BAB IV
PEMBAHASAN

Pada bab ini akan dijelaskan mengenai implementasi sistem yang telah dirancang pada bab sebelumnya. Pembahasan implementasi disini meliputi *development* aplikasi web admin dan aplikasi Android dengan serangkaian *tool* dan teknologi yang telah dibahas pada bab sebelumnya. Dimana secara keseluruhan sistem terdiri dari aplikasi web administrator dan aplikasi Android pengguna. Keduanya terintegrasi dengan sebuah layanan Google yakni *Firecase Cloud Messaging* yang memungkinkan aplikasi web dapat mengirimkan notifikasi ke aplikasi Android pengguna.

Sebagai basis dari sistem aplikasi web digunakan CMS (*Content Management System*) OpenCart yang telah memiliki *framework* tersendiri. Sistem aplikasi web akan dibangun diatas *framework* OpenCart tentu dengan menghilangkan banyak dari fitur di dalamnya.

Kemudian pada tahap pembuatan aplikasi Android, digunakan *PhoneGap* (http://phonegap.com) yang merupakan *framework* yang menjembatani pembuatan aplikasi Android dengan kode sumber berbasis web. Untuk keperluan ini juga digunakan Node.js yang merupakan dependensi dari *PhoneGap* dan JQuery Mobile yang digunakan untuk meningkatkan interaktifitas antarmuka pada aplikasi Android.

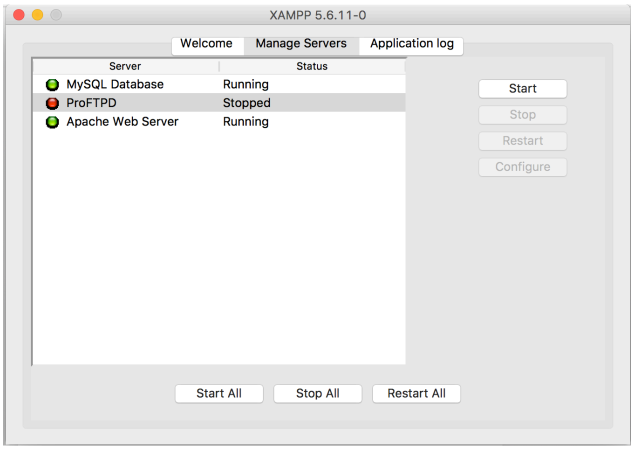
4.1 Implementasi Aplikasi Web Administrator

Pada bagian ini akan dijelaskan lebih spesifik mengenai implementasi aplikasi web administrator mulai dari lingkungan kerja, *tools* dan teknologi yang digunakan.

4.1.1 Lingkungan Kerja

Untuk membuat sebuah aplikasi berbasis web tentu dibutuhkan sebuah web *server*, *browser* dan *teks editor* untuk membuat / edit kode. Sebuah *server* web yang telah terpasang PHP dan MySQL dapat dibuat dengan mudah di komputer sendiri dengan bantuan XAMPP (https://www.apachefriends.org). Sementara untuk keperluan *development* ini digunakan peramban *Mozilla Firefox* (https://www.mozilla.org) dan Bracket teks editor (http://brackets.io).

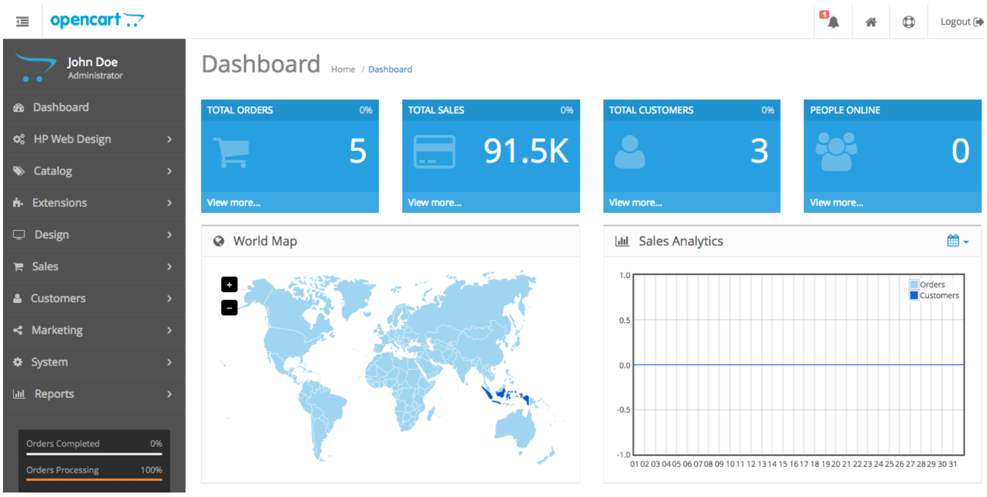
MySQL Workbench digunakan sebagai *MySQL Client* untuk membuat desain tabel berupa EER (*Enhanced Entity-Relationship*). Selain itu juga digunakan *Sequel PRO*, yakni *platform* MySQL Client yang digunakan untuk manajemen *database* dan tabel didalamnya. Perangkat lunak ini akan digunakan pada tahap *development* yang berhubungan dengan *database*. Manajemen *local server* pada XAMPP diperlihatkan pada Gambar 4.1.



Gambar 4.1 Manajemen *local server* pada XAMPP

4.2.1.1 Instalasi CMS OpenCart

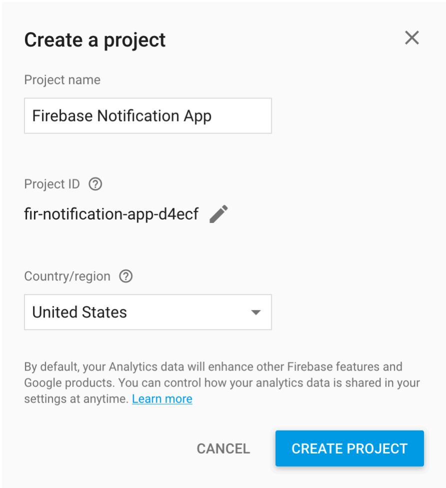
Sebelum memulai *development* aplikasi web administrasi, terlebih dahulu dipasang *basic system* untuk aplikasi web yang menggunakan CMS OpenCart. Kode sumber lengkap dapat diunduh gratis via situs resminya (http://opencart.com). Installasi membutuhkan *server* yang aktif dan *database* yang telah dibuat sebelumnya. Halaman *default* administrasi OpenCart ditunjukkan oleh Gambar 4.2.



Gambar 4.2 Halaman administrasi OpenCart

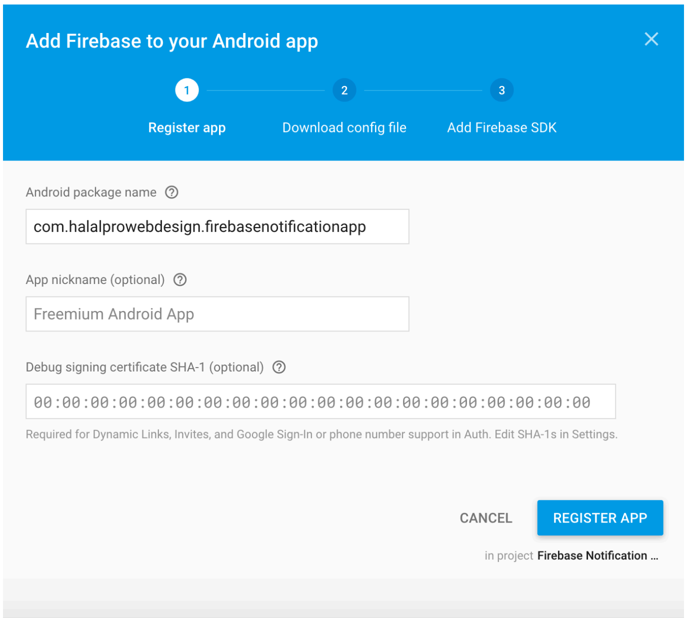
4.1.1.2 Membuat Proyek Firebase

Pembuatan proyek *Firebase* ini dibutuhkan agar sistem dapat menggunakan layanan FCM. Registrasi dilakukan via URL: https://console.*Firebase*.google.com/ yang mengharuskan *login* dengan akun Google. Setelah itu buat proyek baru untuk aplikasi Android dengan nama tertentu. Lalu klik pada tombol *Create Project* seperti pada Gambar 4.3.

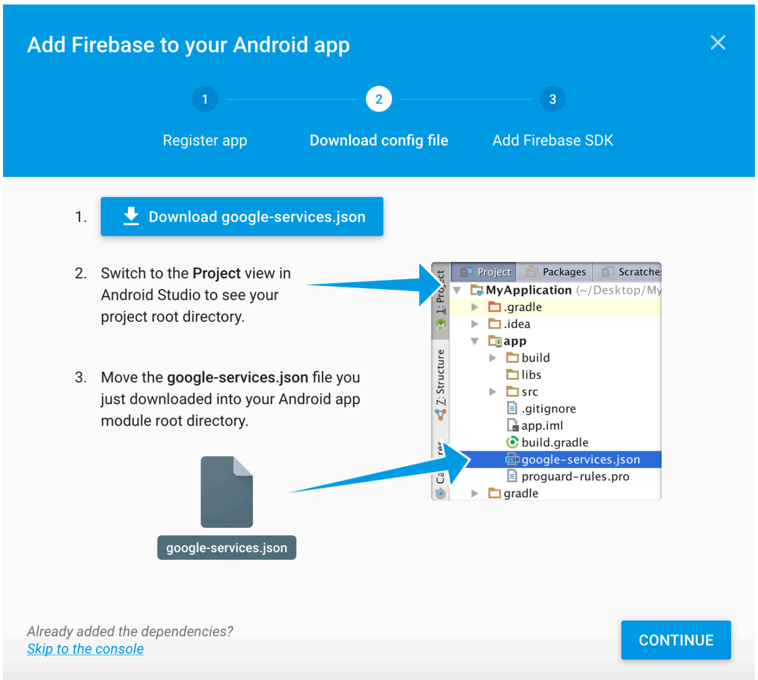


Gambar 4.3 Membuat proyek *Firebase* pada *Firebase console*

Kemudian kita juga harus menentukan nama paket Android (*package name*) yang merupakan nama domain yang dibalik diikuti dengan nama aplikasi. Lalu klik tombol *Register App*. Klik pada tombol *Register App* akan membawa ke langkah selanjutnya, yakni halaman download untuk berkas *google-services.json.* Berkas ini berisi properti tentang proyek *Firebase* yang baru saja dibuat beserta dependensi yang dibutuhkan. Berkas ini kemudian disertakan pada proyek PhoneGap untuk pengembangan aplikasi Android yang ingin diintegrasikan dengan layanan FCM. Registrasi aplikasi dan halaman unduh berkas JSON masing-masing terdapat pada Gambar 4.4 dan Gambar 4.5.



Gambar 4.4 Registrasi aplikasi Android pada *Firebase console*



Gambar 4.5 Halaman download berkas JSON saat registrasi aplikasi

4.1.2 *Database* Aplikasi Web

Desain EER yang telah didesain pada *MySQL Workbench* dapat dilakukan *reverse engineering* untuk mendapatkan *statement SQL* yang relevan untuk pembuatan tabel di *database*. Hasil dari *reverse engineering* tersebut dapat diaplikasikan pada *database* MySQL.

Untuk keperluan pengembangan aplikasi web admin maka dibuatlah *database* "university_assistance". *SQL statement* untuk pembuatan *database* aplikasi web adalah sebagai berikut:

```
CREATE DATABASE university_assistance;
```

Semua *SQL statement*s diatas baik *create database* maupun *create table sintax* dapat dijalankan pada MySQL Client seperti PHPMyAdmin.

Pengguna juga dapat melihat profil pribadinya, melihat sebaran informasi yang telah disebarkan oleh administrator dan mengajukan komplain. Data komplain yang telah dimoderasi oleh administrator dapat diberikan *reply* oleh administrator dan dari sisi pengguna sendiri dapat memberikan *feedback* sebagai respon dari *reply*. Selain itu, sistem pada aplikasi Android pengguna juga memungkinkan untuk merima notifikasi dari layanan FCM.

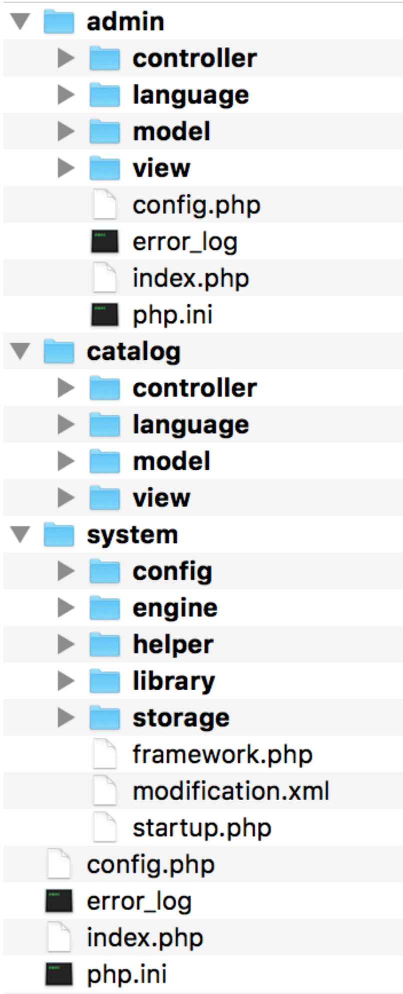
4.1.3 Arsitektur Aplikasi

Aplikasi web admin dibuat dengan kombinasi HTML, CSS, Java*Script*, PHP dan MySQL. HTML dan CSS digunakan untuk mendefinisikan berbagai elemen di halaman web dan mengatur bagaimana sebuah elemen html ditampilkan dalam sebuah halaman. Sementara itu PHP akan menangani *request* halaman web tertentu yang membutuhkan penanganan form secara dinamis. Kemudian data form akan diatur penyimpanannya menggunakan MySQL.

Lebih dari itu implementasi dari teknologi web diatas dilakukan dengan pola MVC (*Model View Controller*) yang membagi aplikasi menjadi tiga bagian modular yang memiliki fungsi masing-masing. Secara spesifik ketiga bagian tersebut dijelaskan memiliki fungsi sebagai berikut:

- Bagian *model* menangani data dan aktifitas aplikasi yang secara fisik berisi kumpulan dari *mysql statements*. Dimana statemen tersebut melakukan aktifitas CRUD (*Create, Read, Update, Delete*) sesuai *request* dari pengguna yang ditangani oleh bagian *view*.
- Bagian *view* menangani bagaimana informasi ditampilkan kepada pengguna sesuai permintaan kondisi yang diinginkan pengguna.
- Bagian *controller* akan mengatur pengiriman perintah kepada bagian model untuk mengubah kondisi dari model.

Untuk memenuhi metode implementasi dengan artisektur diatas maka digunakanlah *framework* CMS OpenCart yang telah memiliki arsitektur diatas. OpenCart sendiri merupakan platform *ecommerce* yang di dalamnya terdapat berbagai fitur untuk pendukung proses jual beli mulai dari *insert product*, *display*, sistem *cart*, *checkout*, sistem *mailer*, dll. Sehingga untuk jadi sistem yang relevan digunakan untuk proyek ini harus dieleminasi fitur yang tidak relevan dan hanya mengambil *core system*-nya saja. Kemudian dari sistem utama ini baru kemudian dibangun sistem aplikasi sesuai rancangan sebelumnya. Struktur direktori aplikasi web administrator akhir dimana sistem aplikasi web administrator diimplementasikan diperlihatkan pada Gambar 4.6.



Gambar 4.6 Struktur direktori aplikasi web admin dengan arsitektur MVC

Seperti terlihat pada gambar diatas di direktori utama terdapat 3 direktori, yakni *admin*, *catalog* dan *system*. Folder admin berisi berkas *model, view, controller* dan *language* untuk *interface* halaman administrasi. Sementara pada direktori *catalog* terdapat pula berkas *model, view, controller* dan *language* yang secara khusus menangani *request* dari aplikasi Android untuk mengembalikan informasi yang diminta. Folder language pada masing-masing direktori digunakan untuk menyimpan berkas-berkas yang berisi variabel-variabel bahasa untuk keperluan translasi.

