

PERSONALISASI DAN NOTIFIKASI APLIKASI UBEACON

TUGAS AKHIR

Oleh

ASTARI PURNOMO

NIM : 13212037

Program Studi Teknik Elektro



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2016

PERSONALISASI DAN NOTIFIKASI APLIKASI UBEACON

Oleh:

Astari Purnomo

Tugas Akhir ini telah diterima dan disahkan
sebagai persyaratan untuk memperoleh gelar

SARJANA TEKNIK

di

**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Bandung, Juni 2016

Disetujui oleh:

Pembimbing I

Pembimbing II

Ir. Emir M. Husni M.Sc., Ph.D
NIP. 19670707 2006041 016

Andi Sama
CIO PT Sinergi Wahana Gemilang

ABSTRAK

PERSONALISASI DAN NOTIFIKASI APLIKASI UBEACON

Oleh:

Astari Purnomo

NIM : 13212037

PROGRAM STUDI TEKNIK ELEKTRO

Pertumbuhan jumlah mahasiswa di kampus berbanding lurus dengan banyaknya jumlah informasi yang harus tersebar di lingkungan kampus. Segala informasi tersebut berperan penting dalam meningkatkan kemampuan *hard skill* maupun *soft skill* mahasiswa. Namun seringkali banyaknya informasi yang tersebar tidak dapat dikontrol sehingga menimbulkan masalah baru. Banyaknya informasi yang harus didistribusikan tidak didukung dengan sistem penyebaran informasi yang sesuai sehingga terkadang tidak semua mahasiswa dapat memperoleh informasi yang seharusnya mereka peroleh secara lengkap.

uBeacon merupakan aplikasi berbasis android yang terintegrasi dengan teknologi iBeacon. Aplikasi ini menggunakan protokol komunikasi *Message Queuing Telemetry Transport* (MQTT). uBeacon memiliki beberapa fitur utama, yaitu *User Personalization*, *Push-Notification*, *Attend Class*, *My Location*, dan *Beacon Info*. Fitur *Push-Notification* dibuat agar mahasiswa dapat dengan mudah memperoleh informasi secara lengkap dan personal. Fitur *Push-Notification* terintegrasi dengan *web server*. Penggunaan fitur ini dilakukan dengan menjalankan aplikasi uBeacon pada *smartphone* mahasiswa serta terhubungnya *smartphone* pengguna dengan koneksi internet.

Kata kunci : Android, MQTT, *web server*

ABSTRACT

USER PERSONALIZATION AND NOTIFICATION FOR UBEACON

Oleh:

Astari Purnomo

NIM : 13212037

ELECTRICAL ENGINEERING

Growth in the number of students on campus is directly proportional with the amount of information that should be spread on campus. All such information plays an important role in enhancing the ability of hard skills and soft skills of the students. But often many scattered information can not be controlled, giving rise to new problems. The amount of information to be distributed is not supported by appropriate information dissemination system that sometimes not all students can get the information they are supposed to get complete.

uBeacon an android-based *application* that integrates with iBeacon technology. This *application* uses a communications protocol Message Queuing Telemetry Transport (MQTT). uBeacon has several *key* features, namely User Personalization, Push-Notification, Attend Class, My Location, and Beacon Info. Features Push-Notification is made so that students can easily obtain complete information and personal. Push-Notification feature is integrated with the web *server*. The use of this feature needs to be done by running the *application* on an user's *smartphone* as well as the connection of *smartphone* users with an Internet connection.

Key words: Android, MQTT, web *server*

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan karunia-Nya sehingga Tugas Akhir: *Internet of Things*- Personalisasi dan Promosi di Lingkungan Kampus dengan iBeacon dapat selesai dengan baik. Banyak pengetahuan, relasi, dan pembelajaran yang penulis peroleh selama pengerjaan Tugas Akhir ini. Penulis juga mengucapkan syukur atas selesainya Buku Tugas Akhir dengan judul Personalisasi dan Notifikasi Aplikasi uBeacon.

Penulis memohon maaf kepada pihak yang terkait bila terdapat kesalahan selama pengerjaan Tugas Akhir dan juga dalam penulisan laporan ini.

Berbagai motivasi, semangat dan pembelajaran penulis dapatkan selama pengerjaan Tugas Akhir juga penulisan laporan ini. Untuk itu penulis mengucapkan terimakasih terutama kepada :

1. Bapak Arief Sasongko, selaku Ketua Program Studi Teknik Elektro ITB.
2. Bapak Emir Mauludi Husni dan Bapak Andi Sama, selaku Dosen Pembimbing selama pengerjaan Tugas Akhir yang selalu memberikan pembelajaran, bimbingan dan masukan kepada penulis.
3. Bapak, Ibu, Sita, Kakung, Ester, Bulek, dan seluruh keluarga besar yang selalu memberikan doa dan dukungan serta semangat secara moral dan materiil sehingga penulis dapat menyelesaikan tugas akhir dan penyusunan laporan ini dengan semangat.
4. Rizky Indra Syafrian dan Adirga Ibrahim Khairy selaku teman seperjuangan dalam tim dari Tugas Akhir ini yang selalu dengan semangat dan kerja keras dalam suka maupun duka, memberikan yang terbaik selama keberjalanan Tugas Akhir, juga memberikan dukungan dan semangat secara moral kepada penulis sehingga dapat menyelesaikan Tugas Akhir dan laporan ini dengan semangat dan pikiran yang positif.
5. Samuel Adelwin, Irena Yosephine, Vivi Novia, Riksa Andanaswari, Hilmy Aziz, Syaiful Andy, Meynard Danam, Adil Aldianto, dan Afdhal Hanif selaku anak bimbingan Bapak Emir yang bersama-sama saling memberikan dukungan kepada penulis selama pengerjaan Tugas Akhir ini.
6. Teman-teman goa depan dan goa belakang selaku teman seperjuangan Teknik Elektro 2012 yang selalu memberikan keceriaan, cerita penghibur, dan menemani keseharian penulis.

7. Gilang Julian selaku teman penulis yang bersedia memberikan waktu dan membantu penulis saat mengalami kesulitan selama pengerjaan Tugas Akhir.
8. Keluarga besar Teknik Elektro ITB yang telah memberikan bimbingan dan pertolongan kepada penulis selama menjalani Tugas Akhir ini.
9. Teman-teman Teknik Elektro ITB.
10. Seluruh civitas akademika Teknik Elektro, Sekolah Teknik Elektro dan Informatika ITB.
11. Serta seluruh pihak yang telah membantu penulis dalam pelaksanaan Tugas Akhir serta penyusunan laporan ini yang tidak dapat disebutkan satu persatu. Terima kasih banyak.

Penulis menyadari bahwasanya tidak ada hal yang sempurna di dunia ini, seperti pepatah berkata “tidak ada gading yang tak retak” sama halnya dengan laporan ini yang pasti memiliki kekurangan. Penulis menerima dengan lapang dada seluruh kritik dan saran pembaca. Agar kemudian laporan Tugas Akhir ini dapat menjadi lebih baik lagi dan dapat memberi manfaat bagi para pembacanya.

Bandung, Juni 2016

Penulis

DAFTAR ISI

PERSONALISASI DAN NOTIFIKASI APLIKASI UBEACON	i
LEMBAR PENGESAHAN	ii
ABSTRAK.....	iii
ABSTRACT.....	iv
KATA PENGANTAR	v
DAFTAR ISI.....	vii
DAFTAR GAMBAR	ix
DAFTAR TABEL.....	xi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan dan Materi.....	2
1.4 Lingkup Permasalahan.....	2
1.5 Metodologi.....	2
1.6 Sistematika Penulis	3
BAB II TINJAUAN PUSTAKA	5
2.1 Android Operating System.....	5
2.2 Java	5
2.3 Message Queuing Telemetry Transport (MQTT).....	6
2.4 Graphical User Interface (GUI)	6
2.5 JSON file.....	7
BAB III DESAIN DAN IMPLEMENTASI	8
3.1 Rancangan Umum.....	8
3.2 Spesifikasi	11
3.3 Desain	11
3.4 Perancangan Graphical User Interface (GUI)	12
3.5 Implementasi Aplikasi	13
3.5.1 Graphical User Intergace (GUI) uBeacon.....	13
3.5.2 Koneksi MQTT	22
3.5.3 Fitur User Personalization.....	23
3.5.3.1 Sign-Up.....	24
3.5.3.2 Sign-In.....	26

3.5.3.3	Edit Profile	29
3.5.4	Fitur Push-Notification	32
3.5.5	Penyimpanan pada local memory smartphone.....	39
BAB IV HASIL PENGUJIAN		44
4.1	Aspek dan Skenario Pengujian	44
4.2	Prosedur dan Hasil Pengujian	45
4.2.1	Fitur User Personalization.....	45
4.2.1.1	Sign-Up	45
4.2.1.2	Sign-In.....	47
4.2.1.3	Edit Profile	50
4.2.1.4	Logout	51
4.2.1.5	SharedPreference	51
4.2.2	Fitur Push-Notification	54
4.2.2.1	Service.....	54
4.2.2.2	Notification	56
BAB V KESIMPULAN DAN SARAN.....		57
5.1	Kesimpulan	57
5.2	Saran	57
DAFTAR PUSTAKA		58

DAFTAR GAMBAR

Gambar 1 Konfigurasi kerja protokol MQTT. ©Dokumentasi Penulis	6
Gambar 2 Format JSON <i>file</i> . ©Dokumentasi Penulis.....	7
Gambar 3 Diagram rancangan umum sistem uBeacon. ©Dokumentasi Penulis	8
Gambar 4 Diagram blok sub-sistem aplikasi uBeacon. ©Dokumentasi Penulis	9
Gambar 5 Flowchart alir aplikasi uBeacon. ©Dokumentasi Penulis	10
Gambar 6 Diagram Blok aplikasi uBeacon. ©Dokumentasi Penulis.....	12
Gambar 7 Ilustrasi Graphical User Interface untuk aplikasi uBeacon. ©Dokumentasi Penulis ...	13
Gambar 8 Tampilan antarmuka aplikasi uBeacon. ©Dokumentasi Penulis	14
Gambar 9 Flowchart Sign-Up aplikasi uBeacon. ©Dokumentasi Penulis.....	24
Gambar 10 Flowchart Sign-In aplikasi uBeacon. ©Dokumentasi Penulis	26
Gambar 12 Tampilan antarmuka untuk Edit <i>Profile</i> aplikasi uBeacon. ©Dokumentasi Penulis..	29
Gambar 13 Flowchart push-notification aplikasi uBeacon. ©Dokumentasi Penulis	32
Gambar 14 Ilustrasi push-notification aplikasi uBeacon. ©Dokumentasi Penulis	32
Gambar 15 Flowchart Bound Service. ©Dokumentasi Penulis	34
Gambar 16 Tampilan antarmuka push-notification aplikasi uBeacon. ©Dokumentasi Penulis ...	39
Gambar 17 Tampilan pada navigation bar aplikasi uBeacon. ©Dokumentasi Penulis.....	42
Gambar 18 Halaman Sign-In aplikasi uBeacon. ©Dokumentasi Penulis	46
Gambar 19 Tampilan log pada bagian Sign-Up aplikasi uBeacon. ©Dokumentasi Penulis	46
Gambar 20 Tampilan kegagalan registrasi aplikasi uBeacon. ©Dokumentasi Penulis	47
Gambar 21 Tampilan Sign-In berhasil dilakukan di aplikasi uBeacon. ©Dokumentasi Penulis..	48
Gambar 22 Tampilan logcat saat Sign-In aplikasi uBeacon. ©Dokumentasi Penulis	48
Gambar 23 Navigation Bar aplikasi uBeacon. ©Dokumentasi Penulis	49
Gambar 24 Tampilan Sign-In yang gagal dilakukan pada aplikasi uBeacon. ©Dokumentasi Penulis.....	49
Gambar 25 Tampilan Edit <i>Profile</i> aplikasi uBeacon. ©Dokumentasi Penulis.....	50
Gambar 26 Edit <i>Profile</i> berhasil dilakukan aplikasi uBeacon. ©Dokumentasi Penulis.....	50
Gambar 27 Logout aplikasi uBeacon. ©Dokumentasi Penulis	51
Gambar 28 Tampilan awal aplikasi uBeacon. ©Dokumentasi Penulis.....	52
Gambar 29 Status awal Shared Preference aplikasi uBeacon. ©Dokumentasi Penulis	52
Gambar 30 Sign-In berhasil dilakukan. ©Dokumentasi Penulis	52
Gambar 31 Logcat untuk Logout aplikasi uBeacon. ©Dokumentasi Penulis.....	53
Gambar 32 Logcat penyimpanan data pada local memory. ©Dokumentasi Penulis	53
Gambar 33 Hasil <i>parsing</i> dari local memory. ©Dokumentasi Penulis	54
Gambar 34 Edit <i>Profile</i> aplikasi uBeacon. ©Dokumentasi Penulis	54

Gambar 35 Loogcat untuk Service pada notification. ©Dokumentasi Penulis.....	54
Gambar 36 Tampilan front-end untuk fitur notification. ©Dokumentasi Penulis	55
Gambar 37 Informasi yang diterima oleh aplikasi. ©Dokumentasi Penulis.....	55
Gambar 38 Tampilan aplikasi uBeacon saat push notification. ©Dokumentasi Penulis	56

DAFTAR TABEL

Tabel 1 Spesifikasi aplikasi uBeacon. ©Dokumentasi Penulis.....	11
Tabel 2 Spesifikasi <i>back-end</i> aplikasi uBeacon. ©Dokumentasi Penulis	11
Tabel 3 Penjelasan antarmuka aplikasi uBeacon. ©Dokumentasi Penulis	20

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pertumbuhan jumlah mahasiswa baru setiap tahunnya secara tidak langsung berdampak pada jumlah persebaran informasi yang meningkat [1]. Dalam hal ini, informasi yang dimaksud dapat berupa informasi mengenai acara-acara kampus, promosi kantin, berita kampus, berita akademik, berita mengenai kegiatan unit juga himpunan dan masih banyak lagi. Kampus sebagai pihak penyelenggara kegiatan akademik tentu mendukung penuh bertumbuhnya kemampuan *hardskill* dan *softskill* mahasiswanya. Namun, sistem persebaran informasi yang tidak dikelola dengan baik akan menimbulkan banyak kesulitan baik bagi sasaran informasi dalam hal ini mahasiswa juga pengunjung kampus dan tentunya bagi pihak penyebar informasi. Permasalahan yang kerap terjadi adalah keadaan dimana informasi tidak terdistribusi dengan baik, sehingga tidak semua mahasiswa mengetahui informasi tersebut. Contoh lain adalah tidak tersampainya informasi secara lengkap, sehingga tujuan dari penyampaian informasi menjadi hilang. Hal ini berakibat secara langsung pada kegiatan yang bersangkutan karena tidak memenuhi sasaran dan juga bagi mahasiswa.

Dengan semakin banyaknya jumlah informasi yang harus didistribusikan kepada mahasiswa, maka dibuatlah beberapa metode pendistribusian informasi. Namun metode-metode tersebut tidak mampu sepenuhnya memastikan bahwa informasi terdistribusikan secara lengkap dan personal kepada mahasiswa. Kedepannya, solusi terbaik untuk menangani permasalahan ini adalah dengan membangun sistem yang mampu mengawasi serta mengontrol alir pendistribusian informasi serta mampu memastikan bahwa mahasiswa menerima informasi secara lengkap dan personal.

Dalam makalah ini, akan dipaparkan sebuah solusi dari permasalahan di atas, yaitu sebuah aplikasi pengelola sistem informasi berbasis Android – uBeacon yang mengkhususkan pembahasan mengenai salah satu fitur utamanya yaitu *User Personalization* dan *Push-Notification*. Aplikasi ini diharapkan mampu membantu pendistribusian informasi dengan menampilkan informasi secara lengkap dan personal sesuai dengan keanggotaan mahasiswa di unit dan program studi melalui jaringan

internet, sebagai media presensi kelas bagi mahasiswa, serta dapat memperoleh informasi yang diunduh secara langsung dari *web server* setelah aplikasi melakukan *scan Beacon*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan pada bagian sebelumnya, maka rumusan masalah dalam tugas akhir ini sebagai berikut:

- Bagaimana rancangan sistem pendistribusian serta pengelolaan informasi yang dapat diimplementasikan?
- Bagaimana rancangan aplikasi yang digunakan untuk menghubungkan antara mahasiswa dan pengunjung kampus dengan pengelola informasi?

1.3 Tujuan dan Materi

Tujuan umum dari tugas akhir ini adalah sebagai berikut:

- Pemenuhan salah satu syarat kelulusan pada program sarjana program studi Teknik Elektro. Sekolah Teknik Elektro dan Informatika ITB.
- Riset untuk pengembangan teknologi iBeacon.
- Riset untuk pengembangan sistem *smart campus*.

Tujuan khusus dari tugas akhir ini adalah sebagai berikut:

- Merancang sistem pengelolaan informasi dan mengimplementasikannya ke dalam aplikasi berbasis android yang dapat menghubungkan antara mahasiswa dan pengunjung kampus dengan pengelola informasi (pihak kampus).
- Merancang bentuk komunikasi antara aplikasi berbasis android dan penggunaan teknologi iBeacon.

1.4 Lingkup Permasalahan

Tugas akhir ini dibuat dengan asumsi dan batasan sebagai berikut:

- Aplikasi hanya dapat dijalankan dengan menggunakan *smartphone* android dengan minimal fitur Bluetooth 4.0 dan koneksi internet yang selalu aktif.

1.5 Metodologi

Seluruh dokumentasi pada buku tugas akhir ini merupakan seluruh data-data kegiatan selama pengerjaan tugas akhir. Metode yang digunakan untuk penyelesaian tugas akhir ini adalah sebagai berikut:

1. Penentuan Topik

Penentuan topik dilakukan dengan memilih topik yang telah ditentukan oleh tim Tugas Akhir berdasarkan pilihan penulis.

2. Penentuan Spesifikasi

Tahap menentukan spesifikasi dari keluaran yang diharapkan.

3. Studi Literatur

Mempelajari desain dari sistem dan algoritma yang sudah ada sebagai dasar perancangan untuk memenuhi spesifikasi yang ingin dicapai.

