TUGAS AKHIR – KI1502

IMPLEMENTASI OTENTIKASI *SINGLE SIGN ON* DAN OTORISASI *ROLE BASED ACCESS CONTROL* BERBASIS STANDAR OPENID CONNECT

KADEK WINDA DWIASTINI
NRP 05111440000008

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom., M.Eng.

Dosen Pembimbing II
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI1502

IMPLEMENTASI OTENTIKASI *SINGLE SIGN ON* DAN OTORISASI *ROLE BASED ACCESS CONTROL* BERBASIS STANDAR OPENID CONNECT

KADEK WINDA DWIASTINI
NRP 05111440000008

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom., M.Eng.

Dosen Pembimbing II
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

IMPLEMENTATION OF SINGLE SIGN ON AUTHENTICATION AND ROLE BASED ACCESS CONTROL AUTHORIZATION BASED ON OPENID CONNECT STANDARD

KADEK WINDA DWIASTINI
NRP 05111440000008

Supervisor I
Rizky Januar Akbar, S.Kom., M.Eng..

Supervisor II
Abdul Munif, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information And Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI OTENTIKASI *SINGLE SIGN ON* DAN OTORISASI *ROLE BASED ACCESS CONTROL* BERBASIS STANDAR OPENID CONNECT

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh
KADEK WINDA DWIASTINI
NRP : 05111440000008

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Rizky Januar Akbar, S.Kom., M.Eng. .....
   NIP: 19870103 201404 1 001 (Pembimbing 1)

2. Abdul Munif, S.Kom., M.Sc. .....
   NIP: 19860823 201504 1 004 (Pembimbing 2)

SURABAYA
JULI, 2018

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI OTENTIKASI *SINGLE SIGN ON* DAN OTORISASI *ROLE BASED ACCESS CONTROL* BERBASIS STANDAR OPENID CONNECT

Nama Mahasiswa	**: KADEK WINDA DWIASTINI**
NRP	**: 05111440000008**
Departemen	**: Informatika FTIK - ITS**
Dosen Pembimbing 1	**: Rizky Januar Akbar, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Abdul Munif, S.Kom., M.Sc.**

Abstrak

Institut Teknologi Sepuluh Nopember (ITS) memiliki banyak sistem informasi (SI) untuk menunjang jalannya operasi dan manajemen bisnis. Dengan terus berkembangnya jumlah SI yang dikelola dan dikembangkan ITS, maka login terpusat perlu untuk diimplementasikan. Single Sign On (SSO) digunakan untuk memudahkan pengguna dalam mengakses SI. Dimana pengguna hanya perlu melakukan login satu kali saja untuk dapat mengakses banyak SI. ITS SSO saat ini belum menerapkan suatu standar otentikasi dan otorisasi serta pengaturan akses pengguna belum dilakukan secara terpusat.

Tugas akhir ini akan menangani masalah tersebut dengan membuat suatu sistem otentikasi dan otorisasi terpusat menggunakan standar otorisasi OAuth2 dan standar otentikasi OpenID Connect. Dalam sistem akan diterapkan pengaturan akses atau role-based-access-control (RBAC) secara terpusat sehingga pengaturan akses dapat diatur dan diperlihara secara terpusat.

Kata kunci: Single Sign On, OpenID Connect, Role-Based-Access-Control

[Halaman ini sengaja dikosongkan]

IMPLEMENTATION OF SINGLE SIGN ON AUTHENTICATION AND ROLE BASED ACCESS CONTROL AUTHORIZATION BASED ON OPENID CONNECT STANDARD

Student's Name	**: KADEK WINDA DWIASTINI**
Student's ID	**: 05111440000008**
Department	**: Informatics Faculty of ICT - ITS**
First Advisor	**: Rizky Januar Akbar, S.Kom., M.Sc.**
Second Advisor	**: Abdul Munif, S.Kom., M.Sc.**

Abstract

Institut Teknologi Sepuluh Nopember (ITS) has many information systems (ISs) to support its business operations and management. Due to the growing numbers of ISs that are managed and developed, there is a need to use centralized login. Single Sign On (SSO) is used to ease users in accessing ISs. So that the users only need to do login once to access many ISs. ITS SSO has not applied any authentication and authorization standard and user access management is not done centrally.

This undergraduate thesis will overcome this problem by making a centralized authorization and an authentication system using OAuth2 as an authorization standard and OpenID Connect as an authentication standard. Centralized role-based-access-control (RBAC) will also be implemented so that user access management can be managed and maintained centrally.

Keywords: Single Sign On, OpenID Connect, Role-Based-Access-Control

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puja dan rasa syukur kehadapan Tuhan Yang Maha Esa yang atas izin dan karunianya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"IMPLEMENTASI OTENTIKASI *SINGLE SIGN ON* DAN OTORISASI *ROLE BASED ACCESS CONTROL* BERBASIS STANDAR OPENID CONNECT".**

Dalam pengerjaan tugas akhir ini, penulis mendapatkan banyak sekali ilmu baru dan memperdalam ilmu-ilmu yang sebelumnya telah diajarkan selama masa perkuliahan di Teknik Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Ida Sang Hyang Widhi Wasa atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Keluarga penulis di Bali, Ibu, Kakak dan tentu saja Bapak yang senantiasa menjadi sumber semangat penulis selama pengerjaan tugas akhir.
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku pembimbing I dan Bapak Abdul Munif S.Kom., M.Sc. selaku pembimbing II yang telah membantu, membimbing, dan memberikan ilmunya kepada penulis dalam menyelesaikan tugas akhir ini.
4. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Departemen Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
5. Rifat, Mas Rangga, Mas Yoga, Mas Umar, Mas Ali, Mbak Fitri dan rekan-rekan dari DPTSI yang telah banyak membantu di saat penulis kesulitan.

6. Teman-teman seperjuangan tugas akhir di laboratorium RPL: Farhan, Valdy, Faishal, Aldo, Nia, Rara, Elva, Raras, Steven, William, Sabila, Nurul serta kawan-kawan yang telah menemani dan menyemangati penulis selama pengerjaan tugas akhir.
7. Denny Krishnantara yang sering menjadi tempat keluh kesah sekaligus pemberi solusi dan semangat.
8. Teman-teman TPKH : Wira Mahardika, Fandy Aditya, Murata Nata, Bagus Andi, Dharmawan, dan semua TPKH Laksmana 14.
9. Resha yang secara sukarela telah mebuatkan beberapa logo dan menjadi penasehat UI.
10. Rina Wijaya dan seluruh teman-teman angkatan 2014 yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis.
11. Pihak lain yang telah membantu penulis menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan. Oleh karena itu, penulis menerima dengan rendah hati kritik dan saran untuk pembelajaran dan perbaikan kedepannya. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Juli 2018

Penulis

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Institut Teknologi Sepuluh Nopember Surabaya merupakan perguruan tinggi di bidang *science* dan teknologi juga salah satu perguruan tinggi terbesar di Indonesia. Untuk menunjang jalannya aspek pendidikan, manajemen, dan aspek lainnya, tentunya ITS mempunyai berbagai sistem berupa aplikasi-aplikasi yang mendukung. Aplikasi-aplikasi tersebut dapat diakses oleh seluruh civitas akademik ITS, mulai dari tenaga pengajar (dosen), mahasiswa, pegawai dengan menggunakan kredensial berupa *username* dan *password*. Setiap pengguna ingin mengakses suatu aplikasi, sistem akan melakukan verifikasi identitas dan pengguna diwajibkan untuk *login* dengan memasukkan *username* dan *password*. Misalkan aplikasi A dan B adalah aplikasi yang berada pada sistem atau jaringan ITS. Ketika pengguna sudah masuk ke dalam aplikasi A (*logged in*) dan ingin masuk ke aplikasi B, pengguna diharuskan melakukan *login* atau otentikasi kembali pada aplikasi B. Dengan jumlah sistem di ITS yang banyak, mekanisme *login* yang berulang seperti ini dapat menyulitkan pengguna dimana pengguna harus mengingat semua kredensial mereka pada setiap aplikasi. Selain itu akan terjadi redundasi penyimpanan kredensial pada setiap aplikasi.

Pendekatan yang digunakan aplikasi pada umumnya dalam proses identifikasi pengguna adalah dengan membuat *database* lokal untuk menyimpan akun dan kredensial pengguna. Namun, otentikasi lokal dapat menjadi masalah untuk sistem *enterprise* ITS yang memiliki banyak aplikasi. Teknologi *Single-Sign-On* (SSO) adalah teknologi *login* terpusat dimana sistem-sistem yang berbeda dalam satu perusahaan besar dapat terintegrasi dengan satu akun pengguna yang valid. Dengan menggunakan SSO, seorang pengguna hanya cukup melakukan proses otentikasi sekali saja untuk mendapat izin akses terhadap semua layanan yang terdapat

dalam jaringan. Dengan adanya SSO, pengguna hanya perlu mengingat satu *username* dan *password* yang bisa digunakan untuk masuk ke dalam semua sistem di ITS, sehingga dengan cara ini dapat memudahkan aplikasi untuk diintegrasikan tanpa harus membuat sistem otentikasi pada setiap aplikasi. ITS sudah menerapkan SSO, namun belum menggunakan standar sehingga fasilitas SSO ITS tidak dapat digunakan aplikasi eksternal ITS yang dalam hal ini merupakan aplikasi yang tidak diperlihara secara langsung oleh Direktorat Pengembangan Teknologi dan Informasi (DPTSI) ITS dan tidak dapat digunakan oleh aplikasi berbasis *mobile*.

Menggunakan protokol OpenID Connect pada sistem di ITS merupakan salah satu solusi dalam mengatasi permasalahan tersebut. Dengan menerapkan otentikasi dan otorisasi menggunakan standar OpenID Connect maka nantinya fasilitas *Single Sign On* ITS dapat digunakan tidak hanya oleh aplikasi-aplikasi dalam lingkup ITS, namun juga aplikasi luar, berbasis web maupun berbasis *mobile*. Selain kredensial pengguna yang akan tersimpan secara terpusat, hak akses pengguna pada tiap aplikasi juga akan diatur secara terpusat dengan metode *Role-Based Access Control* (RBAC) sehingga basis data terpusat ini akan memudahkan pemeliharaan sistem.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana mengimplementasikan otentikasi dan otorisasi menggunakan protokol OpenID Connect?
2. Bagaimana mekanisme *authorization code flow* pada OAuth 2.0?
3. Bagaimana menggabungkan *Role-Based Access Control* dan OpenID Connect?

1.3 Batasan Masalah

Permasalan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Sistem dibuat dengan menggunakan bahasa pemrograman PHP dengan kerangka kerja Phalcon dan dengan sistem manajemen basis data SQL Server.
2. Protokol otorisasi yang digunakan adalah OAuth 2.0 dan menggunakan standar OpenID Connect.
3. Menggunakan *authorization code flow* untuk tipe aplikasi web berbasis server.

1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah sebagai berikut:

1. Mengetahui mekanisme implementasi otentikasi dan otorisasi menggunakan OpenID Connect.
2. Mengetahui mekanisme *authorization code flow* pada OAuth 2.0.
3. Mengetahui mekanisme *Role-Based Access Control* (RBAC).

1.5 Manfaat

Manfaat yang diharapkan dari hasil tugas akhir ini antara lain adalah sebagai berikut:

1. Aplikasi yang menggunakan MyITS SSO dapat dimudahkan dalam hal otentikasi dan otorisasi yang terpusat sehingga tidak perlu dilakukan pemeliharaan kredensial user pada masing-masing aplikasi.
2. Aplikasi yang menggunakan MyITS SSO dapat mengatur hak akses tiap pengguna pada provider.

1.6 Metodologi

1.6.1 Penyusunan Proposal Tugas Akhir

Usulan tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Kemudian pada sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Tahap ini merupakan tahap pengumpulan informasi dan pembelajaran yang akan digunakan pada tugas akhir ini. Studi literatur meliputi diskusi dan pemahaman mengenai topik tugas akhir ini, diantaranya mengenai:

1. OpenID Connect
2. JWT (JSON Web Token)
3. Phalcon sebagai kerangka kerja (*framework*)
4. SQLServer sebagai basis data

1.6.3 Analisis dan Desain Perangkat Lunak

Adapun pembagian tahap analisa kebutuhan dan perancangan dari kerangka kerja sistem informasi akademik sebagai berikut:

1. Mempelajari kebutuhan umum sistem *Single Sign On* myITS secara garis besar

Tahap ini dilakukan agar dapat membuat rancang bangun sistem yang sesuai kebutuhan pada tahap selanjutnya.

2. Merancang sistem *Single Sign On* myITS
   Pada tahap ini dilakukan perancangan diagram kasus penggunaan dan basis data dalam bentuk *physical data model* dan *conceptual data model*.

1.6.4 Implementasi Perangkat Lunak

Sistem dibuat dengan menggunakan bahasa pemrograman PHP dengan kakas bantu phalcon dan penggunaan JSON dengan sistem manajemen basis data SQLServer.

1.6.5 Pengujian dan Evaluasi

Pengujian sistem ini dilakukan dengan metode *blackbox*. Pengujian *blackbox* merupakan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Tugas Akhir
   d. Tujuan
   e. Metodologi
   f. Sistematika Penulisan
2. Tinjauan Pustaka

3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem, rancangan basis data, diagram kasus penggunaan, diagram aktivitas dan rancangan antarmuka pengguna.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *pseudocode* yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

2.1 HTTP

HTTP *(Hypertext Transfer Protocol*) adalah protokol dasar yang digunakan oleh World Wide Web dan protokol ini mendefinisikan bagaimana pesan diformat, dikirimkan, dan tindakan apa yang dilakukan oleh server dan *web browser* (klien) dalam menanggapi berbagai perintah [1]. HTTP berfungsi sebagai protokol *request-response* pada model klien-server. Sumber daya yang hendak diakses dengan HTTP diidentifikasi dengan menggunakan *Uniform Resource Identifier* (URI), atau lebih khusus menggunakan *Uniform Resource Locator* (URL), dengan skema URI http atau https.

Klien atau web browser mengirimkan pesan permintaan (*request)* HTTP ke server. Server, yang menyediakan sumber daya seperti file HTML dan konten lainnya, atau melakukan fungsi lain atas nama klien, mengembalikan pesan tanggapan (*response*) ke klien. *Response* berisi informasi status penyelesaian atas *request* dan mungkin juga berisi konten yang diminta dalam badan pesannya (*message body*).

Pada tugas akhir, metode HTTP yang digunakan adalah sebagai berikut [2]:

1. GET

   Merupakan salah satu metode HTTP yang sering digunakan. Metode GET digunakan untuk meminta data dari *resource* tertentu. Pada *request* GET, *string query* berisi nama dan nilainya dikirim dalam URL seperi pada Gambar 2.1 di bawah.

```
http://my.its.ac.id?response_type=code&prompt=none
```

Gambar 2.1 HTTP GET

2. POST
   Metode POST digunakan untuk mengirimkan data ke server untuk membuat atau memperbaharui suatu *resource*. Data yang dikirim ke server melalui metode POST disimpan dalam badan (*body*) dari HTTP *request*. Metode ini juga merupakan salah satu metode yang umum digunakan. Gambar 2.2 berikut contoh *request* menggunakan metode POST.

```
POST /token HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded
  Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxS
bIA
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Gambar 2.2 HTTP POST

2.2 RESTful API

RESTful API/ REST API merupakan implementasi API (*Application Programming Interface*). REST (*Representation State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi. Dimana tujuannya adalah untuk menjadikan sistem yang memiliki performa yang baik, cepat, dan mudah untuk dikembangkan terutama dalam pertukaran dan komunikasi data. *RESTful API* memiliki 4 komponen penting di dalamnya, antara lain adalah sebagai berikut [3].

1. URL *Design*
   RESTful API diakses menggunakan protokol HTTP dimana penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah untuk dimengerti *developer*. URL API biasa disebut *endpoint* dalam pemanggilannya. Contoh penamaan URL/ endpoint yang

baik adalah: /user, /user/1234, /user/1234/photos, /user/1234/photos/abc.

2. HTTP *Verbs*
   Setiap *request* yang dilakukan terdapat metode yang dipakai agar server mengerti apa yang sedang di *request* oleh klien, diantaranya yang umum dipakai adalah GET, POST, PUT, dan DELETE.
3. HTTP *Response Code*
   HTTP Response Code adalah kode standarisasi dalam menginformasikan hasil *request* kepada klien. Secara umum terdapat 3 kelompok yang biasa dijumpai pada *RESTful API* yaitu:
   - 2xx, adalah kode respon yang menampilkan bahwa *request* berhasil.
   - 4xx, adalah kode respon yang menampilkan bahwa *request* mengalami kesalahan pada sisi klien.
   - 5xx, adalah kode respon yang menampilkan bahwa *request* mengalami kesalahan pada sisi server.
4. *Format Response*
   Setiap *request* yang dilakukan klien akan menerima data response dari server, respon tersebut biasanya berupa data XML ataupun JSON. Setelah mendapakan data response tersebut barulah klien bisa menggunakannya dengan cara mem-parsing data tersebut dan diolah sesuai kebutuhan.

2.3 PHP

PHP (PHP: *Hypertext Preprocessor*) adalah bahasa *scripting* pada sisi server yang umumnya digunakan untuk pengembangan web [4]. Kode PHP dapat disematkan ke *markup* HTML atau HTML5, atau dapat digunakan dalam kombinasi dengan berbagai sistem template web, sistem pengelolaan konten web (*web content management system*) dan kerangka web (*framework*). File PHP

dapat berisi text, HTML, CSS, Javascript, dan kode PHP. Kode PHP dieksekusi pada server dan hasilnya dikembalikan ke browser dalam bentuk HTML.

PHP dapat berjalan pada berbagai platform seperti Windows, Linux, Unix, Mac OS X, dan lainnya serta kompatibel dengan hampir seluruh server seperti Apache, IIS, dan lain sebagainya. PHP dapat menghasilkan konten web yang dinamis, membuat (*create*), membuka (*open*), menulis (*write)*, menghapus (*delete*), dan menutup (*close*) *file* pada server. Selain itu, PHP dapat mengirim dan menerima cookie, menambah, menghapus, dan memodifikasi data pada *database*, mengontrol akses pengguna, dan juga dapat mengenkripsi data.

Saat seorang pengguna mengetik URL http://example.org pada sebuah web client (*browser*), *browser* tersebut mengirimkan sebuah GET *request* ke server (diasumsikan menggunakan Apache). Setelah menerima *request* tersebut, Apache kemudian mencari file bernama index.php dan memberikan file tersebut ke PHP interpreter untuk dijalankan. PHP membaca keseluruhan file dan mengeksekusi semua kode PHP yang ditemukan. Setelah selesai, PHP intepreter memberikan output dari kode kembali ke Apache. Apache kemudian mengirimkan output yang didapat dari PHP ke browser untuk ditampilkan pada layar pengguna.

2.4 *Single Sign On*

Teknologi *Single Sign On* (SSO) adalah teknologi yang mengizinkan pengguna jaringan agar dapat mengakses sumber daya dalam jaringan hanya dengan menggunakan satu akun pengguna saja. Teknologi ini sangan menguntungkan, khusunya dalam jaringan yang sangat besar dan bersifat heterogen (di saat sistem operasi serta aplikasi yang digunakan oleh komputer berasal dari banyak *vendor*, dan pengguna diminta untuk mengisi informasi dirinya ke dalam setiap *platform* yang berbeda tersebut yang hendak diakses oleh pengguna) [5].

Dengan menggunakan SSO, seorang pengguna hanya cukup melakukan proses autentikasi sekali saja untuk mendapatkan izin akses terhadap semua layanan yang terdapat di dalam jaringan.

2.5 *Single Sign On* pada Integra

Sistem *Single Sign On* yang saat ini digunakan oleh ITS diterapkan pada Integra, dimana sistem-sistem internal yang dipelihara oleh Direktorat Pengembangan Sistem Informasi (DPTSI) ITS, seperti SI Akademik, SI SKEM, SI Beasiswa, SI Kurikulum, dan lainnya terintegrasi dan dapat diakses oleh pengguna hanya dengan satu kali *login*.

SSO pada Integra belum menggunakan suatu protokol atau standar tertentu. Sistem SSO saat ini diterapkan dengan menyimpan data pengguna yang sedang aktif pada *cookie* yang disimpan pada *browser* yang telah dienkripsi. Pada setiap aplikasi *client* yang telah terintegrasi seperti SI Akademik, SI SKEM, dan lainnya telah menggunakan *library* SSO dari Integra. Selain belum menggunakan suatu protokol tertentu, kelemahan dari sistem SSO ITS saat ini adalah pada masing-masing aplikasi *client* tetap harus mengakses *database* pada *its-gate* secara langsung untuk dapat membaca pembatasan akses pengguna atau *role base access control*.

2.6 Otentikasi dan Otorisasi

Otentikasi dan otorisasi diperlukan pada aplikasi atau sistem yang dibatasi untuk para pengguna tertentu. Otentikasi adalah verifikasi apakah seseorang itu adalah seseorang. Biasanya melibatkan *username* dan *password*, tapi dapat menyertakan metode lain yang menunjukan identitas, seperti kartu pintar, sidik jari, dan lain-lain.

Otorisasi adalah pencarian apakah orang yang sudah diidentifikasi (diotentikasi) diijinkan untuk memanipulasi sumber daya tertentu. Ini biasanya ditentukan dengan mencari apakah

orang itu merupakan bagian dari aturan khusus yang memiliki akses ke sumber daya. Otorisasi biasanya muncul setelah otentikasi. *Role Based Access Control* biasanya yang diimplementasikkan untuk otorisasi [6].

2.7 OpenID Connect

OpenID Connect adalah standar baru yang muncul untuk *single-sign-on* dan penyedia identitas di internet. OpenID Connect merupakan lapisan identitas di atas OAuth 2.0, yang menggunakan semantik dan aliran (*flow*) dari OAuth 2.0 untuk memungkinkan aplikasi klien mengakses identitas pengguna, yang dikodekan sebagai ID Token dalam JSON Web Token (JWT) [7]. Token ID menyerupai konsep kartu identitas dalam format JWT standar yang ditandatangani oleh OP (OpenID Provider). Untuk mendapatkan ID Token, klien perlu mengirim permintaan otentikasi ke OP.

OAuth 2.0 adalah protokol terbuka yang memungkinkan otorisasi aman dengan metode standar dan sederhana dari aplikasi *web, mobile*, maupun *desktop*. OAuth tidak mempunyai lapisan otentikasi. Ruang lingkup penggunaannya adalah otorisasi. OAuth memungkinkan server untuk mendelegasikan akses yang aman bagi pihak ketiga untuk menggunakan sumber daya terbatas dengan ijin dari pemilik sumber daya [8]. Terdapat 3 jenis aliran (*flows*) pada OAuth yang didesain untuk tipe aplikasi: aplikasi web tradisional berbasis server, aplikasi browser (JavaScript), dan aplikasi mobile. Ketiga flow tersebut adalah sebagai berikut.

1. Authorization Code Flow

Adalah aliran kode yang paling umum digunakan, ditujukan untuk aplikasi web tradisional serta aplikasi mobile. Melibatkan pengalihan *browser* awal (initial browser redirection) ke/ dari OP (OpenID Provider) untuk mendapatkan otentikasi dan persetujuan dari pengguna, kemudian mengirimkan permintaan (*request*) melalui back-channel untuk mengambil ID Token. Aliran ini

menawarkan keamanan optimal, karena token tidak dibeberkan ke browser dan aplikasi klien juga dapat diotentikasi.

2. Implicit Flow

Ditujukan untuk aplikasi berbasis browser (JavaScript) yang tidak memiliki *back-end*. ID Token didapatkan langsung dengan respon pengalihan (*redirection response*) dari OP. Dalam hal ini tidak diperlukan *back-channel.*

3. Hybrid Flow

Memungkinkan aplikasi *front-end* dan *back-end* untuk menerima token secara terpisah dari satu sama lainnya. Merupakan kombinasi dari kedua aliran sebelumnya. Aliran ini jarang digunakan.

Authorization Code Flow

Berikut adalah contoh bagaimana mendapatkan ID Token sebuah pengguna dari OP (*OpenID Provider*) menggunakan *authorization code flow.* Aliran kode memiliki 2 langkah seperti Gambar 2.3 berikut.

	Step 1	Step 2
Purpose	1. Authenticate user 2. Receive user consent	1. Authenticate client (optional) 2. Exchange code for token(s)
Channel *Request*	Front-channel *request* (browser redirection)	Back-channel *request* (app to web server)
Endpoint	Authorisation endpoint	Token endpoint
Result on success	Authorisation code	ID Token

Gambar 2.3 Langkah-langkah Authorization Code Flow

Code Flow: Langkah 1

Klien menginisiasi otentikasi pengguna dengan mengalihkan browser ke OAuth 2.0 authorisation endpoint dari OpenID Provider. Permintaan otentikasi OpenID pada dasarnya adalah permintaan otorisasi OAuth 2.0 untuk mengakses identitas pengguna ditunjukan oleh nilai openid pada parameter scope. Contoh pengalihan otentikasi ke OP adalah pada Gambar 2.4 berikut.

```
HTTP/1.1 302 Found
Location: https://openid.c2id.com/login?
          response_type=code
          &scope=openid
          &client_id=s6BhdRkqt3
          &state=af0ifjsldkj

&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Gambar 2.4 Pengalihan Otentikasi ke OP

Parameter *request* dikodekan dalah kueri URI:

- response_type mengatur response_type ke code untuk mengindikasikan atau menandakan authorisation code flow.
- scope digunakan untuk menentukan scope dari permintaan otorisasi pada OAuth. Nilai scope yaitu openid menandakan sebuah permintaan untuk otentikasi OpenID dan ID Token
- client_id adalah sebuah client identifier dari klien pada OP. Identifier ini biasanya diperoleh saat klien terdaftar dengan OP melalui API pendaftaran klien, konsol pengembang, atau metode lainnya
- state merupakan nilai buram yang ditetapkan oleh klien untuk menjaga keadaan (state) antara permintaan (*request*) dan callback.

- redirect_uri URI callback klien untuk respon otentikasi.

Pada OP, pengguna biasanya akan diotentikasi dengan memeriksa apakah mereka memiliki session yang valid (ditentukan oleh cookie browser) dan ketika tidak valid maka pengguna akan diarahkan untuk log in. Setelah itu, pengguna akan ditanya apakah mereka setuju untuk sign in ke dalam aplikasi klien. OP kemudian akan memanggil redirect_uri klien dengan kode otorisasi sukses atau kode error jika akses ditolak.

Code Flow: Langkah 2

Kode otorisasi adalah kredensial penengah (intermediate credential) yang mengkodekan otorisasi yang didapatkan dari langkah 1. Oleh karena itu klien masih merasa kabur dan kode tersebut hanya memiliki arti bagi server OP. Untuk mendapatkan kembali ID Token, klien diharuskan untuk mengirimkan kode tersebut ke OP namun kali ini memerlukan permintaan direct back-channel. Pertukaran kode dengan token terjadi pada token endpoint dari OP dapat dilihat pada Gambar 2.5:

```
POST /token HTTP/1.1
Host: openid.c2id.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Gambar 2.5 ***Token Request***

ID dan rahasia klien dikirimkan melalui header Authorization. Berikut adalah parameter token *request*:

- grant_type mengatur authorization_code
- code kode yang didapatkan dari langkah 1

- redirect_uri mengulangi callback URI dari langkah 1.

Jika sukses, OP akan mengembalikan objek JSON yang berisi ID Token, token akses, dan token refresh opsional seperti pada Gambar 2.6.

```
   HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
  {
      "id_token":
      "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.
      ewogImlzcyI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUu
      Y29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiwKICJh
      dWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0w
      UzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAog
      ImlhdCI6IDEzMTEyODA5NzAKfQ.ggW8hZ1EuVLuxNuuI
      JKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6qJp6
      IcmD3HP99Obi1PRswh3LO146waJ8IhehcwL7F09Jdijm
      BqkvPeB2T9CJNqeGpegccMg4vfKjkM8FcGvnzZUN4_KS
      P0aAp1tOJ1zZwgjxqGByKHiOtX7TpdQyHE5lcMiKPXfE
      IQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbO
      EoRoSK5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3
      _osjzw2PAithfubEEBLuVVk4XUVrWOLrLl0nx7RkKU8N
      XNHq-rvKMzqg"
      "access_token": "SlAV32hkKG",
      "token_type": "Bearer",
      "expires_in": 3600,
   }
```

Gambar 2.6 ***Token Response***

Userinfo Endpoint

Pada *endpoint* ini RP (*Relying Party*) atau aplikasi *client* mendapatkan *userinfo* pengguna sesuai *scope* yang diberikan pada *authorization request*. *Userinfo Endpoint* diakses dengan

mengirimkan *userinfo request* menggunakan HTTP GET ataupun HTPP POST dan menyertakan *access token* pada bagian *header*. Gambar 2.7 berikut adalah contoh dari *userinfo request*.

```
GET /userinfo HTTP/1.1
  Host: server.example.com
  Authorization: Bearer SlAV32hkKG
```

Gambar 2.7 *Userinfo Request*

Jika *access token* yang diberikan valid, makan OP akan mengirimkan *userinfo response* seperti pada Gambar 2.8 di bawah.

```
HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "preferred_username": "j.doe",
   "email": "janedoe@example.com",
   "picture": "http://example.com/janedoe/me.jpg"
  }
```

Gambar 2.8 *Userinfo Response*

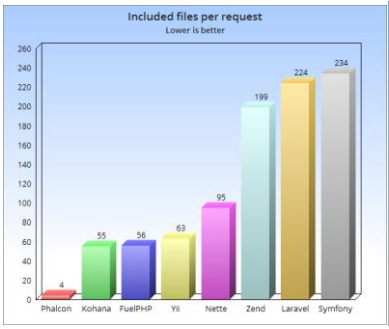
2.8 Phalcon

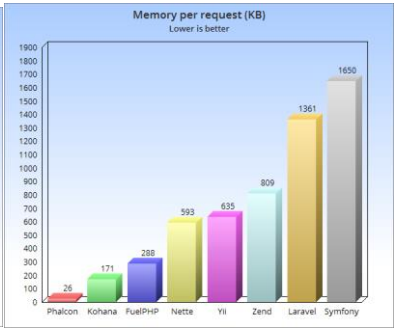
Phalcon adalah kerangka kerja *open-source* berbasis web PHP yang menggunakan pola MVC (model-view-controller). Tidak seperti kerangka kerja PHP pada umumnya, Phalcon dibangun menggunakan Zephir dan C dengan tujuan untuk meningkatkan kecepatan eksekusi, mengurangi penggunaan sumber daya, dan dapat menangani lebih banyak HTTP *request* per detik.

Kerangka kerja ini menyediakan mekanisme *autoloading* kelas PHP mengikuti PSR-4. Untuk data dan penyimpanan

(*storage*), Phalcon menyediakan ORM (*object Relational Mapping*), PHQL (Phalcon Query Language), ODM untuk MongoDB (pemetaan objek dokumen MongoDB). Untuk tampilan dan *frontend*, Phalcon memiliki *template machine* yang dibangun dari C untuk PHP bernama volt [9].

Melalui *benchmark* terhadap *framework* lain (Kohana, FuelPHP, Yii, Nete, Zend, Laravel, Symfony) [10] yang dilakukan phalcon, didapatkan bahwa menggunakan phalcon dapat meningkatkan performa aplikasi. *Benchmark* yang dilakukan mengukur waktu yang dibutuhkan setiap *framework* untuk memulai, menjalankan setiap aksi, menampilkan hasil dan membersihkan *resource* pada akhir *request*. *Benchmark* dilakukan menggunakan ab *benchmarking tool* dari Apache untuk uji coba ini dengan 1000 *request* menggunakan 5 koneksi untuk masing-masing *framework*. Dari hasil yang ditunjukan Gambar 2.10 didapatkan phalcon membutuhkan paling sedikit file yang harus diproses dalam satu *request* dan yang paling banyak memproses file adalah *framework* Symfony. Gambar 2.9 menunjukan hasil bahwa phalcon membuthkan paling sedikit menggunakan *memory* dalam tiap *request*, disusul oleh Kohana, FuelPHP.

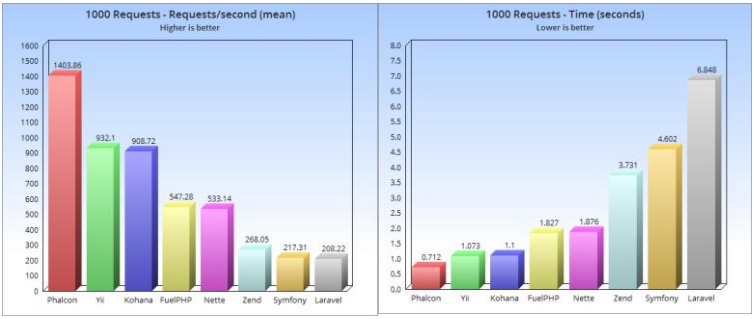




Gambar 2.10 ***Benchmark phalcon (1)*** **Gambar 2.9** ***Benchmark phalcon (2)***

Gambar 2.12 menunjukan grafik hasil bahwa phalcon dapat menangani sebanyak 1403.86 *request* dalam satu detik, yaitu tertinggi dari *framework* lainnya. Gambar 2.11 menunjukan grafik

bahwa phalcon membutuhkan paling sedikit waktu untuk menyelesaikan 1000 *request* yaitu selama 0.712 detik, dan yang paling lama adalah *framework* Laravel yaitu selama 6.848 detik. Dari keempat hasil tersebut dapat disimpulkan phalcon lebih unggul daripada *framework* lainnya.



Gambar 2.12 ***Benchmark phalcon (3)*** **Gambar 2.11** ***Benchmark phalcon (4)***

Keunggulan dari *framework* phalcon adalah performanya yang lebih cepat dibandingkan *framework* lain karena tidak mengkonsumsi banyak *resource*, ringkas dan efisien, berinteraksi dengan *database* lebih optimal karna menggunakan 'C *language ORM'* untuk PHP, dan mengusung konsep MVC yang *familiar*. Namun terdapat kekurang juga pad *framework* ini, yaitu kurangnya tutorial dan pemecahan masalah dari pihak ketiga (apalagi dalam bahasa Indonesia), dikarenakan umur *framework* yang masih sangat muda. Kemudian ketika ditemukan *error* dan *bug* pada *framework* phalcon, pemecahan masalah dilakukan lebih lambat karna menggunakan bahasa C [11].

2.9 JWT (JSON Web Token)

JSON Web Token merupakan spesifikasi yang dirancang untuk keperluan otentikasi dan pertukaran pesan yang aman. JWT adalah standar terbuka (RFC 7519) yang mendefinisikan cara yang

ringkas dan bisa dilihat pada isinya untuk mentransmisikan informasi secara aman atar berbagai pihak sebagai objek JSON. Informasi ini bisa diverifikasi dan dipercaya karena sudah ditandatangani secara digital. JWT dapat ditandatangani dengan menggunakan rahasia (dengan algoritma HMAC) atau pasangan kunci publik / swasta menggunakan RSA [8].

Pesan yang dikirimkan akan diserialisasikan dalam bentuk objek JSON. Struktur JWT terdiri atas 3 bagian, dipisahkan oleh tanda titik (.): <base64-encoded header>.<base64-encoded claims>.<base64-encoded signature>. Berikut ini adalah penjelasan masing-masing.

1. Header

Header biasanya terdiri dari dua bagian: jenis token, yaitu JWT, dan algoritma hashing yang digunakan, seperti HMAC SHA256 atau RSA. Contoh:

```
{
   "typ":"JWT",
   "alg":"HS256"
}
```

2. Payload

Payload berisi klaim. Klaim adalah pernyataan tentang entitas (biasanya, pengguna) dan metadata tambahan. Terdapat tiga jenis klaim, yaitu:

a. Registered Claims

Registered claim adalah serangkaian klaim yang telah ditetapkan sebelumnya yang tidak wajib namun disarankan, untuk menyediakan satu set *interoperable claim* yang berguna. Beberapa diantaranya adalah: iss *(issuer*), exp (*expiration time*), sub (*subjek*), aud (*audience*), dan lain-lain. *Reserved claims* dapat dilihat pada Gambar 2.13 di bawah.

Atribut	Tipe	Keterangan
iss	String	*Issuer Claim*. untuk mengidentifikasi aplikasi client
iat	Number	*Issued Time*, tanggal (UTC Unix time) token di buat (dalam second)
exp	Number	*Expire Time*. Masa berlaku token (dalam second)
sub	String	Subjek token
aud	String	Audien token, *client*
nbf	String	*Not Before*
jti	Long	JWT ID

Gambar 2.13 ***Registered Claims JWT***

b. Public Claims

Public claim dapat didefinisikan sesuka hati oleh mereka yang menggunakan JWT. Tapi untuk menghindari terjadinya *collision* yaitu nilai klaim yang sama dengan yang sudah ada (*reserved claim)*, harus didefinisikan di IANA JSON Web *Token Registry* atau didefinisikan sebagai URI yang berisi *namespace collision*.

c. Private Claims

Private claims adalah klaim yang tidak termasuk *registered claim* maupun *public claim* [12].

3. Signature

Signature digunakan untuk memverifikasi bahwa pengirim JWT adalah yang dikatakannya dan untuk memastikan bahwa pesan tersebut tidak berubah sepanjang jalan. Signature berisi header + payload + secret pass, dienkripsi menggunakan algoritma yang terdapat pada header (nilai dari alg).

Sebuah kunci rahasia (*secret key*) digunakan untuk mengenkripsi data berisi user-id dalam format JSON. Enkripsi data dengan kunci tersebut akan menghasilkan sebuah token yang dikirim ke klien dan digunakan dalam setiap *request*. Setiap kali

klien mengirimkan *request* menggunakan token, server akan mencoba untuk mendekripsi token tersebut menggunakan kunci (*secret key*) yang sama dengan sebelumnya. Jika berhasil, maka akan didapatka user-id dari data JSON yang sesuai dengan pengguna. Masa berlaku token (*expire time*) dapat diatur pada bagian claim dalam satuan detik.

Salah satu kelemahan dari JWT adalah JWT hanya menggunakan satu kunci. Jika kunci tersebut bocor dan disalahgunakan, maka keseluruhan sistem akan terganggu. Salah satu cara untuk mengatasi permasalahan tersebut adalah dengan men¬-generate kunci baru (*key-pair*). Kelemahan lainnya adalah ukuran data token JWT akan lebih besar dari sekedar token session normal. Semakin banyak data yang ditambahkan di token JWT, maka akan semakin lama pula token dibuat (*generate*).

Pada tugas akhir, *payload* atau klaim yang akan digunakan adalah sub, iss, aud, iat, dan exp. Berikut adalah contoh penggunaannya. Sub atau ID pengguna bernilai ‘alice’, ‘https://openid.c2id.com’ sebagai *issuer*, *‘client-12345’* sebagai nama *client*, *iat* merupakan waktu token dibuat (dalam detik) dan *exp* waktu token *expire* (dalam detik).

```
{
  "sub"        : "alice",
  "iss"        : "https://openid.c2id.com",
  "aud"        : "client-12345",
  "iat"        : 1311280970,
  "exp"        : 1311281970,
}
```

Gambar 2.14 *ID Token Payload*

2.10 OAuth2 Server Library (PHP) Bshaffer

OAuth2 Server Library Bshaffer merupakan library *open-source* OAuth2 Server dan OpenID Connect yang dapat implementasikan pada aplikasi berbasis PHP. Library tersedia pada

Git *https://github.com/bshaffer/oauth2-server-php*. Untuk mengimplementasikan *library* ini, dibutuhkan PHP 5.3.9 ke atas. *Library* ini menggunakan standar PSR-0 Zen. Untuk mengimplementasikan *library*, dapat dilakukan menggunakan *composer* dengan perintah seperti:

```
composer.phar require bshaffer/oauth2-server-php "1.10"
```

Library Bshaffer ini melibatkan beberapa konsep utama, diantaranya sebagai berikut [13].

1. *Grant Types*
   Grant types memungkinkan *client* untuk menerima *access token* dalam beberapa cara. Berikut adalah *grant types* yang didukung dalam *library*.
   - Authorization Code
     Grant type ini merupakan yang paling umum digunakan. *Authorization code* mengimplementsi *3-Legged OAuth* dan melibatkan pengguna dalam mengizinkan *client* untuk mendapatkan *authorization code* yang dapat ditukarkan dengan *access token*.
   - Resource Owner *Password* Credentials
     Username dan *password* pemilik sumber daya diberikan sebagai bagian dari *request*, dan sebuah token diberikan jika otentikasi berhasil.
   - Client Credentials
     Client menggunakan kredensialnya untuk mendapatkan *access token* secara langsung sehingga dapat mengakses sumber daya dibawah pengawasan *client*.
   - Refresh Token
     Client dapat mengajukan sebuah *refresh token* dan mendapatkan *access token* baru jika *access token* telah kadaluarsa.
   - Implicit

Yang berbeda dengan *grant type authorization code* adalah token diberikan saat *authorization request* dikirimkan oleh client dan bukan *authorization*. *Grant type* ini umum digunakan untuk perangkat sisi *client* (perangkat *mobile*) dimana kredensial *client* tidak dapat disimpan dengan aman.

- JWT Bearer
  Client dapat mengajukan JWT (JSON Web Token) dalam *request* ke *token endpoint*. *Access token (*tanpa *refresh token*) akan diberikan secara langsung sebagai balasan.

2. *Controllers*
   Pada umumnya OAuth API akan memiliki *endpoint* untuk a*uthorization request, token request,* dan *resource request* Objek OAuth/Server memiliki metode untuk menangani tiap *request* tersebut yaitu dengan controller: *authorize controller, token controller,* dan *resource controller*.
3. *Storage Objects*
   Library ini mendukung adapter untuk beberapa mesin penyimpanan (*storage engines*). Beberapa diantaranya adalah PDO (untuk MySQL, SQLite, PostgreSQL, dll), MongoDB, Redis, dan Cassandra.

2.11 Kriptografi Asimetris

Kriptografi merupakan suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Kriptografi terdiri dari dua proses utama yakni proses enkripsi dan proses dekripsi. Proses enkripsi mengubah *plaintext* menjadi *ciphertext* dengan menggunakan kunci tertentu sehingga isi informasi pada pesan tersebut sukar dimengerti.

Terdapat dua jenis kriptografi, yaitu kriptografi asimetris dan kriptografi simetris. Kriptografi simetris yakni kriptografi yang dalam proses enkripsi dan dekripsinya menggunakan sebuah

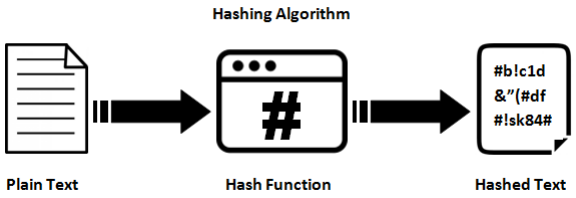
kunci yang sama. Sedangkan kriptografi asimetris menggunakan kunci yang berbeda dalam proses enkripsi dan dekripsi.

Kriptografi asimetris *(asymmetric cryptography*), juga dikenal dengan kriptografi *public key*, yang menggunakan *public* dan *private key* untuk mengenkripsi dan dekripsi data. Kuncinya (key) menggunakan bilangan yang besar yang telah dipasangkan tetapi tidak identikal (asimetris). Satu kunci dapat dibagikan dengan semua orang dinamakan *public key*. Satu kunci lainnya harus dirahasiakan bernama *private key*. Salah satu kunci tersebut dapat digunakan untuk mengenkripsi data, dan kunci satunya dapat digunakan untuk dekripsi [14].

Pada tugas akhir, kriptografi asimetris digunakan untuk mengenkripsi ID Token yang diberikan pada *token endpoint*. *Signature* pada ID Token akan dienkripsi menggunakan algoritma RS256 dan juga menggunakan *private key* masing-masing aplikasi *client* yang telah di-*generate* saat registrasi *client*. Kemudian, untuk melakukan validasi *signature* pada ID Token, *client* menggunakan *public key*-nya.

2.12 *Hashing*

Dalam ilmu komputer, fungsi *hashing* biasanya dipelajari bersama-sama dengan kriptografi, yaitu ilmu tentang pembuatan pesan rahasia dan cara memecahkannya. Secara sederhana, fungsi *hashing* adalah sebuah fungsi yang digunakan untuk mengacak sebuah kata menjadi kata lain yang tidak bermakna dan sedapat mungkin kata hasil hashing tidak bisa ditebak dari kata aslinya. Pada umumnya fungsi hashing diterapkan saat terjadi pertukaran data sensitif seperti *password* pengguna. Untuk menjaga keamaan *password* pengguna, aplikasi harus melakukan hashing terhadap *password* terlebih dahulu. Pada Gambar 2.15 dijelaskan proses *hashing* dimana terdapat sebuah input yaitu *plain text* kemudian diproses menggunakan fungsi *hash (*algoritma *hash*) sehingga menghasilkan *output* berupa *hashed text*.



Gambar 2.15 Ilustrasi *Hashing*

Ada beberapa tipe atau algoritma *hash*, diantaranya [15]:

- MD4 (*Message-Digest algorithm 4*)
  MD4 dibuat oleh Ronald Rivest, merupakan fungsi hash yang dipakai sebelum MD5, namun karena banyaknya kelemahan, maka diganti oleh MD5. Panjang 16 bytes (32 karakter). Contoh: 31d6cfe0d16ae931b73c59d7e0c089c0.
- MD5 (*Message-Digest algorithm 5*)
  MD5 didesain oleh Ronald Rivest pada tahun 1991, yang menggantikan fungsi hash sebelumnya, MD4. Digunakan di phpBB v2.x, Joomla versi dibawah 1.0.13, dan digunakan oleh beberapa CMS dan forum. Panjang 16 bytes (32 karakter).
- SHA-1 (*Secure Hash Algorithm*)
  SHA-1 diciptakan oleh National Institure of Standars and Technology atau U.S. Federal Information Processing Standard. Panjang 20 bytes (40 karakter). Contoh: 356a192b7913b04c54574d18c28d46e6395428ab.
- SHA-256 (*Secure Hash Algorithm*)
  Hashnya dimulai oleh tanda 5 kemudian dilanjutkan dengan saltnya (8 karakter yang disusun secara acak) lalu dilanjutkan oleh karakter $, kemudian dilanjutkan oleh hashnya. Panjang 55 karakter. Contoh: $5$12345678$jBWLgeYZbSvREnuBr5s3gp13vqi…
- SHA-512 (*Secure Hash Algorithm*)
  Hashnya dimulai oleh tanda 6 kemudian dilanjutkan oleh saltnya, lalu dilanjutkan oleh karakter $ kemudian dilanjutkan oleh hashnya. Panjang 98 karakter. Contoh: $6$12345678$U6Yv5E1lWn6mEESzKen42o6rbEm…

– Base64

  Algoritma yang berfungsi untuk *encoding* dan *decoding* suatu data ke dalam format ASCII. Panjang maksimal 64 karakter terdiri dari A..Z, a..z, dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol + dan / serta satu buah karakter sama dengan (=). Contoh: Y3liZXJfY3JpbWluYWw=

Fungsi Hash merupakan suatu fungsi yang mengubah *key* menjadi alamat dalam tabel. Fungsi Hash memetakan sebuah *key* ke suatu alamat dalam tabel. Idealnya, *key-key* yang berbeda seharusnya dipetakan ke alamat-alamat yang berbeda juga. Pada kenyataannya, tidak ada fungsi *Hash* yang sempurna. Kemungkinan besar yang terjadi adalah dua atau lebih *key* yang berbeda dipetakan ke alamat yang sama dalam tabel. Peristiwa ini disebut dengan *collision* (tabrakan). Karena itulah diperlukan langkah berikutnya, yaitu *collision resolution* (pemecahan tabrakan).

Collision resolution merupakan proses untuk menangani kejadian dua atau lebih *key* di-*hash* ke alamat yang sama. Cara yang dilakukan jika terjadi *collision* adalah mencari lokasi yang kosong dalam tabel Hash secara terurut. Cara lainnya adalah dengan menggunakan fungsi *Hash* yang lain untuk mencari lokasi kosong tersebut [16].

Pada tugas akhir, *hashing* digunakan pada *password* pengguna menggunakan algoritma *hash* SHA-256. Sebelum di *hash, password* juga akan diberi *salt*. *Salt* antar satu pengguna dan pengguna lainnya berbeda dan disimpan pada tabel *user* pada basis data.

2.13 *Role Based Access Control* (RBAC)

Role based access control (RBAC) atau kontrol akses berbasis peran membatasi akses jaringan berdasarkan peran seseorang dalam suatu organisasi dan telah menjadi salah satu metode utama untuk kontrol akses lanjutan. Peran dalam RBAC mengacu pada tingkat akses yang dimiliki pengguna ke jaringan.

Pengguna hanya diperbolehkan mengakses informasi yang diperlukan untuk menjalankan tugas pekerjaan sesuai peran yang dimilikinya. Akses dapat didasarkan pada beberapa faktor, seperti wewenang, tanggung jawab, dan kompetensi pekerjaan. Selain itu, akses ke sumber daya dapat terbatas pada tugas-tugas tertentu, seperti kemampuan untuk melihat, membuat, atau memodifikasi file.

Mengelola dan mengaudit akses jaringan sangat penting untuk keamanan informasi. Akses dapat dan harus diberikan berdasarkan kebutuhan untuk tahu. Dengan ratusan atau ribuan pengguna, keamanan lebih mudah dijaga dengan membatasi akses yang tidak perlu ke informasi sensitif berdasarkan peran yang ditetapkan setiap pengguna dan organisasi [17].

Pada aplikasi yang akan dibangun, penerapan *role-based-access-control* dapat dilakukan dengan memberikan informasi mengenai data-data (contoh: *menu* dan *resource*) saat pengguna melakukan proses *login* pada sistem kepada aplikasi *client* yang mengirimkan *request*. Sehingga aplikasi *client* mengetahui hak akses pengguna dan dapat membatasinya dengan membaca informasi tersebut.

BAB III
ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas analisis dan perancangan perangkat lunak dari sistem *Single Sign On* myITS ini. Hasil dari proses ini berupa diagram yang akan digunakan sebagai acuan untuk proses implementasi perangkat lunak. Selain digunakan sebagai acuan untuk proses selanjutnya, beberapa diagram hasil dari proses perancangan digunakan sebagai dokumentasi dari implementasi perangkat lunak. Diagram yang dihasilkan pada proses ini disajikan dalam bentuk *Unified Modelling Language* (UML).

3.1 Analisis

Analisis meliputi analisis domain permasalahan, pendeskripsian perangkat lunak secara umum, penggambaran dan penjelasan kasus penggunaan dalam bentuk diagram kasus penggunaan, dan penggambaran dan penjelasan alur aktivitas tiap kasus penggunaan dalam bentuk diagram aktivitas.

3.1.1 Analisis permasalahan

Pada penggalian informasi di DPTSI (Direktorat Pengembangan Teknologi Sistem Informasi) ITS, ditemukan kebutuhan untuk mengakomodasi sistem otentikasi dan otorisasi terpusat (sistem *Single Sign On*), dimana pengguna hanya perlu memberikan kredensialnya satu kali saja untuk dapat masuk ke berbagai aplikasi dalam suatu jaringan. Sistem *Single Sign On* (SSO) yang diinginkan adalah yang menggunakan standar umum dan mampu menangani aplikasi internal (merupakan aplikasi yang ditangani langsung oleh DPTSI) dan aplikasi eksternal (aplikasi di luar jangkauan DPTSI), dimana sistem SSO yang digunakan sebelumnya belum menyediakan hal tersebut.

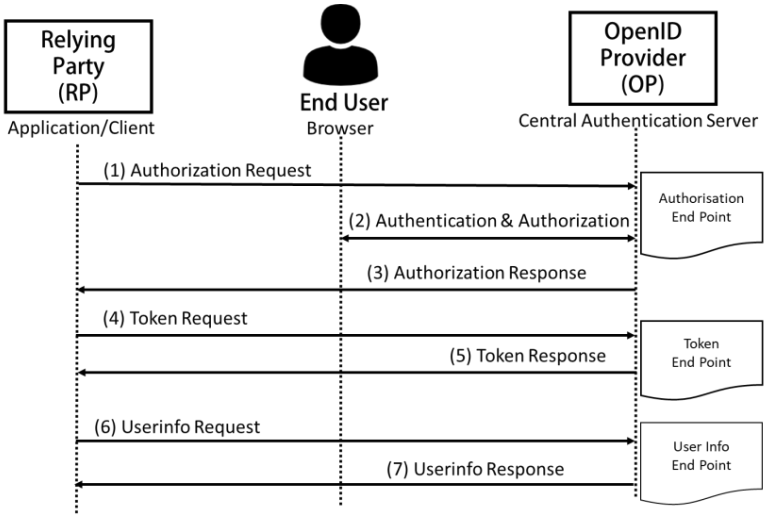
Selain itu, terdapat kebutuhan untuk menggabungkan sistem SSO dengan pengaturan hak akses berdasarkan peran atau *role based access control* sehingga aplikasi yang menggunakan sistem SSO dapat mengatur hak akses pengguna secara terpusat pada sistem dan mendapat informasi mengenai hak akses pengguna seperti *role* dan unit pengguna, serta *menu* dan *resource* apa saja yang dapat diakses pengguna yang sedang *login*. Aplikasi *client* juga membutuhkan informasi mengenai bahasa yang digunakan pengguna (*locale*) untuk menentukan bahasa yang akan ditampilkan pertama kali kepada pengguna. Hal lainnya yang diinginkan untuk dapat diintegrasikan dalam sistem adalah kebutuhan untuk mencatat *device token* pengguna pada tiap aplikasi (*mobile*) yang dimiliki ITS untuk keperluan *push notification*.

Pada sistem SSO yang dibangun, akan menggunakan standar OpenID Connect untuk otentikasi dan OAuth2 untuk otorisasi, serta penggabungan SSO dengan sistem RBAC. Aplikasi *client* dapat memilih untuk menggunakan sistem pengaturan hak akses terpusat pada sistem MyITS SSO maupun mengaturnya sendiri pada aplikasinya dengan menentukan *scope*. Informasi mengenai hak akses pengguna (untuk RBAC terpusat) dan informasi pengguna akan diberikan kepada aplikasi *client* saat pengguna telah terotentikasi pada sistem.

3.1.2 Deskripsi umum sistem

Dalam tugas akhir ini, akan dibuat sistem otentikasi Single-Sign-On (SSO) dan otorisasi *role-based-access-control* (RBAC) ITS menggunakan standar OpenID Connect. Cara ini dipilih karena sistem SSO ITS belum menggunakan suatu standar sehingga penggunannya terbatas dan tidak dapat digunakan untuk aplikasi luar ITS. Pada sistem SSO sebelumnya identitas pengguna disimpan dalam *cookie browser*. ID pengguna sebelumnya telah dienkripsi dengan penambahan salt khusus. RBAC diletakkan pada *database* sistem sehingga harus mengakses *database* secara

langsung untuk mendapatkannya. Gambar 3.1 adalah gambar mekanisme sistem yang akan dibuat.



Gambar 3.1 Mekanisme Sistem Otentikasi dan Otorisasi

Sistem akan menggunakan *authorisation code flow* untuk mendapatkan ID Token pengguna. Sistem yang akan dibuat adalah berupa *service* penyedia kredensial pengguna atau *Identity Provider* yang menggunakan standar OpenID Connect. Selanjutnya *Identity Provider* ini akan disebut OpenID Provider (OP). Klien atau aplikasi yang akan menggunakan fasilitas otentikasi dan otorisasi dari OP kemudian disebut *Relying Party* (RP).

RP menginisiasi otentikasi pengguna dengan mengalihkan browser pengguna ke *authorisation end point* pada OP. Kemudian pada OP pengguna akan di otentikasi dengan memeriksa apakah pengguna tersebut memiliki *valid session* (untuk mengetahui jika pengguna sebelumnya telah *login*), dan sebaliknya jika *session* tidak valid maka akan mendorong pengguna untuk *log in*. OP akan meminta persetujuan pengguna untuk *log in* pada RP. Berikut

adalah contoh tampilan proses *log in* dan permintaan persetujuan akses pada *authorization end-point*. Permintaan persetujuan hanya akan ditampilkan pada aplikasi (*client*) dari luar/ aplikasi eksternal. OP akan mengirimkan kode otorisasi kepada RP jika proses *log in* dan persetujuan berhasil.

Untuk mendapatkan ID Token, RP akan mengirimkan permintaan ID Token menggunakan *direct back-channel request* dengan menyertakan ID dan *secret* klien. ID dan *secret* klien didapatkan ketika klien pertama kali mendaftar untuk dapat menggunakan *service* OP. Kemudian OP akan mengirimkan *ID Token* serta *Access Token* yang dikodekan dalam JWT (JSON Web Token) ke RP saat ID dan *secret* klien sudah terdaftar. Proses pertukaran ID Token ini berada pada *token endpoint*.

Userinfo seperti data email, telepon, profil, alamat, dan informasi mengenai hak akses pengguna, seperti *roleunit, menu*, dan *resource* juga dapat diminta oleh RP dengan menggunakan *access token* tergantung pada *scope* yang diminta. Pertukaran *user info* terjadi pada *userinfo endpoint.* Selain *service* OP, juga akan dibuat sistem untuk manajemen data pengguna dan klien yang terdaftar pada OP yang nantinya disebut myITS *Security Management*.

Sistem otentikasi dan otorisasi dengan standar OpenID Connect yang dibangun menggunakan *Library OAuth 2.0 Bshaffer.* Untuk menggabungkan RBAC pada sistem dilakukan modifikasi pada nilai *scope* atau klaim. “Klaim lain di luar klaim standar MUNGKIN dapat digunakan bersama klaim standar. Saat menggunakan klaim tersebut, direkomendasikan untuk tidak sama persis dengan klaim standar” [18].

3.1.3 Spesifikasi Kebutuhan Perangkat Lunak

Sesuai dengan uraian mengenai cakupan perangkat lunak yang dibangun, dibutuhkan adanya spesifikasi perangkat lunak agar dapat memberikan solusi dari permasalahan yang diberikan dan dapat bekerja dengan baik dalam mengakomodasi kebutuhan.

Diharapkan dengan adanya spesifikasi ini dapat menyesuaikan kebutuhan-kebutuhan pengguna. Spesifikasi kebutuhan perangkat lunak adalah penjelasan mengenai kebutuhan sistem yang diinginkan pelanggan atau klien dalam bentuk tulisan. Spesifikasi kebutuhan perangkat lunak pada tugas akhir ini terdiri dari kebutuhan fungsional yang dapat dilihat pada Tabel 3.1.

3.1.3.1 Kebutuhan Fungsional

Tabel 3.1 menjelaskan kebutuhan fungsional yang didapat dari penggalian kebutuhan.

Tabel 3.1 Kebutuhan fungsional perangkat lunak

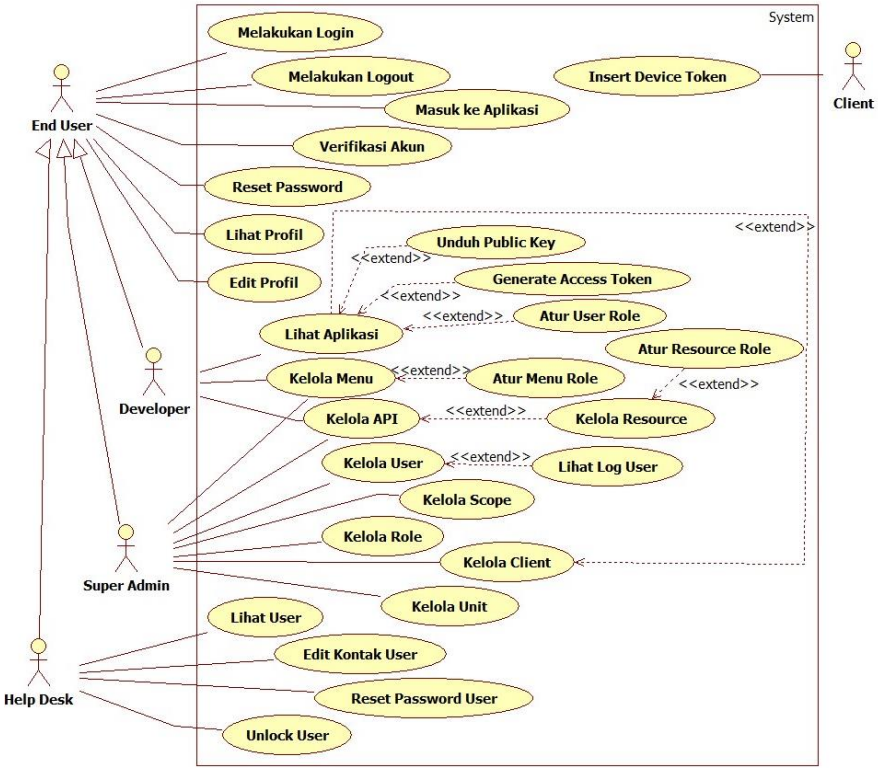
No.	Kebutuhan Fungsional	Deskripsi
1.	Melakukan *login* ke dalam sistem	Melakukan proses otentikasi menggunakan akun *user*
2.	Masuk ke aplikasi pada dashboard	Masuk ke aplikasi yang dipilih melalui dashboard myITS
3.	Melihat profil	Melihat detail profil pribadi
4.	Memverifikasi akun	Melakukan verifikasi akun email, email alternatif, maupun no. ponsel pada akun pribadi
5.	Mereset *password* pribadi	Melakukan pengaturan ulang *password* pribadi
6.	Memperbaharui profil pribadi	Melakukan pembaharuan terhadap beberapa data pribadi
7.	Melakukan *Logout*	Keluar dari sistem
8.	Mengelola *client*	Melakukan penambahan atau pendaftaran *client* baru dan pembaharuan *client*

9.	Mengelola *user*	Melakukan penambahan atau pendaftaran *user* baru dan pembaharuan user
10.	Mengelola *role*	Melakukan penambahan *role*, pembaharuan *role*, dan penghapusan *role*
11.	Mengelola unit	Melakukan penambahan unit, pembaharuan unit, dan penghapusan unit
12.	Mengelola *scope*	Melakukan penambahan *scope*, pembaharuan *scope*, dan penghapusan *scope*
13.	Mengelola menu	Melakukan penambahan menu, pembaharuan menu, pengapusan menu pada tiap aplikasi (*client*)
14.	Mengelola API	Melakukan penambahan API, pembaharuan API, dan penghapusan API pada tiap aplikasi (*client*)
15.	Mengelola *resource*	Melakukan penambahan *resource*, pembaharuan *resource*, dan penghapusan *resource* pada tiap API
16.	Mengatur *resource role*	Melakukan penambahan *resource role* dan penghapusan *resource role* pada tiap *resource*
17.	Mengatur *user role*	Melakukan penambahan *user role*, pembaharuan *user role*, penghapusan *user role* pada tiap aplikasi (*client*)
18.	Mengatur *menu role*	Melakukan penambahan *menu role*, pembaharuan

		menu role, penghapusan *menu role* pada tiap menu
19.	Melihat log *user*	Melihat log aktivitas tiap *user*
20.	Melihat daftar aplikasi	Melihat aplikasi yang ditangani pengguna (pada aplikasi dimana *user* menjadi PIC) dan informasi detail mengenai aplikasi
21.	Mengunduh *public key*	Mendapatkan file berisi *public key* pada tiap aplikasi (*client*) dengan cara mengunduh
22.	Membuat *access token*	Mendapatkan *access token* yang telah di-*generate* sistem untuk tiap aplikasi (*client*)
23.	Memperbaharui kontak *user*	Melakukan pembaharuan email, email alternatif atau no. ponsel *user*
24.	Membuka *user* yang terkunci	Membuka akun *user* yang terkunci sehingga *user* dapat melakukan *login* kembali
25.	Mereset *password user*	Melakukan pengaturan ulang *password* user
26.	Melihat *user*	Melihat seluruh daftar *user* yang terdaftar dalam sistem dan melihat detail informasi *user*
27.	Menyiman *device token* pengguna	Menyimpan *device token* pengguna pada tiap aplikasi yang terdaftar dalam sistem.

3.1.3.2 Kasus penggunaan

Bagian ini menjelaskan analisa dan perancangan kasus-kasus penggunaan yang terdapat pada fitur perangkat lunak. Kasus penggunaan tersebut ditunjukkan pada Gambar 3.2 dan tiap kasus penggunaan akan disertakan spesifikasi dan diagram aktivitasnya.



Gambar 3.2 Diagram kasus penggunaan

3.1.3.3 Aktor

Aktor adalah entitas luar yang terlibat langsung dalam penggunaan perangkat lunak. Pada sistem ini aktor adalah *super admin* (*administrator*), *developer*, *helpdesk*, *end-user* dan *client*.

Aktor *super admin* merupakan pengguna yang memiliki seluruh hak akses untuk mengelola aplikasi. Aktor *help*desk yaitu pengguna pada *service desk* yang memiliki akses untuk melihat user dan memperbaharui beberapa data pengguna. *Developer* adalah pengembang aplikasi yang mengelola aplikasi *client* yang terdaftar dalam sistem. Aktor *end-user* adalah seluruh pengguna yang terdaftar dalam sistem dan merupakan generalisasi dari aktor lainnya. Aktor *client* adalah aplikasi *mobile* yang menggunakan fitur *login* MyITS SSO.

3.1.3.4 Melakukan *Login*

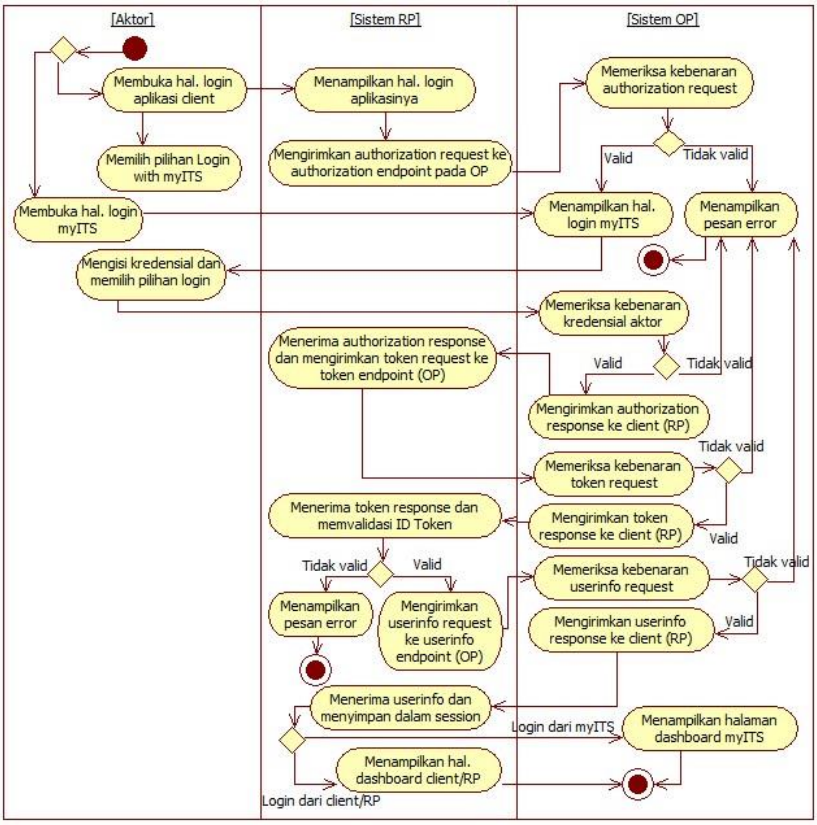
Pengguna atau *end-user* dapat masuk ke dalam sistem dengan melakukan proses otentikasi (*login*) dengan memasukkan kredensial berupa *username* dan *password* yang telah tersimpan dalam sistem. *End-user* dapat melakukan *login* dengan dua cara, yaitu dengan langsung *login* pada aplikasi MyITS SSO, atau dengan *login* melalui aplikasi *client* (*Relying Party*) yang menggunakan fitur *login* with myITS. Tabel 3.2 menunjukan spesifikasi kasus penggunaan *login* dan Gambar 3.3 menunjukan diagram aktivitasnya.

Tabel 3.2 Spesifikasi kasus penggunaan *login*

Komponen	Deskripsi
Nama	Melakukan *Login*
Nomor	UC-001
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat masuk ke sistem dengan melakukan proses otentikasi atau *login*
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor belum terotentikasi atau *login* ke dalam sistem
Kondisi Akhir	Aktor berhasil terotentikasi dan sistem menampilkan halaman dashboard myITS

Alur Normal	1. Aktor membuka halaman awal/ halaman *login* sistem (langsung pada myITS) 2. Sistem (OP) menampilkan halaman *login* myITS 3. Aktor mengisi kredensial (*username* dan *password*) dan memilih pilihan *login* 4. Sistem (OP) memeriksa kebenaran kredensial pengguna 5. Jika valid, sistem (OP) mengirimkan *authorization response* ke *client* (RP) 6. Sistem (*client*/RP) menerima *authorization response* dan mengirimkan *token request* ke *token endpoint* (OP) 7. Sistem (OP) memeriksa kebenaran *token request* 8. Jika valid, sistem (OP) mengirimkan *token response* ke *client* (RP) 9. Sistem (*client*/RP) menerima *token response* dan memvalidasi ID Token 10. Jika valid, sistem (*client*/RP) mengirimkan *userinfo request* ke *userinfo endpoint* (OP) 11. Sistem (OP) memeriksa kebenaran *userinfo request* 12. Jika valid, sistem (OP) mengirimkan *userinfo reponse* ke *client* (RP) 13. Sistem (RP) menerima *userinfo* dan menyimpan dalam *session* 14. Aktor berhasil terotentikasi dan sistem menampilkan halaman dashboard myITS
Alur Alternatif	1.a Aktor masuk melalui aplikasi *client* yang terdaftar dalam sistem OP 1.a.1 Aktor membuka halaman awal/ halaman *login* aplikasi *client* 1.a.2 Sistem (*client*) menampilkan halaman *login* aplikasinya

	1.a.3 Aktor memilih pilihan *Login with myITS* 1.a.4 Sistem (*client*) mengirimkan *authorization request* ke *authorization endpoint* pada OP 1.a.5 Sistem (OP) memeriksa kebenaran *authorization request* 14.a Aktor masuk melalui aplikasi *client* yang terdaftar dalam sistem OP 14.a.1 Aktor berhasil terotentikasi dan sistem menampilkan halaman dashboard aplikasi *client*/RP
Eksepsi	1.a.5.a *Authorization request* yang dikirimkan tidak valid 1.a.5.a.1 Sistem OP mengirimkan pesan error 5.a Kredensial yang dimasukkan tidak valid 5.a.1 Sistem OP mengirimkan pesan error 8.a *Token request* yang dikirimkan tidak valid 8.a.1 Sistem OP mengirimkan pesan error 10.a ID Token yang diterima tidak valid 10.a.1 Sistem RP menampilkan pesan error 12.a *Userinfo request* yang dikirimkan tidak valid 12.a.1 Sistem OP mengirimkan pesan error



Gambar 3.3 Diagram aktivitas *login*

3.1.3.5 Masuk ke Aplikasi

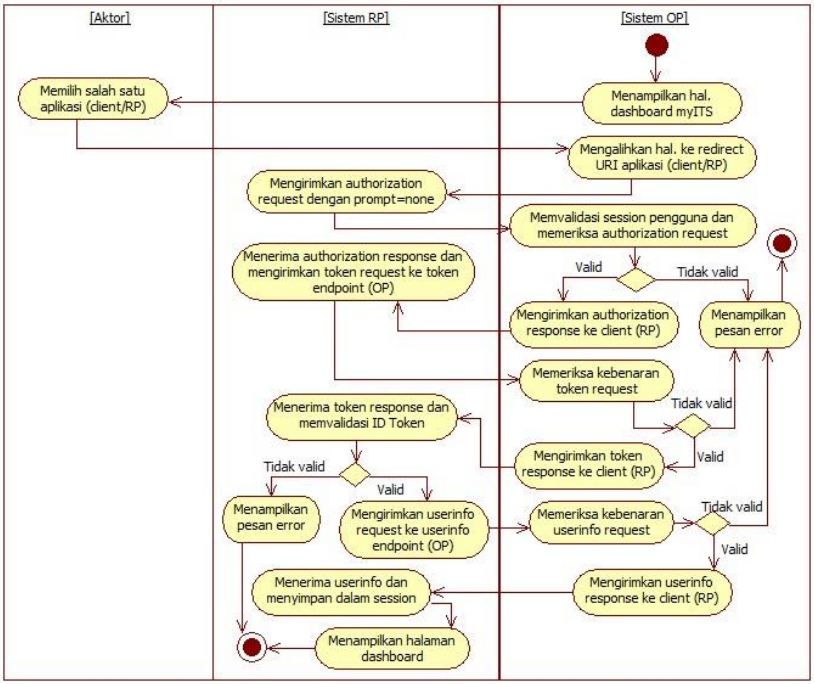
Setelah e*nd-user* terotentikasi, sistem menampilkan halaman *dashboard* myITS berisi daftar aplikasi yang dapat diakses pengguna. Pengguna dapat masuk dan mengakses aplikasi tersebut tanpa harus melakukan *login* kembali dengan memilih salah satu aplikasi. Tabel 3.3 menunjukan spesifikasi kasus

penggunaan masuk ke aplikasi dan Gambar 3.4 menunjukan diagram aktivitasnya.

Tabel 3.3 Spesifikasi kasus penggunaan masuk ke aplikasi

Komponen	Deskripsi
Nama	Masuk ke Aplikasi
Nomor	UC-002
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat masuk dan mengakses aplikasi yang tampil pada *dashboard* myITS tanpa perlu melakukan *login* kembali
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor telah terotentikasi
Kondisi Akhir	Sistem (RP) menampilkan halaman dashboard aplikasi yang dipilih
Alur Normal	1. Sistem (OP) menampilkan halaman *dashboard* myITS berisi daftar aplikasi (*client*/RP) yang dapat diakses aktor 2. Aktor memilih salah satu aplikasi (*client*/RP) 3. Sistem (OP) mengalihkan halaman ke *redirect* URI aplikasi (*client*/RP) 4. Sistem (*client*/RP) mengirimkan *authorization request* dengan prompt=none 5. Sistem (OP) memvalidasi *session* pengguna dan memeriksa *authorization request* 6. Jika valid, sistem (OP) mengirimkan *authorization response* 7. Sistem (*client*/RP) menerima *authorization response* dan mengirimkan *token request* 8. Sistem (OP) memeriksa kebenaran *token request* 9. Jika valid, sistem (OP) akan mengirimkan *token response*

	10. Sistem (*client*/RP) menerima *token response* dan memvalidasi ID Token 11. Jika valid, sistem (*client*/RP) akan mengirimkan *userinfo request* 12. Sistem (OP) memvalidasi *userinfo request* 13. Jika valid, sistem (OP) akan mengirimkan *userinfo response* 14. Sistem (*client*/RP) menerima dan menyimpan *userinfo* pada *session* 15. Sistem (*client*/RP) menampilkan halaman dashboard
Alur Alternatif	-
Eksepsi	6.a *Authorization request* tidak valid 6.a.1 Sistem OP mengirimkan pesan error 9.a *Token request* yang dikirimkan tidak valid 9.a.1 Sistem OP mengirimkan pesan error 11.a *ID Token* yang diterima tidak valid 11.a.1 Sistem RP menampilkan pesan error 13.a *Userinfo request* yang dikirimkan tidak valid 13.a.1 Sistem OP mengirimkan pesan error



Gambar 3.4 Diagram aktivitas masuk ke aplikasi

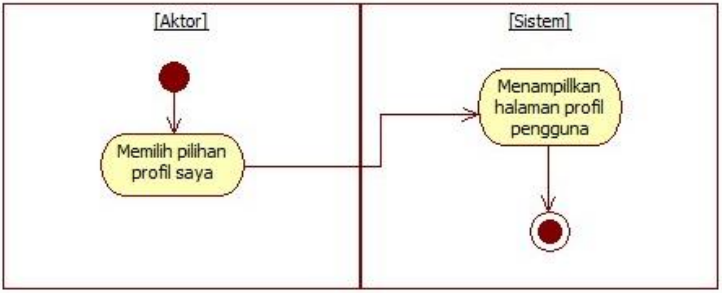
3.1.3.6 Lihat Profil

Dalam sistem myITS, *end-user* dapat melihat profil dirinya yang berisi data pribadi *end-user* seperti nama, nama panggilan, tanggal lahir, foto profil, dan data lainnya. Tabel 3.4 menunjukan spesifikasi kasus penggunaan lihat profil dan Gambar 3.5 menunjukan diagram aktivitasnya.

Tabel 3.4 Spesifikasi kasus penggunaan lihat profil

Komponen	Deskripsi
Nama	Lihat Profil
Nomor	UC-003

Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat melihat data pribadinya pada bagian profil
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor telah terotentikasi
Kondisi Akhir	Sistem menampilkan halaman profil pengguna
Alur Normal	1. Aktor memilih pilihan profil saya 2. Sistem menampilkan halaman profil pengguna
Alur Alternatif	-
Eksepsi	-



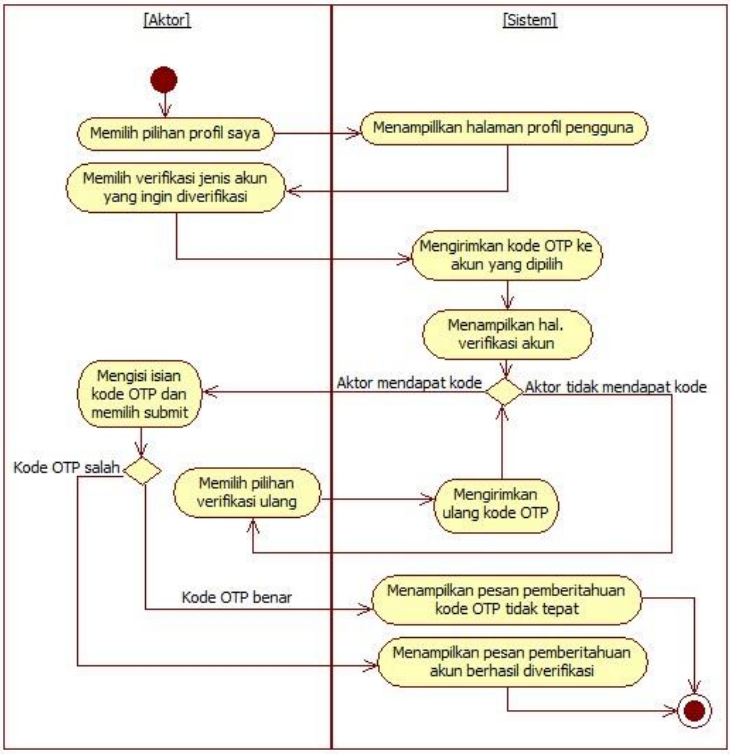
Gambar 3.5 Diagram aktivitas lihat profil

3.1.3.7 Verifikasi Akun

End-user dapat melakukan verifikasi terhadap akunnya dalam sistem. Adapun akun pengguna yang tersimpan dalam sistem adalah email, email alternatif, dan nomor ponsel. Untuk dapat melakukan verifikasi, sistem akan mengirimkan kode OTP ke jenis akun yang dipilih pengguna. Tabel 3.5 menunjukan spesifikasi kasus penggunaan verifikasi akun dan Gambar 3.6 menunjukan diagram aktivitasnya.

Tabel 3.5 Spesifikasi kasus penggunaan verifikasi akun

Komponen	Deskripsi
Nama	Verifikasi Akun
Nomor	UC-004
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat melakukan verifikasi akun sesuai jenis akun yang dipilih
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor telah terotentikasi dan akun yang dipilih belum terverifikasi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan bahwa akun telah terverifikasi
Alur Normal	1. Aktor memilih pilihan profil saya 2. Sistem menampilkan halaman profil pengguna 3. Aktor memilih verifikasi jenis akun yang ingin diverifikasi 4. Sistem mengirimkan kode OTP ke akun yang dipilih pengguna 5. Sistem menampilkan halaman verifikasi akun 6. Aktor mengisi isian kode OTP dan memilih submit 7. Sistem menampilkan pesan pemberitahuan bahwa akun telah terverifikasi
Alur Alternatif	6.a Aktor tidak mendapatkan kode OTP 6.a.1 Aktor memilih pilihan verifikasi ulang 6.a.2 Sistem mengirimkan ulang kode OTP 6.a.3 Aktor mengisi isian kode OTP dan memilih submit
Eksepsi	8.a Kode OTP yang dimasukkan aktor salah 8.a.1 Sistem menampilkan pesan pemberitahuan kode OTP tidak tepat



Gambar 3.6 Diagram aktivitas verifikasi akun

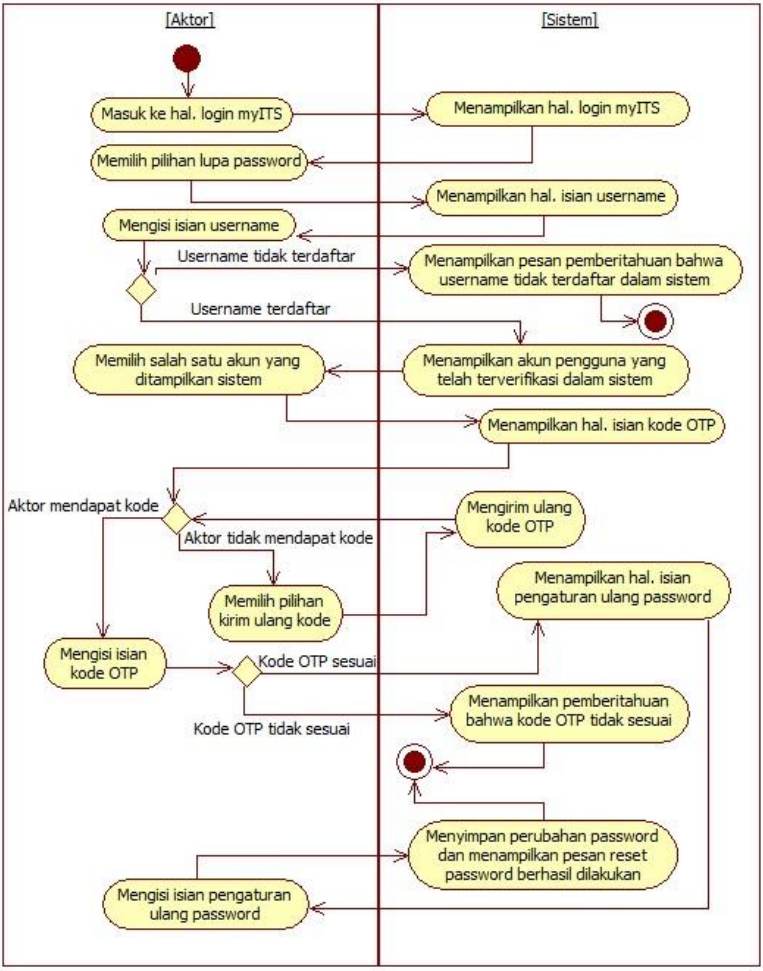
3.1.3.8 Reset *Password*

Jika *end-user* lupa *password*, *end-user* dapat melakukan pengaturan ulang pada *password*nya. Untuk melakukan pengaturan ulang *password*, dapat menggunakan akun email maupun no.ponsel yang telah terverifikasi. Jika salah kedua akun belum terverifikasi, pengguna tidak dapat mengatur ulang *password*nya. Tabel 3.6 menunjukan spesifikasi kasus penggunaan dan Gambar 3.7 menunjukan diagram aktivitasnya.

Tabel 3.6 Spesifikasi kasus penggunaan reset *password*

Komponen	Deskripsi
Nama	Reset *Password*
Nomor	UC-005
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat mengatur ulang *password*nya.
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor belum terotentikasi dan salah satu akun (email atau no. ponsel) telah terverifikasi
Kondisi Akhir	Sistem menyimpan perubahan *password* dan menampilkan pesan reset *password* berhasil dilakukan
Alur Normal	1. Aktor masuk ke halaman *login* myITS 2. Sistem menampilkan halaman *login* myITS 3. Aktor memilih pilihan lupa *password* 4. Sistem menampilkan halaman isian *username* 5. Aktor mengisi isian *username* 6. Sistem menampilkan akun pengguna yang telah terverifikasi dalam sistem 7. Aktor memilih salah satu akun yang ditampilkan sistem 8. Sistem menampilkan halaman isian kode OTP 9. Aktor mengisi isian kode OTP 10. Sistem menampilkan halaman isian untuk pengaturan ulang *password* 11. Aktor mengisi isian pengaturan ulang *password* 12. Sistem menyimpan perubahan *password* dan menampilkan pesan reset *password* berhasil dilakukan
Alur Alternatif	9.a Aktor tidak mendapatkan kode OTP 9.a.1 Aktor memilih pilihan kirim ulang kode

	9.a.2 Sistem mengirimkan ulang kode OTP 9.a.3 Aktor mengisi isian kode OTP
Eksepsi	6.a *Username* yang dimasukkan pengguna tidak terdaftar dalam sistem 6.a.1 Sistem menampilkan pesan pemberitahuan bahwa *username* tidak terdaftar dalam sistem 10.a Kode OTP yang dimasukkan tidak sesuai 10.a.1 Sistem menampilkan pemberitahuan bahwa kode OTP tidak sesuai



Gambar 3.7 Diagram aktivitas reset *password*

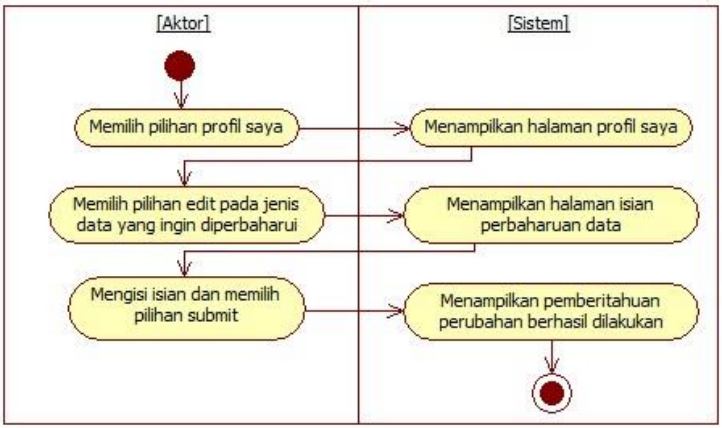
3.1.3.9 Edit Profil

Selain dapat melihat informasi profilnya, *end-user* juga dapat melakukan pembaharuan terhadap beberapa data profilnya

seperti, email alternatif, nama panggilan (*nickname*), preferensi bahasa (*locale*), informasi lokasi (*zoneinfo*) dan foto profil. Tabel 3.7 menunjukan spesifikasi kasus penggunaan dan Gambar 3.8 menunjukan diagra aktivitasnya.

Tabel 3.7 Spesifikasi kasus penggunaan edit profil

Komponen	Deskripsi
Nama	Edit Profil
Nomor	UC-006
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat memperbaharui informasi profilnya, seperti data email alternatif, nama panggilan (*nickname)*, preferensi bahasa (*locale)*, informasi lokasi (*zoneinfo*), dan foto profil.
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor telah terotentikasi
Kondisi Akhir	Sistem menampilkan pemberitahuan perubahan berhasil dilakukan
Alur Normal	1. Aktor memilih pilihan profil saya 2. Sistem menampilkan halaman profil saya 3. Aktor memilih pilihan edit pada jenis data yang ingin diperbaharui 4. Sistem menampilkan halaman isian perbaharuan data 5. Aktor mengisi isian dan memilih pilihan submit 6. Sistem menampilkan pemberitahuan perubahan berhasil dilakukan
Alur Alternatif	-
Eksepsi	-



Gambar 3.8 Diagram aktivitas edit profil

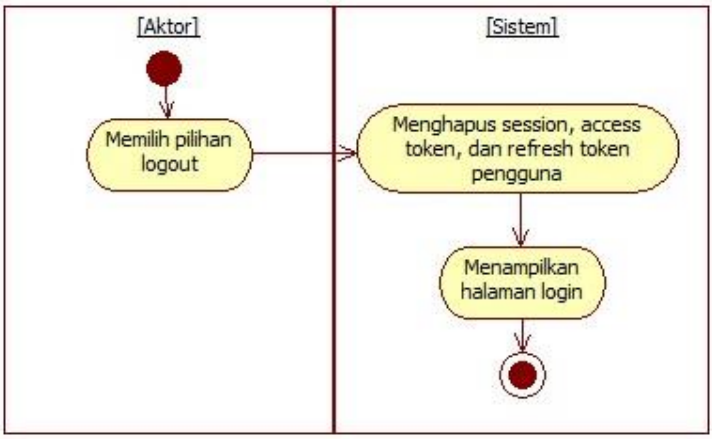
3.1.3.10 Melakukan *Logout*

Untuk keluar dari aplikasi MyITS SSO, pengguna dapat melakukan proses *logout*. Tabel 3.8 menunjukan spesifikasi kasus penggunaan *logout* dan Gambar 3.9 menunjukan diagram aktivitasnya.

Tabel 3.8 Spesifikasi kasus penggunaan *logout*

Komponen	Deskripsi
Nama	Melakukan *Logout*
Nomor	UC-007
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat keluar dari aplikasi dengan melakukan proses *logout*.
Tipe	Fungsional
Aktor	End-user
Kondisi Awal	Aktor telah terotentikasi
Kondisi Akhir	Sistem menampilkan halaman *login*

Alur Normal	1. Aktor memilih pilihan *logout* 2. Sistem menghapus *session* pengguna, *access token* dan *refresh token* pengguna 3. Sistem menampilkan halaman *login*
Alur Alternatif	-
Eksepsi	-



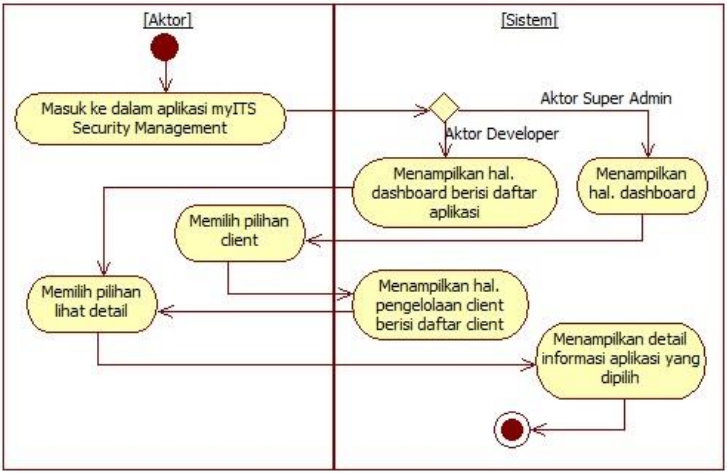
Gambar 3.9 Diagram aktivitas *logout*

3.1.3.11 Lihat Aplikasi

Sebagai pengembang aplikasi, *developer* dapat melihat aplikasi yang dikembangkannya yang telah terdaftar dalam sistem MyITS SSO. *Developer* dapat melihat detail informasi mengenai aplikasinya seperti ID client, client secret, redirect URI, scope, dan lainnya. Tabel 3.9 menunjukan spesifikasi kasus penggunaan dan Gambar 3.10 menunjukan diagram aktivitasnya.

Tabel 3.9 Spesifikasi kasus penggunaan lihat aplikasi

Komponen	Deskripsi
Nama	Lihat aplikasi
Nomor	UC-008
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat melihat daftar aplikasi yang dikembangkan dan melihat detail informasi aplikasi.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin
Kondisi Akhir	Sistem menampilkan daftar dan detail informasi aplikasi yang dikembangkan
Alur Normal	1. Aktor masuk ke dalam aplikasi myITS Security Management 2. Sistem menampilkan halaman dashboard berisi daftar aplikasi (peran Developer) 3. Aktor memilih pilihan lihat detail 4. Sistem menampilkan detail informasi aplikasi yang dipilih
Alur Alternatif	2.a.1 Aktor dengan peran Super Admin 2.a.1 Sistem menampilkan halaman dashboard 2.a.2 Aktor memilih pilihan *client* 2.a.3 Sistem menampilkan halaman pengelolaan *client* berisi daftar *client*
Eksepsi	-



Gambar 3.10 Diagram aktivitas lihat aplikasi

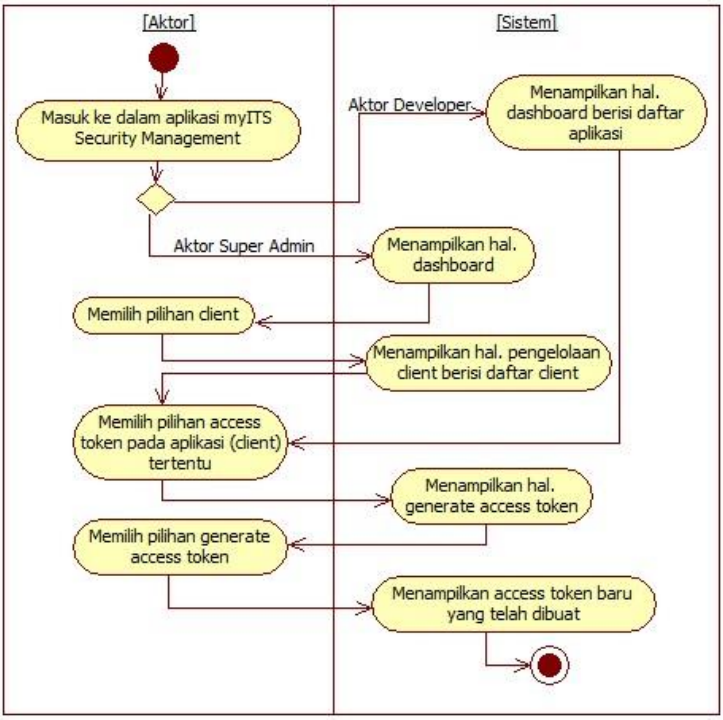
3.1.3.12 Generate Access Token

Selain dapat melihat detail informasi mengenai aplikasi, aktor juga dapat membuat sebuah *access token* untuk dapat digunakan. Tabel 3.10 menunjukan spesifikasi kasus penggunaan *generate access token* dan Gambar 3.11 menunjukan diagram aktivitasnya.

Tabel 3.10 Spesifikasi kasus penggunaan generate access

Komponen	Deskripsi
Nama	Generate a*ccess token*
Nomor	UC-009
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat membuat access token.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin

Kondisi Akhir	Sistem menampilkan *access token* yang telah dibuat.
Alur Normal	1. Aktor masuk ke dalam aplikasi myITS Security Management 2. Sistem menampilkan halaman dashboard berisi daftar aplikasi (peran Developer) 3. Aktor memilih pilihan access token pada aplikasi (*client*) tertentu 4. Sistem menampilkan halaman *generate access token* 5. Aktor memilih pilihan *generate access token* 6. Sistem menampilkan *access token* baru yang telah dibuat
Alur Alternatif	2.a Aktor dengan peran Super Admin 2.a.1 Sistem menampilkan halaman dashboard 2.a.2 Aktor memilih pilihan *client* 2.a.3 Sistem menampilkan halaman pengelolaan *client* berisi daftar *client*
Eksepsi	-



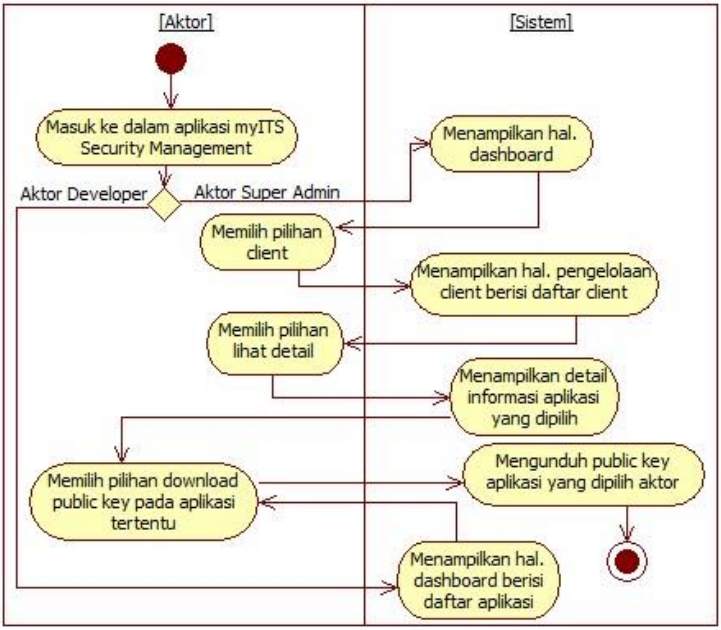
Gambar 3.11 Diagram aktivitas generate access token

3.1.3.13 Unduh Public Key

Aktor dapat mengunduh *public key* aplikasinya yang telah terdaftar dalam sistem. *Public key* digunakan untuk memvalidasi ID Token yang didapat dari sistem. Tabel 3.11 menunjukan spesifikasi kasus penggunaan unduh *public key* dan Gambar 3.12 menunjukan diagram aktivitasnya.

Tabel 3.11 Spesifikasi kasus penggunaan unduh public key

Komponen	Deskripsi
Nama	Unduh *public key*
Nomor	UC-010
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat mengunduh *public key* aplikasi yang dipilih.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer
Kondisi Akhir	Sistem mengunduh *public key* aplikasi yang dipilih aktor.
Alur Normal	1. Aktor masuk ke dalam aplikasi myITS Security Management 2. Sistem menampilkan halaman dashboard berisi daftar aplikasi (peran Developer) 3. Aktor memilih pilihan *download public key* pada aplikasi tertentu 4. Sistem mengunduh *public key* aplikasi yang dipilih aktor.
Alur Alternatif	2.a Aktor dengan peran Super Admin 2.a.1 Sistem menampilkan halaman dashboard 2.a.2 Aktor memilih pilihan *client* 2.a.3 Sistem menampilkan halaman pengelolaan *client* berisi daftar *client* 2.a.4 Aktor memilih pilihan lihat detail *client* 2.a.5 Sistem menampilkan halaman detail *client*
Eksepsi	-



Gambar 3.12 Diagram aktivitas download public key

3.1.3.14 Atur User Role

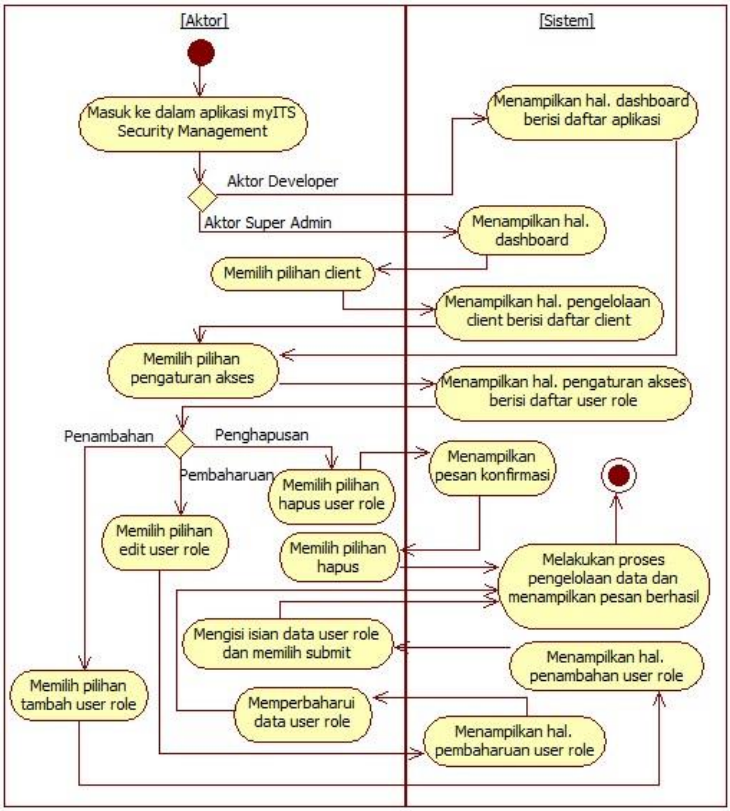
Aktor dapat mengatur peran pengguna pada aplikasi yang ditampilkan sistem. Aktor dapat menambah, memperbaharui, dan menghapus peran pengguna pada suatu aplikasi. Tabel 3.12 menunjukan spesifikasi kasus penggunaan atur user role dan Gambar 3.13 menunjukan diagram aktivitasnya.

Tabel 3.12 Spesifikasi kasus penggunaan atur user role

Komponen	Deskripsi
Nama	Atur user role
Nomor	UC-011

Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus peran pengguna pada suatu aplikasi.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor masuk ke dalam aplikasi myITS Security Management 2. Sistem menampilkan halaman dashboard berisi daftar aplikasi (peran Developer) 3. Aktor memilih pilihan pengaturan akses 4. Sistem menampilkan halaman pengaturan akses berisi daftar *user role* 5. Aktor memilih salah satu pilihan pengelolaan data 6. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	2.a Aktor dengan peran Super Admin 2.a.1 Sistem menampilkan halaman dashboard 2.a.2 Aktor memilih pilihan *client* 2.a.3 Sistem menampilkan halaman pengelolaan *client* berisi daftar *client* 5.a Penambahan *user role* 5.a.1 Aktor memilih pilihan tambah *user role* 5.a.2 Sistem menampilkan halaman penambahan *user role* 5.a.3 Aktor mengisi isian data *user role* dan memilih submit 5.b Pembaharuan *user role* 5.b.1 Aktor memilih pilihan edit *user role*

	5.b.2 Sistem menampilkan halaman pembaharuan *user role* 5.b.3 Aktor memperbaharui data *user role* 5.c Penghapusan *user role* 5.c.1 Aktor memilih pilihan hapus *user role* 5.c.2 Sistem menampilkan pesan konfirmasi 5.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.13 Diagram aktivitas atur user role

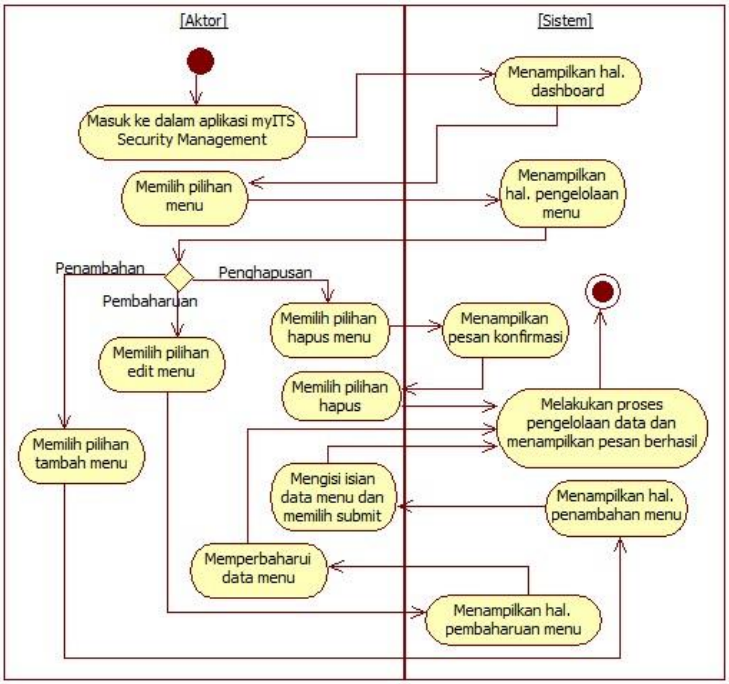
3.1.3.15 Kelola Menu

Aktor dapat mengelola menu aplikasi yang terdaftar dalam sistem. Aktor dapat menambah, memperbaharui, dan menghapus peran pengguna pada suatu aplikasi. Tabel 3.13 menunjukan spesifikasi kasus penggunaan kelola menu dan Gambar 3.14 menunjukan diagram aktivitasnya.

Tabel 3.13 Spesifikasi kasus penggunaan kelola menu

Komponen	Deskripsi
Nama	Kelola menu
Nomor	UC-012
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data menu pada suatu aplikasi.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor masuk ke dalam aplikasi myITS Security Management 2. Sistem menampilkan halaman dashboard 3. Aktor memilih pilihan menu 4. Sistem menampilkan halaman pengelolaan menu 5. Aktor memilih salah satu pilihan pengelolaan data 6. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	5.a Penambahan *menu* 5.a.1 Aktor memilih pilihan tambah *menu* 5.a.2 Sistem menampilkan halaman penambahan menu

	5.a.3 Aktor mengisi isian data *menu* dan memilih submit 5.b Pembaharuan *menu* 5.b.1 Aktor memilih pilihan edit menu 5.b.2 Sistem menampilkan halaman pembaharuan menu 5.b.3 Aktor memperbaharui data menu 5.c Penghapusan menu 5.c.1 Aktor memilih pilihan hapus menu 5.c.2 Sistem menampilkan pesan konfirmasi 5.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.14 Diagram aktivitas kelola menu

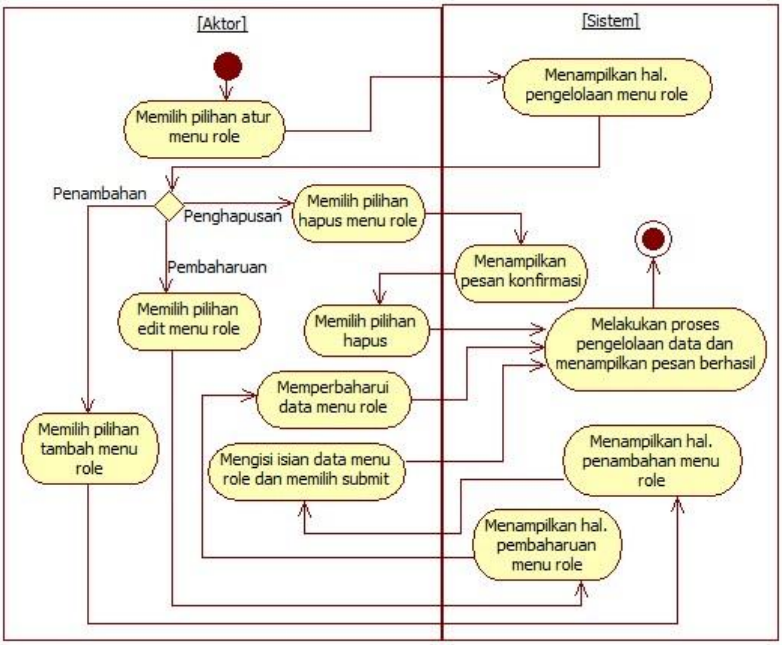
3.1.3.16 Atur Menu Role

Aktor dapat mengatur menu role pada menu yang tersimpan dalam sistem. Aktor dapat menambah, memperbaharui, dan menghapus menu role pada suatu menu. Tabel 3.14 menunjukan spesifikasi kasus penggunaan atur menu role Gambar 3.15 menunjukan diagram aktivitasnya.

Tabel 3.14 Spesifikasi kasus penggunaan atur menu role

Komponen	Deskripsi
Nama	Atur Menu Role
Nomor	UC-013
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data menu role pada suatu menu.
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin dan berada pada halaman pengelolaan menu
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan atur menu role pada salah satu menu 2. Sistem manampilkan halaman pengelolaan menu 3. Aktor memilih salah satu pilihan pengelolaan data 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan *menu role* 3.a.1 Aktor memilih pilihan tambah *menu role* 3.a.2 Sistem menampilkan halaman penambahan menu role

	3.a.3 Aktor mengisi isian data *menu role* dan memilih submit 3.b Pembaharuan *menu role* 3.b.1 Aktor memilih pilihan edit menu role 3.b.2 Sistem menampilkan halaman pembaharuan menu role 3.b.3 Aktor memperbaharui data menu role 3.c Penghapusan menu role 3.c.1 Aktor memilih pilihan hapus menu role 3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Ekspesi	-



Gambar 3.15 Diagram aktivitas kelola menu role

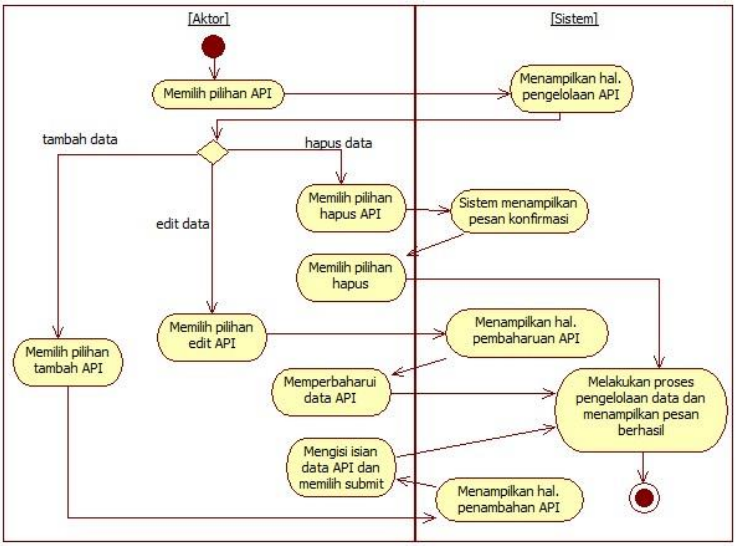
3.1.3.17 Kelola API

Aktor dapat mengelola API aplikasi yang terdaftar dalam sistem. Aktor dapat menambah, memperbaharui, dan menghapus API pada suatu aplikasi. Tabel 3.15 menunjukan spesifikasi kasus penggunaan kelola API dan Gambar 3.16 menunjukan diagram aktivitasnya.

Tabel 3.15 Spesifikasi kasus penggunaan kelola API

Komponen	Deskripsi
Nama	Kelola API
Nomor	UC-014
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data API pada suatu aplikasi
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan API 2. Sistem menampilkan halaman pengelolaan API 3. Aktor memilih salah satu pilihan pengelolaan API 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan API 3.a.1 Aktor memilih pilihan tambah API 3.a.2 Sistem menampilkan halaman penambahan API 3.a.3 Aktor mengisi isian data API dan memilih submit

	3.b Pembaharuan API 3.b.1 Aktor memilih pilihan edit API 3.b.2 Sistem menampilkan halaman pembaharuan API 3.b.3 Aktor memperbaharui data API 3.c Penghapusan API 3.c.1 Aktor memilih pilihan hapus API 3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.16 Diagram aktivitas kelola API

3.1.3.18 Kelola Resource

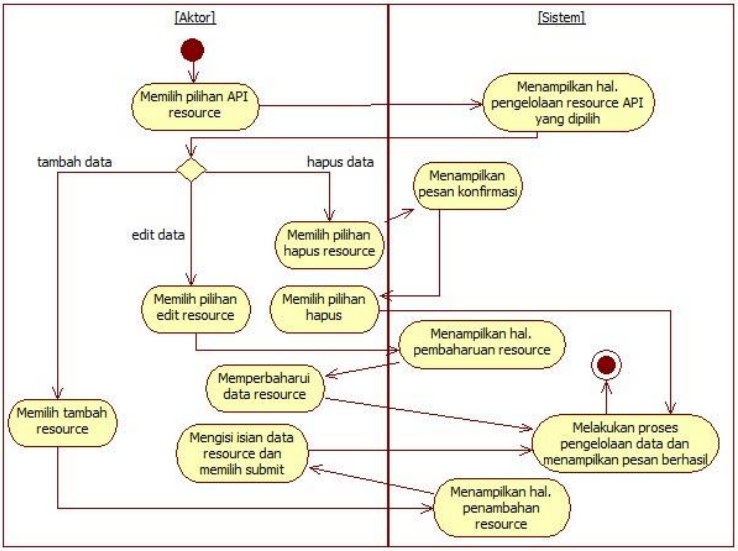
Aktor dapat mengatur resource pada API yang terdaftar dalam sistem. Aktor dapat menambah, memperbaharui, dan menghapus resource pada suatu API. Tabel 3.16 menunjukan

spesifikasi kasus penggunaan kelola resource dan Gambar 3.17 menunjukan diagram aktivitasnya.

Tabel 3.16 Spesifikasi kasus penggunaan kelola resource

Komponen	Deskripsi
Nama	Kelola Resource
Nomor	UC-015
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data resource pada suatu API
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin dan berada pada halaman pengelolaan API
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan API resource 2. Sistem menampilkan halaman pengelolaan resource API yang dipilih 3. Aktor memilih salah satu pilihan pengelolaan resource 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan resource 3.a.1 Aktor memilih pilihan tambah resource 3.a.2 Sistem menampilkan halaman penambahan resource 3.a.3 Aktor mengisi isian data resource dan memilih submit 3.b Pembaharuan resource 3.b.1 Aktor memilih pilihan edit resource 3.b.2 Sistem menampilkan halaman pembaharuan resource 3.b.3 Aktor memperbaharui data resource

	3.c Penghapusan resource 3.c.1 Aktor memilih pilihan hapus resource 3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Eksepsi	-



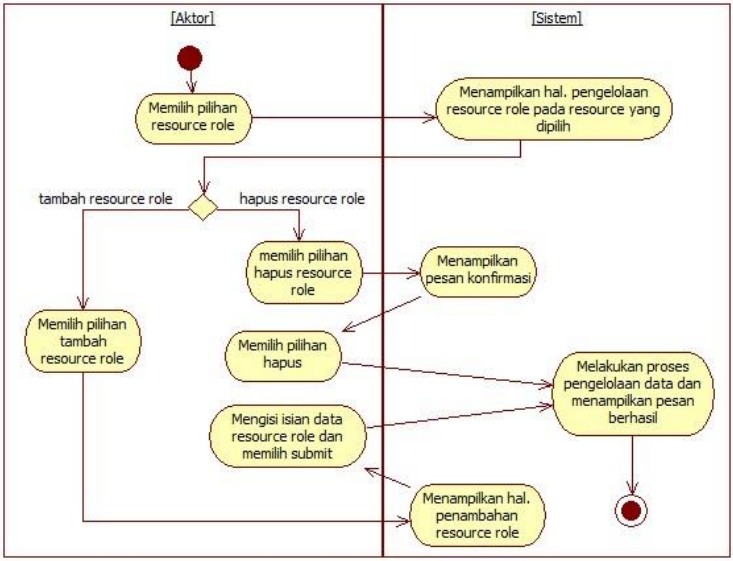
Gambar 3.17 Diagram aktivitas kelola resource

3.1.3.19 Atur Resource Role

Aktor dapat mengatur resource role pada resource yang tersimpan dalam sistem. Aktor dapat menambah, memperbaharui, dan menghapus resource role pada suatu resource. Tabel 3.17 menunjukan spesifikasi kasus penggunaan dan Gambar 3.18 menunjukan diagram aktivitasnya.

Tabel 3.17 Spesifikasi kasus penggunaan atur resource role

Komponen	Deskripsi
Nama	Kelola Resource Role
Nomor	UC-016
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data resource role pada suatu resource
Tipe	Fungsional
Aktor	Developer, Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Developer atau Super Admin dan berada pada halaman pengelolaan resource
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan resource role 2. Sistem menampilkan halaman pengelolaan resource role pada resource yang dipilih 3. Aktor memilih salah satu pilihan pengelolaan resource role 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan resource role 3.a.1 Aktor memilih pilihan tambah resource role 3.a.2 Sistem menampilkan halaman penambahan resource role 3.a.3 Aktor mengisi isian data resource role dan memilih submit 3.b Penghapusan resource role 3.b.1 Aktor memilih pilihan hapus resource role 3.b.2 Sistem menampilkan pesan konfirmasi 3.b.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.18 Diagram aktivitas atur resource role

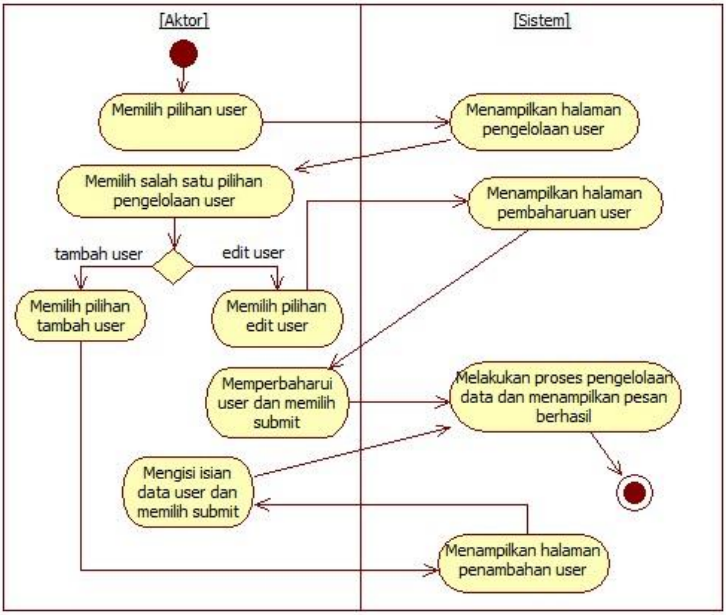
3.1.3.20 Kelola User

Aktor dapat mengelola data user pada sistem. Aktor dapat menambah dan memperbaharui user. Tabel 3.18 menunjukan spesifikasi kasus penggunaan kelola user dan Gambar 3.19 menunjukan diagram aktivitasnya.

Tabel 3.18 Spesifikasi kasus penggunaan kelola user

Komponen	Deskripsi
Nama	Kelola User
Nomor	UC-017
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data user

Tipe	Fungsional
Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan user 2. Sistem menampilkan halaman pengelolaan user 3. Aktor memilih salah satu pilihan pengelolaan user 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan user 3.a.1 Aktor memilih pilihan tambah user 3.a.2 Sistem menampilkan halaman penambahan user 3.a.3 Aktor mengisi isian data user dan memilih submit 3.b Pembaharuan user 3.b.1 Aktor memilih pilihan perbaharui user 3.b.2 Sistem menampilkan halaman pembaharuan user 3.b.3 Aktor memperbaharui data user dan memilih submit
Eksepsi	-



Gambar 3.19 Diagram aktivitas kelola user

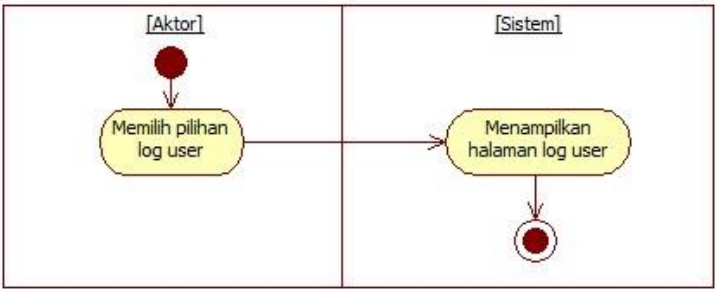
3.1.3.21 Lihat Log User

Aktivitas *end-user* tercatat dalam sistem. Aktor dapat melihat aktivitas *end-user* dari log user. Tabel 3.19 menunjukan spesifikasi kasus penggunaan lihat log user dan Gambar 3.20 menunjukkan diagram aktivitasnya.

Tabel 3.19 Spesifikasi kasus penggunaan lihat log user

Komponen	Deskripsi
Nama	Lihat Log User
Nomor	UC-018
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat melihat log user

Tipe	Fungsional
Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman pengelolaan user
Kondisi Akhir	Sistem menampilkan halaman log user yang dipilih
Alur Normal	1. Aktor memilih pilihan lihat log user 2. Sistem menampilkan halaman log user yang dipilih
Alur Alternatif	-
Eksepsi	-



Gambar 3.20 Diagram aktivitas lihat log user

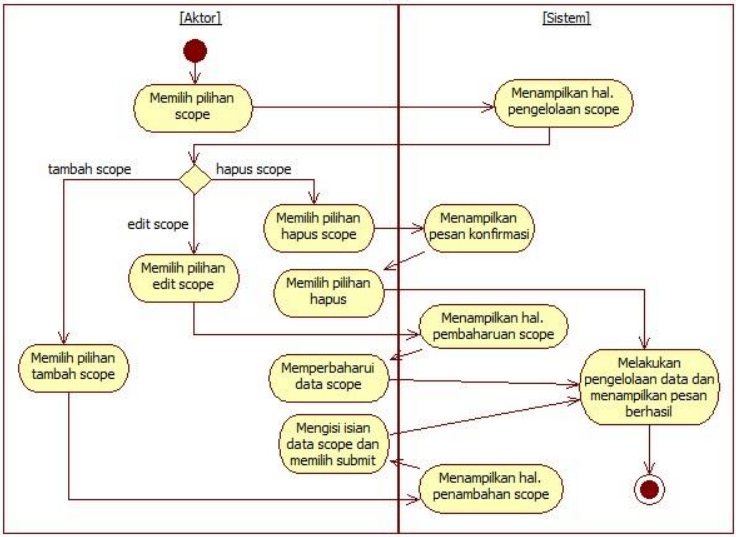
3.1.3.22 Kelola Scope

Aktor dapat mengelola data scope pada sistem. Scope digunakan untuk menentukan informasi yang diberikan kepada *client* dalam proses otentikasi dan otorisasi. Aktor dapat menambah, memperbaharui, dan menghapus scope. Tabel 3.20 menunjukan spesifikasi kasus penggunaan kelola scope dan Gambar 3.21 menunjukan diagram aktivitasnya.

Tabel 3.20 Spesifikasi kasus penggunaan kelola scope

Komponen	Deskripsi
Nama	Kelola Scope
Nomor	UC-019
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data scope
Tipe	Fungsional
Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan scope 2. Sistem menampilkan halaman pengelolaan scope 3. Aktor memilih salah satu pilihan pengelolaan scope 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan scope 3.a.1 Aktor memilih pilihan tambah scope 3.a.2 Sistem menampilkan halaman penambahan scope 3.a.3 Aktor mengisi isian data scope dan memilih submit 3.b Pembaharuan scope 3.b.1 Aktor memilih pilihan edit scope 3.b.2 Sistem menampilkan halaman pembaharuan scope 3.b.3 Aktor memperbaharui data scope 3.c Penghapusan scope 3.c.1 Aktor memilih pilihan hapus scope

	3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.21 Diagram aktivitas kelola scope

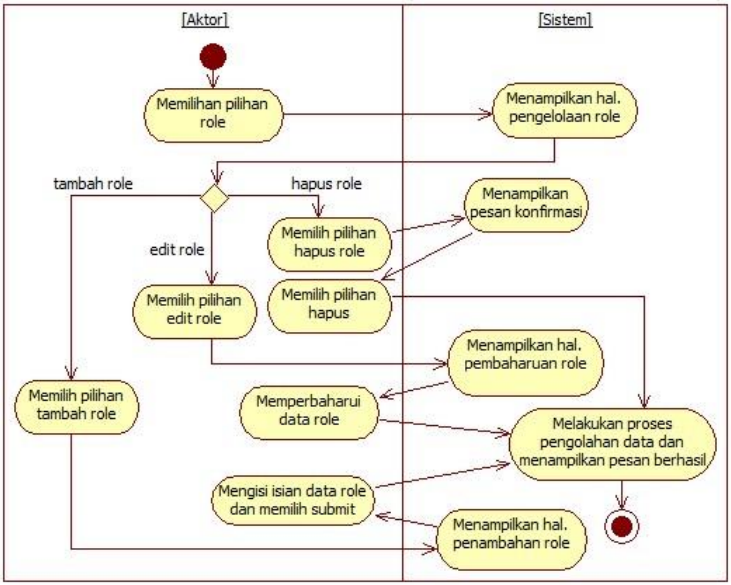
3.1.3.23 Kelola Role

Aktor dapat mengelola data peran (*role*) pada sistem. Aktor dapat menambah, memperbaharui, dan menghapus *role*. Tabel 3.21 menunjukan spesifikasi kasus penggunaan kelola role dan Gambar 3.22 menunjukan diagram aktivitasnya.

Tabel 3.21 Spesifikasi kasus penggunaan kelola role

Komponen	Deskripsi
Nama	Kelola Role
Nomor	UC-020

Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data role
Tipe	Fungsional
Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan role 2. Sistem menampilkan halaman pengelolaan role 3. Aktor memilih salah satu pilihan pengelolaan role 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan role 3.a.1 Aktor memilih pilihan tambah role 3.a.2 Sistem menampilkan halaman penambahan role 3.a.3 Aktor mengisi isian data role dan memilih submit 3.b Pembaharuan role 3.b.1 Aktor memilih pilihan edit role 3.b.2 Sistem menampilkan halaman pembaharuan role 3.b.3 Aktor memperbaharui data role 3.c Penghapusan role 3.c.1 Aktor memilih pilihan hapus role 3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.22 Diagram aktivitas kelola role

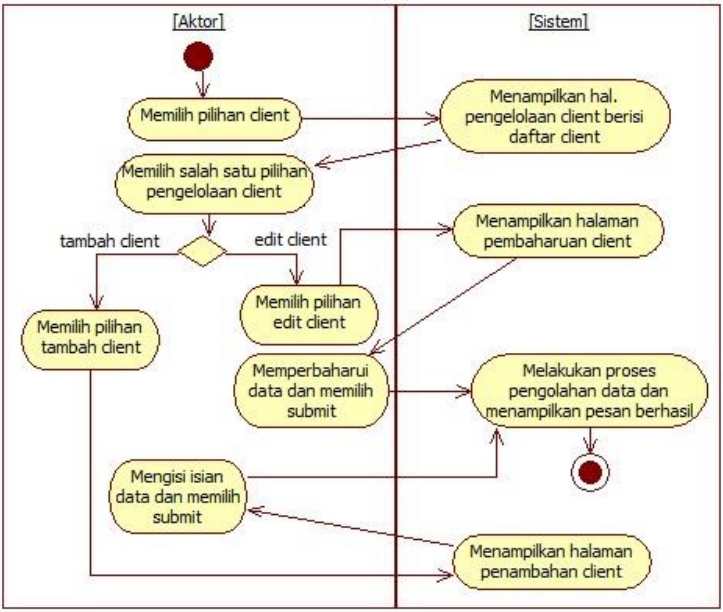
3.1.3.24 Kelola Client

Aktor dapat mengelola data *client* pada sistem. Aktor dapat menambah atau mendaftarkan *client* pada sistem dan memperbaharuinya. Tabel 3.22 menunjukan spesifikasi kasus penggunaan kelola client dan Gambar 3.23 menunjukan diagram aktivitasnya.

Tabel 3.22 Spesifikasi kasus penggunaan kelola client

Komponen	Deskripsi
Nama	Kelola Client
Nomor	UC-021
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah dan memperbaharui client
Tipe	Fungsional

Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan client 2. Sistem menampilkan halaman pengelolaan client 3. Aktor memilih salah satu pilihan pengelolaan client 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan client 3.a.1 Aktor memilih pilihan tambah client 3.a.2 Sistem menampilkan halaman penambahan client 3.a.3 Aktor mengisi isian data client dan memilih submit 3.b Pembaharuan client 3.b.1 Aktor memilih pilihan edit client 3.b.2 Sistem menampilkan halaman pembaharuan client 3.b.3 Aktor memperbaharui data dan memilih submit
Eksepsi	-



Gambar 3.23 Diagram aktivitas kelola client

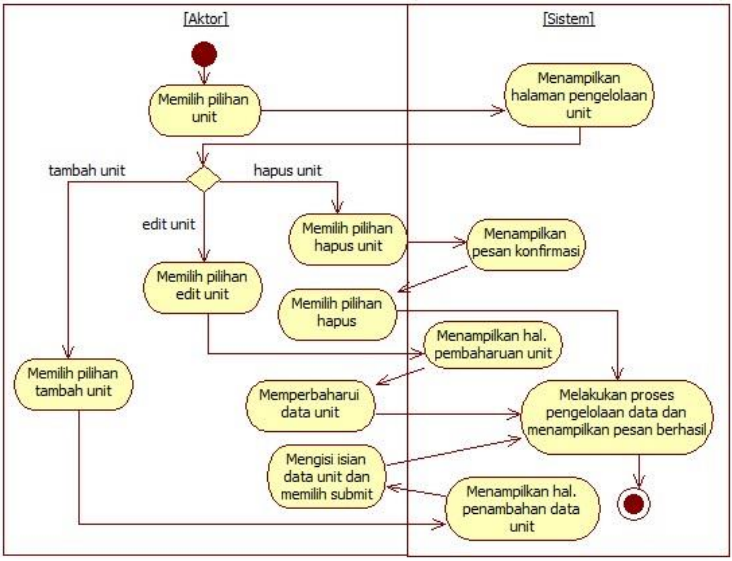
3.1.3.25 Kelola Unit

Aktor dapat mengelola data unit pada sistem. Aktor dapat menambah, memperbaharui, dan menghapus unit. Tabel 3.23 menunjukan spesifikasi kasus penggunaan kelola unit dan Gambar 3.24 menunjukan diagram aktivitasnya.

Tabel 3.23 Spesifikasi kasus penggunaan kelola unit

Komponen	Deskripsi
Nama	Kelola Unit
Nomor	UC-022
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menambah, memperbaharui, dan menghapus data unit

Tipe	Fungsional
Aktor	Super Admin
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin dan berada pada halaman dashboard myITS Security Management
Kondisi Akhir	Sistem menyimpan perubahan yang dilakukan aktor.
Alur Normal	1. Aktor memilih pilihan unit 2. Sistem menampilkan halaman pengelolaan unit 3. Aktor memilih salah satu pilihan pengelolaan unit 4. Sistem melakukan proses pengelolaan data dan menampilkan pesan berhasil
Alur Alternatif	3.a Penambahan unit 3.a.1 Aktor memilih pilihan tambah unit 3.a.2 Sistem menampilkan halaman penambahan unit 3.a.3 Aktor mengisi isian data unit dan memilih submit 3.b Pembaharuan unit 3.b.1 Aktor memilih pilihan edit unit 3.b.2 Sistem menampilkan halaman pembaharuan unit 3.b.3 Aktor memperbaharui data unit 3.c Penghapusan unit 3.c.1 Aktor memilih pilihan hapus unit 3.c.2 Sistem menampilkan pesan konfirmasi 3.c.3 Aktor memilih pilihan hapus
Eksepsi	-



Gambar 3.24 Diagram aktivitas kelola unit

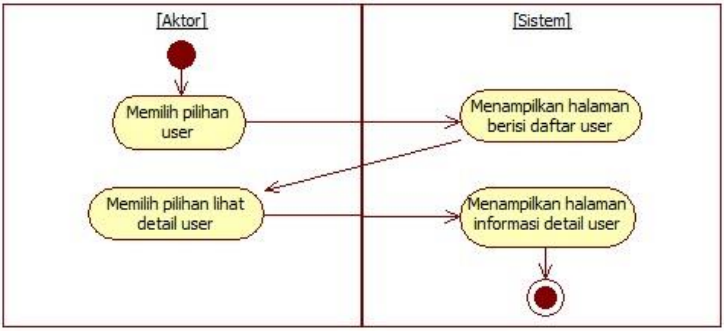
3.1.3.26 Lihat User

Pengguna dengan peran *Helpdesk* atau *Super Admin*, dapat melihat data seluruh *end-user* yang terdaftar dalam sistem. Tabel 3.24 menunjukan spesifikasi kasus penggunaan lihat user dan Gambar 3.25 menunjukan diagram aktivitasnya.

Tabel 3.24 Spesifikasi kasus penggunaan lihat user

Komponen	Deskripsi
Nama	Lihat User
Nomor	UC-023
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat melihat data user
Tipe	Fungsional
Aktor	Super Admin, Helpdesk

Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin atau Helpdesk dan berada pada dashboard myITS Security Management
Kondisi Akhir	Sistem menampilkan halaman informasi detail user
Alur Normal	1. Aktor memilih pilihan user 2. Sistem menampilkan halaman berisi daftar user 3. Aktor memilih pilihan lihat detail user 4. Sistem menampilkan halaman informasi detail user
Alur Alternatif	-
Eksepsi	-



Gambar 3.25 Diagram aktivitas lihat user

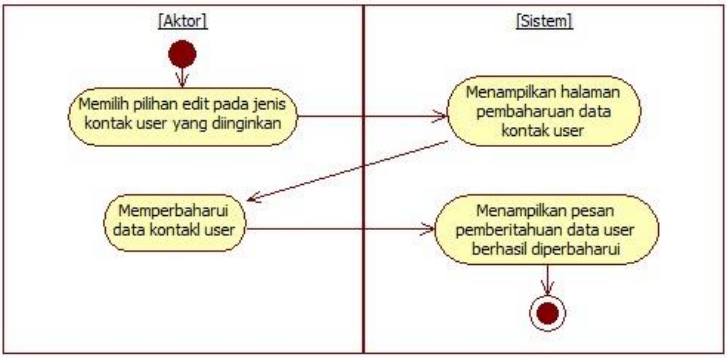
3.1.3.27 Edit Kontak User

Pengguna dengan peran Super Admin atau H*elpdesk* dapat memperbaharui data email dan email alternatif pengguna (*end-user*) yang terdaftar dalam sistem. Tabel 3.25 menunjukan spesifikasi kasus penggunaan edit email user dan Gambar 3.26 menunjukan diagram aktivitasnya.

Tabel 3.25 Spesifikasi kasus penggunaan edit email user

Komponen	Deskripsi

Nama	Edit Kontak User
Nomor	UC-024
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat memperbaharui email dan no. ponsel user
Tipe	Fungsional
Aktor	Super Admin, Helpdesk
Kondisi Awal	Aktor telah terotentikasi dengan peran SuperAdmin atau Helpdesk dan berada pada halaman daftar user
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan data user berhasil diperbaharui
Alur Normal	1. Aktor memilih pilihan edit pada jenis kontak user yang diinginkan 2. Sistem menampilkan halaman perbaharuan data kontak user 3. Aktor memperbaharui data kontak user 4. Sistem menampilkan pesan pemberitahuan data user berhasil diperbaharui
Alur Alternatif	-
Eksepsi	-



Gambar 3.20 Diagram aktivitas edit kontak user

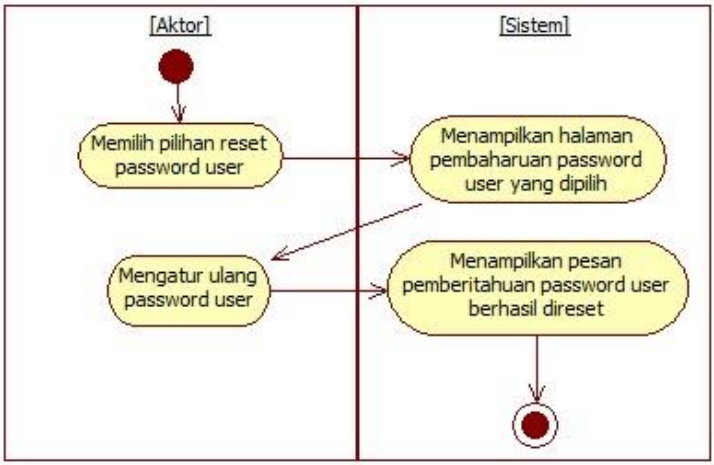
3.1.3.28 Reset *Password* User

Pengguna dengan peran Super Admin atau H*elpdesk* dapat mengatur ulang *password* pengguna (*end-user*) yang terdaftar

dalam sistem. Tabel 3.26 menunjukan spesifikasi kasus penggunaan reset *password* user dan Gambar 3.27 menunjukan diagram aktivitasnya.

Tabel 3.26 Spesifikasi kasus penggunaan reset *password* user

Komponen	Deskripsi
Nama	Reset *Password* User
Nomor	UC-025
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat mengatur ulang *password* user
Tipe	Fungsional
Aktor	Super Admin, Helpdesk
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin atau Helpdesk dan berada pada halaman daftar user
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan *password* user berhasil di reset
Alur Normal	1. Aktor memilih pilihan reset *password* user 2. Sistem menampilkan halaman perbaharuan *password* user yang dipilih 3. Aktor mengatur ulang *password* user 4. Sistem menampilkan pesan pemberitahuan *password* user berhasil di reset
Alur Alternatif	-
Eksepsi	-



Gambar 3.27 Diagram aktivitas reset *password* user

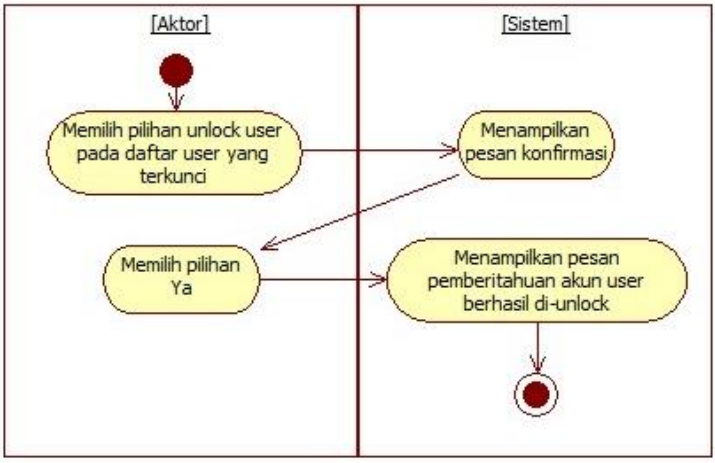
3.1.3.29 Unlock User

Akun e*nd-user* dapat ter-*suspend* selama 15 menit oleh sistem apabila telah gagal melakukan *login* sebanyak 5 kali dalam 5 menit. Jika *end-user* melakukan kesalahan kembali seperti sebelumnya setelah pernah ter-*suspend* maka akun *end-user* akan terkunci dan tidak dapat melakukan *login*. Pengguna dengan peran Super Admin atau H*elpdesk* dapat membuka akun *end-user* yang terkunci. Tabel 3.27 menunjukan spesifikasi kasus penggunaan unlock user dan Gambar 3.28 menunjukan diagram aktivitasnya.

Tabel 3.27 Spesifikasi kasus penggunaan unlock user

Komponen	Deskripsi
Nama	Unlock User
Nomor	UC-026
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat membuka akun *end-user* yang terkunci
Tipe	Fungsional

Aktor	Super Admin, Helpdesk
Kondisi Awal	Aktor telah terotentikasi dengan peran Super Admin atau Helpdesk dan berada pada halaman daftar user
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan akun user berhasil di *unlock*
Alur Normal	1. Aktor memilih pilihan *unlock user* pada daftar user yang terkunci 2. Sistem menampilkan pesan konfirmasi 3. Aktor memilih pilihan 'Ya' 4. Sistem menampilkan pesan pemberitahuan akun user berhasil di *unlock*
Alur Alternatif	-
Eksepsi	-



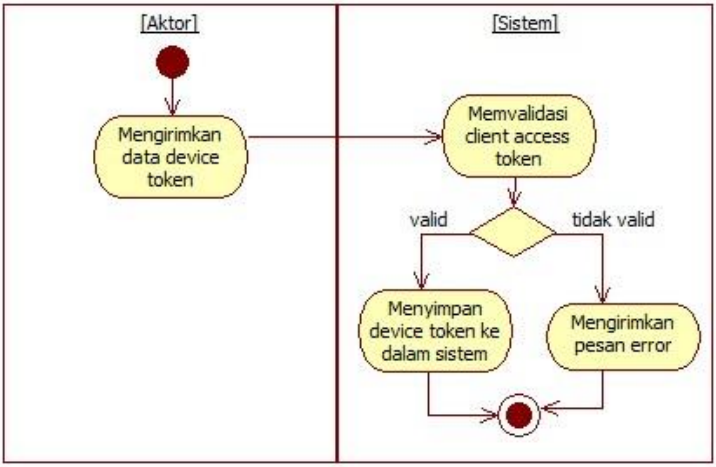
Gambar 3.28 Diagram aktivitas unlock user

3.1.3.30 Insert Device Token

Aplikasi (*client*) yang terdaftar dalam sistem, dapat mengakses API untuk menyimpan *device token* pengguna pada suatu aplikasi dalam sistem. Untuk mengakses API, *client* menggunakan *access token* untuk *client* yang telah diperoleh sebelumnya. Tabel 3.28 menunjukan spesifikasi kasus penggunaan insert device token dan Gambar 3.29 menunjukan diagram aktivitasnya.

Tabel 3.28 Spesifikasi kasus penggunaan insert device token

Komponen	Deskripsi
Nama	Insert Device Token
Nomor	UC-027
Deskripsi	Kasus penggunaan ini adalah kasus dimana aktor dapat menyimpan device token dalam sistem
Tipe	Fungsional
Aktor	Client
Kondisi Awal	Aktor yang telah terdaftar dalam sistem memiliki *client access token*
Kondisi Akhir	Sistem manyimpan *device token* dalam sistem
Alur Normal	1. Aktor mengirimkan data device token 2. Sistem mem-validasi *client access token* 3. Sistem menyimpan *device token* dalam sistem
Alur Alternatif	-
Eksepsi	*3.a Client access token* yang dikirimkan tidak valid 3.a.1 Sistem mengirimkan pesan error



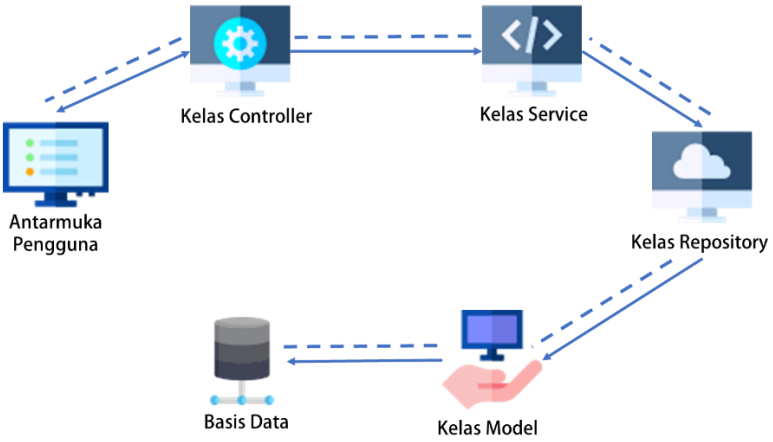
Gambar 3.29 Diagram aktivitas insert device token

3.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak yang dibahas meliputi arsitektur sistem yang digunakan, perancangan sistem otentikasi dan otorisasi, perancangan basis data yang disajikan dalam bentuk *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM), serta perancangan antarmuka pengguna.

3.2.1 Perancangan Arsitektur Sistem

Arsitektur sistem yang digunakan pada tugas akhir ini menggunakan arsitektur sistem *framework* Phalcon-MVC. Arsitektur Phalcon-MVC terdapat pada Gambar 3.30.



Gambar 3.30 Ilustrasi arsitektur sistem

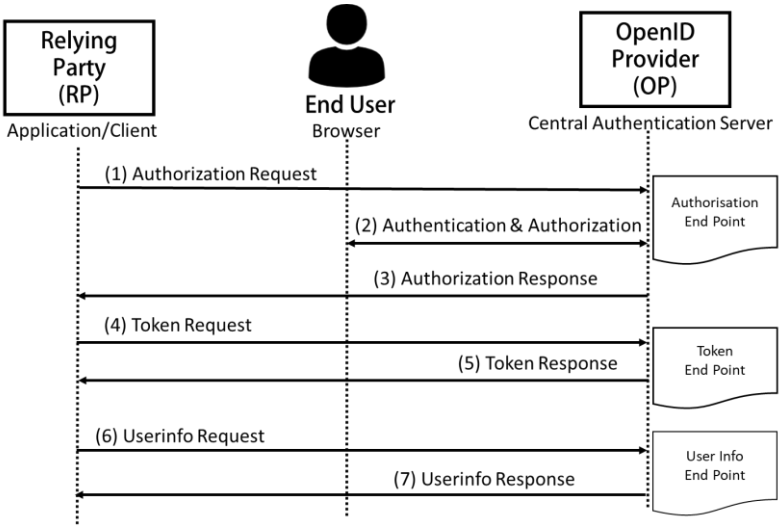
Antarmuka pengguna merupakan lapisan yang berhubungan langsung dengan pengguna. *Controller* adalah penghubung antara antarmuka dengan lapisan *service* dari sistem. Lapisan *service* menyediakan proses pengolahan data dari lapisan *repository*. Kelas *repository* menyediakan akses ke basis data melalui kelas *model*. Kelas r*epository* akan diakses oleh kelas *service* untuk menindaklanjuti proses yang dikirim lewat *controller*. Setelah *controller* mendapatkan data yang dikembalikan, *controller* akan menampilkan pada antarmuka pengguna.

Kelas *repository* dan kelas *model* merupakan representasi dari *model* dalam arsitektur MVC (*model-view-controller*). Kelas *service* dan *controller* merupakan representasi dari kelas *controller*, sedangkan antarmuka pengguna merupakan representasi dari *view*.

3.2.2 Perancangan Sistem Otentikasi dan Otorisasi

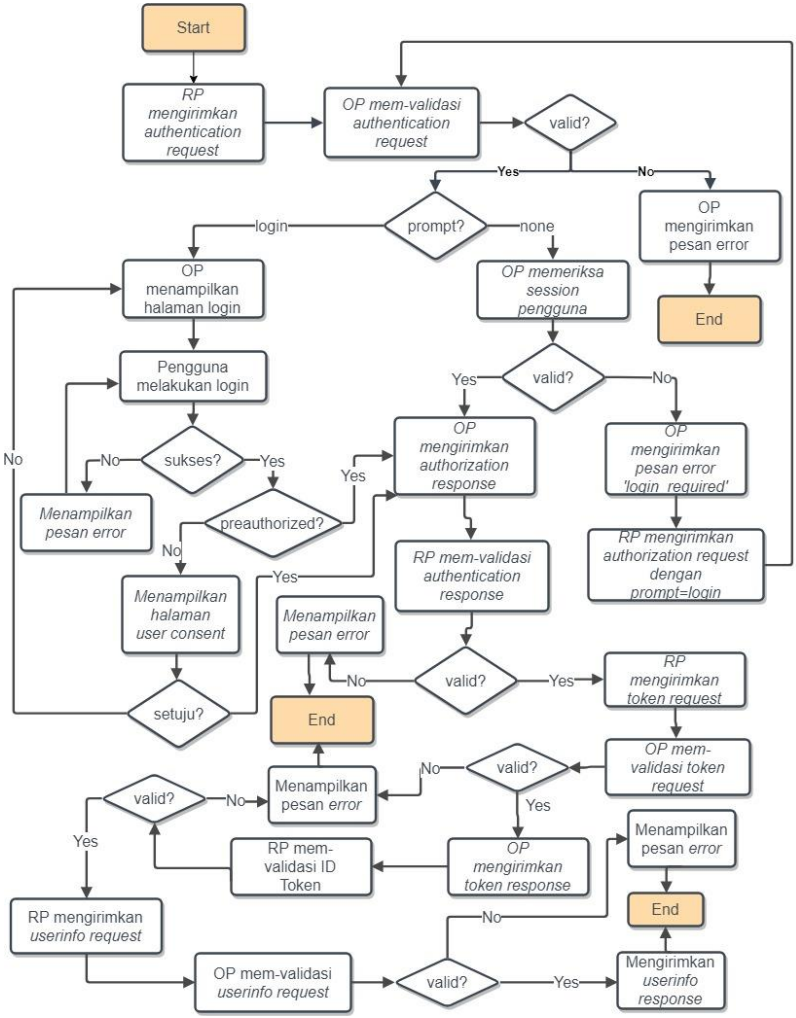
Penerapan sistem otentikasi dan otorisasi yang akan dibagun menggunakan standar OpenID Connect yang merupakan

lapisan otentikasi pada protokol otorisasi OAuth2. *Flow* yang digunakan adalah *authorization code flow*, dimana terdapat 3 *endpoint*, yaitu *authorization endpoint*, *token endpoint*, dan *userinfo endpoint* yang dapat dilihat pada Gambar 3.31.



Gambar 3.31 Rancangan Mekanisme Otentikasi dan Otorisasi Sistem

Gambar 3.32 berikut merupakan rancangan sistem yang dijelaskan dalam bentuk diagram alir (*flowchart*) sistem.



Gambar 3.32 *Flowchart* Sistem

Terdapat 7 langkah proses otentikasi dan otorisasi sistem sesuai *authorization code flow*, yang masing-masing akan dijelaskan sebagai berikut.

1. *Authorization request*

Aplikasi *client* atau dapat disebut *relying party* (RP) mengirimkan *authorization request* ke sistem myITS atau server *OpenID Provider* (OP) pada *authorization endpoint* (/authorization). *Request* yang dikirimkan dapat menggunakan metode GET maupun POST. Berikut adalah parameter yang disertakan.

- response_type (*required)*
  Nilai dari parameter ini adalah *code*, karena menggunakan *authorization code flow*.
- client_id (*required)*
  Nilai dari parameter ini adalah ID Client yang didapatkan *developer* ketika aplikasi didaftarkan atau ditambahkan di dalam sistem.
- redirect_uri (*required*)
  Nilai dari parameter ini adalah *redirect_uri* yang disimpan dalam sistem. *Redirect URI* ini adalah dimana aplikasi atau *client* (RP) menangani *response* dari sistem (OP).
- scope *(required*)
  Parameter *scope* dapat diisi lebih dari satu dan minimal satu nilai wajib, yaitu: *openid*. Nilai *openid* akan mengembalikan nilai *sub* yaitu ID *User* pengguna. Parameter lainnya yang dapat ditambahkan adalah:
  - profile
    Scope *profile* mengembalikan informasi mengenai nilai *name, nickname, username, picture, gender, birthdate, zoneinfo, locale, last_update,* dan *integra_id* pengguna.
  - email
    Scope *email* mengembalikan informasi mengenai nilai *email, email_verified, alternate_email,* dan *alternate_email_verified* pengguna.
  - phone
    Scope *phone* mengembalikan informasi mengenai nilai *phone* dan *phone_verified* pengguna.

- roleunit
  Scope *roleunit* mengembalikan informasi mengenai nilai *role_id, role_name, unit_id, unit_name,* dan *role_default* pengguna.
- menu
  Scope *menu* mengembalikan informasi mengenai nilai *menu_id, parent_id, menu_name, name_en, path, can_insert, can_update, can_delete, menu_order, role_name,* dan *icon* menu yang dapat diakses pengguna.
- resource
  Scope *resource* mengembalikan informasi mengenai nilai *resource_id, name, path, method* resource yang dapat diakses pengguna.

– prompt (*optional)*
  Parameter *prompt* merupakan parameter pilihan, dapat diisi maupun tidak. Terdapat dua nilai parameter *prompt*, yaitu:
  - *login*
    Prompt *login* menandakan bahwa pengguna harus melakukan otentikasi dengan memasukkan *username* dan *password*. Sistem harus menampilkan halaman *login* untuk pengguna. Prompt *login* adalah nilai *default* jika parameter prompt tidak disertakan pada *authorization request*.
  - none
    Prompt *none* menandakan bahawa pengguna tidak perlu melakukan proses otentikasi. Saat menerima *authorization request* dengan nilai *prompt=none*, maka sistem akan memeriksa *session* pengguna. Jika valid, maka akan langsung menampilkan halaman dashboard myITS dan jika tidak maka pesan *error "login_required"* akan dikembalikan ke *redirect_uri* client.

– state (*required)*
  Parameter *state* merupakan parameter yang wajib diisi. Nilai dari parameter *state* berisi *random string* yang dibuat oleh *client*. Nilai ini nantinya yang digunakan *client* untuk

memeriksa apakah *authorization response* yang diterima valid dan berasal dari sumber (OP) yang benar.

Kode Sumber 3.1 berikut adalah contoh isi dari *authorization request* dengan method GET dan POST.

