

BAB IV PERANCANGAN DAN PENGUJIAN

4.1 Perancangan

4.1.1 Kebutuhan *Software* dan *Hardware*

Spesifikasi *software* yang digunakan untuk membuat aplikasi iPresence adalah sebagai berikut.

1. Sistem Operasi : Windows 10 64 bit
2. Bahasa Pemrograman : Java dan XML
3. *Software* Pengembangan : Android Studio 2.3.3
4. Java SDK : Android Development Tools V.
Android SDK Platform-tools Rev.
5. *Database* : Mesosfer Cloud

Adapun spesifikasi *hardware* yang digunakan adalah sebagai berikut:

1. Asus N46VZ
2. Core i7 Gen 3 (*codename: Ivy Bridge*) 3630QM 2.40 GHz
3. 8 GB RAM

4.1.2 Implementasi Rancangan Sistem Sisi Perangkat Bergerak

Implementasi rancangan sistem terbagi dari dua sisi, yaitu dari sisi perangkat bergerak (android) dan perangkat *database*. Rancangan sistem android memiliki 3 mode *user*, yaitu, dosen, mahasiswa, dan admin. Namun terdapat fitur yang terdapat di semua mode, yaitu halaman untuk menampilkan profil dan jadwal kuliah.

Sebelum masuk ke mode masing-masing, ada beberapa sistem yang akan dijalankan oleh *device* android. Tampilan pertama kali saat aplikasi dibuka adalah tampilan Splash. Activity yang akan dijalankan adalah *class* SplashActivity.java dengan *activity_splash* sebagai tampilannya. Tampilan *splash* ditunjukkan pada gambar 4.1. *Class* SplashActivity akan melakukan pengecekan *session* apakah sebelumnya *user* sudah pernah melakukan *login*. Jika belum, maka sistem akan

memanggil *class* IntroActivity yang akan menampilkan tampilan activity_intro sebagai halaman intro. Namun jika user sudah pernah *login* sebelumnya, maka sistem akan mengecek status *user*, apakah dosen, mahasiswa, atau admin. Cuplikan kode pengecekan *session* ditunjukkan oleh gambar 4.2.

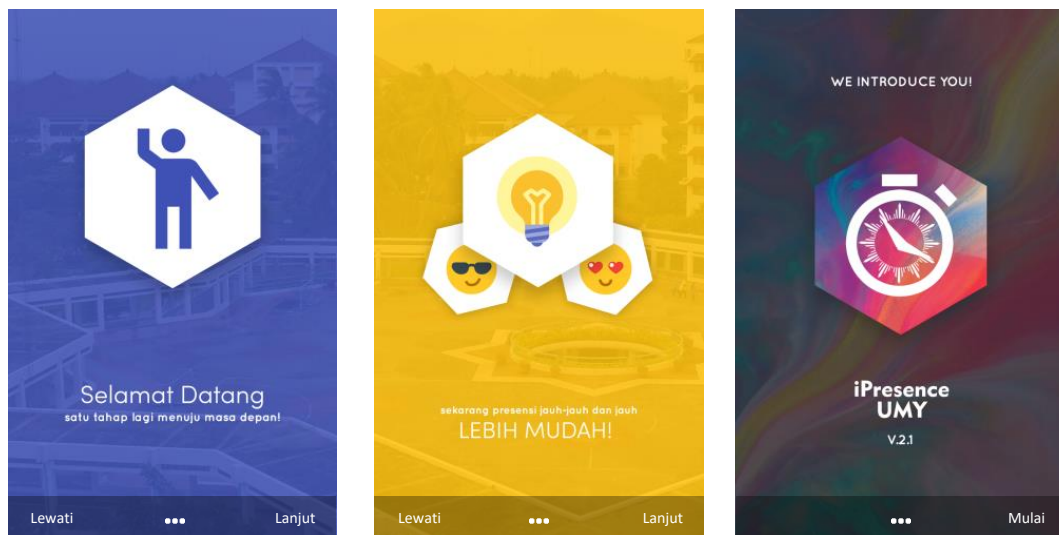


Gambar 4.1 Tampilan SplashActivity

```
if (user != null) {
    user.fetchAsync(new GetCallback<MesosferUser>() {
        @Override
        public void done(MesosferUser mesosferUser, MesosferException e) {
            Number status = user.getDataNumber("statuta");
            String status_string = status.toString();
            if(status_string=="0"){
                Intent intent = new Intent(SplashActivity0.this,
MainAdminActivity.class);
                startActivity(intent);
                finish();
            }
            else
            if(status_string=="1") {
                // user logged in, open main activity
                Intent intent = new Intent(SplashActivity0.this,
MainDosenActivity.class);
                startActivity(intent);
                finish();
            }
            else
            if(status_string=="2"){
                Intent intent = new Intent(SplashActivity0.this,
MainMahasiswaActivity.class);
                intent.putParcelableArrayListExtra("BEACONS", beacons);
                startActivity(intent);
                finish();
            }
        }
    });
} else {
    Intent intent = new Intent(SplashActivity0.this, IntroActivity.class);
    startActivity(intent);
}
```

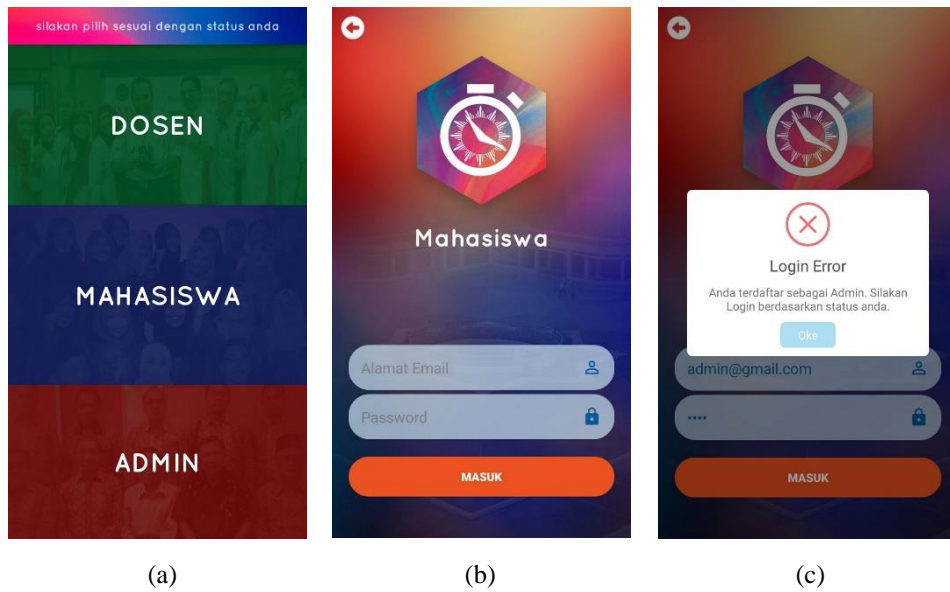
Gambar 4.2 Cuplikan kode *check session*

Setiap *user* memiliki status yang dipresentasikan dari 0-2. Nilai 0 adalah status admin, 1 adalah status dosen, sedangkan 2 adalah status mahasiswa. Jika sistem menemukan status 0, maka sistem akan memanggil *class* `MainAdminActivity.java` dengan `activity_admin.xml` sebagai tampilannya. Jika status yang terdeteksi 1, maka *class* yang dipanggil adalah `MainDosenActivity.java` dan `activity_main_dosen.xml`. Lalu saat statusnya 2, maka *class* yang dipanggil adalah `MainMahasiswaActivity` dan `mhs_activity_main.xml` sebagai tampilannya. Jika ternyata belum ada *user* yang login, maka *class* yang dipanggil adalah `IntroActivity.java`, dimana terdapat tampilan intro berupa pengenalan sekilas tentang aplikasi, kemudian opsi login dan kemudian login menurut status masing-masing.



Gambar 4.3 Tampilan IntroActivity

Gambar 4.3 menunjukkan halaman intro saat belum ada *user* yang login. Setelah halaman intro, maka aplikasi akan mengarahkan ke tampilan opsi *user*. *Class* yang dipanggil pada halaman ini adalah `OpsiActivity.java` dengan `all_activity_opsi.xml` sebagai tampilannya, seperti pada gambar 4.4a. Opsi *user* pada halaman tersebut harus dipilih sesuai dengan status *user*. Karena jika tidak sesuai dengan status, maka akan muncul *dialog* yang menginformasikan bahwa *user* salah dalam memilih mode *user*. Seperti pada gambar 4.4c yang menampilkan pesan *error* karena login tidak sesuai dengan status *user*.



Gambar 4.4 (a). Halaman Opsi, (b). Halaman Login, (c). Error saat *user* tidak sesuai dengan status

4.1.2.1 Mode Admin

Mode admin memiliki tampilan halaman utama seperti gambar 4.5.



Gambar 4.5 Halaman utama Admin

Pada halaman utama Admin, terdapat 4 fitur utama admin dalam menjalankan aktivitas perkuliahan, terdiri dari menu Input User, Lihat Database, Input Jadwal, dan Menu Reset. Selain itu, saat menekan tombol menu di sisi kiri atas, maka akan muncul *Navigation Drawer* yang berfungsi untuk menampilkan profil singkat beserta foto dan beberapa pilihan. *Navigation Drawer* dapat dibuka juga dengan cara *swipe* ke sebelah kiri. Adapun tampilan *Navigation Drawer* seperti gambar 4.6.



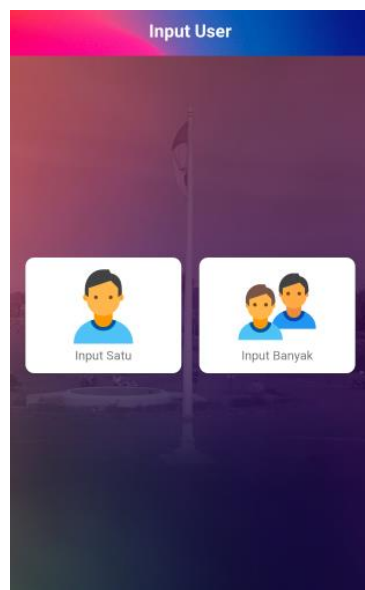
Gambar 4.6 *Navigation Drawer* pada Admin

Berikut adalah penjelasan setiap menu yang terdapat pada Admin.

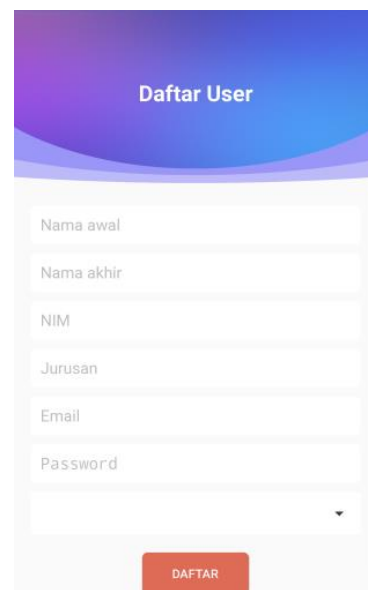
4.1.2.1.1 Input User

Input user merupakan sebuah menu yang berfungsi untuk memasukkan user. Saat menu input user ditekan, maka akan muncul ke tampilan opsi input *user* seperti gambar 4.7. Terdapat dua mode input user, yaitu input dalam jumlah satuan dan dalam jumlah banyak. Ketika input satuan dipilih, maka akan muncul tampilan seperti

gambar 4.8. Pada sistem, input satuan akan memanggil *class* InputSatuan.java dengan admin_activity_input_satuan.xml sebagai *interface*. Kelas ini berfungsi untuk mendaftarkan 1 user, baik itu mahasiswa, dosen, maupun admin. Identitas yang harus diisi adalah nama awal, nama akhir, *user ID*, email, password, dan status user. Status user tersedia dalam bentuk *Option Box* dengan pilihan admin, dosen, dan mahasiswa.