Di halaman utama terdapat berkas index.*php* yang berisi sintak PHP untuk memanggil berkas *config.php* dan *startup.php*. Isi berkas *index.php* diperlihatkan pada Gambar 4.7.

```
<?php
// Configuration
if (is_file('config.php')) {
      require_once('config.php');
}

require_once(DIR_SYSTEM . 'startup.php');

start('catalog');
```

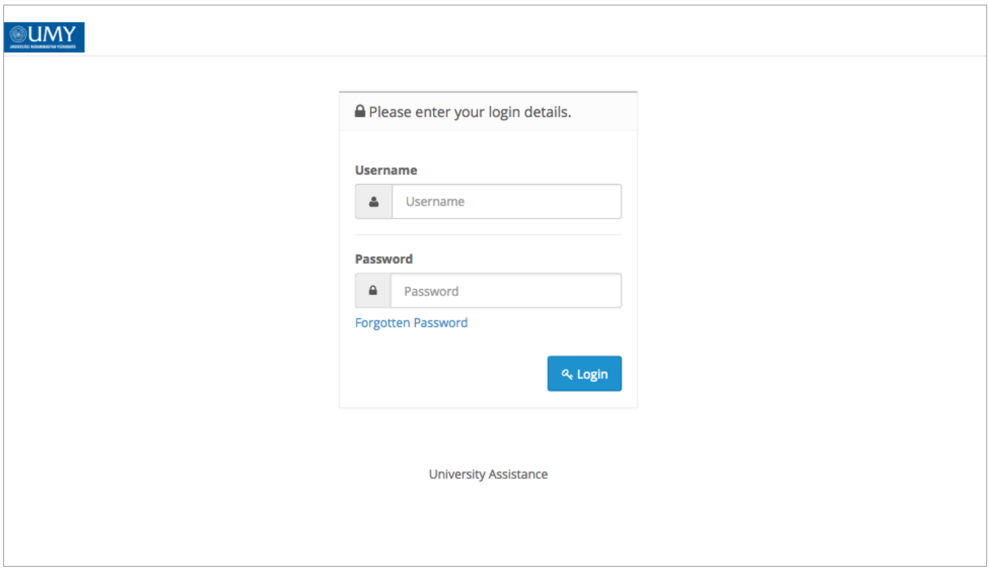
Gambar 4.7 Berkas index.*php* pada direktori utama aplikasi

Pada berkas config.*php* yang berisi kumpulan konstant]a baik berupa nama domain, path ke direktori tertentu dalam sistem dan informasi terkait koneksi ke *database*. Berkas *config.php* di direktori utama dari sistem. Terdapat juga berkas *error_log* untuk *logging* kesalahan selama *scripting* dan *php.ini* untuk mengatur seting *server* saat masa produksi di *server hosting*.

4.1.4 Implementasi Sistem dan Antarmuka Pengguna

4.1.4.1 Halaman Login

Halaman *login* difungsikan untuk alasan securitas sekaligus bagian dari *access control* dimana hanya administrator yang telah terdaftar dalam sistem aplikasi web yang dapat melakukan operasional di dalamnya. Autentikasi pada halaman *login* meliputi *username* dan *password*. Untuk *top administrator user*name dan *password* dihasilkan secara manual. Sementara untuk akun administrasi yang lain dapat ditambahkan via halaman manajemen administrator. Tampilan halaman *login* ini ditunjukkan oleh Gambar 4.8.



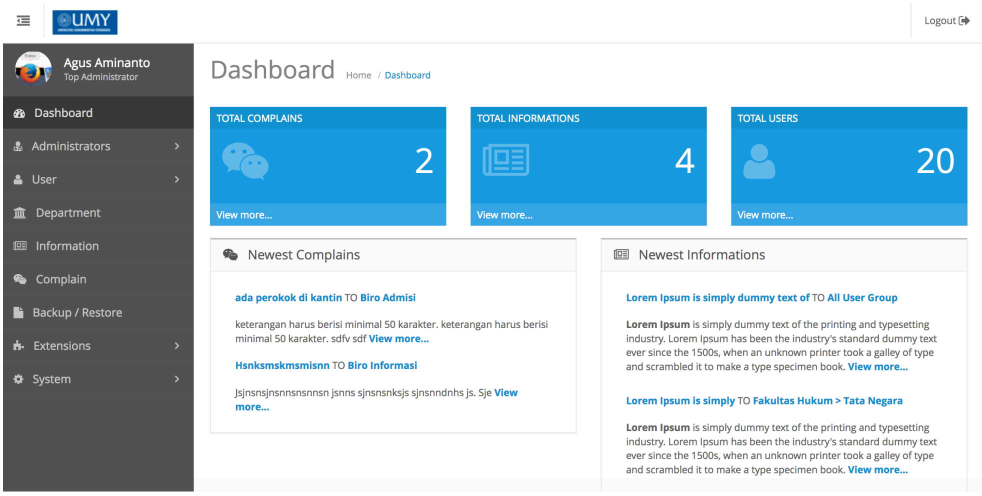
Gambar 4.8 Halaman *login* sistem aplikasi web administrasi

Validasi dilakukan saat klik pada tombo *login* dan akan diproses oleh skrip PHP. Proses autentikasi meliputi pengecekan *username* dan *password* apakah sesuai dengan data yang ada pada tabel administrator di *database*. Fungsi validasi dilakukan oleh prosedur *validate()* yang dipanggil secara internal oleh skrip validasi.

Sementara prosedur *validate()* akan menggunakan produsedur lain, yakni *login()* yang ada pada class *user* untuk *username* dan *password*. Secara fisik prosedur *login()* melakukan *query* pada *database* apakah *username* dan *password* tersedia di satu baris tabel administrator.

4.1.4.2 Halaman Administrasi

Halaman administrasi dapat diakses setelah administrator melewati proses autentikasi pada halaman *login*. Halaman administrasi ini berisi serangkaian halaman untuk menjalakan fungsi administrasi, yakni add, edit dan delete. Lebih dari itu ada beberapa fitur tambahan di dalam di halaman administrasi yang memungkinkan administrator untuk menyebarkan informasi, *backup* dan *restore* terhadap tabel yang ada di *database*. Untuk keperluan navigasi pada tiap halaman maka dibuatlah menu pada sidebar kiri dari halaman seperti terlihat pada Gambar 4.9.

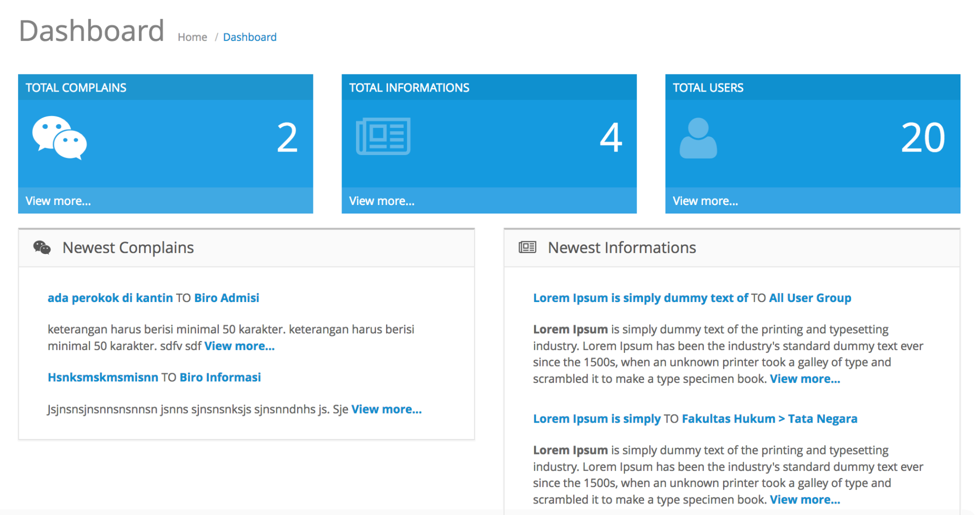


Gambar 4.9 Halaman utama pada sistem administrasi

4.1.4.3 Halaman Dashboard

Halaman dashboard berisi rangkuman dari fitur utama sistem aplikasi web. Pada halaman ini ditampilkan beberapa informasi terbaru yang telah ditambahkan oleh administrator, komplain yang disubmit oleh pengguna melalui aplikasi Android mereka dan jumlah *user* yang saat ini terdaftar.

Terdapat link untuk tiap sub bagian informasi untuk melihat konten secara lengkap dengan klik *view more* pada link yang disediakan. Selain menampilkan komplain dan informasi yang akan disebarkan, administrator juga dapat melihat total complain yang telah di-*submit* oleh pengguna. Begitu juga dengan total informasi yang ditambahkan dan total *user* yang telah terdaftar. Secara detail halaman ini ditunjukkan oleh Gambar 4.10.

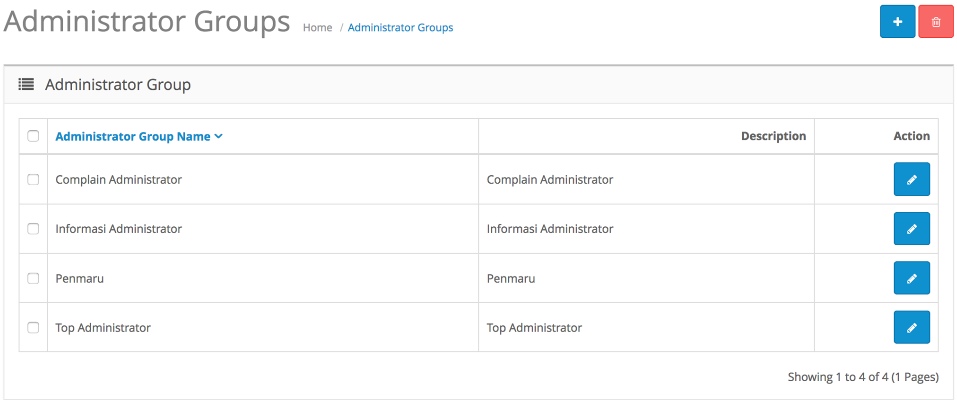


Gambar 4.10 Halaman dashboard pada aplikasi web administrasi

Tiap bagian dari halaman ini merupakan keluaran prosedur *dashboad()* dari *class* yang berbeda. Tiap prosedur dashboard pada masing-masing berkas diformat untuk menampilkan informasi yang ingin ditampilkan pada halaman *dashboard*. Contoh untuk bagian yang menampilkan informasi terbaru. Prosedur yang sama terdapat pada *berkas* lain dengan format informasi yang berbeda. Sesuai informasi yang ingin ditampilkan pada halaman dashboard.

4.1.4.4 Halaman Administrator

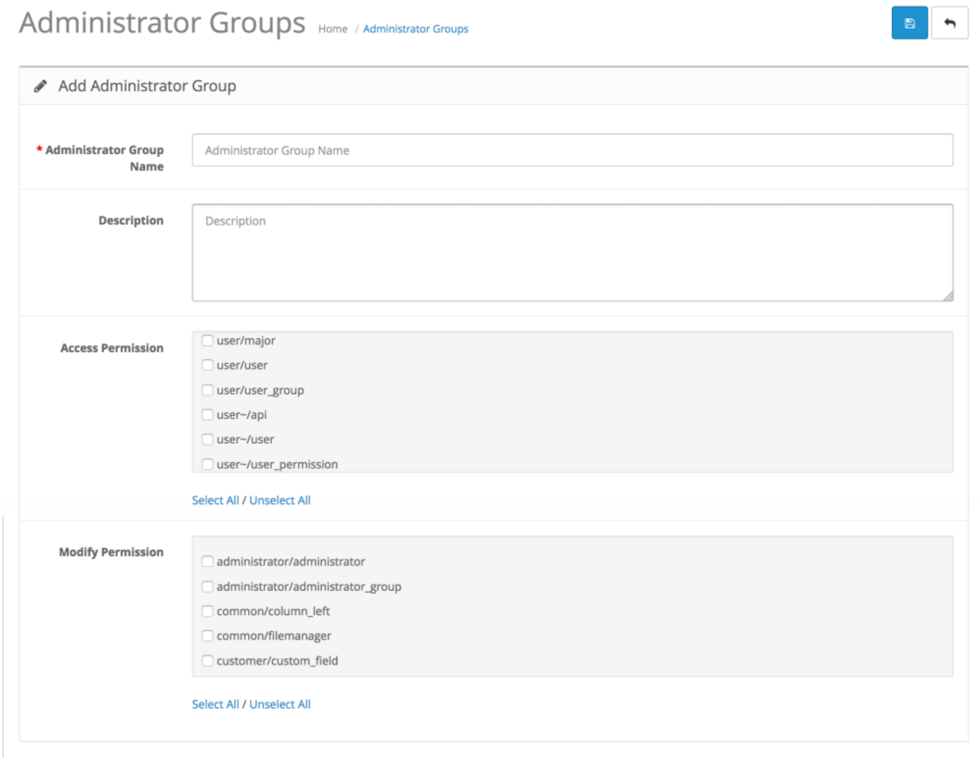
Halaman ini terdiri dari dua halaman, yakni halaman untuk manageen grup administrator dan menagemen *user administrator* sendiri. Seorang administrator akan berelasi dengan sebuah grup administrator. Sehingga untuk menambahkan seorang administrator maka terlebih dahulu harus menambahkan grup administrator. Halaman manajemen grup administrator merupakan presentasi data dari seleksi tabel grup administrator. Halaman ini ditunjukkan oleh Gambar 4.11.



Gambar 4.11 Halaman grup administrator

Dapat dilihat terdapat *keyword* LIMIT untuk membatasi jumlah seleksi data. Hal ini digunakan untuk *paging*. Nilai limit telah diset disebelumnya yakni, 20 baris. Dengan nilai *start* / baris awal seleksi yang dihasilkan secara dinamis dengan bantuan interaksi dari pengguna saat navigasi ke halaman selanjutnya dari baris data.

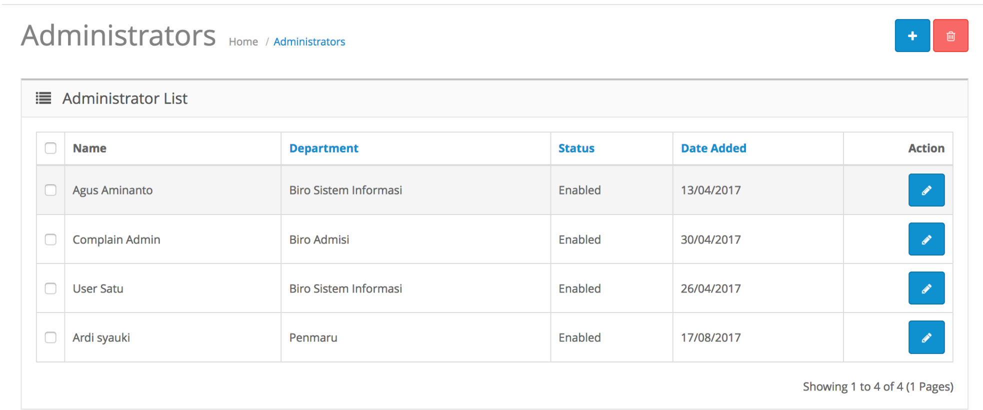
Penambahan grup administrator baru dilakukan dengan klik pada tombol biru kanan atas. Sementara untuk menghapus baris grup administarator maka harus pilih select box yang relevan lalu klik pada tombol merah. Klik pada tombol *Add* akan membawa administrator pada halaman edit administrator seperti yang telihat pada Gambar 4.12.



Gambar 4.12 Halaman edit grup administrator

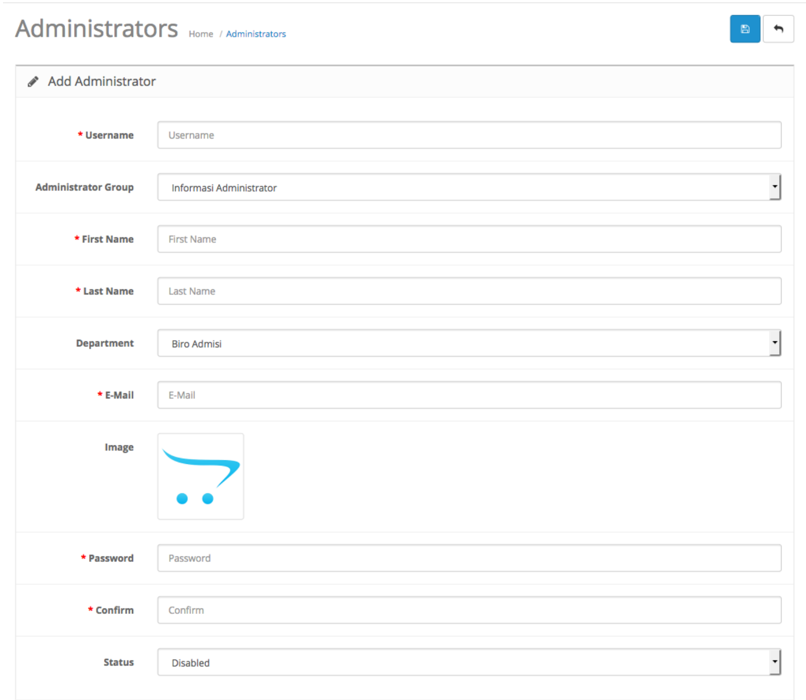
Pada halaman ini sebuah *group* administrator dapat dibatas hak akses dan modifikasinya terhadap suatu halaman. Ini berfungsi sebagai *access control* pada akun administrator yang ada pada sistem. Sehingga tiap administrator hanya dapat melakukan manajemen pada halaman tertentu sesuai dengan seting administrator *group* yang relevan.

