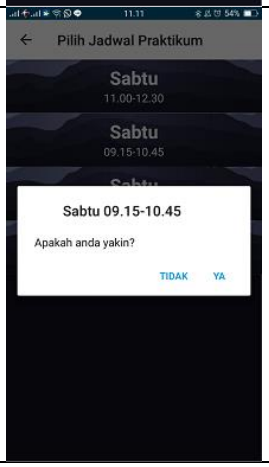
Pengujian sistem dilakukan untuk mengevaluasi apakah seluruh fungsi yang ada pada aplikasi sudah sesuai dengan rancangan.

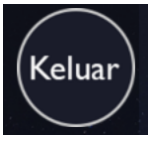
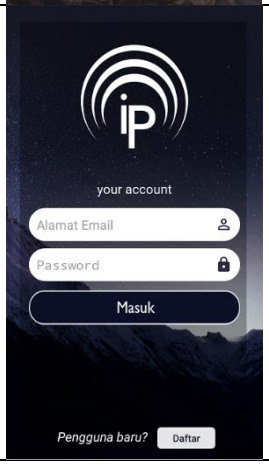
Pengujian *User Interface*

Pengujian user interface bertujuan untuk mengetahui fungsionalitas setiap elemen interface yang terdapat dalam sistem aplikasi. Elemen yang diujian yaitu elemen button pada setiap activity aplikasi. Hasil pengujian dapat dilihat pada tabel berikut.

Tabel 4.6 Tabel Pengujian *User Interface*

No.	Kasus yang diuji	Test Case	Hasil yang diharapkan	Hasil yang di dapat	Status
1.	Tombol login pada activity login		Sistem menampilkan activity main utama		Berhasil
2.	Tombol registrasi pada activity registrasi		Sistem menampilkan notifikasi berhasil melakukan registrasi		Berhasil

3.	Tombol daftar pada activity daftar		Sistem menampilkan list praktikum		Berhasil
4.	Tombol list praktikum pada activity daftar		Sistem menampilkan list jadwal praktikum		Berhasil
5.	Tombol list jadwal praktikum pada activity daftar		Sistem menampilkan dialog box berupa konfirmasi jadwal kepada user		Berhasil

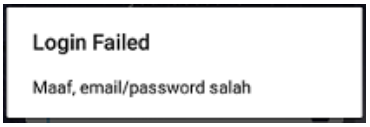
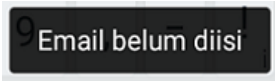
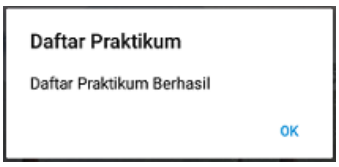
6.	Tombol presensi pada ranging activity		Sistem menampilkan praktikum (beacon) yang terdeteksi		Berhasil
7.	Tombol list praktikum pada ranging activity		Sistem berhasil melakukan presensi pada activity main.		Berhasil
8.	Tombol logout pada activity main utama		Sistem keluar dari activity main utama.		Berhasil

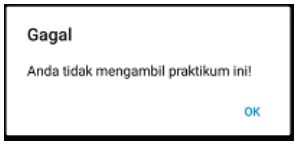
Tabel 4.6 di atas berisi data-data hasil pengujian interface pada aplikasi presensi otomatis iPresence.

Pengujian Validasi

Pengujian validasi bertujuan untuk mengetahui apakah validasi-validasi yang ada pada sistem sudah berfungsi dengan baik. Hasil pengujian validasi dapat dilihat pada tabel 4.7.