```
GET /authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
HTTP/1.1
  Host: server.example.com
```

Kode Sumber 3.1 Contoh a*uthentication request* dengan method GET

2. *Authentication* dan *authorization*

Setelah mengirimkan *authorization request*, sistem OP akan memeriksa apakah *request* yang diterima valid, yaitu dengan memeriksa ke-validan masing-masing parameter. Jika tidak sesuai atau tidak valid, sistem akan menampilkan pesan *error*. Jika sesuai dan valid, maka sistem akan menampilkan halaman *login*.

Pengguna kemudian melakukan proses otentikasi atau proses *login* dengan memasukkan *username* dan *password*. Saat proses otentikasi berhasil sistem akan melanjutkan dengan dua pilihan. Yang pertama, jika aplikasi *client* merupakan aplikasi internal (*preauthorized application*), maka sistem langsung menampilkan halaman dashboard sistem atau dashboard aplikasi *client*. Namun jika aplikasi *client* merupakan aplikasi eksternal (*non-preauthorized application*), maka sistem OP akan menampilkan halaman *user consent* untuk meminta persetujuan pengguna.

3. *Authorization response*

Saat proses otentikasi dan otorisasi berhasil, tahap selanjutnya adalah sistem mengirimkan balasan berupa

authentication response. Sistem OP akan mengirimkan *authorization code*, dan *state*. Nilai *state* yang dikirimkan adalah nilai *state* awal yang diterima pada *authentication request*. Kode Sumber 3.2 *b*erikut adalah contoh *authorization response* yang dikirimkan OP.

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

Kode Sumber 3.2 Contoh a*uthentication response*

4. *Token request*

Aplikasi *client* (RP) menerima *authentication response* dari sistem OP. RP harus memeriksa bahwa *authentication response* dikirim dari sumber yang benar dengan mencocokan nilai *state* yang diterima dengan yang awal dikirimkan. Jika sesuai, langkah selanjutnya adalah mengirimkan *token request* menggunakan method POST ke *token endpoint* pada OP menggunakan *authorization code* yang didapatkan. Kode Sumber 3.3 berikut adalah contoh *token request* yang dikirimkan.