Kemudian sebuah akun administrasi sendiri dimanagemen pada halaman adminisrators. Dimana tiap administrator berelasi *one to one* dengan departemen dan grup administrator. Sehingga departemen dan grup administrator terkait harus dibuat terlebih dahulu sebelum menambahkan akun administrator. Halaman manajemen administrator ditunjukkan oleh Gambar 4.13.



Gambar 4.13 Halaman manajemen administrator

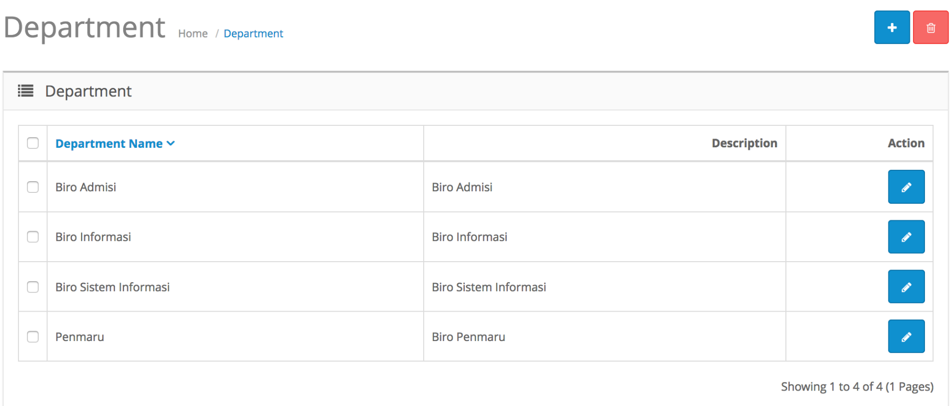
Klik pada tombol Add akan membawa pada halaman edit administrator dimana sebuah akun administrator dibuat. Isian pada kolom *username* dan *password* akan digunakan dalam proses autentikasi saat *login* ke sistem administrasi. Sementara *firstname* dan *lastname* digunakan untuk identifikasi nama administrator pada administrasi yang dilakukannya, seperti penambahan informasi, penyebaran informasi dan *reply* terhadap suatu komplain. Status *enable / diable* akan menentukan status aktivasi dari akun administator. Lebih detail halaman ini ditunjukkan oleh Gambar 4.14.



Gambar 4.14 Halaman edit administrator

4.1.4.5 Halaman Departemen

Halaman ini digunakan untuk menagemen biro yang ada di kampus. Managemen biro lebih sederhana karena baris data departemen secara tidak berelasi dengan tabel lain. Halaman manajemen dan edit departemen masing-masing ditunjukkan oleh Gambar 4.15 dan 4.16.



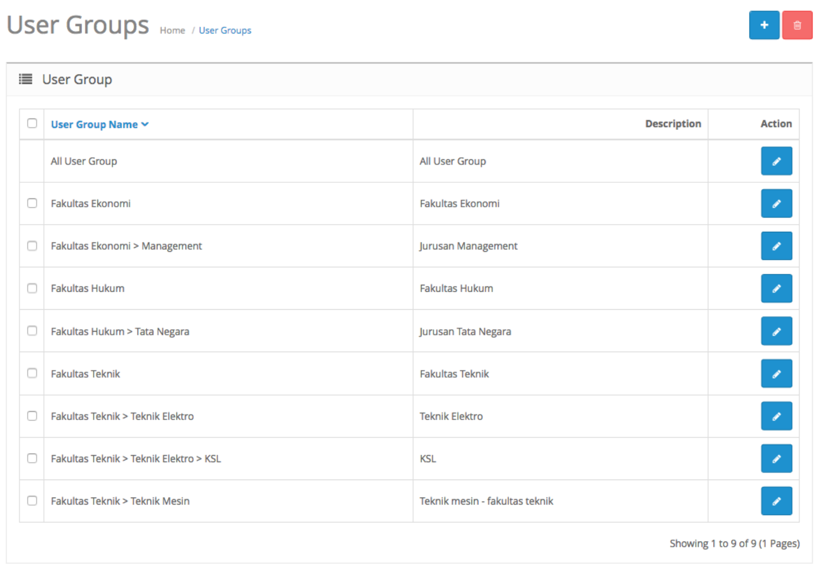
Gambar 4.15 Halaman manajemen departemen

Gambar 4.16 Halaman edit departemen

4.1.4.6 Halaman Pengguna

Sesuai namanya halaman ini digunakan untuk menagemen *user* yakni pengguna aplikasi Android. *User* pada sistem dapat diedit/ditambahkan secara manual pada halaman ini. Penambahan pengguna baru juga dapat dilakukan via aplikasi Android pengguna.

Managemen pengguna ini terdiri dari manajemen grup pengguna dan manajemen pengguna sendiri. Seorang pengguna akan relelasi one to many terhadap beberapa *group* pengguna. Karena tiap pengguna merupakan bagian dari suatu jurusan dan fakultas tertentu. Sehingga seorang pengguna dapat dikelompokkan ke dalam *user group* jurusan dan fakultas. Lebih dari itu, pengelompokkan pengguna ini dapat bebas mengikuti kebutuhan. Bahkan pengindukkan (*parenting*) dari suatu grup pengguna ke grup pengguna tidak dibatasi jumlahnya. Managemen *user group* ditunjukan oleh Gambar 4.17.

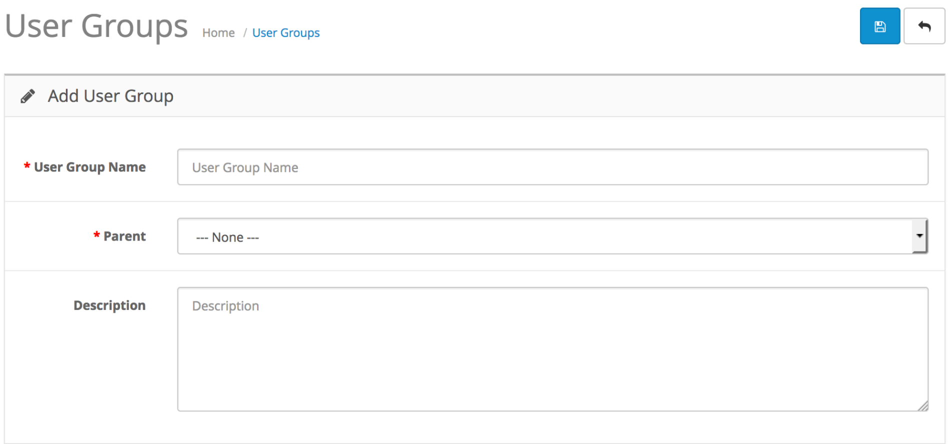


Gambar 4.17 Halaman manajemen *user group*

Seleksi data pada halaman manajemen *user group* terbilang *straightforward* seperti halnya pada halaman sebelumnya. Sebagian kompleksitas terdapat pada saat penambahan grup pengguna, yakni untuk mendapatkan *nested user group name* dari suatu *user group* atau yang dapat diistilahkan sebagai *path*. Hal ini untuk menunjukkan posisi dari suatu *user group* terhadap *parent user group*nya.

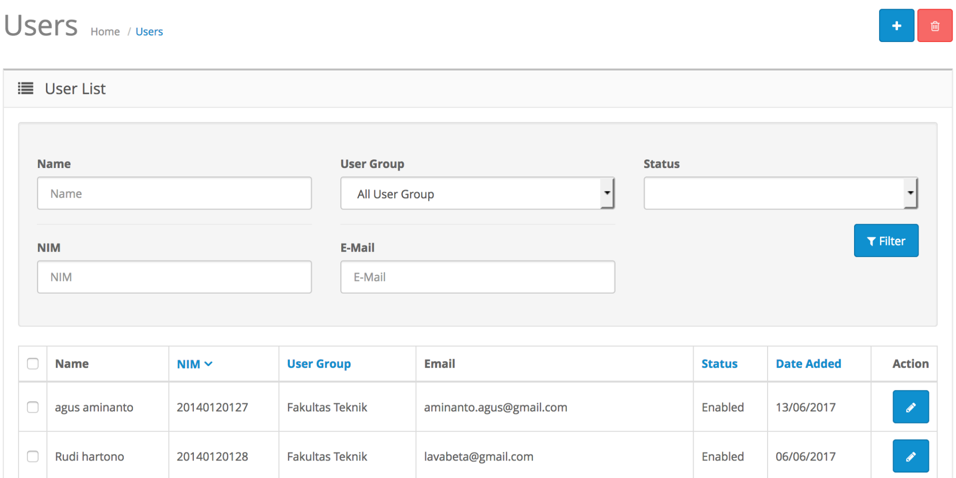
Pada skrip diatas, setelah eksekusi pada statement *INSERT* maka didapatkan id *user group* yang baru saja dimasukkan datanya. Kemudian statement *UPDATE* akan memberikan *path* dari *user group* tersebut dengan memanfaatkan prosedur *getPath()*. Prosedur secara khusu ini digunakan untuk mendapatkan *path* dari suatu *user group* berdasarkan *id*-nya.

Penambahan suatu *user group* dapat dipilih untuk menginduk pada *user group* yang lain yang diistilahkan dengan *parent*. Skema *parent* dan *child* dapat digunakan untuk merepresentasikan kelompok fakultas dan jurusan. Lebih dari itu dapat dibuat pula *user group* untuk kelompok individual tertentu yang ada dalam universitas. Form untuk penambahan sebuah *user group* ditunjukkan oleh Gambar 4.18.



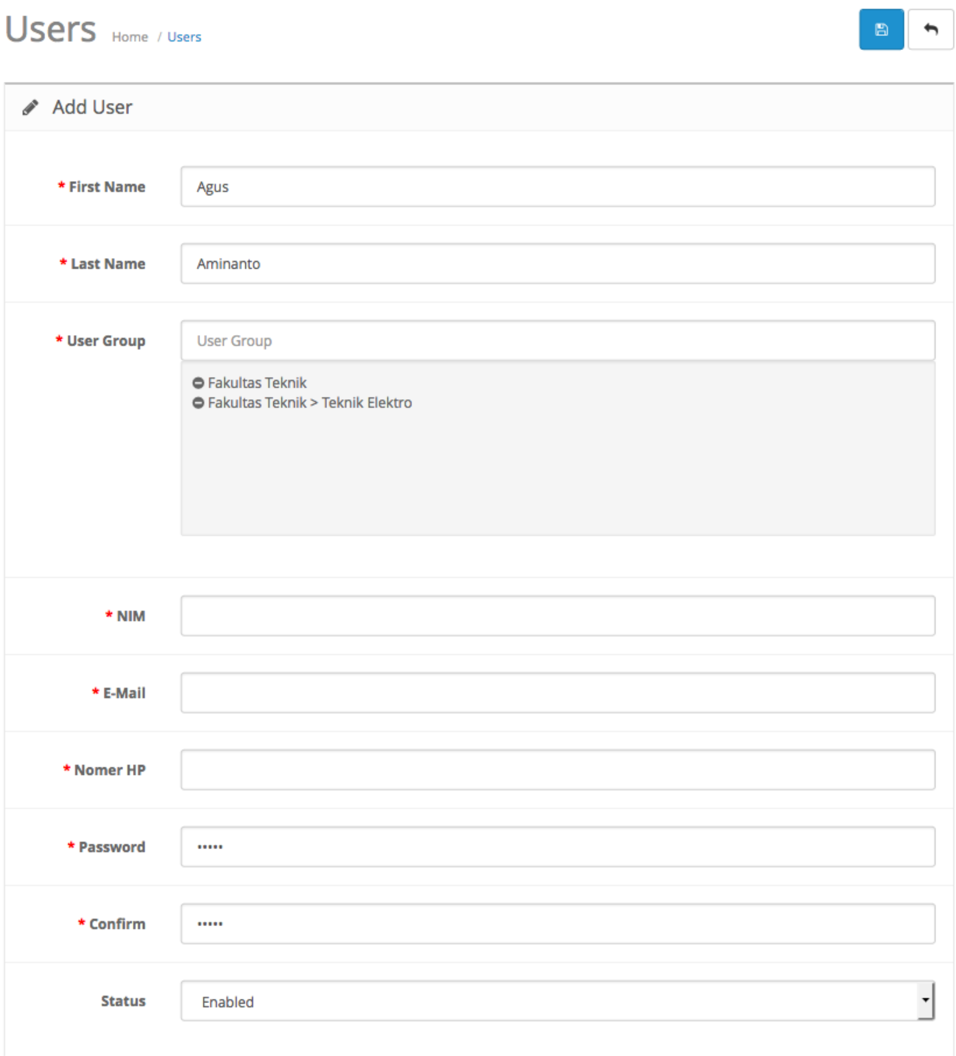
Gambar 4.18 Halaman tambah/edit *user group*

Setelah penambahan *user group*, maka administrator dapat menambahkan *user* yang menginduk pada suatu *user group* tertentu. Halaman manajemen pengguna berisi daftar pengguna yang ada di *database*. Pada halaman ini disertakan form untuk melakukan *filter* terhadap pengguna yang meliputi nama, NIM, grup pengguna, *email* dan status aktif seorang pengguna. Lebih detail halaman ini dapat dilihat pada Gambar 4.19.



Gambar 4.19 Halaman manajemen pengguna

Fungsi *multiple parent* tidak hanya ada pada grup pengguna, tetapi juga pada pengguna. Seorang pengguna dapat menginduk pada lebih dari satu grup pengguna baik induk dari grup pengguna (*parent*), maupun anak dari grup pengguna (*child*). Dari skema ini maka dapat dimanfaatkan untuk penyebaran informasi tersegmentasi, yakni dengan mengumpulkan semua data pengguna untuk suatu *group* pengguna dimana pengguna menginduk. Halaman tambah/edit pengguna ditunjukkan oleh Gambar 4.20.



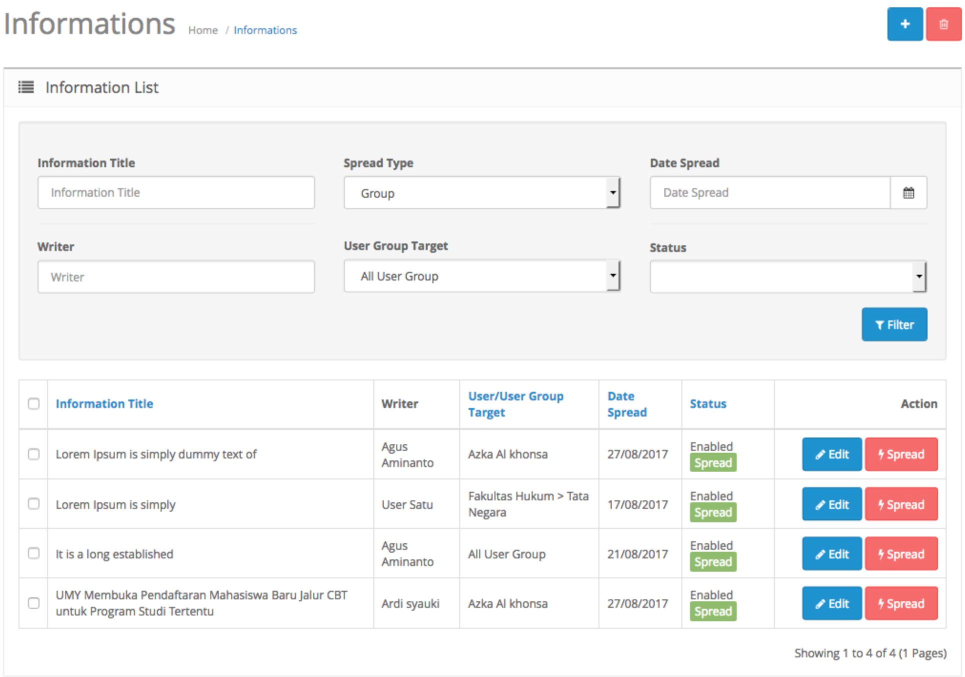
Gambar 4.20 Form tambah/edit pengguna

Input data pada *user group* menggunakan *autcomplete,* dimana *user* memasukkan frasa nama grup pengguna dan memilih dari daftar yang muncul. Untuk keperluan ini digunakan bantuan JQuery yang menggunakan fungsi AJAX untuk mendapatkan data dari prosedur di *class user_group*.