Tabel 4.7 Tabel Pengujian Validasi

No.	Test Case	Hasil yang diharapkan	Hasil yang didapat	Status
1.	Validasi email dan password terdaftar pada activity login	Sistem akan menampilkan login berhasil		Berhasil
2.	Validasi email dan password tidak terdaftar pada activity login	Sistem akan menmapilkan notifikasi bahwa email atau password yang dimasukan salah.		Berhasil
3.	Validasi kolom data yang belum diisi pada activity registrasi	Sistem akan menampilkan notifikasi bahwa ada kolom data yang belum diisi		Berhasil
4.	Validasi berhasil registrasi pada registrasi activity	Sistem akan menampilkan notifikasi berhasil registrasi		Berhasil
5.	Validasi berhasil menyimpan jadwal praktikum pada activity daftar.	Sistem akan menampilkan notifikasi berhasil menyimpan jadwal praktikum.		Berhasil

6.	Validasi berhasil presensi atau check-in pada activity main.	Sistem menampilkan notifikasi berhasil melakukan check-in.		Berhasil
7.	Validasi gagal presensi atau check-in pada activity main.	Sistem menampilkan notifikasi gagal melakukan presensi.		Berhasil

Tabel 4.7 di atas merupakan data hasil pengujian aplikasi presensi otomatis iPresence.

Hasil Kuisioner

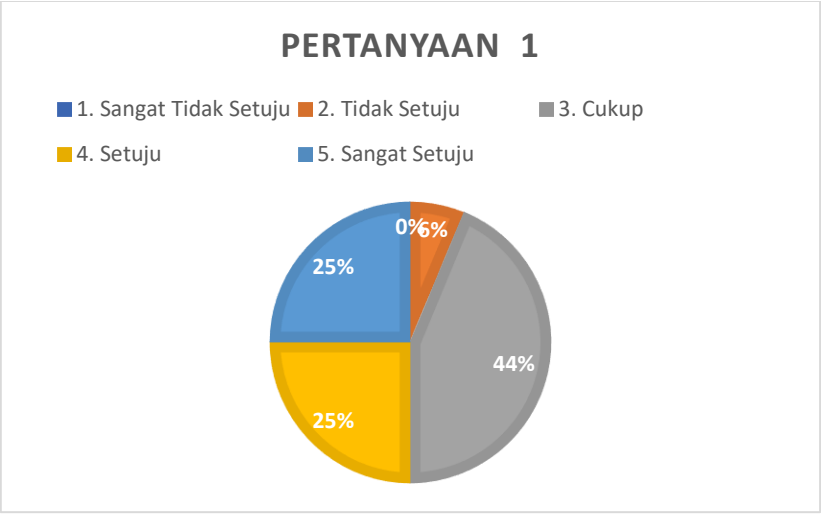
Hasil kuisioner merupakan data yang didapat saat pengujian aplikasi yang dilakukan pada praktikum Sistem Kontrol dan Instrumentasi pada hari Rabu, 4 April 2018. Dari 16 praktikan (responden) yang dibagi kuesioner, hanya 15 orang yang mengembalikan jawaban lengkap, dan 1 orang anonim dengan jawaban lengkap. Berikut hasil kuisioner pada tabel 4.8.

Tabel 4.8 Tabel Hasil Kuisioner

Hasil Kuisioner						Jumlah Peserta
Pertanyaan ke-	Jawaban					
	1	2	3	4	5	
Apakah tampilan aplikasi menarik?	0	1	7	4	4	16
Apakah tampilan menu dalam aplikasi mudah dikenali?	0	1	2	6	7	16
Apakah infromasi yang disediakan oleh aplikasi mudah dimengerti?	0	1	2	8	5	16
Apakah penggunaan menu atau fitur pada aplikasi mudah digunakan?	0	1	0	7	8	16
Apakah aplikasi nyaman digunakan?	0	1	0	7	8	16
Apakah aplikasi mudah dipahami?	0	0	4	7	5	16
Apakah aplikasi mudah dioperasikan?	0	0	1	7	8	16
Apakah aplikasi sudah memenuhi kebutuhan di labolatorium Teknik Elektro UMY?	0	0	9	1	6	16
Apakah aplikasi bermanfaat bagi pengguna?	0	0	0	10	6	16
Apakah aplikasi mempunyai kemampuan dan fungsi yang diharapkan?	0	1	6	4	5	16
Jumlah	0	6	31	61	62	

Berdasarkan kuisioner tersebut didapatkan hasil sebagai berikut:

1. Pertanyaan 1 : Apakah tampilan aplikasi menarik?

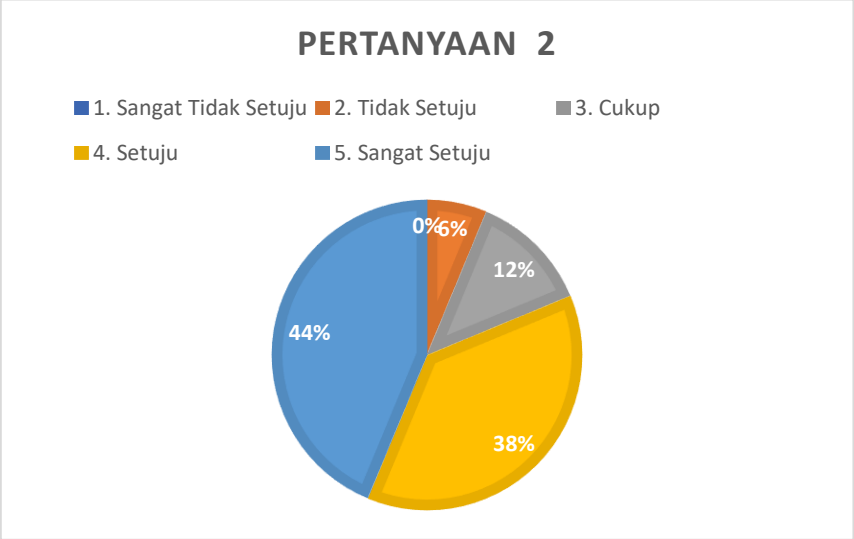


Gambar 4.49 Diagram lingkaran pertanyaan 1

Berdasarkan diagram lingkaran di atas, rata-rata responden setuju dengan tampilan aplikasi yang sudah menarik yaitu sekitar 25% atau 4 orang menjawab ‘setuju’, begitu juga dengan jawaban ‘sangat setuju’, 44% atau 7 orang responden menjawab cukup, dan 6% atau 1 orang responden menjawab tidak setuju dengan pertanyaan 1.

Dari data di atas dapat disimpulkan bahwa tampilan aplikasi sudah cukup menarik untuk pengguna.

2. Pertanyaan 2 : Apakah tampilan menu dalam aplikasi mudah dikenali?

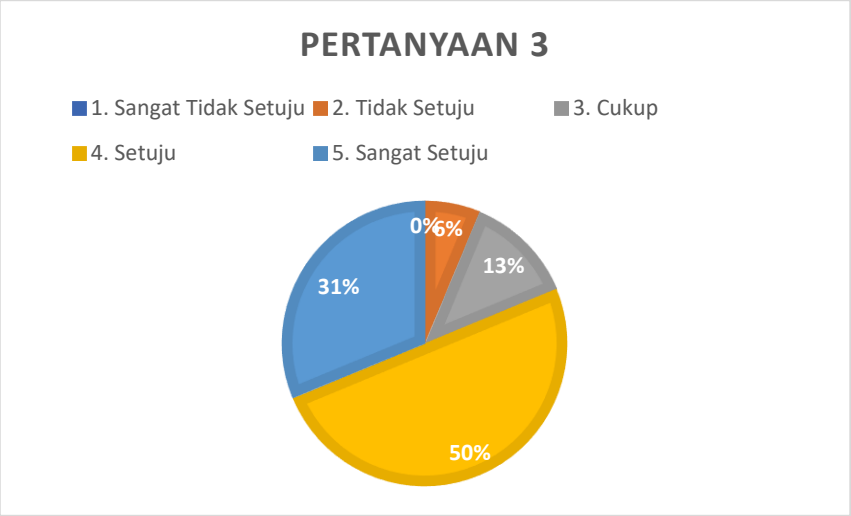


Gambar 4.50 Diagram lingkaran pertanyaan 2

Hasil kuisioner untuk nomor 2, 44% atau 7 orang responden sangat setuju, begitu juga untuk jawaban ‘setuju’ didapat 38% atau 6 orang responden, 12% responden atau 2 orang menjawab ‘cukup’ dan 6% atau 1 orang responden menjawab tidak setuju.

Berdasarkan data di atas, hampir semua responden mudah mengenali menu-menu atau fitur yang ada pada aplikasi presensi otomatis iPresence.