4. Perancangan dan Implementasi Sistem

Aplikasi yang akan dibuat dirancang berdasarkan memperhatikan masukan, kebutuhan untuk melakukan proses, dan keluaran yang diharapkan. Kemudian aplikasi dibuat dengan menggunakan *environment* pemrograman yang sesuai.

5. Pengujian dan Perbaikan Sistem

Pengujian sistem dilakukan untuk melihat performansi dan kinerja sistem dan melakukan perbaikan untuk *failure/bug* yang dihasilkan.

6. Penarikan Analisis dan Kesimpulan

Berhasil atau tidaknya hasil pengujian sistem dianalisis dan ditarik kesimpulannya.

1.6 Sistematika Penulis

Sistematika penulisan yang digunakan dalam penyusunan laporan adalah sebagai berikut:

- BAB I PENDAHULUAN

Berisikan latar belakang, rumusan masalah, tujuan dari Tugas Akhir, lingkup permasalahan, metodologi penulisan laporan, dan sistematika penulisan buku Tugas Akhir.

- BAB II TINJAUAN PUSTAKA

Berisikan tinjauan pustaka dan landasan teori yang akan digunakan selama pengerjaan Tugas Akhir untuk personalisasi dan notifikasi aplikasi uBeacon.

- BAB III DESAIN DAN IMPLEMENTASI

Berisikan rincian desain dan implementasi dari sistem personalisasi dan notifikasi aplikasi uBeacon.

- BAB IV HASIL PENGUJIAN

Berisikan hasil pengujian dari personalisasi dan notifikasi aplikasi uBeacon.

- BAB V KESIMPULAN DAN SARAN

Berisikan saran dan kesimpulan selama pengerjaan Tugas Akhir

BAB II

TINJAUAN PUSTAKA

2.1 Android Operating System

Android adalah *mobile operating system* (OS) yang dikembangkan oleh Google, berbasis Linux yang dirancang untuk perangkat layar sentuh seperti telepon seluler dan *computer tablet* [2-3]. Awalnya Android dikembangkan oleh Android, Inc., sebelum dilakukan transaksi jual beli dengan Google pada tahun 2005. Android merupakan sistem operasi terbuka (*open source*) sehingga sangat dimungkinkan dilakukan modifikasi perangkat lunak secara bebas. Pendistribusian hasil modifikasi ini pun dilakukan langsung oleh pihak pengembang aplikasi, operator nirkabel, dan pembuat perangkat. Selain itu, Android memiliki sejumlah besar komunitas pengembang aplikasi yang mampu memperluas fungsionalitas dari perangkat. Pengembangan dari aplikasi umumnya ditulis dalam bahasa pemrograman Java.

Untuk dapat membuat aplikasi baru para Android, umumnya tools yang digunakan adalah *Android Software Development Kit* (SDK). Didalam Android SDK terdapat seperangkat *tools* *separate debugger*, *library*, *emulator*, dokumentasi, contoh kode dan tutorial. Pengembangan Android SDK sejalan dengan pengembangan *platform* Android secara keseluruhan. SDK ini mensupport *platform* Android dengan versi yang lebih lama jika *developer* ingin mengembangkan aplikasinya pada *device* yang lama. Aplikasi Android dikemas dalam format .apk yang berisi *file .dex* (*Dalvik executables*), *file resources* dan lain-lain.

2.2 Java

Java merupakan bahasa pemrograman yang dapat dijalankan di berbagai jenis komputer termasuk telepon genggam. Bahasa pemrograman ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan model yang lebih sederhana. Aplikasi berbasis Java umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Dari segi fungsionalitasya sangat dimungkinkan bagi aplikasi berbasis Java

untuk dijalankan di *platform* sistem operasi yang berbeda-beda.. Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

2.3 Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) merupakan protokol komunikasi dengan sistem *publish* dan *subscribe* yang didesain dengan sistem pengiriman yang sangat ringan (*lightweight*) dan sederhana sehingga sangat ideal untuk digunakan bagi *smartphone* karena konsumsi dayanya yang rendah. MQTT diciptakan pada tahun 1999 oleh Dr Andy Stanford-Clark dari IBM dan Arlen Nipper dari Arcom (sekarang Eurotech). Implementasi MQTT pada *mobile application* sangat dimungkinkan untuk koneksi *machine-to-machine* (M2M) atau *Internet of Things* (IoT). Pola transmisi data menggunakan *publish* dan *subscribe* membutuhkan sebuah Broker. Broker bertanggung jawab untuk mendistribusikan data kepada *client* yang tertarik, sesuai dengan *topic* yang digunakan. Berikut ilustrasi kerja protokol MQTT:



Gambar 1 Konfigurasi kerja protokol MQTT. ©Dokumentasi Penulis

2.4 Graphical User Interface (GUI)

Graphical Pengguna Interface (GUI) adalah tampilan grafis dari satu atau beberapa jendela yang memiliki kontrol (disebut komponen), sehingga memungkinkan pengguna untuk melakukan pekerjaan secara interaktif. Pengguna dari GUI tidak perlu membuat *script* atau mengetikkan *command* pada *command line* untuk menyelesaikan suatu pekerjaan.

Komponen GUI dapat berupa *menu*, *toolbar*, *push button*, *radio button*, *list box*, *slider*, dan sebagainya. Beberapa tools dapat digunakan untuk mendesain GUI, di antaranya Eclipse, NetBeans, Microsoft Visual Studio, Microsoft Visual Basic, dan Matlab. Developer juga memiliki beberapa pilihan untuk bahasa pemrograman yang digunakan seperti Java, C++, C#, Python atau HTML. GUI *engineer* dapat memilih bahasa dan *tools* yang digunakan sesuai dengan keperluan sistem.

Umumnya GUI menunggu pengguna untuk memanipulasi sebuah kontrol dan kemudian merespon kepada setiap aksi pengguna pada gilirannya. GUI dan setiap kontrol memiliki sebuah satu atau lebih *callbacks*, sehingga yang disebut perancangan GUI adalah membuat kode *callbacks* yang mendefinisikan apa yang dilakukan oleh masing-masing komponen untuk masing-masing perintah dari pengguna.

2.5 JSON file

JSON adalah format standar terbuka yang menggunakan teks *human-readable* untuk mengirimkan data objek yang terdiri dari pasangan *key-value*. Format ini merupakan format data yang paling umum digunakan untuk komunikasi *browser* atau *server asynchronous*.

JSON adalah format data yang *language-independent*. Format ini berasal dari JavaScript, tetapi pada 2106 format JSON data telah tersedia dalam banyak bahasa pemrograman. Ekstensi yang digunakan untuk JSON adalah `.json`.

```
{
  "_id": "rizkyindra",
  "password": "asdfghjkl;",
  "fullname": "Rizky Indra Syafrian",
  "email": "rzkyndr@gmail.com",
  "birthdate": "1994-08-02",
  "occupation": "Mahasiswa",
  "interest": "Seminar"
}
```

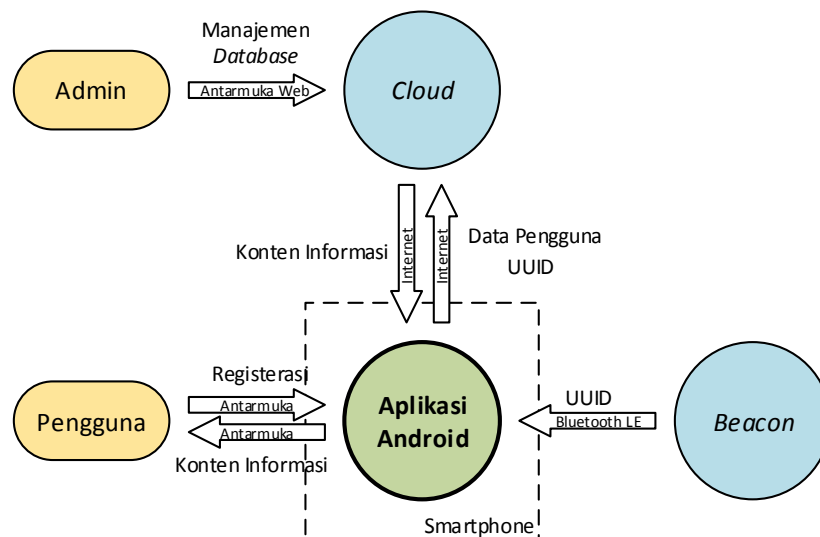
Gambar 2 Format JSON *file*. ©Dokumentasi Penulis

BAB III

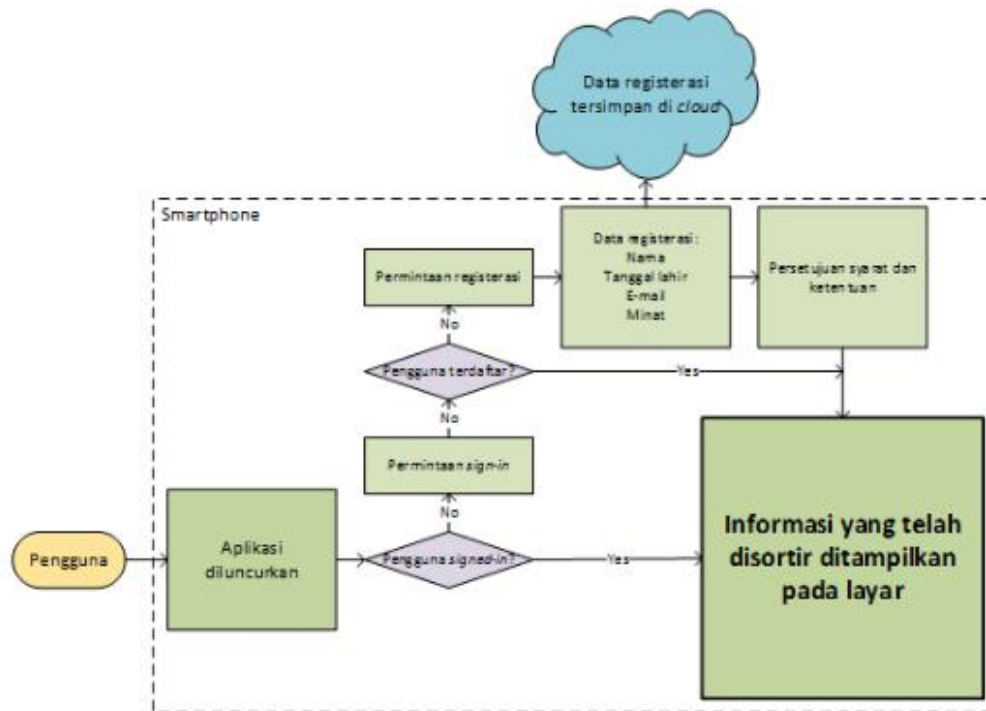
DESAIN DAN IMPLEMENTASI

3.1 Rancangan Umum

Aplikasi yang diberi nama uBeacon (*University Beacon*) merupakan aplikasi utama pengelolaan sistem informasi di lingkungan kampus sebagai kesatuan sistem *Smart Campus* yang diusung. Aplikasi ini menghubungkan antara pengguna (mahasiswa maupun pengunjung kampus) dengan informasi di kampus melalui *back-end application* dan beacon dalam kesatuan sistem. Perancangan aplikasi ini diimplementasikan dengan menggunakan sistem operasi berbasis Android. Melalui aplikasi ini, pengguna dapat memperoleh informasi seputar kampus secara mudah. Selain itu pihak kampus juga mampu mendistribusikan informasi secara efektif dan efisien.



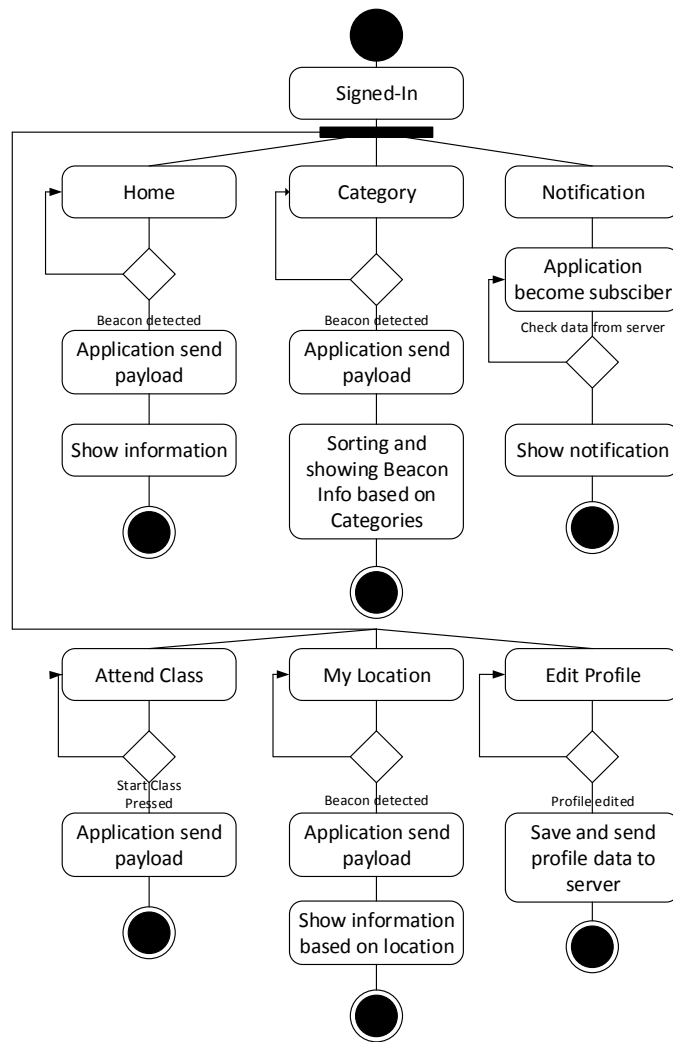
Gambar 3 Diagram rancangan umum sistem uBeacon. ©Dokumentasi Penulis



Gambar 4 Diagram blok sub-sistem aplikasi uBeacon. ©Dokumentasi Penulis

Gambar diatas menunjukkan diagram sistem secara keseluruhan. Aliran informasi dimulai dari konten informasi yang dikelola oleh administrator sebagai pemegang akses *cloud server*. Kode UUID pada beacon yang terdeteksi oleh *smartphone* pengguna akan dibaca dan dikirimkan ke *cloud*. Pada *cloud*, konten informasi dengan UUID yang sesuai informasi akan dikirimkan kembali ke *smartphone* pengguna. Informasi yang sampai ke *smartphone* pengguna dan disortir berdasarkan minat pengguna yang berdasarkan pada data registrasi, serta dipengaruhi oleh jarak relatif pengguna dengan beacon.

Smartphone akan menerima informasi ketika pengguna bergerak mendekati perangkat beacon yang memancarkan sinyal BLE. Informasi keluaran yang diterima pengguna adalah konten informasi yang sesuai dengan minat pengguna yang telah disortir berdasarkan pada *input* data saat registrasi akun. *Back-end application* juga mampu mengirimkan informasi ke *smartphone* pengguna tanpa adanya *trigger* dari beacon. Dalam hal ini aplikasi dengan protokol komunikasi MQTT akan bekerja sebagai *subscriber* dan mampu menerima informasi dari *cloud* kapanpun *cloud* mengirimkan berita tersebut. *Cloud* akan mengirimkan informasi ke pengguna berdasarkan pada data registrasi akun pengguna.



Gambar 5 Flowchart alir aplikasi uBeacon. ©Dokumentasi Penulis

Skenario sistem diatas dapat dijelaskan sebagai berikut:

1. Pengguna harus memiliki *smartphone* yang dilengkapi dengan Bluetooth 4.0 dan aplikasi uBeacon (*University Beacon*). Jika *smartphone* pengguna tidak dilengkapi dengan Bluetooth 4.0 maka uBeacon tidak dapat melakukan *scanning* pada Beacon sehingga fungsi aplikasi tidak akan berjalan dengan sempurna.
2. Sebelum menggunakan aplikasi, pengguna harus melakukan registrasi akun sehingga pengguna dapat memperoleh informasi sesuai dengan data pribadinya.
3. Setelah proses Sign In berhasil dilakukan, terdapat beberapa pilihan menu, yaitu: Home, Categories, My Location, Notifications, Attend Class, Edit Profile, About, Feedback, and Log Out.
4. Pada menu Home akan ditampilkan hasil *scanning* dari Beacon berupa informasi yang telah terunduh dan telah di-*parsing* pada layar *smartphone* pengguna.

Pengguna cukup menyentuh informasi yang diinginkan untuk memperoleh informasi secara lengkap dari informasi yang bersangkutan.

5. Pada menu Attend Class pengguna dapat melakukan absensi kelas tanpa melakukan absensi pada kertas karena sistem telah memiliki *database* mahasiswa dalam kelas.
6. Pada menu Edit Profile, pengguna dapat melakukan update pada data diri yang sebelumnya telah dilakukan pada registrasi.
7. Pada menu Logout, pengguna akan keluar dari sistem sehingga perlu dilakukan Sign-In kembali.

