TUGAS AKHIR – IF184802

IMPLEMENTASI PROTOKOL CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) DI SISI KLIEN PADA MYITS SINGLE SIGN-ON

ANDIKA ANDRA
NRP 05111640000058

Dosen Pembimbing
Rizky Januar Akbar, S.Kom., M.Eng.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

IMPLEMENTASI PROTOKOL CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) DI SISI KLIEN PADA MYITS SINGLE SIGN-ON

ANDIKA ANDRA
NRP 05111640000058

Dosen Pembimbing
Rizky Januar Akbar, S.Kom., M.Eng.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

IMPLEMENTATION OF CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) PROTOCOL ON THE CLIENT SIDE ON MYITS SINGLE SIGN-ON

ANDIKA ANDRA
NRP 05111640000058

Supervisor
Rizky Januar Akbar, S.Kom., M.Eng.
Nurul Fajrin Ariyani, S.Kom., M.Sc.

INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI PROTOKOL CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) DI SISI KLIEN PADA MYITS SINGLE SIGN-ON

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Rekayasa Perangkat Lunak
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
ANDIKA ANDRA
NRP: 05111640000058

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

Nurul Fajrin Ariyani, S.Kom., M.Sc. .....
NIP: 19860722 201504 2 003 (pembimbing 2)

SURABAYA
JULI 2020

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI PROTOKOL CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) DI SISI KLIEN PADA MYITS SINGLE SIGN-ON

Nama Mahasiswa	: ANDIKA ANDRA
NRP	: 05111640000058
Departemen	: Teknik Informatika
Dosen Pembimbing I	: Rizky Januar Akbar, S.Kom., M.Eng.
Dosen Pembimbing II	: Nurul Fajrin Ariyani, S.Kom., M.Sc.

Abstrak

Saat ini pada aplikasi myITS Single Sign-On telah mengimplementasikan alur authorization code yang hanya dapat diterapkan jika proses otentikasi terdapat pada perangkat yang sama menggunakan proses redirect, sehingga alur ini tidak dapat digunakan ketika proses otentikasi dilakukan pada perangkat yang berbeda.

Untuk menangani kebutuhan tersebut, dalam tugas akhir ini dibangun sebuah sistem otentikasi yang mengimplementasikan alur baru dari OpenID Connect, yaitu Client Initiated Backchannel Authentication yang dapat memisahkan proses otentikasi pada perangkat yang berbeda. Dalam implementasi alur Client Initiated Backchannel Authentication pada aplikasi myITS Single Sign-On, alur ini dapat digunakan pada banyak kebutuhan untuk mendapatkan otentikasi dari pengguna dengan perangkatnya secara remote.

Tugas akhir ini ingin menyelesaikan masalah tersebut dengan mengimplementasikan alur Client Initiated Backchannel Authentication di sisi klien pada aplikasi myITS Single Sign-On. Sehingga, dalam proses otentikasi yang membutuhkan perangkat yang berbeda dapat dilakukan pada myITS Single Sign-On.

Kata kunci: Client Initiated Backchannel Authentication, myITS Single Sign-On, OpenID Connect.

[Halaman ini sengaja dikosongkan]

IMPLEMENTATION OF CLIENT INITIATED BACKCHANNEL AUTHENTICATION (CIBA) PROTOCOL ON THE CLIENT SIDE ON MYITS SINGLE SIGN-ON

Name	: ANDIKA ANDRA
NRP	: 05111640000058
Major	: Informatics Engineering
Supervisor I	: Rizky Januar Akbar, S.Kom., M.Eng.
Supervisor II	: Nurul Fajrin Ariyani, S.Kom., M.Sc.

Abstract

Currently on the myITS Single Sign-On application has implemented an authorization code flow that can only be applied if the authentication process is on the same device using the redirect process, so this flow cannot be used when the authentication process is carried out on different devices.

To deal with these needs, in this final project an authentication system is built that implements a flow from OpenID Connect, the Client Initiated Backchannel Authentication which can decoupled the authentication process on different devices.

In implementing the Client Initiated Backchannel Authentication flow in the myITS Single Sign-On application, this flow can be used on many needs to get authentication from users with their devices remotely.

This final project wants to solve this problem by implementing the Client Initiated Backchannel Authentication flow on the client side in the myITS Single Sign-On, Thus, authentication process that requires different devices can be done on myITS Single Sign-On.

Keywords: Client Initiated Backchannel Authentication, myITS Single Sign-On, OpenID Connect.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga dapat di selesaikan tugas akhir ini yang berjudul **"Implementasi Protokol *Client Initiated Bachchannel Authentication (CIBA)* di Sisi Klien pada myITS *Single Sign-On*"**.