```
POST /token HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded
  Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6W
xSbIA
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Kode Sumber 3.3 Contoh *token request*

5. *Token response*

Token request kemudian diterima oleh sistem OP dan sistem melakukan pemeriksaan *request*. OP memeriksa apakah *authorization code* yang dikirimkan valid dan belum kadaluarsa.

Jika tidak valid atau telah kadaluarsa, sistem akan mengirimkan *response* berisi pesan *error*. Jika valid, maka sistem akan mengirimkan *token response* berisi *token ID, access token, refresh token, scope, expires_in,* dan *token_type*. Nilai *expires_in* merupakan waktu *access token* kadaluarsa. Kode Sumber 3.4 berikut adalah contoh *token response* yang diterima aplikasi *client* (RP).

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlz
cyI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIi
OiAiMjQ4Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIs
CiAibm9uY2UiOiAibi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzEx
MjgxOTcwLAogImlhdCI6IDEzMTEyODA5NzAKfQ.ggW8hZ1EuVLux
NuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6qJp6IcmD
3HP99Obi1PRscwh3LOp146waJ8IhehcwL7F09JdijmBqkvPeB2T9
CJNqeGpegccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqG
ByKHiOtX7TpdQyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZT
F_m0_N0YzFC6g6EJbOEoRoSK5hoDalrcvRYLSrQAZZKflyuVCyix
EoV9GfNQC3_osjzw2PAithfubEEBLuVVk4XUVrWOLrLl0nx7RkKU
8NXNHq-rvKMzqg"
  }
```

Kode Sumber 3.4 Contoh *token response*

Setelah menerima *token response*, aplikasi *client* (RP) disarankan melakukan validasi ID Token, dengan mem-validasi *signature* dari JWT ID Token. Proses validasi ID Token menggunakan *public key* yang didapat oleh *client* yang telah terdaftar pada sistem. Jika ID Token tidak valid, maka aplikasi

client harus menghentikan proses dan menampilkan pesan *error*. Jika berhasil maka proses dapat dilanjutkan ke tahap *userinfo*.

6. *Userinfo request*

Aplikasi *client* (RP) mengirimkan *userinfo request* menggunakan method GET ke *userinfo endpoint* pada sistem OP menggunakan *access token* yang didapat pada *token response*. Kode Sumber 3.5 berikut adalah contoh *userinfo request*.

```
GET /userinfo HTTP/1.1
  Host: server.example.com
  Authorization: Bearer SlAV32hkKG
```

Kode Sumber 3.5 Contoh *userinfo request*

SlAV32hkKG merupakan contoh nilai *access token*.

7. *Userinfo response*

Sistem OP menerima *userinfo request* dari RP dan mem-*validasi request* dengan memeriksa apakah *access token* valid dan belum kadaluarsa. Jika tidak valid atau sudah kadaluarsa, maka sistem OP akan mengirimkan pesan *error*. Jika valid, sistem akan mengirimkan *userinfo response* kepada RP dalam format JSON. Nilai-nilai yang dikirimkan OP bergantung pada *scope* yang dikirimkan RP dalam *authentication request*. Kode Sumber 3.6 berikut adalah contoh *userinfo response* dengan nilai *scope*: *openid profile*.

```
HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "sub": "248289761001",
   "name": "Jane Doe",
   "nickname": "Jane",
   "username": "janedoe",
   "picture": "http://example.com/janedoe/me.jpg",
```

```
  "gender": "p",
  "birthdate": "1996-8-26",
  "zoneinfo": "Asia/Jakarta",
  "locale": "ID",
  "last_update": "2018-02-06",
  "integra_id": "5114100008"
}
```

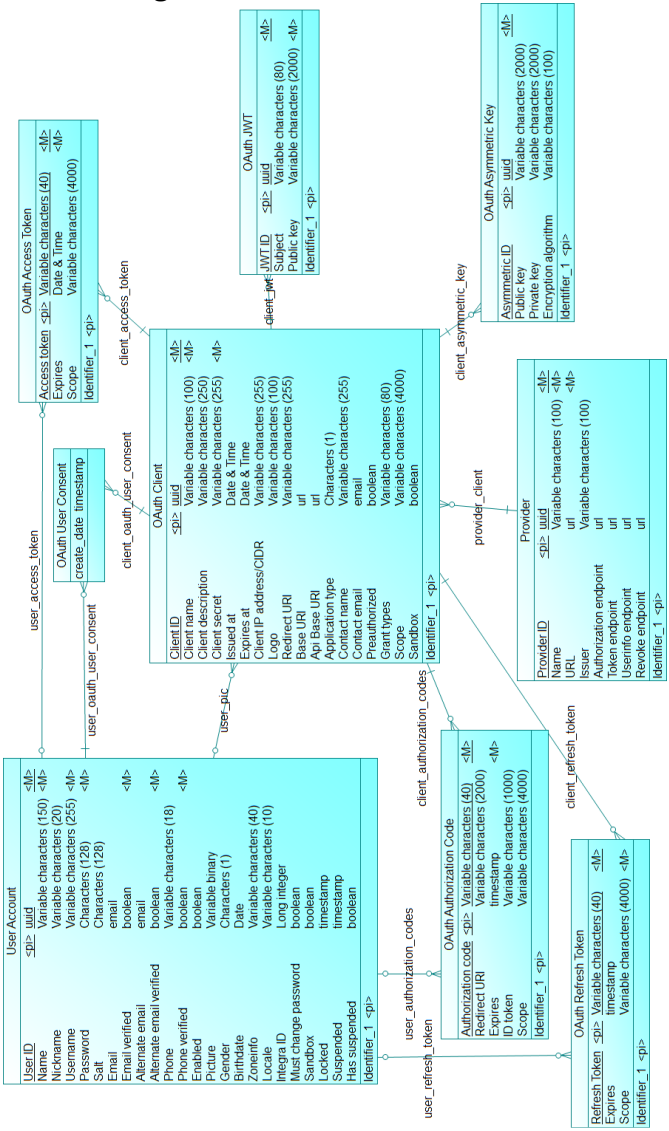
Kode Sumber 3.6 Contoh *userinfo response*

3.2.3 Perancangan Basis Data

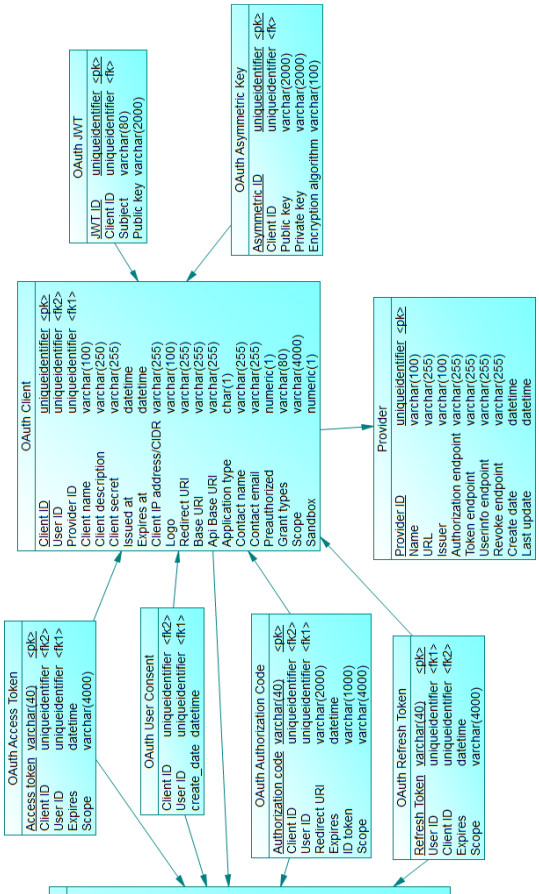
Dalam membuat suatu aplikasi berupa sistem informasi, diperlukan analisis kebutuhan berupa perancangan basis data. Basis data yang digunakan nantinya adalah SQL Server yang dipilih karena mudah digunakan dan mendukung penggunaan *Universally Unique Indentifiers* (selanjutnya disebut UUID).

Rancangan basis data ditampilkan dalam bentuk *Conceptual Data Model* (CDM) ditunjukan pada Gambar 3.33, Gambar 3.35, Gambar 3.37, Gambar 3.39, Gambar 3.41, Gambar 3.43, dan Gambar 3.45 dan *Physical Data Model* (PDM) ditunjukan pada Gambar 3.34, Gambar 3.36, Gambar 3.38, Gambar 3.40, Gambar 3.42, Gambar 3.44, dan Gambar 3.46 yang dibagi menjadi beberapa bagian untuk mempermudah menampilkan relasi antar objek dan penjelasan masing-masing objek pada diagram.

3.2.3.1 Diagram OAuth2



Gambar 3.33 Diagram CDM OAuth2



Gambar 3.34 Diagram PDM OAuth2

Diagram OAuth2 berisi tabel-tabel yang digunakan untuk proses otentikasi dan otorisasi menggunakan protokol OAuth2 dan standar otentikasi OpenID Connect. Masing-masing tabel pada diagram ini akan dijelaskan sebagai berikut.

3.2.3.1.1 Tabel Provider

Tabel Provider digunakan untuk menyimpan data provider sistem, dimana hanya terdapat 1 provider, yaitu sistem OpenID Provider myITS. Tabel provider dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

3.2.3.1.2 Tabel OAuth Client

Tabel OAuth Client menyimpan data *client* atau aplikasi (*Relying Party*) yang telah terdaftar dalam sistem. Tabel ini memiliki relasi terhadap tabel Provider sehingga Tabel OAuth Client wajib memiliki satu provider dan menyimpan *provider_id.* Tabel OAuth Client dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

3.2.3.1.3 Tabel OAuth Asymmetric Key

Tabel OAuth Asymmetric Key adalah tabel yang menyimpan kunci asimetris *client*. Setiap OAuth Client memiliki satu Asymmetric Key. Setiap asymmetric key menyimpan *private key* dan *public key* dari *client*. *Private key* digunakan OP untuk mengenkripsi *ID Token* dan p*ublic key* digunakan RP untuk mendekripsi atau memvalidasi *ID Token*. Tabel ini juga menyimpan algoritma enkripsi yang digunakan, yaitu RS256. Tabel OAuth Asymmetric Key dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34

3.2.3.1.4 Tabel OAuth Access Token

Tabel OAuth Access Token menyimpan *access token* yang telah di-*generate* sistem untuk digunakan pada *token response*. Tabel ini memiliki relasi dengan tabel OAuth Client dan tabel User Account. Satu *access token* dimiliki oleh satu *user* pada *client* tertentu. Terdapat *client* a*ccess token* dimana *access token* dimiliki oleh sebuah *client* dan *field user_id* dikosongkan. Pada tabel ini juga disimpan *scope* tergantung pada nilai *scope* yang diberikan saat *client* mengirimkan *authentication request*. Untuk *client access token, field scope* diisi nilai: *openid application*. Tabel OAuth *Access Token* dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

3.2.3.1.5 Tabel OAuth User Consent

Tabel OAuth User Consent menyimpan persetujuan *user* untuk sebuah *client* yang tidak *preauthorized* atau aplikasi eksternal menggunakan kredensialnya. Tabel ini memiliki relasi dengan tabel User Account dan tabel OAuth Client. Dalam tabel ini disimpan *ID user dan ID client* yang telah diberikan *user consent*. Tabel OAuth User Consent dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

3.2.3.1.6 Tabel OAuth Authorization Code

Tabel OAuth Authorization Code adalah tabel yang menyimpan *authorization code* yang telah di-*generate* sistem untuk digunakan sebagai *authorization response*. Tabel ini memiliki relasi dengan tabel User Account dan tabel OAuth Client. Selain menyimpan *authorization code, ID Token* pengguna juga disimpan pada tabel ini. Data akan dihapus ketika *authorization code* telah digunakan untuk mendapatkan *token response*. Tabel OAuth Authorization Code dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

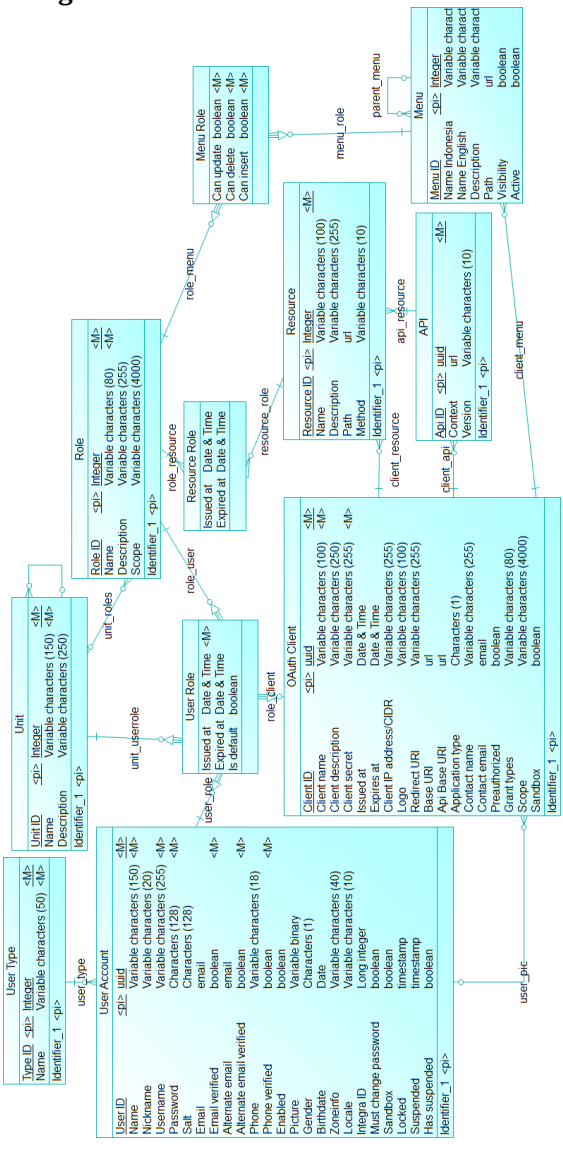
3.2.3.1.7 Tabel OAuth Refresh Token

Tabel OAuth Refresh Token menyimpan *refresh token* yang di-*generate* saat membuat *token response*. *Refresh token* digunakan untuk mendapatkan *access token* baru. Tabel ini memiliki relasi tabel User Account dan Tabel OAuth Client. Tabel OAuth Refresh Token dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

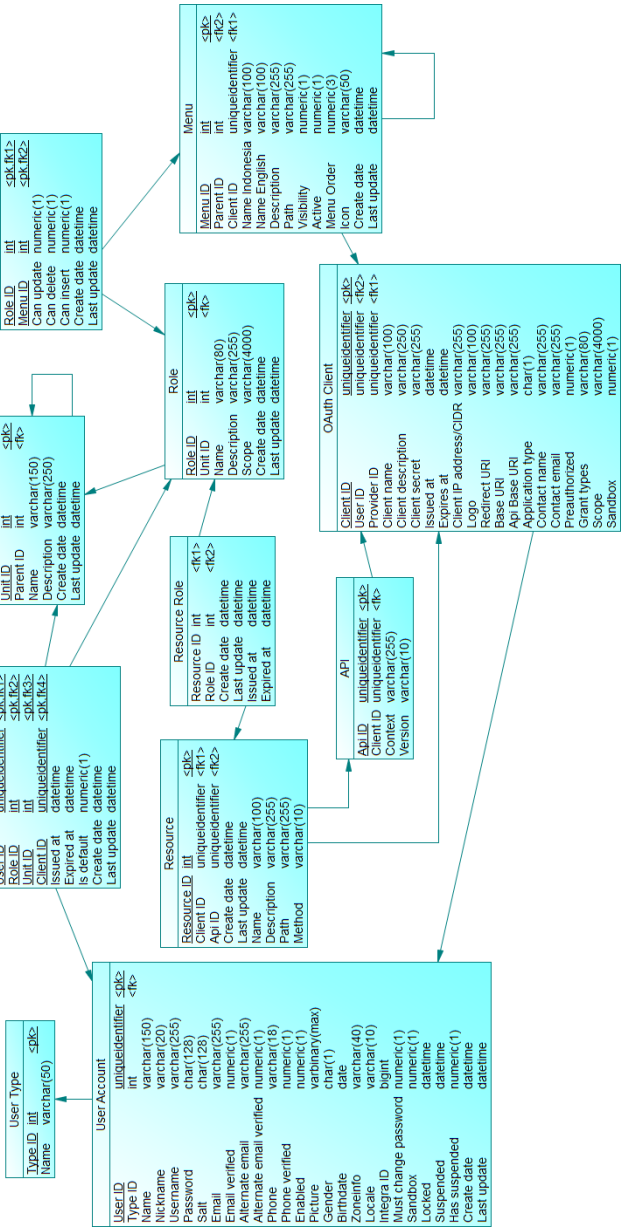
3.2.3.1.8 Tabel OAuth Scope

Tabel OAuth Scope adalah tabel yang menyimpan nilai *scope* yang dapat digunakan. Data *scope* dapat ditambahkan, diperbaharui, maupun dihapus oleh pengguna dengan peran Super Admin. Tabel OAuth Scope dapat dilihat pada diagram OAuth2 pada Gambar 3.33 dan Gambar 3.34.

3.2.3.2 Diagram RBAC



Gambar 3.35 Diagram CDM RBAC



Gambar 3.36 Diagram PDM RBAC

Diagram RBAC atau *role-based-access-control* berisi tabel-tabel yang digunakan untuk mengatur akses pengguna terhapad aplikasi berdasarkan peran yang diberikan aplikasi. Masing-masing tabel akan dijelaskan di bawah.

3.2.3.2.1 Tabel User Account

Tabel User Account adalah tabel yang menyimpan data pengguna yang terdaftar dalam sistem. Selain menyimpan informasi personal pengguna, tabel ini menyimpan informasi mengenai status akun pengguna. Dengan melihat field *locked, suspended,* dan *has_suspended,* dapat diketahui apakah pengguna dapat melakukan proses *login* atau tidak. Jika field *sandbox* bernilai 1, maka pengguna dapat melihat aplikasi yang berstatus *development*. Field *must_change_password* memberikan informasi apakah pengguna disarankan untuk mengganti *password*nya atau tidak. Tabel ini memiliki relasi dengan tabel User Type, dimana tiap pengguna memiliki satu tipe user. Tabel User Account dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.2 Tabel User Type

Tabel User Type adalah tabel *master* yang berisi data kategori tipe pengguna. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.3 Tabel User Role

Tabel User Role menyimpan data peran yang dimiliki pengguna. Tabel ini memiliki relasi dengan tabel Unit, tabel Role, dan tabel Client. Dari tabel ini dapat diketahui seorang pengguna memiliki *role* apa pada *unit* apa dan pada *client* apa. Pada tabel ini juga disimpan masa aktif peran pengguna dalam *field expired_at*. Tabel User Role dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.4 Tabel Unit

Tabel Unit adalah tabel *master* yang menyimpan data unit. Tabel Unit memiliki relasi ke dirinya sendiri yaitu pada *field parent_unit*. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.5 Tabel API

Tabel API menyimpan data API untuk setiap *client*. Tabel API memiliki relasi dengan tabel OAuth Client dan tabel Resource. API dapat memiliki satu atau lebih *resource*. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.6 Tabel Resource

Tabel Resource menyimpan data *resource* pada tiap API. Tabel ini memiliki relasi dengan tabel API dan tabel OAuth Client. Tabel Resource dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.7 Tabel Resource Role

Tabel Resource Role merupakan tabel yang menyimpan peran yang dapat mengakses suatu *resource* tertentu. Tabel ini memiliki relasi dengan tabel Resource dan tabel Role. Tabel Resource Role dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.2.8 Tabel Role

Tabel Role merupakan tabel yang menyimpan data peran dalam sistem. Tabel ini memiliki relasi terhadap tabel Unit. *Field unit_id* dapat dikosongkan. Tabel Role dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

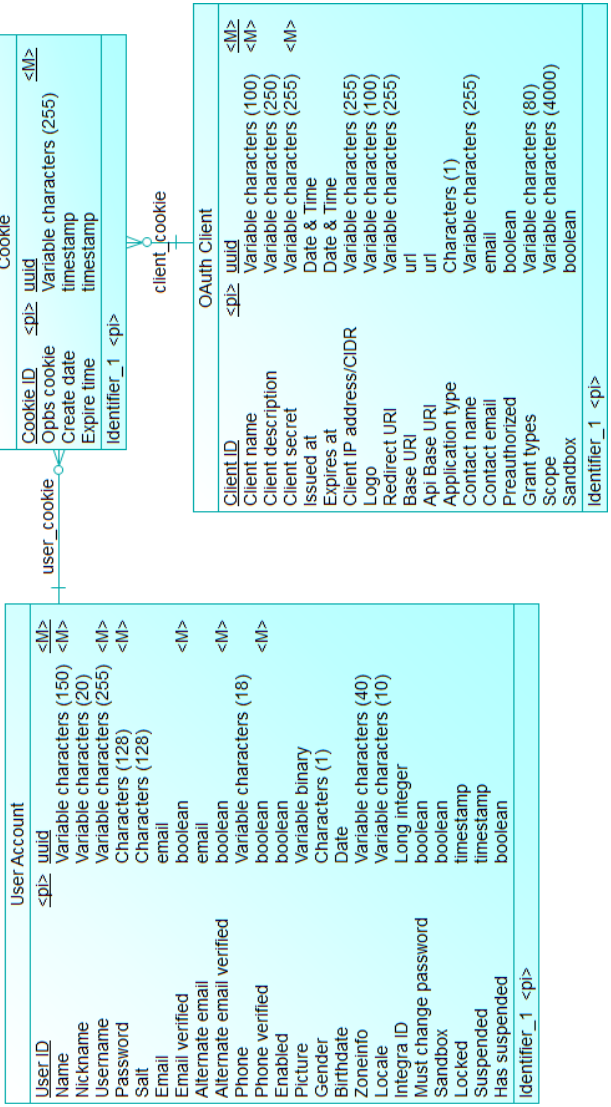
3.2.3.2.9 Tabel Menu

Tabel Menu merupakan tabel yang menyimpan data menu yang terdapat dalam sistem. Tabel ini memiliki relasi dengan tabel OAuth Client dan tabel Menu itu sendiri pada *field parent_id*. Tabel Menu dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

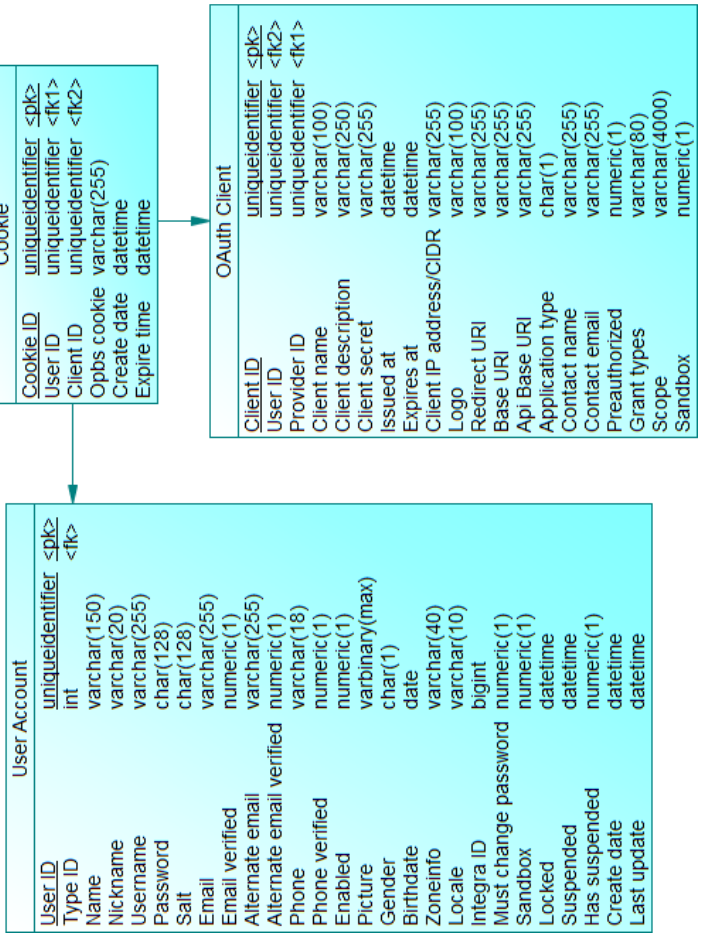
3.2.3.2.10 Tabel Menu Role

Tabel Menu Role menyimpan data peran yang dapat mengakses sebuah menu tertentu. Tabel ini memiliki relasi dengan tabel Menu dan tabel Role. Pada tabel ini juga disimpan hak akses pengguna untuk dapat melakukan penambahan, pembaharuan, atau penghapusan data pada menu tersebut. Informasi tersebut didapatkan dari *field can_insert, can_update,* dan *can_delete*. Jika bernilai 1 maka suatu peran dapat melakukan aksi tersebut dan sebaliknya jika 0 maka aksi tidak dapat dilakukan. Tabel Menu Role dapat dilihat pada diagram OAuth2 pada Gambar 3.35 dan Gambar 3.36.

3.2.3.3 Diagram Session Management



Gambar 3.37 Diagram CDM Session Management



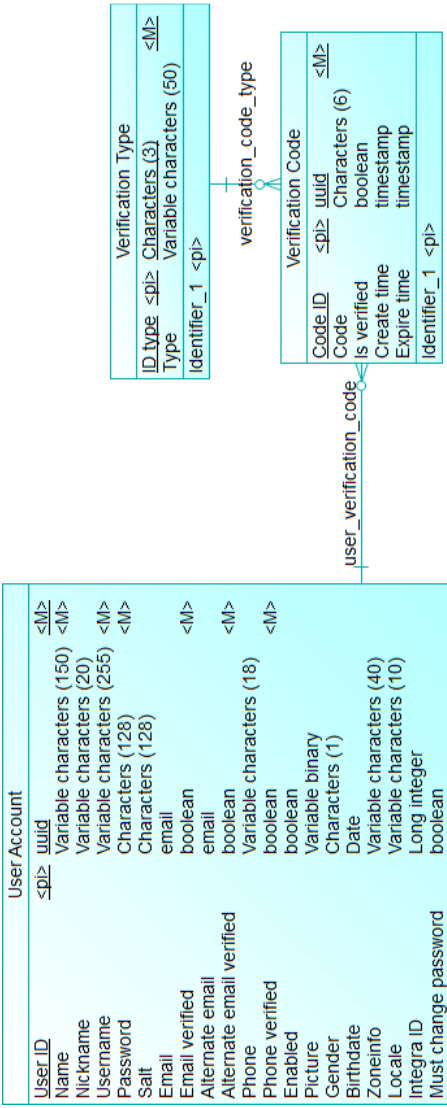
Gambar 3.38 Diagram PDM Session Management

Diagram Session Management berisi tabel-tabel yang digunakan untuk mengatur *session* pengguna yang digunakan untuk menyimpan informasi bahwa pengguna tersebut telah terotentikasi atau telah *login* ke dalam sistem. Berikut dijelaskan tabel-tabel pada diagram ini.

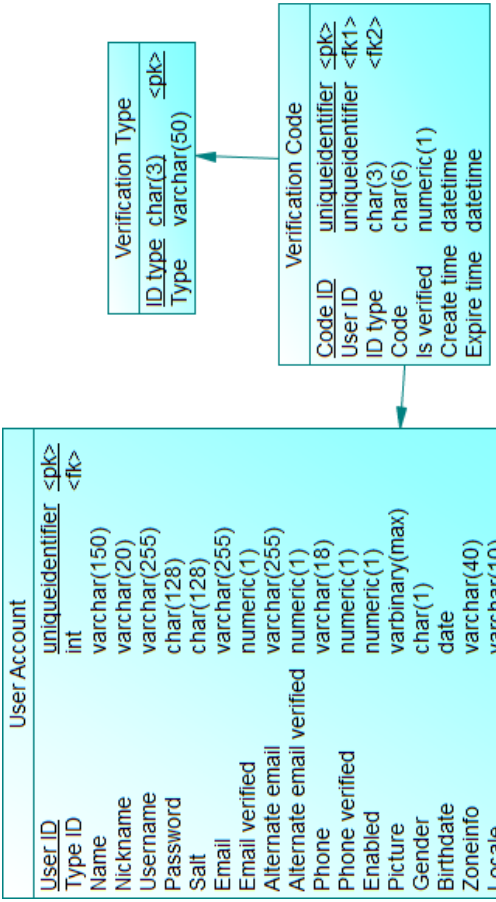
3.2.3.3.1 Tabel Cookie

Tabel Cookie merupakan tabel yang menyimpan sebuah *cookie* yang telah di-*generate* sistem. *Cookie* di-*generate* saat pengguna berhasil *login* ke dalam sistem dan *cookie* pengguna ini disimpan dalam *session* untuk menandakan bahwa status pengguna tersebut adalah telah *login* ke dalam sistem. Tabel ini memiliki relasi dengan tabel User Account dan tabel OAuth Client. Tabel Cookie dapat dilihat pada diagram OAuth2 pada Gambar 3.37 dan Gambar 3.38.

3.2.3.4 Diagram OTP



Gambar 3.39 Diagram CDM OTP



Gambar 3.40 Diagram PDM OTP

Diagram OTP berisi tabel-tabel yang digunakan untuk mengirimkan *One Time Password* pada proses verifikasi akun dan fitur lupa *password*. Berikut dijelaskan tabel-tabel dalam diagram ini.

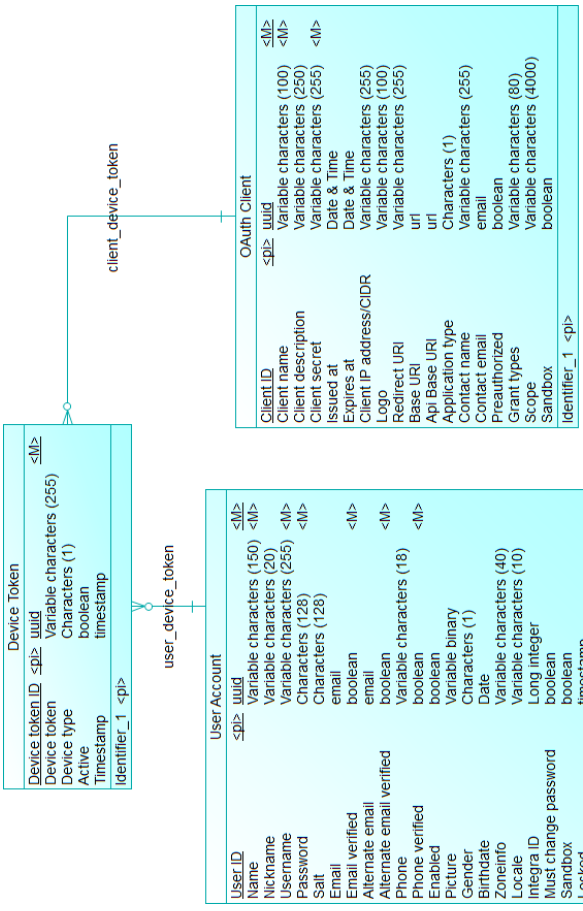
3.2.3.4.1 Tabel Verification Type

Tabel Verification Type merupakan tabel *master* yang menyimpan data kategori penggunakan kode OTP. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.39 dan Gambar 3.40.

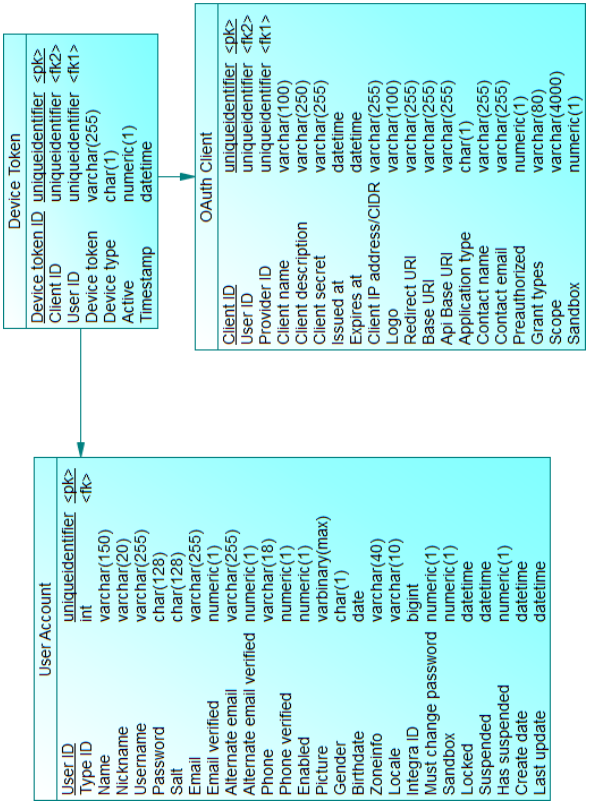
3.2.3.4.2 Tabel Verification Code

Tabel Verification Code menyimpan kode verifikasi atau kode OTP yang telah di-*generate* sistem. Setiap data pada tabel ini menyimpan kode, pengguna yang memiliki kode, kategori atau tipe verifikasi, dan status apakah kode telah digunakan atau belum. Tabel ini memiliki relasi dengan tabel User Account dan tabel Verification Type. Tabel Verification Code dapat dilihat pada diagram OAuth2 pada Gambar 3.37 dan Gambar 3.38.

3.2.3.5 Diagram Push Notification



Gambar 3.41 Diagram CDM Push Notification



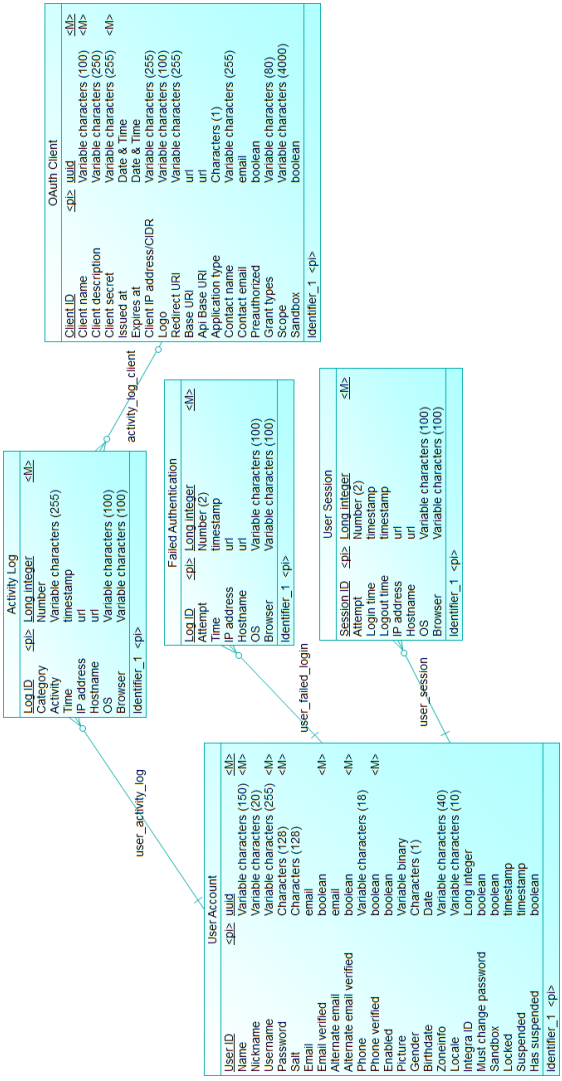
Gambar 3.42 Diagram PDM Push Notification

Diagram Push Notification berisi tabel-tabel yang digunakan untuk menyimpan *device token* pengguna pada suatu aplikasi. Saat pengguna *login* pada suatu *client* atau aplikasi, maka aplikasi *client* tersebut akan mengirimkan data *device token* pengguna ke sistem OP untuk disimpan. Berikut adalah penjelasan tabel pada diagram ini.

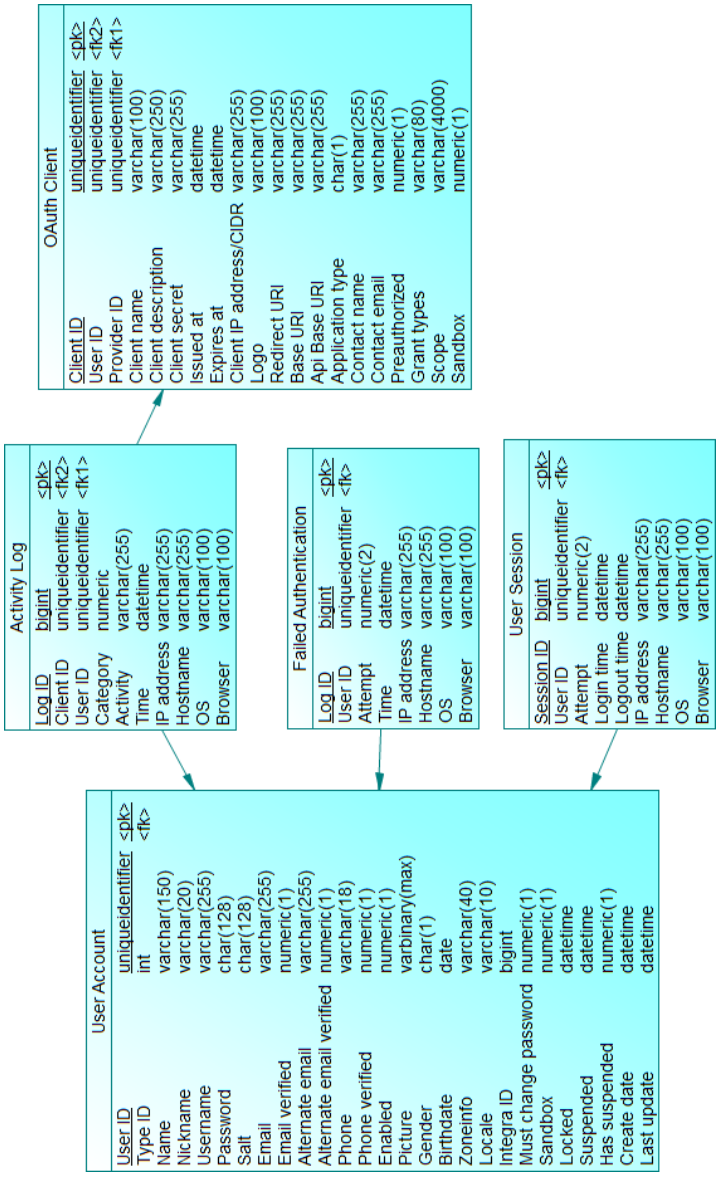
3.2.3.5.1 Tabel Device Token

Tabel Device Token menyimpan *device token* pengguna pada suatu aplikasi *client*. Selain itu, setiap data *device token* juga disimpan tipe perangkat pengguna (*android* atau iOS) dan status keaktifannya. Tabel ini memilik relasi dengan tabel OAuth Client dan tabel User Account. Tabel Device Token dapat dilihat pada diagram OAuth2 pada Gambar 3.41 dan Gambar 3.42.

3.2.3.6 Diagram Log



Gambar 3.43 Diagram CDM Log



Gambar 3.44 Diagram PDM Log

Diagram Log merupakan diagram berisi tabel-tabel yang digunakan untuk menyimpan *log* pengguna. Terdapat 3 jenis log yang digunakan yaitu log aktivitas, log gagal *login* dan log berhasil *login*. Masing-masing tabel pada diagram ini akan dijelaskan sebagai berikut.

3.2.3.6.1 Tabel Activity Log

Tabel Activity Log mencatat aktivitas yang dilakukan pengguna pada sistem. Tabel ini memiliki relasi dengan tabel User Account dan tabel OAuth Client. Tabel Activity Log dapat dilihat pada diagram OAuth2 pada Gambar 3.43 dan Gambar 3.44

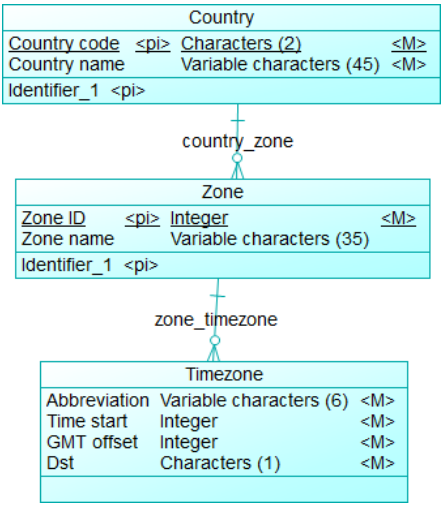
3.2.3.6.2 Tabel Failed Authentication

Tabel Failed Authentication mencatat pengguna yang gagal melakukan proses otentikasi atau gagal *login*. Setiap pengguna yang salah menginputkan *password* akan dicatat pada tabel ini. Tabel ini digunakan untuk penguncian akun pengguna (*lock user*) dan *suspend user*. Tabel ini memiliki relasi dengan tabel User Account. Tabel Failed Authentication dapat dilihat pada diagram OAuth2 pada Gambar 3.43 dan Gambar 3.44.

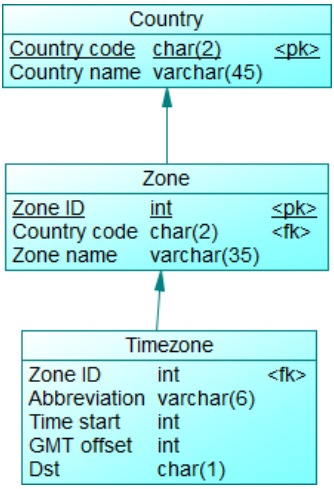
3.2.3.6.3 Tabel User Session

Tabel User Session mencatat pengguna yang berhasil melakukan *login* ke dalam sistem. Selain itu, pada tabel ini juga tedapat *field login_time* dan *logout_time* untuk mencatat waktu *login* dan *logout* pengguna. Tabel User Session memiliki relasi dengan tabel User Account. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.43 dan Gambar 3.44.

3.2.3.7 Diagram Timezone



Gambar 3.45 Diagram CDM Timezone



Gambar 3.46 Diagram PDM Timezone

Diagram Timezone merupakan diagram yang berisi tabel-tabel master yang mencatat zona waktu. Berikut penjelasan masing-masing tabel.

3.2.3.7.1 Tabel Country

Tabel Country merupakan tabel *master* yang menyimpan data seluruh negara. Pada tabel ini terdapat 2 *field,* yaitu *country_code* dan *country_name*. Tabel Country dapat dilihat pada diagram OAuth2 pada Gambar 3.45 dan Gambar 3.46**.**

3.2.3.7.2 Tabel Zone

Tabel Zone merupakan tabel yang menyimpan data zona dan negara yang berada pada zona tersebut. Tabel ini memiliki 3 *field*, yaitu *zone_id, country_code,* dan *zone_name* serta memiliki relasi dengan tabel Country. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.45 dan Gambar 3.46**.**

3.2.3.7.3 Tabel Timezone

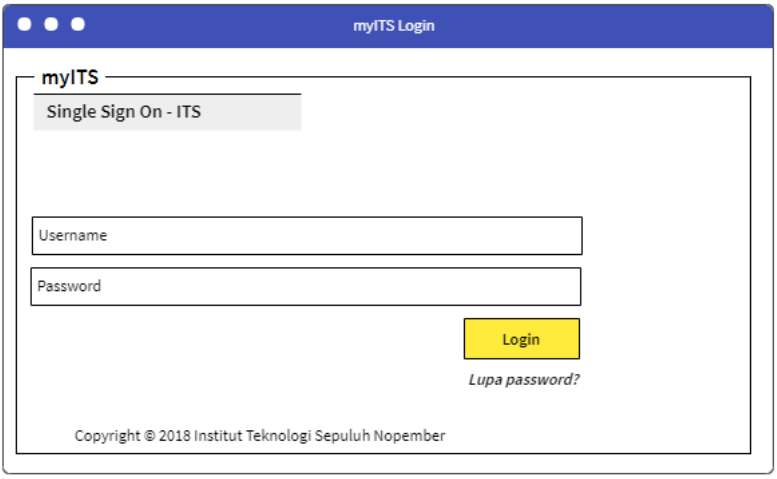
Tabel Timezone merupakan tabel yang menyimpan informasi mengenai waktu setiap zona. Pada tabel ini terdapat 5 *field* yaitu, *zone_id, abbreviation, time_start, gmt_offset,* dan *dst* serta memiliki relasi dengan tabel Zone. Tabel ini dapat dilihat pada diagram OAuth2 pada Gambar 3.45 dan Gambar 3.46**.**

3.2.4 Perancangan Antarmuka Pengguna

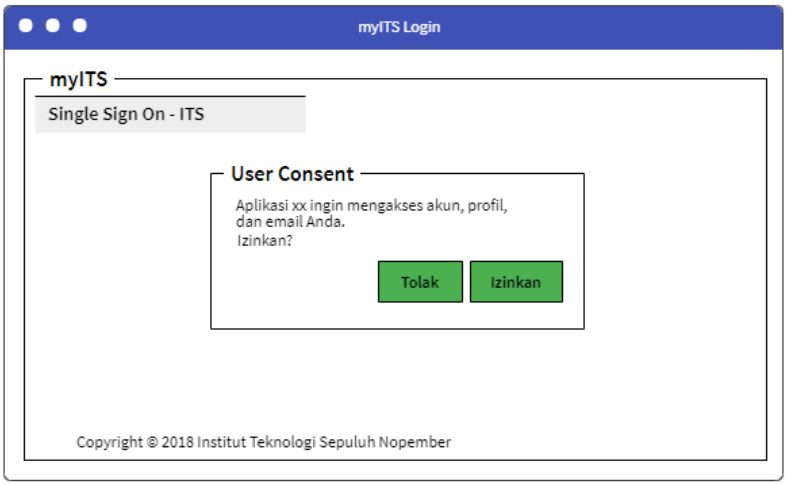
Perancangan antarmuka pengguna merupakan hal yang penting dalam melakukan perancangan aplikasi. Antarmuka pengguna yang berhubungan langsung dengan aktor harus memiliki kemudahan-kemudahan dan tampilan yang rapi serta menarik bagi penggunanya. Sistem memiliki beberapa antarmuka pengguna yang mana akan dijelaskan di bawah ini.

3.2.4.1 Rancangan Halaman Antarmuka *Login*

Halaman ini digunakan untuk kasus penggunaan *login* (UC-001). Pada halaman ini terdapat 2 isian yang harus diisi pengguna untuk melakukan *login*, yaitu isian *username* dan isian *password*. Untuk melakukan *login* pengguna menekan tombol *login*. Terdapat tombol lupa *password* yang akan mengarah pada halaman *reset password*. Rancangan halaman antarmuka dapat dilihat pada Gambar 3.47.



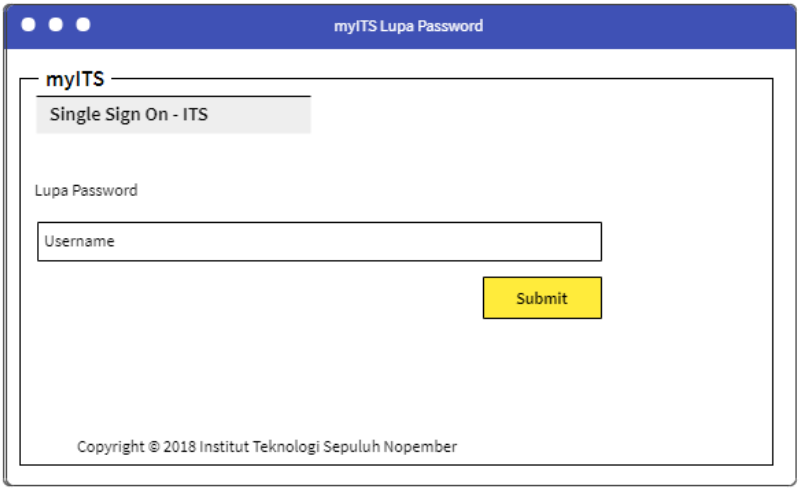
Gambar 3.47 Rancangan Halaman Antarmuka *Login*



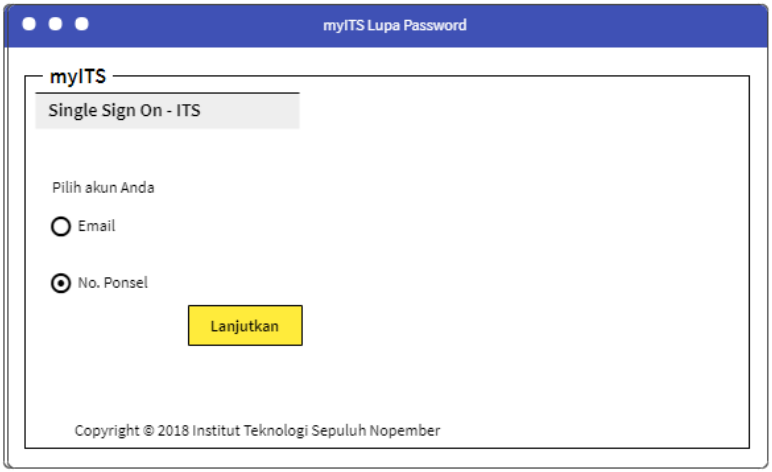
Gambar 3.48 Rancangan Halaman Antarmuka User Consent

3.2.4.2 Halaman Antarmuka Reset *Password*

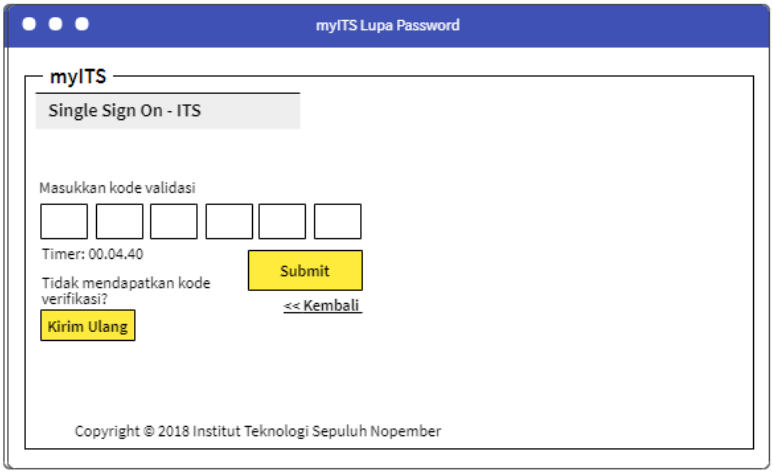
Halaman ini digunakan untuk kasus penggunaan Reset *Password* (UC-005). Pada halaman antarmuka reset *password* terdapat 4 halaman antarmuka untuk kasus penggunaan ini. Rancangan masing-masing halaman dapat dilihat pada Gambar 3.49, Gambar 3.50, Gambar 3.51, dan Gambar 3.52.



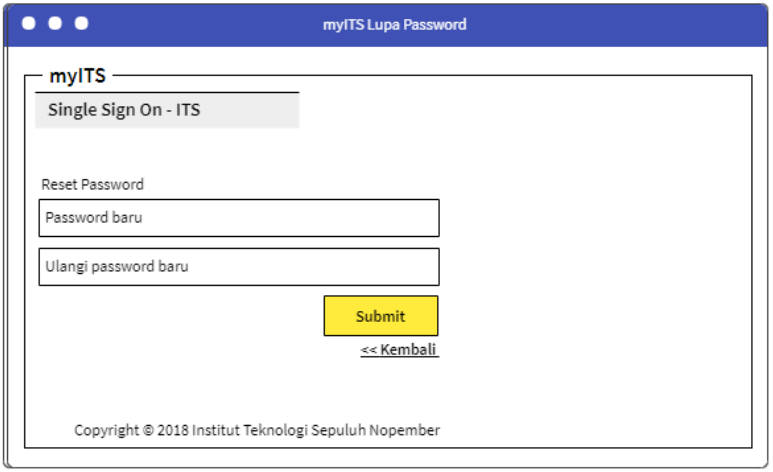
Gambar 3.49 Rancangan Halaman Antarmuka Lupa *Password* (1)



Gambar 3.50 Rancangan Halaman Antarmuka Lupa *Password* (2)



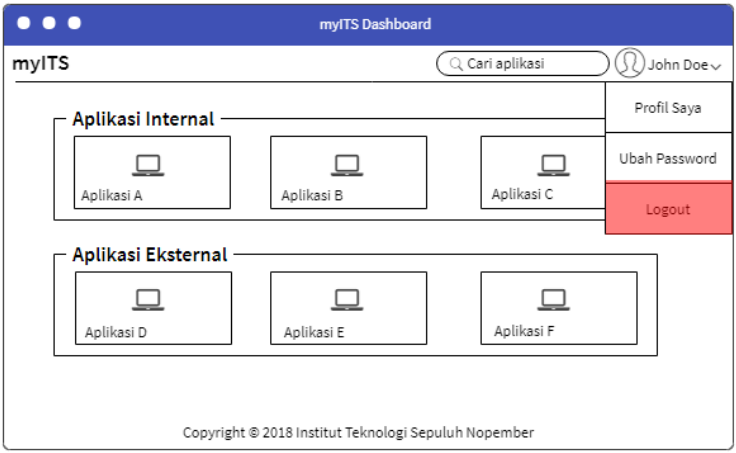
Gambar 3.51 Rancangan Halaman Antarmuka Lupa *Password* (3)



Gambar 3.52 Rancangan Halaman Antarmuka Lupa *Password* (4)

3.2.4.3 Halaman Antarmuka Dashboard myITS

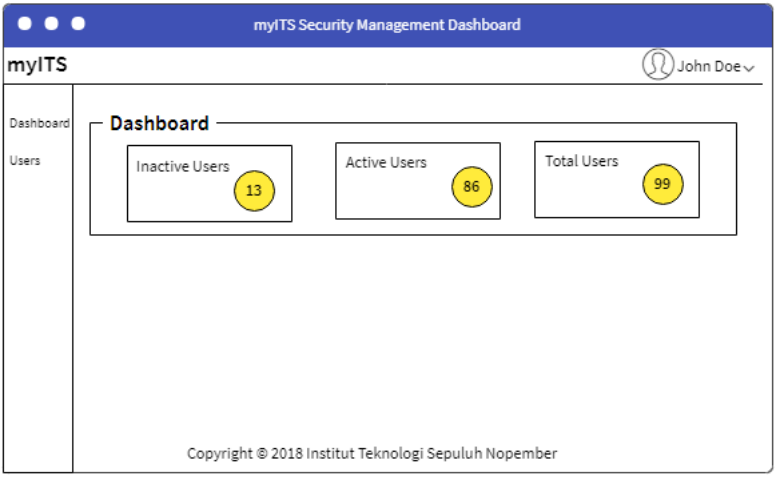
Halaman antarmuka dashboard myITS digunakan untuk kasus penggunaan Masuk ke Aplikasi (UC-002) dam *Logout* (UC-007). Rancangan halaman antarmuka dashboard myITS dapat dilihat pada Gambar 3.53.



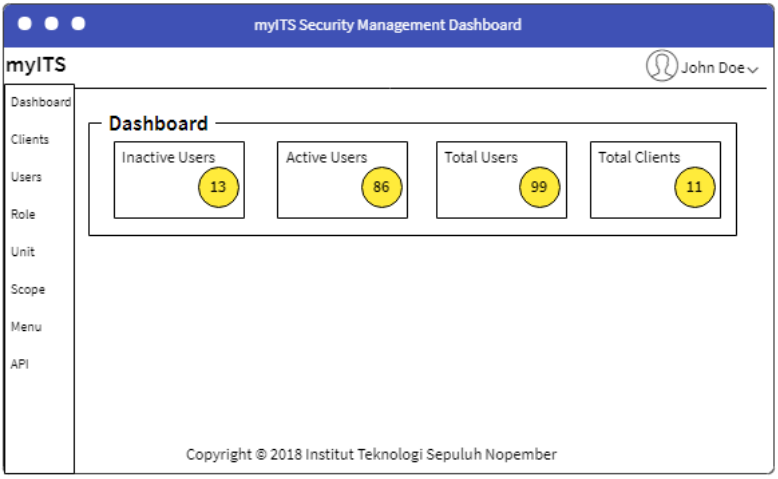
Gambar 3.53 Rancangan Halaman Dashboard myITS

3.2.4.4 Halaman Antarmuka Dashboard myITS Security Management

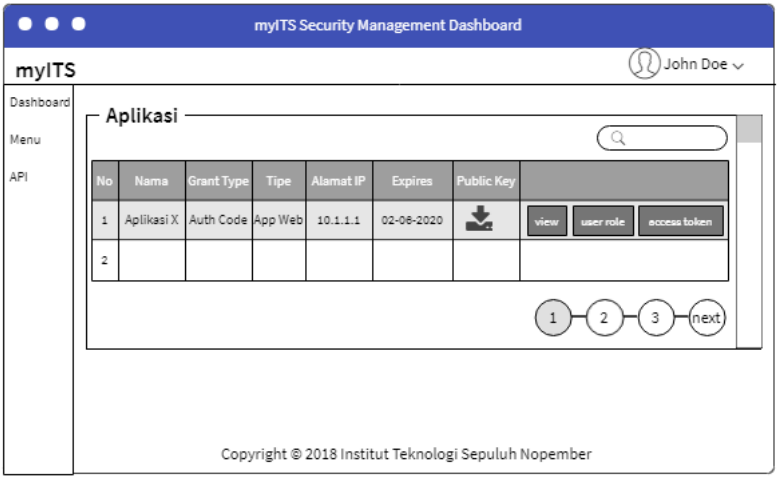
Halaman antarmuka dashboard myITS Security Management digunakan untuk kasus penggunaan Lihat Aplikasi (UC-008), Generate Access Token (UC-009), dan Unduh Public Key (UC-010) untuk pengguna dengan peran *developer*. Pengguna yang dapat masuk ke halaman ini adalah pengguna dengan peran *super admin*, *developer*, dan *helpdesk*. Setiap peran memiliki tampilan dashboard yang berbeda. Rancangan halaman-halaman dashboard myITS Security Management ditunjukan pada Gambar 3.54, Gambar 3.55, dan Gambar 3.56.



Gambar 3.54 Rancangan Halaman Dashboard myITS Security Management (Role *Helpdesk*)



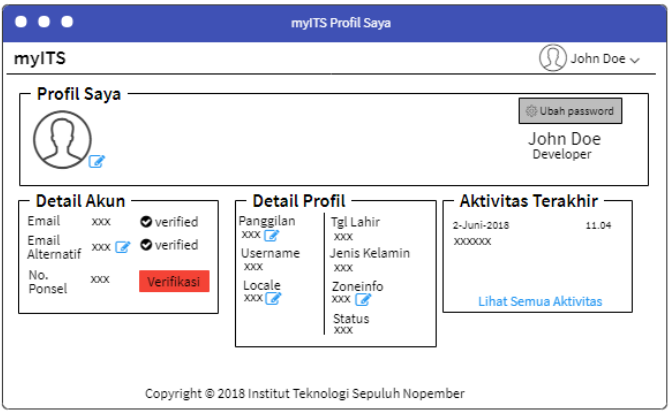
Gambar 3.55 Rancangan Halaman Dashboard myITS Security Management (Role *Super Admin*)



Gambar 3.56 Rancangan Halaman Dashboard myITS Security Management (Role *Developer*)

3.2.4.5 Halaman Antarmuka Profil Saya

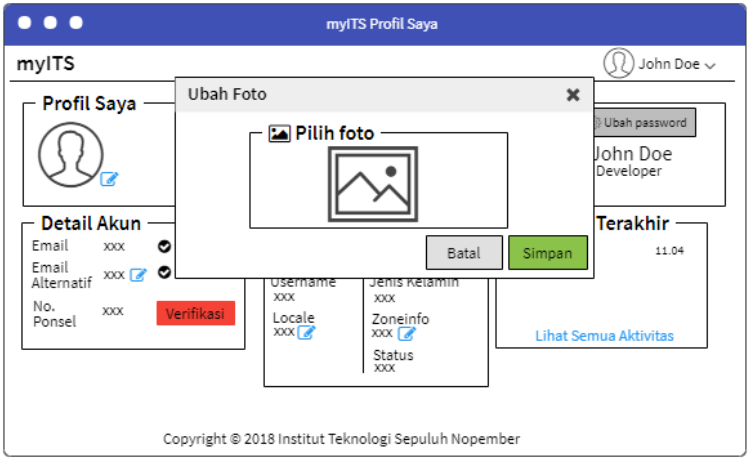
Halaman antarmuka Profil Saya digunakan untuk kasus penggunaan Lihat Profil (UC-003). Rancangan halaman Profil Saya dapat dilihat pada Gambar 3.57.



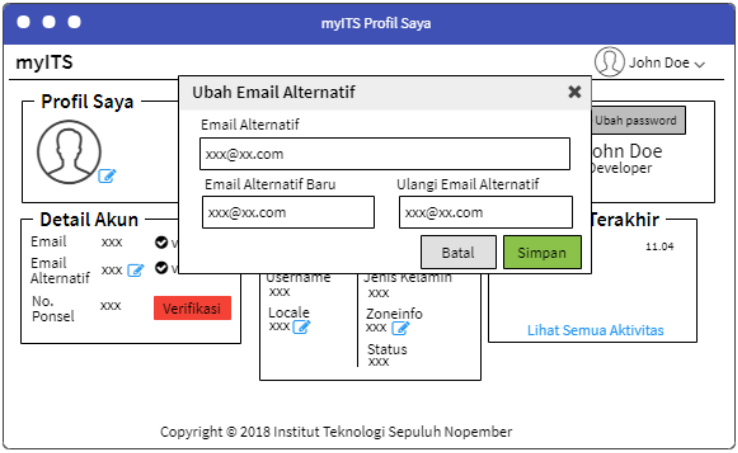
Gambar 3.57 Rancangan Halaman Profil Saya

3.2.4.6 Halaman Antarmuka Edit Profil

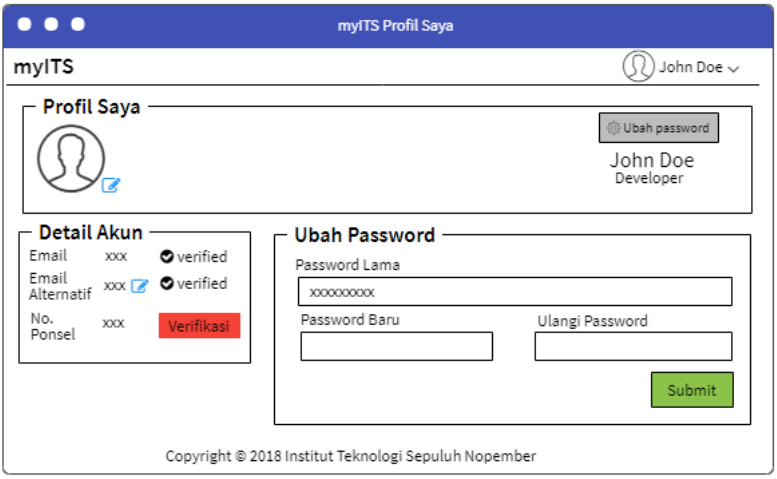
Halaman antarmuka Edit Profil digunakan untuk kasus penggunaan Edit Profil (UC-006). *End-user* dapat memperbaharui foto profil, email alternatif, nama panggilan, locale, dan zoneinfonya. Rancangan halaman Edit Profil dapat dilihat pada Gambar 3.58, Gambar 3.59 dan Gambar 3.60.



Gambar 3.58 Rancangan Halaman Edit Profil (1)



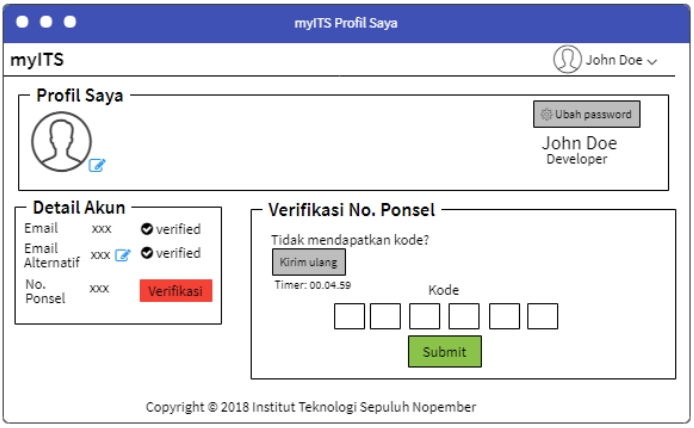
Gambar 3.59 Rancangan Halaman Edit Profil (2)



Gambar 3.60 Rancangan Halaman Ubah *Password*

3.2.4.7 Halaman Antarmuka Verifikasi Akun

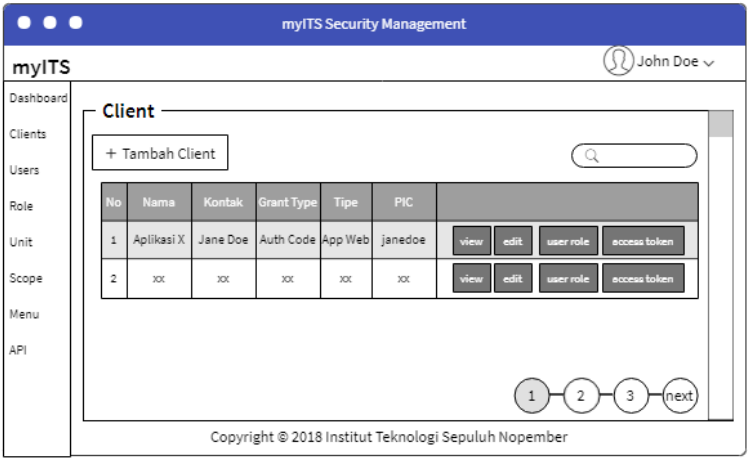
Halaman antarmuka Verifikasi Akun digunakan untuk kasus penggunaan Verifikasi Akun (UC-004). *End-user* dapat melakukan verifikasi terhadap akun email, email alternatif, maupun no. ponselnya yang belum terverifikasi. Rancangan halaman Verifikasi Akun dapat dilihat pada Gambar 3.61.



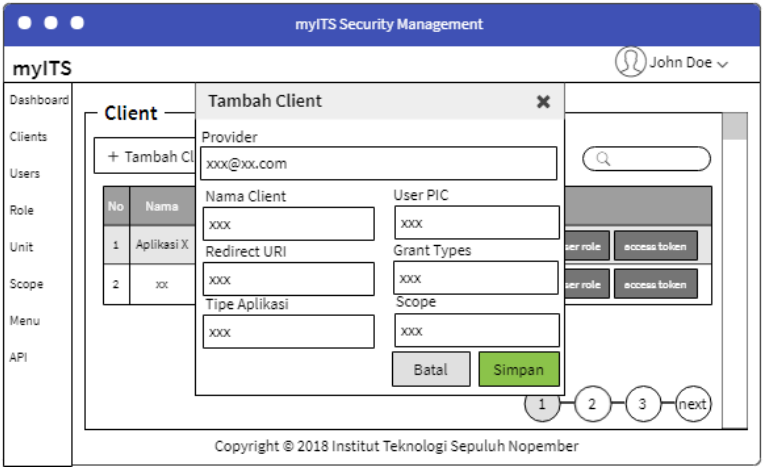
Gambar 3.61 Rancangan Halaman Verifikasi Akun

3.2.4.8 Halaman Antarmuka Kelola Client

Halaman antarmuka Kelola Client digunakan untuk kasus penggunaan Kelola Client (UC-021). Pengguna dengan hak akses dapat melakukan penambahan dan pembaharuan *client*. Rancangan halaman Kelola Client dapat dilihat pada Gambar 3.62 dan Gambar 3.63.



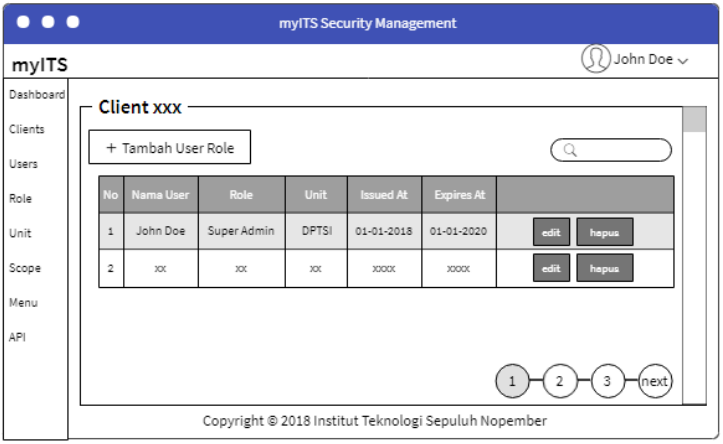
Gambar 3.62 Rancangan Halaman Kelola Client (1)



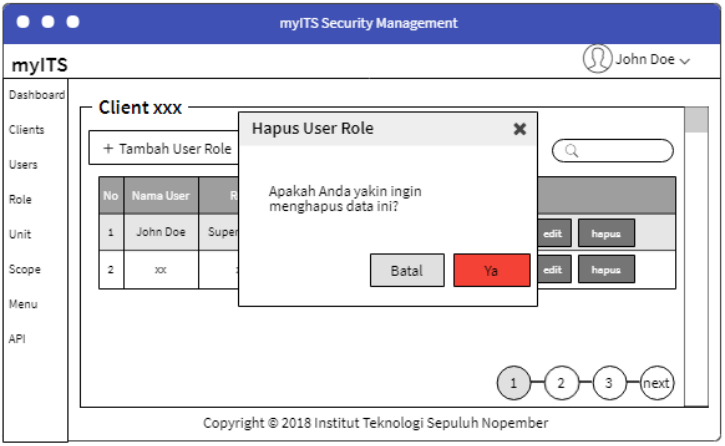
Gambar 3.63 Rancangan Halaman Kelola Client (2)

3.2.4.9 Halaman Antarmuka Atur User Role

Halaman antarmuka Atur User Role digunakan untuk kasus penggunaan Atur User Role (UC-011). Pengguna dengan hak akses dapat mengatur peran yang dimiliki tiap pengguna pada suatu *client* dengan menambahkan, memperbaharui, atau menghapus *user role*. Rancangan halaman antarmuka Atur User Role dapat dilihat pada Gambar 3.64 dan Gambar 3.65.



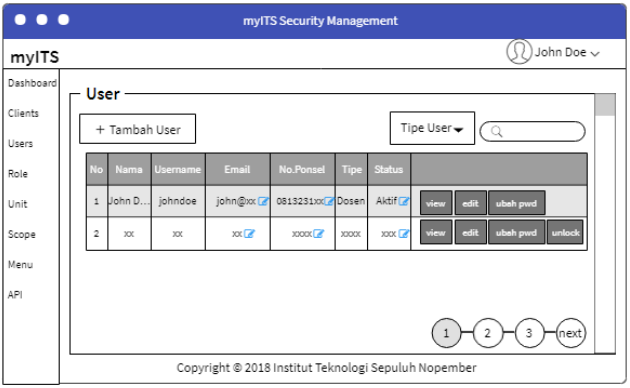
Gambar 3.64 Rancangan Halaman Atur User Role (1)



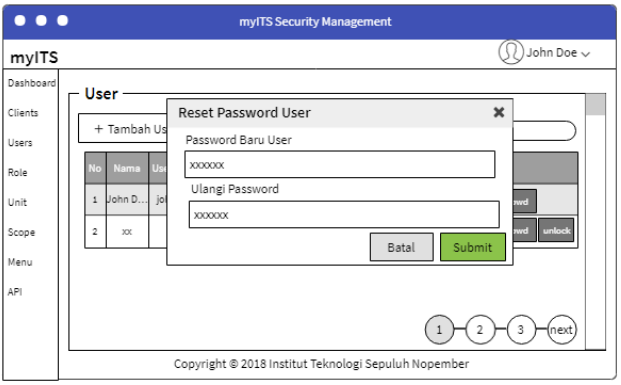
Gambar 3.65 Rancangan Halaman Atur User Role (2)

3.2.4.10 Halaman Antarmuka Kelola User

Halaman antarmuka Kelola User digunakan untuk kasus penggunaan Lihat User(UC-023), Kelola User (UC-017), Reset *Password* User (UC-026), Edit Kontak User (UC-024), dan Unlock User (UC-026). Pengguna dengan hak akses dapat menambah dan memperbaharui data user. Rancangan halaman antarmuka Kelola User dapat dilihat pada Gambar 3.66 dan Gambar 3.67.



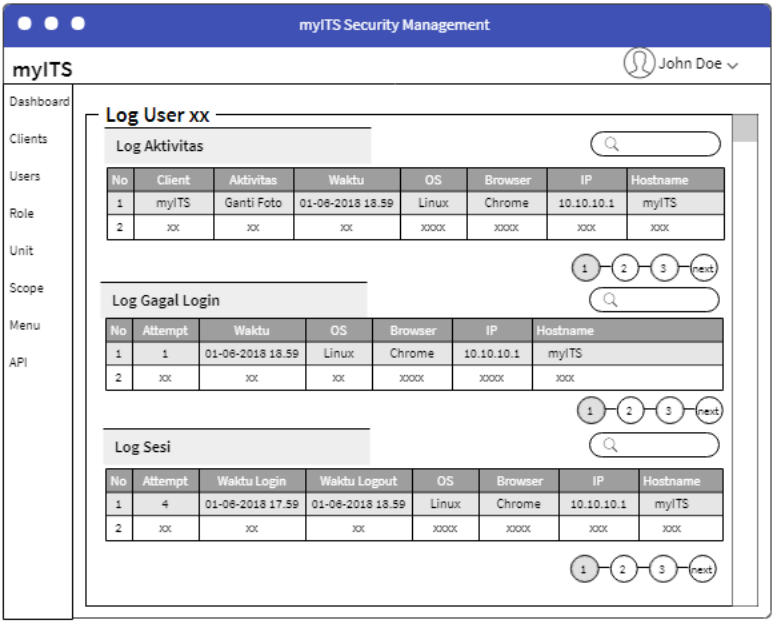
Gambar 3.66 Rancangan Halaman Kelola User



Gambar 3.67 Rancangan Halaman Reset *Password* User

3.2.4.11 Halaman Antarmuka Lihat Log User

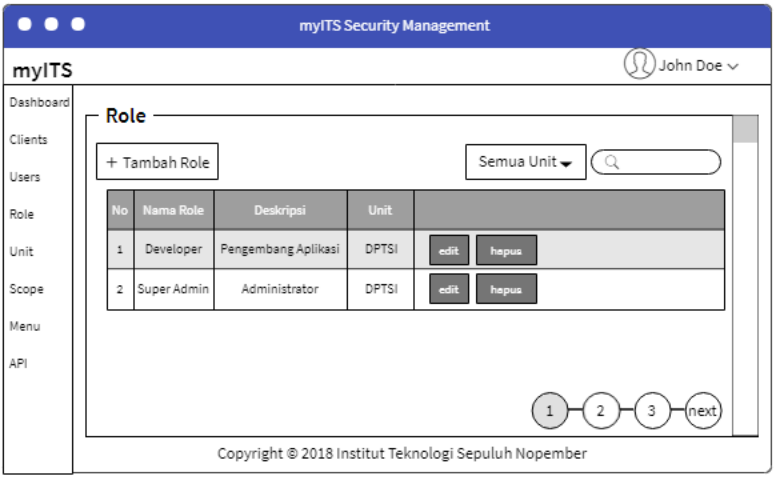
Halaman antarmuka Lihat Log User digunakan untuk kasus penggunaan Lihat Log User (UC-018). Pengguna dengan peran Super Admin dapat melihat log user. Rancangan halaman antarmuka Lihat Log User dapat dilihat pada Gambar 3.68.



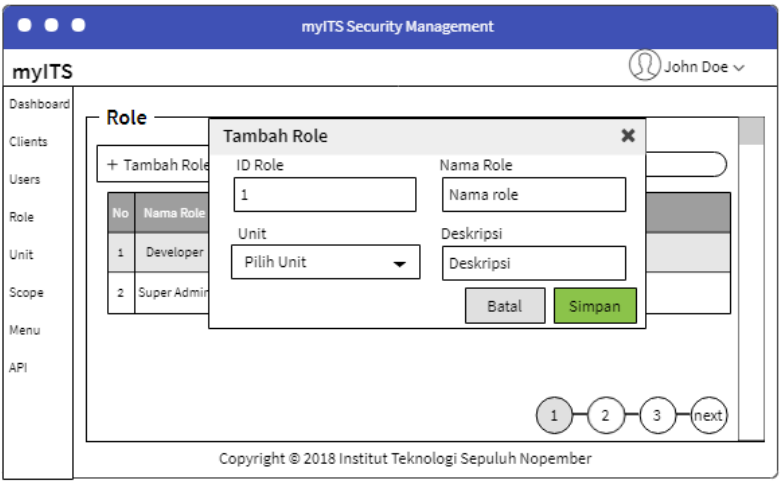
Gambar 3.68 Rancangan Halaman Log User

3.2.4.12 Halaman Antarmuka Kelola Role

Halaman antarmuka Kelola Role digunakan untuk kasus penggunaan Kelola Role (UC-020). Pengguna dengan peran Super Admin dapat menambahkan, memperbaharui, dan menghapus data role. Rancangan halaman Keloa Role dapat dilihat pada Gambar 3.69 dan Gambar 3.70.



Gambar 3.69 Rancangan Halaman Kelola Role (1)

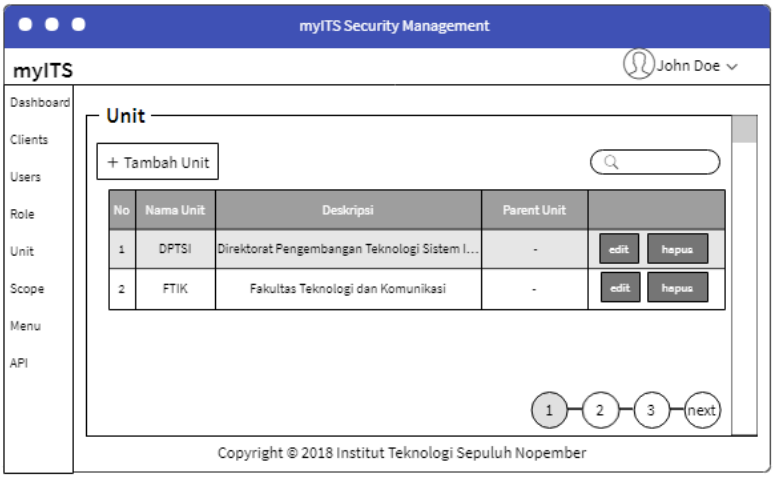


Gambar 3.70 Rancangan Halaman Kelola Role (2)

3.2.4.13 Halaman Antarmuka Kelola Unit

Halaman antarmuka Kelola Unit digunakan untuk kasus penggunaan Kelola Unit (UC-22). Pengguna dengan peran Super

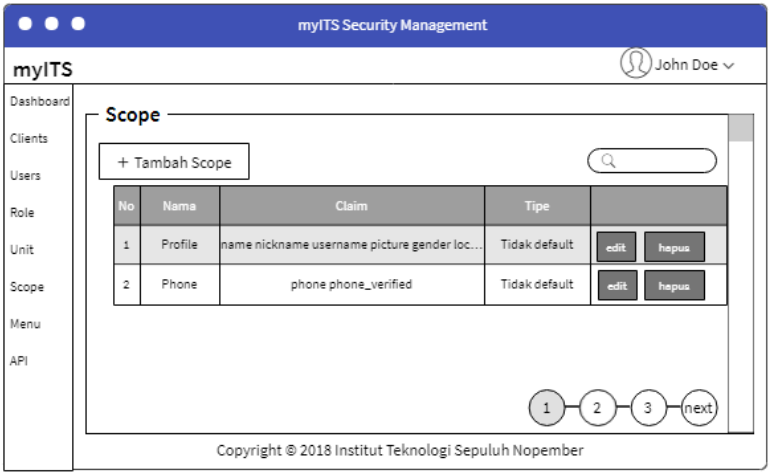
Admin dapat menambah, memperbaharui, dan menghapus data untit. Rancangan halaman antarmuka Kelola Unit dapat dilihat pada Gambar 3.71.



Gambar 3.71 Rancangan Halaman Kelola Unit

3.2.4.14 Halaman Antarmuka Kelola Scope

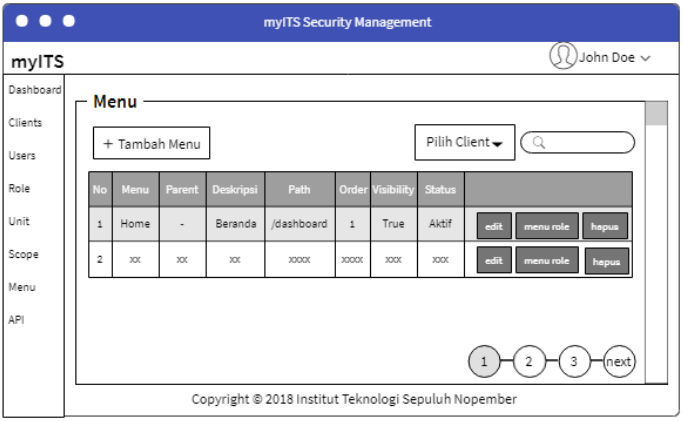
Halaman antarmuka Kelola Scope digunakan untuk kasus penggunaan Kelola Scope (UC-019. Pengguna dengan peran Super Admin dapat menambahkan, memperbaharui, dan menghapus data *scope*. Rancangan halaman antarmuka Kelola Scope dapat dilihat pada gambar Gambar 3.72.



Gambar 3.72 Rancangan Halaman Kelola Scope

3.2.4.15 Halaman Antarmuka Kelola Menu

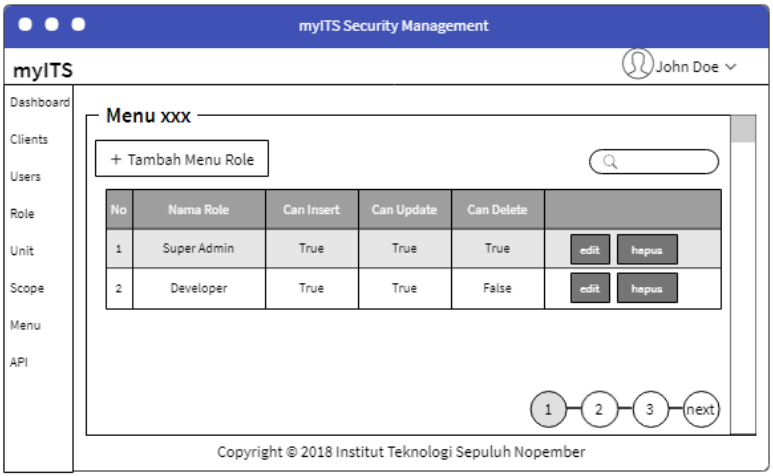
Halaman antarmuka Kelola Menu digunakan untuk kasus penggunaan Kelola Menu (UC-012). Pengguna denga hak akses dapat menambahkan, memperbaharui, dan menghapus data menu. Rancangan halaman antarmuka Kelola Menu dapat dilihat pada Gambar 3.73.



Gambar 3.73 Rancangan Halaman Kelola Menu

3.2.4.16 Halaman Antarmuka Atur Menu Role

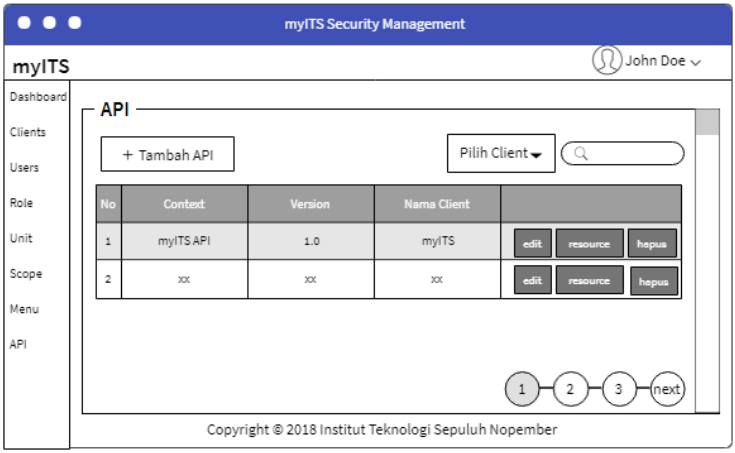
Halaman antarmuka Kelola Menu Role digunakan untuk kasus penggunaan Kelola Menu Role (UC-013). Pengguna denga hak akses dapat menambahkan, memperbaharui, dan menghapus data menu role atau data peran yang dapat mengakses *menu*. Rancangan halaman antarmuka Kelola Menu Role dapat dilihat pada Gambar 3.74.



Gambar 3.74 Rancangan Halaman Kelola Menu Role

3.2.4.17 Halaman Antarmuka Kelola API

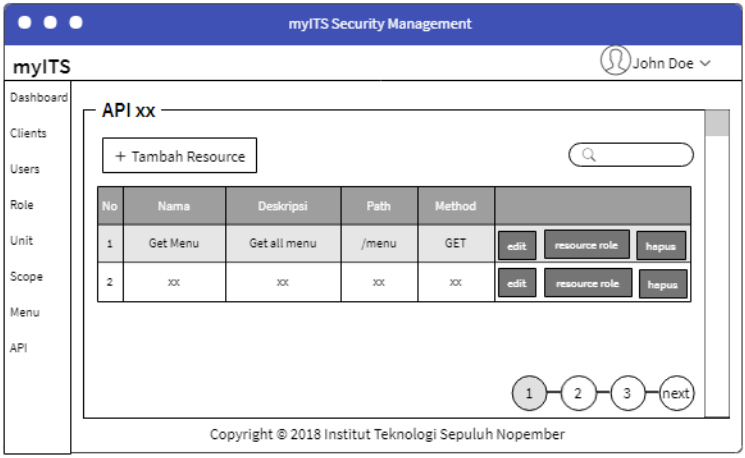
Halaman antarmuka Kelola API digunakan untuk kasus penggunaan Kelola API (UC-014). Pengguna denga hak akses dapat menambahkan, memperbaharui, dan menghapus data API. Rancangan halaman antarmuka Kelola API dapat dilihat pada Gambar 3.75.



Gambar 3.75 Rancangan Halaman Kelola API

3.2.4.18 Halaman Antarmuka Kelola Resource

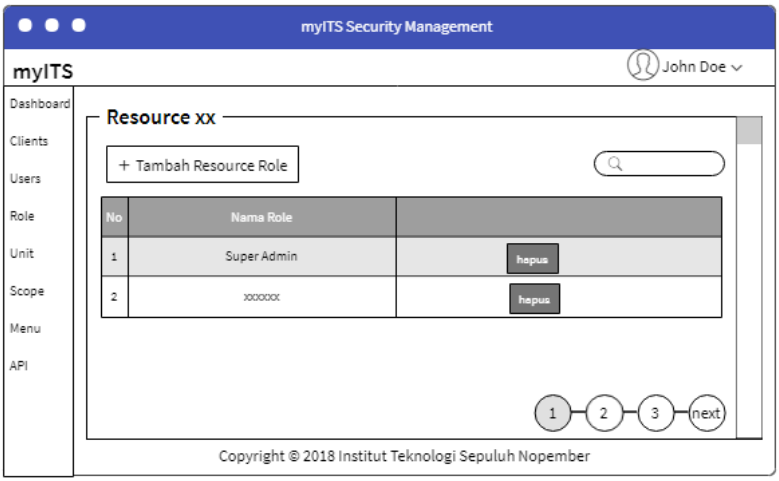
Halaman antarmuka Kelola Resource digunakan untuk kasus penggunaan Kelola Resource (UC-015). Pengguna denga hak akses dapat menambahkan, memperbaharui, dan menghapus data resource. Rancangan halaman antarmuka Kelola Resource dapat dilihat pada Gambar 3.76.



Gambar 3.76 Rancangan Halaman Kelola Resource

3.2.4.19 Halaman Antarmuka Atur Resource Role

Halaman antarmuka Kelola Resource Role digunakan untuk kasus penggunaan Kelola Resoure Role (UC-016). Pengguna denga hak akses dapat menambahkan, memperbaharui, dan menghapus data menu. Rancangan halaman antarmuka Kelola Menu dapat dilihat pada Gambar 3.77.



Gambar 3.77 Rancangan Halaman Atur Resource Role

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sistem sesuai dengan analisis dan perancangan proses bisnis secara umum pada sistem *Single Sign On* myITS yang dijabarkan pada bab sebelumnya.

Implementasi yang akan dijelaskan meliputi lingkungan pembangunan sistem atau perangkat lunak, kode sumber utama berisi *pseudocode*, implementasi antarmuka perangkat lunak, dan implementasi client. Bahasa pemrograman yang digunakan adalah PHP dengan arsitektur sistem MVC dan dengan menggunakan kerangka kerja Phalcon.

4.1 Lingkungan Implementasi

Berikut kakas bantu yang digunakan pada proses implementasi perangkat lunak ini:

1. Windows 10 Pro 64 bit sebagai sistem operasi
2. JetBrains PhpStrom 2017.1.4 sebagai *Integrated Development Environment* (IDE)
3. Phalcon 3.3.1 sebagai kerangka kerja (*framework)*
4. PHP 7.1.13 sebagai bahasa pemrograman yang digunakan
5. SQL Server 2017 dan Navicat Premium 12 sebagai sistem manajemen basis data
6. Apache 2.4.29 sebagai *web server*

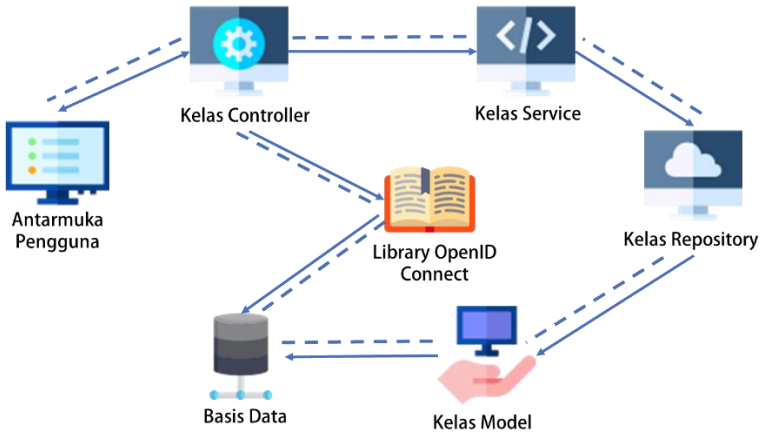
4.2 Implementasi *Model-View-Controller*

Sistem yang dibuat memiliki lapisan-lapisan yang direpresentasikan dalam kelas, yaitu view sebagai lapisan antarmuka pengguna, *controller* sebagai tempat untuk menerima *request* yang dikirim oleh aplikasi *client* dan mengirim balik *response*, s*ervice* sebagai tempat pemrosesan data komputasi,

repository sebagai tempat untuk melakukan pengelolaan terhadap basis data dan *model* sebagai representasi dari setiap tabel di basis data.

Implementasi MVC pada sistem aplikasi dilakukan dengan pengadaan folder atau *package controller* yang berisikan kelas-kelas *controller, package service* yang berisikan kelas-kelas *service*, *package repository* yang berisikan kelas-kelas *repository*, dan *package model* yang berisikan representasi tabel basis data.

Sistem yang dibangun mengimplementasikan protokol otentikasi dan otorisasi dengan menggunakan *library* OAuth2 Server Bshaffer. Implementasi arsitektur sistem ditunjukan pada Gambar 4.1.



Gambar 4.1 Implementasi Arsitektur Sistem

4.2.1 Implementasi Kelas *Model*

Implementasi model dalam aplikasi berada pada *package* atau folder *Models* dimana terdapat kelas-kelas model yang masing-masing merepresentasikan tabel dan kolom.

4.2.1.1 Model Access Token

Model Access Token merepresentasikan tabel OAuth Access Token pada basis data. Pada kelas model berisi nama variabel *field* dan sebuah fungsi getSource(). Penggunaan kelas ini terdapat pada kasus penggunaan *Login* (UC-001) dan Generate Access Token (UC-009). Berikut adalah kode sumber kelas model Access Token. Kode model ini dapat dilihat pada Kode Sumber 3.1.