Gambar 4.7 Opsi Input User



Gambar 4.8 Input 1 user

Setelah semua data diisi, maka ketika *user* menekan tombol Daftar, maka sistem akan menyimpan data ke database user yang ada di mesosfer. Penyimpanan tersebut menggunakan *method* yang sudah tersedia di Mesosfer SDK, yaitu Register Callback. Cuplikan kode untuk menyimpan *user* terdapat pada gambar 4.9.

Selain memasukkan 1 *user*, admin juga dapat memasukkan *user* dalam jumlah banyak. Data mahasiswa teknik elektro digolongkan menjadi dua kelompok di setiap angkataannya. Hal ini dilakukan karena *database* mesosfer yang mampu melakukan *record* sebanyak 100 *users* saja dalam satu waktu. Adapun data yang tersedia adalah mulai dari angkatan 2014 sampai 2017. Data mahasiswa

tersebut dibuat dalam satu kolom berbentuk Java Script Object Notation atau JSON yang terdapat pada *database* DataMahasiswa. Tabel *database* DataMahasiswa dapat dilihat pada tabel 4.1.

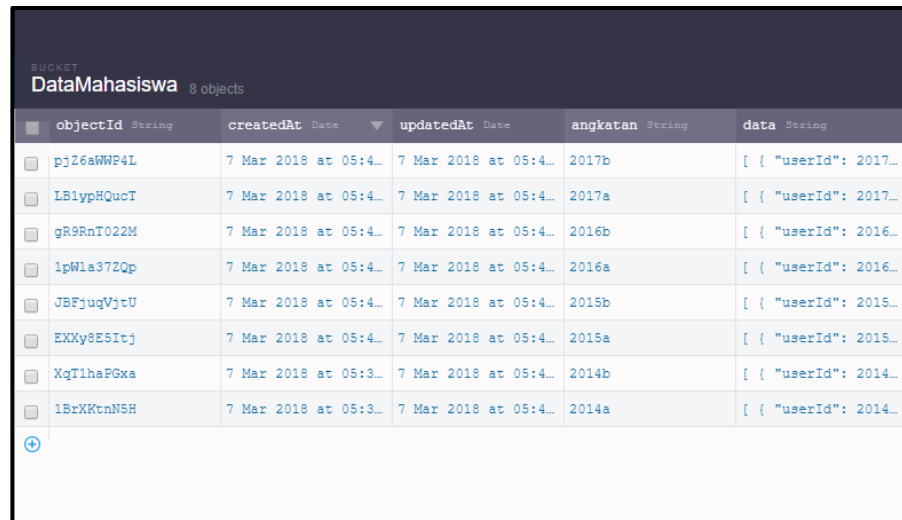
```
MesosferUser newUser = MesosferUser.createUser();
newUser.setEmail(email);
newUser.setPassword(password);
newUser.setFirstName(firstname);
newUser.setLastName(lastname);
newUser.setData("userId", nim);
newUser.setData("jurusan", jurusan);
newUser.setData("status", posisi-1);
newUser.setData("matkul1", "0");
newUser.registerAsync(new RegisterCallback() {
    @Override
    public void done(MesosferException e) {
        loading.dismiss();
        AlertDialog.Builder builder = new AlertDialog.Builder(InputSatuan.this);
        if (e != null) {
            builder.setNegativeButton(android.R.string.ok, null);
            builder.setTitle("Error Happen");
            builder.setMessage(
                String.format(Locale.getDefault(), "Error code: %d\nDescription: %s",
                    e.getCode(), e.getMessage())
            );
            dialog = builder.show();
            return;
        }
        builder.setNegativeButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                setResult(Activity.RESULT_OK, new Intent());
                finish();
            }
        });
        builder.setTitle("Register Succeeded");
        builder.setMessage("Thank you for registering.");
        dialog = builder.show();
    }
});
```

Gambar 4.9 Cuplikan Kode saat menambahkan 1 user

```
[
  {
    "userId": 20170120001,
    "userFullName": "M. Febri Nanda Wahidi",
    "userFirstName": "M.",
    "userLastName": "Febri Nanda Wahidi",
    "email": "m.febri.ft17@mail.umy.ac.id",
    "password": "elektroumy",
    "jurusan": "Teknik Elektro",
    "status": 2
  },
  {
    "userId": 20170120003,
    "userFullName": "Ikbal Maulana",
    "userFirstName": "Ikbal",
    "userLastName": "Maulana",
    "email": "ikbal.maulana.ft17@mail.umy.ac.id",
    "password": "elektroumy",
    "jurusan": "Teknik Elektro",
    "status": 2
  }
]
```

Gambar 4.10 Cuplikan JSON Data Mahasiswa

Tabel 4.1 Database DataMahasiswa



objectID	String	createdAt	Date	updatedAt	Date	angkatan	String	data	String
<input type="checkbox"/>	pj26aWWF4L	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2017b		[{ "userId": 2017...	
<input type="checkbox"/>	LB1ypHQucT	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2017a		[{ "userId": 2017...	
<input type="checkbox"/>	gR9RnT022M	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2016b		[{ "userId": 2016...	
<input type="checkbox"/>	1pW1a37ZQp	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2016a		[{ "userId": 2016...	
<input type="checkbox"/>	JBFjuqVjtU	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2015b		[{ "userId": 2015...	
<input type="checkbox"/>	EXXy8ESItj	7 Mar 2018 at 05:4...		7 Mar 2018 at 05:4...		2015a		[{ "userId": 2015...	
<input type="checkbox"/>	XqTlhaPGxa	7 Mar 2018 at 05:3...		7 Mar 2018 at 05:4...		2014b		[{ "userId": 2014...	
<input type="checkbox"/>	1BrXKtnN5H	7 Mar 2018 at 05:3...		7 Mar 2018 at 05:4...		2014a		[{ "userId": 2014...	

Data berupa JSON terdapat pada kolom data. Cuplikan kode JSON dapat dilihat pada gambar 4.10. Data JSON tersebut berisikan identitas lengkap yang nanti akan di *parsing* dan diinputkan ke database User. Saat *user* memilih pilihan input banyak, maka sistem akan memanggil *class* `InputBanyakUserActivity.java` dengan `admin_activity_input_user.xml` sebagai tampilannya. Pada *class* ini akan ditampilkan daftar kelompok angkatan yang bisa diinputkan ke dalam user.

Saat *user* memilih salah satu kelompok *user* seperti di atas, maka data kelompok *user* tersebut akan diolah menjadi data yang dapat dimasukkan ke database *user*. Garis besarnya adalah data yang berupa JSON akan diinputkan satu persatu dengan perintah *looping for*. Berikut cuplikan perintah input banyak *user*.


```

final MesosferData dataUser = InputBanyakUserActivity.this.akhir.get(i);
String data = dataUser.getDataString("data");
String angkatan = dataUser.getDataString("angkatan");

try {
    JSONArray contacts = new JSONArray(data);

    for (int j = 0; j <= contacts.length(); j++) {
        JSONObject c = contacts.getJSONObject(j);

        final String namaawal = c.getString("userFirstName");
        final String namaakhir = c.getString("userLastName");
        String nim = c.getString("userId");
        String email = c.getString("email");
        String password = c.getString("password");
        String jurusan = c.getString("jurusan");
        String status = c.getString("status");
        Number statuta = Double.valueOf(status);

        MesosferUser newUser = MesosferUser.createUser();
        newUser.setEmail(email);
        newUser.setData("userId", nim);
        newUser.setData("jurusan", jurusan);
        newUser.setData("statuta", statuta);
        newUser.setFirstName(namaawal);
        newUser.setLastName(namaakhir);
        newUser.setPassword(password);
        newUser.setData("matkul1", "0");
        newUser.registerAsync(new RegisterCallback() {
            @Override
            public void done(MesosferException e) {

            }
        });
    }
}

```

Gambar 4.11 Cuplikan kode parsing JSON dan input ke *database user*

Jika terdapat 100 data mahasiswa pada satu kelompok, maka perintah input data akan dilakukan 100 kali dengan menggunakan *looping for*. Program akan melakukan *parsing* data JSON, kemudian setiap variabel akan dideklarasikan. Setelah itu, data akan disimpan dengan *command* **MesosferUser.createUser()**.

4.1.2.1.2 Lihat Database

Setiap basis data atau *database* perlu dipantau agar tidak terjadi kesalahan dalam input dan output data. Maka dari itu, menu admin dilengkapi oleh menu **lihat database** yang berfungsi sebagai halaman untuk memantau semua *database* di Mesosfer. Adapun java

class yang digunakan untuk menu ini adalah MenuDatabase.java dengan admin_activity_database.xml sebagai *interface*. Tampilan lihat database dapat dilihat pada gambar 4.12a. Pada menu ini terdapat empat pilihan yang akan dijelaskan sebagai berikut.

1. Lihat User

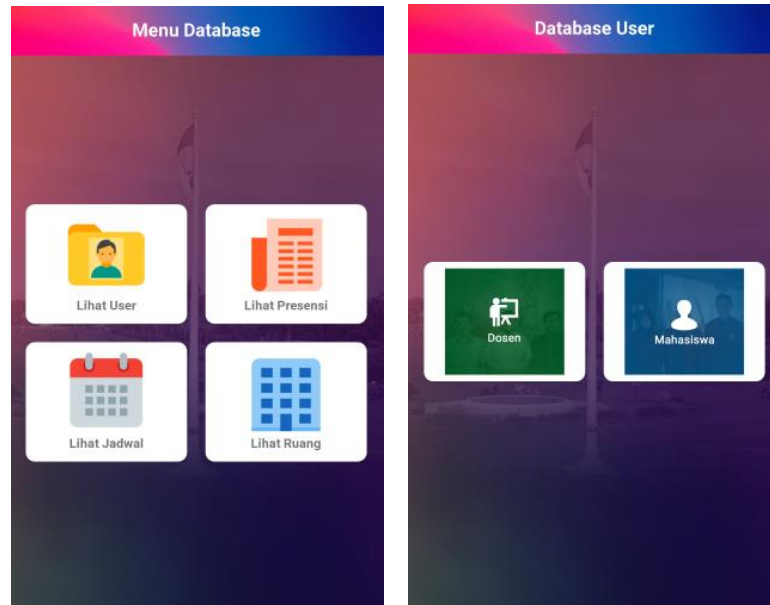
Menu ini memiliki dua pilihan menu, yaitu *database user* mahasiswa dan dosen. Jika memilih opsi mahasiswa, maka aplikasi akan menuju ke tampilan database mahasiswa dengan Database Mahasiswa.java sebagai *class* dan admin_activity_user.xml sebagai tampilan. Menu *database* mahasiswa ini akan meminta *user* untuk memilih kelompok angkatan yang akan ditampilkan datanya. Setelah *user* memilih angkatan, maka *list user* mahasiswa akan tampil.

Selain *database* mahasiswa, admin juga dapat melihat *database* dari dosen. Jika pada tampilan menu *database* admin memilih opsi Dosen, maka tampilan yang muncul adalah list dari *user* dosen seperti gambar 4.12d.