4.1.4.7 Halaman Informasi

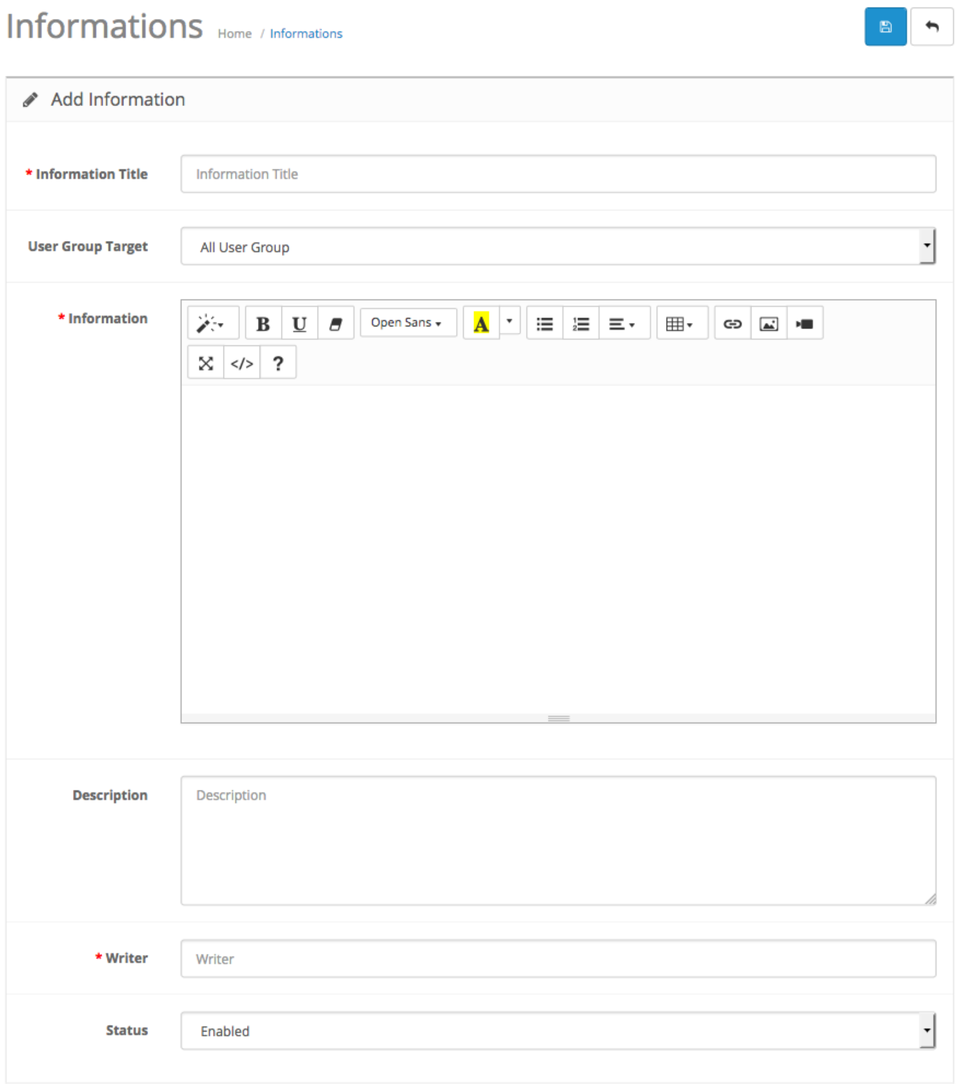
Halaman ini digunakan untuk menagemen informasi yang akan disebarkan pada pengguna aplikasi Android. Halaman ini merupakan presentasi data yang cukup kompleks dari gabungan tabel informasi, *user*, *user group*, dan departemen. Lebih detail halaman ini ditunjukkan oleh Gambar 4.21.

Terdapat seleksi data yang cukup rumit di sisi model untuk ditampilkan pada halaman manajemen informasi. *SQL statement* dasar akan menyeleksi semua data yang ada pada tabel *user*, *user group*, *department*, *information* dan *information spread*. Informasi *filter* akan diaplikasikan secara individual ke *SQL statement* dengan seleksi yang lebih spesifik sesuai informasi *filter*. Dengan *sorting default* berdasarkan kolom *date_added* secara *ascending* pada tabel informasi.



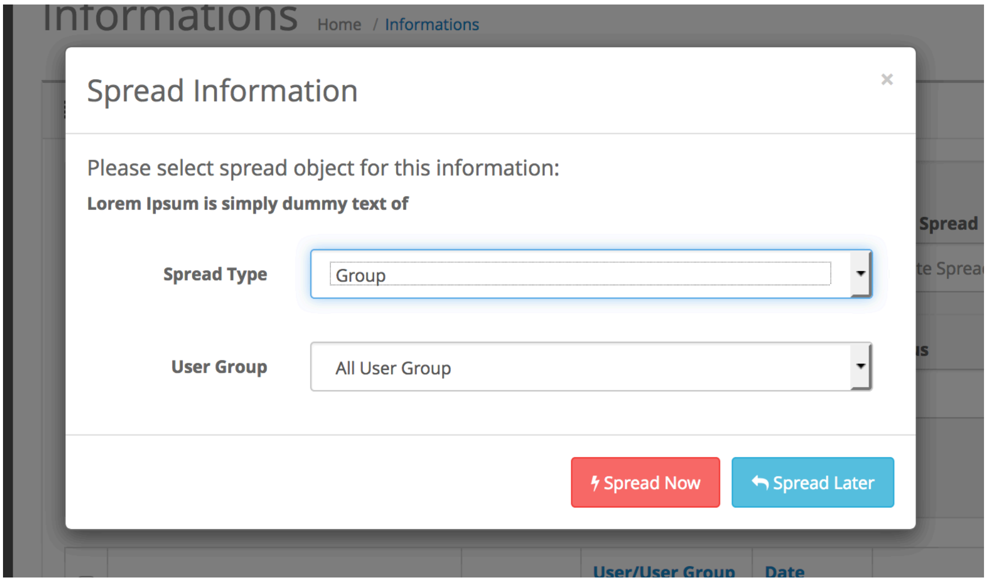
Gambar 4.21 Halaman manajemen informasi

Penambahan informasi dilakukan via form tambah informasi. Dimana administrator terlebih dahulu memasukkan informasi yang ingin disebarkan berikut dengan target *user grou-p*nya. Kemudian kolom *writer* berupa *autocomplete* yang mengisikan nama penulis sekaligus departemen yang mempublikasikan informasi tersebut. Halaman ini ditunjukkan oleh Gambar 4.22.



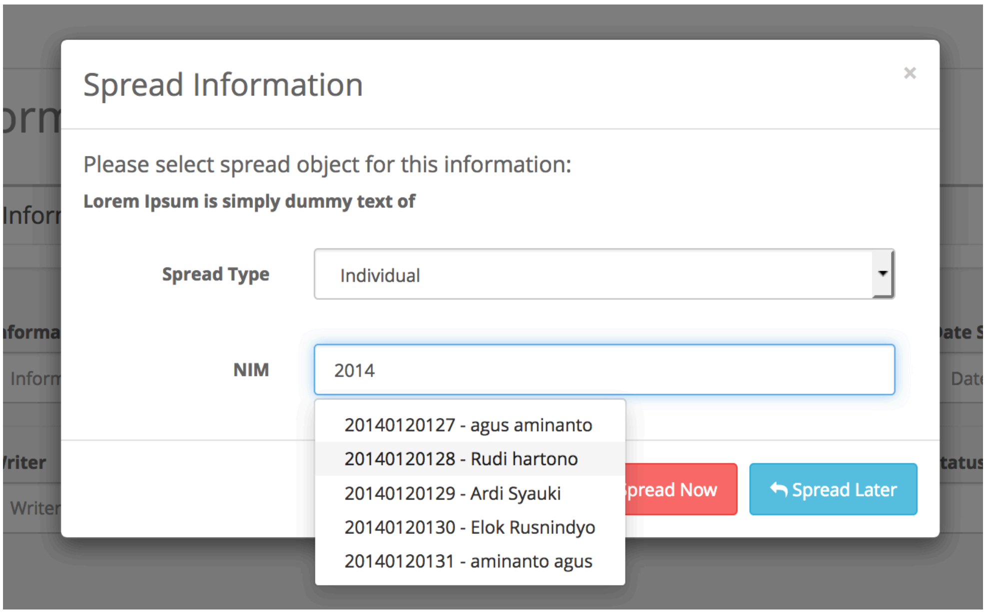
Gambar 4.22 Halaman tambah informasi

Setelah penambahan informasi maka administrator dapat menyebarkan informasi kepada para pengguna yang telah memasang aplikasi Android. Secara teknis hal ini dilakukan dengan aksi klik pada tombol *spread* yang ada pada kolom aksi. Maka akan muncul sebuah *popup modal* yang digunakan untuk menentukan *spread type*, yakni *group/individual* dan target *user group* yang ingin mendapatkan notifikasi di *smarphone* Androidnya. Hal ini ditunjukkan oleh Gambar 4.23.



Gambar 4.23 *Popup modal* untuk seting penyebaran informasi

Pemilihan *spread type* individual akan mengubah pemilihan *user group* menjadi *input* NIM untuk menunjuk individu mahasiswa yang akan menjadi target sebaran informasi. Seperti sebelumnya, masukkan NIM ini dibuat *autocomplete* dan administrator cukup memasukkan bagian dari nim dan pengguna (mahasiswa) yang relevan akan muncul pada daftar pilihan. Jendela *modal* dengan *spread type* individual ditunjukkan oleh Gambar 4.24.



Gambar 4.24 Perubahan *form input* dengan *spread type individual*

Klik pada tombol *Spread Now* maka halaman akan mengeksekusi baris kode JQUERY yang melakukan *request* via AJAX ke prosedur *spread()* di *class information*. Pada *request* ini disertakan id informasi yang ingin disebarkan pada *URL* agar dapat diambil sebagai variabel *GET*. Sementara form *spread information* yang ada pada *modal window* juga disertakan pada bagian data.

Sementara itu pada prosedur *spread()* terdapat beris kode untuk mengubah status bahwa suatu informasi telah disebarkan dengan detail departemen, *user group* / NIM target yang dilewatkan pada prosedur lain, yakni *spreadInformation()* yang ada pada berkas model *information*. Nilai kembalian dari prosedur ini disimpan dalam variabel *$spread_info* yang digunakan untuk menentukan pesan yang menentukan keluaran data dari prosedur *spread()*. Semua data keluaran disimpan pada variabel *$json* yang bertipe *array*. Penambahan *header* dan *encoding* ke format JSON akan memberikan format data JSON pada semua elemen di *array $json*.

Lebih jauh mengenai prosedur *spreadInformation()*. Prosedur memanggil beberapa prosedur lain untuk menyelesaikan proses penyebaran informasi. Tiga diantaranya adalah *getUserIds(), send()* dan *sendNotification()*. Prosedur *getUserIds()* digunakan untuk mendapatkan semua *user_id* dari tabel *user* yang menjadi target penyebaran informasi. Prosedur *send()* akan manangani pengiriman notifikasi ke pengguna dan *logging* ke *database* tentang detail informasi sebaran, seperti *user*_id mana saja yang mendapatkan notifikasi, *id_informasi* dan *user group* yang disimpan dalam tabel *information_spread*. Sampel konten tabel dari *information_spread* ditunjukkan oleh Gambar 4.25.

information_spread_id	information_id	spread_type	user_group_id	user_id
7	2	INDIVIDUAL	0	["39"]
11	4	GROUP	10	[]
12	5	GROUP	0	["7","12","13","14","15","16","18","19","...
13	6	INDIVIDUAL	0	["40"]

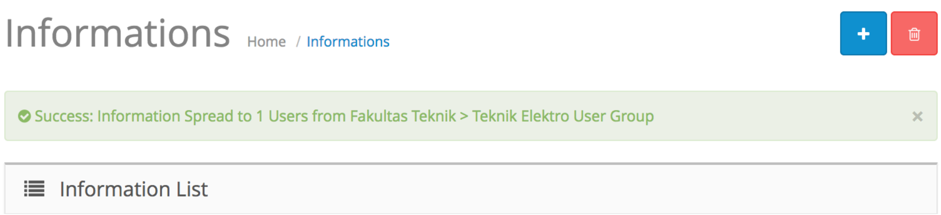
Gambar 4.25 Tabel *information_spread* yang menyimpan detail data informasi yang disebarkan

Proses lain yang ada di dalam prosedur *send()* adalah mendapatkan token registrasi dari FCM. *Token* ini didapatkan ketika pengguna melakukan registrasi via aplikasi Android dan oleh sistem akan tersimpan pada tabel *token*. Lebih dari itu detail informasi juga diambil kembali dengan *query* ke tabel informasi dengan mengkombinasikan dengan tabel departemen sehingga diperoleh data judul informasi dan nama departemen yang menyebarkan.

Prosedur ini juga memanfaatkan prosedur lain untuk menangani *request* ke *server* FCM, yakni prosedur *sendNotification()*. Prosedur ini dipanggil berulang dengan fungsi pengulangan pada PHP sejumlah *user_id* yang mendapatkan notifikasi. Skrip melakukan *request* ke URL yang telah disediakan oleh FCM API via CURL PHP dengan menyertakan informasi token registrasi FCM dan informasi mengenai notifikasi, yakni *body message*, *title, icon dan sound.* Untuk *icon* maka akan menggunakan resource icon pada aplikasi Android pengguna. Sementara *sound* akan menggunakan *sound default* notifikasi yang ada pada smartphone Android pengguna.

Kemudian data yang dilewatkan ke URL juga berisi header, yang berisi GOOGLE_API_KEY, yakni API KEY yang didapatkan saat membuat proyek *Firebase*. Baris kode sebelihnya adalah CURL yang menghandle komunikasi saat *request* ke URL yang disediakan oleh API Firebase. Termasuk diantaranya adalah melewatkan informasi notifikasi.

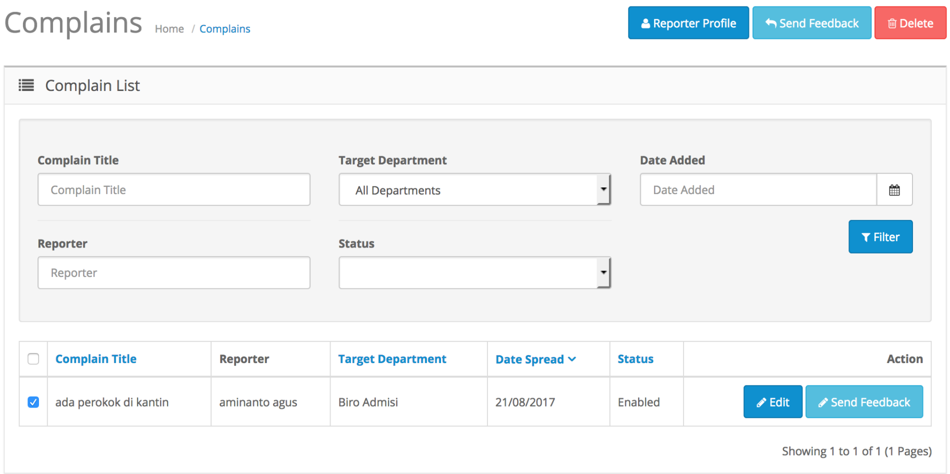
Setelah eksekusi prosedur diatas, *Firebase server* akan mengirimkan notifikasi ke semua HP Android pengguna yang telah menginstall aplikasi Android yang disediakan. Kemudian, setelah proses penyebaran informasi selesai ditambahkan pula *flag spread* pada kolom status informasi dan administrator juga menerima pesan sukses dan ditambahkan pula seperti terlihat pada Gambar 4.26.



Gambar 4.26 Pesan kepada administrator setelah penyebaran informasi.

4.1.4.8 Halaman Komplain

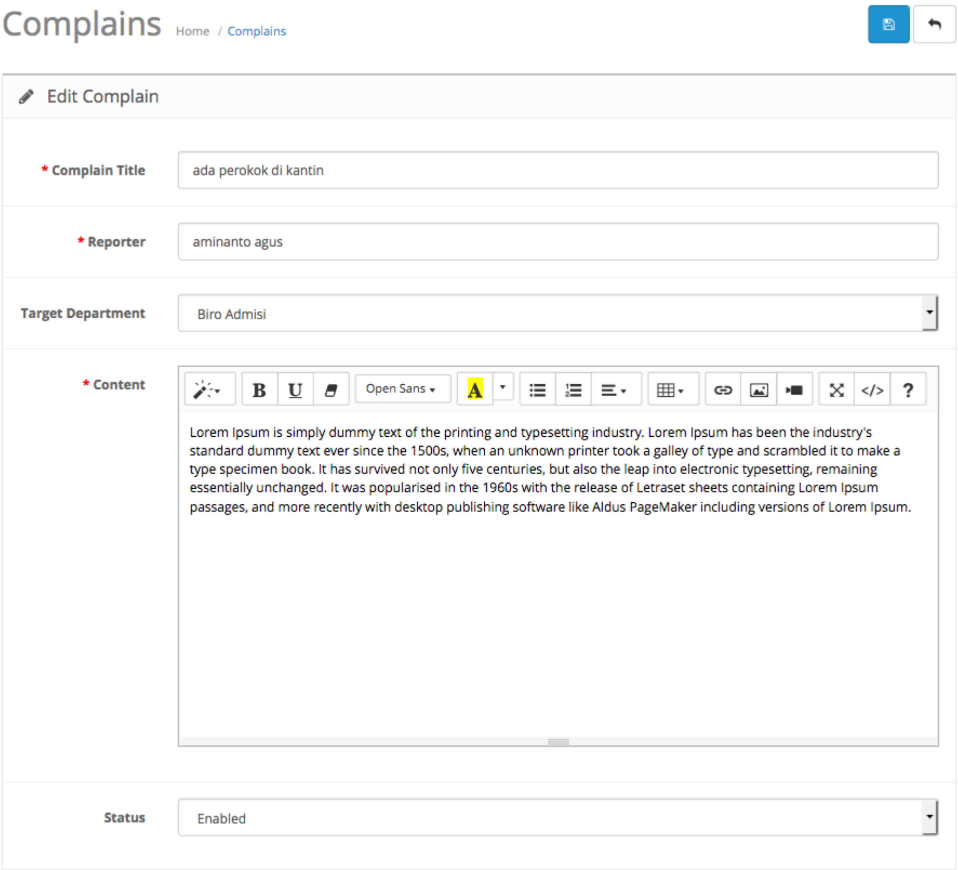
Halaman ini digunakan untuk manajemen data komplain yang disubmit dari aplikasi Android. Oleh aplikasi data akan disimpan di tabel *complain* dan ditampilkan pada halaman manajemen. Presentasi data pada halaman ini merupakan *join* dari tabel *complain* dan *department*. Halaman ini ditunjukkan oleh Gambar 4.27.