```
1.  <?php
2.
3.  namespace Its\Oidc\Models;
4.
5.  use Phalcon\Mvc\Model;
6.  use Phalcon\Validation;
7.
8.  class AccessToken extends Model
9.  {
10.     public $access_token;
11.     public $client_id;
12.     public $user_id;
13.     public $expires;
14.     public $scope;
15.
16.     public function getSource()
17.     {
18.         return 'oauth_access_token';
19.     }
20. }
```

Kode Sumber 4.1 Model AccessToken.php

4.2.1.2 Model Activity Log

Model Activity Log merepresentasikan tabel Activity Log pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Lihat Log User (UC-018).

4.2.1.3 Model Api

Model Api merepresentasikan tabel API pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola API (UC-014).

4.2.1.4 Model Asymmetric Key

Model Asymmetric Key merepresentasikan tabel OAuth Asymmetric Key pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Unduh Public Key (UC-010) dan Kelola Client (UC-021) .

4.2.1.5 Model Client

Model Client merepresentasikan tabel OAuth Client pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Client (UC-021) dan Lihat Aplikasi (UC-008).

4.2.1.6 Model Code

Model Code merepresentasikan tabel Verification Code pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Verifikasi Akun (UC-004) dan Reset *Password* (UC-005).

4.2.1.7 Model Cookie

Model Cookie merepresentasikan tabel Cookie pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan *Login* (UC-001) dan *Logout* (UC-007).

4.2.1.8 Model Failed Auth

Model Failed Auth merepresentasikan tabel Failed Auth pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan *Login* (UC-001) dan Lihat Log User (UC-018).

4.2.1.9 Model Menu

Model Menu merepresentasikan tabel Menu pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Menu (UC-012) dan Atur Menu Role (UC-013).

4.2.1.10 Model Menu Role

Model Menu Role merepresentasikan tabel Menu Role pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Atur Menu Role (UC-013).

4.2.1.11 Model Provider

Model Provider merepresentasikan tabel Provider pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Client (UC-021).

4.2.1.12 Model Refresh Token

Model Refresh Token merepresentasikan tabel OAuth Refresh Token pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan *Login* (UC-001).

4.2.1.13 Model Resource

Model Resource merepresentasikan tabel Resource pada basis data. Penggunaan kelas model ini terdapat pada kasus

penggunaan Kelola Resource (UC-015) dan Atur Resource Role (UC-016).

4.2.1.14 Model Resource Role

Model Resource Role merepresentasikan tabel Resource Role pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Atur Resource Role (UC-016).

4.2.1.15 Model Role

Model Role merepresentasikan tabel Role pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Role (UC-020), Atur User Role (UC-011), Atur Menu Role (UC-013), dan Atur Resource Role (UC-016).

4.2.1.16 Model Scope

Model Scope merepresentasikan tabel OAuth Scope pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Scope (UC-018), dan Kelola Client (UC-021).

4.2.1.17 Model Unit

Model Unit merepresentasikan tabel Unit pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Unit (UC-022), dan Atur User Role (UC-011).

4.2.1.18 Model User

Model User merepresentasikan tabel User Account pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola User (UC-017), Lihat User (UC-023), Kelola Client (UC-021), dan Atur User Role (UC-011).

4.2.1.19 Model User Consent

Model User Consent merepresentasikan tabel OAuth User Consent pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan *Login* (UC-001).

4.2.1.20 Model User Role

Model User Role merepresentasikan tabel User Role pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Atur User Role (UC-011).

4.2.1.21 Model User Session

Model User Session merepresentasikan tabel User Session pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan *Login* (UC-001) dan Lihat Log User (UC-018).

4.2.1.22 Model User Type

Model User Type merepresentasikan tabel User Type pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola User (UC-017).

4.2.1.23 Model Zone

Model Zone merepresentasikan tabel Zone pada basis data. Penggunaan kelas model ini terdapat pada kasus penggunaan Kelola Client (UC-021) dan Edit Profil (UC-006).

4.2.2 Implementasi Kelas *Repository*

Kelas *repository* berisikan fungsi-fungsi yang digunakan untuk mengambil, memperbaharui, dan memasukkan data baru ke dalam basis data. Kelas-kelas *repository* berada dalam *package*

atau folder *Repositories*. Pada subbab ini, akan dijelaskan fungsi-fungsi umum pada kelas *repository*.

4.2.2.1 Fungsi GetAll

Fungsi *getAll* dipanggil hampir pada seluruh kelas *service*. Fungsi ini berfungsi untuk mengambil seluruh data pada tabel tertentu. Ketika dipanggil, fungsi *getAll* mengembalikan *list* berupa *array* dari objek. *Pseudocode* untuk fungsi ini dapat dilihat pada Kode Sumber 4.2.

```
1.  public function getAll()
2.  {
3.      CALL model with method find()
4.      RETURN value of the model
5.  }
```

Kode Sumber 4.2 Pseudocode fungsi getAll kelas *repository*

4.2.2.2 Fungsi GetById

Fungsi getById digunakan untuk mengambil data pada tabel yang memiliki *id* bernilai *idValue* yang dikirimkan oleh kelas *service*. Fungsi mengembalikan berupa satu objek kelas model. *Pseudocode* untuk fungsi ini dapat dilihat pada Kode Sumber 4.3.

```
1.  public function getById(idValue)
2.  {
3.      CALL model with method findFirst(id = idValue)
4.      RETURN value of the model
5.  }
```

Kode Sumber 4.3 Pseudocode fungsi getById kelas *repository*

4.2.2.3 Fungsi Insert

Fungsi insert digunakan untuk menambah *record* pada tabel. Fungsi menerima parameter *input* berupa objek kelas model dan menambahkan pada tabel menggunakan *method save()*. Fungsi mengembalikan nilai *boolean*, *true* atau *false*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.4.

```
1.  public function insert(Model modelName)
2.  {
3.      resultValue = CALL modelName with method save()
4.      RETURN the resultValue
5.  }
```

Kode Sumber 4.4 Pseudocode fungsi insert kelas *repository*

4.2.2.4 Fungsi Update

Fungsi update digunakan untuk memperbaharui *record* pada tabel. Fungsi menerima parameter *input* berupa objek kelas model dan memperbaharui *record* pada tabel menggunakan *method update()*. Fungsi mengembalikan nilai *boolean*, *true* atau *false*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.5.

```
1.  public function update(Model modelName)
2.  {
3.      resultValue=CALL modelName with method update()
4.      RETURN the resultValue
5.  }
```

Kode Sumber 4.5 Pseudocode fungsi update kelas *repository*

4.2.2.5 Fungsi Delete

Fungsi delete digunakan untuk menghapus *record* pada tabel. Fungsi menerima parameter *input* berupa objek kelas model dan menghapus *record* pada tabel menggunakan *method delete()*. Fungsi mengembalikan nilai *boolean*, *true* atau *false*. *Pseudocode* dapat dilihat pada Kode Sumber 4.6

```
1.  public function delete(Model modelName)
2.  {
3.      resultValue=CALL modelName with method delete()
4.      RETURN the resultValue
5.  }
```

Kode Sumber 4.6 Pseudocode fungsi delete kelas *repository*

4.2.3 Implementasi Kelas *Service*

Kelas *service* berisikan fungsi-fungsi yang digunakan untuk pemrosesan data dan komputasi. Pengolahan data yang dikirim kelas *repository* diambil sesuai dengan kebutuhan kasus penggunaan. Kelas *service* menjadi penghubung antara kelas *repository* dengan kelas *controller*. Kelas-kelas *service* berada pada *package* atau folder *Services*. Pada subbab ini akan dijelaskan fungsi-fungsi umum pada kelas *service*.

4.2.3.1 Fungsi GetAll

Fungsi getAll pada kelas *service* digunakan untuk memanggil fungsi getAll pada kelas *repository*. Fungsi ini dipanggil dari kelas *controller*. Fungsi akan mengembalikan nilai yang didapat dari pemanggilan kelas *repository*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.7.

```
1.  public function getAll()
2.  {
3.      CALL class repository and its method getAll()
```

```
4.      RETURN value of called method
5.  }
```

Kode Sumber 4.7 Pseudocode fungsi getAll kelas *service*

4.2.3.2 Fungsi GetById

Fungsi getById pada kelas *service* digunakan untuk memanggil fungsi getById pada kelas *repository*. Fungsi ini dipanggil dari kelas *controller*. Fungsi akan mengembalikan nilai yang didapat dari pemanggilan kelas *repository*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.8

```
1.  public function getById(idValue)
2.  {
3.      CALL class repository and its method getById(idValue)
4.      RETURN value of called method
5.  }
```

Kode Sumber 4.8 Pseudocode fungsi getById kelas *service*

4.2.3.3 Fungsi Insert

Fungsi insert pada kelas *service* digunakan untuk memanggil fungsi insert pada kelas *repository* dengan parameter kelas model. Fungsi ini dipanggil dari kelas *controller* dengan parameter objek *request* yang berisi nilai yang ingin disimpan. Fungsi akan mengembalikan nilai yang didapat dari pemanggilan kelas *repository*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.9.

```
1.  public function insert(Request request)
2.  {
3.      CREATE new class model
4.      SAVE request value in model
5.      resultValue = CALL class repository and its
        method insert(model)
```

```
6.      RETURN the resultValue
7.  }
```

Kode Sumber 4.9 Pseudocode fungsi insert kelas *service*

4.2.3.4 Fungsi Update

Fungsi update pada kelas *service* digunakan untuk memanggil fungsi insert pada kelas *repository* dengan parameter kelas model. Fungsi ini dipanggil dari kelas *controller* dengan parameter objek *request* yang berisi nilai yang ingin disimpan. Fungsi akan mengembalikan nilai yang didapat dari pemanggilan kelas *repository*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.10

```
1.  public function update(Request request)
2.  {
3.      CREATE new class model
4.      SAVE request value in model
5.      resultValue = CALL class repository and its
        method update(model)
6.      RETURN the resultValue
7.  }
```

Kode Sumber 4.10 Pseudocode fungsi update kelas *service*

4.2.3.5 Fungsi Delete

Fungsi delete pada kelas *service* digunakan untuk memanggil fungsi delete pada kelas *repository* dengan parameter kelas model. Model yang dikirimkan didapat dari hasil pemanggilan fungsi getById pada kelas *service*. Fungsi ini dipanggil dari kelas *controller* dengan parameter nilai *id* model yang ingin dihapus. Fungsi akan mengembalikan nilai yang didapat dari pemanggilan kelas *repository*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.11

```
1.  public function delete(idValue)
2.  {
3.      model = CALL method getById(idValue) in class
        service
4.      resultValue = CALL class repository and its
        method delete(model)
5.      RETURN the resultValue
6.  }
```

Kode Sumber 4.11 Pseudocode fungsi delete kelas *service*

4.2.4 Implementasi Kelas *Controller*

Kelas *controller* berisikan fungsi-fungsi yang digunakan untuk menangkap *request* dan mengirimkan *response* baik berupa string JSON kepada *client* yang mengirimkan *response* maupun berupa tampilan antarmuka dengan memproses terlebih dahulu di dalam kelas *controller* itu sendiri atau pada kelas *service*. Kelas-kelas *controller* berada pada *package* atau folder *Controllers*. Pada subbab ini akan dijelaskan fungsi-fungsi umum pada kelas *controller* dan tiga *controller* untuk masing-masing *endpoint*, yaitu *authorize endpoint*, *token endpoint*, dan *userinfo endpoint*.

4.2.4.1 Fungsi Index

Fungsi *index* pada kelas *controller* digunakan untuk menampilkan *view* dan memanggil kelas *service* untuk mendapatkan data yang akan ditampilkan. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.12

```
1.  public function index()
2.  {
3.      GET userId value from userSession
4.      userInfoValue = CALL class service and its
        method getById(userId)
5.      dataValue = CALL calss service and its method
        getAll()
6.      SET userInfoValue & dataValue to view
```

```
7.      RETURN view
8.  }
```

Kode Sumber 4.12 Pseudocode fungsi index kelas *controller*

4.2.4.2 Fungsi Insert

Fungsi *insert* pada kelas *controller* digunakan untuk menyimpan data dengan memanggil kelas *service*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.13.