2. Lihat Presensi

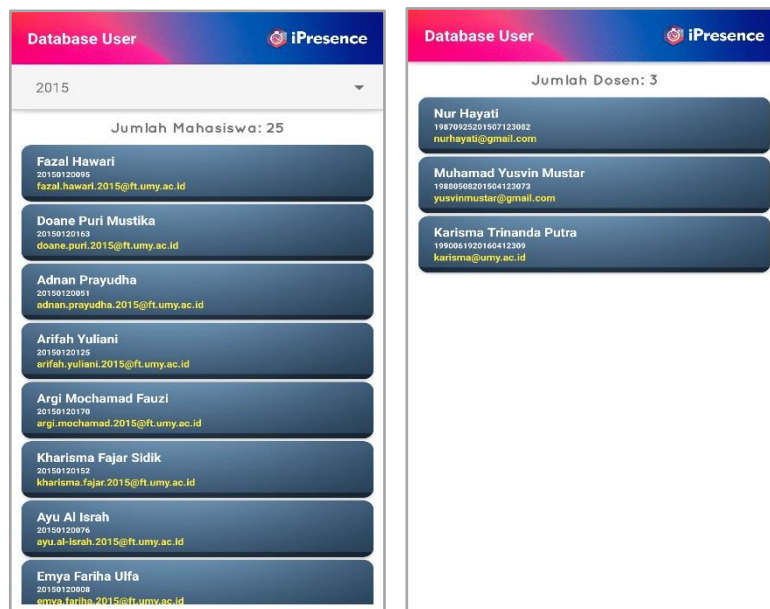
Menu lihat presensi berfungsi untuk melihat data presensi yang terdapat pada setiap mata kuliah. Pada menu ini dapat diketahui kehadiran mahasiswa di setiap pertemuan kuliah. Class yang digunakan oleh menu lihat presensi adalah AdminPresensiActivity.java. Saat admin memilih opsi ini, maka akan ada tampilan untuk memilih mata kuliah yang hendak dilihat presensinya. Setelah memilih mata kuliah, maka selanjutnya tampilan akan menuju ke *class* AdminPresensiMatkulActivity.java. dengan all_activity_presensi_masuk.xml sebagai tampilan. Pada tampilan tersebut akan diminta untuk memilih salah satu pertemuan. Saat pertemuan dipilih, maka list mahasiswa akan muncul beserta keterangan hadir atau tidaknya. Halaman presensi mata kuliah juga

dilengkapi dengan fitur menginputkan presensi ke *database* HasilPresensi yang kemudian dapat diubah ke dalam bentuk tabel.



(a)

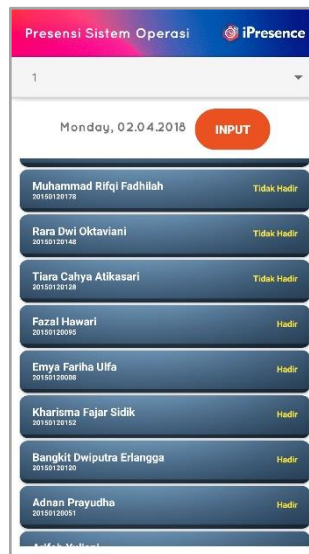
(b)



(c)

(d)

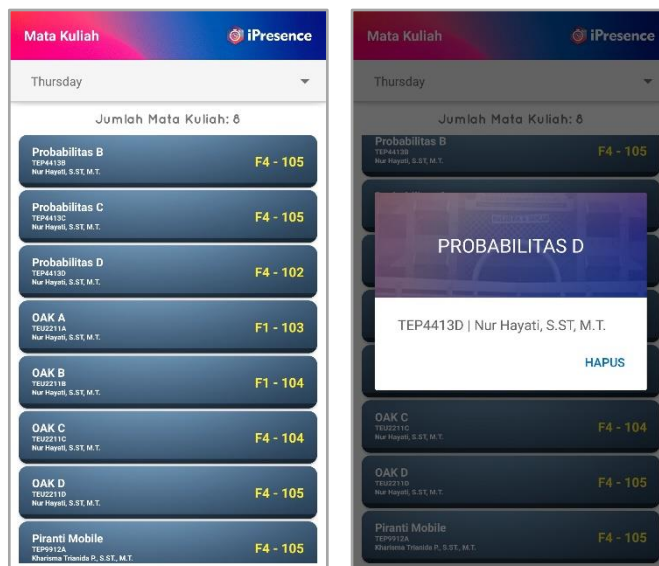
Gambar 4.12 Tampilan Lihat Database 1 (a). Menu Database, (b). Menu Database User, (c). List *user* mahasiswa, (d). List *user* dosen



Gambar 4.13 Tampilan daftar kehadiran mahasiswa

3. Lihat Jadwal

Menu selanjutnya adalah lihat jadwal yang berfungsi untuk melihat semua jadwal yang ada pada *database*, berikut dengan nama pengampu, ruang, dan kode mata kuliah masing-masing. Selain itu, admin memiliki hak akses untuk menghapus mata kuliah lewat menu ini sehingga mata kuliah yang terdapat pada *database* akan dihapus.



Gambar 4.14 Tampilan daftar mata kuliah (a) dan tampilan untuk menghapus mata kuliah (b)

4. Lihat Ruang

Menu lihat ruang akan menampilkan seluruh ruang yang ada pada *database*. Variabel yang akan muncul pada daftar ruang tersebut adalah nama ruang, lokasi, fakultas, dan identitas major beacon. Menu ini yang dapat memantau identitas beacon pada setiap koridor.



Gambar 4.15 Tampilan daftar ruang

4.1.2.1.3 Input Jadwal

Input jadwal merupakan menu yang berfungsi untuk menginputkan data mata kuliah lengkap beserta waktu dan ruangan yang telah tersedia pada pangkal data. Data tersebut kemudian akan diinputkan ke database **MataKuliah**. Pangkal data adalah sebuah data yang berbentuk JSON. Cara inputnya sama seperti menginputkan banyak *user*, yaitu dengan cara *parsing* JSON, lalu diinputkan dengan perintah *looping for*. Pangkal data terdapat pada *database* **DataMatkul**. Gambar *database* **DataMatkul** dapat dilihat di gambar 4.15.

objectId	createdAt	updatedAt	data	nomor
febUoyBnRh	27 Feb 2018 at 09:...	28 Feb 2018 at 07:...	[{ "daftarMataKul...	1

Gambar 4.16 Database DataMatkul

Adapun cuplikan kode input mata kuliah terdapat pada gambar 4.17.

```
JSONArray contacts = new JSONArray(angkatan.getDataString("data"));

// Getting JSON Array node
//JSONArray contacts = jsonObj.getJSONArray("angkatan2015");
contactList.clear();
for (int i = 0; i < contacts.length(); i++) {
    JSONObject c = contacts.getJSONObject(i);

    final String daftarMahasiswa = c.getString("daftarMahasiswa");
    final String kelas = c.getString("kelas");
    final String matkulKode = c.getString("matkulKode");
    final String matkulNama = c.getString("matkulNama");
    final String matkulNamaGet = c.getString("matkulNamaGet");
    final String matkulPengampu = c.getString("matkulPengampu");
    final String ruang = c.getString("ruang");
    final String ruangText = c.getString("ruangText");

    final HashMap<String, String> contact = new HashMap<>();
    contact.put("nama", matkulNama);
    contactList.add(contact);

    MesosferData put_Matkul = MesosferData.createData("MataKuliah");
    put_Matkul.setData("daftarMahasiswa", daftarMahasiswa);
    put_Matkul.setData("kelas", kelas);
    put_Matkul.setData("matkulKode", matkulKode);
    put_Matkul.setData("matkulNama", matkulNama);
    put_Matkul.setData("matkulNamaGet", matkulNamaGet);
    put_Matkul.setData("matkulPengampu", matkulPengampu);
    put_Matkul.setData("ruang", ruang);
    put_Matkul.setData("ruangText", ruangText);
    put_Matkul.saveAsync(new SaveCallback() {
        @Override
        public void done(MesosferException e) {

        }
    });
});
```

Gambar 4.17 Cuplikan kode input mata kuliah

4.1.2.1.4 Menu Reset

Menu reset merupakan suatu fungsi dimana user dapat menghapus semua data yang ada pada *database*. Saat *user* memilih opsi menu reset, maka tampilan akan menuju ke opsi reset yang terdapat pada *class* OpsiResetActivity.java. *Class* ini menampilkan tiga opsi yaitu sebagai berikut.

1. Opsi Reset Mata Kuliah

Opsi ini akan menghapus semua daftar mata kuliah yang ada pada *database* **MataKuliah**. Cuplikan kode dari reset mata kuliah adalah sebagai berikut.

```
public void hapusdatamatkul(){
    MesosferQuery<MesosferData> DataMatkul = MesosferData.getQuery("MataKuliah");
    DataMatkul.findAsync(new FindCallback<MesosferData>() {
        @Override
        public void done(List<MesosferData> list, MesosferException e) {
            for (MesosferData aku : list){
                aku.deleteAsync(new DeleteCallback() {
                    @Override
                    public void done(MesosferException e) {
                        }
                });
            }
        }
    });
}
```

Gambar 4.18 Cuplikan kode reset mata kuliah

2. Opsi Reset Presensi

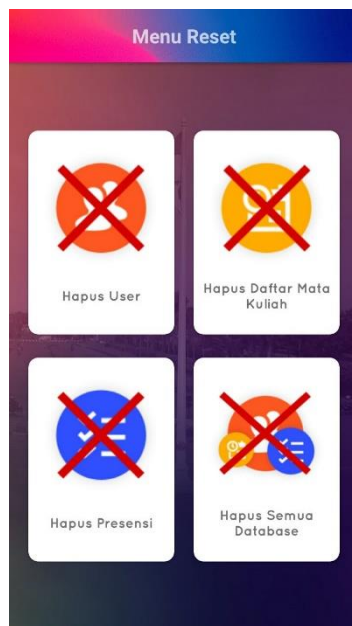
Opsi reset presensi akan menghapus semua data presensi yang ada pada *database*. Berikut cuplikan kodenya.

```
public void hapuspresensi(){
    MesosferQuery<MesosferData> hapusPresensi = MesosferData.getQuery("MataKuliah");
    hapusPresensi.findAsync(new FindCallback<MesosferData>() {
        @Override
        public void done(List<MesosferData> list, MesosferException e) {
            for (MesosferData getMatkul : list){
                String matkulnama = getMatkul.getDataString("matkulNamaGet");
                MesosferQuery<MesosferData> matkul = MesosferData.getQuery(matkulnama);
                matkul.findAsync(new FindCallback<MesosferData>() {
                    @Override
                    public void done(List<MesosferData> list, MesosferException e) {
                        for (MesosferData matkulnya : list){
                            matkulnya.deleteAsync(new DeleteCallback() {
                                @Override
                                public void done(MesosferException e) {
                                    }
                            });
                        }
                    }
                });
            }
        }
    });
}
```

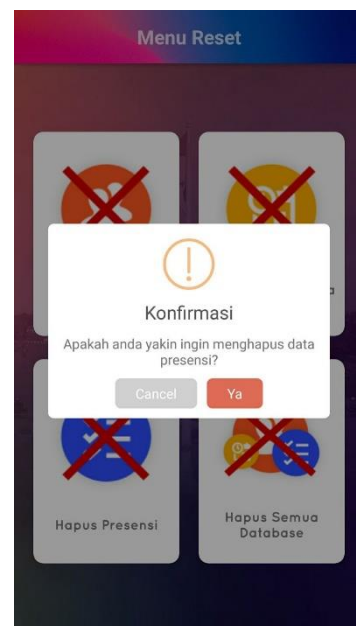
Gambar 4.19 Cuplikan kode reset presensi

3. Opsi Reset Semua *Database*

Opsi reset semua berfungsi untuk menghapus seluruh data pada *database* secara keseluruhan. Semua data presensi dan mata kuliah akan terhapus dengan memilih opsi ini. Adapun kode pemrogramannya adalah gabungan dari kode opsi reset presensi dan opsi reset mata kuliah.