3. Pertanyaan 3 : Apakah infromasi yang disediakan oleh aplikasi mudah dimengerti?

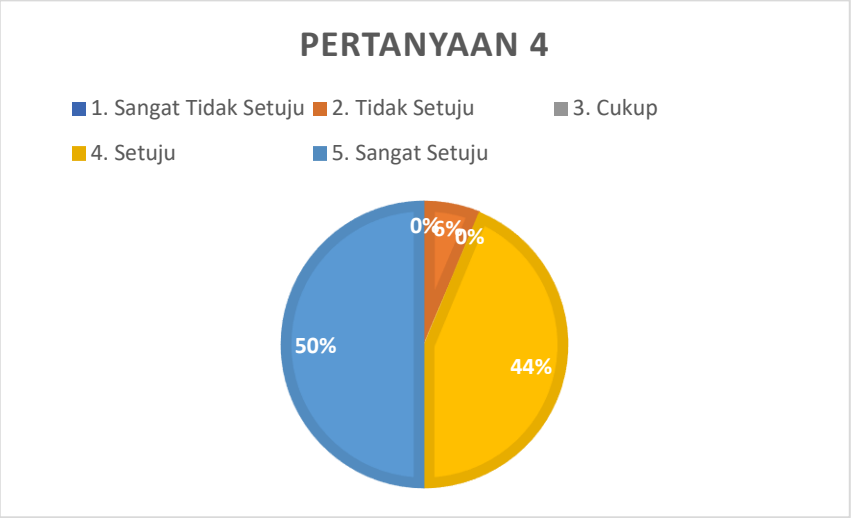


Gambar 4.51 Diagram lingkaran pertanyaan 3

Hasil kuisioner untuk pertanyaan 3, 31% atau 5 orang responden menjawab ‘sangat setuju’ begitu juga dengan jawaban ‘setuju’ didapat responden sebesar 50% atau sebanyak 8 orang. 13% atau 2 orang responden menjawab cukup, dan 6% atau 1 orang responden menjawab ‘tidak setuju’.

Berdasarkan data di atas dapat disimpulkan bahwa sebagian besar responden mudah mengerti dengan informasi yang disediakan oleh aplikasi presensi otomatis iPresence.

4. Pertanyaan 4 : Apakah penggunaan menu atau fitur pada aplikasi mudah digunakan?

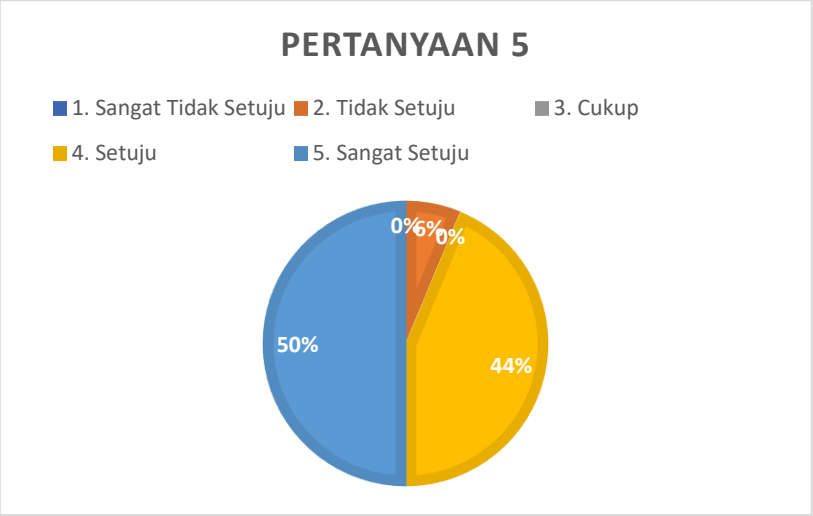


Gambar 4.52 Diagram lingkaran pertanyaan 4

Hasil kuisioner untuk pertanyaan 4 didapatkan 50% responden atau 8 orang responden menjawab ‘sangat setuju’ begitu juga dengan jawaban ‘setuju’ didapat hasil sebesar 44% atau 7 orang responden dan 6% responden atau 1 orang menjawab ‘tidak setuju’ dengan pertanyaan 4.