Dalam pelaksanaan tugas akhir ini tentunya tidak dapat diselesaikan tanpa bantuan dari pihak lain. Tanpa mengurangi rasa hormat, diberikan penghargaan serta ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Kedua Orang Tua yang sudah melahirkan dan membesarkan serta memberi dukungan hingga saat ini.
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku dosen wali dan dosen pembimbing tugas akhir pertama yang telah membimbing, memotivasi dan memberikan banyak masukan dalam pengerjaan tugas akhir ini.
4. Ibu Nurul Fajrin Ariyani, S.Kom., M.Sc. selaku dosen pembimbing tugas akhir kedua yang selalu memberikan koreksi serta masukan-masukan yang dapat dikembangkan pada tugas akhir ini.
5. Bapak dan Ibu dosen Departemen Informatika ITS yang telah mengajarkan banyak ilmu berharga.
6. Wardha Savitri sebagai penyemangat dalam pengerjaan tugas akhir ini.
7. Adistya Azhar sebagai sahabat dan rekan kerja yang selalu mengingatkan dalam hal apapun.
8. Clan Lor 51 sebagai tempat berteduh saat lelah mengerjakan tugas akhir.
9. Teman-teman Lab Rekayasa Perangkat Lunak yang saling menyayangi satu sama lain.

10. Bapak dan Ibu karyawan Departemen Informatika ITS atas berbagai bantuan yang telah diberikan selama masa perkuliahan.
11. Teman-teman satu angkatan Informatika ITS 2016 yang saling menyemangati satu sama lain.
12. Pihak-pihak lain yang tidak bisa penulis sebutkan satu per satu.

Diharapkan bahwa apa yang dihasilkan dari tugas akhir ini bisa memberikan manfaat bagi semua pihak, khususnya bagi diri sendiri dan seluruh *civitas academica* Informatika ITS, serta bagi agama, bangsa, dan negara. Tidak ada manusia yang sempurna sekalipun berusaha sebaik mungkin dalam menyelesaikan tugas akhir ini. Karena itu, mohon maaf apabila terdapat kesalahan, kekurangan, maupun kelalaian yang telah dilakukan. Kritik dan saran yang membangun sangat diharapkan untuk dapat disampaikan guna perbaikan selanjutnya.

Surabaya, Mei 2020

Andika Andra

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR ISTILAH

Activity	**:**	Komponen dari pemrograman Java khususnya menggunakan *android studio* yang dapat dilihat oleh pengguna, sehingga mereka dapat berinteraksi dengan aplikasi.
Access Token	**:**	Sebuah Token yang digunakan untuk mengakses *Protected Resourc*e. Contoh: G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN
Black Box	**:**	Black Box Testing atau yang sering dikenal dengan sebutan pengujian fungsional merupakan metode pengujian perangkat lunak yang digunakan untuk menguji perangkat lunak tanpa mengetahui struktur internal kode atau Program.
Civitas Academic	**:**	Seluruh komponen atau akademisi di dalam suatu tempat, terutama di bidang akademis.
Client	**:**	Aplikasi yang mengakses data dari sebuah API. Contoh: Aplikasi MyITS Mobile, Aplikasi Integra dan lain-lain.
Endpoint	**:**	URL dimana suatu servis dapat diakses oleh client. Contoh: my.its.ac.id
End User	**:**	Pengguna aplikasi yang hanya bertujuan untuk memakai aplikasi tanpa harus mengetahui proses dibaliknya.
Firebase	**:**	Firebase merupakan salah satu layanan dari Google yang memudahkan para app developer dalam mengembangkan aplikasi mereka. Firebase termasuk ke dalam kategori BAAS (Backend as a Service).
Resource	**:**	Data yang tersimpan pada resource server yang dapat diakses oleh client/user tertentu, *resource* dapat berupa *public resource* atau *protected*

		reousrce. Contoh: Alamat surel pengguna, Foto pengguna dan lain-lain.
UUID	**:**	Kumpulan 32 karakter (*string*) yang dibuat secara acak (*random*) dengan teknik khusus yang dijamin unik untuk setiap data. Contoh: 25769c6cd34d4bfeba98e0ee856f3e7a dan ee66dbc8-b962-4b6c-a2da-90af9379a34c
User Agent	**:**	*User Agent* adalah segala perangkat lunak yang menerima, me-*render* dan memfalisitasi interaksi *end-user* dengan konten dari *web.* Contoh: *Web Browser.*
User Requestee	**:**	Pengguna yang diminta persetujuannya oleh *user requester*. Contoh: Mahasiswa
User Requester	**:**	Pengguna yang meminta persetujuan dari *user requestee*. Contoh: Pihak DPTSI ITS.