Gambar 4.20 Menu Reset



Gambar 4.21 Menu Konfirmasi

Semua perintah pada menu memiliki *dialog* konfirmasi seperti gambar 4.21. Dialog konfirmasi berfungsi untuk memastikan *user* sesuai dalam memilih opsi.

4.1.2.2 Mode Dosen

User selanjutnya adalah dosen. Dosen memiliki status bernilai 1. *Class* yang berjalan di halaman utama mode dosen adalah `MainDosenActivity.java` dengan `dosen_activity_main.xml` sebagai tampilannya. Tampilan seperti gambar 4.22. Halaman utama dosen memuat identitas dosen, yaitu nama lengkap, nomor induk kepegawaian, jurusan, serta

status yaitu pengajar. Lalu di bawahnya terdapat daftar mata kuliah yang diambil beserta ruang dan waktu kuliah.



Gambar 4.22 Halaman Utama Dosen

Selain identitas dosen dan jadwal mata kuliah, terdapat 2 *button* utama, yaitu tombol presensi dan tombol kirim notifikasi. Berikut penjelasan kedua *button* tersebut.

1. Presensi

Presensi pada dosen berfungsi untuk mengaktifkan mata kuliah pada satu pertemuan mata kuliah. Presensi mata kuliah mahasiswa bergantung kepada dosen. Jika dosen tidak mengaktifkan mata kuliah, maka tidak akan ada mata kuliah di pertemuan itu, atau dengan kata lain mahasiswa tidak akan bisa melakukan presensi di pertemuan tersebut.

Saat memilih *button* presensi, maka tampilan akan berpindah ke tampilan untuk memilih ruang. Adapun *class* yang bekerja pada proses ini adalah `RuangKuliahActivity.java` dengan `all_activity_room`.

xml sebagai *interface*-nya. *Class* ini membutuhkan konektivitas bluetooth karena *device* akan mencari sinyal yang dipancarkan oleh beacon. Jika *device* android belum menyalakan bluetooth, maka akan muncul *dialog box* berupa permintaan mengaktifkan bluetooth. Jika bluetooth sudah menyala, maka *device* akan menerima sinyal dari beacon yang memiliki identitas yang merepresentasikan identitas tiap ruangan yang berada dalam satu koridor.

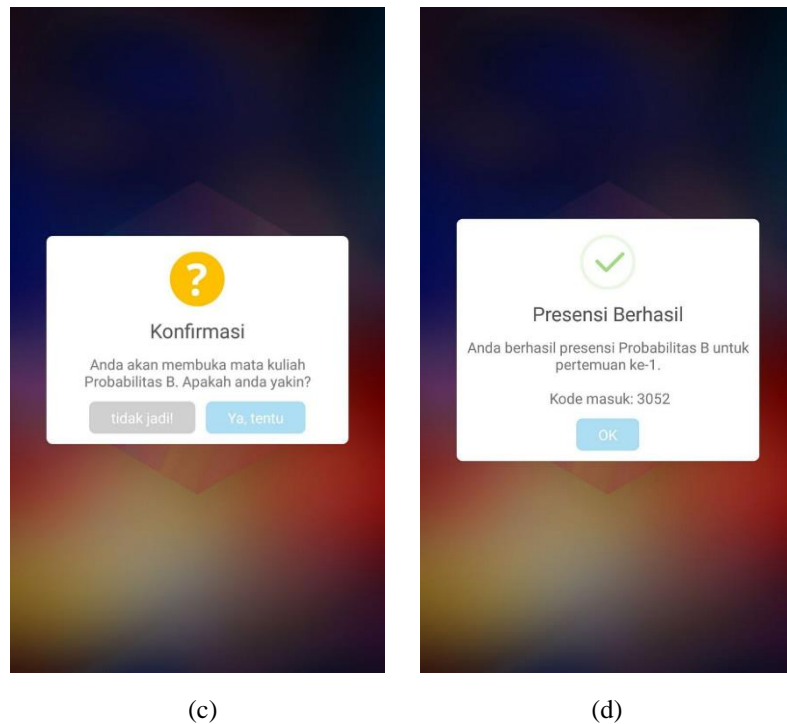
Saat nama ruangan telah terdeteksi, maka yang selanjutnya adalah memilih mata kuliah yang akan diaktifkan. Lalu selanjutnya akan ada *dialog* konfirmasi apakah dosen akan mengaktifkan mata kuliah di pertemuan tersebut. Jika dosen memilih ya, maka mata kuliah di pertemuan itu berhasil diaktifkan. *Dialog box success* akan muncul. Selain notifikasi berhasil terdapat juga kode masuk yang diberikan secara *random* oleh sistem. Kode masuk tersebut berfungsi sebagai *password* bagi mahasiswa yang akan melakukan presensi. Hal ini bertujuan untuk mengurangi kecurangan mahasiswa saat presensi. Berikut adalah simulasi mengaktifkan mata kuliah oleh dosen.



(a)



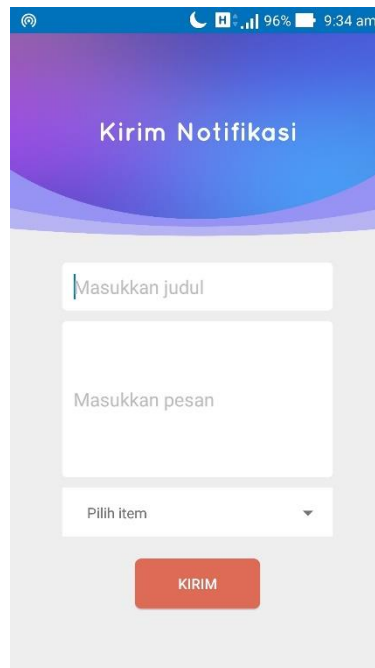
(b)



Gambar 4.23 (a) Tampilan memilih ruangan, (b) Memilih mata kuliah,
(c) Dialog box konfirmasi, (d) Dialog box berhasil

2. Kirim Notifikasi

Button kirim notifikasi berfungsi untuk menginformasikan kepada mahasiswa melalui aplikasi iPresence. Saat dosen memilih *button* kirim notifikasi, maka akan bekerja *class* NotifikasiActivity.java dan tampilan akan berpindah ke dosen_activity_notifikasi.xml (gambar 4.23). Tampilan ini terdapat 2 *form*, yaitu *form* untuk judul notifikasi dan *form* isi. Lalu di bawahnya terdapat opsi koridor mana yang akan dikirimkan notifikasi tersebut. Koridor sendiri mewakili identitas beacon. Dengan kata lain, *range* notifikasi bisa dipilih sesuai dengan koridor. Setelah opsi koridor, terdapat *button* **kirim** di bagian paling bawah. Saat dosen menekan *button* kirim, maka *current class* akan melakukan perintah membuat notifikasi dan disimpan pada *database* Storyline pada mesosfer. Cuplikan kode penyimpanan notifikasi terdapat pada gambar 4.24.



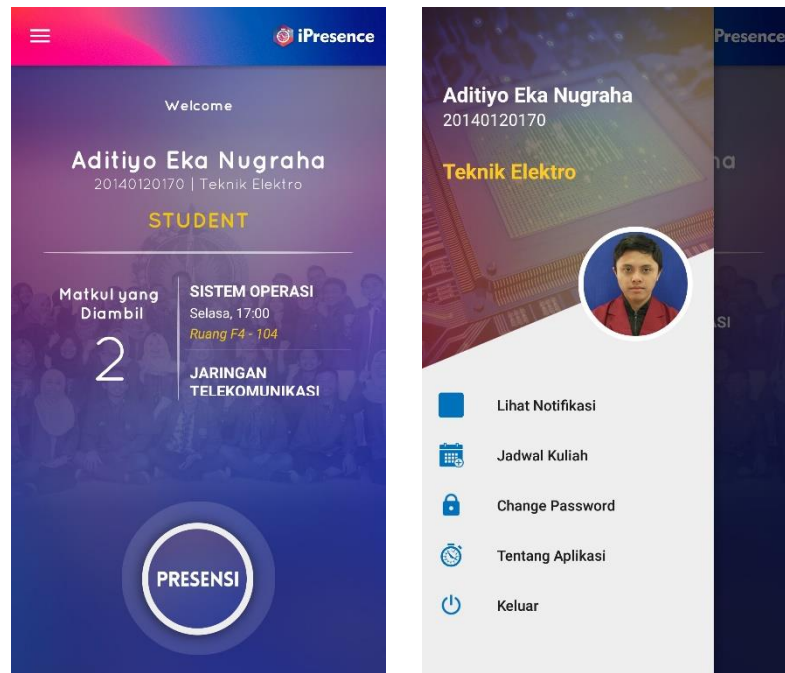
Gambar 4.24 Tampilan Kirim Notifikasi

```
MesosferStoryLineDetail baru = MesosferStorylineDetail.createWithObjectId("5A1jb98mVD");
baru.setAlertMessage(isipesan);
baru.setAlertTitle(judul);
baru.setBeacons(buat.getBeacons());
baru.setShowOn(buat.getShowOn());
baru.setEvent(MesosferBeacon.Event.ENTER);
baru.setCampaign(MesosferStoryline.Campaign.TEXT);
baru.saveAsync(new SaveCallback() {
    @Override
    public void done(MesosferException e) {
    }
});
```

Gambar 4.25 Cuplikan Kode Kirim Notifikasi

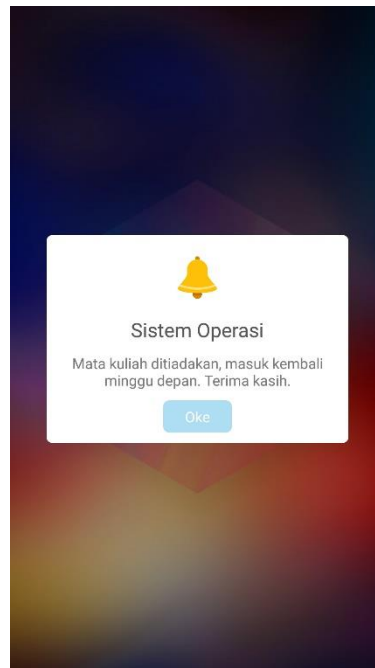
4.1.2.3 Mode Mahasiswa

Mode yang terakhir pada iPresence adalah mode mahasiswa. Mode Mahasiswa memiliki identitas status nomor 2. *Class* yang berjalan pada halaman utama mode mahasiswa adalah `MainMahasiswa Activity.java` dengan `mhs_activity_main.xml` sebagai tampilannya. Tampilan halaman utama mahasiswa persis dengan tampilan mode dosen, namun tidak ada *button* kirim notifikasi. Mode mahasiswa memiliki fitur untuk menerima notifikasi dari dosen. Cara untuk menampilkannya adalah dengan membuka *navigation drawer*, lalu pilih **Lihat Notifikasi**.