```
1.  public function insert()
2.  {
3.      GET request object
4.      resultValue = CALL class service and its method
        insert(request)
5.      IF resultValue is true THEN
6.          SET alertMessage
7.      ELSE
8.          SET successMessage
9.      ENDIF
10.     RETURN view
11. }
```

Kode Sumber 4.13 Pseudocode fungsi insert kelas *controller*

4.2.4.3 Fungsi Update

Fungsi *update* pada kelas *controller* digunakan untuk menyimpan perubahan data dengan memanggil kelas *service*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.14.

```
1.  public function update() {
2.      GET request object
3.      resultValue = CALL class service and its method
        update(request)
4.      IF resultValue is true THEN
5.          SET alertMessage
6.      ELSE
```

```
7.          SET successMessage
8.      ENDIF
9.      RETURN view }
```

Kode Sumber 4.14 Pseudocode fungsi update kelas *controller*

4.2.4.4 Fungsi Delete

Fungsi *delete* pada kelas *controller* digunakan untuk menghapus data dengan memanggil kelas *service*. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 4.15.

```
1.  public function delete()
2.  {
3.      GET idValue from request object
4.      resultValue = CALL class service and its method
        delete(idValue)
5.      IF resultValue is true THEN
6.          SET alertMessage
7.      ELSE
8.          SET successMessage
9.      ENDIF
10.     RETURN view
11. }
```

Kode Sumber 4.15 Pseudocode fungsi delete kelas *controller*

4.2.4.5 Controller Authorize

Kelas *AuthorizeController* digunakan untuk menangkap dan menangani *request* yang dikirimkan ke *authorize endpoint* sistem. *Pseudocode* kelas ini dapat dilihat pada Kode Sumber 4.16 dan Kode Sumber 4.17.

```
1.  public function authorize()
2.  {
3.      GET authorizationRequest from request object
4.      check authorizationRequest
5.      IF not valid THEN
```

```
6.          SEND error message
7.      ENDIF
8.      IF prompt = login THEN
9.          IF has credential THEN
10.             IF attemptLogin(request) is true THEN
11.                 isAuthorized is true
12.             ELSE
13.                 isAuthorized is false
14.             ENDIF
15.         ELSE
16.             RETURN to login page
17.         ENDIF
18.     ELSEIF prompt = none THEN
19.         IF user has no session THEN
20.             SEND not authorize message
21.         ELSE
22.             isAuthorized is true
23.             GET idUser from session
24.         ENDIF
25.     ELSE
26.         CALL attemptLogin(request)
27.     ENDIF
28.     IF isAuthorized is true THEN
29.         SET user session
30.     ENDIF
31.
32.     IF client is trusted THEN
33.         RETURN CALL handleAuthorizedRequest(request
            Request) function from library
34.     ELSE
35.         IF isAllowed false THEN
36.             RETURN user consent page
37.         ELSE
38.             RETURN CALL handleAuthorizedRequest(req
                uestRequest) function from library
39.         ENDIF
40.     ENDIF
41. }
```

Kode Sumber 4.16 Pseudocode *authorizeController* fungsi *authorize()*

```
1.  public function attemptLogin(request)
2.  {
3.      IF has credential is true THEN
4.          IF checkLoginStatus() is false THEN
5.              RETURN false
6.          ELSE
7.              IF login success THEN
8.                  RETURN true
9.              ELSE
10.                 RETURN false
11.             ENDIF
12.         ENDIF
13.     ELSE
14.         RETURN login page
15.     ENDIF
16. }
```

Kode Sumber 4.17 Pseudocode *authorizeController* fungsi *attemptLogin()*

4.2.4.6 Controller Token

Kelas *TokenController* digunakan untuk menangkap dan menangani *request* yang dikirimkan ke *token endpoint* sistem. *Pseudocode* kelas ini dapat dilihat pada Kode Sumber 4.18.

```
1.  public function handle()
2.  {
3.      GET tokenRequest from request object
4.      CREATE new response
5.       CALL handleTokenRequest(request, response) from
         library
6.      RETURN response
7.  }
```

Kode Sumber 4.18 Pseudocode *tokenController*

4.2.4.7 Controller UserInfo

Kelas *UserInfoController* digunakan untuk menangkap dan menangani *request* yang dikirimkan ke *userinfo endpoint* sistem. *Pseudocode* kelas ini dapat dilihat pada Kode Sumber 4.19.

```
1.  public function handle()
2.  {
3.      GET userinfoRequest from request object
4.      CREATE new response
5.      CALL handleUserInfoRequest(request, response)
        from library
6.      RETURN response
7.  }
```

Kode Sumber 4.19 Pseudocode *userInfoController*

4.2.5 Implementasi *Library*

Implementasi protokol otentikasi dan otorisasi OpenID Connect dilakukan menggunakan *library* OAuth2 Server Bshaffer dengan memodifikasi bagian agar sesuai dengan kebutuhan sistem. Bagian yang dimodifikasi adalah pada kelas Pdo.php pada *package Storage.* Berikut adalah beberapa bagian *library* yang dimodifikasi pada kelas Pdo.php.

4.2.5.1 Fungsi GetScope

Pada kelas Pdo.php sebelumnya nilai *scope* dan *claim* didefinisikan secara statis pada kelas UserClaimInterface. Sesuai kebutuhan sistem dimana nilai *scope* dan *claim* dapat ditambahkan dan dimodifikasi secara dinamis maka ditambahakan sebuah fungsi baru yaitu fungsi getScope untuk mendapatkan nilai *scope* dan *claim*nya dari *database*. Kode sumber dapat dilihat pada Kode Sumber 4.20.

```
1.  public function getScope() {
```

```
2.        $stmt = $this->db->prepare ( $sql = sprintf
          ("SELECT * FROM %s",$this->config
          ['scope_table']));
3.        $stmt->execute();
4.        if (!$scopes = $stmt->fetchAll
          (\PDO::FETCH_ASSOC)) {
5.            return false;
6.        }
7.        foreach ($scopes as $scope) {
8.             $this->VALID_CLAIMS = $this->VALID_CLAIMS
              .' '.$scope['scope'];
9.                    $this->CLAIMS_VALUES
              [$scope['scope']] = $scope['claim'];
10.       }
11.   }
```

Kode Sumber 4.20 Kode Sumber getScope

4.2.5.2 Fungsi GetUserClaims

Fungsi getUserClaims adalah fungsi yang dipanggil untuk mendapatkan data *userinfo* sebagai *userinfo response* berdasarkan nilai *scope* pada *request* yang dikirimkan. Fungsi ini dimodifikasi sesuai penambahan dan modifikasi nilai *scope* dan *claim*nya. Kode sumber untuk fungsi ini dapat dilihat pada Kode Sumber 4.21

```
1.  public function getUserClaims($user_id,$claims,$client_
    id) {
2.        if (!$userDetails = $this->getUserDetails
          ($user_id)) {
3.            return false;
4.        }
5.        $claims = explode(' ', trim($claims));
6.        $userClaims = array();
7.        // for each requested claim, if the user has the
          claim, set it in the response
8.        $validClaims = explode(' ', $this->VALID_CLAIMS);
9.        foreach ($validClaims as $validClaim) {
10.           if (in_array($validClaim, $claims)) {
11.               if ($validClaim == 'roleunit') {
```

```
12.                 $userRoleUnit = $this->getRoleUnit
                    ($user_id, $client_id);
13.                 $userClaims['roleunit'] = array();
14.                 if (count($userRoleUnit) > 1 ) {
15.                     for ($x = 0; $x < count($userRoleUn
                        it); $x++) {
16.                         $userClaims['roleunit'][$x] =
                            $this->getUserClaim
                            ($validClaim, $userDetails
                            ['roleunit'], $userRoleUnit[$x]
                            );
17.                     }
18.                 }
19.                 else {
20.                     $userClaims['roleunit'] = $this-
                        >getUserClaim($validClaim,
                        $userDetails['roleunit'], $userRole
                        Unit[0]);
21.                 }
22.             }
23.             else if ($validClaim == 'menu') {
24.                 $userMenu = $this-
                    >getMenu($user_id, $client_id);
25.                 $userClaims['menu'] = array();
26.                 if (count($userMenu) > 1 ) {
27.                     for ($x = 0; $x < count($userMenu);
                         $x++) {
28.                         $userClaims['menu'][$x] =
                            $this-> getUserClaim (
                            validClaim, $userDetails['menu
                            '], $userMenu[$x]);
29.                     }
30.                 }
31.                 else {
32.                     $userClaims['menu'] = $this-
                        >getUserClaim ($validClaim,
                        $userDetails['menu'],$userMenu[0]);
33.                 }
34.             }
35.             else if ($validClaim == 'resource') {
36.                 $userResource = $this-
                    >getResource($user_id, $client_id);
37.                 $userClaims['resource'] = array();
```

```
38.                 if (count($userResource) > 1 ) {
39.                     for ($x = 0; $x < count
                        ($userResource); $x++) {
40.                         $userClaims['resource'][$x] =
                            $this->getUserClaim (
                            $validClaim, $userDetails['reso
                            urce'], $userResource[$x]);
41.                     }
42.                 }
43.                 else {
44.                     $userClaims['resource'] = $this-
                        >getUserClaim($validClaim,
                        $userDetails['resource'], $userReso
                        urce[0]);
45.                 }
46.             }
47.             else {
48.                 $userClaims = array_merge($userClaims,
                    $this->getUserClaim
                    ($validClaim, $userDetails));
49.             }
50.         }
51.     }
52.     return $userClaims;
53. }
```

Kode Sumber 4.21 Kode Sumber getUserClaims

4.2.5.3 Fungsi GetUserClaim

Fungsi getUserClaim merupakan fungsi yang dipanggil pada fungsi getUserClaims untuk mendapatkan nilai masing-masing *claim* pada suatu *scope*. Kode sumber modifikasi yang dilakukan pada fungsi ini dapat dilihat pada Kode Sumber 4.22

```
1.  protected function getUserClaim($claim,$userDetails
    , $data = null)
2.      {
3.          $userClaims = array();
```

```
4.          $claimValuesString = $this->
            CLAIMS_VALUES[$claim];
5.          $claimValues = explode(' ', $claimValuesStr
            ing);
6.          if ($data != null) {
7.              foreach ($claimValues as $value) {
8.                  $userClaims[$value] = isset($data[$
                    value]) ? $data[$value] : null;
9.              }
10.         }
11.         else {
12.             foreach ($claimValues as $value) {
13.                 $userClaims[$value] = isset(
                    $userDetails[$value]) ? $userDetail
                    s[$value] : null;
14.             }
15.         };
16.         return $userClaims;
17.     }
```

Kode Sumber 4.22 Kode Sumber getUserClaim

4.2.5.4 Fungsi GetUser

Fungsi getUser digunakan untuk mendapatkan nilai *claim* pada *scope profile*. Kode sumber yang dimodifikasi dapat dilihat pada Kode Sumber 4.23

```
1.  public function getUser($username)
2.  {
3.      $stmt = $this->db-> prepare
        ($sql = sprintf('SELECT user_id, name, nicknames
        AS nickname, username,
        email, email_verified, alternate_email,
        alternate_email_verified, phone, phone_verified,
        enabled, gender, birthdate, zoneinfo, locale,
        last_update,locked, suspended, integra_id from %
        s where user_id=:username', $this->
        config['user_table']));
4.      $stmt->execute ( array
        ('username' => $username));
```

```
5.      if (!$userInfo = $stmt->
        fetch(\PDO::FETCH_ASSOC)) {
6.          return false;
7.      }
8.      $picture = 'https://my.its.ac.id/photo?id='.
        $userInfo['user_id'];
9.      return array_merge(array(
10.         'user_id' => $username,
11.         'picture' => $picture
12.       ), $userInfo);
13.  }
```

Kode Sumber 4.23 Kode Sumber getUser

4.2.5.5 Fungsi GetResource

Fungsi getResource digunakan untuk mendapatkan nilai *claim* pada *scope resource*. Kode sumber yang dimodifikasi dapat dilihat pada Kode Sumber 4.24

```
1.  public function getResource($userId, $clientId) {
2.      $stmt = $this->db-> prepare
        ($sql = sprintf("SELECT DISTINCT
        rr.resource_id AS resource_id, r.name AS name,
        r.path AS path, r.method AS method, r.api_id AS
         api_id,a.context AS context, a.version AS vers
        ion  FROM %s AS ur
3.        JOIN %s AS rr ON ur.role_id = rr.role_id
4.        JOIN %s as r ON rr.resource_id= r.resource_id
5.        JOIN %s as a ON r.api_id = a.api_id
6.       WHERE ur.user_id =:userid AND ur.client_id =
         :clientid
7.        AND ur.expired_at > getdate()",$this->
         config['user_role_table'],
8.       $this->config['resource_role_table'],$this->
         config['resource_table'],
9.        $this->config['api_table']));
10.    $stmt->execute(array
       ('userid'=>$userId,'clientid'=>$clientId));
11.
```

```
12.      if (!$userResource = $stmt->fetchAll
         (\PDO::FETCH_ASSOC)) {
13.          return false;
14.      }
15.
16.      return $userResource;
17. }
```

Kode Sumber 4.24 Kode Sumber getResource

4.2.5.6 Fungsi GetMenu

Fungsi getMenu digunakan untuk mendapatkan nilai *claim* pada *scope menu*. Kode sumber yang dimodifikasi dapat dilihat pada Kode Sumber 4.25.

```
1.  public function getMenu($userId, $clientId) {
2.
3.        $stmt = $this->db->prepare
          ($sql = sprintf("SELECT DISTINCT
4.       mr.menu_id AS menu_id,m.parent_id AS parent_id,
5.       m.name AS menu_name, r.name AS role_name,m.path
         AS path, mr.can_insert AS can_insert, mr.can_upd
         ate AS can_update, mr.can_delete AS can_delete,
         m.visibility AS visibility, m.active AS active,
         m.menu_order AS menu_order, m.icon AS icon, m.na
         me_en AS name_en
6.        FROM %s AS ur
7.        JOIN %s AS mr ON ur.role_id = mr.role_id
8.        JOIN %s as m ON mr.menu_id = m.menu_id
9.        JOIN %s AS r ON mr.role_id = r.role_id
10.       WHERE ur.user_id =:userid AND ur.client_id =
          :clientid
11.      AND m.visibility=1 AND m.active=1
12.      AND ur.expired_at > getdate()
13.       ORDER BY m.menu_order",$this->
          config['user_role_table'],
14.      $this->config['menu_role_table'],
15.       $this->config['menu_table'],$this-
          >config['role_table']));
16.      $stmt->execute(array('userid' => $userId,
```

```
17.      'clientid'  =>  $clientId));
18.     if (!$userMenu = $stmt->
        fetchAll(\PDO::FETCH_ASSOC)) {
19.         return false;
20.     }
21.     return $userMenu;
22. }
```

Kode Sumber 4.25 Kode Sumber getMenu

4.2.5.7 Fungsi GetRoleUnit

Fungsi getRoleUnit digunakan untuk mendapatkan nilai *claim* pada *scope roleunit*. Kode sumber yang dimodifikasi dapat dilihat pada Kode Sumber 4.26.