Gambar 4.27 Halaman manajemen komplain

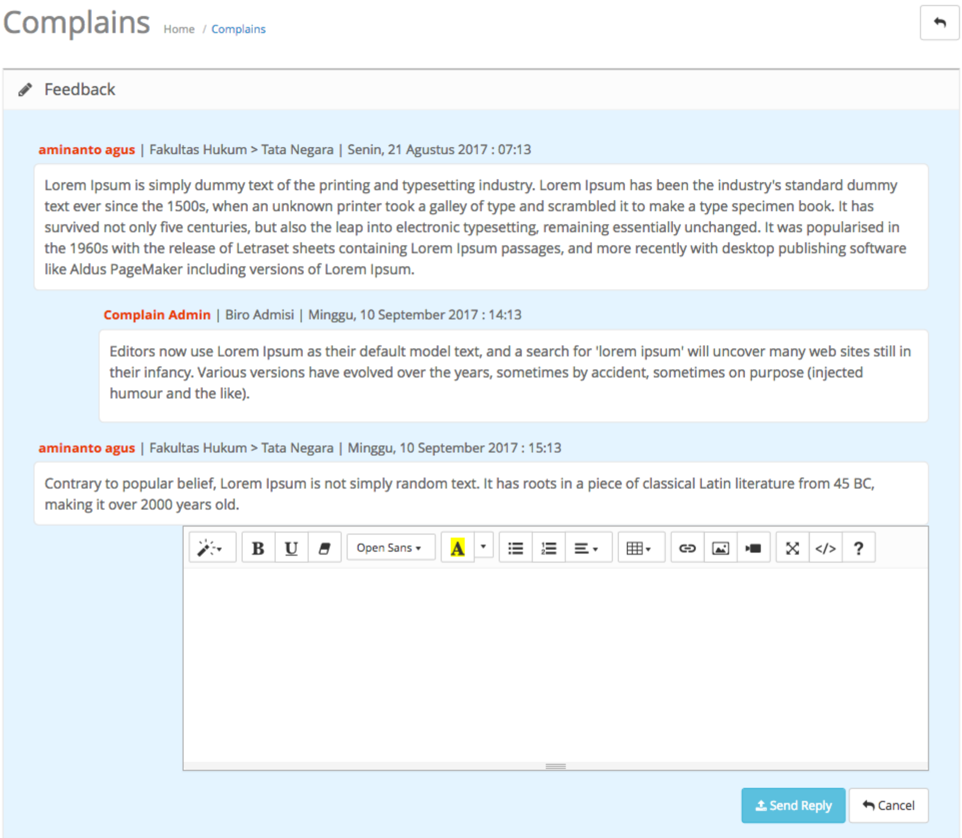
Administrator dapat menyunting informasi komplain yang masuk ke sistem aplikasi. Fitur ini diadakan dalam rangka moderasi dan pengallihan penanganan komplain oleh biro yang tepat jika pengguna tidak tepat dalam menarget departemen yang akan menangani komplainnya. Akses terhadap halaman edit komplain dapat melalui button edit yang ada pada kolom aksi.

Halaman ini ditunjukkan oleh Gambar 4.28.



Gambar 4.28 Halaman edit komplain untuk moderasi komplain yang masuk

Kemudian untuk memberikan tanggapan terhadap komplain administrator memiliki halaman tersendiri, yakni halaman *feedback*. Klik pada menu send *feedback* pada kolom aksi akan membawa pada halaman *feedback* dimana tersedia *form* memberikan *feedback*. Pada halaman ini administrator juga dapat melihat informasi detail terkait komplain, seperti nama mahasiswa, jurusan dan tanggal *submit* komplain. Detail informasi yang sama ada pada *feedback* yang telah diberikan oleh administrator. Halaman ini ada pada Gambar 4.29.

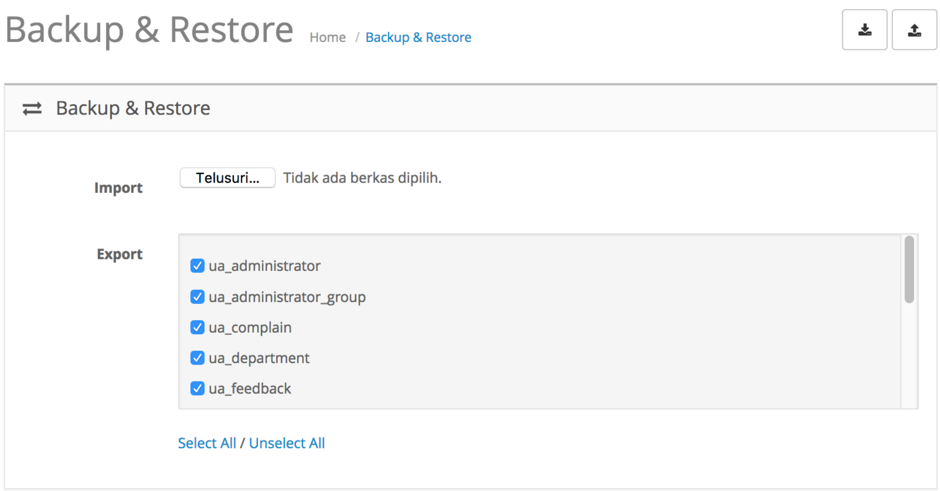


Gambar 4.29 Halaman *feedback* yang disediakan untuk memberikan *feedback* kepada pengguna

Pengiriman *feedback* oleh administrator akan memberikan notifikasi kepada pengguna via aplikasi Android bahwa telah ada *reply* dari biro terkait. Dan pengguna dapat melihat *feedback* dari administrator via halaman *feedback* di aplikasi Android mereka.

4.1.4.9 Halaman Backup/Restore

Sesuai namanya, halaman ini digunakan untuk menjalankan fungsi *maintenance*, yakni mencadangkan *database* dan melakukan impor jika diperlukan. Proses *backup* sederhana, cukup memilih tabel yang ingin dicadangkan lalu klik pada tombol *Backup*. Sistem akan menghasilkan berkas dengaa ekstensi *.sql*. Sementara untuk mengembalikan data dari berkas *backup*, administrator harus memilih berkas backup lalu klik pada tombol *import*. Halaman ini ada pada Gambar 4.30.



Gambar 4.30 Halaman *backup* dan *restore*

4.2 Implementasi Aplikasi Android Pengguna

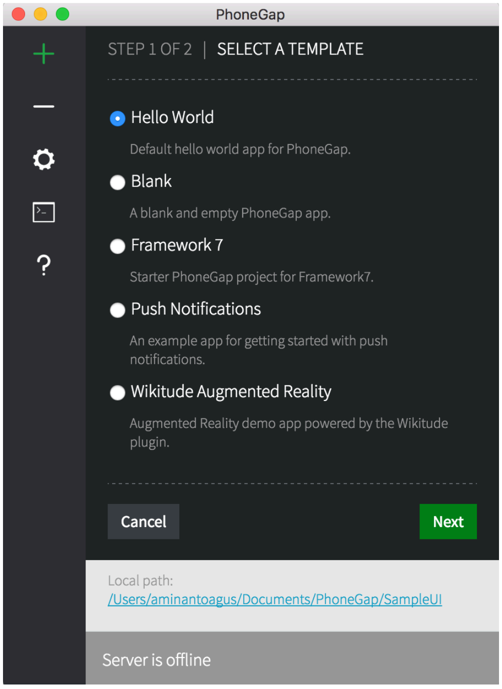
Pengembangan aplikasi Android untuk pengguna melibatkan banyak aplikasi, pustaka dan dependensi, seperti *PhoneGap*, JQuery Mobile dan Android Studio. Sementara *PhoneGap* sendiri memiliki ketergantungan dengan Node.js. Pada bagian ini akan dijelaskan proses pengembangan dan pemenuhan dependensi dalam prosesnya hingga integrasi dengan layanan FCM agar dapat menerima notifikasi. Kemudian akan dijelaskan pula integrasi aplikasi Android pengguna dengan aplikasi web administrasi.

4.2.1 Lingkungan Kerja

4.2.1.1 PhoneGap Project dan PhoneGap Firebase Library

Terdapat versi desktop dari *PhoneGap* yang dapat diunduh di situs resminya (http://*PhoneGap.com). PhoneGap Project* dibuat untuk membuat sistem aplikasi Android yang dapat menerjemahkan kode sumber HTML, CSS dan JavaScript menjadi kode sumber aplikasi Android yang dapat dimodifikasi via Android Studio.

Setelah menentukan nama aplikasi dan domainnya maka proyek *PhoneGap* dapat dibuat. Pembuatan proyek ini dapat dilihat pada Gambar 4.31.



Gambar 4.31 Pembuatan *PhoneGap project* pada aplikasi desktop *PhoneGap*

Kemudian dilakukan installasi *Firebase* plugin (https://github.com/arnesson/cordova-plugin-*Firebase*). Dengan menggunakan console/terminal installasi plugin dilakukan dengan mengeksekusi *command* berikut:

```
phone plugin add cordova-plugin-Firebase@0.1.24 --save
```

4.2.1.2 JQuery Mobile

Framework ini digunakan untuk meningkatkan interaktifitas dan responsifitas aplikasi Android pada layar *mobile*. Terdapat pustakan JavaScript dan CSS yang dapat digunakan untuk melakukan format pada kode HTML yang menyusun halaman. *Framework* ini harus disertakan dengan cara *linking* pada dokumen HTML. Berkas *framework* ini dapat diunduh di situs resminya (https://jquery*mobile*.com/).

4.2.1.3 Android Studio

Aplikasi ini merupakan aplikasi berbasis desktop yang akan digunakan untuk pengembangan aplikasi *native* Android. Aplikasi ini dibutuhkan untuk menyunting beberapa kode dan menggunakan fitur yang ada pada aplikasi untuk *troubleshooting* dan *build* aplikasi menjadi berkas aplikasi Android yang dapat dipasang dengan ekstensi *.apk*. Aplikasi ini dapat diunduh di situs resminya (https://developer.Android.com/studio/index.html).

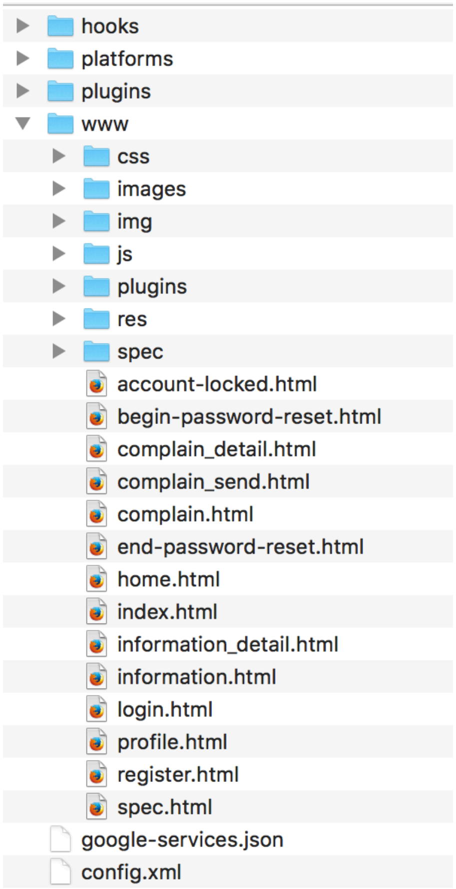
4.2.1.4 Android Virtual Device

AVD (Android Virtual Device) adalah satu fitur pada Android studio yang digunakan untuk melakukan *preview* kinerja aplikasi Android.

4.3.2 Halaman Web Aplikasi

Halaman web aplikasi ini berupa kode HTML, CSS dan JavaScript yang telah disertakan *framework* JQuery Mobile dan pustaka yang lainnya untuk membuat sebuah aplikasi *mobile* yang lengkap. Selanjutnya halaman-halaman ini dapat diterjemahkan ke kode sumber *native* pemprograman Android dengan bantuan *PhoneGap*.

Terdapat beberapa direktori pada direktori utama, yakni *hooks*, *platforms*, *plugins* dan *www*. Halaman web aplikasi terdapat pada direktori *www* dimana terdapat direktori *css*, *images*, *img*, *js*, *plugins*, *res* dan *spec* yang menunjukkan pengelompokkan berkas sesuai jenisnya. Halaman pada aplikasi disusun dari kode HTML yang terdapat pada direktori *www*. Struktur direktori halaman aplikasi web ditunjukkan oleh Gambar 4.32.



Gambar 4.32 Struktur direktori pada *PhoneGap project*

Direktori *js* digunakan untuk menyimpan berkas-berkas JavaScript dan pustaka yang diperlukan dalam aplikasi. Direktori *img* dan *images* akan menyimpan sumber gambar. Kemudian direktori *res* juga digunakan untuk menyimpan gambar namun penggunaannya lebih spesifik saat aplikasi Android telah berjalan.

Di dalam direktori *www* juga disertakan berkas *google-service.json* untuk integrasi aplikasi dengan layanan FCM. Berkas *config.xml* akan menangani properti dari aplikasi yang akan dibuat, seperti nama aplikasi, *plugin* apa saja yang digunakan dan dependensi lain yang dibutuhkan aplikasi.

Pada direktori *js* dibuat sebuah berkas yang menyimpan semua variabel yang berisi URL untuk *request* ke sistem aplikasi web yang diberi nama *settings.js*.

Berkas ini kemudian di-*linking* ke dalam dokumen HTML sehingga semua variabel dapat diakses via dokumen tersebut.

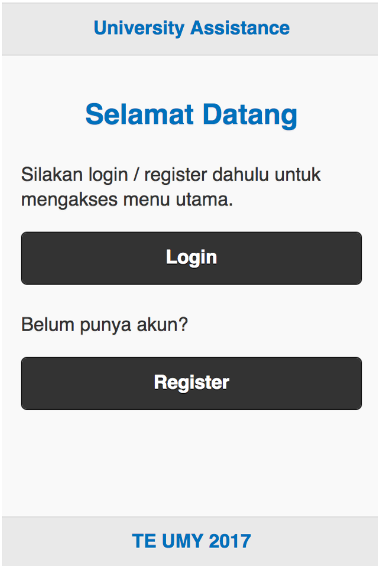
Pengembangan aplikasi Android pada tahap ini dapat dilihat kinerjanya melalui *browser* dengan akses ke alamat IP lokal komputer. PhonGap akan membuat *server* lokal yang diakses menggunakan alamat IP tersebut. Akses ke URL tersebut akan memberikan *preview* dari aplikasi Android yang masih dalam dokumen HTML, CSS dan JavaScript.

Pada tahap ini pula dokumen harus dibuat agar dapat berinteraksi dengan *database* aplikasi web secara dinamis. Untuk hal ini digunakan AJAX dengan menyiapkan suatu *class* dan *prosedur* yang menangani *query* data ke *database* yang selanjutnya dikembalikan pada halaman yang meminta informasi dalam format JSON. Sebuah *class app.php* dibuat untuk menangani *request* data dari aplikasi. Dilengkapi pula dengan banyak prosedur sesuai kebutuhan aplikasi, seperti penyimpanan data registrasi pengguna, pengambilan informasi, pengambilan profile pengguna, dll. Berikut adalah daftar prosedur yang digunakan untuk menangani *request* dari aplikasi Android pengguna:

- *addUser()*
- *addComplain()*
- *addFeedback()*
- *getInformations()*
- *login()*
- *departments()*
- *getUserProfile()*
- *checkEmail()*
- *checkNim()*
- *getExcerpt()*
- *userGroup()*
- *saveToken()*
- *getComplains()*
- *getFeedbacks()*

4.3.2.1 Halaman Beranda

Disebut halaman beranda (*home*) karena halaman ini adalah yang kali pertama tampil saat pengguna membuka aplikasi. Pada halaman ini terdapat akses untuk masuk ke aplikasi (*login*) dan registrasi. Klik pada tombol *Login* akan membawa pengguna pada halaman *login* dan tombol *Register* untuk beralih ke halaman registrasi. Gambar halaman utama ini ditunjukkan oleh Gambar 4.33.