(a)

(b)

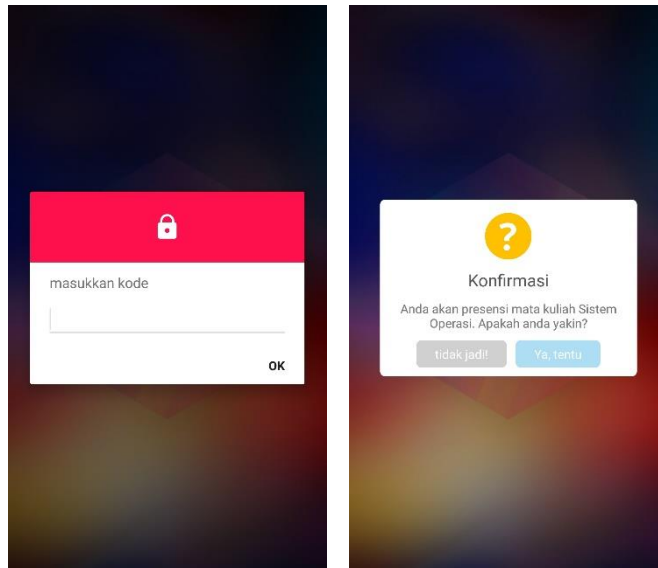


(c)

Gambar 4.26 (a) Halaman utama mode Mahasiswa, (b) *Navigation drawer* pada mode mahasiswa, (c) Tampilan notifikasi.

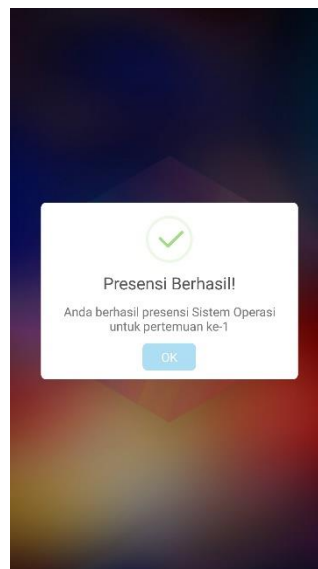
Saat *user* memilih *button* presensi di halaman utama, maka tampilan yang selanjutnya adalah sama seperti di mode dosen, yaitu opsi ruang kuliah

dan mata kuliah. Perbedaannya adalah setelah *user* mahasiswa memilih mata kuliah. *User* akan diminta untuk memasukkan kode masuk sesuai dengan kode yang terdapat pada *user* dosen saat mengaktifkan presensi. Setelah itu baru muncul konfirmasi untuk menginputkan presensi ke *database*.



(a)

(b)



(c)

Gambar 4.27 (a) Input kode masuk, (b) Konfirmasi presensi, (c) Presensi Berhasil

Saat *user* memilih opsi **Ya**, maka *class* akan menjalankan fungsi input presensi sesuai dengan *user* tersebut. Berikut cuplikan kodenya.

```
for (MesosferData terakhir : list){
    for(int k=1; k<=jumlahkestring; k++){
        //kondisi = terakhir.getDataString("pertemuan"+jumlah);
        String satu = Integer.toString(k);
        String selanjutnya = terakhir.getDataString("pertemuan"+satu);
        terakhir.setData("pertemuan"+satu, selanjutnya);
        terakhir.setData("userNama", namafix);
        terakhir.setData("pertemuan"+jumlah, "1");
        terakhir.setData("userNim", userIdfix);
        terakhir.setData("kelas", terakhir.getDataString("kelas"));
        terakhir.setData("jarak", jarak);
        terakhir.setData("akurasi", akurasi);
        terakhir.setData("rssi", rssi);
        terakhir.setData("txpower", txpower);

        terakhir.setData("jumlah", jumlah);
        terakhir.saveAsync(new SaveCallback() {
            @Override
            public void done(MesosferException e) {
                loading.dismiss();
            }
        });
    }
}
```

Gambar 4.28 Cuplikan kode menyimpan data presensi

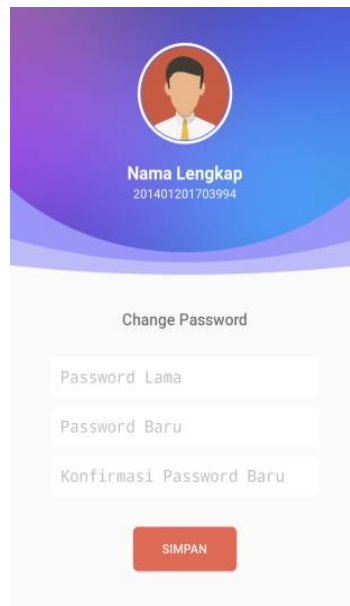
4.1.2.4 Fitur Tambahan

Selain presensi dan notifikasi, aplikasi iPresence dilengkapi dengan fitur menampilkan *list* jadwal. Fitur tersebut terdapat pada halaman utama mode dosen dan mahasiswa. *List* jadwal menampilkan nama mata kuliah, hari, jam masuk, hingga ruang kuliah mata kuliah tersebut.



Gambar 4.29 Tampilan Kirim Notifikasi

Fitur tambahan lainnya adalah semua mode *user* iPresence dapat mengubah password sesuai dengan keinginan *user*. Fitur ubah password terdapat pada *navigation drawer* seperti yang ditunjukkan gambar 4.29.



Gambar 4.30 Tampilan ubah password

4.1.3 Implementasi Sisi *Database*

Implementasi sisi *database* menjelaskan bagaimana aplikasi dapat terhubung dengan *database* dan melakukan transfer data. *Database* yang digunakan pada aplikasi iPresence adalah Mesosfer. Mesosfer adalah *database* yang sudah terintegrasi dengan android via Mesosfer *Software Development Kit* (SDK) yang memudahkan *developer* dalam mengolah data antar android dan *database*. Selain Mesosfer SDK, terdapat Cubeacon SDK yang berfungsi untuk mengintegrasikan antara aplikasi dan beacon dengan *method* yang diolah di program.

Terdapat class yang memiliki fungsi penting dalam menghubungkan antara *device* android dengan *database*, yaitu class MesosferApp.java yang menginisiasikan API key dari Mesosfer (Gambar 4.30). API key mesosfer terdiri dari Application ID dan Application Key. Adapun API key yang digunakan aplikasi iPresence adalah sebagai berikut.

Application ID : **MrKJXn89XH**

Application Key : **v6LS5MpMsqbIQ1Qbj5vUy18cnKDVk9PG**

API key ini tersedia di Mesosfer pada menu Application. Setiap *user* akan mendapatkan API key yang berbeda.


```

package com.umy.ProjectnyaAdit.All;

import android.app.Application;
import com.eyro.cubeacon.*;
import com.eyro.mesosfer.Mesosfer;

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

        // set Cubeacon SDK log level to verbose mode
        com.eyro.cubeacon.Logger.setLevel(LogLevel.VERBOSE);

        // enable background power saver to save battery life up to 60%
        Cubeacon.setBackgroundPowerSaver(true);

        Cubeacon.initialize(this);

        Mesosfer.setPushNotification(true);

        // initialize Mesosfer SDK
        Mesosfer.initialize(this, "MrKJXn89XH", "v6LS5MpMsqbIQ1Qbj5vUy18cnKDVk9PG");

        // initialize Cubeacon SDK

    }
}

```

Gambar 4.31 Cuplikan kode MesosferApp.java

Setelah menginisiasi aplikasi, selanjutnya adalah melakukan *managing* pada *database*. *Database* Mesosfer memiliki 6 *database* utama yang diantaranya dibagi lagi menjadi beberapa bagian. Selain itu terdapat opsi fitur tambahan seperti *hosting*. Implementasi aplikasi iPresence pada *database* dijelaskan sebagai berikut.

4.1.3.1 User

Database User memuat identitas seluruh *user* yang terdaftar. *Database* tersebut bisa diisi langsung di *database* atau melalui mode admin. Saat *SplashActivity* dan *LoginActivity* dipanggil, maka program akan menjalankan perintah untuk *check session* pada *database* User. Adapun variabel yang terdapat pada *database* User adalah sebagai berikut.

Tabel 4.2 Database User

No.	Nama	Variable	Tipe Data
1	Object ID	objectId	string
2	Email	email	string
3	Password	password	string
4	Nama Awal	firstname	string
5	Nama Akhir	lastname	string
6	Tanggal Dibuat	createdAt	date
7	Tanggal <i>Update</i>	updatedAt	date
8	Jurusan	jurusan	string
9	Mata Kuliah ke-n	matkul(n)	string
10	Status	status	string
11	NIM/NIK	userId	string

Check session yang dilakukan adalah melakukan pengecekan terhadap *database User*, apakah *user* terdaftar atau tidak. Variabel yang memuat *check session* adalah email sebagai *username*, *password*, dan status *user*. Jika ketiganya sudah memenuhi sesuai dengan program, maka *check session* akan mengarahkan ke halaman utama mode masing-masing. Jika tidak maka terdapat 2 kemungkinan, yaitu karena salah satu variabel tidak sesuai atau karena *user* tidak terdaftar di *database User*.

Selain melakukan *check session*, perintah untuk menampilkan daftar mata kuliah mahasiswa diambil juga dari variabel *database User*. *Database User* memiliki variabel *matkul(n)*, yaitu *matkul1* sampai *matkul10* yang diisi oleh kode mata kuliah. Kode mata kuliah tersebut yang nantinya akan dijadikan sebuah *statement* untuk memunculkan identitas mata kuliah di halaman utama. Adapun cuplikan kodenya seperti pada gambar 4.31.

```

package com.umy.ProjectnyaAdit.All;

import android.app.Application;
import com.eyro.cubeacon.*;
import com.eyro.mesosfer.Mesosfer;
public class MesosferApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // set Cubeacon SDK log level to verbose mode
        com.eyro.cubeacon.Logger.setLogLevel(LogLevel.VERBOSE);

        // enable background power saver to save battery life up to 60%
        Cubeacon.setBackgroundPowerSaver(true);
        Cubeacon.initialize(this);
        Mesosfer.setPushNotification(true);

        // initialize Mesosfer SDK
        Mesosfer.initialize(this, "MrKJXn89XH", "v6LS5MpMsqbIQ1Qbj5vUy18cnKDVK9PG")
    }
}

```

Gambar 4.32 Cuplikan kode MesosferApp.java

Selain menjadi *statement* kondisi untuk menampilkan mata kuliah, variabel pada *database* User juga menjadi *statement* kondisi untuk melakukan presensi. Tabel presensi memuat nama mahasiswa, NIM, kelas, dan pertemuan yang dihadiri. Untuk menentukan *user* mana yang melakukan presensi, maka diperlukan sebuah kondisi untuk melakukan filtrasi sehingga mengerucut dengan 1 data, yaitu data *user* tersebut. Berikut cuplikan kodenya.

```

MesosferQuery<MesosferData> setDosen = MesosferData.getQuery("PresensiDosen");
setDosen.whereEqualTo("matkulNama", matkulNama);
setDosen.whereEqualTo("userPengampu", userId);
setDosen.countAsync(new CountCallback() {
    @Override
    public void done(int i, MesosferException e) {
        if(i==0){
            new SweetAlertDialog(CheckInActivity.this, SweetAlertDialog.ERROR_TYPE)
                .setTitleText("Error!")
                .setContentText("Anda tidak mengampu matkul tersebut!")
                .setConfirmClickListener(new SweetAlertDialog.OnSweetClickListener() {
                    @Override
                    public void onClick(SweetAlertDialog sweetAlertDialog) {
                        Intent intent = new Intent(CheckInActivity.this, SplashActivity.class);
                        startActivity(intent);
                        recreate();

                        finishAffinity();
                    }
                })
                .show();
        }
        else {
            masukdosen();
        }
    }
});
sendLog(region, MesosferBeacon.Event.ENTER);

```

Gambar 4.33 Cuplikan kode PresensiActivity.java

Gambar 4.31 merupakan cuplikan kode untuk menyaring data dosen. Filter yang dilakukan adalah menghitung data dengan kondisi userPengampu sama dengan NIK yang teridentifikasi dari *database* User. Filter ini sendiri dijalankan saat dosen memilih mata kuliah. Perintah selanjutnya adalah mencocokkan apakah mata kuliah yang ingin diaktifkan dosen merupakan mata kuliah yang diampu atau tidak. Hasil penyaringannya berupa nilai jumlah kecocokan. Nilai tersebut hanya ada dua kemungkinan, 0 atau 1. Nilai 0 berarti mata kuliah yang dipilih bukan diampu oleh dosen tersebut. Output yang keluar pun adalah sebuah *dialog* bahwa dosen tidak mengampu mata kuliah tersebut. Nilai 1 berarti ada data yang sesuai dan menunjukkan bahwa dosen tersebut memang mengampu mata kuliah yang dipilih, lalu selanjutnya akan masuk ke perintah `masukdosen()` untuk proses input ke *database*.