```
1.  public function getRoleUnit($userId, $clientId)
2.  {
3.      $stmt = $this->db->
        prepare($sql = sprintf("SELECT DISTINCT ur.role
        _id AS role_id, ur.is_default AS role_default,
        r.name AS role_name, ur.unit_id AS unit_id,
        u.name AS unit_name FROM %s AS ur
        INNER JOIN %s AS r ON ur.role_id = r.role_id
4.      INNER JOIN %s AS u ON ur.unit_id = u.unit_id
5.      WHERE ur.user_id =:userid AND ur.client_id=:cli
        entid AND ur.expired_at > getdate()", $this-
        >config ['user_role_table'], $this->config
        ['role_table'], $this-
        >config['unit_table']));
6.
7.      $stmt->execute(array('userid'   =>  $userId,
8.                      'clientid'  =>  $clientId));
9.
10.     if (!$userRole = $stmt->
        fetchAll(\PDO::FETCH_ASSOC)) {
11.         return false;
12.     }
13.
14.     return $userRole;
15. }
```

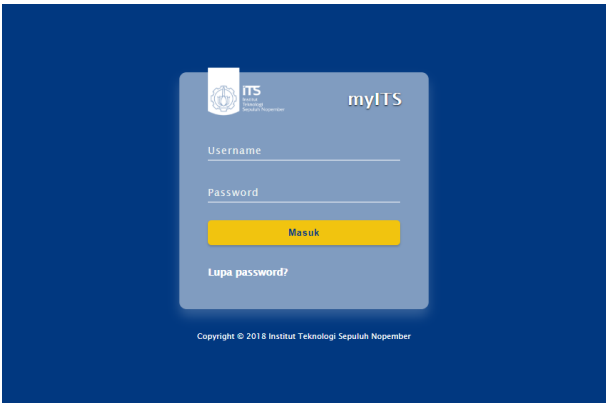
Kode Sumber 4.26 Kode Sumber getRoleUnit

4.3 Implementasi Antarmuka Pengguna

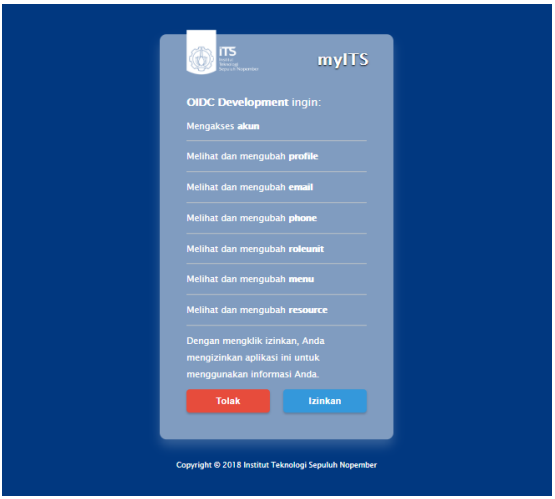
Implementasi antarmuka pengguna dibuat dengan menggunakan HTML dengan *template* Ample Admin dan dengan *template engine* dari Phalcon: volt. Pada subbab ini akan menjelaskan dan menampilkan tampilan halaman antarmuka yang diimplementasikan sesuai dengan rancangan antarmuka yang terdapat pada bab 3.

4.3.1 Halaman *Login*

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-001, yaitu *Login*. Halaman antarmuka *login* menampilkan halaman untuk pengguna melakukan proses *login* dengan menginputkan *username* dan *password*. Tampilan implementasi halaman *login* dapat dilihat pada Gambar 4.2 dan tampilan halaman *user consent* untuk aplikasi eksternal dapat dilihat pada Gambar 4.3.



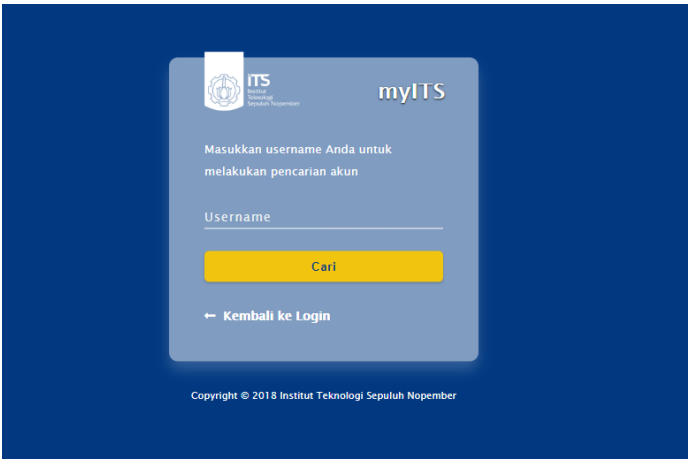
Gambar 4.2 Implementasi Halaman *Login*



Gambar 4.3 Implementasi Halaman User Consent

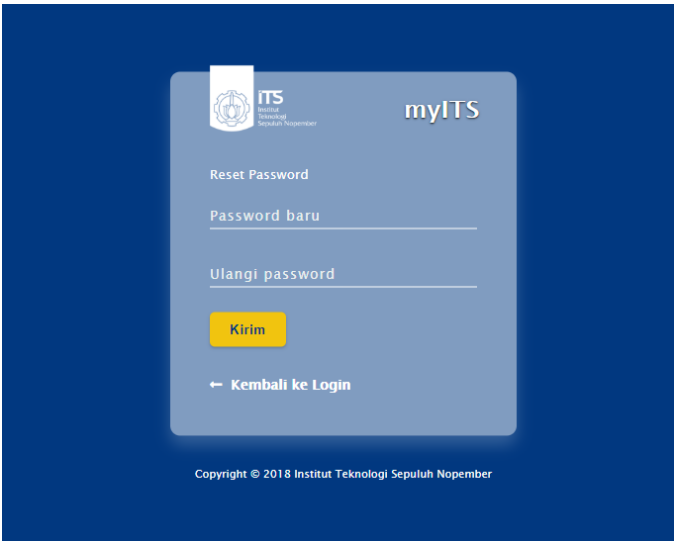
4.3.2 Halaman Reset *Password*

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-005, yaitu Reset *Password*. Halaman antarmuka reset *password* menampilkan halaman yang digunakan pengguna untuk melakukan pengaturan ulang *password* jika pengguna lupa akan *password*nya. Tampilan implementasi halaman Reset *password* ditunjukan pada Gambar 4.4, Gambar 4.5, dan Gambar 4.6.



Gambar 4.4 Implementasi Halaman Reset *Password* (1)

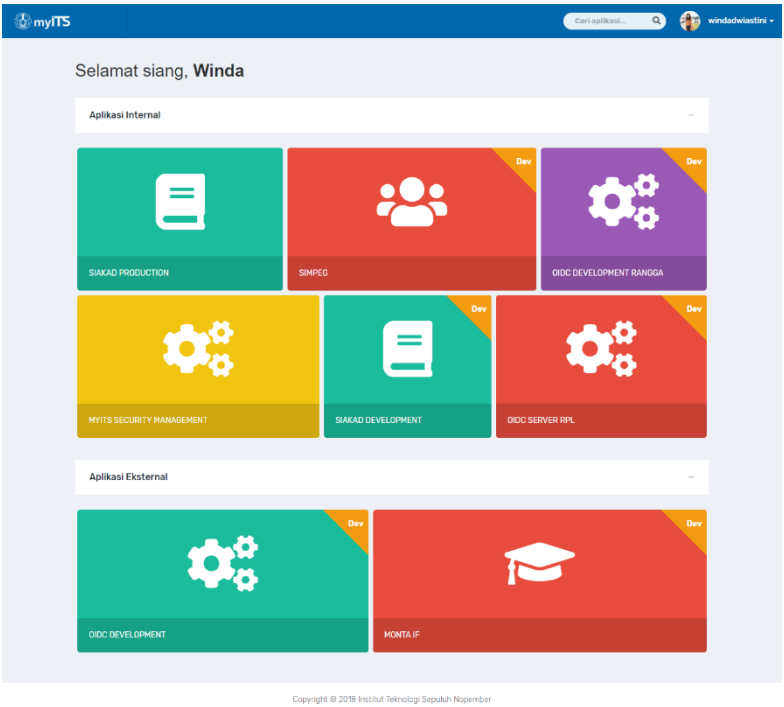
Gambar 4.5 Implementasi Halaman Reset *Password* (2)



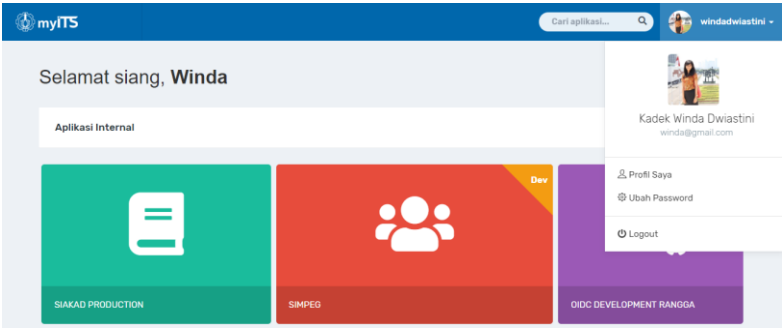
Gambar 4.6 Implementasi Halaman Reset *Password* (3)

4.3.3 Halaman Dashboard myITS

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-002 Masuk ke Aplikasi dan UC-007 *Logout*. Pengguna dapat masuk ke dalam aplikasi yang menggunakan fitur *Single Sign On* myITS yang ditampilkan pada halaman dashboard ini dengan memilih dan menekan aplikasi yang diinginkan. Pengguna juga dapat melakukan *logout* dengan memilih pilihan *logout*. Implementasi halaman ini dapat dilihat pada Gambar 4.7 dan Gambar 4.8.



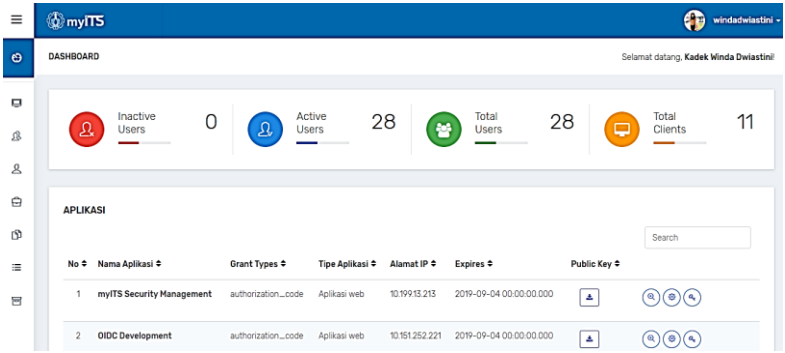
Gambar 4.7 Implementasi Halaman Dashboard myITS (1)



Gambar 4.8 Implementasi Halaman Dashboard myITS (2)

4.3.4 Halaman Dashboard myITS Security Management

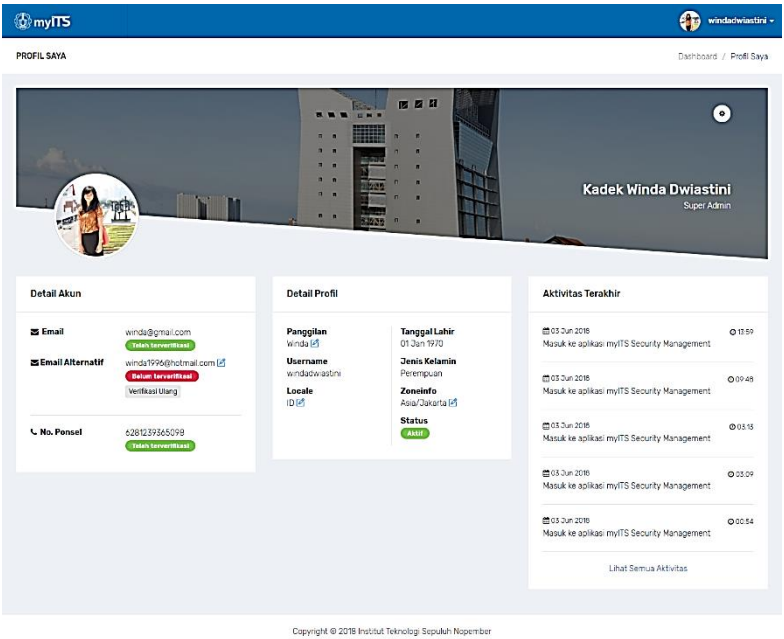
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-008 Lihat Aplikasi, UC-009 Generate Access Token, dan UC-010 Unduh Public Key untuk peran *developer*. Pengguna yang memiliki hak akses pada aplikasi myITS Security Management dapat menemukan halaman ini. Implementasi halaman ini dapat dilihat pada Gambar 4.9



Gambar 4.9 Implementasi Halaman Dashboard myITS Security Management

4.3.5 Halaman Profil Saya

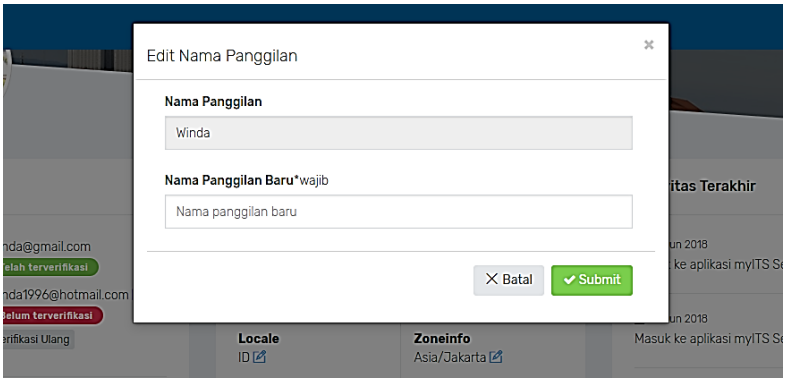
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-003 Lihat Profil dan UC-004 Verifikasi akun. Pengguna dapat melihat detail profilnya dan memperbaharui profilnya pada halaman ini. Implementasi halaman profil saya dapat dilihat pada Gambar 4.10.



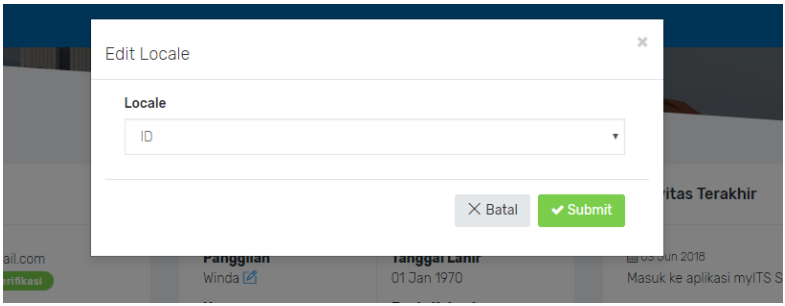
Gambar 4.10 Implementasi Halaman Profil Saya

4.3.6 Halaman Edit Profil

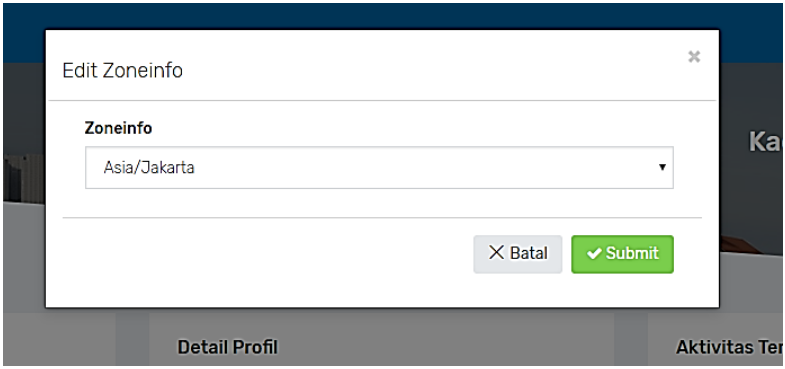
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-006 Edit Profil. Perbaharuan yang dapat dilakukan pengguna adalah pada nama panggilan (ditunjukan pada Gambar 4.11), locale (ditunjukan pada Gambar 4.12), zoneinfo (ditunjukan pada Gambar 4.13) , alternatif email (ditunjukan pada Gambar 4.14) dan foto profil (ditunjukan pada Gambar 4.15) serta melakukan perubahan *password* (ditunjukan pada Gambar 4.16).



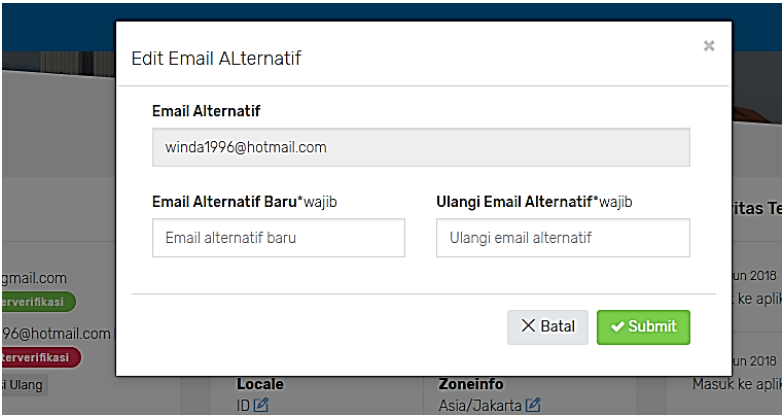
Gambar 4.11 Implementasi Halaman Edit Nama Panggilan



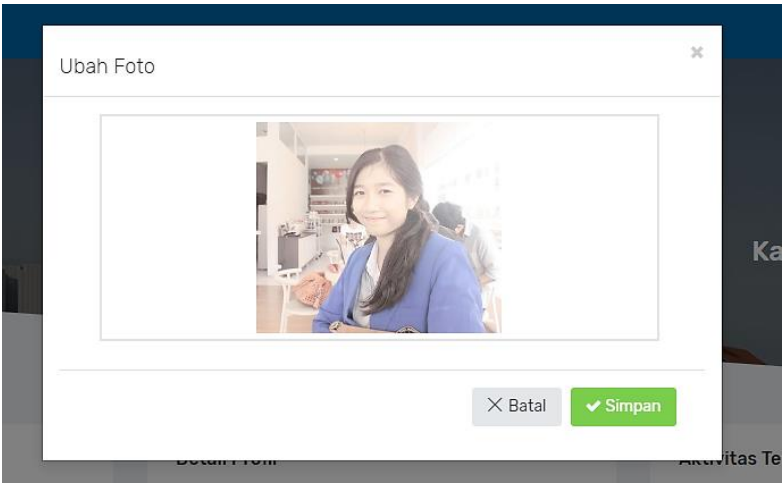
Gambar 4.12 Implementasi Halaman Edit Locale



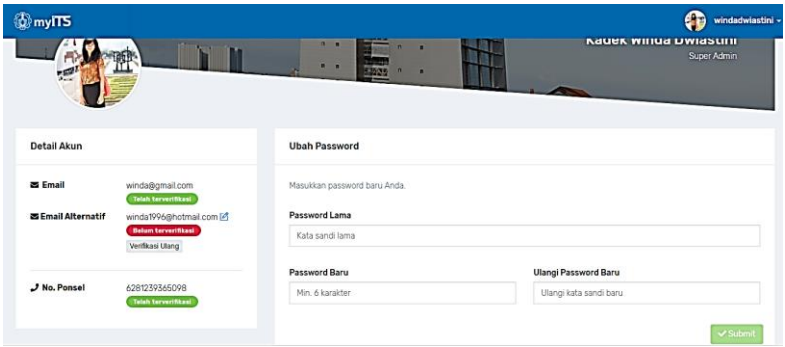
Gambar 4.13 Implementasi Halaman Edit Zoneinfo



Gambar 4.14 Implementasi Halaman Edit Email Alternatif



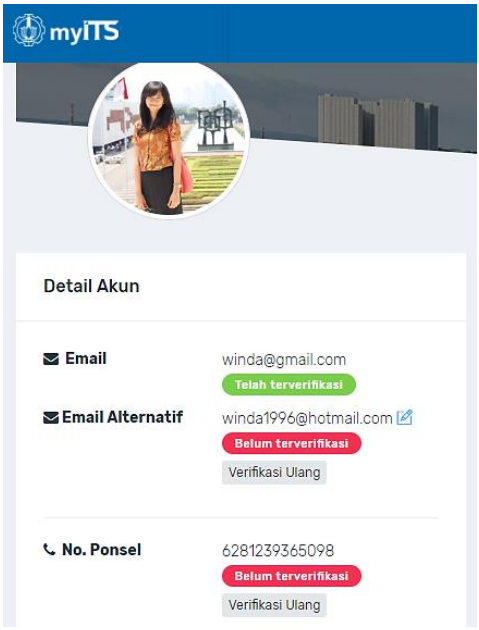
Gambar 4.15 Implementasi Halaman Edit Foto Profil



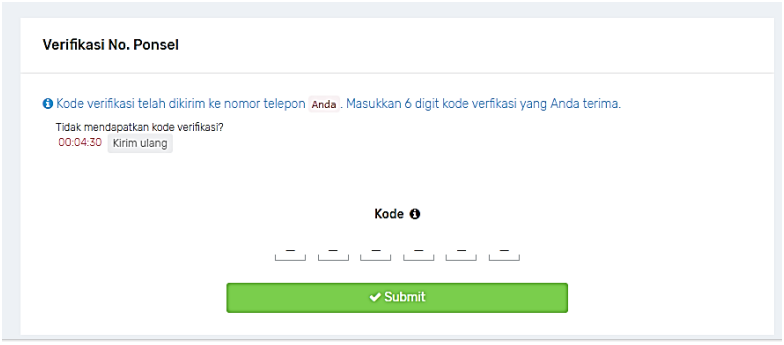
Gambar 4.16 Implementasi Halaman Ubah *Password*

4.3.7 Halaman Verifikasi Akun

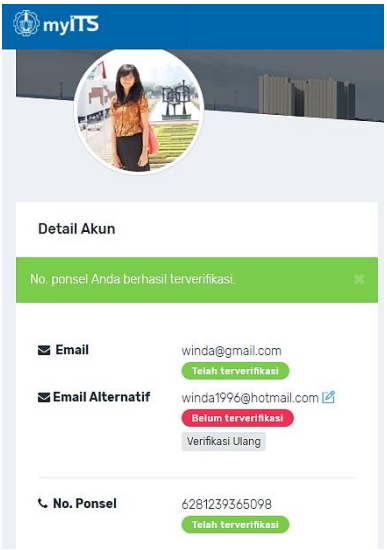
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-004 Verifikasi akun. Pengguna dapat melakukan verifikasi terhadap akun email, email alternatif, dan no. ponselnya yang belum terverifikasi sistem. Pengguna kemudian akan mendapatkan kode verifikasi pada akun yang dipilihnya kemudian menginputkan kode pada form kode verifikasi. Implementasi halaman verifikasi akun dapat dilihat pada Gambar 4.17, Gambar 4.18, dan Gambar 4.19.



Gambar 4.17 Implementasi Halaman Verifikasi Akun (1)



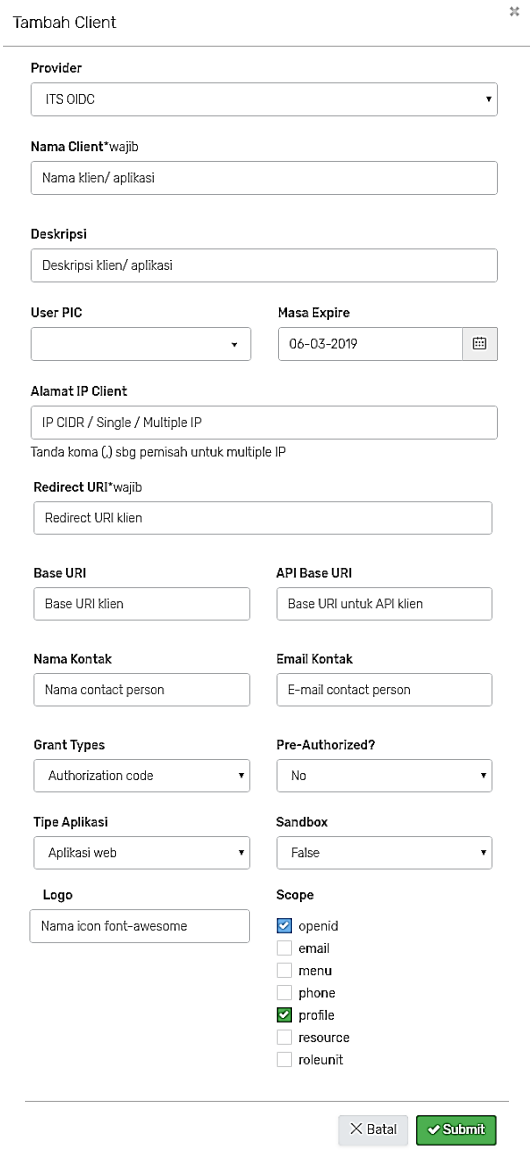
Gambar 4.18 Implementasi Halaman Verifikasi Akun (2)



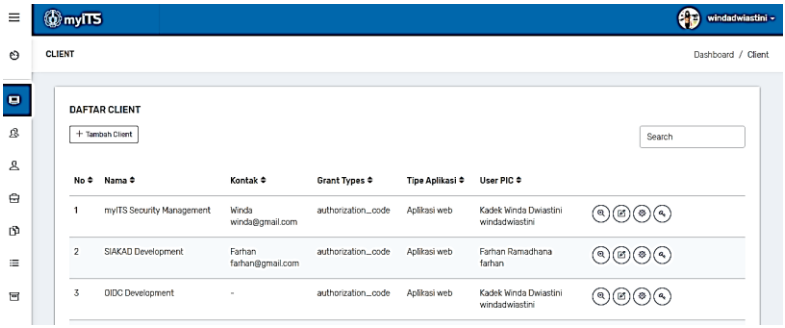
Gambar 4.19 Implementasi Halaman Verifikasi Akun (3)

4.3.8 Halaman Kelola Client

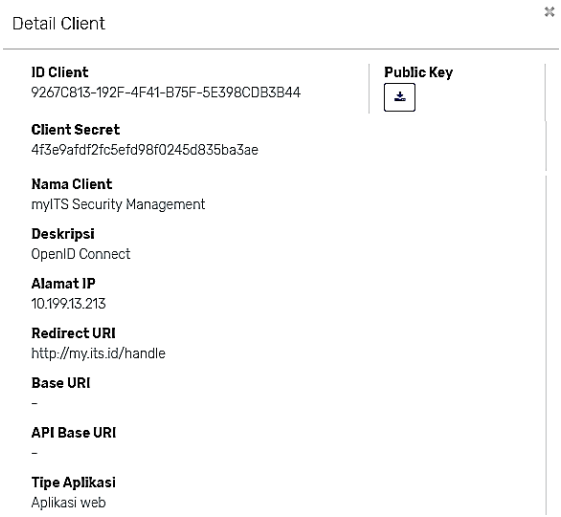
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-021 Kelola Client dan UC-009 Generate Access Token. Pengguna dengan peran *super admin* dapat mengelola *client* dengan menambah *client* baru dan memperbaharu *client* halaman ini. Selain itu pengguna juga dapat membuat *access token.* Implementasi halaman kelola c*lient* dapat dilihat pada Gambar 4.21, Gambar 4.20, Gambar 4.23, dan Gambar 4.22.



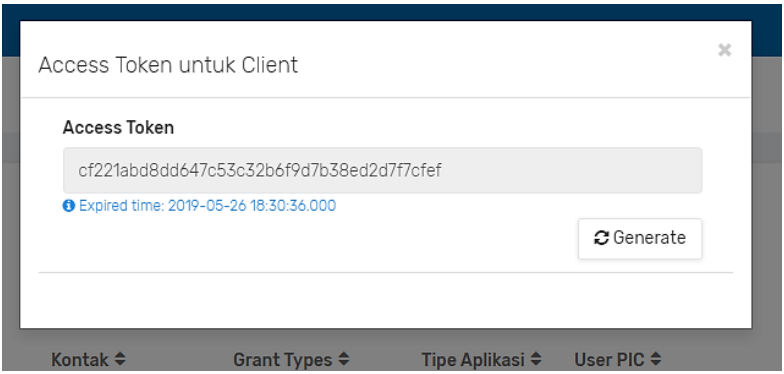
Gambar 4.20 Implementasi Halaman Tambah Client



Gambar 4.21 Implementasi Halaman Kelola Client



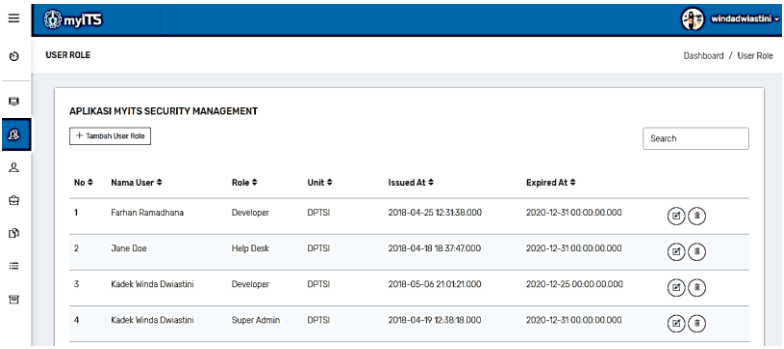
Gambar 4.22 Implementasi Halaman Detail Client



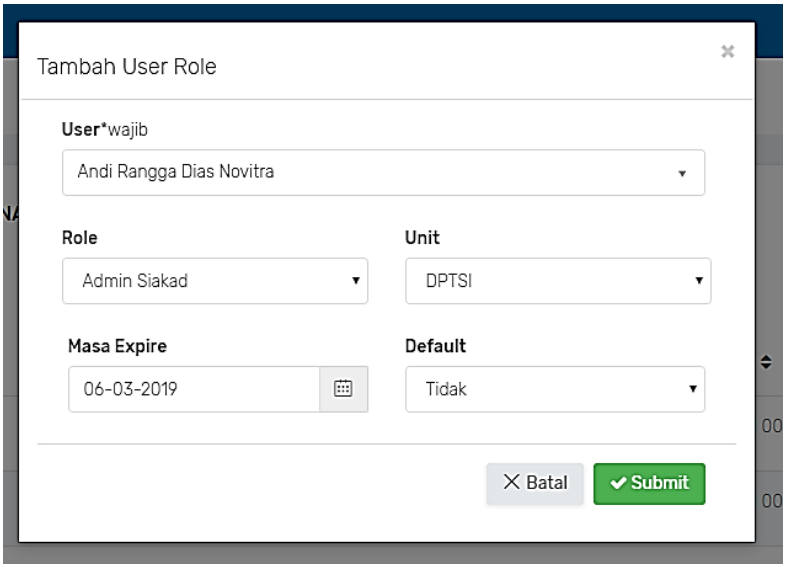
Gambar 4.23 Implementasi Halaman Generate Access Token

4.3.9 Halaman Atur User Role

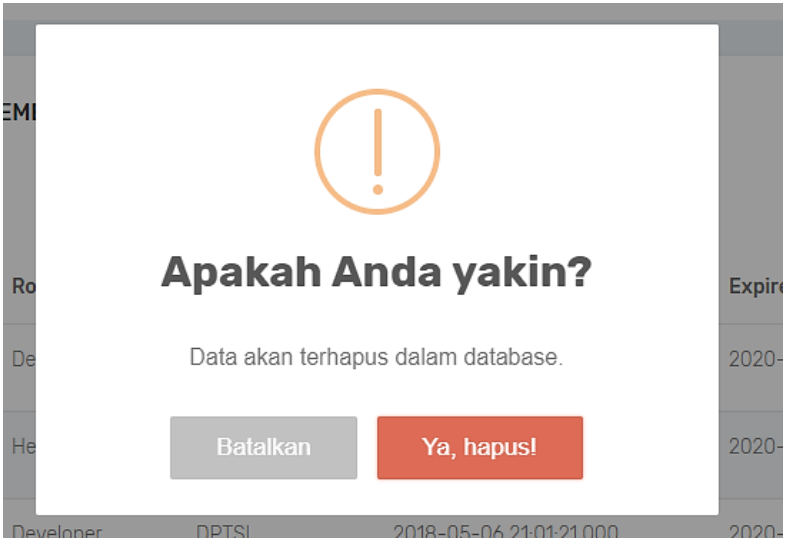
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-011 Atur User Role. Pengguna dengan hak akses *super admin* dan *developer* dapat mengatur *user role* dengan menambah, memperbaharui dan menghapus *user role* pada halaman ini. Implementasi halaman atur user role dapat dilihat pada Gambar 4.24, Gambar 4.25, dan Gambar 4.26.



Gambar 4.24 Implementasi Halaman Atur User Role



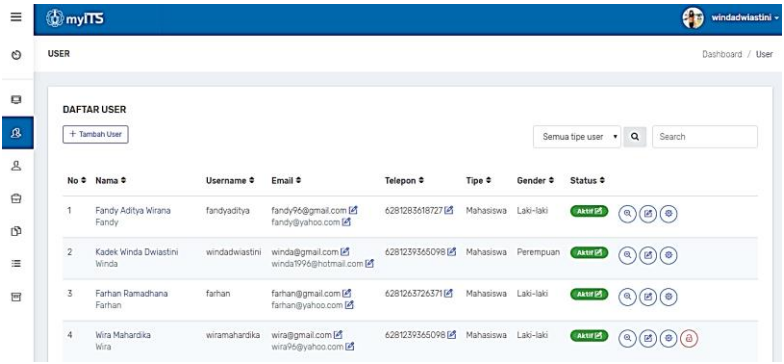
Gambar 4.25 Implementasi Halaman Tambah User Role



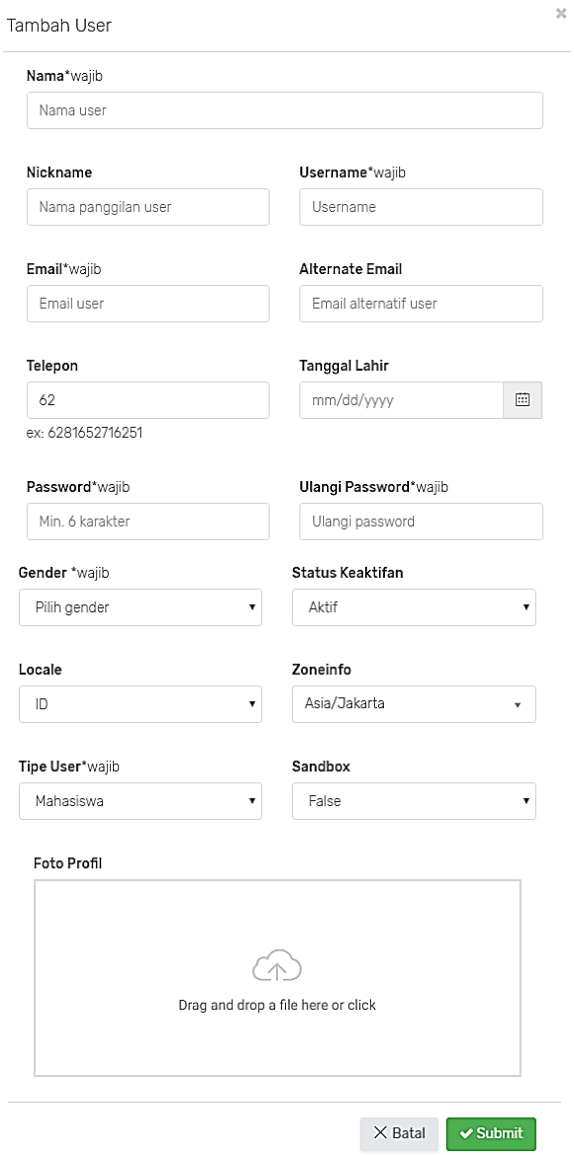
Gambar 4.26 Implementasi Halaman Hapus User Role

4.3.10 Halaman Kelola User

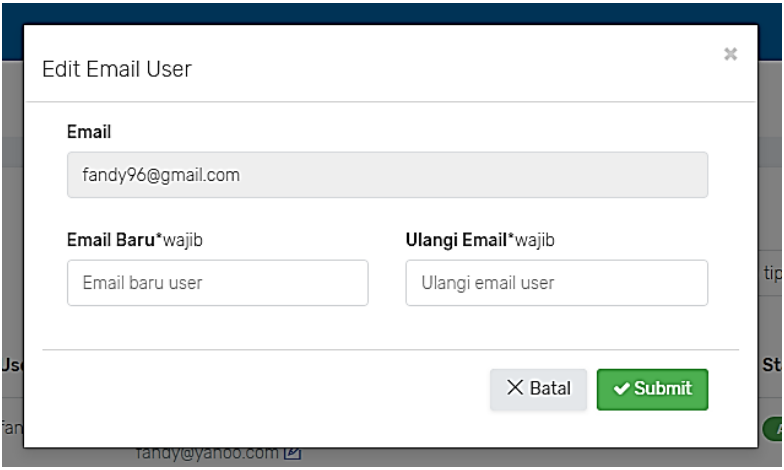
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-017 Kelola User. Pengguna dengan hak akses *super admin* dapat mengelola data *user* dengan menambah dan memperbaharui *user* pada halaman ini. Implementasi halaman profil saya dapat dilihat pada Gambar 4.27, Gambar 4.28, Gambar 4.29, Gambar 4.30, Gambar 4.31, Gambar 4.32, Gambar 4.33, dan Gambar 4.34.



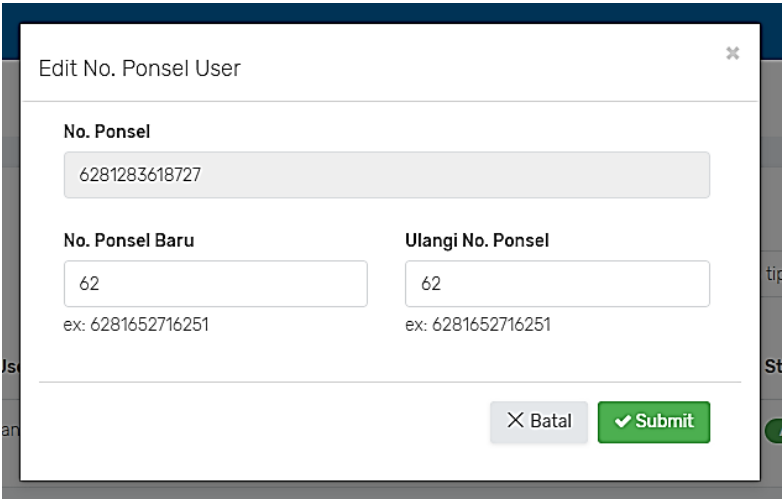
Gambar 4.27 Implementasi Halaman Kelola User



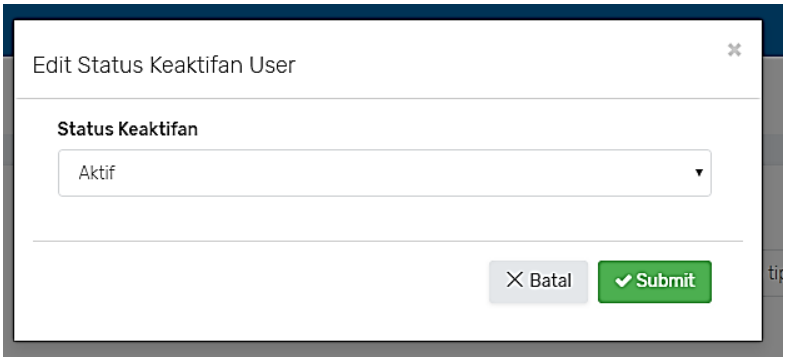
Gambar 4.28 Implementasi Halaman Tambah User



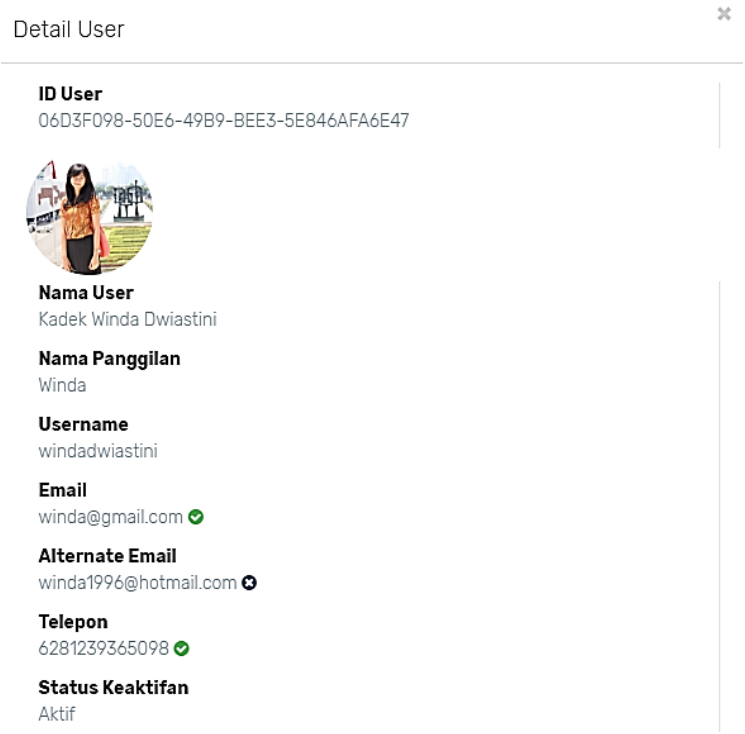
Gambar 4.29 Implementasi Halaman Edit Email User



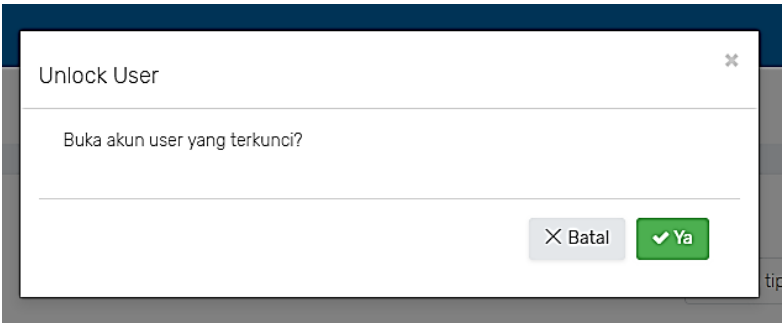
Gambar 4.30 Implementasi Halaman Edit No. Ponsel User



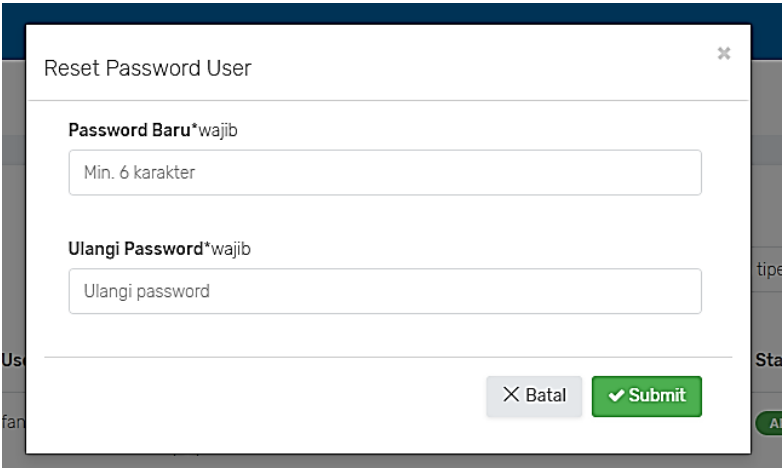
Gambar 4.31 Implementasi Halaman Edit Status Keaktifan User



Gambar 4.32 Implementasi Halaman Detail User



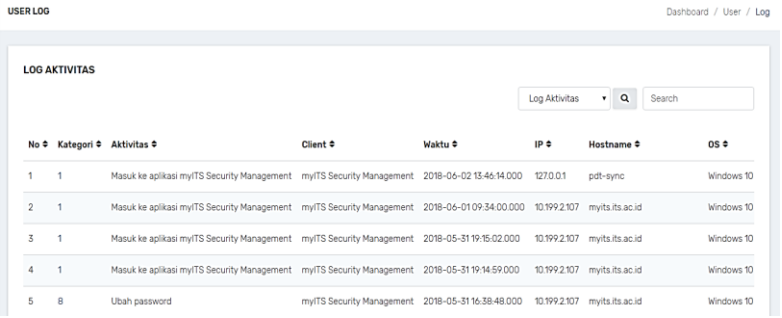
Gambar 4.33 Implementasi Halaman Unlock User



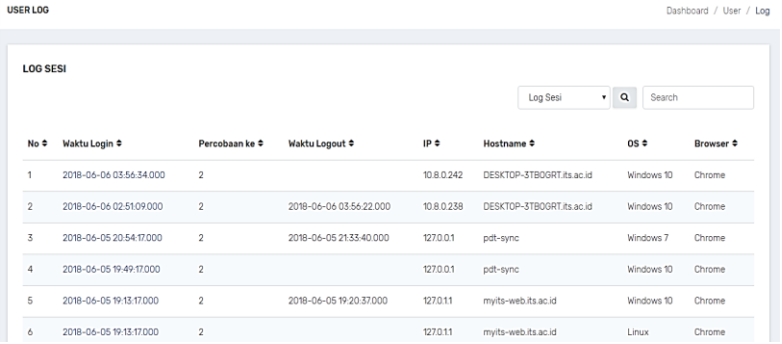
Gambar 4.34 Implementasi Halaman Reset *Password* User

4.3.11 Halaman Lihat Log User

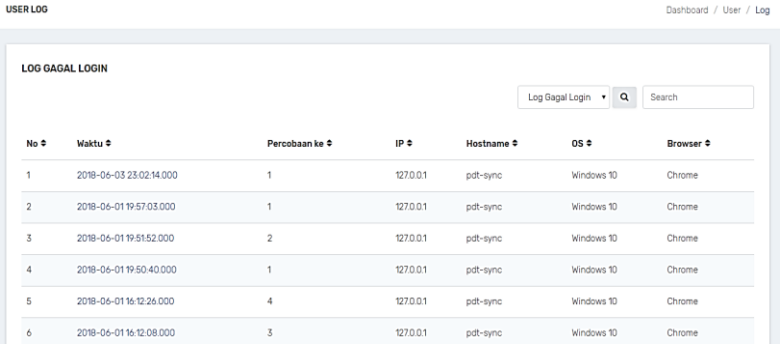
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-018 Lihat Log User. Pengguna dengan hak akses *super admin* dapat melihat log *user* pada halaman ini. Implementasi halaman lihat log user dapat dilihat pada Gambar 4.35, Gambar 4.36, dan Gambar 4.37.



Gambar 4.35 Implementasi Halaman Lihat Log User (Log Aktivitas)



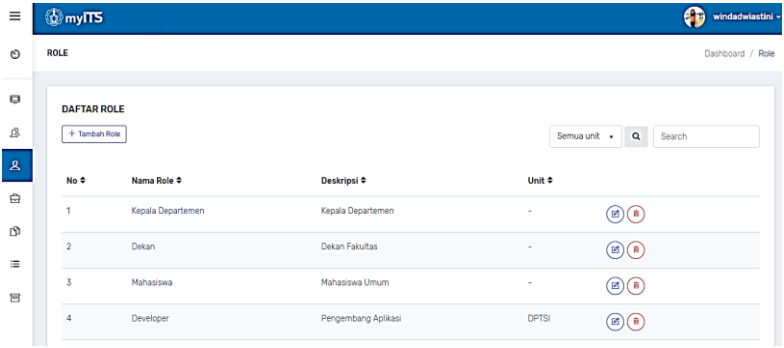
Gambar 4.36 Implementasi Halaman Lihat Log User (Log Sesi)



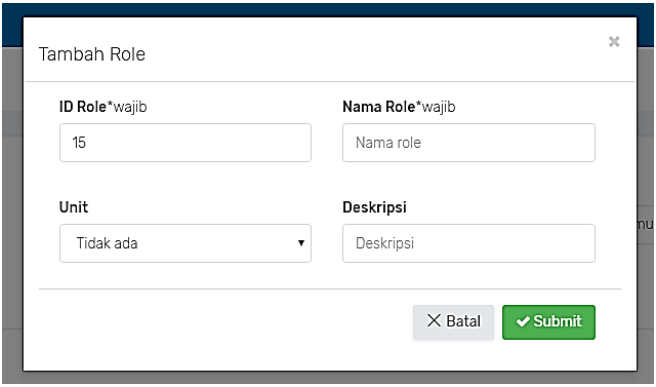
Gambar 4.37 Implementasi Halaman Lihat Log User (Log Gagal *Login*)

4.3.12 Halaman Kelola Role

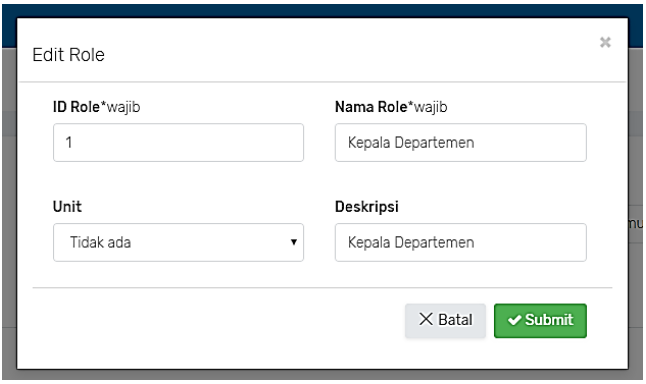
Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-020 Kelola Role. Pengguna dengan hak akses *super admin* dapat mengatur *role* dengan menambah, memperbaharui dan menghapus *role* pada halaman ini. Implementasi halaman kelola *role* dapat dilihat pada Gambar 4.38, Gambar 4.39, dan Gambar 4.40.



Gambar 4.38 Implementasi Halaman Kelola Role



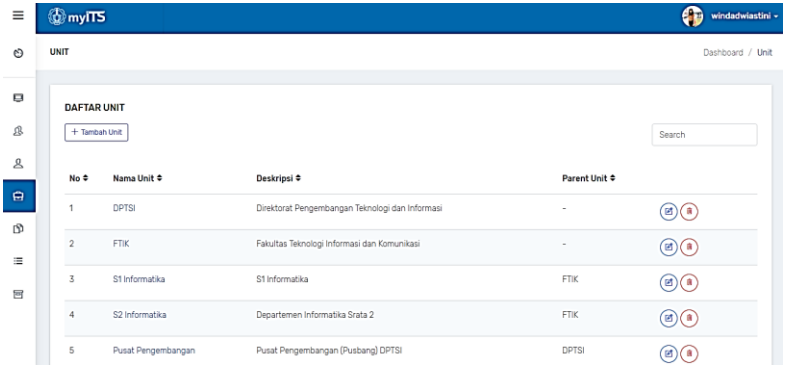
Gambar 4.39 Implementasi Halaman Tambah Role



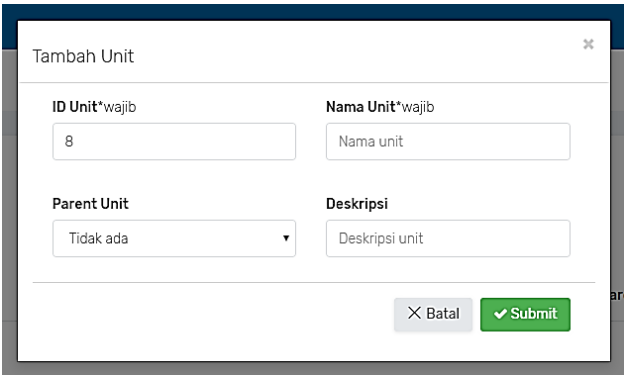
Gambar 4.40 Implementasi Halaman Edit Role

4.3.13 Halaman Kelola Unit

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-022 Kelola Unit. Pengguna dengan hak akses *super admin* dapat mengelola data *unit* dengan menambah, memperbaharui, dan menghapus *unit* pada halaman ini. Implementasi halaman kelola *unit* dapat dilihat pada Gambar 4.41 dan Gambar 4.42



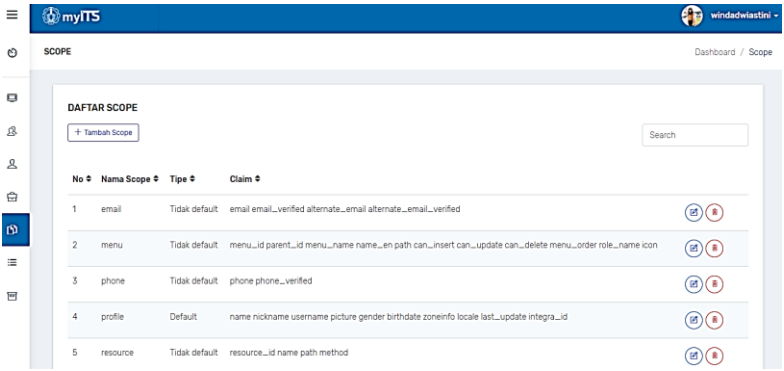
Gambar 4.41 Implementasi Halaman Kelola Unit



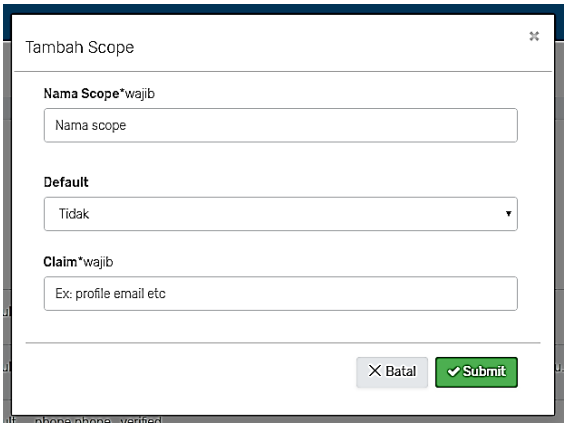
Gambar 4.42 Implementasi Halaman Tambah Unit

4.3.14 Halaman Kelola Scope

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-019 Kelola Scope. Pengguna dengan hak akses *super admin* dapat mengelola data *scope* dengan menambah, memperbaharui dan menghapus *scope* pada halaman ini. Implementasi halaman kelola *scope* dapat dilihat pada Gambar 4.43 dan Gambar 4.44.



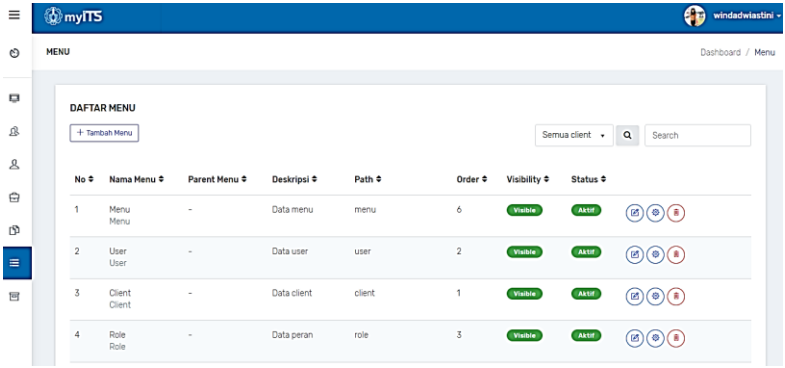
Gambar 4.43 Implementasi Halaman Kelola Scope



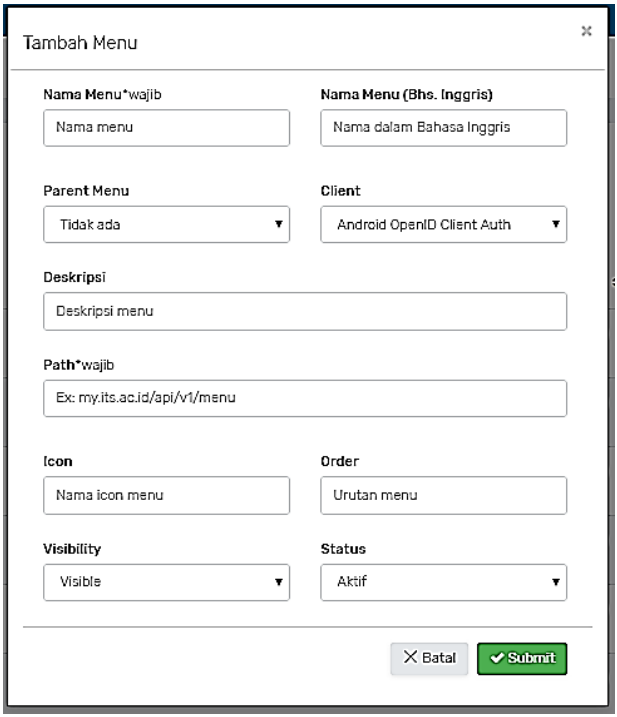
Gambar 4.44 Implementasi Halaman Tambah Scope

4.3.15 Halaman Kelola Menu

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-012 Kelola Menu. Pengguna dengan hak akses *super admin* dan *developer* dapat mengelola data *menu* dengan menambah, memperbaharui dan menghapus *menu* pada halaman ini. Implementasi halaman kelola *menu* dapat dilihat pada Gambar 4.45 dan Gambar 4.46.



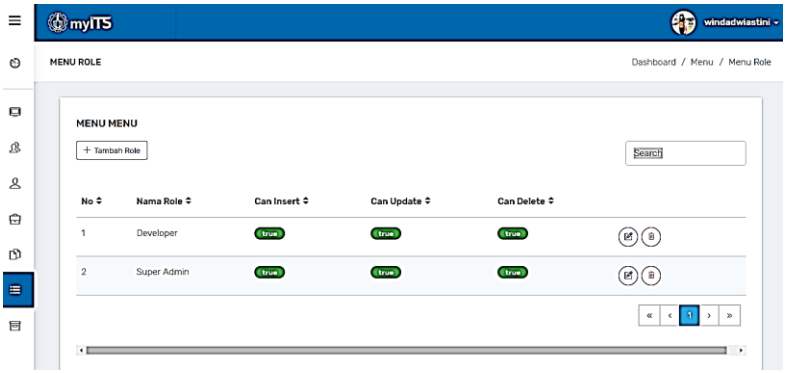
Gambar 4.45 Implementasi Halaman Kelola Menu



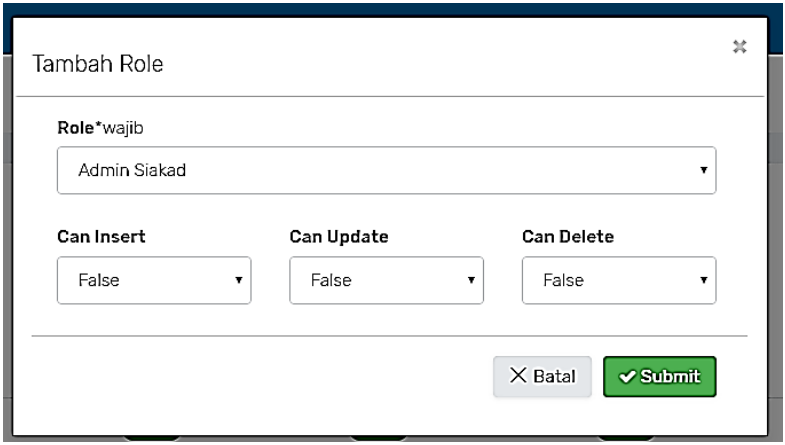
Gambar 4.46 Implementasi Halaman Tambah Menu

4.3.16 Halaman Atur Menu Role

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-013 Atur Menu Role. Pengguna dengan hak akses *super admin* dan *developer* dapat mengelola data *menu role* dengan menambah, memperbaharui dan menghapus *menu role* pada halaman ini. Implementasi halaman pengaturan *menu role* dapat dilihat pada Gambar 4.47 dan Gambar 4.48



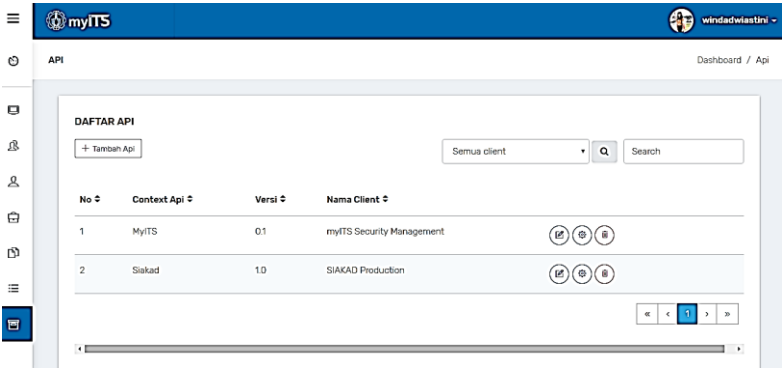
Gambar 4.47 Implementasi Halaman Atur Menu Role



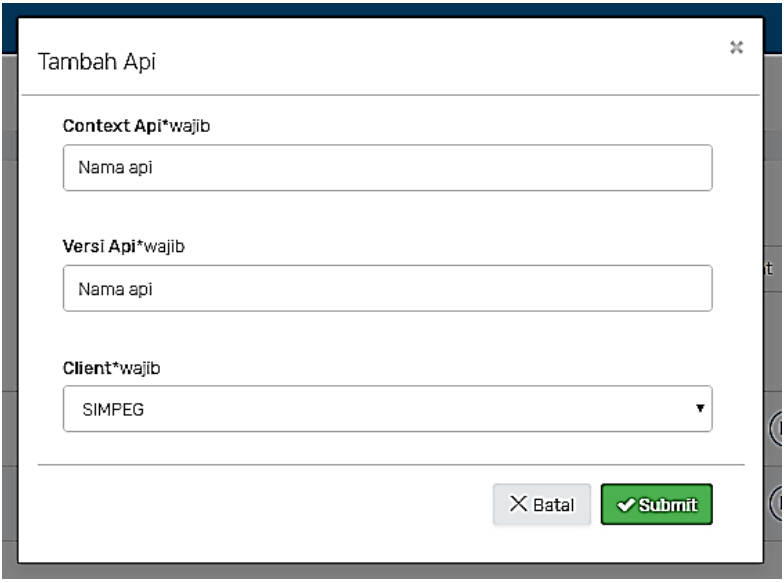
Gambar 4.48 Implementasi Halaman Tambah Menu Role

4.3.17 Halaman Kelola API

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-014 Kelola API. Pengguna dengan hak akses *super admin* dan *developer* dapat mengelola data *API* dengan menambah, memperbaharui dan menghapus *API* pada halaman ini. Implementasi halaman kelola *API* dapat dilihat pada Gambar 4.49 dan Gambar 4.50.



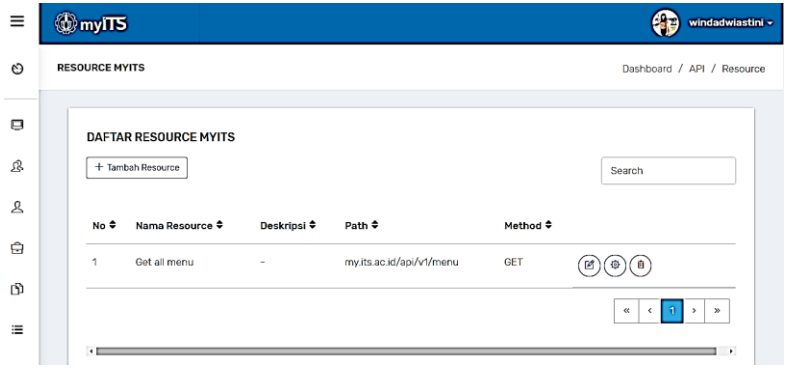
Gambar 4.49 Implementasi Halaman Kelola API



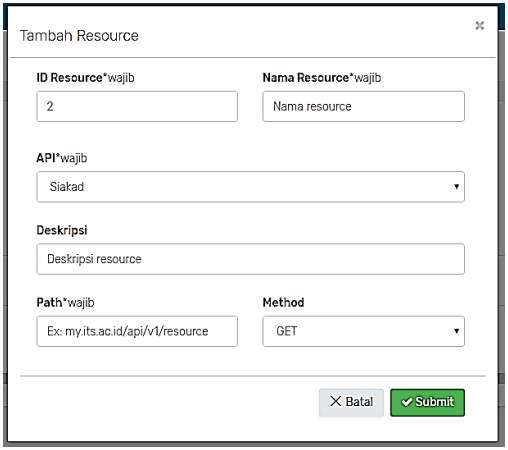
Gambar 4.50 Implementasi Halaman Tambah API

4.3.18 Halaman Kelola Resource

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-015 Kelola Resource. Pengguna dengan hak akses *super admin* dan *developer* dapat mengelola data *resource* dengan menambah, memperbaharui dan menghapus *resource* pada halaman ini. Implementasi halaman kelola *resource* dapat dilihat pada Gambar 4.51 dan Gambar 4.52.



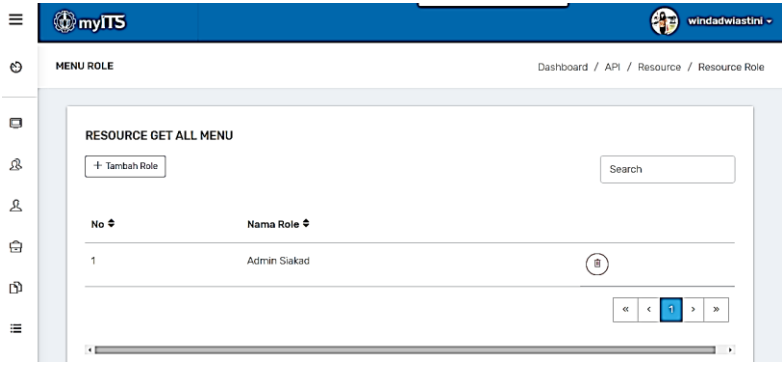
Gambar 4.51 Implementasi Halaman Kelola Resource



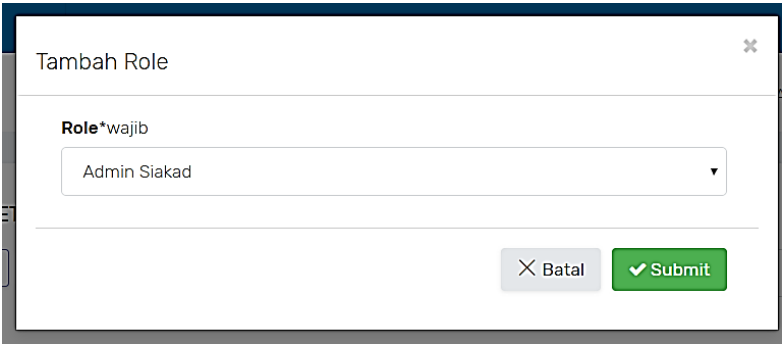
Gambar 4.52 Implementasi Halaman Tambah Resource

4.3.19 Halaman Atur Resource Role

Halaman ini merupakan halaman yang digunakan untuk kasus penggunaan UC-016 Atur Resource Role. Pengguna dengan hak akses *super admin* dan *developer* dapat mengelola data *resource role* dengan menambah, memperbaharui dan menghapus *resource role* pada halaman ini. Implementasi halaman atur *resource role* dapat dilihat pada Gambar 4.53 dan Gambar 4.54.



Gambar 4.53 Implementasi Halaman Atur Resource Role



Gambar 4.54 Implementasi Halaman Tambah Resource Role

4.4 Implementasi *Client*

Pada subbab ini akan dibahas langkah-langkah yang perlu dilakukan agar aplikasi *client* dapat menggunakan fitur *Single Sign On* myITS atau *Login with myITS Account*. Subbab ini diperlukan sebagai panduan kepada pengembang lain yang ingin menerapkan atau menggunakan fitur *login* ini. Terdapat 2 jenis aplikasi *client* yang akan dibahas, yaitu aplikasi berbasis web dan aplikasi berbasis Android.

4.4.1 Implementasi *Client* Berbasis Web

Pada bagian ini akan dijelaskan 2 aplikasi *client* yang menggunakan fitur MyITS SSO. Masing-masing merepresentasikan jenis aplikasi yang berbeda, satu untuk aplikasi internal (*preauthorized*) yaitu aplikasi SIAKAD dimana pengaturan hak akses pengguna atau *role-based-access-control* (RBAC) dilakukan secara terpusat pada sistem myITS, dan satu lagi untuk aplikasi eksternal (*non-preauthorized*) yaitu aplikasi MonTA IF dimana pengaturan RBAC dilakukan dari aplikasi *client* itu sendiri. Berikut akan dijelaskan implementasi masing-masing *client*.

4.4.1.1 Implementasi *Client* SIAKAD

Pada aplikasi *client* SIAKAD, pengguna diwajibkan untuk *login* menggunakan akun myITS. Berikut akan dijelaskan langkah-langkah yang harus dilakukan *client* untuk dapat menggunakan *login* dengan akun myITS.

1. Registrasi *Client*

Untuk dapat menggunakan fitur *login* dengan myITS, aplikasi *client* harus terdaftar di dalam sistem OP myITS. Registrasi *client* hanya dapat dilakukan oleh pengguna dengan

peran Super Admin. Super Admin mendaftarkan *client* dengan memilih pilihan Tambah Client pada halaman Kelola Client dan mengisi isian.

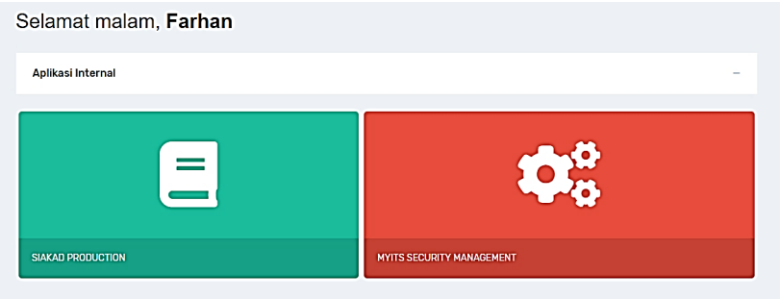
Saat registrasi *client* selesai dilakukan, maka sebuah *client* akan mempunyai *Client ID, Client Secret, Public Key* yang nantinya digunakan oleh *client* untuk proses otorisasi dan otentikasi.

2. Registrasi Pengguna sebagai *Developer*

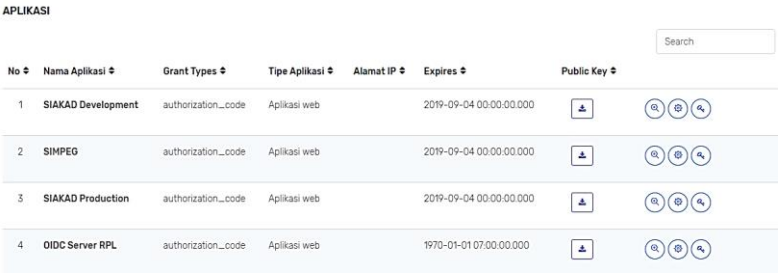
Setelah aplikasi *client* terdaftar dalam sistem, untuk mendapatkan *client ID, client secret,* dan *public key* aplikasi, seorang pengguna harus didaftarkan dalam sistem myITS Security Management untuk dapat melihat detail aplikasi. Pendaftaran pengguna dilakukan oleh Super Admin dengan menambah user pada halaman Kelola User dan mengatur *user role* pada *client* myITS Security Management.

3. *Client ID, Client Secret,* dan *Public Key*

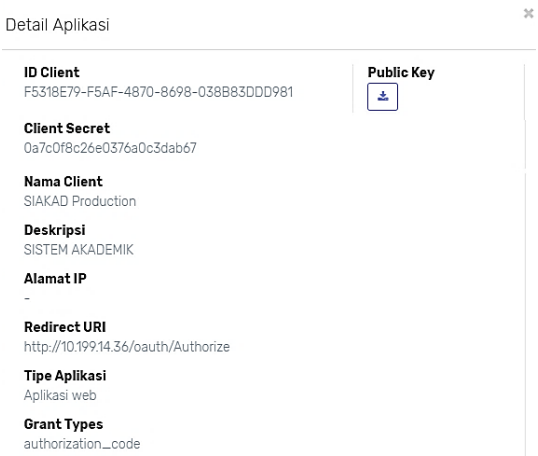
Setelah proses registrasi *client* dan registrasi *user* sebagai *developer*, maka *developer* dapat *login* dan masuk ke dalam aplikasi myITS Security Management seperti ditunjukan pada Gambar 4.55 untuk mendapatkan *client ID, client secret*, dan *public key* untuk *client* SIAKAD. Saat *login*, pada halaman dashboard myITS Security Management akan menampilkan daftar aplikasi yang dapat dikelola *developer* seperti ditunjukan oleh Gambar 4.56. Untuk dapat melihat detail aplikasi, pilih pilihan lihat detail aplikasi dan untuk mengunduh *public key* pilih pilihan *download* pada kolom *Public Key* (ditunjukan pada Gambar 4.57).



Gambar 4.55 Halaman Dashboard myITS Developer SIAKAD



Gambar 4.56 Halaman Dashboard myITS Security Management Developer SIAKAD



Gambar 4.57 Detail Aplikasi SIAKAD

4. Setting *Authorization Controller*

Tahapan selanjutnya adalah membuat sebuah *controller* untuk menerima *response* dan mengirimkan *request* ke *endpoint* pada sistem OP myITS sesuai pada *redirect URI client* yang didaftarkan. Sebelum itu, lakukan pengaturan pada halaman *login* agar ketika tombol ditekan maka akan mengirimkan *authorization request* ke *authorization endpoint* pada OP: https://my.its.ac.id/authorize. *Request* dapat dikirimkan dengan metode POST maupun GET.

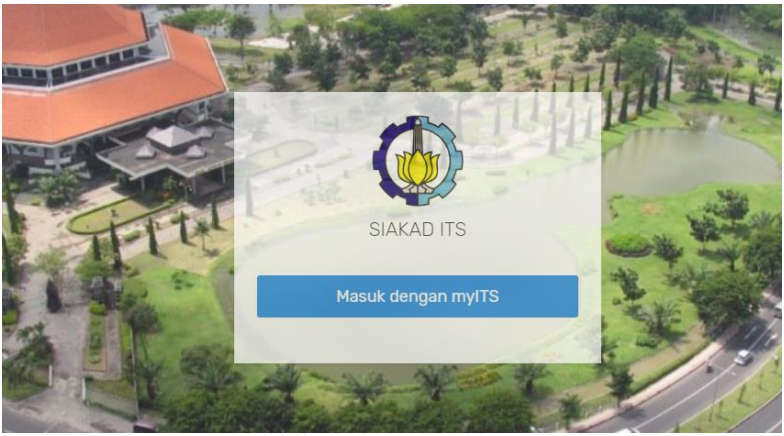
Controller yang dibuat adalah *controller* yang dipanggil saat *client redirect URI* dipanggil. Adapun nilai *scope* yang harus dikirimkan adalah 'openid profile phone email roleunit menu resource'. Nilai *scope* roleunit, menu, dan resource digunakan karna aplikasi SIAKAD menggunakan RBAC terpusat pada myITS. Selanjutnya pengaturan menu, menu role, API, resource, dan resource role dilakukan pada aplikasi myITS Security Management. Kode Sumber 4.27 berikut adalah *pseudocode* untuk *authorization controller*.

```
1.  public function index()
2.  {
3.        IF authorizationResponse has 'login=true'
          parameter THEN
4.            redirect and send authorization request
              with prompt=none
5.        ELSE
6.            CALL getAuthorizationResponse function
7.            SET code to the value returned from
              getAuthorizationResponse function
8.            IF code = 'login_required' THEN
9.                redirect and send authorization request
                  with prompt=login
10.           ENDIF
11.           SEND token request with POST method to
              token endpoint at OP
12.           GET token response
13.           IF token response has error parameter THEN
```