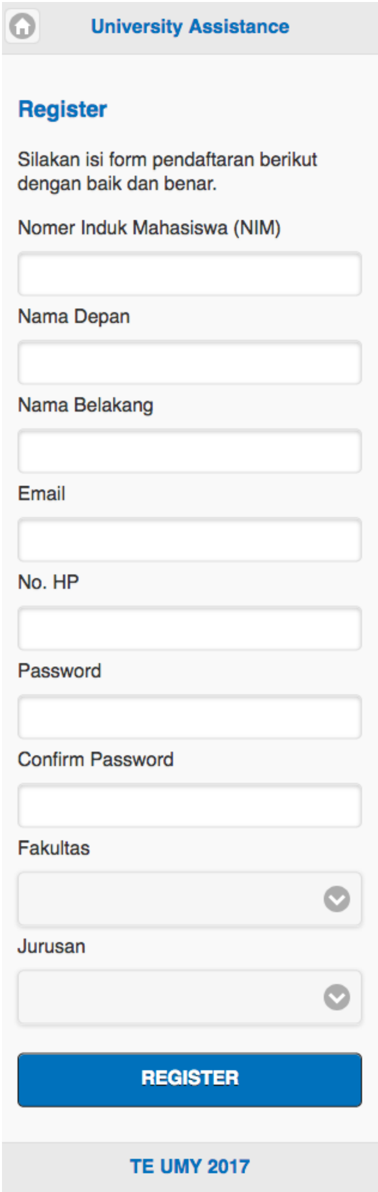
Gambar 4.33 Halaman utama aplikasi Android pengguna

Secara teknis berkas HTML yang digunakan adalah *index.html*. Berkas ini berisi beberapa berkas lain yang di-*linking* dan inisialisasi *framework PhoneGap*. Inisialisasi *framework PhoneGap* dilakukan dengan pemanggilan fungsi *initialize()* pada objek *app()*.

4.3.2.2 Halaman Registrasi

Sebelum dapat menggunakan menu utama, pengguna diharuskan untuk mendaftar terlebih dahulu. Data registrasi membutuhkan masukkan NIM , nama depan, nama belakang, *email*, *password*, fakultas dan jurusan. NIM merupakan data yang unik dan akan dilakukan validasi secara *real time* saat pengguna memasukkan

NIM. Begitu pula dengan alamat *email* harus unik untuk tiap pengguna. Validasi form juga dipicu saat pengguna melakukan klik pada tombol registrasi. Halaman ini ditunjukkan oleh Gambar 4.34.



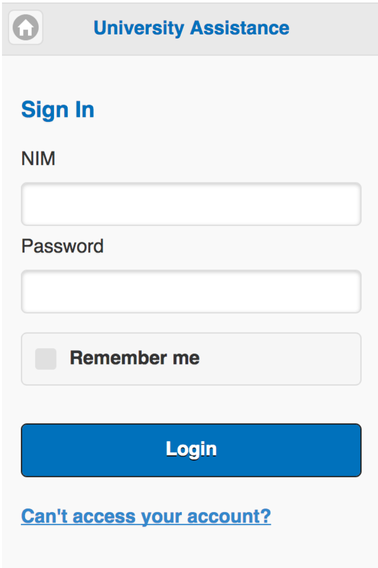
Gambar 4.34 Halaman registrasi pada aplikasi Android

Klik pada tombol registrasi maka akan terjadi proses validasi. Jika data isian form valid, maka data masukkan pada form akan dimasukkan ke database aplikasi web via melalui sebuah prosedur di aplikasi web administrasi yang URL nya ditentukan pada variabel objek Ua.Settings.registerUrl dengan request via AJAX. Prosedur pada aplikasi web yang menangani registrasi adalah addUser(). Setelah proses pemasukkan data sukses prosedur ini memiliki keluaran yang telah diformat dalam format JSON yang mengembalikan informasi pesan sukses dan user_id yang diperoleh dari tabel user.

Selanjutnya skrip juga akan menghandle proses lanjutan, yakni registrasi token *Firebase* yang ditangani oleh fungsi *getToken()* pada objek *app()*. Permintaan token ditangani oleh fungsi *getToken()* yang ada pada objek *FirebasePlugin*. Setelah mendapatkan token dari *server* FCM, token ini disimpan pada tabel di *database* aplikasi web yang ditangani oleh fungsi *saveToken()*. Fungsi *saveToken()* sendiri akan melewatkan data via AJAX ke sebuah prosedur di aplikasi web dengan URL yang telah disimpan pada *Ua.Settings.saveToken*. Dimana prosedur ini akan menangani *user_id* dan token *Firebase* pada tabel *token*.

4.3.2.3 Halaman Login

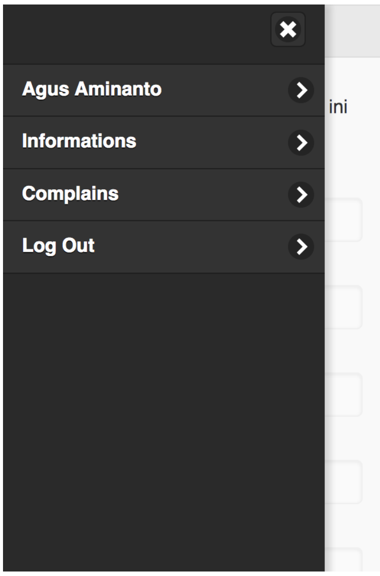
Halaman ini membutuhkan NIM dan *password* yang telah dimasukkan pada halaman registrasi. Seperti sebelumnya, proses *login* juga melibatkan AJAX *request* ke aplikasi web untuk keperluan validasi. Hasilnya akan dikembalikan dalam fungsi JSON yang dapat diproses secara lokal oleh aplikasi. Jika NIM dan *password* yang dimasukkan valid, maka pengguna akan melihat menu utama. Halaman *login* pengguna terdapat pada Gambar 4.35.



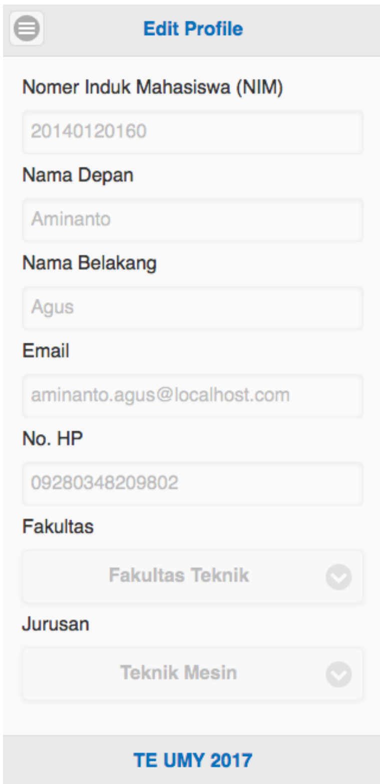
Gambar 4.35 Halaman *login* aplikasi Android pengguna

4.3.2.4 Menu dan Halaman Profil

Setelah pengguna berhasil *login* ke sistem, maka akan diarahkan ke halaman profil. Halaman ini digunakan untuk melihat profile pengguna. Informasi di halaman ini statis. Namun berguna untuk memastikan bahwa pengguna telah memasukkan informasi profilnya dengan benar. Halaman ini dapat diakses via menu *slider* yang ada pada kiri halaman. Nama *link* yang menuju halaman ini akan menyesuaikan dengan nama pengguna. Daftar menu yang ada pada aplikasi dapat dilihat pada Gambar 4.58 dan halaman profil pengguna pada Gambar 4.36.



Gambar 4.36 Daftar menu aplikasi Android pengguna

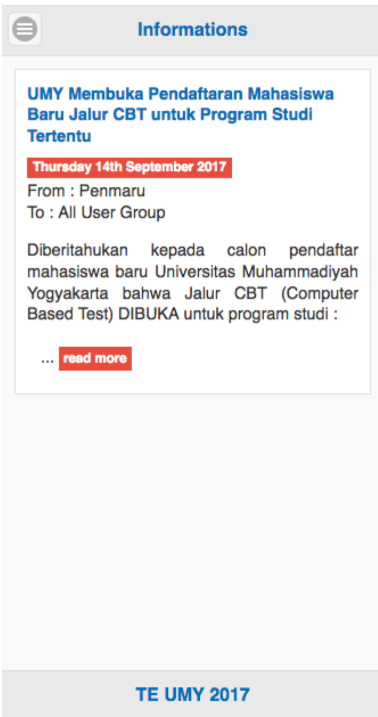


Gambar 4.37 Halaman profil pengguna

Halaman profil ini dihasilkan dari *database* aplikasi web yang telah diformat dalam format JSON oleh sebuah prosedur yang ditunjuk oleh variabel *Ua.Settings.getUserProfile.* Data JSON tersebut kemudian ditampilkan pada halaman dengan *set value* untuk masing-masing *form element static* di halaman. Permintaan data akan membutuhkan data *user_id* yang disertakan pada URL saat proses *request* via AJAX.

4.3.2.5 Halaman Informasi

Halaman ini merupakan salah satu halaman inti dari aplikasi yang menampilkan semua informasi sebaran yang ditujukan pada suatu pengguna. Baik yang disebarkan secara kelompok atau individual ke suatu pengguna tertentu oleh administrator. Secara fisik halaman terdiri dari daftar informasi yang telah diambil cuplikan kontennya. Terdapat *link* untuk membaca keseluruhan informasi terkait. Halaman ini ditunjukkan oleh Gambar 4.38.

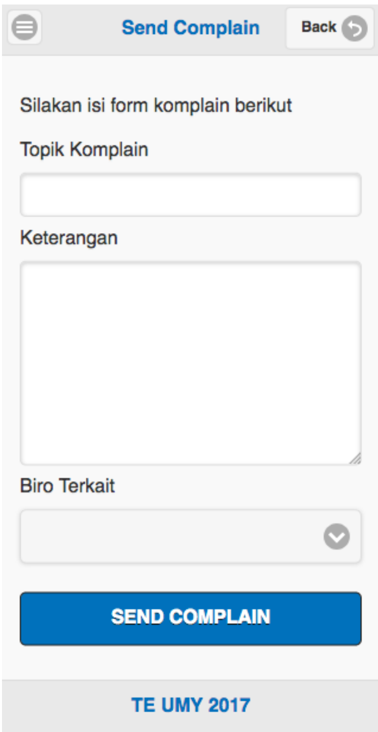


Gambar 4.38 Halaman informasi untuk menampilkan informasi sebaran

Seperti halaman konten lain, informasi pada halaman ini dihasilkan dari *query* dari aplikasi web administrasi. Informasi *user_id* juga disertakan untuk seleksi data informasi yang peruntukkan oleh pengguna yang saat ini *login*. Informasi *user_id* ini diperoleh dari *local storage*, yang disimpan saat pengguna *login*. Semua data yang diperoleh kemudian ditampilkan pada halaman dengan bantuan perulangan *$.each()* pada JQuery.

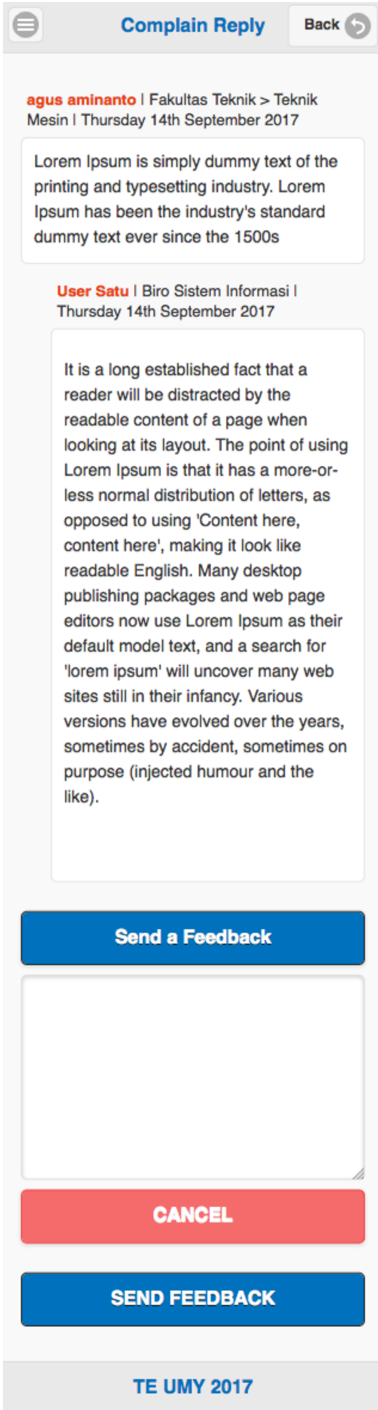
4.3.2.6 Halaman Komplain

Halaman ini digunakan untuk melihat daftar komplain yang telah di-*submit* oleh pengguna. Sebelumnya seorang pengguna harus mengirimkan komplain melalui form yang disediakan. Pengguna harus mengisikan topik aduan, keterangan dan biro terkait yang menjadi tujuan komplain/diharapkan dapat menangani komplain yang dimaksud. *Form* isian komplain ini dapat diliat pada Gambar 4.39.



Gambar 4.39 Form isian untuk mengirimkan komplain kepada biro tertentu

Data isian form diatas akan disimpan pada *database* aplikasi web untuk kemudian ditangani oleh biro terkait. Penanganan komplain oleh biro tertentu dapat berupa memberikan *feedback* kepada pengguna. Pemberian *feedback* oleh administrator akan memberikan notifiikasi kepada aplikasi Android pengguna pengguna. Kemudian dari sisi pengguna dapat melihat balasan dari administrator bahkan memberikan *feedback* lain atas jawaban dari administrator seperti yang terlihat pada Gambaar 4.40.



Gambar 4.40 Halaman *complain reply* untuk memberikan tanggapan kepada *feedback* administrator

4.3 Build Aplikasi Android

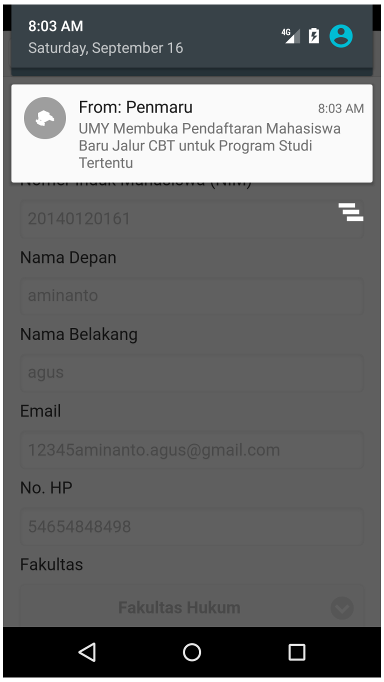
Setelah selesai dengan interaktifitas dokumen web dengan aplikasi web, maka dokumen web tersebut dapat dikonversi ke aplikasi *native* Android dengan bantuan *framework PhoneGap*. Yakni dengan menjalan *command* berikut pada *PhoneGap main* proyek *directory*.

```
PhoneGap build Android
```

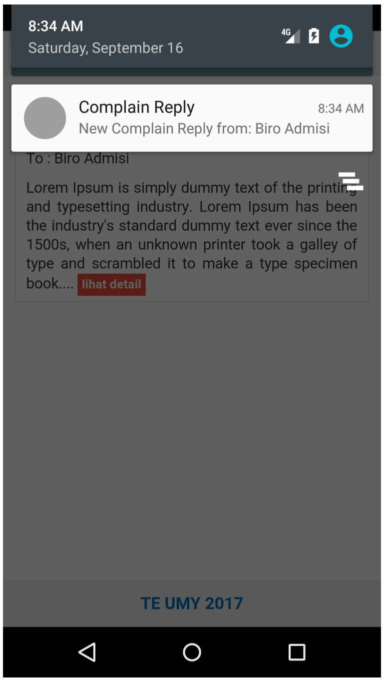
Maka akan dibuat Android proyek yang ada di direktori *Android* pada direktori *platforms*. Selanjutnya *development* dapat dilanjutkan via aplikasi Android Studio dengan menggunakan hasil konversi dari *PhoneGap*. Pada aplikasi ini tahap *development* yang ada adalah tes terhadap fungsi notifikasi dari layanan FCM. Untuk keperluan ini perlu dibuatlah *Android Virtual Device* (AVD), yang menyimulasikan perangkat fisik namun berupa perangkat lunak yang dapat diseting pada aplikasi Android Studio. Pada tahap ini pula dapat dilakukan *debugging* terhadap kesalahan yang mungkin timbul saat operasional fitur aplikasi Android.