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab pendahuluan membahas garis besar penyusunan tugas akhir yang meliputi latar belakang, tujuan pembuatan, rumusan dan batasan permasalahan, metodologi penyusunan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Client Initiated Backchannel Authentication (CIBA) merupakan alur otentikasi dari OpenID Connect 1.0 dimana klien yang akan menggunakan *identity provider* dapat memperoleh identifikasi yang valid dari *user* yang akan diotentikasi[1].

Platform myITS *Single Sign-On* atau myITS *SSO* adalah *authorization server* yang memanfaatkan protokol OAuth2 [2]. myITS *Single Sign-On* menyimpan data akun pengguna, data klien yang dapat diberi otorisasi, dan memberikan otorisasi dengan mengeluarkan *access token* serta dapat melakukan verifikasi *access token*.

Di dalam Direktorat Pengembangan Teknologi dan Sistem Informasi Institut Teknologi Sepuluh Nopember (DPTSI-ITS), alur otentikasi OpenID Connect 1.0 yang telah diimplementasi adalah menggunakan basis *redirect*, yaitu dengan cara mengalihkan halaman untuk masuk ke dalam proses otentikasi. oleh karena itu terdapat masalah jika alur otentikasi dibutuhkan pada perangkat yang berbeda. Sedangkan di lain pihak, alur otentikasi berbasis *redirect* hanya dapat berjalan pada perangkat yang sama.

Permasalahan ini menjadikan proses otentikasi tidak dapat berjalan jika otentikasi membutuhkan perangkat yang berbeda, sebagai contoh mahasiswa melaporkan terjadi masalah pada akun myITS SSO nya dan melapor kepada DPTSI-ITS, pihak DPTSI-ITS mencoba mengetahui permasalahannya dengan cara melakukan *login* seolah-olah sebagai akun mahasiswa tersebut, proses ini dinamakan dengan *Impersonate*. Dalam proses

Impersonate diperlukan persetujuan dari mahasiswa tersebut, jika mahasiswa tersebut tidak berada pada DPTSI-ITS maka *Impersonate* tidak dapat berjalan.

Tugas akhir ini ingin menyelesaikan masalah tersebut dengan menambahkan alur otentikasi *Client Initiated Backchannel Authentication* (CIBA) Client OpenID Connect 1.0. Sehingga otentikasi yang membutuhkan perangkat yang berbeda dapat diselesaikan menggunakan myITS *Single Sign-On*. Seperti proses *Impersonate* tetap dapat dijalankan meskipun mahasiswa tidak berada pada DPTSI-ITS yaitu dengan melakukan persetujuan secara *remote* melalui aplikasi myITS Authenticator atau dalam alur CIBA disebut sebagai *Authentication Device*, sedangkan perangkat yang digunakan oleh DPTSI-ITS disebut sebagai *Consumption Device.*

1.2. Rumusan Masalah

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

1. Bagaimana mengimplementasikan protokol *Client Initiated Backchannel Authentication* (CIBA) di sisi klien pada myITS *Single Sign-On*?
2. Bagaimana mengimplementasikan komunikasi antara *Authentication Device* dengan *Authorization Server* pada myITS *Single Sign-On*?
3. *B*agaimana mengimplementasikan *client notification endpoint* pada myITS *Single Sign-On* sehingga *Authorization Server* dapat berkomunikasi dengan *Consumption Device*?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir memiliki beberapa batasan antara lain:

1. Platform yang digunakan adalah aplikasi myITS *Single Sign-On* yang berbasis PHP.

2. Bahasa pemrograman yang digunakan untuk implementasi protokol Client Initiated Backchannel Authentication (CIBA) adalah PHP.
3. Spesifikasi *Client Initiated Backchannel Authentication* (CIBA) ini yang digunakan adalah versi 1.0 *draft*-03.
4. *Framework* yang digunakan untuk implementasi protokol CIBA adalah *Phalcon* 3.4 PHP.
5. Library OpenID Connect Client adalah php-openid-connect-client versi 1.2.0.

1.4. Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah:

1. Mengetahui mekanisme implementasi protokol *Client Initiated Backchannel Authentication* (CIBA) di sisi klien pada myITS *Single Sign-On*.
2. Mengetahui mekanisme implementasi komunikasi antara *Authentication Device* dengan *Authorization Server* pada myITS *Single Sign-On*.
3. Mengetahui mekanisme implementasi *client notification endpoint* pada myITS *Single Sign-On* sehingga *Authorization Server* dapat berkomunikasi dengan *Consumption Device*.