Berdasarkan dari data di atas dapat disimpilkan bahwa menu atau fitur yang terdapat pada aplikasi iPresence mudah digunakan oleh pengguna.

5. Pertanyaan 5 : Apakah aplikasi nyaman digunakan?

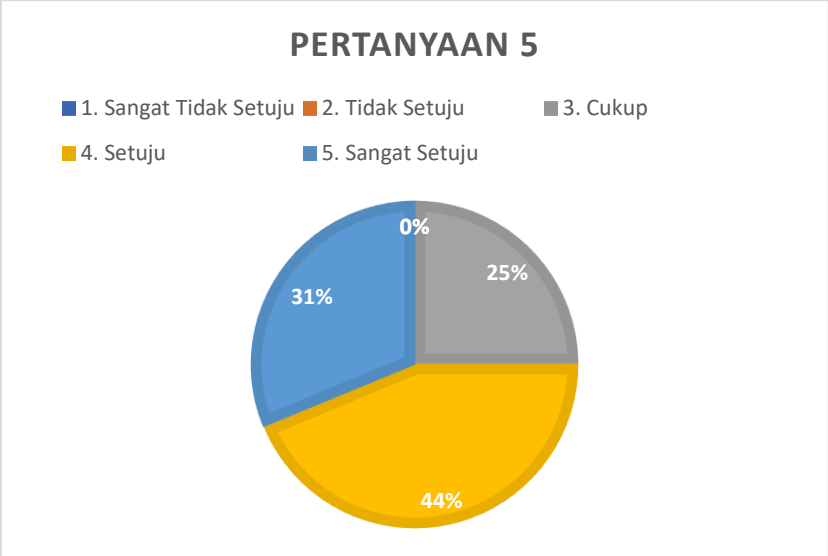


Gambar 4.53 Diagram lingkaran pertanyaan 5

Hasil kuisioner untuk pertanyaan 5 didapat hasil yang sama dengan pertanyaan sebelumnya yaitu 50% responden atau 8 orang menjawab ‘sangat setuju’ dengan pertanyaan 5. 44% responden atau 7 orang responden menjawab ‘setuju’ dan 6% responden atau 1 orang menjawab ‘tidak setuju’.

Berdasarkan data dari diagram lingkaran di atas dapat disimpulkan bahwa aplikasi nyaman digunakan oleh pengguna.

6. Pertanyaan 6 : Apakah aplikasi mudah dipahami?

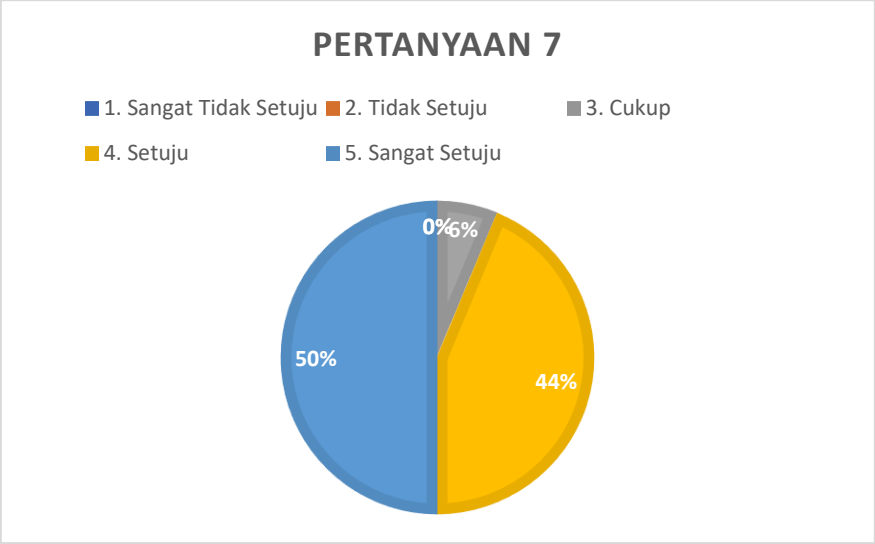


Gambar 4.54 Diagram lingkaran pertanyaan 6

Gambar 4.54 di atas merupakan diagram lingkaran dari data kuisioner pertanyaan 6 yaitu sebesar 31% responden atau 5 orang menjawab ‘sangat setuju’, begitu juga dengan jawaban ‘setuju’ yang didapat sebesar 44% atau sebanyak 7 orang dan 25% responden atau 4 orang menjawab ‘cukup’ untuk pertanyaan 6.