4.1.3.2 Installation

Database selanjutnya yang terdapat pada mesosfer adalah *database* Installation. *Database* Installation memuat identitas semua *device* yang telah memasang aplikasi iPresence. Berikut tabel variabelnya.

Tabel 4.3 List Variabel Database Installation

No.	Nama	Variable	Tipe Data
1	Object ID	objectId	string
2	Object ID User	user	pointer
3	Device Brand	deviceManufacturer	string
4	Nama Bundle Project	appIdentifier	string
5	Nama Aplikasi	appName	string
6	Model Device	deviceModel	string
7	Identitas Instalasi	installationId	string
8	Zona Waktu	timeZone	string
9	Seri Android	deviceOsVersion	string
10	Seri Aplikasi	appVersion	string

Fungsi dari *database* Installation adalah untuk mengidentifikasi *device* yang dipakai oleh *user*. Variabel *user* pada *database* Installation mendefinisikan *objectId* yang terdapat di *database* User. Dengan demikian, dapat diketahui identitas *device* setiap *user*. Berikut cuplikan data yang diperoleh dari *database* Installation.

Tabel 4.4 Cuplikan Database Installation

objectId	user	deviceManufacturer	appIdentifier	appName	deviceModel	installationId	timeZone	deviceOsVersion	appVersion
p621yophQa	(undefined)	samsung	com.eyro.umy.i	iPresence	SM-J200G	d64c4319-c8c8...	Asia/Jakarta	5.1.1	1.0
e80YGFd1d2	5q6f02H8p	Xiaomi	com.iot.umy.i	iPresence	Redmi 4K	c6f77d04-2fba...	Asia/Jakarta	6.0.1	2.1
W60mDk05H	fkYLIbq5e	OPPO	com.iot.umy.i	iPresence	A1601	e908147a-b233...	Asia/Jakarta	5.1	2.1
a2QC21vEN7	z0c3e7zxv	samsung	com.iot.umy.i	iPresence	SM-G355H	1500a693-8cbe...	Asia/Jakarta	4.4.2	2.1
qeqw5aa7RA	H0Yw82qVq	Xiaomi	com.iot.umy.i	iPresence	Mi-4c	3a891b18-3914...	Asia/Jakarta	5.1.1	2.1
qb0UvdYXP	LTe1aWVR2G	LENOVO	com.iot.umy.i	iPresence	Lenovo P1m40	99b068de-6e53...	Asia/Jakarta	5.1	2.1
YnJaZDccQY	(undefined)	asus	com.iot.umy.i	iPresence	ASUS_Z00AD	a2166be1-70d2...	Asia/Jakarta	5.0	2.1
DB0vYX2bSY	Xa2XUg9pR	OPPO	com.iot.umy.i	iPresence	A1601	e5784206-3c2c...	Asia/Jakarta	5.1	2.1
EF7r1VFGIL	EA4qv9x7yM	Xiaomi	com.iot.umy.i	iPresence	Redmi Note 4	3bb2b137-7b47...	Asia/Jakarta	7.0	2.1
nWgDbaeT00	UaMe2ae10J	LGE	com.iot.umy.i	iPresence	LG-R945	17247409-c73a...	Asia/Jakarta	7.0	2.1
UFZV4lCOOG	eq8LW4E6H	Xiaomi	com.iot.umy.i	iPresence	Redmi 3	a8bf6d4f-56f8...	Asia/Jakarta	5.1.1	2.1
Qc2pTcAF5m	NCwEFK0Wp	Sony	com.iot.umy.i	iPresence	F5321	219e631a-e4c0...	Asia/Jakarta	8.0.0	2.1
B0N1rj0CJA	c7JREYRDE1	Xiaomi	com.iot.umy.i	iPresence	Redmi Note 4	b8ca783a-a497...	Asia/Jakarta	7.0	2.1
z1W1XULhCJ	MjFla9Kac8	Xiaomi	com.iot.umy.i	iPresence	Redmi 3	54c54048-116d...	Asia/Jakarta	5.1.1	2.1
SRYHAMvriP1	t0Dv0Bm8cd	Xiaomi	com.iot.umy.i	iPresence	Redmi 5A	3543b330-ee29...	Asia/Jakarta	7.1.2	2.1
89H4Sfbq2G	PqNa0kqGU	samsung	com.iot.umy.i	iPresence	SM-T285	a482c6f8-4b57...	Asia/Jakarta	5.1.1	2.1

Object Id pertama dan yang ke tujuh tidak memiliki identitas *user*. Hal tersebut disebabkan *user* tersebut berhasil melakukan instalasi pada *device*, namun tidak login atau tidak berhasil login, sehingga tidak teridentifikasi di *database* User. Jika *user* terdaftar dan berhasil login, maka pada kolom *user* akan terisi dengan *objectId* dari *database* User.

4.1.3.3 Bucket

Database Bucket adalah *database* yang memuat seluruh data tabel selain data *user* dan data notifikasi. *Database* Bucket merupakan fitur Mesosfer yang memungkinkan *developer* dapat membuat tabel sesuai kebutuhan. Aplikasi *iPresence* memiliki 13 data, memuat data mata kuliah, ruang kuliah, serta data presensi setiap mata kuliah. Setiap data memiliki huruf awal kapital, lalu antara kata pertama dan kata kedua tidak dipisahkan dengan spasi. Berikut adalah uraian data yang terdapat pada Bucket.

1. MataKuliah

Bucket MataKuliah memuat semua data mata kuliah yang memiliki variabel sebagai berikut.

Tabel 4.5 List Variabel Data MataKuliah

No.	Nama	Variable	Tipe Data
1	Nama Matkul	matkulNama	string
2	Nama Matkul Get	matkulNamaGet	string
3	Kode Matkul	matkulKode	string
4	Kelas	kelas	string
5	Hari	hari	string
6	Pukul	jam	string
7	Ruang Kuliah	ruang	string
8	Ruang Text	ruangText	string
9	Pengampu Mata Kuliah	matkulPengampu	string
10	NIK Pengampu	matkulPengampuId	string
11	Mahasiswa	daftarMahasiswa	string

Variabel `matkulNama` merupakan nama dari mata kuliah. Variabel `matkulNamaGet` juga hampir mirip dengan `matkulNama`, namun `matkulNamaGet` merupakan variabel untuk memanggil data lain pada *database* bucket, sehingga kata pertama dan kedua variabel `matkulNamaGet` tidak dihubungkan dengan spasi.

Selain itu, terdapat variabel `daftarMahasiswa` yang merupakan daftar mahasiswa yang mengambil mata kuliah tersebut. Data pada variabel tersebut memiliki format JSON yang memuat baris dan kolom, lalu kemudian di *parsing* untuk dimasukkan ke tabel presensi mata kuliah.

2. RuangKuliah

Bucket RuangKuliah adalah data yang memuat identitas ruang yang dipakai sebagai tempat perkuliahan, seperti nama ruang, lokasi, fakultas, dan lain-lain. Selain itu, terdapat satu identitas yang merupakan identitas beacon sebagai representasi dari identitas tiap koridor. Adapun identitas tersebut berupa nilai major dan minor. Berikut adalah variabel yang terdapat pada bucket RuangKuliah.

Tabel 4.6 List Variabel Data RuangKuliah

No.	Nama	Variable	Tipe Data
1	Nama Ruang Text	ruangText	string
2	Nama Ruang	namaRuang	string
3	Lokasi	lokasi	string
4	Fakultas	fakultas	string
5	Major	major	string
6	Minor	minor	string

Variabel pertama adalah Nama Ruang Text dengan nama variabel ruangText. Variabel tersebut berfungsi sebagai nilai yang akan ditampilkan untuk *user*. Adapun entitas namaRuang berfungsi sebagai nilai untuk eksekusi pada program. Perbedaannya variabel ruangText memiliki spasi sebagai pemisah antar huruf, sehingga memudahkan *user* untuk melihat tampilan, sedangkan variabel namaRuang tidak terdapat spasi di dalamnya untuk memudahkan eksekusi di pemrograman.

Variabel lain adalah major dan minor yang berfungsi untuk mengintegrasikan bucket RuangKuliah dengan *database* Beacon yang terdapat di luar *database* Bucket.

3. Bucket Presensi Setiap Mata Kuliah

Nama bucket presensi setiap mata kuliah sama seperti nama variabel `matkulNamaGet` pada bucket `MataKuliah`. Saat program memanggil `matkulNamaGet`, maka selanjutnya akan dieksekusi bucket presensi dengan nama yang sama dengan `matkulNamaGet` tersebut. Bucket ini ditambahkan sebanyak mata kuliah yang ada.

Data di dalam bucket Presensi memuat variabel sebagai berikut.

Tabel 4.7 List Variabel Data Presensi Setiap Mata Kuliah

No.	Nama	Variable	Tipe Data
1	Nama Mahasiswa	<code>userNama</code>	string
2	NIM	<code>userNim</code>	string
3	Kelas	<code>kelas</code>	string
4	Pertemuan(n)	<code>Pertemuan(n)</code>	string
5	Akurasi	<code>major</code>	string
6	Jarak	<code>minor</code>	string
7	TxPower	<code>txpower</code>	string
8	RSSI	<code>rss</code>	string

Data pada bucket setiap presensi memiliki 4 variabel sebagai parameter pengujian, yaitu akurasi yang memiliki output jarak antara *device* ke beacon saat melakukan presensi (dalam meter), jarak yang merupakan *range* default dari beacon, `txpower` yaitu daya transmisi yang dipancarkan oleh beacon (dBm), dan yang terakhir adalah `rss` yaitu indikator nilai daya yang diterima oleh *device* (dBm).

4. PresensiDosen

PresensiDosen merupakan bucket yang di dalamnya terdapat data presensi yang dilakukan oleh dosen. Sebelumnya telah dijelaskan bahwa presensi dosen adalah untuk mengaktifkan perkuliahan pada satu pertemuan. Bucket ini yang berperan untuk melakukan *record* presensi dosen untuk pertemuan ke-n, sehingga saat mahasiswa melakukan presensi, pertemuan yang akan dipresensikan adalah pertemuan yang sesuai dengan pertemuan ke-n pada bucket PresensiDosen. Selain itu, terdapat juga kode masuk yang otomatis terisi saat dosen melakukan presensi. Berikut variabel-variabel yang terdapat pada bucket PresensiDosen.