3.2 Spesifikasi

<i>Mobile Application</i>		
Spesifikasi		Keterangan
Sistem Operasi	Android Kitkat	Versi 4.3 ke atas
<i>Bluetooth Type</i>	Bluetooth 4.0	Bluetooth Low Energy
<i>Internal Storage</i>	50 MB	<i>Minimum storage</i>
<i>Internet Connection</i>	<i>Active</i>	<i>Active</i>

Tabel 1 Spesifikasi aplikasi uBeacon. ©Dokumentasi Penulis

<i>Back-end Application</i>		
Spesifikasi		Keterangan
<i>Disk Quota</i>	1024 MB	Menyesuaikan dengan batasan yang diberikan oleh IBM Bluemix
<i>Domain</i>	Sydney, Australia	<i>Latency</i> kurang dari 100 ms
<i>Instances</i>	2	Jumlah aplikasi yang <i>running</i> di IBM Bluemix
<i>Memory</i>	512 MB	Untuk menyimpan informasi <i>beacon</i> , basis data, dan presensi kelas
<i>Services & APIs</i>	3	Digunakan untuk kebutuhan analisis data

Tabel 2 Spesifikasi *back-end* aplikasi uBeacon. ©Dokumentasi Penulis

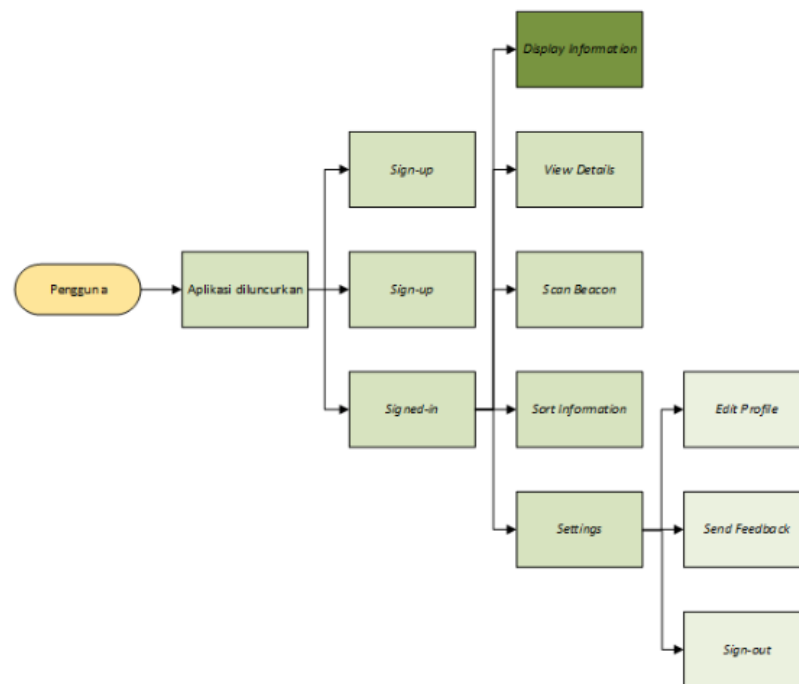
3.3 Desain

Aplikasi pengelola sistem informasi, uBeacon, dibuat untuk *smartphone* dengan sistem operasi berbasis Android. Perancangan dari interaksi pengguna dan aplikasi ini dibuat menggunakan *software* Android Studio dengan bahasa pemrograman Java. Pada

skenario sistem, pengguna diminta untuk melakukan registrasi. Registrasi dilakukan dengan memasukkan data profil pengguna berupa Nama, *Username*, *Password*, *Email*, *Birthdate*, NIM, *Occupation*, *Interest*, dan Unit. Registrasi dilakukan untuk menjalankan fitur personalisasi yang dimiliki oleh aplikasi. Segala fitur yang dimiliki oleh aplikasi uBeacon mengharuskan penggunanya untuk melakukan registrasi.

Selain didesain untuk berinteraksi dengan pengguna, aplikasi ini juga didesain untuk berinteraksi dan berkomunikasi dengan *back-end application* untuk transfer data dan informasi.

Pada fitur User Personalization dan Push-Notification dibutuhkan data profil pengguna untuk menerima informasi berdasarkan keanggotaan pengguna di Unit dan Program Studi. Aplikasi harus didesain untuk siap menerima informasi dari *back-end* kapanpun dan dimanapun selama memiliki koneksi internet.



Gambar 6 Diagram Blok aplikasi uBeacon. ©Dokumentasi Penulis

3.4 Perancangan Graphical User Interface (GUI)

GUI dari fitur *User Personalization* dan *Push-Notification* dibuat agar pengguna dapat dengan mudah memperoleh informasi berdasarkan dengan keanggotaan pengguna di Unit dan Program Studi. Secara keseluruhan ada tiga keadaan (*state*) yang akan dialami pengguna setelah mengunduh aplikasi uBeacon, yaitu:

1. Pengguna yang belum memiliki akun dalam hal ini harus melakukan proses registrasi atau Sign-Up terlebih dahulu.

2. Pengguna yang sudah memiliki akun namun belum melakukan Sign-In.
3. Pengguna yang telah memiliki akun dan telah melakukan Sign-In.

Semua proses yang dijalankan pada aplikasi secara umum dibagi kedalam dua buah proses yaitu *Foreground Process* dan *Background Process*. *Foreground process* dimulai dari tampilan Sign-In yang muncul sesaat setelah pengguna menjalankan aplikasi uBeacon. Alur rancangan yang akan dibuat terlihat seperti gambar di bawah:



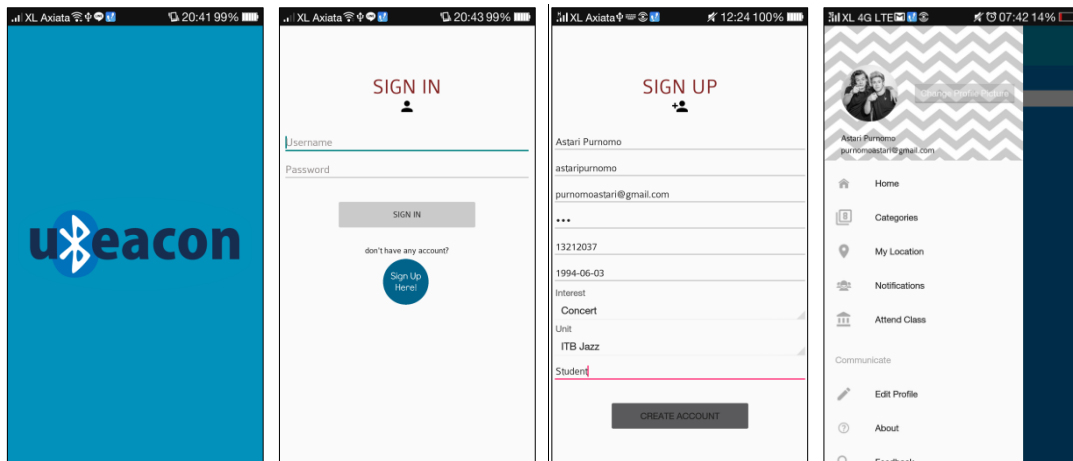
Gambar 7 Ilustrasi Graphical User Interface untuk aplikasi uBeacon. ©Dokumentasi Penulis

Aplikasi dibuat beralur untuk memenuhi kebutuhan dari masing-masing data masukan.

3.5 Implementasi Aplikasi

3.5.1 Graphical User Intergace (GUI) uBeacon

Foreground process dimulai dari *splash screen* yang muncul saat pengguna meluncurkan uBeacon *mobile application* seperti yang terlihat pada gambar dibawah:



Gambar 8 Tampilan antarmuka aplikasi uBeacon. ©Dokumentasi Penulis


Pada tampilan pertama dari gambar diatas, merupakan *splash screen* dari Android. *Splash screen* adalah tampilan awal dari aplikasi sebelum masuk ke *state* atau menu utama dari aplikasi. Untuk implementasi dari *splash screen*, dibentuk sebuah aktivitas baru yang dideklarasikan dalam Android Manifest. Aktivitas ini diatur sebagai **LAUNCHER** sedangkan aktivitas utama pada *mobile application* diatur sebagai **DEFAULT**. Selanjutnya dilakukan pengaturan waktu untuk menentukan rentang waktu *splash screen* muncul dalam layar *smartphone* menggunakan sebuah *class* Animation. Pada kasus kali ini digunakan *splash screen* dengan animasi `fade_out`. Setelah animasi ini berakhir, tampilan akan masuk ke *state* selanjutnya tergantung dengan *state* yang sedang dialami pengguna (Sign-In atau Signed-In).



```
@Override
public void onAnimationEnd(Animation animation) {
    //memulai splash
    iv.startAnimation(an2);
    //mengakhiri splash
    finish();
    Intent i = new Intent(SplashWelcome.this, SignIn.class);
    startActivity(i);
}
```





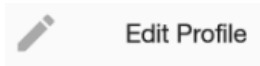
Selanjutnya setelah durasi animasi `abc_fade_out` pada *splash screen* berakhir, program akan memanggil *default activity*. Pada kasus ini, *default activity* yang dimaksud tergantung dari *state* pengguna apakah sebelumnya pengguna telah melakukan proses Sign-In. Pada sepenggal baris kode diatas, dapat dijabarkan bahwa setelah animasi berakhir akan dilakukan pergantian tampilan atau *state* yang dilakukan menggunakan *Intent*. *Intent* merupakan barisan deskripsi yang menyatakan operasi

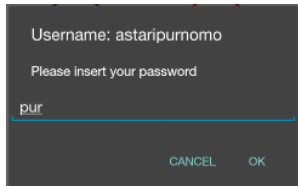
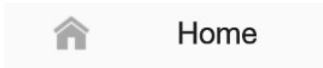

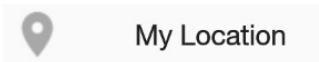
yang harus ditampilkan, pada hal ini *Intent* dilakukan dari *splash screen* menuju *layout* Sign-In. Pada transisi ini akan dilakukan pengecekan pada *SharedPreferences* dimana sebelumnya telah di-*store state* yang sedang dijalani pengguna. Jika sebelumnya pengguna telah melakukan Sign-In maka pengguna langsung masuk ke tampilan menu utama, tapi jika Sign-in belum dilakukan akan dilakukan proses Sign-In terlebih dahulu untuk masuk ke dalam menu utama.



Pada table dibawah akan dijabarkan setiap fungsi pada *layout* yang digunakan pada keseluruhan aplikasi sebelum dibahas lebih lanjut:

Nama Fitur	Tampilan	Deskripsi
Sign In		<ul style="list-style-type: none"> • Mengubah <i>input</i> dari pengguna berupa Username dan Password ke dalam bentuk <i>JSON file</i>. • Mengirimkan data pengguna berupa <i>username</i> dan <i>password</i> ke <i>database</i> untuk dilakukan pengecekan. • Apabila data tidak cocok dengan <i>database</i> maka akan ditampilkan pesan “Wrong password” atau ”Username not exist” sedangkan bila <i>input</i> yang dimasukan cocok dengan <i>database</i> akan diterima pesan status “Sign-in success”. • Aplikasi akan menampilkan halaman utama dari program berupa <i>Navigation.class</i> yang menampilkan semua informasi dari beacon ketika Sign-In berhasil dilakukan. • Balasan dari <i>server</i> akan disimpan dalam <i>memory local smartphone</i> pengguna menggunakan <i>Shared Preference</i> untuk menyimpan status <i>login</i> serta data diri pengguna. • Saat Sign In berhasil dilakukan, aplikasi

	<p>menjalankan Service pada <i>background</i> dan berubah menjadi <i>subscriber</i> ke alamat <i>topic</i> tertentu dan harus selalu siap menerima data dari <i>server</i> kapanpun itu.</p>
<p>Sign-Up Here</p> 	<ul style="list-style-type: none"> • Bila pengguna belum pernah melakukan registrasi sebelumnya maka pengguna diminta untuk melakukan registrasi terlebih dahulu untuk mendapatkan akun pribadinya. Dengan meng'klik' tombol ini maka akan ditampilkan <i>layout</i> Sign Up menggunakan baris kode <code>Intent</code>. • Apabila <i>layout</i> telah berubah ke <i>layout</i> Sign Up maka akan ditampilkan pesan "Please make a new account".
<p>Create Account</p> 	<ul style="list-style-type: none"> • Mengubah <i>input</i> dari pengguna berupa Nama, Username, Password, Email, NIM, Birthdate, <i>Interest</i>, Unit, dan <i>Occupation</i> ke dalam bentuk <i>JSON file</i>. • Mengirimkan data pengguna ke <i>database</i> untuk dilakukan pengecekan apakah <i>username</i> yang dipilih sudah pernah digunakan. • Apabila data tidak cocok dengan <i>database</i> maka akan ditampilkan pesan "Username taken" dan pengguna harus melakukan <i>input username</i> ulang. Sedangkan bila <i>input</i> yang dimasukan cocok dengan <i>database</i> akan diterima pesan status "Registration success". • Aplikasi akan menampilkan halaman Sign-In setelah proses registrasi berhasil dilakukan.

<p>Menu</p>	 <ul style="list-style-type: none"> Dengan meng'klik'tombol menu akan ditampilkan <i>navigation drawer</i> atau pilihan menu yang ada pada aplikasi berupa <i>sorting</i> informasi sesuai dengan jenis informasi yang ada. Terdapat 8 pilihan dalam menu yaitu Category, Unit Notification, Unread Information, Advertisement, Attend Class, Edit Profile, Help, dan Log Out.
<p>eter input pengg</p>	 <ul style="list-style-type: none"> Tampilan untuk meminta <i>input</i> pengguna berupa <i>username</i> dan <i>password</i>.
<p>Profile Picture</p>	 <ul style="list-style-type: none"> Dengan meng'klik'tombol ini maka aplikasi akan melakukan aksi dengan melakukan akses ke <i>gallery smartphone</i> pengguna sehingga pengguna dapat memilih <i>file .jpg</i> atau <i>.png</i> untuk diubah kedalam bitmap dan ditampilkan dalam <i>ImageView</i>. Uri dari <i>file</i> tersebut disimpan dalam <i>Shared Preference</i> agar dapat diakses sewaktu aplikasi dibuka kembali.
<p>Nama dan Email pengguna</p>	 <ul style="list-style-type: none"> Pada <i>navigation header</i> juga terdapat informasi berupa nama dan <i>email</i> pengguna. Informasi ini ditarik dari <i>Shared Preference</i> yang sebelumnya diperoleh dari proses Sign In.
<p>Edit Profile</p>	 <ul style="list-style-type: none"> Dengan meng'klik; tombol ini maka aplikasi akan meminta <i>input</i> berupa <i>password</i> pengguna untuk validasi. <i>Password</i> pengguna diperoleh aplikasi saat dilakukan proses Sign-In. Jika <i>input</i>

		<p><i>password</i> salah dan tidak berhasil dilakukan maka akan muncul <i>dialog</i> yang menyatakan bahwa <i>password</i> yang diinput pengguna salah. Sedangkan jika <i>input</i> yang dilakukan pengguna benar maka <i>layout</i> akan berubah ke xml dari Edit Profile.</p>
Validasi Password untuk Edit Profile		<ul style="list-style-type: none"> • Tampilan dialog saat aplikasi meminta <i>input</i> dari pengguna berupa <i>password</i> pengguna untuk validasi dilakukannya edit pada <i>profile</i> pengguna.
Home		<ul style="list-style-type: none"> • Dengan meng'klik' tombol ini maka aplikasi akan menampilkan xml dari <i>fragment home</i>. • Ditampilkan informasi berupa <i>array list</i> yang berisi informasi hasil scan Beacon.
Category		<ul style="list-style-type: none"> • Dengan meng'klik' tombol ini maka aplikasi akan menampilkan xml dari <i>fragment category</i>. • Ditampilkan informasi berupa <i>list category</i> yang dapat dipilih oleh pengguna. Terdapat 8 kategori, yaitu Seminar, Campus Info, Promotion, Unit and Himpunan Event, Workshop, Concert, Sprouts, Event.
My Location		<ul style="list-style-type: none"> • Dengan meng'klik' tombol ini maka akan ditampilkan informasi berupa lokasi keberadaan pengguna berdasarkan hasil scan Beacon.

Notifications	 Notifications	<ul style="list-style-type: none"> Dengan meng'klik' tombol ini maka akan ditampilkan <i>array list</i> berisi kumpulan informasi yang diperoleh pengguna dari <i>back-end application</i> berdasarkan keanggotaan pengguna di unit dan program studi.
Attend Class	 Attend Class	<ul style="list-style-type: none"> Dengan meng'klik' tombol ini maka pengguna dapat melakukan presensi sesuai dengan kelas yang terdeteksi dari hasil scan Beacon. Dilakukan pengiriman <i>payload</i> menuju <i>back-end application</i>.
Feedback	 Feedback	<ul style="list-style-type: none"> Dengan meng'klik' tombol ini maka akan dilakukan aksi pada smartphone berupa munculnya pilihan komunikasi yang akan digunakan oleh pengguna untuk mengirimkan feedback pada aplikasi. Feedback akan dikirimkan melalui e-mail.
About	 About	<ul style="list-style-type: none"> Dengan meng'klik' tombol ini maka akan ditampilkan tampilan berupa <i>textview</i> mengenai aplikasi
Logout	 Log Out	<ul style="list-style-type: none"> Dengan menekan tombol ini, akan ditampilkan dialog yang meminta konfirmasi pengguna bila pengguna akan keluar dari kondisi <i>signed-in</i>. Perlu dilakukan <i>sign-in</i> untuk dapat masuk kembali ke aplikasi.

The image shows a vertical registration form. It contains the following input fields from top to bottom: Name, Username, E-mail, Password, NIM, Birthdate (year-month-day), Interest, Unit, and Occupation. Each field is a simple text box. At the bottom of the form is a grey button labeled 'CREATE ACCOUNT'.

- Tampilan untuk meminta *input* pengguna berupa Nama, *Username*, *Password*, *Email*, NIM, *Birthdate*, *Interest*, Unit, dan *Occupation*.

Tabel 3 Penjelasan antarmuka aplikasi uBeacon. ©Dokumentasi Penulis

Pengaturan *layout* dilakukan pada *file* bertipe .xml. Pengaturan *layout* secara sederhana dan efisien dimaksudkan agar pengguna mampu berinteraksi dengan *server* secara mudah.

Pada tampilan Sign-In, Sign-Up dan Edit Profile, *layout* disusun secara *linear* dengan orientasi *vertical* untuk memudahkan dilakukannya pengaturan pada tampilan. Lebar dan panjang diatur dalam tipe `match_parent` sehingga cocok untuk diaplikasikan pada berbagai tipe layar *smartphone*. Tipe `match_parent` membuat tampilan dioperasikan sesuai dengan layar yang dipakai. Berikut adalah beberapa baris kode deskripsi yang menunjukkan pengaturan tampilan:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.astaripurnomo.ubeacon.SignIn"
    android:id="@+id/SignIn"
    android:tag="@string/app_name">
```

Setiap komponen pada tampilan diatur pada *file* .xml ini. Pada tampilan Sign In, Sign Up, Edit Profile digunakan beberapa komponen, diantaranya EditText, *Button*, TextView, ImageView, dan ImageButton. EditText digunakan sebagai komponen untuk meminta masukan dari pengguna dengan *input* berupa *text* baik itu huruf maupun angka. Pengaturan pada EditText dilakukan sebagai berikut:

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
```

```

        android:ems="10"
        android:id="@+id/Password"
        android:layout_gravity="center_horizontal"
        android:hint="Password"
        android:textColorHint="#908989"
        android:maxLines="1"
        android:singleLine="true"
        android:paddingLeft="5dp"
        android:paddingRight="5dp" />

```

Panjang dan lebar dari EditText disesuaikan dengan kebutuhan yang dibutuhkan. Dalam hal ini lebar disesuaikan dengan lebar layar yang akan menampilkan tampilan ini, sedangkan tinggi dari EditText disesuaikan dengan *input* pengguna. Tipe *input* yang dapat dimasukkan oleh pengguna juga diatur dalam *file* .xml. Pada aplikasi ini tipe *input* diatur sesuai dengan kebutuhan yang diperlukan oleh sistem. Untuk memudahkan pengguna dalam memasukan *input*, maka dalam EditText diberikan hint.