Tes meliputi semua *use case* pada aplikasi seperti register, *login*, mengajuan komplain, memberikan *feedback* dari komplain, melihat profile, informasi sebaran dan tentunya fitur notifikasi.

Beberapa hasil tes untuk fitur notifikasi saat ada sebarang informasi pada AVD dapat dilihat pada Gambar 4.41. Kemudian notifikasi saat administrator memberikan *feedback* dari komplain yang dikumpulkan oleh pengguna terdapat pada Gambar 4.42.



Gambar 4.41 Tes notifikasi sebaran informasi pada AVD

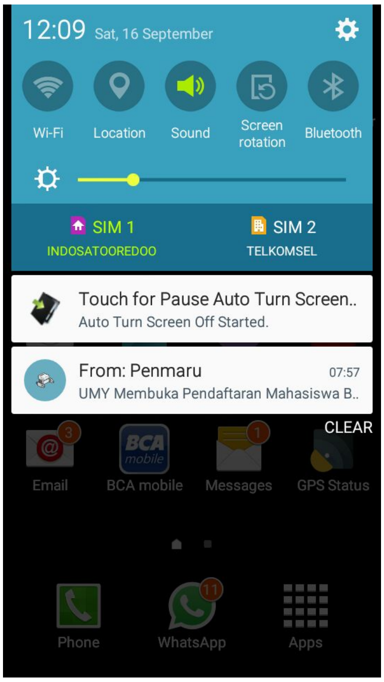


Gambar 4.42 Tes notifikasi *feedback* dari komplain pada AVD

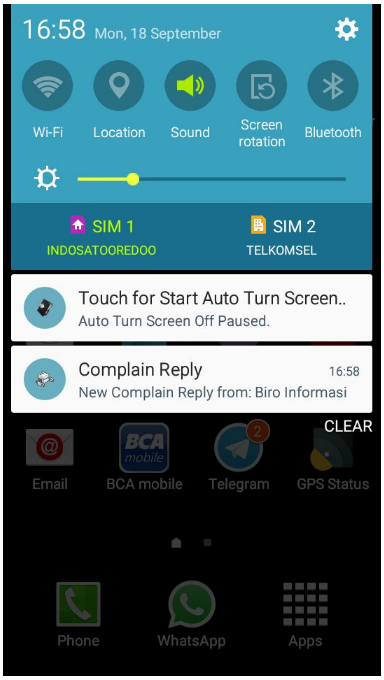
Setelah semua tes berjalan dengan baik, maka aplikasi Android yang *installable* dapat dihasilkan dari proyek Android saat ini. Build aplikasi Android ke berkas yang dapat dipasang (*installable berkas*) juga dilakukan via aplikasi Android Studio.

4.4 Tes Notifikasi Pada Smartphone

Installable berkas yang sebelumnya dibuat dapat dipasang pada *smartphone* Android dan melalukan tes fitur seperti sebelumnya pada perangkat Android. Notifikasi proses penyebaran informasi ditunjukkan oleh Gambar 4.43. Kemudian notifikasi balasan komplain dari administrator ditunjukkan oleh Gambar 4.44.



Gambar 4.43 Notifikasi sebaran informasi pada perangkat Android



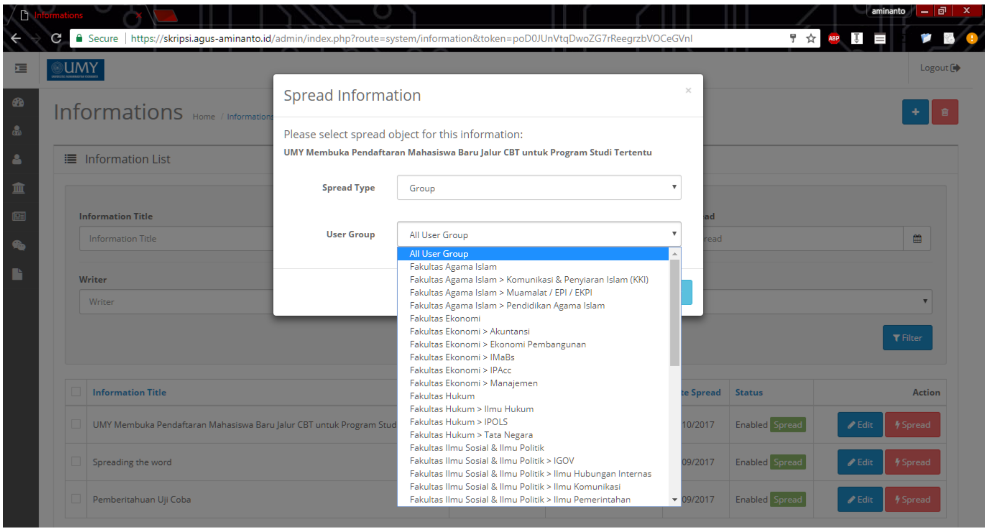
Gambar 4.44 Notifikasi balasan komplain pada perangkat Android

4.5 Tes Interpretasi Browser

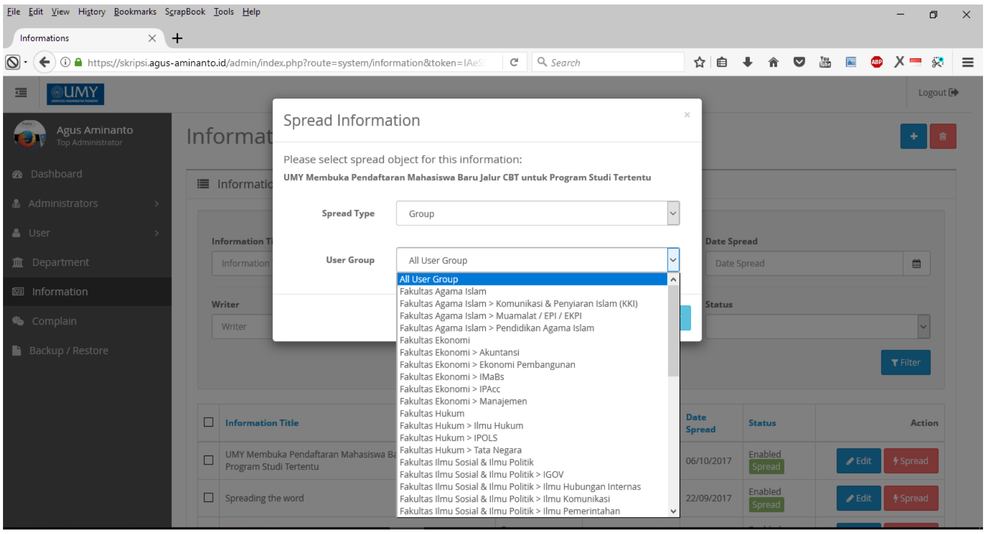
Tes interpretasi *browser* dimaksudkan untuk melakukan uji coba sistem aplikasi web administrasi dari segi konsistensi tampilan halaman web pada berbagai *browser*. Tes ini dilakukan untuk mengatasi perbedaan tampilan yang terjadi karena interpretasi peramban yang bisa saja berbeda untuk elemen HTML atau CSS yang digunakan pada halaman web. Konsitensi tampilan ini penting untuk pengalaman pengguna yang baik dan untuk mengetahui peramban terbaik yang digunakan untuk operasional sistem aplikasi web.

Secara teknis tes kompatilitas meliputi fungsionalitas dasar dari halaman web, seperti link, kotak dialog, menu, dll. Kemudian konsistensi antarmuka, yakni memastikan bahwa tampilan antar peramban sesuai dengan spesifikasi. Tampilan halaman pada sistem aplikasi web juga harus dapat menyesuaikan dengan berbagai resolusi layar pengguna baik peramban *mobile* dan *desktop*.

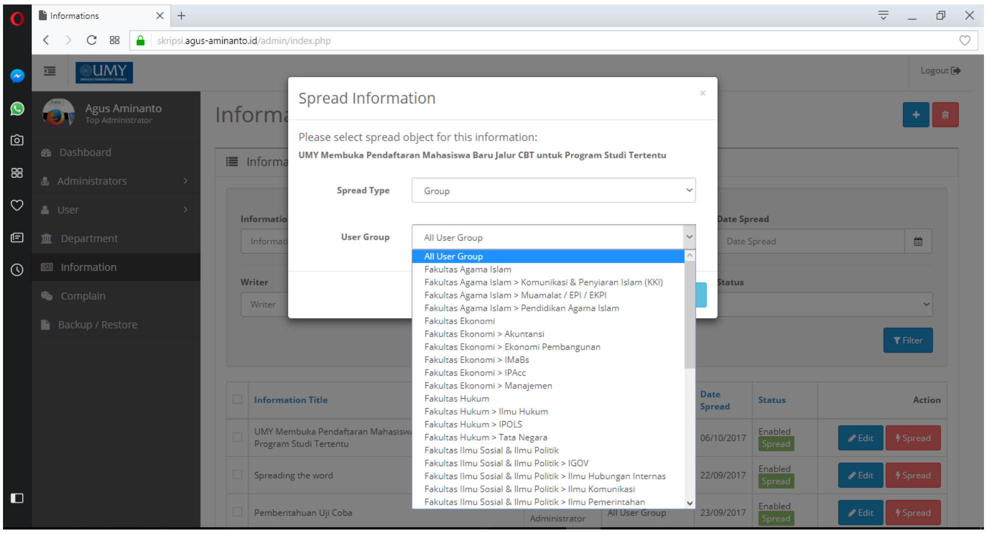
Untuk keperluan pengujian ini dilakukan tes pada 5 peramban mainstream yang paling banyak digunakan, yakni *Google Chrome*, *Mozilla Firefox*, *Safari, Opera* dan *Internet Explorer (IE)*. Resolusi peramban juga ditentukan, yakni 1366x786 yang merupakan salah satu resolusi terbanyak yang digunakan oleh pengguna internet. Tes dilakukan pada halaman information saat melakukan penyebaran informasi. Dimana terdapat kotak dialog dan beberapa *form element* di dalamnya yang dapat menjadi parameter uji. Kemudian pengujian juga dilakukan pada sistem operasi Windows untuk 4 *browser* (*Chrome, Firefox, IE* dan *Opera*) dan pada sistem operasi OSX untuk penggunaan *Safari browser*. Hasil dari pengujian ini ditunjukkan oleh Gambar 4.45 hingga 4.49 berikut.



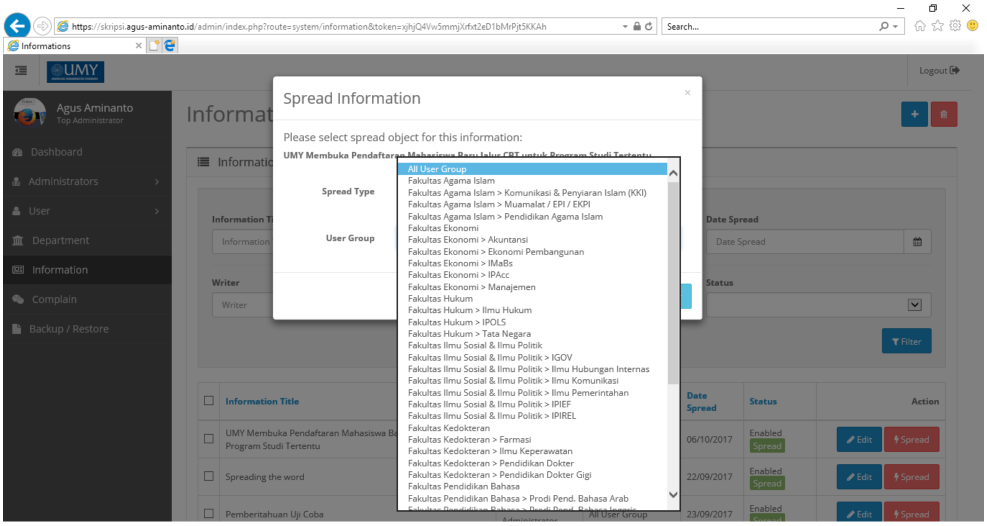
Gambar 4.45 Tampilan pada peramban *Google Chrome*



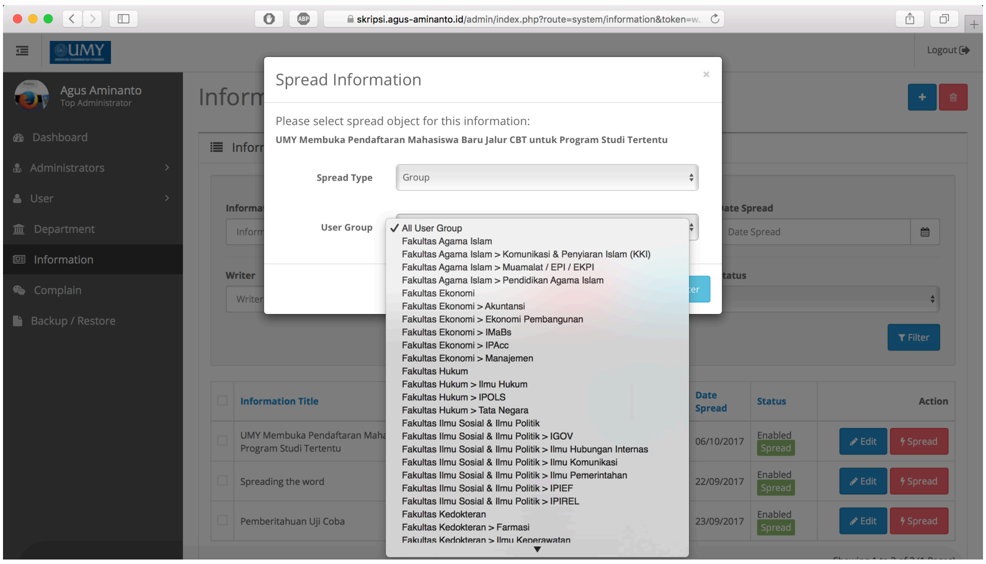
Gambar 4.46 Tampilan pada peramban *Mozilla Firefox*



Gambar 4.47 Tampilan pada peramban *Opera*



Gambar 4.48 Tampilan pada peramban *Internet Explorer*



Gambar 4.49 Tampilan pada peramban *Safari*

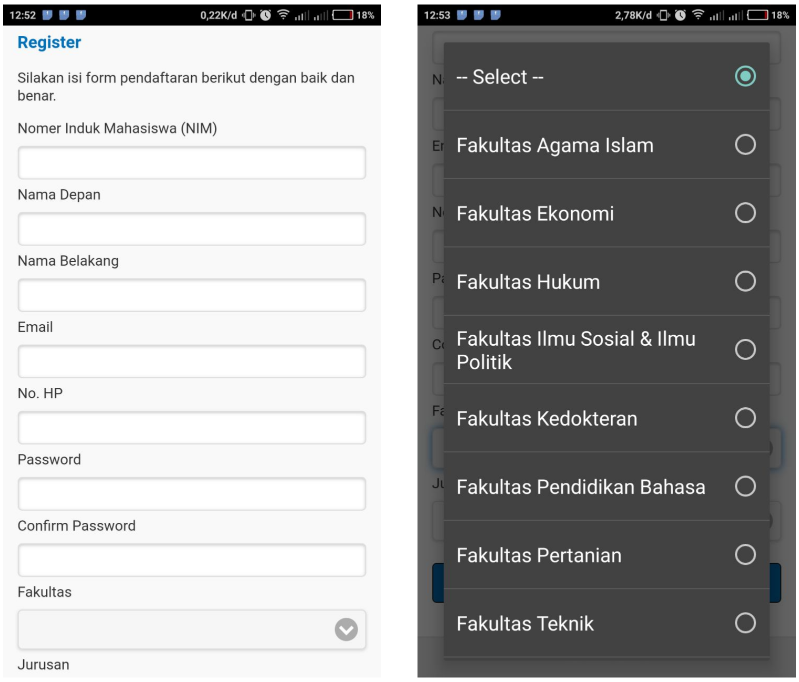
Jika kita perhatikan tampilan untuk semua peramban terlihat sama. Hanya ada sedikit perbedaan pada tampilan *drop down menu* pada *IE* dimana *alignment* daftar menu terlihat lebih keatas. Tidak tampil tepat dibawah *select box*. Namun semua *option* dalam *select box* tampil dengan baik. Daftar opsi dalam kotak pilihan juga memiliki *scroll bar* kecuali pada *Safari* yang mengharuskan pengguna untuk menggunakan fitur *scroll* pada *mouse* atau mengarahkan *pointer* pada tanda panah.