1.5. Metodologi

Tahap yang dilakukan untuk menyelesaikan tugas akhir ini adalah sebagai berikut:

1. **Penyusunan Proposal Tugas Akhir**

   Proposal 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 untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi

pendukung pembuatan tugas akhir. 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.

2. **Studi literatur**

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

   1. *OAuth2*
   2. *OpenID*
   3. Client Initiated Backchannel Authentication.
   4. *Phalcon Framework.*
   5. SQL Server.

3. **Implementasi**

   Pada tahap ini, akan dipelajari beberapa referensi yang diperlukan untuk pengerjaan tugas akhir, yaitu alur Client Initiated Backchannel Authentication, dan cara penerapannya pada aplikasi myITS *SSO*.

4. **Uji Coba dan Evaluasi**

   Pada tahap ini dilakukan uji coba terhadap protokol baru yang telah dikembangkan pada aplikasi myITS *SSO*, dengan menggunakan metode pengujian terhadap *input* serta *output* berdasarkan skenario. Berikut ini adalah kriteria pengujian yang akan dilakukan:

   1. Pengguna dapat mengubah konfigurasi klien yang menggunakan layanan myITS *SSO*.
   2. Pengguna dapat melakukan *Impersonate* menggunakan alur *Client Initiated Backchannel Authentication.*

3. Pengujian terhadap mekanisme alur *Client Initiated Backchannel Authentication* dengan menggunakan *browser* terhadap *platform Windows, macOS, Android* dan *iOS*. *Browser* yang digunakan diantaranya adalah *Chrome, Edge, Firefox, Safari, Chrome for android, Firefox for android* dan *iOS Safari.* Serta *Authentication Device* yang digunakan menggunakan *Operating System Android*.

5. **Penyusunan Buku Tugas Akhir**
   Pada tahapan ini disusun buku yang membuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.6. Sistematika Penulisan

Buku tugas akhir ini terdiri atas beberapa bab yang tersusun secara sistematis, yaitu sebagai berikut.

1. **Bab I. Pendahuluan**
   Bab pendahuluan berisi penjelasan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan tugas akhir.
2. **Bab II. Tinjauan Pustaka**
   Bab tinjauan pustaka berisi penjelasan mengenai dasar teori yang mendukung pengerjaan tugas akhir.
3. **Bab III. Analisis dan Perancangan**
   Bab ini berisi tentang desain sistem, perancangan basis data, diagram kasus penggunaan, diagram aktivitas dan rancangan antarmuka pengguna.
4. **Bab IV. Implementasi**
   Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa tampilan antarmuka yang telah dibuat dan dapat berfungsi untuk mengakomodir kebutuhan fungsional yang ada.
5. **Bab V. Uji Coba dan Evaluasi**

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

6. **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

Bab tinjauan pustaka berisi mengenai penjelasan teori yang berkaitan dengan implementasi perangkat lunak. Penjelasan tersebut bertujuan untuk memberikan gambaran mengenai sistem yang akan dibangun dan berguna sebagai pendukung dalam pengembangan perangkat lunak.

2.1. Penelitian Terkait

Pada sub bab ini akan dibahas tentang penelitian terkait yang menjadi rujukan tugas akhir ini.

2.1.1. MyITS Single Sign-On

Sistem *Single Sign-On* yang saat ini digunakan oleh ITS diterapkan pada myITS *Single Sign-On*, dimana sistem-sistem internal, eksternal maupun layanan yang dipelihara oleh Direktorat Pengembangan Sistem Informasi (DPTSI) ITS, seperti SI Akademik, SI Presensi, myITS Classroom, dan lainnya terintegrasi dan dapat diakses oleh pengguna hanya dengan satu kali login. SSO pada *myITS* telah menerapkan protokol *OpenID Connect* yang menggunakan mekanisme otentikasi dengan kombinasi *username* dan *password* serta *email* dan *password.*

Sistem saat ini menggunakan alur *authorization code.* Yaitu dengan menukarkan *authorization code* dengan *access token.* Setelah pengguna diarahkan melalui *redirect URI*, klien akan mendapatkan *authorization code* dari URL dan menggunakannya untuk meminta *access token* [3]. *Access token* didapatkan ketika mengirimkan *Client ID, Client Secret* dan *Scope* kepada *token endpoint* pada myIts *Single Sign-on.* Klien juga dapat meminta *userinfo* melalui *userinfo endpoint*.