Button digunakan sebagai komponen yang dapat digunakan untuk memulai aktivitas baru. Aktivitas yang dapat diatur dalam metode `setOnClickListener(new View.OnClickListener())`. Dalam metode tersebut dapat dituliskan berbagai aktivitas yang diinginkan. Berikut adalah baris pengaturan pada salah satu *Button* yang dilakukan:

```

<Button
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:text="Sign In"
    android:id="@+id/SignInBtn"
    android:enabled="false"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="20dp"
    android:clickable="true" />

```

Agar pada saat *button* di click dapat dilakukan pengaturan pada *file* .java, maka harus dipastikan bahwa *clickable* harus bernilai *true*.

TextView merupakan metode digunakan untuk menampilkan text pada tampilan tanpa dilakukan tindakan apapun (meskipun sebenarnya bisa dilakukan). Berikut adalah baris kode TextView:

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"

```



```
        android:text="SIGN IN"
        android:textSize="30dp"
        android:layout_marginTop="70dp"
        android:textColor="#842424"
        android:layout_gravity="center_horizontal"
        android:id="@+id/SignInView" />
```

ImageButton pada aplikasi ini digunakan untuk transisi antara tampilan Sign In dengan tampilan Sign Up. Berikut adalah baris kode deskripsi *ImageButton* yang digunakan:

```
<ImageButton
    android:layout_width="75dp"
    android:layout_height="75dp"
    android:id="@+id/imageButton"
    android:layout_gravity="center_horizontal"
    android:src="@drawable/signup"
    android:layout_centerHorizontal="true"
    android:adjustViewBounds="true"
    android:cropToPadding="false"
    android:layout_below="@+id/hlTopBar"
    android:background="#00000000"
    android:scaleType="fitXY"/>
```

3.5.2 Koneksi MQTT

Protokol komunikasi MQTT menggunakan model *publishing* atau *subscribing*. Pada model ini *publisher* dan *subscriber* keduanya adalah *client*. Data yang ditransfer yang akan memanggil *client* tersebut. Transfer data dapat dilakukan saat koneksi *client* dengan broker telah stabil. *Publisher* mengirimkan (*publish*) data dengan *topic* khusus menuju Broker dan *subscriber* menerima (*subscribe*) data dengan *topic* khusus dari Broker. Broker mengatur koneksi antara *publisher* dan *subscriber*. Ketika Broker menerima data yang di-*publish*, Broker selanjutnya akan meneruskan data tersebut ke *subscriber*. Komunikasi menggunakan metode *publish* dan *subscribe* dapat dilakukan oleh lebih dari satu *publisher* dengan *topic* yang sama. Metode ini juga memungkinkan beberapa *client* untuk *subscribe* data dengan subject khusus, selanjutnya Broker akan melakukan *broadcast* ke *subscribers* yang berbeda.

Pada uBeacon, aplikasi uBeacon pada *smartphone* pengguna dan *back-end* berperan sebagai *client*. Secara umum, komunikasi yang dilakukan oleh aplikasi dengan *back-end* dilakukan untuk transfer data. Berikut adalah proses yang dilakukan oleh aplikasi untuk membuat koneksi dengan *back-end* (*client*) melalui

MQTT:

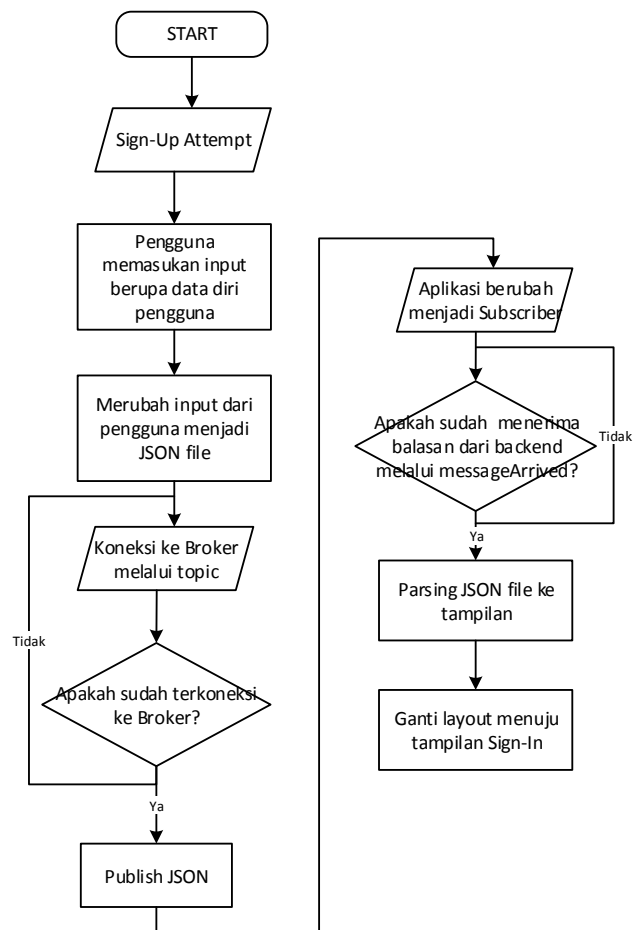
1. Dilakukan koneksi dengan MQTT. Aplikasi menentukan *IP address* (alamat Broker) dan *port* dari Broker yang ditugaskan oleh MQTT, selanjutnya dilakukan koneksi. Pada implementasinya digunakan alamat Broker: `tcp://test.mosquitto.org` dengan *port* 1883. *Client* dalam hal ini aplikasi, akan membuat MQTT *client object*. Pada MQTT object akan dideskripsikan *topic* juga *service quality* (QoS).
2. Dilakukan transfer data menuju Broker. Aplikasi akan bertindak sebagai *publisher* dan melakukan *publish* data ke Broker untuk memastikan bahwa *client* dan Broker telah terkoneksi.
3. Menerima data dari Broker. Pada tahap ini, akan dipastikan bahwa *subscriber* (*back-end*) telah menerima data yang di-*publish* aplikasi dengan memanggil fungsi Callback. Fungsi ini akan memberitahu aplikasi bahwa *back-end* sebagai *client* lain telah menerima data. Selanjutnya aplikasi akan berubah menjadi *subscriber* untuk *subscribe* balasan dari *back-end*.

Proses diatas merupakan gambaran secara umum bagaimana komunikasi menggunakan MQTT dilakukan pada sistem uBeacon.

3.5.3 Fitur User Personalization

Fitur User Personalization memiliki tiga bagian yaitu Sign-Up, Sign-In, dan Edit Profile.

3.5.3.1 Sign-Up



Gambar 9 Flowchart Sign-Up aplikasi uBeacon. ©Dokumentasi Penulis

Pada proses Sign-Up pengguna yang belum memiliki akun uBeacon akan melakukan registrasi. Pada bagian ini, pengguna akan melakukan *input* profil pengguna yang akan berguna untuk pendistribusian informasi bagi setiap pengguna secara personal. Terdapat sembilan komponen masukan untuk pengguna, diantaranya: nama, *username*, *password*, email, NIM, tanggal dan tahun lahir, unit, minat (*interest*), dan pekerjaan (*occupation*).

Sign-Up dinyatakan sebagai Activity yang terdiri dalam dua format yaitu .xml dan .java. Pada bagian .xml dilakukan proses pengaturan *layout* atau tampilan sesuai yang dikehendaki sehingga diharapkan mudah untuk dipahami pengguna. Pada bagian .java segala pengaturan terhadap *input* pengguna dilakukan. Sign-Up mengimplementasikan protokol *MqttCallback* yang memiliki tiga fungsi bawaan yaitu:

```

@Override
public void connectionLost(Throwable throwable) {}

```

```

@Override
public void messageArrived(String s, MqttMessage mqttMessage) throws
Exception {}

@Override
public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {}

```

Pada tampilan Sign-Up, saat tombol “Create Account” ditekan, maka *input* pengguna akan dipaketkan menjadi JSON *file* melalui baris kode:

```
jsonObject.put("key", value);
```

Selanjutnya akan dibuat koneksi ke broker dan *topic* melalui MQTT yang dinyatakan dalam:

```

MqttClient client = new
MqttClient("tcp://test.mosquitto.org:1883", "ubeacon/user/signup", new
MqttDefaultFilePersistence(cacheDir));
client.setCallback(SignUp.this);
client.connect();
client.subscribe("ubeacon/user/signup/" +
UsernameTxt.getText().toString());

```

Broker yang dituju adalah dengan *topic* `ubeacon/user/signup`. Selanjutnya setelah broker `tcp://test.mosquitto.org:1883` dengan port 1883 dan *topic* `ubeacon/user/signup` diinisialisasi, dilakukan koneksi ke broker dan dilakukan *subscribe* terlebih dahulu untuk memastikan bahwa broker siap digunakan.

Selanjutnya, paket JSON yang sudah dibuat diubah menjadi `MqttMessage` dan dikirim atau di-*publish* ke broker melalui baris kode:

```
mqttTopic.publish(mqttMessage);
```

Selanjutnya aplikasi berubah kembali menjadi *subscriber* dan menunggu balasan dari *back-end* mengenai JSON *file* yang telah dikirim. Aplikasi akan menerima balasan dari *back-end* berupa JSON *file* dan memanggil `public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {}` sebagai berikut:

```

@Override
public void messageArrived(String s, MqttMessage mqttMessage) throws
Exception {
    System.out.println("Message arrived: " + mqttMessage.toString());
    try {
        JSONObject object = new JSONObject
(String.valueOf(mqttMessage));
        status = object.getString("status");
        if (status.equals("Registration success")) {
            Intent intent = new Intent(this, SignIn.class);

```

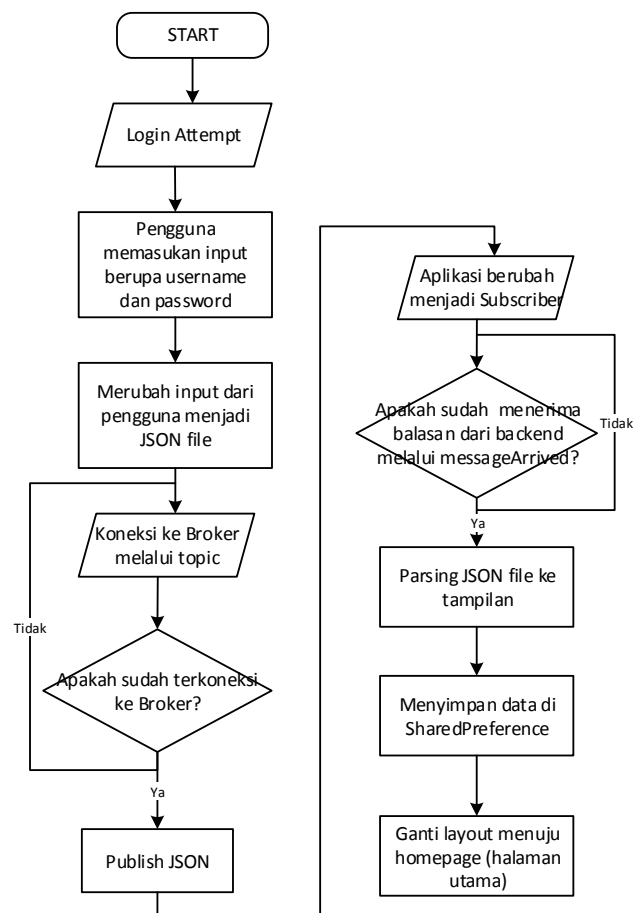
```

        startActivity(intent);
        finish();
    }else if (status.equals("Username taken")){
        UsernameTxt.setError(getString(R.string.UsernameTaken));
    }
} catch (JSONException e) {
    e.printStackTrace();
}
}

```

Balasan dari *back-end* akan diubah kedalam *JSON file* untuk selanjutnya dilakukan *parsing* dan diambil *value* dari paket *JSON* dengan *key* “status”. Jika status yang diterima berupa “*Registration success*” maka akan dilakukan pergantian *layout* ke Sign-In. Sedangkan jika *value* yang diterima berupa “*Username taken*” maka akan ditampilkan pesan yang memberitahu pengguna bahwa username yang dipilih telah digunakan sehingga harus diganti dengan username yang lain.

3.5.3.2 Sign-In



Gambar 10 Flowchart Sign-In aplikasi uBeacon. ©Dokumentasi Penulis

Proses selanjutnya adalah Sign-In. Pada proses ini akan dipastikan bahwa pengguna telah terdaftar dalam *database*. Sign-In digunakan sebagai jembatan bagi pengguna untuk memasuki menu utama. Secara garis besar pada proses Sign-In akan diminta masukan dari pengguna berupa username dan password yang sebelumnya telah ditentukan dan dibuat pada proses Sign-Up. *Input* dari pengguna selanjutnya akan dikirim ke *server* menggunakan protokol komunikasi MQTT. Jika proses berhasil dilakukan maka tampilan akan berubah ke menu utama, tetapi bila proses gagal dilakukan maka akan ditampilkan pesan untuk melakukan *input* ulang karena ada kesalahan dalam *input username* atau *password*.

Sign-In dinyatakan sebagai Activity yang terdiri dalam dua format yaitu .xml dan .java. Pada bagian .xml dilakukan proses pengaturan *layout* atau tampilan sesuai yang dikehendaki sehingga diharapkan mudah untuk dipahami pengguna. Pada bagian .java segala pengaturan terhadap *input* pengguna dilakukan. Sign-In mengimplementasikan protokol *MqttCallback* yang memiliki tiga fungsi bawaan seperti yang ditunjukkan pada bagian Sign-Up diatas.

Sesuai dengan tampilan Sign-In pada gambar diatas, pengguna diminta untuk memberikan *input* berupa Username dan Password. Selanjutnya *input* dari pengguna akan diubah kedalam JSON *file* pada saat *button* “SIGN IN” ditekan, sesuai dengan deskripsi berikut:

```
JSONObject jsonObject = new JSONObject();
try {
    //input from user being packaged in json form
    jsonObject.put("_id", UsernameTxt.getText().toString());
    jsonObject.put("password", PasswordTxt.getText().toString());
}
```

Selanjutnya baris deskripsi ini dikirimkan melalui protokol MQTT via Internet setelah sebelumnya dilakukan koneksi dengan broker melalui *topic* *ubeacon/user/signin/* yang telah disiapkan dan dipastikan bahwa broker siap untuk menerima *passing* data.

Pada ketiga fungsi yang dibawa oleh *MqttCallback* dilakukan pengaturan pada saat fungsi *messageArrived* dijalankan. Deskripsi fungsi yang dilakukan adalah sebagai berikut:

```
@Override
public void messageArrived(String s, MqttMessage mqttMessage) throws
Exception {
    System.out.println("Message arrived: " + mqttMessage.toString());
}
```

```

try {
    JSONObject object = new JSONObject(String.valueOf(mqttMessage));

    loginStatus = object.getString("status");

    switch (loginStatus) {
        case "Sign-in success":

            Name = object.optString("fullname");
            Email = object.optString("email");
            Username = object.optString("_id");
            Password = object.optString("password");
            Birthdate = object.optString("birthdate");
            NIM = object.optString("nim");
            Occupation = object.optString("occupation");
            Interest = object.optString("interest");
            Unit = object.optString("unit");

            SharedPreferences pref = getSharedPreferences(PREF_NAME,
MODE_PRIVATE);

            SharedPreferences.Editor editor = pref.edit();
            editor.putString(KEY_STATUS, "ok");
            editor.commit();

            Intent intent = new Intent(this, Navigation.class);
            startActivity(intent);
            finish();
            break;
        case "Username not exist": {
            System.out.println("Login Gagal");
            UsernameTxt.setError(getString(R.string.UsernameWrong));
            break;
        }
        default: {
            System.out.println("Login Gagal");
            PasswordTxt.setError(getString(R.string.PasswordWrong));
            break;
        }
    }
}

```

Pada baris kode diatas, aplikasi yang sebelumnya dijalankan sebagai *publisher* ke *topic* berubah menjadi *subscriber* menuju *topic*. Melalui baris tersebut maka aplikasi siap untuk menerima balasan dari *back-end* tepat setelah aplikasi berhasil mengirimkan JSON *file* ke *back-end*. Data balasan yang berasal dari *back-end* akan diubah menjadi JSON *file* untuk selanjutnya diambil *value* dari setiap *key* yang dibawa.

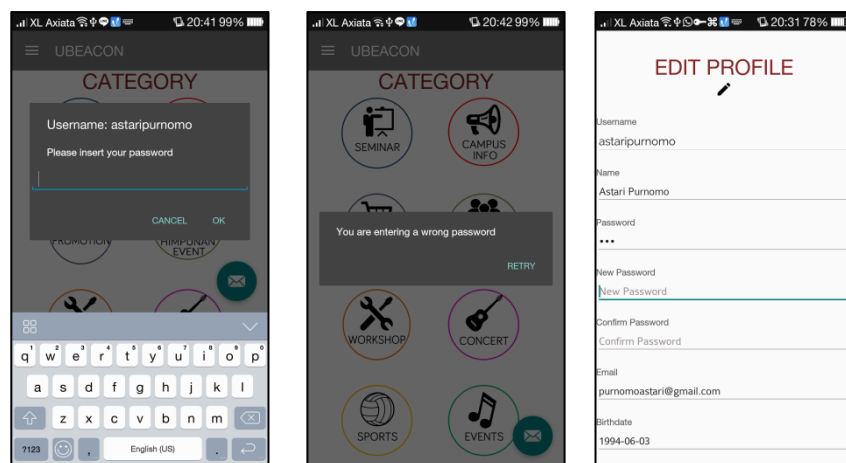
Pada implementasinya, dilakukan perbandingan string dari *value* balasan *server* dengan *key* "status". Setiap *value* dari *key* "status" diberikan perlakuan yang berbeda-beda. Sign-In yang berhasil ditandai dengan diterimanya balasan dari *server* berupa *value* "Sign-in success" pada *key* "status". Setiap *value* dari *file* JSON yang dibawa akan disimpan pada SharedPreference untuk digunakan bila pengguna ingin melakukan Edit Profile. Pada bagian ini, state Sign-In pengguna juga disimpan pada SharedPreference. Jika proses Sign-In berhasil dilakukan, maka *value* "ok" akan disimpan untuk selanjutnya akan dipanggil dan dilakukan perbandingan pada saat Sign-In.class dipanggil.