```
14.              PRINT error description
15.          ELSE
16.              validate ID Token with public key
17.              IF valid THEN
18.                  SEND userinfo request with GET
                     method to userinfo endpoint at OP
19.                  GET userinfo response
20.              ELSE
21.                  PRINT error
22.              ENDIF
23.          ENDIF
24. }
25.
26. public function getAuthorizationResponse()
27. {
28.      IF authorizationResponse has parameter 'error
29.      THEN
30.         IF authorizationResponse error = 'login_req
            uired'
31.              RETURN 'login_required'
32.          ELSE
33.              PRINT error_description
34.         ENDIF
35.      ELSE
36.          GET code value & state value
37.          IF state value = origin state value THEN
38.              RETURN code value
39.          ELSE
40.              PRINT error
41.          ENDIF
42.      ENDIF
43. }
```

Kode Sumber 4.27 Pseudocode *Authorization Controller* untuk *Client*

Implementasi halaman *login* aplikasi *client* SIAKAD ditunjukan pada Gambar 4.58.

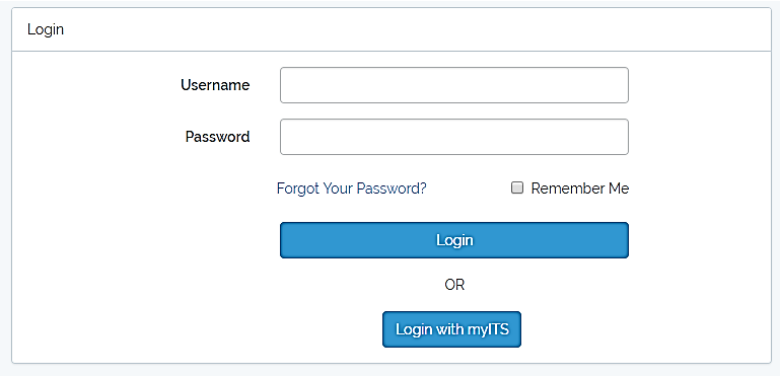


Gambar 4.58 Halaman *Login* SIAKAD

4.4.1.2 Implementasi *Client* MonTA IF

Implementasi pada *client* MonTA IF sama dengan pada *client* SIAKAD yang telah dijelaskan sebelumnya. *Client* MonTA IF harus didaftarkan pada sistem OP myITS, mendaftarkan *user developer* sehingga *client ID, client secret,* dan *public key* aplikasi MonTA dapat diperoleh. Kemudian *client* dapat menerapkan *pseudocode* untuk *authorization controller* yang dipanggil oleh *redirect URI client* MonTA.

Perbedaan terletak pada nilai *scope* yang dikirimkan pada *authorization request*, yaitu: ‘openid profile’. Nilai lainnya tidak dicantumkan karena aplikasi MonTA tidak menerapkan RBAC terpusat pada myITS dan mengatur kontrol akses penggunanya secara terpisah. Jadi informasi yang dibutuhkan hanyalah ID *User*. Implementasi halaman *login* MonTA dapat dilihat pada Gambar 4.59.

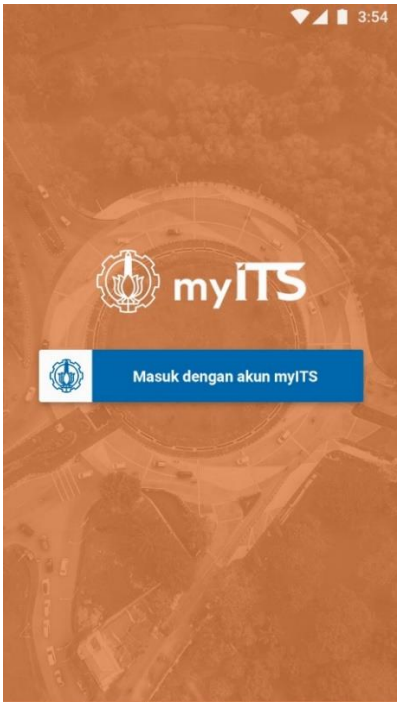


Gambar 4.59 Halaman *Login* MonTA IF

4.4.2 Implementasi *Client* berbasis Aplikasi *Mobile*

Pada bagian ini akan dijelaskan aplikasi *client* berbasis *mobile* Android yaitu aplikasi myITS Wali yang menggunakan fitur MyITS SSO. Sistem *login* pada aplikasi Android myITS Wali hanya dapat dilakukan oleh pengguna yang memiliki akun myITS. Aplikasi ini tidak menggunakan layanan RBAC terpusat pada myITS dan mengaturnya pada aplikasi myITS Wali sendiri.

Sama seperti implementasi *client* berbasis aplikasi web, *client* myITS Wali harus terdaftar di dalam sistem sehingga *developer* aplikasi ini dapat memperoleh *client ID, client secret*, dan ID Token yang nantinya digunakan untuk proses otorisasi dan otentikasi pada myITS. Halaman *login* aplikasi myITS Wali dapat dilihat pada Gambar 4.60.



Gambar 4.60 Halaman *Login* myITS Wali

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan pengujian dan evaluasi dari sistem MyITS SSO yang meliputi rincian lingkungan pengujian dan pengujian pada semua fungsionalitas sistem. Aspek yang diperhatikan dalam pengujian sistem ini adalah terpenuhinya fungsionalitas dari sistem.

5.1 Lingkungan Uji Coba

Lingkungan uji coba merupakan perangkat keras dan perangkat lunak yang digunakan selama melakukan pengujian. Pengujian dilakukan dengan menggunakan beberapa lingkungan pengujian yaitu dua klien, *web server* dan *server* basis data. Rincian dari keempat lingkungan pengujian ditunjukkan pada Tabel 5.1 dan Tabel 5.4.

Tabel 5.1 Tabel lingkungan uji coba klien MonTA

Spesifikasi	Deskripsi
CPU	Intel ® Core ™ i5-4200U CPU @ 1.60 GHz
RAM	4.00 GB
Sistem Operasi	Windows 10 Pro 64-bit
Browser	Google Chrome versi 66.0
Web Server	Apache XAMPP v.3.2.2
Basis Data	MySQL XAMPP v.3.2.2

Tabel 5.2 Tabel lingkungan uji coba klien SIAKAD

Spesifikasi	Deskripsi
CPU	Intel® Xeon® CPU E5-2690 v4 @ 2.6 GHz
RAM	8.00 GB
Sistem Operasi	Debian GNU/ Linux 9.4
Browser	Google Chrome versi 66.0
Web Server	Nginx 1.10.3
Basis Data	SQL Server

Tabel 5.3 Tabel lingkungan uji coba klien myITS Wali

Spesifikasi	**Deskripsi**
Smartphone	Oppo R7S
CPU	Octa-core (4x1.5 GHz Cortex-A53 & 4x1.0 GHz Cortex-A53)
RAM	4 GB
Sistem Operasi	Android 5.1 (Lollipop)

Tabel 5.4 Tabel lingkungan uji coba web server

Spesifikasi	**Deskripsi**
CPU	Intel® Xeon® CPU E5-2690 v4 @ 2.6 GHz
RAM	8.00 GB
Sistem Operasi	Debian GNU/ Linux 9.4
Web Server	Nginx 1.10.3

Tabel 5.5 Tabel lingkungan uji coba server basis data

Spesifikasi	**Deskripsi**
CPU	Intel® Xeon® CPU E7-4890 v2 @ 2.8 GHz
RAM	4.00 GB
Sistem Operasi	Debian GNU
Basis Data	SQL Server 2017

5.2 Pengujian

Pengujian fungsionalitas sistem adalah uji coba yang dilakukan terhadap fungsionalitas yang dapat dilakukan pengguna atau aktor pada sistem. Fungsionalitas tersebut merupakan kasus penggunaan yang sebelumnya telah dijelaskan pada Subbab 3.1.3. Penjelasan uji coba meliputi skenario uji coba dan hasil uji coba yang diharapkan.

Uji coba dilakukan dengan metode *black box* yang artinya fungsionalitas diperiksa apakah terpenuhi atau tidak tanpa melihat struktur internal maupun metode yang digunakan dalam pengerjaan fungsionalitas tersebut.

5.2.1 Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas sistem dilakukan dengan menguji kasus-kasus uji tiap kasus penggunaan. Berikut adalah kasus uji dari fungsionalitas sistem.

5.2.1.1 Kasus Uji *Login*

Pada kasus uji ini terdapat enam skenario pengujian yaitu 1. Saat aktor melakukan *login* langsung pada halaman web myITS, 2. Saat aktor melakukan *login* pada aplikasi *client* dengan menekan tombol *Login* dengan akun myITS, 3. Saat aktor salah memasukkan *username* dan *password*nya, 4. Saat aktor gagal melakukan *login* sebanyak 5 kali dalam waktu 5 menit, 5. Saat aktor yang telah ter-suspend sebelumnya melakukan gagal *login* sebanyak 5 kali dalam 5 menit, 6. Saat aktor dengan status akun yang terkunci melakukan *login*, dan 7. Saat aktor telah *login* pada myITS dan memilih tombol *login* dengan akun myITS pada halaman *client*. Sistem akan mengeluarkan pesan error seperti Gambar 5.21 untuk skenario 3, pesan error seperti Gambar 5.2 untuk skenario 4, pesan error seperti Gambar 5.3 untuk skenario 5, dan pesan error seperti Gambar 5.4 untuk skenario 6. Rincian kasus uji ini ditunjukan pada Tabel 5.6.

Tabel 5.6 Tabel kasus uji *login*

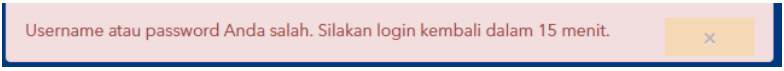
ID	**TC-001**
Kasus Penggunaan	*Login*
Nama	Pengujian *login* atau otentikasi ke dalam sistem
Tujuan Pengujian	Menguji apakah pengguna dapat *login* ke myITS dan aplikasi *client* yang terdaftar dalam sistem dapat menggunakan fitur *login* dengan akun myITS

Skenario 1	*Aktor login langsung pada web myITS*
Kondisi Awal	Aktor belum terotentikasi
Data Uji	*Username* dan *password* yang terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman myITS 2. Aktor mengisi *username* dan *password* 3. Aktor menekan tombol *login*
Hasil yang Diharapkan	Aktor berhasil *login* dan sistem menampilkan halaman dashboard myITS
Kondisi Akhir	Aktor berhasil *login* dan sistem menampilkan halaman dashboard myITS
Skenario 2	*Aktor login dari aplikasi client*
Kondisi Awal	Aktor belum terotentikasi dalam sistem aplikasi myITS
Data Uji	*Username* dan *password* yang terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman aplikasi *client* 2. Aktor mengisi *username* dan *password* pada halaman *login* myITS 3. Aktor menekan tombol *login*
Hasil yang diharapkan	Aktor berhasil *login* dan sistem menampilkan halaman dashboard *client*
Kondisi Akhir	Aktor berhasil *login* dan sistem menampilkan halaman dashboard *client*
Skenario 3	*Aktor salah memasukkan username dan password*
Kondisi Awal	Aktor belum terotentikasi dalam sistem aplikasi myITS
Data Uji	*Username* dan/ atau *password* yang tidak terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman myITS 2. Aktor mengisi *username* dan *password* 3. Aktor menekan tombol *login*
Hasil yang diharapkan	Aktor tidak berhasil *login* dan sistem menampilkan pesan error

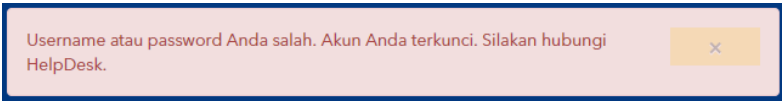
Kondisi Akhir	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Skenario 4	*Aktor gagal melakukan login sebanyak 5 kali dalam waktu 5 menit*
Kondisi Awal	Aktor belum terotentikasi dalam sistem aplikasi myITS
Data Uji	*Username* dan/ atau *password* yang tidak terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman myITS 2. Aktor mengisi *username* dan *password* 3. Aktor menekan tombol *login*
Hasil yang diharapkan	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Kondisi Akhir	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Skenario 5	*Aktor yang sebelumnya telah ter-suspend melakukan gagal login 5 kali dalam 5 menit*
Kondisi Awal	Aktor belum terotentikasi dalam sistem aplikasi myITS
Data Uji	*Username* dan/ atau *password* yang tidak terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman myITS 2. Aktor mengisi *username* dan *password* 3. Aktor menekan tombol *login*
Hasil yang diharapkan	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Kondisi Akhir	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Skenario 6	*Aktor dengan akun yang terkunci melakukan login*
Kondisi Awal	Aktor belum terotentikasi dalam sistem aplikasi myITS
Data Uji	*Username* dan/ atau *password* yang tidak terdaftar dalam sistem

Langkah Pengujian	1. Aktor mengakses halaman myITS 2. Aktor mengisi *username* dan *password* 3. Aktor menekan tombol *login*
Hasil yang diharapkan	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Kondisi Akhir	Aktor tidak berhasil *login* dan sistem menampilkan pesan error
Skenario 7	*Aktor login dari aplikasi client dan memiliki session login aktif*
Kondisi Awal	Aktor telah terotentikasi dalam sistem aplikasi myITS dan memiliki session *login* aktif
Data Uji	*Username* dan *password* yang terdaftar dalam sistem
Langkah Pengujian	1. Aktor mengakses halaman *client* 2. Aktor menekan tombol *login* dengan akun myITS
Hasil yang diharapkan	Sistem menampilkan halaman dashboard *client* tanpa menampilkan halaman *login*
Kondisi Akhir	Sistem menampilkan halaman dashboard *client* tanpa menampilkan halaman *login*

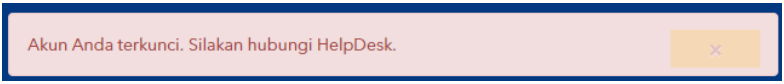
Gambar 5.1 Pesan Error Gagal *Login*



Gambar 5.2 Pesan Error Gagal *Login* dan Akun Ter-suspend



Gambar 5.3 Pesan Error Gagal *Login* dan Akun Terkunci



Gambar 5.4 Pesan Error Akun Terkunci

5.2.1.2 Kasus Uji Masuk ke Aplikasi

Kasus uji ini digunakan untuk menguji apakah aktor yang dapat masuk ke dalam aplikasi yang dipilih pada halaman dashboard myITS. Rincian kasus uji ditunjukkan pada Tabel 5.7.

Tabel 5.7 Tabel kasus uji masuk ke aplikasi

ID	**TC-002**
Kasus Penggunaan	Masuk ke Aplikasi
Nama	Pengujian masuk ke aplikasi *client*
Tujuan Pengujian	Menguji apakah aktor dapat masuk ke dalam aplikasi *client* tanpa perlu melakukan *login*
Kondisi Awal	Aktor telah terotentikasi dalam sistem dan berada pada halaman dashboard myITS
Langkah Pengujian	1. Aktor memilih aplikasi *client* pada halaman dashboard myITS
Hasil yang Diharapkan	Sistem menampilkan halaman dashboard *client* tanpa menampilkan halaman *login*
Kondisi Akhir	Sistem menampilkan halaman dashboard *client* tanpa menampilkan halaman *login*

5.2.1.3 Kasus Uji Lihat Profil

Kasus uji ini digunakan untuk menguji apakah aktor dapat melihat profilnya. Rincian kasus uji ditunjukkan pada Tabel 5.8.

Tabel 5.8 Tabel kasus uji lihat profil

ID	**TC-003**
Kasus Penggunaan	Lihat Profil

Nama	Pengujian lihat profil saya
Tujuan Pengujian	Menguji apakah aktor dapat melihat profilnya
Kondisi Awal	Aktor telah terotentikasi dalam sistem dan berada pada halaman dashboard myITS
Langkah Pengujian	1. Aktor memilih pilihan profil saya
Hasil yang Diharapkan	Sistem menampilkan halaman profil saya
Kondisi Akhir	Sistem menampilkan halaman profil saya

5.2.1.4 Kasus Uji Verifikasi Akun

Kasus uji ini digunakan untuk menguji apakah aktor dapat melakukan verifikasi akun email, email alternatif, maupun nomor ponselnya. Terdapat lima skenario pada kasus uji ini dimana rincian kasus uji ditunjukkan pada Tabel 5.9.

Tabel 5.9 Tabel kasus uji verifikasi akun

ID	**TC-004**
Kasus Penggunaan	Verifikasi Akun
Nama	Pengujian verifikasi akun
Tujuan Pengujian	Menguji apakah pengguna dapat melakukan verifikasi akun email, email alternatif, maupun nomor ponselnya.
Skenario 1	*Aktor melakukan verifikasi akun email*
Kondisi Awal	Aktor telah terotentikasi dan akun email belum terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol verifikasi ulang pada akun email pada halaman profil saya

	2. Aktor mendapatkan kode verifikasi pada akun emailnya 3. Aktor menginputkan kode verifikasi 4. Aktor menekan tombol submit
Hasil yang Diharapkan	Akun email aktor berhasil terverifikasi
Kondisi Akhir	Akun email aktor berhasil terverifikasi
Skenario 2	*Aktor melakukan verifikasi akun email alternatif*
Kondisi Awal	Aktor telah terotentikasi dan akun email alternatif belum terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol verifikasi ulang pada akun email alternatif pada halaman profil saya 2. Aktor mendapatkan kode verifikasi pada akun email alternatifnya 3. Aktor menginputkan kode verifikasi 4. Aktor menekan tombol submit
Hasil yang diharapkan	Akun email alternatif aktor berhasil terverifikasi
Kondisi Akhir	Akun email alternatif aktor berhasil terverifikasi
Skenario 3	*Aktor melakukan verifikasi akun nomor ponsel*
Kondisi Awal	Aktor telah terotentikasi dan akun nomor ponsel belum terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol verifikasi ulang pada akun nomor ponsel pada halaman profil saya 2. Aktor mendapatkan SMS kode verifikasi pada ponselnya 3. Aktor menginputkan kode verifikasi 4. Aktor menekan tombol submit

Hasil yang diharapkan	Akun nomor ponsel aktor berhasil terverifikasi
Kondisi Akhir	Akun nomor ponsel aktor berhasil terverifikasi
Skenario 4	*Aktor tidak mendapatkan kode verifikasi*
Kondisi Awal	Aktor telah menekan tombol verifikasi ulang pada suatu akun dan tidak mendapatkan kode verifikasi lebih dari waktu 5 menit
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol kirim ulang kode 2. Aktor mendapatkan kode verifikasi pada akun yang dipilih 3. Aktor menginputkan kode verifikasi 4. Aktor menekan tombol submit
Hasil yang diharapkan	Akun aktor berhasil terverifikasi
Kondisi Akhir	Akun aktor berhasil terverifikasi
Skenario 5	*Aktor menginputkan kode verifikasi yang tidak sesuai*
Kondisi Awal	Aktor telah terotentikasi dan akun yang dipilih belum terotentikasi
Data Uji	Kode verifikasi yang tidak sesuai
Langkah Pengujian	1. Aktor menekan tombol verifikasi ulang pada akun yang dipilih 2. Aktor mendapatkan kode verifikasi pada akun yang dipilih 3. Aktor menginputkan kode verifikasi 4. Aktor menekan tombol submit
Hasil yang diharapkan	Akun aktor gagal terverifikasi dan sistem menampilkan pesan error
Kondisi Akhir	Akun aktor gagal terverifikasi dan sistem menampilkan pesan error

5.2.1.5 Kasus Uji Reset *Password*

Kasus uji ini digunakan untuk menguji apakah pengguna dapat melakukan pengaturan ulang *password* saat lupa *password* menggunakan akun email atau nomor ponsel yang telah terverifikasi. Terdapat enam skenario pada kasus uji ini yang secara rinci ditunjukkan pada Tabel 5.10.

Tabel 5.10 Tabel kasus uji reset *password*

ID	**TC-005**
Kasus Penggunaan	Reset *Password*
Nama	Pengujian reset *password*
Tujuan Pengujian	Menguji apakah pengguna dapat melakukan pengaturan ulang *password* saat lupa *password*
Skenario 1	*Aktor melakukan reset password menggunakan akun email*
Kondisi Awal	Aktor belum terotentikasi dan akun email telah terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukkan *username* 3. Aktor memilih akun email 4. Aktor mendapatkan kode verifikasi pada akun emailnya 5. Aktor menginputkan kode verifikasi 6. Aktor menekan tombol submit 7. Aktor mereset ulang *password*nya 8. Aktor menekan tombol submit
Hasil yang Diharapkan	*Password* pengguna berhasil diperbaharui
Kondisi Akhir	*Password* pengguna berhasil diperbaharui

Skenario 2	*Aktor melakukan reset password menggunakan akun nomor ponsel*
Kondisi Awal	Aktor belum terotentikasi dan akun nomor ponsel telah terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukan *username* 3. Aktor memilih akun nomor ponsel 4. Aktor mendapatkan SMS kode verifikasi pada ponselnya 5. Aktor menginputkan kode verifikasi 6. Aktor menekan tombol submit 7. Aktor mereset ulang *password*nya 8. Aktor menekan tombol submit
Hasil yang diharapkan	*Password* pengguna berhasil diperbaharui
Kondisi Akhir	*Password* pengguna berhasil diperbaharui
Skenario 3	*Aktor tidak mendapatkan kode verifikasi*
Kondisi Awal	Aktor belum terotentikasi dan salah satu akun telah terverifikasi
Data Uji	Kode verifikasi yang sesuai
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukkan *username* 3. Aktor memilih akun yang diinginkan 4. Aktor tidak mendapatkan kode verifikasi pada akunnya dalam rentang waktu 5 menit 5. Aktor memilih tombol kirim ulang 6. Aktor mendapatkan kode verifikasi pada akunnya 7. Aktor menginputkan kode verifikasi 8. Aktor menekan tombol submit 9. Aktor mereset ulang *password*nya

	10. Aktor menekan tombol submit
Hasil yang diharapkan	*Password* pengguna berhasil diperbaharui
Kondisi Akhir	*Password* pengguna berhasil diperbaharui
Skenario 4	*Aktor menginputkan kode verifikasi yang tidak sesuai*
Kondisi Awal	Aktor belum terotentikasi dan salah satu akun telah terverifikasi
Data Uji	Kode verifikasi yang tidak sesuai
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukan *username* 3. Aktor memilih akun yang diinginkan 4. Aktor mendapatkan kode verifikasi pada akunnya 5. Aktor menginputkan kode verifikasi 6. Aktor menekan tombol submit
Hasil yang diharapkan	Sistem menampilkan pesan error
Kondisi Akhir	Sistem menampilkan pesan error
Skenario 5	*Aktor tidak memiliki akun yang telah terverifikasi*
Kondisi Awal	Aktor belum terotentikasi dan kedua akun belum terverifikasi
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukkan *username*
Hasil yang diharapkan	Sistem menampilkan pesan error tidak dapat melakukan reset *password*
Kondisi Akhir	Sistem menampilkan pesan error tidak dapat melakukan reset *password*
Skenario 6	*Username yang diinputkan tidak terdaftar dalam sistem*

Kondisi Awal	Aktor belum terotentikasi
Langkah Pengujian	1. Aktor menekan tombol lupa *password* pada halaman *login* 2. Aktor memasukkan *username*
Hasil yang diharapkan	Sistem menampilkan pesan error *username* tidak terdaftar
Kondisi Akhir	Sistem menampilkan pesan error *username* tidak terdaftar

5.2.1.6 Kasus Uji Edit Profil

Kasus uji ini digunakan untuk menguji apakah pengguna dapat melakukan pembaharuan terhadap data email alternatif, nama panggilan, locale, zoneinfo, dan foto profilnya. Terdapat lima skenario pada kasus uji ini yang secara rinci ditunjukkan pada Tabel 5.11.

Tabel 5.11 Tabel kasus uji menghapus edit profil.

ID	**TC-006**
Kasus Penggunaan	Edit Profil
Nama	Pengujian edit profil
Tujuan Pengujian	Menguji apakah pengguna dapat melakukan permbaharuan foto profil, email alternatif, nama panggilan, locale, dan zoneinfo.
Skenario 1	*Aktor melakukan pembaharuan foto profil*
Kondisi Awal	Aktor telah terotentikasi dan berada pada halaman profil saya
Data Uji	Foto berukuran kurang dari 500Kb
Langkah Pengujian	1. Aktor menekan tombol ubah foto 2. Aktor memilih foto 3. Aktor menekan tombol submit
Hasil yang Diharapkan	Foto profil pengguna berhasil diperbaharui

Kondisi Akhir	Foto profil pengguna berhasil diperbaharui
Skenario 2	*Aktor melakukan pembaharuan email alternatif*
Kondisi Awal	Aktor telah terotentikasi dan berada pada halaman profil saya
Data Uji	Data email alternatif yang sesuai
Langkah Pengujian	1. Aktor menekan tombol edit email alternatif 2. Aktor menginputkan email alternatif baru 3. Aktor menekan tombol sumbit
Hasil yang diharapkan	Email alternatif pengguna berhasil diperbaharui
Kondisi Akhir	Email alternatif pengguna berhasil diperbaharui
Skenario 3	*Aktor melakukan pembaharuan nama panggilan*
Kondisi Awal	Aktor telah terotentikasi dan berada pada halaman profil saya
Data Uji	Data nama panggilan
Langkah Pengujian	1. Aktor menekan tombol edit nama panggilan 2. Aktor menginputkan nama panggilan baru 3. Aktor menekan tombol sumbit
Hasil yang diharapkan	Nama panggilan pengguna berhasil diperbaharui
Kondisi Akhir	Nama panggilan pengguna berhasil diperbaharui
Skenario 4	*Aktor melakukan pembaharuan locale*
Kondisi Awal	Aktor telah terotentikasi dan berada pada halaman profil saya
Langkah Pengujian	1. Aktor menekan tombol edit locale 2. Aktor memilih pilihan data locale 3. Aktor menekan tombol sumbit
Hasil yang diharapkan	Locale pengguna berhasil diperbaharui
Kondisi Akhir	Locale pengguna berhasil diperbaharui

Skenario 5	*Aktor melakukan pembaharuan zoneinfo*
Kondisi Awal	Aktor telah terotentikasi dan berada pada halaman profil saya
Langkah Pengujian	1. Aktor menekan tombol edit zoneinfo 2. Aktor memilih pilihan data zoneinfo 3. Aktor menekan tombol sumbit
Hasil yang diharapkan	Zoneinfo pengguna berhasil diperbaharui
Kondisi Akhir	Zoneinfo pengguna berhasil diperbaharui

5.2.1.7 Kasus Uji *Logout*

Kasus uji ini digunakan untuk menguji apakah pengguna dapat melakukan *logout*. Rincian kasus uji ditunjukkan pada Tabel 5.12.

Tabel 5.12 Tabel kasus uji *logout*

ID	**TC-007**
Kasus Penggunaan	*Logout*
Nama	Pengujian *logout*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan *logout*
Kondisi Awal	Aktor telah terotentikasi dalam sistem dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih pilihan *logout*
Hasil yang Diharapkan	Sistem menampilkan halaman *login* myITS
Kondisi Akhir	Sistem menampilkan halaman *login* myITS

5.2.1.8 Kasus Uji Lihat Aplikasi

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melihat aplikasi *client* yang telah terdaftar dalam sistem. Terdapat dua skenario dalam kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.13.

Tabel 5.13 Tabel kasus uji lihat aplikasi

ID	**TC-008**
Kasus Penggunaan	Lihat Aplikasi
Nama	Pengujian lihat aplikasi
Tujuan Pengujian	Menguji apakah aktor dapat melihat daftar aplikasi
Skenario 1	*Aktor dengan peran developer*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *developer* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management 2. Aktor memilih lihat detail aplikasi pada daftar aplikasi yang diinginkan
Hasil yang Diharapkan	Sistem menampilkan halaman detail aplikasi yang dipilih aktor
Kondisi Akhir	Sistem menampilkan halaman detail aplikasi yang dipilih aktor
Skenario 2	*Aktor dengan peran super admin*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management 2. Aktor memilih menu client 3. Aktor memilih lihat detail aplikasi pada daftar *client* yang diinginkan
Hasil yang diharapkan	Sistem menampilkan halaman detail aplikasi yang dipilih aktor

Kondisi Akhir	Sistem menampilkan halaman detail aplikasi yang dipilih aktor

5.2.1.9 Kasus Uji Unduh Public Key

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat mengunduh *public key* aplikasi *client* yang telah terdaftar dalam sistem. Terdapat dua skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.14.

Tabel 5.14 Tabel kasus uji unduh public key

ID	**TC-009**
Kasus Penggunaan	Unduh Public Key
Nama	Pengujian unduh public key
Tujuan Pengujian	Menguji apakah aktor dapat mengunduh *public key* aplikasi
Skenario 1	*Aktor dengan peran developer*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *developer* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management 2. Aktor memilih tombol unduh *public key* pada daftar aplikasi yang diinginkan
Hasil yang Diharapkan	File *public key* terunduh
Kondisi Akhir	File *public key* terunduh
Skenario 2	*Aktor dengan peran super admin*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management

	2. Aktor memilih menu client 3. Aktor memilih lihat detail aplikasi pada daftar *client* yang diinginkan 4. Aktor memilih tombol *unduh public key*
Hasil yang diharapkan	File *public key* terunduh
Kondisi Akhir	File *public key* terunduh

5.2.1.10 Kasus Uji Generate Access Token

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat men-*generate access token* untuk aplikasi *client* yang telah terdaftar dalam sistem. Terdapat dua skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.15.

Tabel 5.15 Tabel kasus uji generate access token

ID	**TC-010**
Kasus Penggunaan	Generate Access Token
Nama	Pengujian generate access token
Tujuan Pengujian	Menguji apakah aktor dapat melakukan *generate access token* aplikasi *client*
Skenario 1	*Aktor dengan peran developer*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *developer* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management 2. Aktor memilih tombol *access token* pada daftar aplikasi yang diinginkan 3. Aktor memilih tombol *generate*

Hasil yang Diharapkan	*Access token* terbuat dan sistem menampilkan pemberitahuan *access token* berhasil dibuat
Kondisi Akhir	*Access token* terbuat dan sistem menampilkan pemberitahuan *access token* berhasil dibuat
Skenario 2	*Aktor dengan peran super admin*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman myITS
Langkah Pengujian	1. Aktor memilih aplikasi myITS Security Management 2. Aktor memilih menu client 3. Aktor memilih tombol *access token* pada daftar *client* yang diinginkan 4. Aktor memilih tombol *generate*
Hasil yang diharapkan	*Access token* terbuat dan sistem menampilkan pemberitahuan *access token* berhasil dibuat
Kondisi Akhir	*Access token* terbuat dan sistem menampilkan pemberitahuan *access token* berhasil dibuat

5.2.1.11 Kasus Uji Atur User Role

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengaturan *user role* pada aplikasi *client* yang telah terdaftar dalam sistem. Terdapat tiga skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.16.

Tabel 5.16 Tabel kasus uji atur user role

ID	**TC-011**
Kasus Penggunaan	Atur User Role
Nama	Pengujian atur user role
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengaturan *user role* pada suatu aplikasi *client*

Skenario 1	*Aktor melakukan penambahan user role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengaturan *user role*
Langkah Pengujian	1. Aktor memilih tombol tambah *user role* 2. Aktor mengisi form penambahan *user role* 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *user role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *user role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan pembaharuan user role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengaturan *user role*
Langkah Pengujian	1. Aktor memilih tombol *edit* pada daftar *user role* yang diinginkan 2. Aktor mengisi form pembaharuan *user role* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *user role* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *user role* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 3	*Aktor melakukan penghapusan user role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengaturan *user role*
Langkah Pengujian	1. Aktor memilih tombol *hapus* pada daftar *user role* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!

Hasil yang diharapkan	Data *user role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *user role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.12 Kasus Uji Kelola Menu

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengelolaan *menu* untuk setiap aplikasi *client* yang telah terdaftar dalam sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.17.

Tabel 5.17 Tabel kasus uji kelola menu

ID	**TC-012**
Kasus Penggunaan	Kelola Menu
Nama	Pengujian kelola menu
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan menu pada suatu aplikasi *client*
Skenario 1	*Aktor melakukan penambahan menu*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih tombol tambah *menu* 2. Aktor mengisi form penambahan *menu* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *menu* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *menu* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan menu dengan isian tidak lengkap*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih tombol tambah *menu* 2. Aktor mengisi form penambahan *menu* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan menu*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih tombol edit *menu* pada daftar menu yang diinginkan 2. Aktor mengisi form pembaharuan *menu* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *menu* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *menu* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan menu*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih tombol hapus *menu* pada daftar menu yang diinginkan 2. Aktor memilih pilihan Ya, hapus!

Hasil yang diharapkan	Data *menu* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *menu* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.13 Kasus Uji Atur Menu Role

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengaturan *menu role* untuk setiap *menu* yang telah terdaftar dalam sistem. Terdapat tiga skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.18.

Tabel 5.18 Tabel kasus uji atur menu role

ID	**TC-013**
Kasus Penggunaan	Atur Menu Role
Nama	Pengujian atur menu role
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengaturan *menu role* pada suatu menu
Skenario 1	*Aktor melakukan penambahan menu role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih *role menu* pada daftar *menu* yang diinginkan 2. Aktor memilih tombol tambah *menu role* 3. Aktor mengisi form penambahan *menu role* dengan lengkap 4. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *menu role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan

Kondisi Akhir	Data *menu role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan pembaharuan menu role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih *role menu* pada daftar *menu* yang diinginkan 2. Aktor memilih tombol edit pada daftar *menu role* yang diinginkan 3. Aktor mengisi form pembaharuan *menu role* 4. Aktor memilih tombol *submit*
Hasil yang diharapkan	Perbaharuan data *menu role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Perbaharuan data *menu role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 3	*Aktor melakukan penghapusan menu role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *menu*
Langkah Pengujian	1. Aktor memilih *role menu* pada daftar *menu* yang diinginkan 2. Aktor memilih tombol hapus *menu* pada daftar menu yang diinginkan 3. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *menu role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *menu role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.14 Kasus Uji Kelola API

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengaturan *API* untuk setiap aplikasi *client* yang telah terdaftar dalam sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.19

Tabel 5.19 Tabel kasus uji kelola API

ID	**TC-014**
Kasus Penggunaan	Kelola API
Nama	Pengujian API
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan API pada suatu aplikasi *client*
Skenario 1	*Aktor melakukan penambahan API*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan API
Langkah Pengujian	1. Aktor memilih tombol tambah *API* 2. Aktor mengisi form penambahan API dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *API* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *API* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan API dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *API*

Langkah Pengujian	1. Aktor memilih tombol tambah *API* 2. Aktor mengisi form penambahan *API* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan API*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *API*
Langkah Pengujian	1. Aktor memilih tombol edit *API* pada daftar *API* yang diinginkan 2. Aktor mengisi form pembaharuan*API* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *API* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *API* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan API*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *API*
Langkah Pengujian	1. Aktor memilih tombol hapus *API* pada daftar *API* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *API* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *API* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.15 Kasus Uji Kelola Resource

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengelolaan *resource* untuk setiap aplikasi *client* yang telah terdaftar dalam sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.20.

Tabel 5.20 Tabel kasus uji kelola resource

ID	**TC-015**
Kasus Penggunaan	Kelola Resource
Nama	Pengujian kelola resource
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *resource* pada suatu *API*
Skenario 1	*Aktor melakukan penambahan resource*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih tombol tambah *resource* 2. Aktor mengisi form penambahan *resource* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *resource* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *resource* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan resource dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*

Langkah Pengujian	1. Aktor memilih tombol tambah *resource* 2. Aktor mengisi form penambahan *resource* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan resource*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih tombol edit *resource* pada daftar *resource* yang diinginkan 2. Aktor mengisi form pembaharuan *resource* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *resource* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *resource* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan resource*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih tombol hapus *resource* pada daftar *resource* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *resource* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *resource* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.16 Kasus Uji Atur Resource Role

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *developer* dapat melakukan pengaturan *resource role* untuk setiap *resource* yang telah terdaftar dalam sistem. Terdapat tiga skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.21.

Tabel 5.21 Tabel kasus uji atur resource role

ID	**TC-016**
Kasus Penggunaan	Atur Resource Role
Nama	Pengujian atur *resource role*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengaturan *resource role* pada suatu menu
Skenario 1	*Aktor melakukan penambahan resource role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih *resource role* pada daftar *resource* yang diinginkan 2. Aktor memilih tombol tambah *resource role* 3. Aktor mengisi form penambahan *resource role* dengan lengkap 4. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *resource role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *resource role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan pembaharuan resource role*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih *resource role* pada daftar *resource* yang diinginkan 2. Aktor memilih tombol edit pada daftar *resource role* yang diinginkan 3. Aktor mengisi form pembaharuan *resource role* 4. Aktor memilih tombol *submit*
Hasil yang diharapkan	Perbaharuan data *resource role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Perbaharuan data *resource role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 3	*Aktor melakukan penghapusan resource role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *resource*
Langkah Pengujian	1. Aktor memilih *resource role* pada daftar *resource* yang diinginkan 2. Aktor memilih tombol hapus *resource role* pada daftar *resource* yang diinginkan 3. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *resource role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *resource role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.17 Kasus Uji Kelola User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melakukan pengelolaan *user* pada sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.22.

Tabel 5.22 Tabel kasus uji kelola user

ID	**TC-017**
Kasus Penggunaan	Kelola User
Nama	Pengujian kelola *user*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *user* pada sistem
Skenario 1	*Aktor melakukan penambahan user*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol tambah *user* 2. Aktor mengisi form penambahan *user* dengan lengkap dan sesuai 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *user* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *user* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan user dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol tambah *user* 2. Aktor mengisi form penambahan *user* dengan tidak lengkap 3. Aktor memilih tombol *submit*

Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan penambahan user dengan isian yang tidak sesuai*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol tambah *user* 2. Aktor mengisi form penambahan *user* dengan data tidak sesuai format 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan pada isian yang tidak sesuai
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan pada isian yang tidak sesuai
Skenario 4	*Aktor melakukan pembaharuan user*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol edit *user* pada daftar *user* yang diinginkan 2. Aktor mengisi form pembaharuan *user* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *user* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *user* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui

5.2.1.18 Kasus Uji Lihat Log User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melihat *log user* pada setiap *user* yang terdaftar dalam sistem. Rincian kasus uji ditunjukkan pada Tabel 5.23

Tabel 5.23 Tabel kasus uji lihat log user

ID	**TC-018**
Kasus Penggunaan	Lihat Log User
Nama	Pengujian lihat *log user*
Tujuan Pengujian	Menguji apakah aktor dapat melihat *log user* pada *user* yang diinginkan
Kondisi Awal	Aktor telah terotentikasi dalam sistem dengan peran *super admin* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol lihat log
Hasil yang Diharapkan	Sistem menampilkan halaman log *user* yang dipih
Kondisi Akhir	Sistem menampilkan halaman log *user* yang dipih

5.2.1.19 Kasus Uji Kelola Scope

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melakukan pengelolaan *scope* pada sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.24.

Tabel 5.24 Tabel kasus uji kelola scope

ID	**TC-019**

Kasus Penggunaan	Kelola Scope
Nama	Pengujian kelola *scope*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *scope* pada sistem
Skenario 1	*Aktor melakukan penambahan scope*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *scope*
Langkah Pengujian	1. Aktor memilih tombol tambah *scope* 2. Aktor mengisi form penambahan *scope* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *scope* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *scope* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan scope dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *scope*
Langkah Pengujian	1. Aktor memilih tombol tambah *scope* 2. Aktor mengisi form penambahan *scope* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan scope*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *scope*
Langkah Pengujian	1. Aktor memilih tombol edit *scope* pada daftar *scope* yang diinginkan 2. Aktor mengisi form pembaharuan *scope* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *scope* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *scope* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan scope*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *scope*
Langkah Pengujian	1. Aktor memilih tombol hapus *scope* pada daftar *scope* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *scope* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *scope* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.20 Kasus Uji Kelola Role

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melakukan pengelolaan *role* pada sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.25.

Tabel 5.25 Tabel kasus uji kelola role

ID	**TC-020**
Kasus Penggunaan	Kelola Role
Nama	Pengujian kelola *role*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *role* pada sistem
Skenario 1	*Aktor melakukan penambahan role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *role*
Langkah Pengujian	1. Aktor memilih tombol tambah *role* 2. Aktor mengisi form penambahan *role* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *role* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan role dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *role*
Langkah Pengujian	1. Aktor memilih tombol tambah *role* 2. Aktor mengisi form penambahan *role* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan role*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *role*
Langkah Pengujian	1. Aktor memilih tombol edit *role* pada daftar *role* yang diinginkan 2. Aktor mengisi form pembaharuan *role* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *role* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *role* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan role*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *role*
Langkah Pengujian	1. Aktor memilih tombol hapus *role* pada daftar *role* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *role* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.21 Kasus Uji Kelola Client

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melakukan pengelolaan *client* pada sistem. Terdapat tiga skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.26.

Tabel 5.26 Tabel kasus uji kelola client

ID	**TC-021**
Kasus Penggunaan	Kelola Client
Nama	Pengujian kelola *client*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *client* pada sistem
Skenario 1	*Aktor melakukan penambahan client*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *client*
Langkah Pengujian	1. Aktor memilih tombol tambah *client* 2. Aktor mengisi form penambahan *client* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *client* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *client* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan client dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *client*
Langkah Pengujian	1. Aktor memilih tombol tambah *client* 2. Aktor mengisi form penambahan *client* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan client*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *developer* dan berada pada halaman pengelolaan *client*
Langkah Pengujian	1. Aktor memilih tombol edit *client* pada daftar *client* yang diinginkan 2. Aktor mengisi form pembaharuan *client* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data c*lient* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Kondisi Akhir	Pembaharuan data *client* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui

5.2.1.22 Kasus Uji Kelola Unit

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dapat melakukan pengelolaan *unit* pada sistem. Terdapat empat skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.27.

Tabel 5.27 Tabel kasus uji kelola unit

ID	**TC-022**
Kasus Penggunaan	Kelola Unit
Nama	Pengujian kelola *unit*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengelolaan *unit* pada sistem
Skenario 1	*Aktor melakukan penambahan unit*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *unit*

Langkah Pengujian	1. Aktor memilih tombol tambah *unit* 2. Aktor mengisi form penambahan *unit* dengan lengkap 3. Aktor memilih tombol *submit*
Hasil yang Diharapkan	Data *unit* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Kondisi Akhir	Data *unit* disimpan dan sistem menampilkan pemberitahuan data berhasil disimpan
Skenario 2	*Aktor melakukan penambahan unit dengan isian tidak lengkap*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *unit*
Langkah Pengujian	1. Aktor memilih tombol tambah *unit* 2. Aktor mengisi form penambahan *unit* dengan tidak lengkap 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Kondisi Akhir	Sistem menampilkan pesan pemberitahuan '*please fill out this field*' pada isian yang wajib diisi
Skenario 3	*Aktor melakukan pembaharuan unit*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *unit*
Langkah Pengujian	1. Aktor memilih tombol edit *unit* pada daftar *unit* yang diinginkan 2. Aktor mengisi form pembaharuan *unit* 3. Aktor memilih tombol *submit*
Hasil yang diharapkan	Pembaharuan data *unit* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui

Kondisi Akhir	Pembaharuan data *unit* disimpan dan sistem menampilkan pemberitahuan data berhasil diperbaharui
Skenario 4	*Aktor melakukan penghapusan unit*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* dan berada pada halaman pengelolaan *unit*
Langkah Pengujian	1. Aktor memilih tombol hapus *unit* pada daftar *unit* yang diinginkan 2. Aktor memilih pilihan Ya, hapus!
Hasil yang diharapkan	Data *unit* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus
Kondisi Akhir	Data *unit* dihapus dan sistem menampilkan pemberitahuan data berhasil dihapus

5.2.1.23 Kasus Uji Lihat User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *helpdesk* dapat melihat data *user* pada sistem. Rincian kasus uji ditunjukkan pada Tabel 5.28.

Tabel 5.28 Tabel kasus uji lihat user

ID	**TC-023**
Kasus Penggunaan	Lihat User
Nama	Pengujian lihat *user*
Tujuan Pengujian	Menguji apakah aktor dapat melihat data *user* yang tersimpan dalam sistem
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman dashboard myITS Security Management

Langkah Pengujian	1. Aktor memilih menu *user* 2. Aktor memilih tombol lihat detail pada *user* yang diinginkan
Hasil yang Diharapkan	Sistem menampilkan detail *user* yang dipilih aktor
Kondisi Akhir	Sistem menampilkan detail *user* yang dipilih aktor

5.2.1.24 Kasus Uji Edit Kontak User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *helpdesk* dapat melakukan pembaharuan data email dan alternatif email setiap *user* pada sistem. Terdapat empat skenario dalam kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.29

Tabel 5.29 Tabel kasus uji edit email user

ID	**TC-024**
Kasus Penggunaan	Edit Kontak User
Nama	Pengujian edit kontak *user*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pembaharuan kontak *user* yang dipilih
Skenario 1	*Aktor mengisi isian pembaharuan kontak user dengan sesuai*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol *edit* pada salah satu data kontak *user* 2. Aktor mengisi form pembaharuan kontak *user* dengan data yang sesuai
Hasil yang Diharapkan	Pembaharuan data kontak *user* tersimpan dan sistem menampilkan pemberitahuan pembaharuan berhasil dilakukan
Kondisi Akhir	Pembaharuan data kontak *user* tersimpan dan sistem menampilkan pemberitahuan pembaharuan berhasil dilakukan
Skenario 2	*Aktor mengisi isian pembaharuan kontak user dengan data tidak sesuai*

Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol *edit* pada salah satu data kontak *user* 2. Aktor mengisi form pembaharuan kontak *user* dengan data yang tidak sesuai
Hasil yang diharapkan	Sistem menampilkan pemberitahuan pada data yang tidak sesuai
Kondisi Akhir	Sistem menampilkan pemberitahuan pada data yang tidak sesuai

5.2.1.25 Kasus Uji Reset *Password* User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *helpdesk* dapat melakukan pembaharuan *password* setiap *user* pada sistem. Terdapat dua skenario pada kasus uji ini. Rincian kasus uji ditunjukkan pada Tabel 5.30.

Tabel 5.30 Tabel kasus uji reset *password* user

ID	**TC-025**
Kasus Penggunaan	Reset *Password* User
Nama	Pengujian edit *password user*
Tujuan Pengujian	Menguji apakah aktor dapat melakukan pengaturan ulang *password user* yang dipilih
Skenario 1	*Aktor mengisi isian pengaturan ulang password user dengan sesuai*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman pengelolaan *user*

Langkah Pengujian	1. Aktor memilih tombol *reset password* pada *user* yang diinginkan 2. Aktor mengisi form pengaturan ulang *password user* dengan data yang sesuai
Hasil yang Diharapkan	Pembaharuan *password user* tersimpan dan sistem menampilkan pemberitahuan pembaharuan berhasil dilakukan
Kondisi Akhir	Pembaharuan *password user* tersimpan dan sistem menampilkan pemberitahuan pembaharuan berhasil dilakukan
Skenario 2	*Aktor mengisi isian pengaturan ulang password user dengan data tidak sesuai*
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol *reset password* pada *user* yang diinginkan 2. Aktor mengisi form pengaturan ulang *password user* dengan data yang tidak sesuai
Hasil yang diharapkan	Sistem menampilkan pemberitahuan pada data yang tidak sesuai
Kondisi Akhir	Sistem menampilkan pemberitahuan pada data yang tidak sesuai

5.2.1.26 Kasus Uji Unlock User

Kasus uji ini digunakan untuk menguji apakah aktor dengan peran *super admin* dan *helpdesk* dapat membuka akun *user* yang terkunci pada sistem. Rincian kasus uji ditunjukkan pada Tabel 5.31.

Tabel 5.31 Tabel kasus uji unlock user

ID	**TC-026**

Kasus Penggunaan	Unlock User
Nama	Pengujian *unlock user*
Tujuan Pengujian	Menguji apakah aktor dapat membuka akun *user* yang terkunci
Kondisi Awal	Aktor telah terotentikasi dalam sistem sebagai *super admin* atau *helpdesk* dan berada pada halaman pengelolaan *user*
Langkah Pengujian	1. Aktor memilih tombol *unlock* pada *user* terkunci yang diinginkan 2. Aktor memilih pilihan Ya
Hasil yang Diharapkan	Akun *user* dibuka dan sistem menampilkan pemberitahuan akun *user* berhasil di *unlock*
Kondisi Akhir	Akun *user* dibuka dan sistem menampilkan pemberitahuan akun *user* berhasil di *unlock*

5.2.1.27 Kasus Uji Insert Device Token

Kasus uji ini digunakan untuk menguji apakah aktor (aplikasi *client*) dapat mengakses API *Insert Device Token* dan menyimpan data *device token* dalam sistem. Terdapat dua skenario dalam kasus uji ini. Rincian kasus uji ditunjukan pada Tabel 5.32.

Tabel 5.32 Tabel kasus uji insert device token

ID	**TC-027**
Kasus Penggunaan	Insert Device Token
Nama	Pengujian *insert device token*
Tujuan Pengujian	Menguji apakah aktor (*client*) dapat melakukan *menambahkan device token* pada sistem dengan mengakses API myITS
Skenario 1	*Aktor mengirimkan request POST dengan header access token yang valid*

Kondisi Awal	Aktor atau aplikasi *client* telah terdaftar pada sistem dan memiliki *access token*
Langkah Pengujian	1. Aktor mengirimkan *POST request* berisi parameter data dan *header* berisi *access token* yang valid
Hasil yang Diharapkan	Sistem mengirimkan pesan *response* berhasil
Kondisi Akhir	Sistem mengirimkan pesan *response* berhasil
Skenario 2	*Aktor mengirimkan request POST dengan header access token yang tidak valid*
Kondisi Awal	Aktor atau aplikasi *client* telah terdaftar pada sistem dan memiliki *access token*
Langkah Pengujian	1. Aktor mengirimkan *POST request* berisi parameter data dan *header* berisi *access token* yang tidak valid
Hasil yang diharapkan	Sistem mengirimkan pesan *response invalid access token*
Kondisi Akhir	Sistem mengirimkan pesan *response invalid access token*

5.2.2 Pengujian Hak Akses

Uji coba ini dilakukan untuk menguji apakah pengaturan hak akses terpusat pada sistem atau RBAC terpusat berhasil diterapkan dan dapat digunakan oleh aplikasi. Aspek pengujian yang dilihat pada pengujian ini adalah apakah suatu menu hanya dapat diakses oleh peran tertentu yang sudah dinyatakan saja dan pembatasan aksi yang dilakukan setiap peran. Peran-peran yang sudah terdaftar dalam sistem ditunjukan pada Tabel 5.33. Daftar menu-menu pada aplikasi myITS Security Management dengan hak aksesnya ditunjukan pada Tabel 5.34. Peran yang memiliki hak akses untuk masuk ke aplikasi myITS Security Management

adalah Super Admin, Developer, dan Helpdesk. End-user tidak dapat mengakses aplikasi myITS Security Management.

Tabel 5.33 Tabel daftar peran

No	Nama Peran
1	Super Admin
2	Developer
3	Helpdesk
4	End-user

Tabel 5.34 Tabel daftar akses menu pada myITS Security Management

No	Nama Menu	Peran	Hak Akses
1	Client	Super Admin	Insert, Update
2	User	Super Admin	Insert, Update
		Helpdesk	View, Update
3	Unit	Super Admin	Insert, Update, Delete
4	Role	Super Admin	Insert, Update, Delete
5	Scope	Super Admin	Insert, Update, Delete
6	Menu	Super Admin	Insert, Update, Delete
		Developer	Insert, Update, Delete
7	API	Super Admin	Insert, Update, Delete
		Developer	Insert, Update, Delete

5.3 Evaluasi

Pada Subbab ini dijelaskan hasil dari pengujian yang dilakukan pada Subbab sebelumnya. Sama seperti Subbab

sebelumnya, evaluasi dibagi menjadi dua, yaitu evaluasi fungsional sistem dan evaluasi hak akses.

5.3.1 Evaluasi Fungsionalitas Sistem

Evaluasi ini adalah hasil dari pengujian kasus uji pada Subbab 5.2.1. Hasil dinyatakan dalam status terpenuhi atau tidak. Hasil tersebut ditunjukkan pada Tabel 5.35.

Tabel 5.35 Tabel evaluasi kasus pengujian fungsionalitas sistem

No.	Kode Kasus Pengujian	Terpenuhi
1	TC-001	√
2	TC-002	√
3	TC-003	√
4	TC-004	√
5	TC-005	√
6	TC-006	√
7	TC-007	√
8	TC-008	√
9	TC-009	√
10	TC-010	√
11	TC-011	√
12	TC-012	√
13	TC-013	√
14	TC-014	√
15	TC-015	√
16	TC-016	√
17	TC-017	√
18	TC-018	√
19	TC-019	√
20	TC-020	√
21	TC-021	√
22	TC-022	√
23	TC-023	√
24	TC-024	√
25	TC-025	√

26	TC-026	√
27	TC-027	√

Berdasarkan Tabel 5.35 dapat disimpulkan bahwa seluruh fungsional sistem dapat dijalankan oleh aktor dan semua kasus uji coba terpenuhi.

5.3.2 Evaluasi Hak Akses

Evaluasi hak akses adalah hasil dari uji coba hak akses. Hasil yang ditampilkan yaitu berupa tabel yang berisi nama menu pada aplikasi myITS Security Management dengan rincian apakah suatu peran dapat mengakses menu tersebut atau tidak. Tabel tersebut ditunjukan pada Tabel 5.36.

Tabel 5.36 Tabel evaluasi hak akses untuk aplikasi myITS Security Management

No	Nama Menu	Hak Akses	Super Admin	Deve-loper	Help-desk
1	Client	View	√	X	X
		Insert	√	X	X
		Update	√	X	X
		Delete	X	X	X
2	User	View	√	X	√
		Insert	√	X	X
		Update	√	X	√
		Delete	X	X	X
3	Role	View	√	X	X
		Insert	√	X	X
		Update	√	X	X
		Delete	√	X	X
4	Unit	View	√	X	X

		Insert	√	X	X
		Update	√	X	X
		Delete	√	X	X
5	Scope	View	√	X	X
		Insert	√	X	X
		Update	√	X	X
		Delete	√	X	X
6	Menu	View	√	√	X
		Insert	√	√	X
		Update	√	√	X
		Delete	√	√	X
7	API	View	√	√	X
		Insert	√	√	X
		Update	√	√	X
		Delete	√	√	X

Berdasarkan tabel evaluasi hak akses yaitu Tabel 5.36, implementasi RBAC terpusat untuk membatasi akses menu pada setiap peran sudah berhasil.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas kesimpulan yang dapat diambil dari hasil uji coba dan perancangan perangkat lunak sebagai jawaban dari rumusan masalah yang telah dikemukakan dan saran yang berisi pengembangan yang dapat dilakukan lebih lanjut untuk menyempurnakan perangkat lunak.

6.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari proses pengembangan dan hasil uji coba.

1. Sistem yang dibangun mengimplementasikan sistem otentikasi dan otorisasi menggunakan protokol OpenID Connect dengan memanfaatkan *library* OAuth2 Server Bshaffer dan dimodifikasi sesuai kebutuhan.
2. Sistem yang dibangun menggunakan mekanisme otorisasi *authorization code flow* pada OAuth 2.0 dimana mengimplementasikan tiga *endpoint*, yaitu *authorize endpoint, token endpoint,* dan *userinfo endpoint*. *Authorization code* diberikan pada *authorization endpoint*. *Access token, ID token,* dan *refresh token* diberikan pada *token endpoint,* dan data pengguna diberikan pada *userinfo endpoint*.
3. Sistem yang dibangun menggabungkan *role-based-access-control (*RBAC) dengan standar OpenID Connect dengan memodifikasi *scope* dan data pengguna yang diberikan pada *userinfo endpoint*. Nilai *scope* yang ditambahkan untuk mengakomodasi RBAC adalah *roleunit, menu,* dan *resource*.
4. Sistem otentikasi dan otorisasi terpusat yang dibangun dapat diimplementasikan pada aplikasi *client* berbasis web pada aplikasi MonTA IF (sebagai aplikasi eksternal) dan SIAKAD (sebagai aplikasi internal dan RBAC terpusat) serta aplikasi *client* berbasis Android pada aplikasi myITS Wali dan myITS Tendik.

6.2 Saran

Berikut ini merupakan pengembangan lebih lanjut yang dapat dilakukan untuk menyempurnakan perangkat lunak.

1. Untuk keluar (*logout*) dari akun myITS, pengguna diharuskan melakukannya melalui myITS SSO (OP). Langkah selanjutnya yang dapat dilakukan adalah menerapkan *front-end channel* untuk sistem *logout* sehingga pengguna dapat melakukan *logout* pada akun myITS SSO melalui aplikasi *client* (RP).
2. Menambahkan fitur *revoke* dimana pengguna dapat melihat aplikasi *client* yang telah disetujui untuk menggunakan *resource*-nya dan membatalkan dengan memilih *revoke application*.
3. Membuat fitur untuk menambahkan data *user* dengan cari *import* dari file sehingga memudahkan dalam migrasi data user.
4. Menambah fitur *login as* untuk peran *super admin* sehingga dapat masuk sebagai pengguna lain tanpa perlu menginputkan *username* dan *password* pengguna tersebut.

DAFTAR PUSTAKA

[1] V. Beal, "What is HTTP - HyperText Transfer Protocol," Webopedia, [Online]. Available: https://www.webopedia.com/TERM/H/HTTP.html. [Accessed 25 October 2017].

[2] Wikipedia, "Hypertext Transfer Protocol," Wikimedia Fondation, Inc, 24 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Accessed 25 October 2017].

[3] S. Haryandi, "Mengenal RESTful API," Kudo Developers, 15 September 2016. [Online]. Available: https://developers.kudo.co.id/2016/09/15/mengenal-restful-api/. [Accessed 26 October 2017].

[4] Wikipedia, "PHP," Wikimedia Foundation, Inc, 13 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/PHP. [Accessed 26 October 2017].

[5] Wikipedia, "Single Sign On," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Single_sign-on. [Accessed 20 May 2018].

[6] Globhy, "Perbedaan Antara Otentikasi dan Otorisasi," March 2018. [Online]. Available: https://www.globhy.com/read-blog/33_perbedaan-antara-otentikasi-authentication-dan-otorisasi-authorization.html. [Accessed 12 July 2018].

[7] Connect2id, "OpenID Connect Explained," Connect2id Ltd, 2017. [Online]. Available: https://connect2id.com/learn/openid-connect. [Accessed 26 October 2017].

[8] B. Purnomosidi, "Memahami OAuth2 dan JWT untuk Delegasi Otorisasi," Refactory, [Online]. Available:

https://refactory.id/post/3-memahami-oauth2-dan-jwt-untuk-delegasi-otorisasi. [Accessed 26 October 2017].

[9] P. PHP, "Phalcon Framework," Phalcon, [Online]. Available: https://phalconphp.com/id. [Accessed 20 May 2018].

[10] P. Team, "Benchmarking Phalcon," Phalcon, 12 April 2017. [Online]. Available: https://blog.phalconphp.com/post/benchmarking-phalcon. [Accessed 6 June 2018].

[11] A. Studio, "Perkenalan Phalcon Framework Beserta Kelebihan dan Kekurangannya," 1 January 2015. [Online]. Available: http://antzstudioblog.blogspot.com/2015/01/phalcon-framework-for-php-perkenalan.html#.WxbmMZ8zbIU. [Accessed 6 June 2018].

[12] M. Jones, "JSON Web Token (JWT)," Internet Engineering Task Force (IETF), May 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519. [Accessed July 2018].

[13] B. Shaffer, "OAuth 2.0 Server PHP," Bshaffer, [Online]. Available: http://bshaffer.github.io/oauth2-server-php-docs/. [Accessed 20 May 2018].

[14] M. Rouse, "Asymmetric Cryptography," TechTarget, June 2016. [Online]. Available: https://searchsecurity.techtarget.com/definition/asymmetric-cryptography. [Accessed 20 May 2018].

[15] Techopedia, "What is Hashing?," Techopedia Inc, [Online]. Available: https://www.techopedia.com/definition/14316/hashing. [Accessed 20 May 2018].

[16] Pickmeup, "Fungsi Hashing," Techinfo, 15 April 2009. [Online]. Available: http://pickmeup-techinfo.blogspot.com/2009/04/fungsi-hashing.html. [Accessed 05 June 2018].

[17] E. Zhang, "What is Role-Based-Access-Control (RBAC)? Examples, Benefits, and More," Digital Guardian, 23 October 2017. [Online]. Available: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more. [Accessed 20 May 2018].

[18] N. Sakimura, "OpenID Connect Core 1.0 incorporating errata set 1," OpenID Connect, 8 November 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html. [Accessed July 2018].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Kadek Winda Dwiastini, lahir di Tabanan, pada tanggal 26 Agustus 1996. Penulis menempuh pendidikan dasar di SD 19 Pemecutan Denpasar (2002-2008), pendidikan menengah pertama di SMP Negeri 1 Denpasar (2008-2011), pendidikan menengah atas di SMA Negeri 4 Denpasar (2011-2014) dan pendidikan tinggi di S1 Teknik Informatika ITS (2014-2018).
Selama kuliah di Teknik Informatika ITS, penulis mendalami bidang minat Rekayasa Perangkat Lunak (RPL) dan Penulis pernah menjadi asisten dosen pada mata kuliah Sistem Basis Data. Penulis dapat dihubungi melalui surel: **winda1996@hotmail.com**.