Dari diagram lingkaran di atas dapat disimpilkan bahwa aplikasi presensi otomatis iPresence mudah dipahami oleh pengguna.

7. Pertanyaan 7 : Apakah aplikasi mudah dioperasikan?

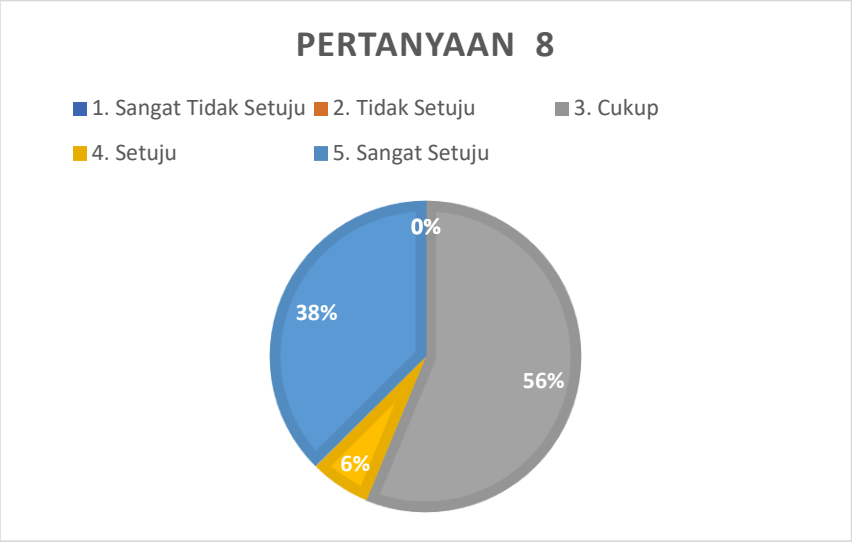


Gambar 4.55 Diagram lingkaran pertanyaan 7

Gambar 4.55 di atas merupakan diagram lingkaran yang menggambarkan hasil kuisioner untuk pertanyaan 7. Dari gambar di atas sebesar 50% atau 8 orang responden menjawab ‘sangat setuju’ untuk pertanyaan 7. 44% responden atau 7 orang responden menjawab ‘setuju’ dan 6% atau 1 orang responden menjawab ‘tidak setuju’.

Berdasarkan data di atas dapat disimpulkan bahwa responden setuju bahwa aplikasi presensi otomatis iPresence mudah dioperasikan.