```
//checking login state
SharedPreferences pref = getSharedPreferences(PREF_NAME, MODE_PRIVATE);
String status = pref.getString(KEY_STATUS, "null");
System.out.println("status pref: " + status);
if (status.equals("ok")) {
    Intent intent = new Intent(this, Navigation.class);
    startActivity(intent);
    finish();
}
```

Dari baris kode berikut dapat dilihat bahwa saat status Sign-In pengguna adalah “ok” maka tampilan layar akan langsung dioperasikan pada `Navigation.class` menggunakan kode deskripsi `Intent`, sedangkan jika status Sign-In pengguna berupa “null” maka aplikasi akan tetap dijalankan pada `SignIn.class`.

3.5.3.3 Edit Profile

Bagian selanjutnya dari *User Personalization* adalah Edit Profile. Pada bagian ini pengguna dapat melakukan perubahan profil untuk pembaharuan data saat diperlukan. Aplikasi akan meminta validasi dari pengguna berupa *input password*. Jika *password* yang dimasukkan salah maka Edit Profile tidak dapat dilakukan. Sementara itu, saat *input password* yang dilakukan pengguna tepat maka Edit Profile dapat dilakukan. Validasi dilakukan dengan melakukan perbandingan string antara *input* pengguna dengan *password* pengguna yang telah disimpan dalam local memory *smartphone* menggunakan metode `SharedPreferences`.



Gambar 12 Tampilan antarmuka untuk Edit Profile aplikasi uBeacon. ©Dokumentasi Penulis

Jika pengguna berhasil melakukan validasi maka tampilan akan berubah ke `EditProfile.class` seperti tampilan diatas. Edit Profile terdiri dari *layout* yang diatur dalam.xml dan pengaturan pada setiap komponen *layout* yang dilakukan dalam *file .java*.

Pada Edit Profile, semua masukan dari pengguna sebelumnya yang disimpan dalam *database*, dan disimpan pada Shared Preference saat proses Sign-In dilakukan, dipanggil. Data tersebut kemudian dipanggil di Edit Profile dan ditampilkan (di-*parsing*) sesuai dengan *layout*nya. Berikut adalah baris kode yang digunakan untuk menampilkan data diri pengguna pada tampilan EditProfile.class:

```
SharedPreferences pref = getSharedPreferences(Data.SPREF_NAME, MODE_PRIVATE);
nameHeader = pref.getString("Name", "");
emailHeader = pref.getString("Email", "");
username = pref.getString("Username", "");
password = pref.getString("Password", "");
birthdate = pref.getString("Birthdate", "");
nim = pref.getString("NIM", "");
occupation = pref.getString("Occupation", "");
interest = pref.getString("Interest", "");
units = pref.getString("Unit", "");

//parsing
UsernameTxt = (EditText) findViewById(R.id.Username);
UsernameTxt.setEnabled(false);
UsernameTxt.setInputType(InputType.TYPE_NULL);
UsernameTxt.setFocusable(false);
UsernameTxt.setText(username);

PasswordTxt = (EditText) findViewById(R.id.Password);
PasswordTxt.setText(password);

NameTxt = (EditText) findViewById(R.id.Name);
NameTxt.setText(nameHeader);
```

Pengguna dapat melakukan perubahan pada setiap kolom *inputan*, kecuali pada bagian *username*. Hal ini dikarenakan *username* merupakan *id* khusus yang dimiliki pengguna dalam *database*, sehingga tidak dapat diganti. Selanjutnya pengguna dapat menyimpan hasil perubahannya dengan menekan *button* “SAVE”.

```
SaveBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //publish
        JSONObject jsonObject = new JSONObject();
        try {
            jsonObject.put("_id", UsernameTxt.getText().toString());
            jsonObject.put("oldpassword", PasswordTxt.getText().toString());
            jsonObject.put("password", NewPasswordTxt.getText().toString());
            jsonObject.put("fullname", NameTxt.getText().toString());
            jsonObject.put("nim", NIMTxt.getText().toString());
            jsonObject.put("email", EmailTxt.getText().toString());
            jsonObject.put("birthdate", BirthdateTxt.getText().toString());
            jsonObject.put("occupation", OccupationTxt.getText().toString());
            jsonObject.put("interest",
multiSelectionSpinner.getSelectedItemsAsString());
            jsonObject.put("unit", unitSelectionSpinner.getSelectedItemsAsString());

            String cacheDir = getCacheDir().getAbsolutePath();

            MqttClient client = null;

            client = new MqttClient("tcp://test.mosquitto.org:1883",
"ubeacon/user/editprofile", new MqttDefaultFilePersistence(cacheDir));
```

```

        client.connect();
        client.setCallback(EditProfile.this);
        client.subscribe("ubeacon/user/editprofile/" +
UsernameTxt.getText().toString());
        MqttTopic mqttTopic = client.getTopic("ubeacon/user/editprofile");
        MqttMessage mqttMessage = new
MqttMessage(jsonObject.toString().getBytes());
        mqttMessage.setQos(1);

        mqttTopic.publish(mqttMessage);

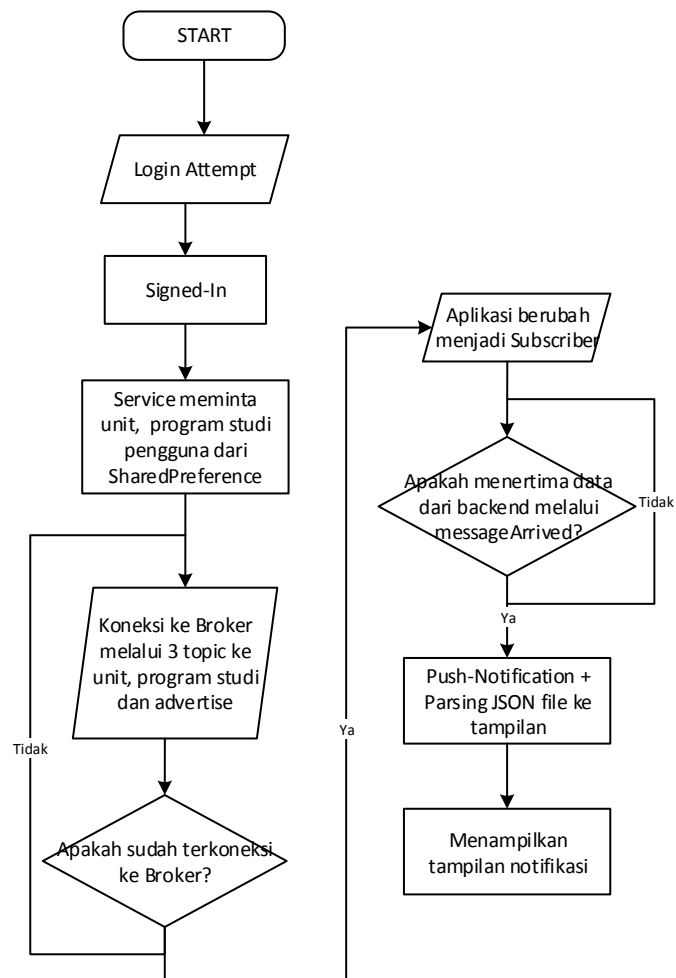
        } catch (MqttException e) {
            throw new RuntimeException(e);
        } catch (JSONException e) {
        }
    }
}
);

```

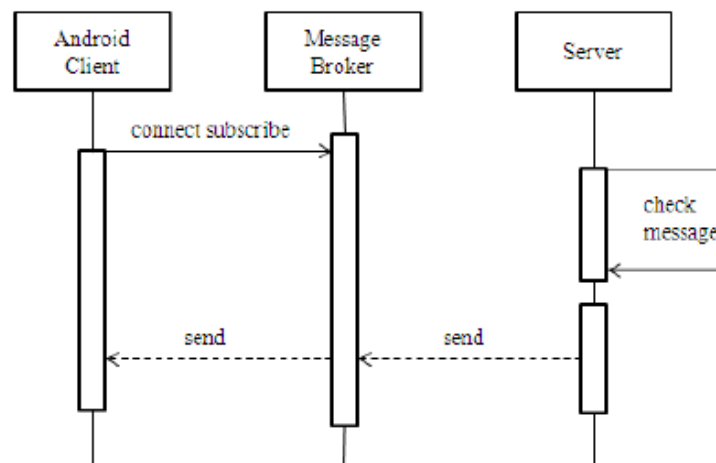
Seperti pada pemrosesan yang dilakukan pada Sign-In dan Sign-Up, proses yang sama juga dilakukan pada Edit Profile. Pada bagian ini, *input* dari pengguna di konversi menjadi JSON *file* dan selanjutnya dibuat koneksi melalui MQTT via internet pada *topic* `ubeacon/user/editprofile`. Setelah koneksi berhasil dibuat, JSON *file* akan di-*publish* ke alamat *topic*. Selanjutnya akan diterima balasan dari *back-end* pada *topic* `ubeacon/user/editprofile/` + `UsernameTxt.getText().toString()` untuk dilanjutkan proses selanjutnya. Bila perubahan berhasil dilakukan, maka akan terjadi pergantian tampilan kembali ke menu utama, dan bila perubahan *profile* gagal, akan muncul *error text* pada nama pengguna.

Jika proses perubahan *profile* berhasil dilakukan, pengguna akan menerima balasan dari *back-end* berupa status keberhasilan dan seluruh data diri terbaru pengguna. *Database* pengguna ini selanjutnya akan disimpan kembali pada `SharedPreferences`.

3.5.4 Fitur Push-Notification



Gambar 13 Flowchart push-notification aplikasi uBeacon. ©Dokumentasi Penulis



Gambar 14 Ilustrasi push-notification aplikasi uBeacon. ©Dokumentasi Penulis

Push-Notification merupakan fitur pendistribusian informasi dari *back-end* tanpa menggunakan trigger antara Bluetooth dan Beacon. Fitur ini dapat berjalan selama pengguna melakukan running pada aplikasi uBeacon dan

memiliki koneksi internet. Komunikasi antara aplikasi dan *back-end* menggunakan protokol MQTT dimana aplikasi berperan sebagai *subscriber* dan *back-end* berperan sebagai *publisher*. Saat pengguna berhasil melakukan running dan Sign-In pada aplikasi, maka secara otomatis Service pada aplikasi akan berjalan dan aplikasi berperan sebagai *subscriber* yang selalu siap menerima data terusan dari Broker. Selanjutnya saat *back-end* mengirimkan data pada aplikasi maka akan dimunculkan notifikasi pada *smartphone* pengguna. Klasifikasi informasi yang dapat diterima oleh *smartphone* adalah berdasarkan keanggotaan pengguna di unit dan program studi. Dapat pula dilakukan push-notification untuk kepentingan promosi atau berita penting dan mendesak lainnya kepada seluruh pengguna aplikasi.

Pada proses push notification, *back-end* menyediakan data dan aplikasi menerima data. Keduanya dilihat sebagai *client* oleh Broker. Saat aplikasi dijalankan, Service sebagai background process akan membuat aplikasi berperan sebagai *subscriber* dan melakukan koneksi dengan MQTT. Digunakan interface sederhana *MqttCallback* untuk melakukan komunikasi antara *client* dengan Broker. Saat push message diterima oleh aplikasi maka metode *Message Arrived* akan dipanggil untuk selanjutnya memanggil mekanisme notification (*Notification Manager*). Saat koneksi antara *client* dan Broker tidak stabil maka akan dipanggil metode *Connection Lost*. Metode ini akan melakukan re-connect dengan Broker jika koneksi internet telah stabil.

Dengan membuat aplikasi berperan sebagai *subscriber* maka aplikasi siap menerima informasi dari *server* kapanpun informasi dikirimkan selama ada koneksi internet. Untuk melakukan koneksi dan proses selama aplikasi di-running tanpa mengganggu pengguna dalam menggunakan fitur lain dalam aplikasi maka proses ini dilakukan menggunakan Service sebagai background process.

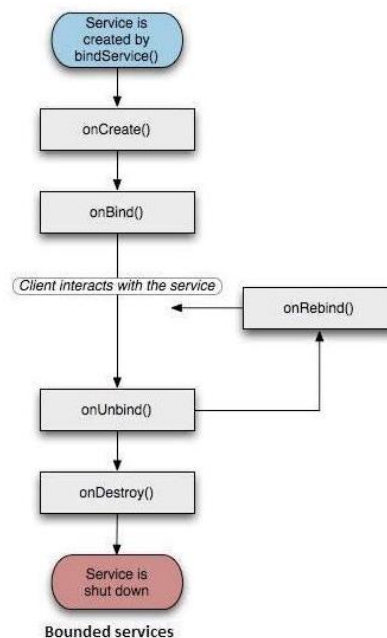
Service

Pada aplikasi ini, digunakan service yang akan melakukan proses pada Background sehingga tidak akan mengganggu aktivitas pengguna selama menggunakan aplikasi. Service adalah komponen yang beroperasi di background untuk melakukan operasi dalam jangka panjang tanpa memerlukan interaksi

dengan pengguna. Service juga akan tetap bekerja meskipun aplikasi dalam state destroy.

State service yang digunakan pada aplikasi ini adalah bound service. Bound service menyediakan interaksi antara *client* (dalam hal ini Navigation.class) dengan service tersebut. Service akan terikat (bound) dengan aplikasi saat komponen dalam aplikasi memanggil `bindService()`. Bound service akan menyediakan interaksi antara *client*-service yang memungkinkan terjadinya interaksi dari elemen dan komponen dalam aplikasi untuk berinteraksi dengan service seperti mengirimkan perintah, menerima hasil, dan melakukan proses lain.

Berikut adalah siklus dari Bound Service:



Gambar 15 Flowchart Bound Service. ©Dokumentasi Penulis

Pada permulaannya akan dilakukan persiapan pada Service sehingga siap untuk digunakan.

```
private final IBinder notifBinder = new NotificationBinder();
```

Baris kode diatas merupakan inisialisasi yang digunakan untuk memulai metode binding antara service dan java class. IBinder digunakan sebagai interface. Metode selanjutnya yang digunakan adalah `onBind()`. Sistem memanggil metode ini saat ada komponen lain yang ingin di-bind dengan service ketika komponen tersebut memanggil `bindService()`. Pada

implementasinya harus disediakan interface yang digunakan *client* (Navigation.class) untuk berkomunikasi dengan service dengan mengembalikan sebuah IBinder, seperti baris kode berikut:

```
@Override
public IBinder onBind(Intent intent) {
    return notifBinder;
}
```

Ketika tidak ada komponen atau *client* yang akan di-bind, maka interface ini harus mengembalikan null.

Untuk melakukan binding juga diperlukan sebuah class yang memiliki kemampuan untuk melakukan bind antara *client* dan service. Class ini akan mengikat keseluruhan service dan *client*. Berikut adalah baris komponen yang menyatakan class tersebut:

```
//create class that has an ability to bind the service and client(app)
public class NotificationBinder extends Binder{
    MyServiceNotification getService(){
        return MyServiceNotification.this;
    }
}
```

System service telah berhasil dibuat. Selanjutnya dilakukan inisialisasi pada main activity sehingga service dapat bekerja pada saat aplikasi diluncurkan. Pada Navigation.java dilakukan inisialisasi sebagai berikut:

```
Intent i = new Intent(this, MyServiceNotification.class);
bindService(i, notifConnection, Context.BIND_AUTO_CREATE);
```

Dari baris deskripsi kode diatas, bind service dimulai dengan deskripsi **BIND_AUTO_CREATE**. Pada Navigation.java juga dilakukan pengaturan pada koneksi service yang diatur sebagai berikut:

```
private ServiceConnection notifConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        NotificationBinder binder = (NotificationBinder) service;
        notificationService = binder.getService();
        isBind = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        unbindService(notifConnection);
        isBind = false;
    }
};
```

Sesuai dengan baris kode diatas, saat service sedang melakukan koneksi maka bind service akan dilakukan dan status bind adalah *true*. Sedangkan bila service melakukan pemutusan koneksi, maka bind service dengan *client* akan diputus dan status bind berubah menjadi *false*.

Setelah kulit sistem berhasil dibuat, selanjutnya akan dilakukan pengaturan pada service. Service pada aplikasi ini dimaksudkan sebagai proses background yang menaungi segala peran aplikasi sebagai *subscriber*. Aplikasi berperan sebagai *subscriber* untuk menerima pesan dari background berupa push notification dari unit, NIM, dan advertisement (berlaku ke semua pengguna, tidak diberlakukan personalisasi) dengan ketentuan *topic* sebagai berikut:

```
unit_client.subscribe("ubeacon/user/notification/unit/" + unit);
major_client.subscribe("ubeacon/user/notification/major/" + nim_notif);
advertise_client.subscribe("ubeacon/user/notification/advertise" );
```

Service, dengan bantuan Asynchronous Task melakukan koneksi ke *server* melalui MQTT via Internet. Penggunaan Asynchronous Task dimaksudkan untuk mengurangi waktu bagi pengguna untuk menunggu hingga aplikasi berhasil melakukan koneksi ke *server*. Asynchronous Task membuat aplikasi dapat menjalankan program secara parallel, sehingga sebelum aplikasi berhasil melakukan koneksi di background process, tampilan menu utama atau Sign In sudah bisa ditampilkan kepada pengguna.

Koneksi ke *server* dituliskan dalam onCreate pada Service. Saat digunakan Asynchronous Task maka baris deskripsi dapat dituliskan sebagai berikut:

```
@Override
public void onCreate() {
    new Connect().execute();
}
```

Fungsi `Connect()` pada baris deskripsi diatas mengimplementasikan kerja dari metode Asynchronous Task. Fungsi ini bertujuan untuk melakukan koneksi ke *topic* menggunakan MQTT sehingga uBeacon siap beroperasi sebagai *subscriber*. Adapun baris kode yang digunakan sebagai berikut:

```
private class Connect extends AsyncTask<String, Void, String> {
    ProgressDialog dialog;
    @Override
    protected String doInBackground(String... params) {
        final String nim_notif;
        Log.i(TAG, "Service onCreate");
        SharedPreferences pref =
        getApplicationContext().getSharedPreferences(Data.SPREF_NAME,
        Context.MODE_PRIVATE);
        String unit = pref.getString("Unit", "");
        System.out.println("List Unit pref" + unit);
        String nim = pref.getString("NIM", "");
        if (nim == null || nim.length() <= 0) {
            nim_notif = "";
        } else {
            nim_notif = nim.substring(0, 5);
        }

        String[] units = unit.split("\\s");

        int qos = 2;
```

```

String broker      = "tcp://test.mosquitto.org:1883";
String content     = "Message from MqttPublishSample";
String clientId    = "JavaSample";

try {
    String cacheDir = getCacheDir().getAbsolutePath();

    //broker and topic inisialisation
    subscribe_client = new MqttClient(BROKER_URL, clientId,
        new MqttDefaultFilePersistence(cacheDir));

    subscribe_client.setCallback(MyServiceNotification.this);

    opt = new MqttConnectOptions();
    opt.setCleanSession(true);
    opt.setKeepAliveInterval(keepAliveInterval);
    opt.setConnectionTimeout(10);

    //connect using Mqtt
    try {
        subscribe_client.connect(opt);

    } catch (MqttException e) {
        System.out.print("Connection error notif!! ");
    }
    if (subscribe_client.isConnected()) {
        //inisialisation callback method in this fragment
        //subscribe
        int size = units.length;
        for (int i=0; i<size; i++)
        {
            subscribe_client.subscribe("ubeacon/user/notification/unit/" + units[i]); }
        subscribe_client.subscribe("ubeacon/user/notification/major/" + nim_notif);
        subscribe_client.subscribe("ubeacon/user/notification/advertise" );
        MqttMessage mqttMessage = new MqttMessage(content.getBytes());
        //inisialisation Qos
        mqttMessage.setQos(qos);

        } else {
            subscribe_client.connect(opt);
        }

    } catch (MqttException e) {

    }
    return null;
}

@Override
protected void onPostExecute(String result) {
    Log.i(TAG, "PostExecute");
}

@Override
protected void onPreExecute() {
}
}

```

Implementasi dari Asynchronous Task memiliki tiga fungsi bawaan yaitu:

```

@Override
protected String doInBackground(String... params) {}

@Override
protected void onPostExecute(String result) {}

Override
protected void onPreExecute() {}

```


Pada fungsi `doInBackground(String... params)` dilakukan koneksi aplikasi ke *topic* menggunakan MQTT via Internet. Dilakukan koneksi ke tiga *topic* yang berbeda. Selanjutnya setelah koneksi berhasil dibuat, maka peran dari Asynchronous Task telah berakhir.

Selanjutnya operasi kembali ke Service. Saat koneksi telah berhasil dibuat, implementasi dari `MqttCallback` dilakukan pada fungsi `messageArrived(String s, MqttMessage mqttMessage)`. Pada fungsi ini akan dilakukan penyimpanan informasi pada `SharedPreferences`. Selanjutnya dibangun sebuah fungsi Notification yang memudahkan pengguna untuk mengetahui adanya informasi baru. Berikut adalah baris deskripsi yang menyatakan kerja dari notification:

```
Random random = new Random();
int m = random.nextInt(9999 - 1000) + 1000;

manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

Intent intent = new Intent(this,
com.example.astaripurnomo.ubeacon.Notification.class);
intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

PendingIntent pendingIntent = PendingIntent.getActivity(this, 1, intent,
PendingIntent.FLAG_UPDATE_CURRENT);

Notification.Builder builder = new Notification.Builder(this);
builder.setAutoCancel(false);
builder.setTicker(title);
builder.setContentTitle("Ubeacon Notification");
builder.setContentText(title);
builder.setSmallIcon(R.drawable.icon);
builder.setContentIntent(pendingIntent);
builder.setOngoing(true);
builder.setAutoCancel(true);
builder.setDefaults(Notification.DEFAULT_SOUND);

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
    builder.setSubText(notifications); //API level 16
}

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
    builder.build();
}

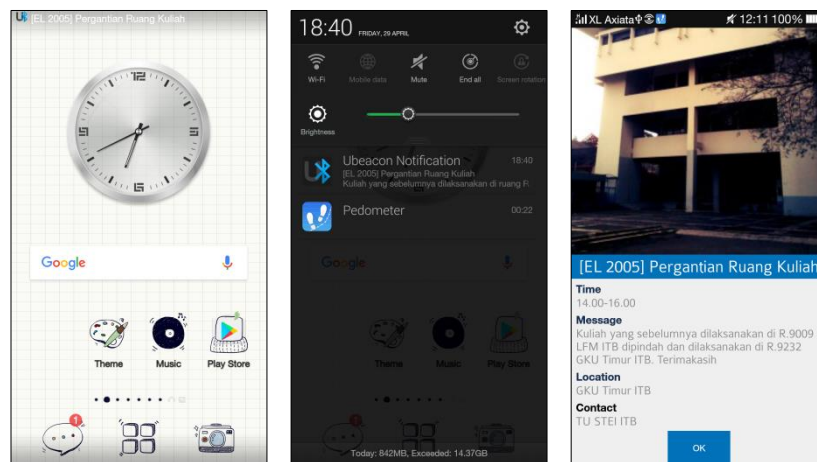
myNotification = builder.getNotification();
manager.notify(m, myNotification);
```

Agar Service tetap running meskipun aplikasi di-destroy, maka diberikan baris deskripsi berikut:

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    super.onStartCommand(intent, flags, startId);
    System.out.println("service on lagi");
    return START_STICKY;
}
```

Pada implementasinya digunakan bilangan random supaya notifikasi yang muncul tidak menumpuk dan meniadakan informasi yang sebelumnya telah muncul. Dilakukan pengaturan sehingga pengguna bisa dengan mudah

mengetahui adanya informasi baru. Pengaturan yang dilakukan antara lain, dilakukannya pengaturan pada suara, sehingga kapanpun notifikasi muncul, ada ringtone sebagai penanda bagi pengguna. Selanjutnya notifikasi juga dilengkapi dengan autocancel, sehingga saat pengguna menekan notifikasi, icon pada notifikasi akan segera menghilang tepat setelah ditekan. Saat pengguna menekan notifikasi maka akan terlihat tampilan Notification.class yang menunjukkan secara lengkap informasi yang dikirimkan oleh *server*. Berikut adalah tampilan dari push-notification:



Gambar 16 Tampilan antarmuka push-notification aplikasi uBeacon. ©Dokumentasi Penulis

3.5.5 Penyimpanan pada local memory *smartphone*

Mobile *application* uBeacon menggunakan local memory pada *smartphone* pengguna sebagai tempat penyimpanan sementara bagi data-data berupa data diri pengguna, Uri dari gambar yang telah dipilih pengguna sebagai *profile picture*, juga sebagai tempat untuk menyimpan state Sign In pengguna. Dalam hal ini, digunakan SharedPreference sebagai metode penyimpanan serta pengolahan data-data tersebut diatas.

Shared Preference merupakan metode yang digunakan untuk menyimpan sebuah set *key-value* dalam jumlah yang tidak terlalu banyak dan disimpan dalam memori local *smartphone* pengguna. SharedPreference diakses melalui sebuah *file* yang terdiri dari beberapa pasang *key-value* dengan metode sederhana untuk membaca dan menuliskannya. SharedPreference hanya dapat digunakan untuk membaca dan menulis pasangan *key-value*.

Pada aplikasi ini, digunakan beberapa *file* SharedPreference, sehingga dalam implementasinya digunakan metode `getSharedPreferences()`. Setiap

file SharedPreferences dibedakan oleh nama *filenya*, dan *file* tersebut dapat diakses (dipanggil) melalui `Context` dalam aplikasi.

Untuk menyimpan status Sign-In pengguna, maka dibuatlah sebuah *file* SharedPreferences yang dituliskan dalam baris deskripsi kode berikut:

```
//save the login state and store it to local memory
SharedPreferences pref = getSharedPreferences(PREF_NAME, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString(KEY_STATUS, "ok");
editor.commit();
```

Pada baris kode diatas, dibuat sebuah *file* SharedPreferences yang dapat diakses dengan variable `pref`. *File* SharedPreferences ini diatur agar bersifat private sehingga aplikasi lain dalam *smartphone* pengguna tidak dapat mengakses *file* ini. Untuk melakukan hal tersebut, digunakan `MODE_PRIVATE`. *File* SharedPreferences harus dinamai dengan penanda yang unik sehingga mudah untuk dikenali oleh aplikasi yang telah dibuat. Nama yang digunakan pada *file* SharedPreferences yang dibuat adalah `"UBEACON_PREF"` yang diinisialisasi sebagai:

```
private static String PREF_NAME = "UBEACON_PREF";
```

Selanjutnya setelah *file* SharedPreferences berhasil disiapkan, maka untuk menulis pada *file* SharedPreferences harus dibuat sebuah `SharedPreferences.Editor` dengan memanggil `edit()` di *SharedPreferences*. Untuk menyimpan status Sign In pengguna, pada *file* SharedPreferences disiapkan sebuah *key*:

```
private static String KEY_STATUS = "login_status";
```

Selanjutnya *key* tersebut diisi dengan *value* `"ok"`. Untuk mengakhiri penulisan pada *file* Shared preference maka diberikan deskripsi kode `editor.commit();`. Baris deskripsi kode untuk penulisan status Sign In pengguna dilakukan tepat saat Sign In berhasil dilakukan.

Sesaat setelah aplikasi di destroy dan diluncurkan lagi, maka secara otomatis akan diluncurkan *SignIn.class* sebagai default. Pada saat class ini diluncurkan, *file* SharedPreferences ini dipanggil untuk dilakukan cek pada status *SignIn* dalam baris kode dibawah:

```
SharedPreferences pref = getSharedPreferences(PREF_NAME, MODE_PRIVATE);
String status = pref.getString(KEY_STATUS, "null");
System.out.println("status pref: " + status);
if (status.equals("ok")) {
```

```

Intent intent = new Intent(this, Navigation.class);
startActivity(intent);
finish();
}

```

Bila *value* dari status adalah "ok" maka aplikasi akan diteruskan pada menu utama yaitu Navigation.class. sedangkan bila didapati *value* dari state ini adalah "null" maka proses akan tetap dilanjutkan pada class SignIn.

Saat Sign In berhasil dilakukan *server* mengirimkan status kepada aplikasi beserta dengan data diri pengguna yang sebelumnya telah diisikan pada saat proses Sign Up. Data ini juga disimpan pada local memory dengan *file* SharedPreference yang berbeda dari *file* yang digunakan untuk menyimpan status Sign-In. deskripsi kode pada penyimpanan data diri pengguna dituliskan sebagai berikut:

```

//keep the data from server in local memory using sharedpreferences
SharedPreferences pref = getSharedPreferences(Data.SPREF_NAME, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString("Name", Name);
editor.putString("Email", Email);
editor.putString("Username", Username);
editor.putString("Password", Password);
editor.putString("Birthdate", Birthdate);
editor.putString("NIM", NIM);
editor.putString("Occupation", Occupation);
editor.putString("Interest", Interest);
editor.putString("Unit", Unit);
editor.commit();

```

Data diri pengguna disimpan dalam pasangan *key-value* seperti diatas. Sebelumnya data-data ini diperoleh dari proses *parsing* paket JSON yang dikirim sebagai balasan dari *server*. Begitu *value* dari masing-masing *key* pada JSON *file* berhasil diperoleh dalam bentuk String, nilai tersebut di-*passing* dan disimpan dalam *file* SharedPreference.

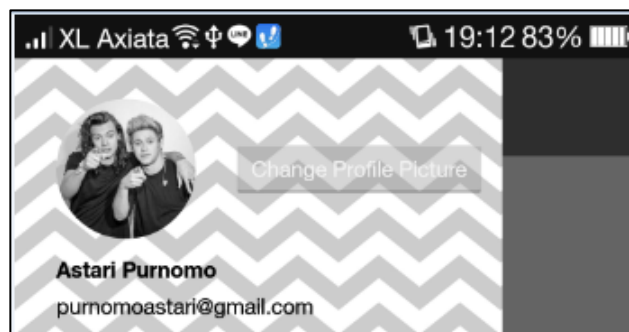
Penyimpanan data diri pengguna dalam local memory menggunakan SharedPreference ini bertujuan untuk validasi dan personalisasi pengguna. Validasi digunakan saat pengguna hendak melakukan edit *profile* pada data dirinya. Untuk memastikan bahwa hanya pengguna yang melakukan edit *profile* maka validasi dilakukan dengan meminta pengguna untuk melakukan *input* password. Password yang dimasukan oleh pengguna selanjutnya dibandingkan dengan password yang telah disimpan aplikasi pada SharedPreference. Password pada SharedPreference dipanggil dengan deskripsi sebagai berikut:

```

SharedPreferences pref = getSharedPreferences(Data.SPREF_NAME, MODE_PRIVATE);
password = pref.getString("Password", "");

```

Untuk memanggil dan menerima data yang telah disimpan dalam *file* SharedPreference, *file* SharedPreference diinisialisasi kemudian dipanggil menggunakan deskripsi `getString("Password", "");`. *Value* yang diperoleh kemudian disimpan dalam variabel `password` yang bertipe String. Variable ini yang selanjutnya dibandingkan dengan *input* yang dimasukan oleh pengguna. Selanjutnya *file* ini juga dipanggil untuk mendapatkan *value* dari email pengguna dan nama pengguna untuk ditampilkan pada header Navigation.class. Berikut adalah hasil *parsing* dari *file* SharedPreference:



Gambar 17 Tampilan pada navigation bar aplikasi uBeacon. ©Dokumentasi Penulis

Selanjutnya SharedPreference juga digunakan dalam melakukan komunikasi antara service-activity. Pada hal ini, dilakukan komunikasi berupa passing data dari MyServiceNotification.class menuju Notificaton.class. Saat aplikasi menerima paket data dari *server* berupa informasi dari unit dan jurusan pada Service selanjutnya data ini disimpan dalam SharedPreference. Data pada SharedPreference kemudian dipanggil pada Notification.class dan ditampilkan sesuai dengan *layout* yang telah diatur dalam Notification.xml. Berikut adalah kode deskripsi saat dilakukan penyimpanan data pada SharedPreference di MyServiceNotification.class

```
SharedPreferences pref =
getApplicationContext().getSharedPreferences(Data.SPREF_INFO, MODE_APPEND);
SharedPreferences.Editor editor = pref.edit();
editor.putString("Image_url", infourl);
editor.putString("Title", title);
editor.putString("Notifications", notifications);
editor.putString("Time", time);
editor.putString("Location", location);
editor.putString("Contact", contact);
editor.commit();
```

Selanjutnya berikut adalah baris deskripsi saat data dalam *file* SharedPreference diambil dan di *parsing* dalam *layout*:

```
SharedPreferences pref = getSharedPreferences(Data.SPREF_INFO, MODE_PRIVATE);
Image = pref.getString("Image_url", "");
```

```
Title = pref.getString("Title", "");
Time = pref.getString("Time", "");
Notif = pref.getString("Notifications", "");
Location = pref.getString("Location", "");
Contact = pref.getString("Contact", "");
```

SharedPreferences juga digunakan untuk menyimpan Uri dari gambar yang dipakai pengguna sebagai *profile* picture. Setelah pengguna berhasil memilih gambar, Uri dari gambar akan disimpan dalam SharedPreferences seperti baris kode dibawah:

```
SharedPreferences pref = getSharedPreferences(PREF_PROFPIC, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString(KEY_PROFPIC, imageUri.toString());
editor.commit();
```

File ini akan dipanggil setiap saat pengguna berhasil masuk ke Navigaation.class, sehingga gamabr yang dipilih pengguna sebagai *profile* picture akan tetap tersimpan.

Selanjutnya untuk proses Log out, *file* SharedPreferences yang sebelumnya telah dibuat pada saat dilakukan penyimpanan status Sign In dipanggil dan ditimpa dengan *value* “null” sehingga saat aplikasi kembali diluncurkan dan baris kode cek status berjalan maka *value* "null" akan membuat aplikasi meminta pengguna untuk melakukan SignIn. Berikut adalah kode yang dituliskan saat Log out dilakukan:

```
SharedPreferences pref = getSharedPreferences(PREF_NAME, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString(KEY_STATUS, "null");
editor.commit();
```

BAB IV

HASIL PENGUJIAN

4.1 Aspek dan Skenario Pengujian

Dalam melakukan pengujian terhadap sub-sistem mobile *application* maka dilakukan pengujian per-blok dari sub-sistem ini. Pengujian per-blok ini bertujuan untuk melakukan pengecekan dan perbaikan pada setiap bagian yang menyusun sistem ini. Dengan dilakukannya pengujian ini diharapkan sub-sistem dapat diintegrasikan dengan baik.

Pengujian pada sub-sistem mobile *application* dilakukan pada konfigurasi software dan hardware sebagai berikut:

1. Personal computer Intel Core i7-4720HQ up to 3.6 GHz, memori RAM 4GB dengan Window 8.1 dan koneksi internet.
2. GUI dijalankan pada *Smartphone* Oppo Find 7a-x9006 dengan serial number 96cf428d dengan OS Android version 4.3.

Terdapat beberapa batasan yang digunakan untuk melakukan pengujian sub-sistem mobile *application*, diantaranya sebagai berikut:

1. Aplikasi dirancang pada sebuah project manager berupa software Android Studio 1.5.1 dengan JRE dan JDK 1.8.0_72
2. Pengguna mengunduh aplikasi pada *smartphone* melalui running project pada project manager yang digunakan.
3. Pengguna memasukkan *input* data berupa nama, username, password, e-mail, jenis kelamin, tanggal dan tahun lahir, pekerjaan, dan minat sesuai dengan kolom yang telah disediakan pada *layout* aplikasi. Berikut batasan *inputan* dari pengguna:
 - Nama, jenis kelamin, pekerjaan, dan minat dimasukkan dalam kombinasi huruf.
 - Username, password, e-mail boleh dimasukkan dengan menggunakan kombinasi huruf dan angka.
 - Tanggal dan tahun lahir dimasukkan dengan menggunakan kombinasi angka.