Pada tugas akhir ini akan dilakukan penambahan alur yaitu *Client Initiated Backchannel Authentication* (CIBA) pada *client* myIts *Single Sign-on.* Untuk mengimplementasikan alur CIBA

pada *client*, dari sisi *server* dibutuhkan *backchannel authentication endpoint* yaitu *endpoint* untuk memulai alur otentikasi dengan cara mengirimkan *request* secara langsung kepada server. Sama dengan alur *authorization code*, alur CIBA juga mengeluarkan token yang berasal dari *token endpoint*. Sedangkan dari pihak *client* dibutuhkan *client notification endpoint* yaitu *endpoint* yang dibutuhkan oleh *server* untuk berkomunikasi dengan *client* dalam pengiriman notifikasi *token*.

2.1.2. Aplikasi yang telah Menggunakan Alur Client Initiated Backchannel Authentication

Saat ini telah ada penyedia layanan CIBA, yaitu authlete.com, layanan yang diberikan adalah berbayar. Untuk implementasi metode otentikasi menggunakan perangkat yang berbeda terdapat berbagai cara, sehingga tidak dapat dipastikan bahwa metode yang digunakan CIBA atau metode lainnya, tetapi ada beberapa perusahaan yang telah menggunakan metode otentikasi menggunakan device yang berbeda sebagai otentikatornya, yaitu OVO, Go-Jek.

2.1.3. Perbandingan Protokol yang Dapat Digunakan

Dalam penyelesaian masalah terkait keamanan tambahan yang dapat digunakan sebagai pemberian persetujuan pengguna, terdapat protokol-protokol yang dapat digunakan, contohnya:

- *Two Factor Authentication* (TFA)
  Merupakan alur khusus otorisasi dari OAuth2 yang memberikan spesifikasi terhadap orotisasi untuk menggunakan otentikasi 2 faktor.
- *Client Initiated Backchannel Authentication* (CIBA)
  Merupakan alur otentikasi dan otorisasi dari OpenID Connect yang digunakan sebagai metode pemberian persetujuan dari pengguna yang otentikatornya memungkinkan pada perangkat yang berbeda.

Pada tugas akhir ini, dipilih menggunakan alur

Client Initiated Backchannel Authentication karena pada spesifikasinya telah dijelaskan tentang otentikasi dan otorisasi, sehingga alur dapat digunakan secara umum sesuai spesifikasi, dan pada sistem yang telah ada, yaitu menggunakan alur *Authorization Code* menggunakan alur dari OpenID Connect, sehingga untuk pengimplementasian alur CIBA tidak perlu penambahan yang dapat mengubah alur yang sudah ada.

2.2. Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) Institut Teknologi Sepuluh Nopember

Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) Institut Teknologi Sepuluh Nopember (ITS) merupakan lembaga milik Institut Teknologi Sepuluh Nopember yang bergerak di bidang teknologi, menyediakan layanan untuk seluruh *civitas academic* di ITS. Salah satu layanannya adalah menyediakan panduan untuk mahasiswa baru seperti penyediaan email mahasiswa, FRS online hingga panduan mengakses koneksi internet di lingkungan ITS. Layanan lainnya adalah pelaporan keluhan tentang sistem maupun layanan lain kepada *service desk. Civitas academic* di ITS dapat melaporkan masalah pada akunnya kepada DPTSI. DPTSI juga menangani sistem *Single Sign-On* yang digunakan sebagai sistem utama untuk menangani akun dari seluruh *civitas academic* di ITS yang memungkinkan pengguna dapat mengakses seluruh aplikasi hanya dengan satu kali login. Beberapa diantaranya adalah MyITS Classroom, MyITS Presensi, dan MyITS Security Management.

Dalam tugas akhir ini, DPTSI berperan untuk menangani masalah atau keluhan yang dialami oleh mahasiswa ataupun *civitas academic* yang lainnya dengan melakukan login seolah-olah sebagai pengguna asli yang mengalami masalah pada akunnya melalui aplikasi MyITS Security Management. Dengan melakukan *impersonate*, pihak DPTSI dapat mengakses dan melakukan perubahan pada akun yang di-*impersonate.*

2.3. OAuth2

OAuth2 merupakan kerangka kerja otorisasi yang memungkinkan aplikasi untuk mendapatkan akses terbatas ke akun pengguna pada layanan HTTP. OAuth2 bekerja dengan mendelegasikan autentikasi pengguna pada layanan yang menyimpan data pengguna dan memberikan otorisasi kepada pihak ketiga untuk mengakses ke akun pengguna. OAuth2 menyediakan aliran otorisasi untuk aplikasi web dan aplikasi mobile [4].