Tabel 4.8 List Variabel Data PresensiDosen

No.	Nama	Variable	Tipe Data
1	Nama Matkul	matkulNama	string
2	Kode Matkul	matkulKode	string
3	Jumlah Pertemuan	Jumlah	string
4	NIK Pengampu	userPengampu	string
5	Tanggal Pertemuan (n)	tglPertemuan(n)	string
6	Kode Masuk	kodemasuk	string

Saat dosen melakukan presensi, maka program akan melakukan pengecekan pada variabel NIK pengampu pada bucket PresensiDosen. Hal tersebut bertujuan untuk memastikan bahwa *user* dosen adalah pengampu dari mata kuliah yang dipilih. Jika pada *database* tidak ditemukan kesesuaian antara *matkulNama*, *matkulKode*, dan *userPengampu*, maka dosen tidak bisa melakukan presensi.

5. DataMatkul

Pengisian data pada bucket MataKuliah tentunya tidak bisa hanya dengan memasukkan data satu persatu. Harus ada juga fitur untuk memasukkan data mata kuliah secara majemuk. DataMatkul merupakan bucket yang berfungsi sebagai penyimpan *list* mata kuliah dalam bentuk JSON. Data JSON tersebut selanjutnya akan di *parsing* oleh aplikasi sehingga terurai dalam bentuk kolom dan baris pada bucket MataKuliah. Variabel pada DataMatkul hanya terdapat kolom dengan nama data berupa tipe data string yang berisi data JSON tersebut.

6. DataMahasiswa

Bucket DataMahasiswa berisi data lengkap mahasiswa dalam bentuk JSON. Data JSON tersebut akan di *parsing* untuk mengisi *database* User. Data ini dikategorikan menurut angkatan yang terbagi menjadi 2. Contohnya pada angkatan 2014, maka terdapat 2014a dan 2014b. Hal tersebut dilakukan karena mesosfer hanya mampu menerima data sebanyak 100 *user* pada setiap *parsing*. Berikut variabel yang terdapat pada bucket DataMahasiswa.

Tabel 4.9 List Variabel Data DataMahasiswa

No.	Nama	Variable	Tipe Data
1	Kelompok Angkatan	angkatan	string
2	Data Mahasiswa	data	string

7. HasilPresensi

Setelah presensi berhasil diinputkan di *database*, maka yang selanjutnya adalah mengkonversi *database* menjadi sebuah data yang dapat digunakan oleh berbagai *platform*. JSON sebagai bahasa pemrograman mampu dibaca oleh

platform lain. Bucket HasilPresensi berfungsi untuk menyimpan *database* pada setiap presensi mata kuliah menjadi bentuk format JSON. Adapun prosesnya dieksekusi oleh aplikasi iPresence pada *device* android. Berikut variabel yang terdapat pada bucket HasilPresensi.

Tabel 4.10 List Variabel Data HasilPresensi

No.	Nama	Variable	Tipe Data
1	Kelompok Angkatan	angkatan	string
2	Data Mahasiswa	data	string

4.1.3.4 Beacon

Database Beacon merupakan *database* yang terintegrasi dengan bucket MataKuliah melalui nilai major dan minor. *Database* Beacon menyimpan identitas beacon yang memiliki variabel sebagai berikut.

Tabel 4.11 List Variabel Data Beacon

No.	Nama	Variable	Tipe Data
1	Nama	name	string
2	Major	major	number
3	Minor	minor	number
4	UUID	uuid	string

Variabel name merepresentasikan nama koridor yang terdapat pada setiap gedung. Major dan minor merupakan dua identitas yang dapat dikonfigurasi sesuai dengan kebutuhan. Nilai major 2 cobebacon boleh saja sama, tapi minornya yang harus berbeda. Begitupun sebaliknya. Major dan minor antar beacon pun boleh berbeda. Semua

tergantung bagaimana merencanakan identitas beacon tersebut. Nilai UUID merupakan *standard identifier* unik untuk sebuah *device*.

4.1.3.5 Presence

Database Presence memuat data yang berisi aktivitas presensi yang dilakukan oleh semua *user*. *User* dapat diketahui dari beacon mana *user* presensi, waktu, serta *event* yang dilakukan (saat *device* melakukan presensi). Berikut variabel yang terdapat pada *database* Presence.

Tabel 4.12 List Variabel Data Presence

No.	Nama	Variable	Tipe Data
1	Object ID Beacon	beaconId	pointer
2	Object ID User	userId	pointer
3	Event	event	eventBeacon
4	Installation ID	installationId	pointer

Object ID Beacon berasal dari *objectId* yang terdapat pada *database* Beacon, sedangkan Object ID User berasal dari variabel *objectId* pada *database* User. Event merupakan kondisi *device* melakukan presensi. Lalu *InstallationId* yang merupakan *objectId* dari *database* Installation.

4.1.3.6 Storyline

Database Storyline berperan untuk menyimpan data notifikasi. Saat dosen mengirimkan notifikasi kepada mahasiswa, maka data notifikasi tersebut akan tersimpan di *database* Storyline. Lalu saat notifikasi ditampilkan di aplikasi mode mahasiswa, maka beacon akan memanggil *database* Storyline sesuai dengan identitas beaconnya. Berikut adalah variabel yang terdapat pada *Database* Storyline.

Tabel 4.13 List Variabel Data Storyline

No.	Nama	Variable	Type Data
1	Jenis Notifikasi	campaign	pointer
2	Judul Notifikasi	alertTitle	pointer
3	Isi Notifikasi	alertMessage	string
4	Event	event	eventBeacon
5	Beacon	beacons	array

Variabel *campaign* merupakan jenis notifikasi yang akan digunakan. Aplikasi *iPresence* menggunakan *campaign* jenis *TEXT*, yaitu dengan pesan *text*. Saat jenis *TEXT* yang dipilih, maka aplikasi akan mengambil dua variabel, yaitu *alertTitle* sebagai judul notifikasi dan *alertMessage* sebagai isi dari notifikasi. *Event* merupakan kondisi *device* pada saat menerima notifikasi tersebut. *iPresence* hanya menggunakan *Event ENTER* atau masuk saja saat *user* menerima notifikasi, sedangkan saat *user* keluar dari area *beacon*, maka notifikasi tidak perlu muncul lagi. Lalu ada variabel *beacons* yang berfungsi untuk mengarahkan *beacon* mana saja yang dapat mengirimkan notifikasi.

4.2 Pengujian

4.2.1 Spesifikasi Perangkat

Pengujian aplikasi *iPresence* menggunakan 17 perangkat dengan spesifikasi sebagai berikut.

Tabel 4.14 Spesifikasi *device* yang digunakan

Nama Device	Model Device	Versi Android	Type Device	Prosesor	RAM (GB)
Asus	ASUS_Z00AD	5.0.0	Mobile	Octa-core 1.8 GHz	4
Himax	M23	7.0.0	Mobile	Quad-core 1.25 GHz	3
Lenovo	P1ma40	5.1.0	Mobile	Quad-core 1.0 GHz	2

Nama Device	Model Device	Versi Android	Tipe Device	Prosesor	RAM (GB)
LGE	LG-H845	7.0.0	Mobile	Octa-core 1.8 GHz	3
Oppo	A1601	5.1.0	Mobile	Hexa-core 1.5 GHz	3
Oppo	A1601	5.1.0	Mobile	Hexa-core 1.5 GHz	3
Samsung	SM-G355H	4.4.2	Mobile	Quad-core 1.2 GHz	0.77
Samsung	SM-T285	5.1.1	Tablet	1.5 GHz	1.5
Sony	F5321	8.0.0	Mobile	Hexa-core 1.4 GHz	3
Xiaomi	Redmi 3	5.1.1	Mobile	Octa-core 1.2 GHz	2
Xiaomi	Redmi 3	5.1.1	Mobile	Octa-core 1.2 GHz	2
Xiaomi	2014817	5.1.1	Mobile	1.2 GHz	1
Xiaomi	Mi-4c	5.1.1	Mobile	Hexa-core 1.4 GHz	2
Xiaomi	Redmi 4X	6.0.1	Mobile	Octa-core 1.4 GHz	4
Xiaomi	Redmi Note 4	7.0.0	Mobile	Octa-core 2.0 GHz	3
Xiaomi	Redmi Note 4	7.0.0	Mobile	Octa-core 2.0 GHz	3
Xiaomi	Redmi 5A	7.1.2	Mobile	Quad-core 1.4 GHz	3

4.2.2 Pengujian dan Analisis

Pengujian aplikasi meliputi dua percobaan. Percobaan pertama yaitu uji fungsionalitas yang bertujuan untuk mengetahui keberhasilan *device* dalam mengolah aplikasi. Percobaan kedua adalah pengujian jarak dan daya yang bertujuan untuk mengetahui daya tangkap *device* terhadap beacon beserta jarak maksimal beacon dapat memancarkan sinyal sampai ke *device*. Percobaan kedua yaitu.

4.2.2.1 Pengujian Fungsionalitas

Uji fungsionalitas adalah uji aplikasi yang dilakukan di banyak *device* untuk mengetahui tingkat keberhasilan sebuah *device* untuk

menginstal, menjalankan, hingga melakukan presensi pada aplikasi iPresence. Uji fungsionalitas dilakukan pada perkuliahan Sistem Operasi dan Jaringan Telekomunikasi. Jumlah *device user* yang diujikan berjumlah 16 *device*, 12 *device* pada mata kuliah Sistem Operasi dan 6 *device* pada mata kuliah Jaringan Telekomunikasi (2 *device* dimiliki oleh *user* yang mengambil kedua mata kuliah). Berikut hasil pengujian fungsional aplikasi.

Tabel 4.15 Hasil Pengujian Fungsionalitas Mata Kuliah Sistem Operasi

No.	Nama	NIM	Kelas	pertemuan1
1	Arifah Yuliani	20150120125	A	1
2	Muhammad Hafiz Aldy	20150120069	A	1
3	Muhammad Fachri	20150120056	A	1
4	Fazal Hawari	20150120095	A	1
5	Doane Puri Mustika	20150120163	A	1
6	Bangkit Dwiputra P.	20150120120	A	1
7	Kharisma Fajar Sidik	20150120152	A	1
8	Adnan Prayudha	20150120051	A	1
9	Ayu Al Israh	20150120076	A	1
10	Tri Handayani Putri	20150120043	A	1
11	Ratna Murti	20150120114	A	1
12	Purwoko Nurhadi	20150120052	A	-

Dari 12 *device* yang diuji, terdapat 1 *device* yang berhasil melakukan instalasi, namun tidak dapat mendeteksi beacon, sehingga ruang kuliah tidak muncul dalam opsi. Akibatnya, aplikasi tidak bisa melakukan presensi. Tapi selain 1 *device* tersebut, semua *device* dapat melakukan instalasi, menjalankan, hingga melakukan presensi dengan sukses. Keberhasilan tersebut terlihat dari nilai “1” pada kolom pertemuan1. Hal tersebut menunjukkan bahwa data mahasiswa yang presensi berhasil diinputkan pada *database*.

Tabel 4.16 Hasil Pengujian Fungsionalitas Mata Kuliah Jaringan Telekomunikasi

No.	Nama	NIM	Kelas	pertemuan1
1	Doane Puri Mustika	20150120163	A	1
2	Nurohman Fadilah	20150120081	A	1
3	Dimas Agung Nugroho	20080120002	A	1
4	Muhammad Wilimilio Rizkidana	20150120158	A	1
5	Kharisma Fajar Sidik	20150120152	A	1
6	Fatkhurrohman	20150120096	A	1

4.2.2.2 Pengujian Jarak dan Daya

Pengujian jarak dan daya dilakukan di dua *event* yang berbeda, yaitu saat menggunakan satu *device* dan saat menggunakan banyak *device*. Saat menggunakan satu *device*, pengujian dilakukan dengan 2 kondisi, yaitu tanpa penghalang dan dengan penghalang.