4. *Smartphone* yang digunakan oleh pengguna terkoneksi dengan internet.
5. *Smartphone* yang digunakan oleh pengguna telah diaktifkan fitur bluetoothnya.
6. *Smartphone* yang digunakan oleh pengguna memiliki Bluetooth minimal versi 4.0.

Pada pengujian sub-sistem mobile *application* dilakukan beberapa prosedur pengujian sebagai berikut:

1. Pengujian fitur penyimpanan dan pengelolaan data pada *local memory smartphone*.
2. Pengujian *Graphic User Interface(GUI)* sebagai interaksi antarmuka dengan pengguna.
3. Pengujian fitur personalisasi penggunaan aplikasi dalam bentuk akun pengguna (*user account*).
4. Pengujian *Push-Notification* sebagai fitur personalisasi pengguna.

4.2 Prosedur dan Hasil Pengujian

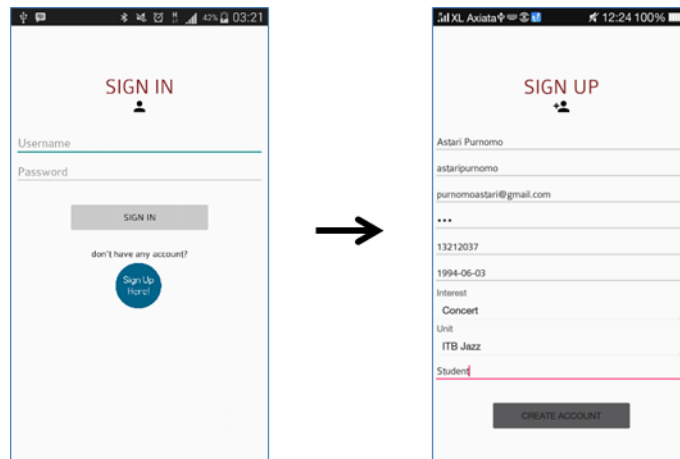
Pengujian pada fitur personalisasi pengguna dilakukan kedalam beberapa bagian.

4.2.1 Fitur User Personalization

Pengujian pada fitur personalisasi pengguna dilakukan kedalam beberapa bagian yaitu Sign-In, Sign-Up, Edit *Profile*, Logout, SharedPreference.

4.2.1.1 Sign-Up

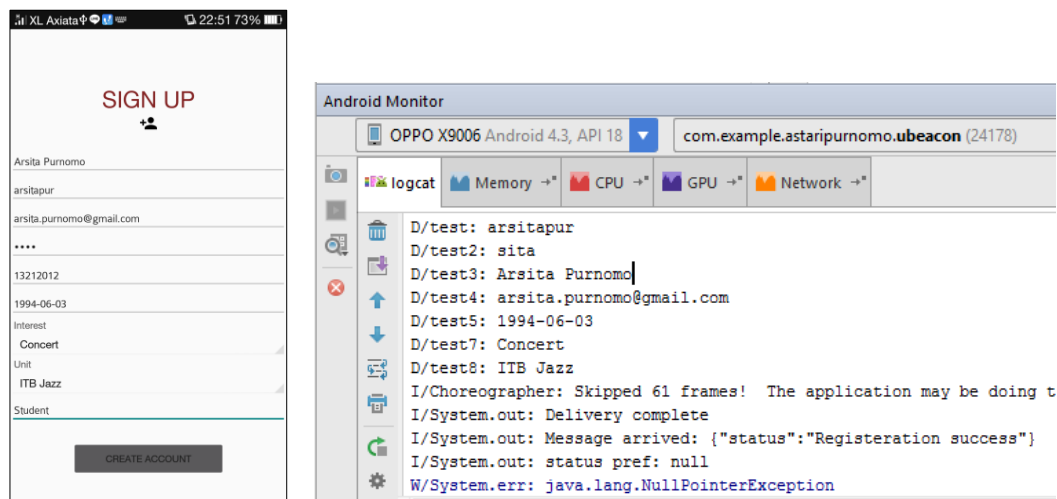
Pada bagian ini, pengguna belum memiliki akun pada aplikasi uBeacon, sehingga pengguna perlu untuk melakukan pendaftaran atau registrasi akun yang dapat langsung dilakukan pada aplikasi uBeacon. Cara untuk mendaftarkan akun dapat dilakukan dengan cara menekan tombol Sign-Up sesuai pada gambar di berikut :



Gambar 18 Halaman Sign-In aplikasi uBeacon. ©Dokumentasi Penulis

Adapun, untuk membuat akun diperlukan koneksi internet pada *smartphone* pengguna untuk mengirimkan dan mencocokkan data dengan *database* pada *server* Ubeacon.

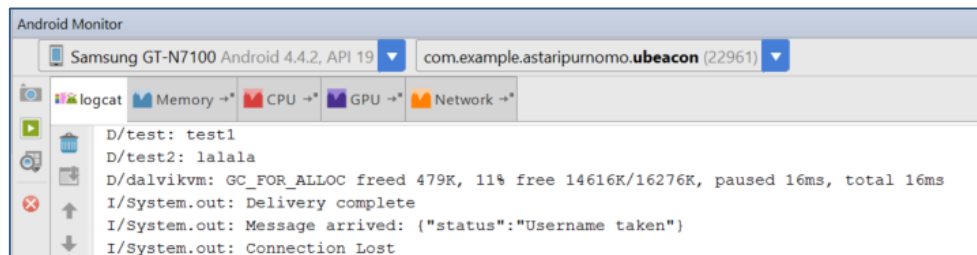
Pada tahap ini, pengguna diminta untuk mengisi sebuah form yang berisi antara lain *Name*, *Username*, *E-mail*, *Password*, *NIM*, *Birthday*, *Interest*, *Unit*, dan *Occupation*. Selanjutnya setelah tombol *CREATE ACCOUNT* diklik, maka data informasi akan dikirimkan dan dicocokkan dengan *database* pada *server* yang telah disediakan. Logcat beserta contoh pengisian pada bagian ini adalah sebagai berikut:



Gambar 19 Tampilan log pada bagian Sign-Up aplikasi uBeacon. ©Dokumentasi Penulis

Pada gambar tersebut dilakukan *input* username berupa “arsitapur” dan beberapa data *input* lainnya dan berhasil dengan ditandai nilai *status* nya adalah *Registraton success*. Jika dilakukan pengujian dengan melakukan *input* user pada

saat Sign Up berupa “arsitapur” lagi, maka registrasi akan gagal dan akan ditampilkan log pada software Android Studio sebagai berikut :



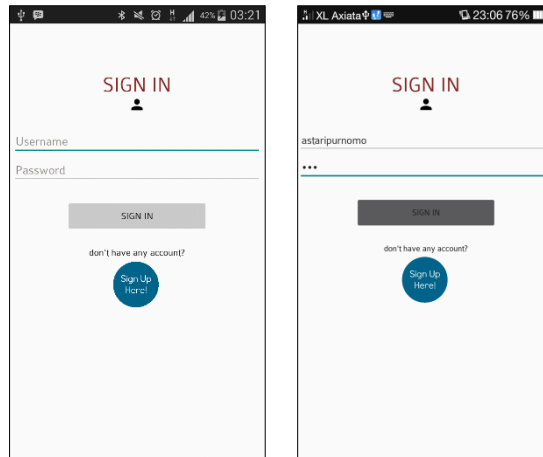
Gambar 20 Tampilan kegagalan registrasi aplikasi uBeacon. ©Dokumentasi Penulis

Berdasarkan hal di atas, jika username sudah terdaftar pada *database*, maka data *status* akan didapatkan *Username taken*. Jika akun sudah terdaftar, maka pengguna dapat masuk ke keadaan selanjutnya, yaitu keadaan dimana user sudah memiliki akun namun belum melakukan Sign In di dalam aplikasi Ubeacon. Pada kondisi ini, setelah Sign Up berhasil dilakukan, state *layout* akan berubah dari Sign Up ke state *layout* Sign In.

Proses Sign-Up telah berjalan dengan baik, aplikasi mampu mengubah *input* dari pengguna menjadi *JSON file*. Aplikasi juga telah mampu melakukan koneksi ke broker menggunakan MQTT. Aplikasi juga berhasil mengirimkan paket JSON ke alamat *topic* yang dimaksud. Aplikasi juga telah berhasil menerima balasan dari *server* dan menampilkan setiap perintah dalam aplikasi dengan baik.

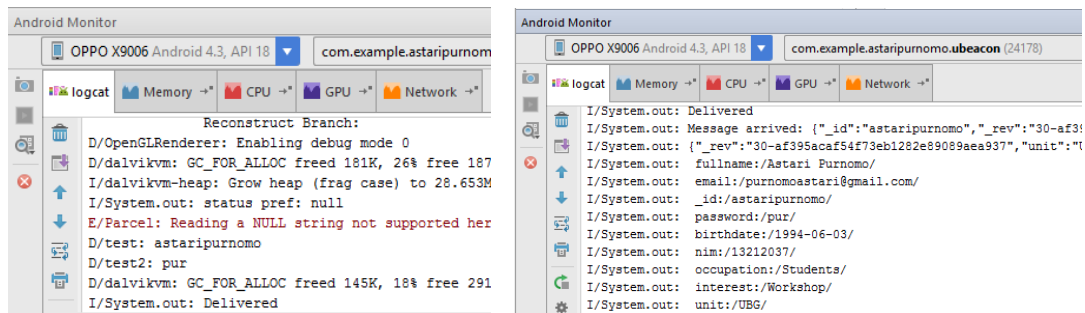
4.2.1.2 Sign-In

Pada bagian ini, pengguna telah memiliki sebuah akun pada uBeacon dengan kombinasi username dan password yang dapat digunakan untuk melakukan sign in. Username dan password ini menjadi identitas pengguna dalam aplikasi. Selanjutnya, pengguna akan diminta untuk melakukan Sign In ke dalam aplikasi dengan mengisi kombinasi *username* dan *password* yang telah dibuat. Berikut form pengisian dari Sign In yang telah dibuat serta contoh pengisian sesuai dengan akun yang telah dibuat :



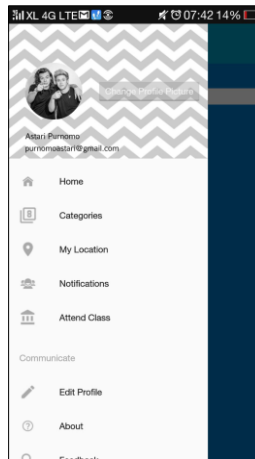
Gambar 21 Tampilan Sign-In berhasil dilakukan di aplikasi uBeacon. ©Dokumentasi Penulis

Pada gambar di atas dilakukan pengujian dengan memasukkan *input* yang tepat dengan kombinasi Username: *astaripurnomo* dan Password: *pur*. Dengan memasukkan *input* yang benar, maka aplikasi akan masuk ke dalam beranda (*homepage*) utama dari aplikasi serta menghasilkan log pada Android Studio seperti sebagai berikut:



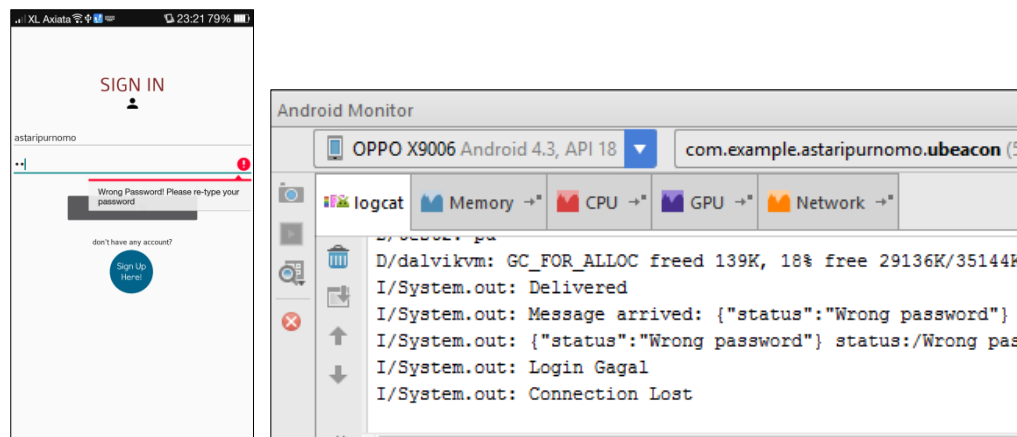
Gambar 22 Tampilan logcat saat Sign-In aplikasi uBeacon. ©Dokumentasi Penulis

Pada proses di atas, dengan menekan tombol *Sign In* pengguna akan mengirimkan data tersebut dalam bentuk *JSON file* ke *server* melalui MQTT. Setelah pengiriman ke *server* dengan alamat *topic* tertentu berhasil dilakukan, ditampilkan log “Delivered” sesuai tampilan diatas. Saat aplikasi berhasil menerima balasan dari *server* maka akan ditampilkan log “Message arrived....”. Log diatas menunjukkan aplikasi mampu mengambil *value* dari setiap *key* yang dibawa oleh *back-end*. *Value* tersebut selanjutnya disimpan dalam *file* *SharedPreferences*. Jika prses Sign-In berhasil dilakukan maka tampilan akan berubah menjadi tampilan menu utama, sebagai berikut:



Gambar 23 Navigation Bar aplikasi uBeacon. ©Dokumentasi Penulis

Sedangkan bila sign in gagal dilakukan, maka tampilan log dan aplikasi adalah sebagai berikut:



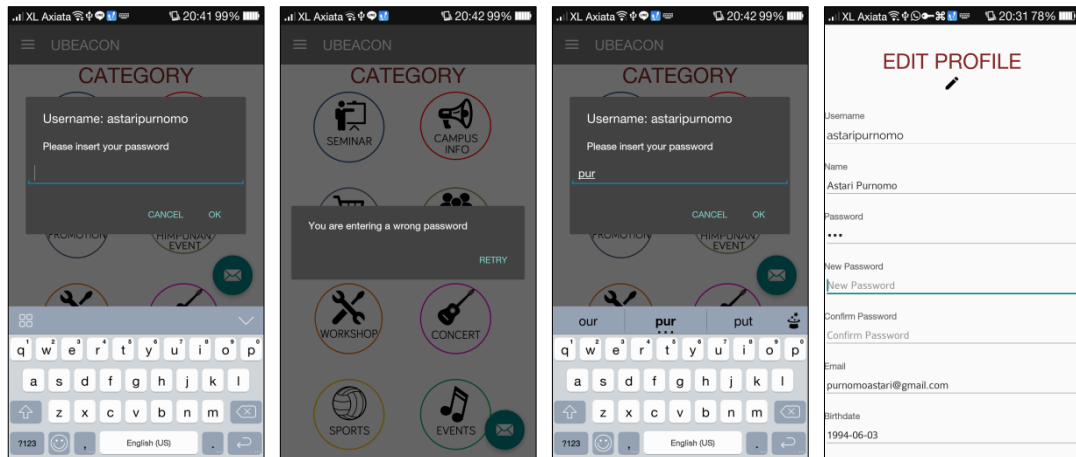
Gambar 24 Tampilan Sign-In yang gagal dilakukan pada aplikasi uBeacon. ©Dokumentasi Penulis

Jika kombinasi username dan password yang dimasukan oleh pengguna salah, maka *server* akan memberikan balasan status seperti diatas. Hal ini membuat aplikasi menghasilkan pesan error seperti gambar diatas.

Nama, e-mail, dan gambar yang pengguna pilih akan ditampilkan pada navigation bar pada uBeacon. Saat proses Sign In berhasil dilakukan, state aplikasi akan diubah dan disimpan dalam Shared Preference sehingga saat pengguna menutup aplikasi dan selanjutnya aplikasi di-launch kembali, pengguna tidak perlu melakukan proses Sign In lagi selama proses Log Out belum dilakukan. Selama pengguna tidak melakukan Log Out pada sebuah *smartphone*, maka pengguna akan terus menjalankan akun yang sama selama belum melakukan Log Out.

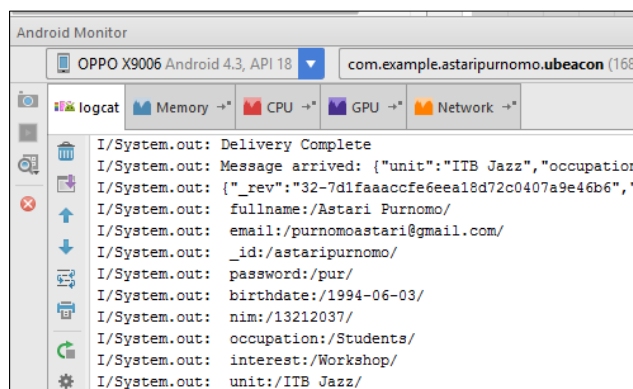
4.2.1.3 Edit Profile

Pada bagian ini, pengguna dapat melakukan perubahan pada data yang sebelumnya telah disimpan dalam *database*. Aplikasi akan melakukan perbandingan variable bertipe String saat hendak membuka tampilan Edit Profile. Untuk melakukan tersebut, aplikasi harus mampu memanggil password yang sebelumnya telah disimpan dalam *file* SharedPreferences.



Gambar 25 Tampilan Edit Profile aplikasi uBeacon. ©Dokumentasi Penulis

Dengan berhasilnya dilakukan pergantian *layout* dari Navigation.class menuju Edit Profile maka fungsi SharedPreferences dalam aplikasi berjalan dengan baik. Setelah dilakukan perubahan pada data diri pengguna, dan dikirimkan ke *server*, diperoleh tampilan log sebagai berikut:

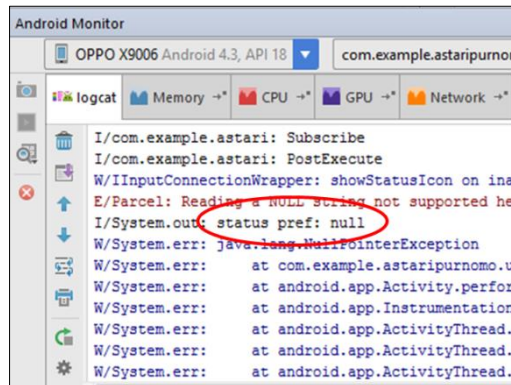


Gambar 26 Edit Profile berhasil dilakukan aplikasi uBeacon. ©Dokumentasi Penulis

Dari hasil log diatas diperoleh bahwa aplikasi berhasil mengirimkan pake JSON ke *topic* dan berhasil juga memperoleh balasan dari *server*. Hal ini ditandai dengan berubahnya tampilan aplikasi kembali ke Navigation.class.