OAuth2 memiliki beberapa *role* penting dalam menjalankan proses otorisasi, yaitu:

1. *Resource Owner*
   Resource Owner adalah pengguna yang melakukan otorisasi aplikasi untuk dapat mengakses ke akun mereka. Akses aplikasi ke akun pengguna terbatas pada otorisasi yang diberikan.
2. OAuth Server / OAuth Provider
   Terdapat 3 komponen didalam OAuth Provider, yaitu:
   1. *Authentication Component*
      menyediakan halaman login yang ditampilkan kepada user.
   2. *Consent Component*
      komponen ini muncul setelah *Authentication Component* selesai, yaitu halaman lain yang berisi persetujuan bahwa *Third-Party Apps* akan mengakses beberapa informasi dari user
   3. *Token Management.*
      komponen ini menjaga token-token yang diberikan oleh *OAuth Provider* yang tersimpan pada database, dan juga memvalidasi token yang diminta oleh client.
3. *Client/RP (Third-Party Apps)*
   Client adalah aplikasi yang ingin mengakses akun pengguna. Sebelum melakukan akses, aplikasi harus memiliki otorisasi dari pengguna.

4. *Resource Server*
   Resource Server berperan sebagai tempat penyimpanan data pengguna. Saat melakukan akses ke *resource server* harus menyertakan *access token* yang diberikan oleh *Authorization Server*
5. *User Agent*
   User Agent adalah segala perangkat lunak yang menerima, merender dan memfasilitasi interaksi *end-user* dengan konten dari *web. User agent* biasanya adalah *Web Browser*.

Secara umum, alur protokol OAuth2 sebagai berikut:

- Aplikasi *client* meminta otorisasi untuk mengakses *resource* pada *resource owner.*
- Jika user mengotorisasi permintaan *client*, *client* akan menerima *authorization grant.*
- Aplikasi *client* meminta *access token* kepada *authorization server* dengan menyertakan otentikasi *client* dan *authorization grant* dari *client.*
- Jika identitas *client* dan *authorization grant* tervalidasi, *authorization server* akan memberikan *access token* kepada *client.*
- Client meminta *resource* kepada *resource server* dengan menyertakan *access token* sebagai otentikasinya
- Jika *access token* tervalidasi, *resource server* akan mengirimkan *resource* yang diminta kepada *client.*

Secara umum bentuk *response* dari *successful access token response* adalah sebagai berikut:

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

{
  "access_token":"MTQ0NjJkZmQ5OTM2NDE1ZTZjNGZmZjI3",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"IwOGYzYTlmM2YxOTQ5MGE3YmNmMDFkNTVk",
  "scope":"create delete"
}
```

Gambar 2.1 Successful Access Token Response

Struktur *access token* berisi:

- access_token (*required*)
  sebuah *string* yang dibuat oleh *Authorization Server.*
- token_type (*required*)
  tipe dari *token*, biasanya berupa "*bearer*"
- expires_in (*recommended*)
  waktu dalam detik yang merepresentasikan *access_token* akan kedaluarsa.
- refresh_token (*optional*)
  jika *access_token* telah kadaluarsa, maka digunakanlah *refresh_token* untuk mendapatkan *access_token* baru.
- scope (*optional*)
  scope digunakan untuk menentukan hak akses apa yang diminta, contoh: *scope* "*email*" digunakan untuk mengakses "*email*" dan "*email_verified*" *claims.* Sedangkan *claims* adalah pasangan nama/nilai yang berisi informasi tentang pengguna, juga *meta-information* tentang layanan OIDC. "*email*" dan "*email_verified"* merupakan *claims* dari *scope "*email".

2.4. 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). *ID Token* menyerupai konsep kartu identitas dalam format standar JWT yang diakomodasi oleh OP (*OpenID Provider*). Untuk mendapatkan *ID Token*, klien perlu mengirim permintaan otentikasi ke OP.

OpenID Connect memiliki beberapa istilah dalam menjalankan proses otentikasi, yaitu:

1. *OpenID Provider* (OP)
   Authorization Server atau server penyedia otorisasi yang menghandle OpenID Connect.
2. *Relying Party* (RP)
   Client yang menggunakan OP sebagai *Authorization Server* nya.
3. *ID Token*
   Token yang direpresentasikan sebagai JWT berisi informasi dari *End-User* yang telah ter-otentikasi. ID Token merupakan pembeda antara OpenID Connect dengan protokol OAuth2. Jika di *decode* maka ID Token memiliki struktur sebagai berikut:

```
{
 "iss": "https://server.example.com",
 "sub": "24400320",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "auth_time": 1311280969,
 "acr": "urn:mace:incommon:iap:silver"
}
```

Gambar 2.2 Contoh Struktur ID Token

ID Token minimal berisi:

- iss : *issuer* ID Token
- sub : *unique identifier* dari *End-User,* berupa id user
- aud : id client atau *identifier* dari RP
- exp : Waktu ID Token kadaluarsa
- iat : Waktu ID Token dibuat

Terdapat 3 *flow* berbeda yang digunakan oleh OpenID Connect 1.0 untuk melakukan otentikasi dan untuk mendapatkan token dari OP ke RP:

1. *Authorization Code Flow*
2. *Implicit Flow*
3. *Hybrid Flow*

2.5. Client Initiated Backchannel Authentication Flow

Client Initiated Backchannel Authentication (CIBA) merupakan alur otentikasi dari OpenID Connect 1.0 dimana *Relying Party* (RP) dapat memperoleh identifikasi yang valid dari user yang akan diotentikasi [1]. CIBA digunakan untuk memisahkan proses otentikasi dengan cara diharuskan melakukan persetujuan dengan *End-User* yang memungkinkan memiliki perangkat terpisah sebagai otentikator. Di dalam implementasi CIBA terdapat beberapa peran yang terlibat, diantaranya adalah sebagai berikut:

1. *Consumption Device* (CD)
   adalah perangkat yang digunakan untuk *hosting* aplikasi klien.
2. *Authentication Device* (AD)
   adalah perangkat yang digunakan untuk melakukan otentikasi dan melakukan konfirmasi persetujuan.
3. *OpenID Provider* (OP)
   Authorization Server atau server penyedia otorisasi yang menghandle OpenID Connect.

4. *Relying Party* (RP)/*Client*
   Client yang menggunakan OP sebagai *Authorization Server* nya.

Secara umum, alur *Client Initiated Backchannel Authentication* (CIBA) sebagai berikut:

1. RP mengirimkan *backchannel authentication request* ke *backchannel authentication endpoint ke OP* yang telah didefinisikan oleh CIBA Core.
2. *Backchannel authentication endpoint* mengirimkan *response* ke RP.
3. *OP* akan mendelegasikan proses persetujuan hak akses/*user-consent* kepada *AD.*
4. *AD* mengirimkan hasil persetujuan ke *OP*.
5. *OP* akan membuatkan token untuk RP.
6. RP dapat mengambil/mendapatkan token tersebut.

2.5.1. Authentication Request

CIBA mendefinisikan *Authentication Request* adalah *request* yang diminta oleh RP ke OP tanpa melalui browser pengguna atau dengan kata lain melalui proses *backchannel*, RP mengirimkan *authentication request* melalui HTTP POST ke OP melalui *backchannel authentication endpoint.* Terdapat beberapa parameter yang dapat digunakan ke dalam *authentication request*, diantaranya:

- scope (*required*)
  merupakan mekanisme untuk mendapatkan *claims* berupa detail dari user, CIBA *authentication request* harus menambahkan scope “openid“ ditambahkan sebagai *extension* dari OAuth2. Scope yang diperbolehkan terdaftar pada daftar scope yang terdefinisikan oleh OP.
- client_notification_token (*required*)
  digunakan ketika RP terdaftar dalam mode Ping atau Push. Sebuah *Bearer Token* yang disediakan oleh RP yang akan digunakan oleh OP untuk melakukan *callback request* ke

RP. Client_notification_token tidak memiliki panjang melebihi 1024 karakter dengan minimal 128 bit untuk menghindari menebak secara *brute force*.

- acr_values (*optional*)
  nilai dari ACR (Authentication Context Class Reference) yang diminta, nilai-nilai dari ACR yang tersedia sesuai pada OIDC *discovery response*.
- login_hint_token (*optional*)
  sebuah *token* berisi informasi yang mengidentifikasi *End-User* yang akan di-*request*. Dapat berupa *user_id* atau *username* yang bersifat *unique*.
- id_token_hint (*optional*)
  sebuah *ID Token* yang sebelumnya dibuat untuk RP oleh OP yang dikirim kembali sebagai *hint* untuk mengidentifikasi *End-User* yang akan di-*request*. Untuk menggunakan id_token_hint, RP harus melakukan *decrypt* pada *ID Token* untuk mendapatkan *signed ID Token*.
- login_hint (*optional*)
  sebuah *hint* untuk OP tentang *End-User* yang akan di-*request*. Dapat berupa email, nomor ponsel, nomor akun ataupun username.
- binding_message (*optional*)
  merupakan identifier atau pesan yang ditampilkan pada CD dan AD untuk memastikan bahwa *consent* yang dibutuhkan benar-benar dari RP dan OP yang sebenarnya. binding_message berupa random karakter yang dapat dibaca. Contoh: W4SCT.
- user_code (*optional*)
  sebuah kode seperti password atau pin yang hanya diketahui oleh *End-User* tetapi dapat diverifikasi oleh OP. Parameter ini hanya dapat digunakan ketika RP mensupport *backchannel_user_code_parameter*.
- requested_expiry (*optional*)
  sebuah angka positif yang merepresentasikan masa kadaluarsa *auth_req_id*.

authentication request ditunjukkan pada Gambar 2.3