1. Pengujian Daya Transmisi terhadap Jarak

Pengujian yang dilakukan adalah mengukur jarak maksimal beacon memancarkan daya pada *device*, serta perubahan jarak maksimal jika *transmission power* (tx power) diubah juga. Pengujian dilakukan di laboratorium Teknik Elektro Universitas Muhammadiyah Yogyakarta. *Device* yang digunakan pada pengujian ini adalah samsung seri SM-J320G dengan OS Android versi 5.1.1. Pengujian dilakukan dengan indikator penghalang dan tanpa penghalang. Setiap percobaan tersebut akan diubah nilai *transmission powernya*, sehingga dapat dilihat karakteristik pengaruh jarak terhadap daya pancar beacon. Pengujian tanpa penghalang dilakukan di koridor. Beacon diposisikan di ujung laboratorium. Adapun saat pengujian dengan penghalang beacon diletakkan di dalam ruangan, lalu *device* diposisikan di luar. Berikut hasil pengujian pertama.

Tabel 4.17 Hasil Pengujian Daya Transmisi terhadap Jarak

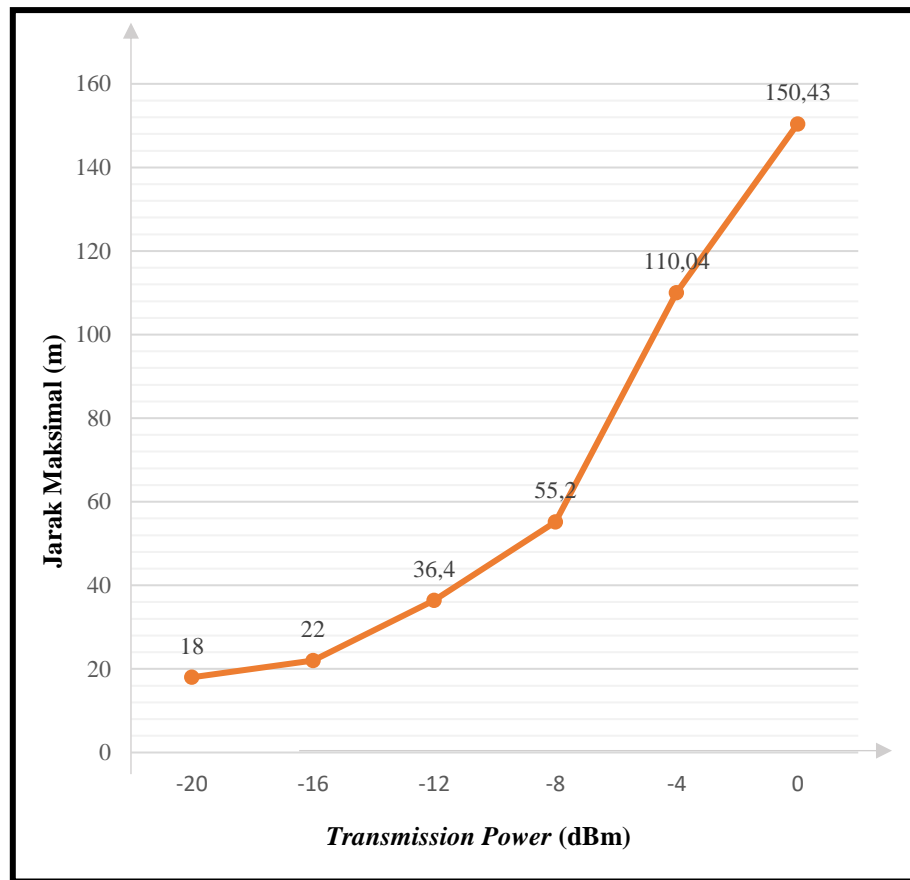
No	Tx Power (dBm)	Tx Power (mW)	Jarak Max Tanpa Penghalang (m)	Jarak Max dengan Penghalang (m)
1	-20	0.010	18	3.25
2	-16	0.025	22	5
3	-12	0.063	36.4	8.93
4	-8	0.158	55.2	14.39
5	-4	0.398	110.04	26
6	0	1.000	150.43	53.41

Pengujian dimulai dengan *set transmission power* pada nilai -20 dBm. Jika dikonversi menjadi mili Watt (mW), nilainya adalah 0.01 mW. Jarak terjauh yang dapat dijangkau oleh android adalah sejauh 15 meter tanpa penghalang, sedangkan dengan penghalang *device* dapat menerima sinyal beacon sampai 5 meter.

Pengujian selanjutnya adalah mengubah *transmission power* menjadi lebih besar, yaitu -16 dBm atau 0.025 mW. Setelah dilakukan pengukuran jarak, ternyata jarak yang dijangkau lebih jauh, yaitu sejauh 20 meter tanpa penghalang dan 8 meter dengan penghalang.

Selanjutnya nilai *transmission power* semakin dinaikkan menjadi -12 dBm atau 0.063 sampai 0 dBm atau 1mW. Hasilnya jarak jangkauan beacon semakin jauh, hingga mampu mencapai 80 meter tanpa penghalang dan 25 meter dengan penghalang.

Pengujian ini disimpulkan bahwa semakin besar nilai *transmission power*, maka jarak jangkauan beacon menuju *device* akan semakin jauh. Adapun bentuk *chart* dari pengujian tersebut terdapat pada gambar 4.32



Gambar 4.34 Pengaruh *Transmission Power* Beacon terhadap Jarak Jangkauan Beacon

2. Pengujian RSSI terhadap jarak

Pengujian RSSI terhadap jarak bertujuan untuk mengetahui pengaruh nilai *Received Signal Strength Indicator* atau yang disebut RSSI terhadap jarak. Pengujian dilakukan di perkuliahan dua mata kuliah yang sebelumnya dilakukan juga untuk menguji fungsionalitas aplikasi. Berikut hasil pengujian RSSI terhadap jarak.

Tabel 4.18 Hasil Pengujian RSSI terhadap Jarak pada Mata Kuliah Sistem Operasi

No.	NIM	Kelas	Jarak (m)	RSSI (dBm)	RSSI (mW)
1	20150120125	A	1.0068	-42	0.000063095734
2	20150120069	A	1.2471	-61	0.000000794328
3	20150120056	A	1.3337	-57	0.000001995262

No.	NIM	Kelas	Jarak (m)	RSSI (dBm)	RSSI (mW)
4	20150120095	A	1.5613	-64	0.000000398107
5	20150120008	A	2.0498	-74	0.000000039811
6	20150120163	A	2.6445	-77	0.000000019953
7	20150120120	A	3.0687	-81	0.000000007943
8	20150120152	A	3.5225	-83	0.000000005012
9	20150120051	A	4.1268	-93	0.000000000501
10	20150120076	A	4.2169	-95	0.000000000316
11	20150120043	A	5.0626	-101	0.000000000079
12	20150120114	A	5.7573	-98	0.000000000158

Data pengujian pengaruh jarak terhadap RSSI didapatkan dari hasil presensi. Saat mahasiswa berhasil melakukan presensi, maka aplikasi akan mengirimkan beberapa variabel yang merupakan hasil pengukuran yang dilakukan oleh aplikasi tersebut. Aplikasi iPresence memberikan *output* data jarak dan RSSI. Lalu nilai RSSI yang memiliki satuan dBm di konversi ke mili-Watt (mW). Adapun nilai daya pancar atau *transmission power* beacon adalah sebesar -12 dBm.

Pengujian didapatkan hasil bahwa terdapat pengaruh daya yang diterima oleh *device* android atau RSSI terhadap jarak. *User* yang diuji memiliki jarak 1-5 meter dari beacon. Nilai RSSI yang didapatkan oleh *user* pertama sampai terakhir cenderung mengecil. *User* pertama menghasilkan nilai RSSI sebesar -42 dBm saat berada pada jarak 1 meter. *User* terakhir memiliki nilai RSSI sebesar -98 dBm. *User* tersebut merupakan *user* dengan jarak terjauh yaitu 5.7 meter. Pengujian ini dapat dilihat bahwa semakin jauh jarak *device* android terhadap beacon, maka nilai RSSI yang diterima oleh *device* android

cenderung lebih kecil. Berikut adalah pengaruh jarak terhadap nilai RSSI dalam bentuk *chart*.



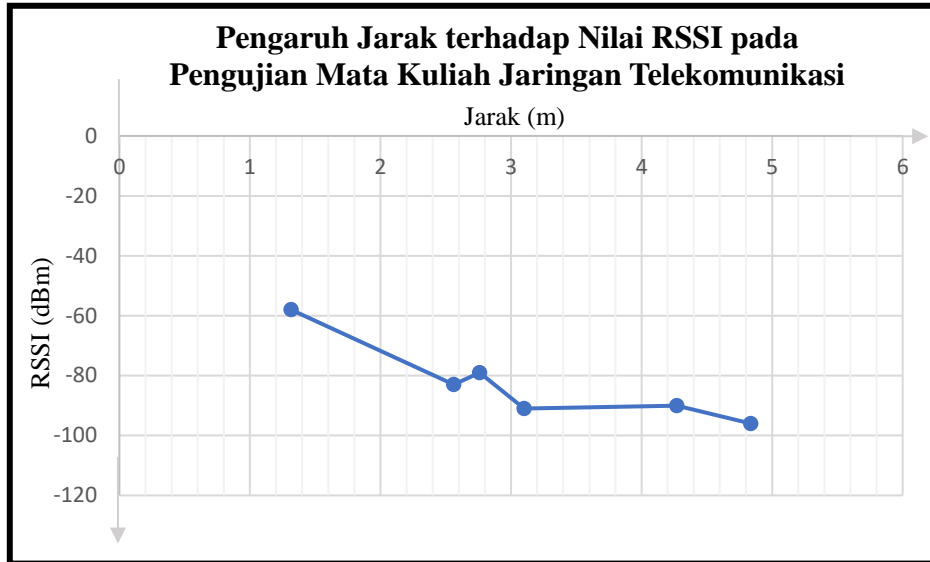
Gambar 4.35 Pengaruh Jarak terhadap Nilai RSSI pada Pengujian di Mata Kuliah Sistem Operasi

Tabel 4.19 Hasil Pengujian RSSI terhadap Jarak pada Mata Kuliah Jaringan Telekomunikasi

No.	NIM	Kelas	Jarak (m)	RSSI (dBm)	RSSI (mW)
1	20150120163	A	1,3147	-58	0.000063095734
2	20150120081	A	2,5601	-83	0.000000794328
3	20080120002	A	2,7610	-79	0.000001995262
4	20150120158	A	3,0996	-91	0.000000398107
5	20150120152	A	4,2709	-90	0.000000039811
6	20150120096	A	4,8378	-96	0.000000019953

Pengujian lain dilakukan di mata kuliah Jaringan Telekomunikasi. Dari 6 perangkat *user* yang diuji dengan jarak yang berbeda, terdapat perbedaan terhadap nilai RSSI untuk setiap *device*. Hasilnya pun sama dengan pengujian di mata kuliah Jaringan Telekomunikasi, dimana saat jarak *device* android semakin jauh dengan beacon, maka nilai RSSI yang diterima oleh *device* tersebut akan semakin kecil. Berikut adalah

bentuk *chart* dari hasil pengujian di mata kuliah Jaringan Telekomunikasi.



Gambar 4.36 Pengaruh Jarak terhadap Nilai RSSI pada Pengujian di Mata Kuliah Jaringan Telekomunikasi