4.2.1.4 Logout

Pada bagian ini, saat pengguna melakukan logout maka aplikasi akan melakukan akses pada *file* *SharedPreferences* dan mengubah *value* dari *file* menjadi “null”. Hal ini dapat dilihat pada tampilan log Android sebagai berikut:



Gambar 27 Logout aplikasi uBeacon. ©Dokumentasi Penulis

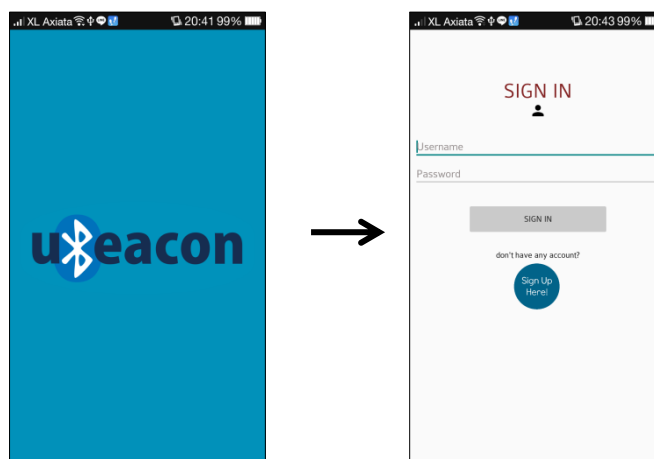
Saat aplikasi di-destroy dan pengguna meluncurkan aplikasi kembali maka aplikasi akan masuk ke state Sign-In, bukan ke menu utama. Fungsi aplikasi dalam menjalankan log out telah berhasil dilakukan dengan baik.

4.2.1.5 SharedPreferences

SharedPreferences digunakan sebagai tempat penyimpanan dan pengelolaan informasi di local memory *smartphone* pengguna. Pengujian yang dilakukan adalah sebagai berikut:

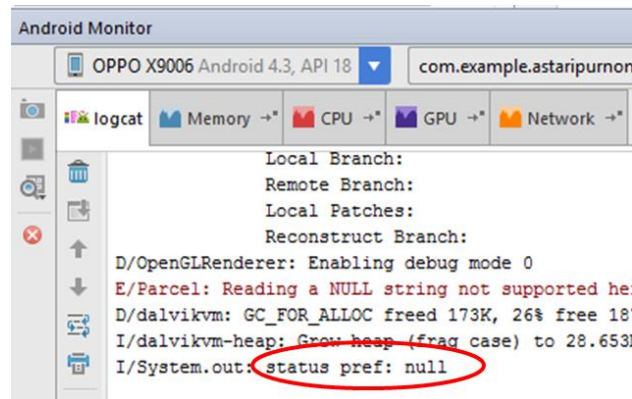
User's state

Untuk melakukan pengujian pada status pengguna akan digunakan sebuah skenario. Pada mulanya setelah aplikasi berhasil diinstal pada *smartphone* pengguna, tampilan yang diperoleh oleh pengguna adalah sebagai berikut:



Gambar 28 Tampilan awal aplikasi uBeacon. ©Dokumentasi Penulis

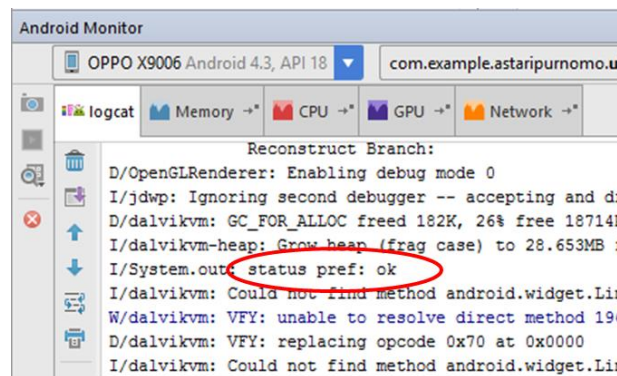
Berikut adalah status pengguna dalam *file* SharedPreferences:



Gambar 29 Status awal Shared Preference aplikasi uBeacon. ©Dokumentasi Penulis

Pada kondisi ini, diperoleh bahwa pengguna masih berada pada state awal sehingga dapat dilihat bahwa *file* SharedPreferences menampilkan *value* “null”.

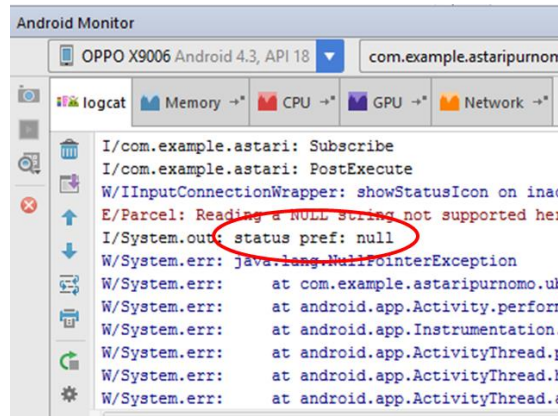
Selanjutnya saat Sign-In berhasil dilakukan, logcat menunjukkan tampilan:



Gambar 30 Sign-In berhasil dilakukan. ©Dokumentasi Penulis

Saat Sign-In berhasil dilakukan, maka tampilan logcat pada *file* SharedPreferences berubah seperti gambar diatas. Selanjutnya saat aplikasi di-destroy dan kembali diluncurkan, tampilan logcat masih menunjukkan *value* dari *file* SharedPreferences yang sama. Hal ini membuktikan bahwa status pengguna pada Sign-In telah berjalan dengan baik.

Saat log out dilakukan oleh pengguna, tampilan pada logcat menunjukkan:



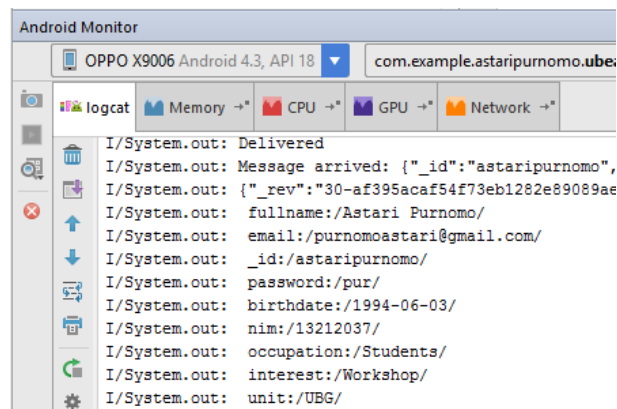
Gambar 31 Logcat untuk Logout aplikasi uBeacon. ©Dokumentasi Penulis

Log out menampilkan perubahan dari *file* SharedPreferences kembali ditulis dengan “null”. Hal ini menunjukkan bahwa alur dari user’s state telah berjalan dengan baik.

Parsing data dari server

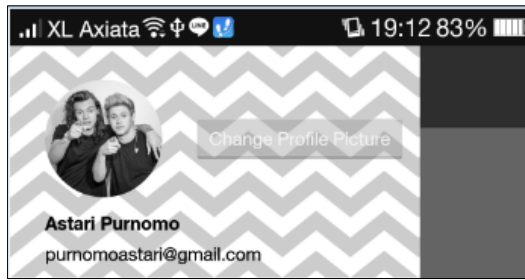
Pengujian penyimpanan data pengguna di *file* SharedPreferences ditunjukkan dalam logcat sebagai berikut:

Penyimpanan data diri pengguna saat Sign-In berhasil dilakukan

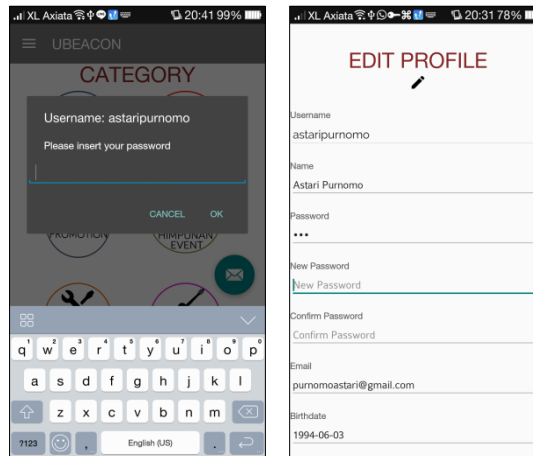


Gambar 32 Logcat penyimpanan data pada local memory. ©Dokumentasi Penulis

Dari tampilan logcat ditunjukkan bahwa setelah diterima balasan dari *server*, setiap *value* dari *key* yang dibawa oleh *server* disimpan dalam *file* SharedPreferences. Tampilan logcat menunjukkan bahwa restorasi data diri pengguna berhasil dilakukan. Selanjutnya akan ditunjukkan beberapa tampilan yang menunjukkan kemampuan aplikasi dalam melakukan *parsing* data dari *file* SharedPreferences sebagai berikut:



Gambar 33 Hasil *parsing* dari local memory. ©Dokumentasi Penulis



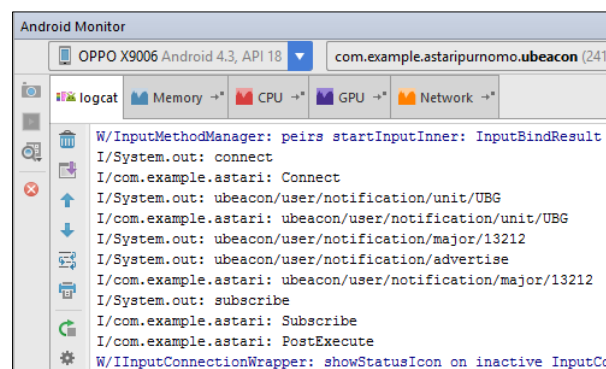
Gambar 34 Edit Profile aplikasi uBeacon. ©Dokumentasi Penulis

Beberapa tampilan diatas menunjukan bahwa *parsing* data dari informasi yang sebelumnya disimpan dalam *file* SharedPreference dan kemudian di-*parsing* telah berhasil dilakukan dengan baik.

4.2.2 Fitur Push-Notification

Pengujian pada fitur push-notificaton uBeacon dilakukan beberapa rincian sebagai berikut.

4.2.2.1 Service

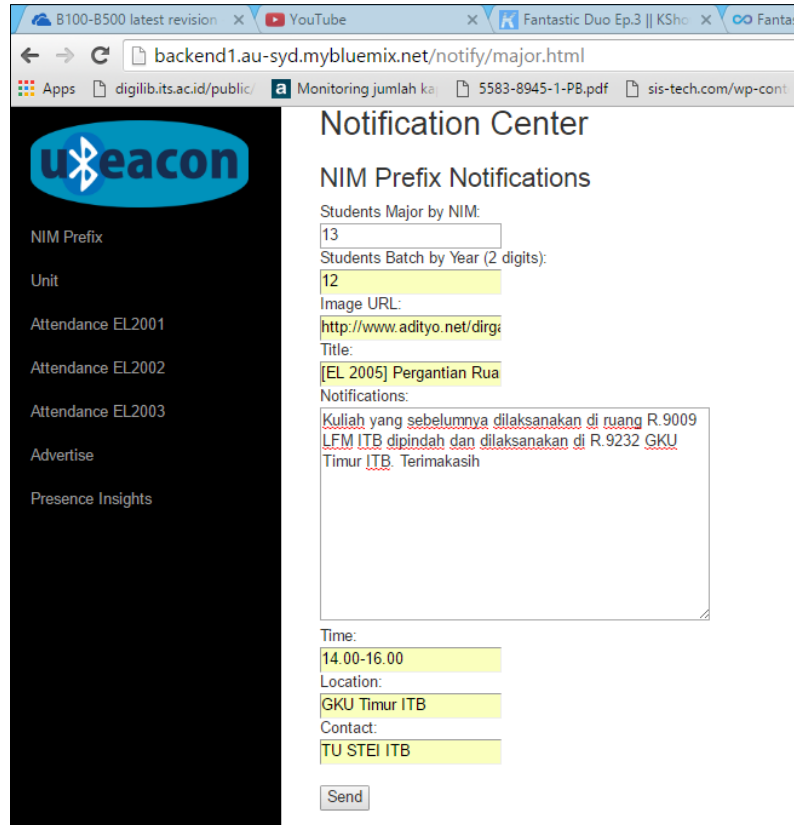


Gambar 35 Logcat untuk Service pada notification. ©Dokumentasi Penulis

Seperti tampilan diatas, saat Navigation.class berjalan aplikasi akan melakukan *subscribe* ke tiga alamat *topic* diatas. Dalam log dapat dilihat bahwa

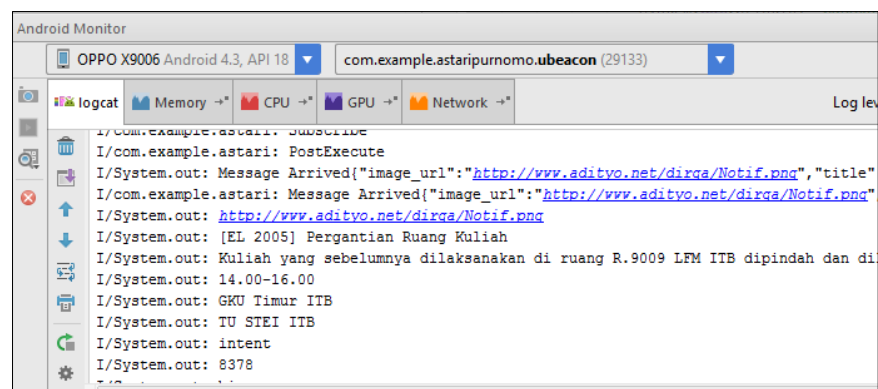
proses koneksi menggunakan MQTT dapat berjalan dengan baik, aplikasi berhasil menempatkan dirinya sebagai *subscriber* dan siap menerima pesan dari *server*.

Selanjutnya dapat dilakukan pengetesan melalui frontend uBeacon seperti terlihat pada tampilan dibawah:



Gambar 36 Tampilan front-end untuk fitur notification. ©Dokumentasi Penulis

Sesaat setelah pesan tersebut dikirim, diperoleh tampilan log sebagai berikut:

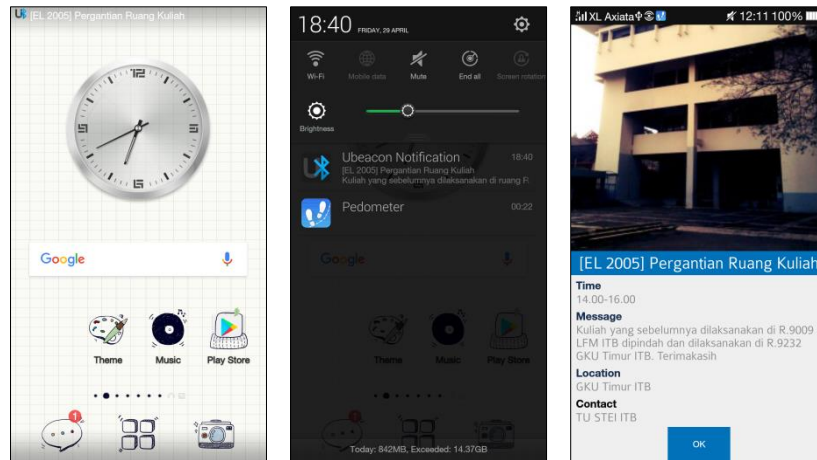


Gambar 37 Informasi yang diterima oleh aplikasi. ©Dokumentasi Penulis

Pada log ditunjukkan bahwa aplikasi berhasil menerima paket data JSON dari *server*. Selanjutnya paket tersebut disimpan dalam *file* SharedPreference.

4.2.2.2 Notification

Proses pada service menghasilkan tampilan aplikasi sebagai berikut:



Gambar 38 Tampilan aplikasi uBeacon saat push notification. ©Dokumentasi Penulis

Pada kondisi ini, aplikasi menarik *value* dari *file* SharedPreference dan menampilkannya pada *layout* Notification.class (dilakukan *parsing*). Dari hasil yang diperoleh diatas, didapati bahwa fitur push-notificaton dapat berjalan dengan baik.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Aplikasi uBeacon memiliki antar muka yang mudah dioperasikan. Fitur User Personalization mempermudah pengguna dalam memperoleh informasi yang bersifat personal. Pengguna dapat memperoleh informasi berdasarkan keanggotaan pengguna di unit dan program studi. Komunikasi antara aplikasi dan *back-end* menggunakan protokol komunikasi MQTT menguntungkan karena konsumsi dayanya yang rendah

Aplikasi ini juga membantu pihak kampus dalam mengatur dan melakukan pengawasan terhadap aliran informasi yang terdistribusi dalam kampus. Selain itu aplikasi ini juga membantu pihak kampus dalam mendistribusikan informasi yang penting dan mendesak secara cepat dan massal.

5.2 Saran

Saran untuk pengembangan aplikasi kedepannya adalah perbaikan pada koneksi MQTT. Dapat digunakan Broker pribadi agar koneksi lebih stabil lagi. Selain itu dapat pula dikembangkan beberapa fitur lain yang mampu menjawab berbagai permasalahan di kampus.

DAFTAR PUSTAKA

- [1] Ministry of Education and Culture “Indonesia Educational Statistics In Brief = Ringkasan Statistik Pendidikan Indonesia” link address:
<http://kemdikbud.go.id/kemdikbud/dokumen/BukuRingkasanDataPendidikan/Final-In-Brief-1112.pdf>
- [2] M. Butler, “Android: Changing the Mobile Landscape”, Pervasive Computing, (2011), pp. 4-7.
- [3] B. Proffitt, “Open Android-For better and for worse”, Spectrum, (2011), pp. 22– 24.
- [4] Tang Konglong, Wang Yong, Liu Hao, Yanxiu Sheng, Xi Wang, Zhiqiang Wei “Design and Implementation of Push Notification System Based on the MQTT Protocol”, Department of Computer Science, Ocean University of China, Qingdao, China.