8. Pertanyaan 8 : Apakah aplikasi sudah memenuhi kebutuhan di labolatorium Teknik Elektro UMY?

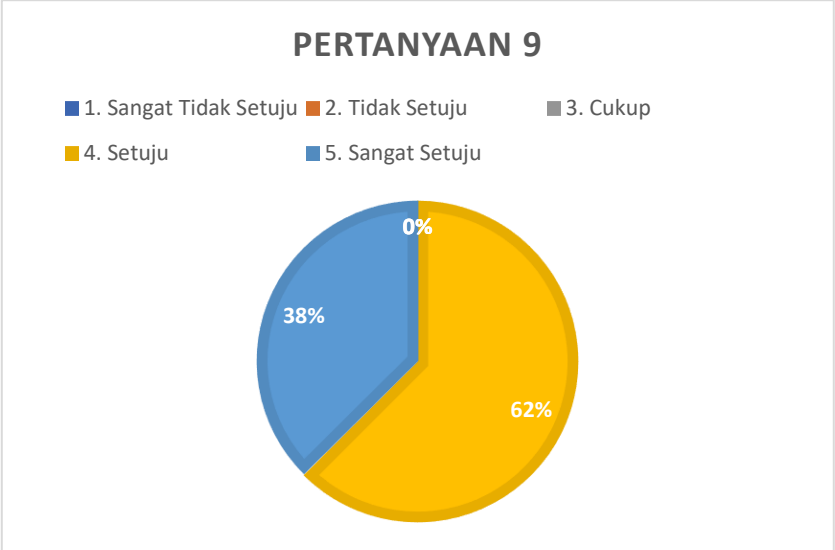


Gambar 4.56 Diagram lingkaran pertanyaan 8

Gambar 4.56 di atas merupakan diagram lingkaran yang menggambarkan hasil kuisioner untuk pertanyaan 8. Dari gambar di atas sebesar 38% responden atau 6 orang menjawab ‘sangat setuju’ dengan pertanyaan 8 dan 6% responden atau 1 orang menjawab ‘setuju’. Sedangkan sisanya sebesar 56% responden atau 9 orang menjawab ‘cukup’.

Dari data di atas dapat disimpulkan bahwa aplikasi cukup memenuhi kebutuhan labolatorium teknik elektro.

9. Pertanyaan 9 : Apakah aplikasi bermanfaat bagi pengguna?

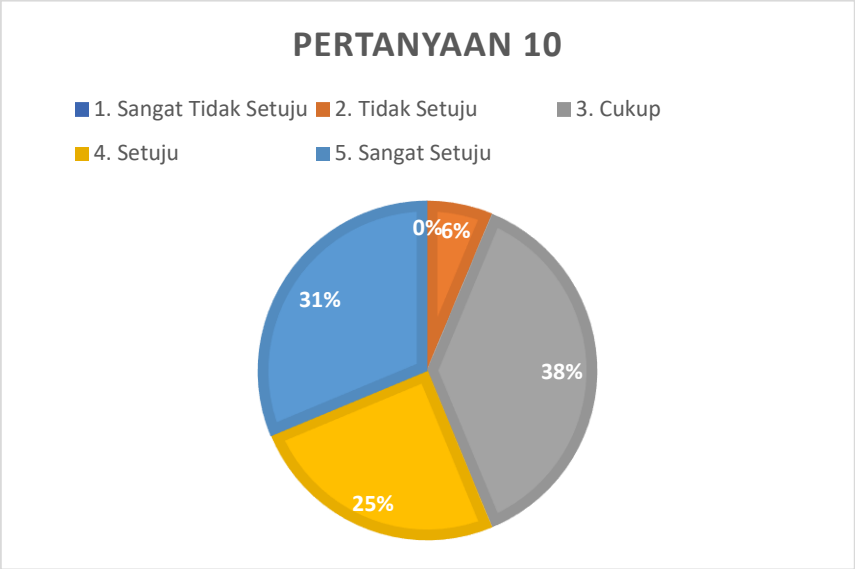


Gambar 4.57 Diagram lingkaran pertanyaan 9

Gambar 4.57 di atas merupakan diagram lingkaran yang menggambarkan hasil kuisioner untuk pertanyaan 9. Pada diagram lingkaran di atas, 38% responden atau 6 orang menjawab ‘sangat setuju’ dan sisanya menjawab ‘setuju’ dengan pertanyaan 9.

Berdasarkan data di atas dapat disimpulkan bahwa responden setuju bahwa aplikasi iPresence bermanfaat bagi pengguna.

10. Pertanyaan 10 : Apakah aplikasi mempunyai kemampuan dan fungsi yang diharapkan?



Gambar 4.58 Diagram lingkaran pertanyaan 10

Gambar 4.58 di atas merupakan diagram lingkaran yang menggambarkan hasil kuisioner untuk pertanyaan 10. Sebesar 31% responden atau 5 orang responden menjawab ‘sangat setuju’ dengan pertanyaan 10. 25% atau 4 orang menjawab ‘setuju’, 38% atau 6 orang menjawab ‘cukup’ dan sisanya sebesar 6% menjawab ‘tidak setuju’.

Dari data di atas dapat disimpulkan bahwa rata-rata pengguna merasa aplikasi iPresence cukup mempunyai fungsi yang diharapkan.