Dari pengujian sederhana diatas dapat diambil kesimpulan bahwa operasional sistem aplikasi web berjalan lebih pada pada *Firefox*, *Chrome* atau *Opera*. Ketiganya memiliki konsistensi tampilan untuk pengalaman pengguna yang lebih baik.

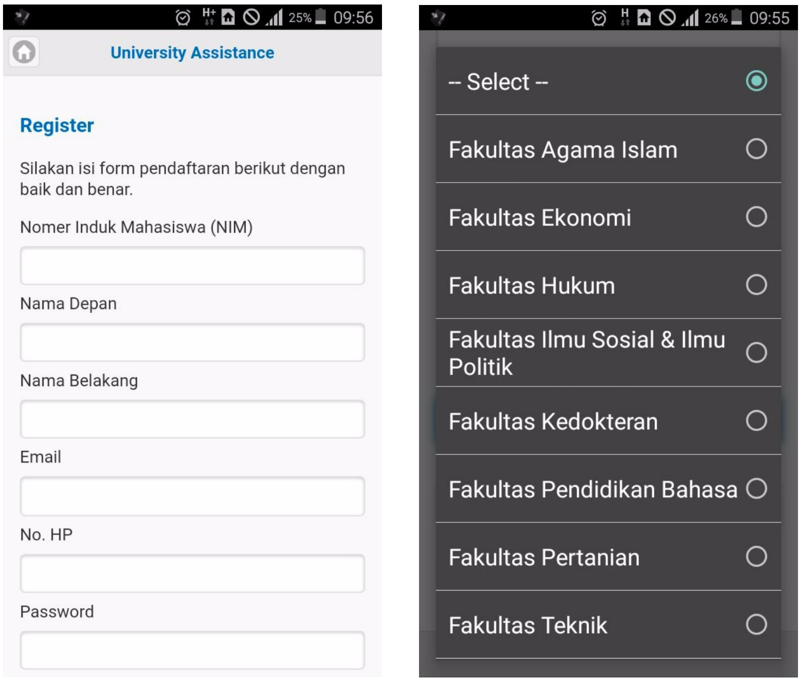
4.6 Tes Antarmuka Aplikasi Android Pengguna

Tes antarmuka ini dilakukan pada aplikasi Android pengguna untuk menguji fungsionalitas dasar dari antarmuka (*interface*) aplikasi secara umum. Seperti fungsionalitas navigasi, menu, tombol, link, kotak dialog dan layout. Untuk keperluan pengujian ini aplikasi diujikan pada beberapa merek *smartphone* yang cukup *mainstream* untuk penggunaan di Indonesia, seperti *Samsung*, *Asus*, *Xiaomi* dan *Oppo*.

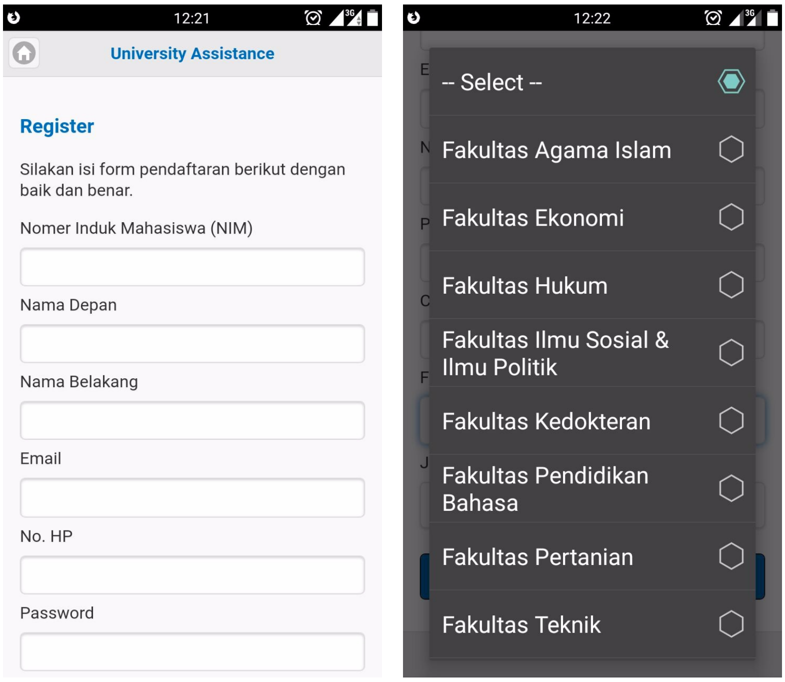
Secara umum spesifikasi untuk menjalankan aplikasi ini dibutuhkan *framework* Android 15. Interpretasi layout pada Android secara umum akan sama pada berbagai versi Android. Hal yang dapat membedakan adalah implementasi UI (*User Interface*) pada tiap vendor yang berbeda. Sehingga tes antarmuka ini dimaksudkan pula untuk menguji konsistensi tampilan pada berbagai pabrikan *smarthphone*. Adapun hasil tesnya ditunjukkan oleh Gambar 4.50 hingga Gambar 4.53 berikut.



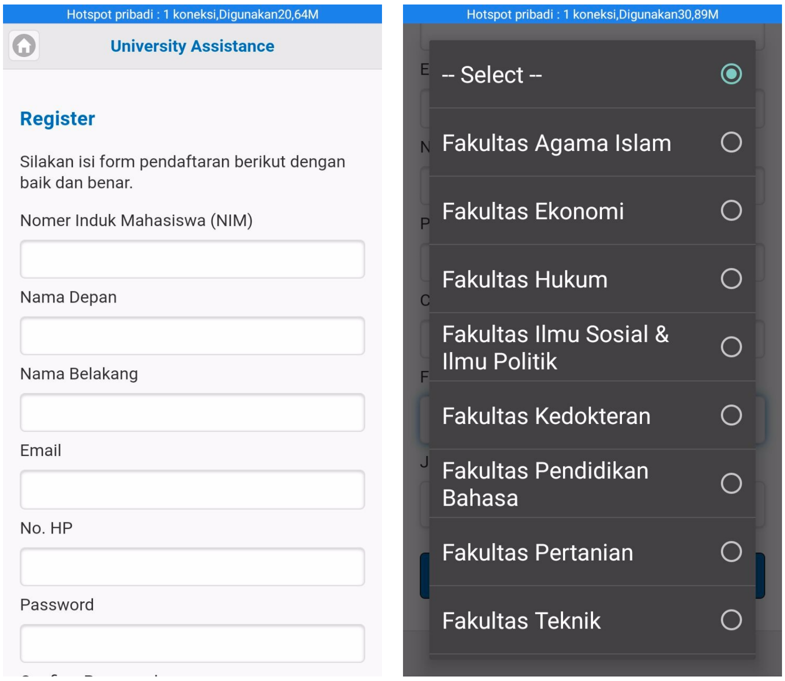
Gambar 4.50 Tes antarmuka pada *Smartphone Xiaomi*



Gambar 4.51 Tes antarmuka pada *Smartphone Samsung*



Gambar 4.52 Tes antarmuka pada *Smartphone Asus*



Gambar 4.53 Tes antarmuka pada *Smartphone Oppo*

Tes antarmuka pada aplikasi pada beberapa smarthphone terlihat tidak mengalami masalah, terutama pada *form input*. Ukuran *teks* juga cukup konsisten di banyak vendor *smartphone*. Lebih dari itu tiap form input berfungsi dengan baik.

4.7 User Usability Test

Tes ini digunakan untuk menguji apakah sistem tersebut akan bermanfaat, diterima *user* dan bertahan lama dalam penggunaannya. Sistem dengan *usability* yang tinggi akan membuat sistem tersebut populer dalam waktu lama dan luas penggunaannya karena banyak orang akan merasakan manfaatnya. Sedangkan, sistem dengan *usability* rendah meskipun dibuat berdasarkan kebutuhan, dan menghabiskan sumber daya yang tidak sedikit, seringkali pada akhirnya diabaikan oleh penggunanya.

Untuk keperluan tes usabilitas pada penelitian ini digunakan metode *USE Questionnaire* yang memiliki 3 parameter, yakni: *Usefullness*, *Satisfaction* dan *Ease of Use*. Secara teknis usabilitas ini dapat ditentukan mengggunakan *usability metrix* yang didapatkan dari data kuantitatif / kualitas yang diinginkan (*disired quality*).

Terdapat tahapan dalam pengukuran usabilitas yang dilakukan seperti halnya dalam penelitian, antara lain: pemilihan kuisioner, pemilihan partisipan dan penentuan ukuran sampel. Untuk mendapatkan data kuantitatif digunakan *USE Questionnaire,* mencakup 3 aspek pengukuran usability menurut ISO yaitu efisiensi, efektivitas dan kepuasan. Daftar kuisioner untuk tes ustabilitas untuk sistem aplikasi web dan aplikasi Android pengguna adalah sebagai berikut:

Usefulness

1. Aplikasi ini membantu saya menjadi lebih efektif
2. Aplikasi ini membantu saya menjadi lebih produktif
3. Aplikasi ini bermanfaat

4. Aplikasi ini memberikan saya kontrol terhadap aktifitas di kehidupan saya
5. Aplikasi ini membuat hal-hal yang ingin saya capai lebih mudah dilakukan
6. Aplikasi ini menghemat wkatu saya ketika saya menggunakannya
7. Aplikasi ini sesuai kebutuhan saya
8. Aplikasi ini dapat melakukan semua hal seperti yang saya harapkan

Ease of Use

9. Aplikasi ini mudah digunakan
10. Aplikasi ini sederhana penggunaannya
11. Aplikasi ini *user friendly*
12. Aplikasi ini membutuhkan langkah yang sedikit untuk menjalankan fungsi yang saya inginkan
13. Aplikasi ini fleksibel
14. Aplikasi ini *effortless* / tidak sulit untuk digunakan
15. Aplikasi ini dapat digunakan tanpa manual penggunaan
16. Saya tidak menemukan adanya ketidak-konsistensian saat menggunakannya
17. Baik pengguna yang jarang atau sering menggunakannya akan menyukainya
18. Saya dapat kembali dari kesalahan dengan cepat dan mudah
19. Saya dapat menggunakannya dengan baik tiap saat

Ease of Learning

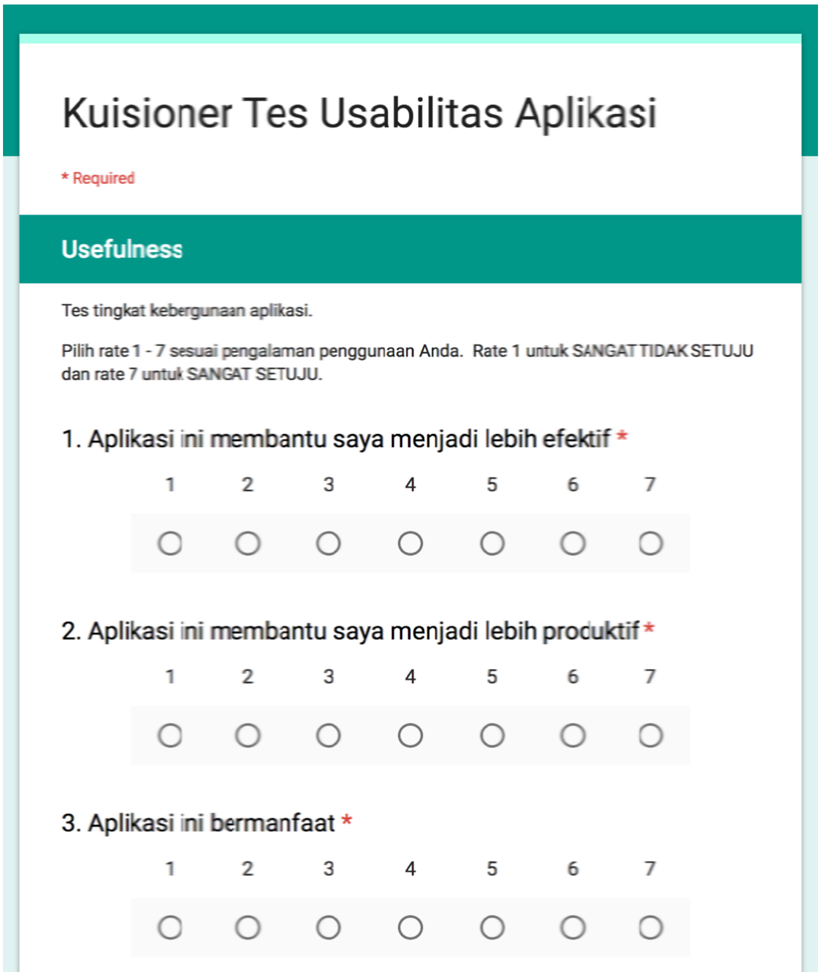
20. Saya belajar menggunakannya dengan cepat
21. Saya mengingat cara menggunakannya dengan mudah
22. Mudah untuk mempelajari cara penggunaannya
23. Saya dengan cepat menjadi mahir saat menggunakan aplikasinya

Satisfaction

24. Saya puas dengan aplikasinya

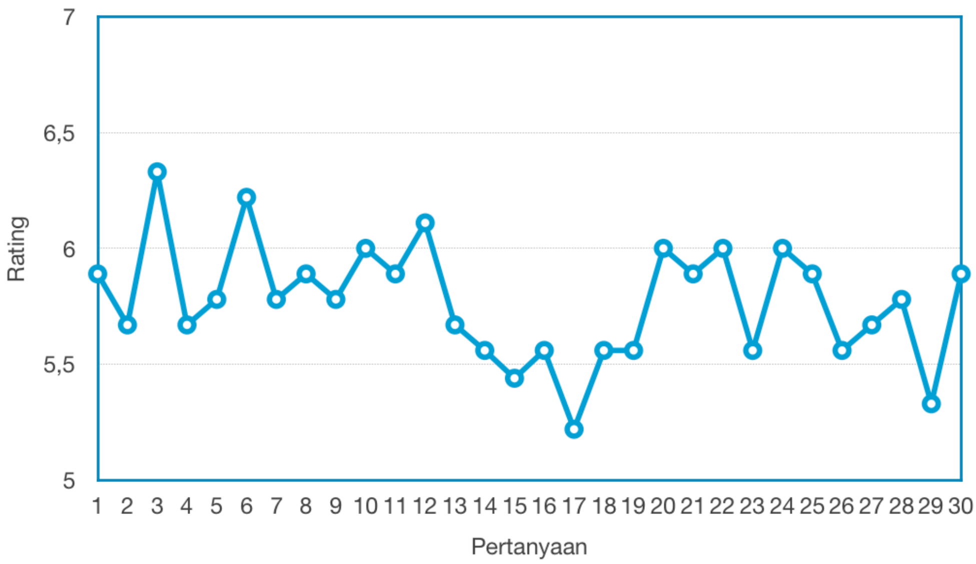
25. Saya ingin merekomendasikan aplikasi ini kepada teman
26. Menyenangkan menggunakan aplikasinya
27. Aplikasinya bekerja sesuai yang saya inginkan
28. Aplikasi ini sangat bagus
29. Saya rasa saya harus memiliki / menggunakannya
30. Saya nyaman menggunakan aplikasinya

Dari daftar pertanyaan diatas dapat diterapkan ke dalam sistem kuisioner yang dapat digunakan secara langsung seperti pada Gambar 4.54.



Gambar 4.54 Kuisioner tes usabilitas aplikasi

Untuk keperluan tes usabilitas aplikasi ini, sistem aplikasi diujikan pada 10 responden dengan mengikuti sesuai *use case* untuk sistem administrator dan pengguna sekaligus. Kemudian responden mengisi kuisioner yang berisi 30 pertanyaan diatas dimana tiap pertanyaan memiliki poin 1 – 7 untuk sangat tidak setuju dan sangat setuju. Dan hasil rata-rata dari tiap poin pertanyaan terdapat pada Gambar 4.56 berikut.



Gambar 4.55 Nilai rata-rata tiap poin pertanyaan pada kuisioner

Kemudian dari data diatas dapat pula diketahui nilai dari tiap parameter, yakni *usefullness, ease of use*, *ease of learning* dan *satisfaction*. Nilai ini disajikan pada Tabel 4.1 berikut.

Tabel 4.1 Nilai rata-rata untuk tiap parameter *USE Test*

Parameter	**Nilai**
Usefullness	5,9
Ease of use	6,23
Ease of learning	5,86
Satisfaction	5,73

Jika dilihat dari hasil diatas nilai terendah ada pada parameter *satisfaction* dan nilai tertinggi ada di *ease of use*. Nilai *satisfaction* yang rendah sering kali ada pada sistem yang sifatnya formal yang berhubungan langsung dengan instantasi seperti kampus. Sementara tingginya nilai *ease of use* menandakan bahwa sistem secara keseluruhan mudah digunakan baik pada sistem administrasi web maupun aplikasi Android. Nilai tertinggi kedua pada *usefullness* menandakan tingkat usabilitas aplikasi yang tinggi. Artinya sistem aplikasi secara umum fungsional sehingga bermanfaat.