```
POST /bc-authorize HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

scope=openid%20email%20example-scope&
client_notification_token=8d67dc78-7faa-4d41-aabd-67707b374255&
binding_message=W4SCT&
login_hint_token=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJ
zdWJfaWQiOnsic3ViamVjdF90eXBlIjoicGhvbmUiLCJwaG9uZSI6IisxMzMwMjg
xODAwNCJ9fQ.Kk8jcUbHjJAQkRSHyDuFQr3NMEOSJEZc85VfER74tX6J9CuUllr8
9WKUHUR7MA0-mWlptMRRhdgW1ZDt7g1uwQ&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJ
pc3MiOiJzNkJoZFJrcXQzIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHB
zOi8vc2VydmVyLmV4YW1wbGUuY29tIiwianRpIjoiYmRjLVhzX3NmLTNZTW80RlN
6SUoyUSIsImlhdCI6MTUzNzgxOTQ4NiwiZXhwIjoxNTM3ODE5Nzc3fQ.Ybr8mg_3
E2OptOSsA8rnelYO_y1L-yFaF_j1iemM3ntB61_GN3APe5cl_-5a6cvGlP154XAK
7fL-GaZSdnd9kg
```

Gambar 2.3 Authentication Request

2.5.1.1. Signed Authentication Request

Signed authentication request dilakukan dengan cara *encode* semua authentication *request* sebagai *claims* dari *signed JWT*. *JWT* juga harus berisi *claims* sebagai berikut:

- iss
  issuer claims berupa *client_id* dari RP.
- aud
  audience claims berisi *issuer identifier* untuk OP, dimana identifier dari OP merupakan audiens yang dituju.
- exp
  waktu kedaluarsa berupa *lifetime* dari *signed authentication request.*
- iat
  waktu *signed authentication request* dibuat.
- nbf
  waktu sebelum *signed authentication request* tidak dapat diproses.
- jti

berupa *unique identifier* dari *signed authentication request.*

signed authentication request ditunjukkan pada Gambar 2.4

```
POST /bc-authorize HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

request=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJpc3MiOiJz
NkJoZFJrcXQzIiwiYXVkIjoiaHR0cHM6Ly9zZXJ2ZXIuZXhhbXBsZS5jb20iLCJl
eHAiOjE1Mzc4MjAwODYsImlhdCI6MTUzNzgxOTQ4NiwibmJmIjoxNTM3ODE4ODg2
LCJqdGkiOiI0TFRDcUFDQzJFU0M1QldDbk4zajU4RW5BIiwic2NvcGUiOiJvcGVu
aWQgZW1haWwgZXhhbXBsZS1zY29wZSIsImNsaWVudF9ub3RpZmljYXRpb25fdG9r
ZW4iOiI4ZDY3ZGM3OC03ZmFhLTRkNDEtYWFiZC02NzcwN2IzNzQyNTUiLCJiaW5k
aW5nX21lc3NhZ2UiOiJXNFNDVCIsImxvZ2luX2hpbnRfdG9rZW4iOiJleUpyYVdR
aU9pSnNkR0ZqWlhOaWR5SXNJbUZzWnlJNklrVlRNalUySW4wLmV5SnpkV0pmYVdR
aU9uc2ljM1ZpYW1WamRGOTBlWEJsSWpvaWNHaHZibVVpTENKd2FHOXVaU0k2SWlz
eE16TXdNamd4T0RBd05DSjlmUS5LazhqY1ViSGpKQVFrUlNIeUR1RlFyM05NRU9T
SkVaYzg1VmZFUjc0dFg2SjlDdVVsbHI4OVdLVUhVUjdNQTAtbVdscHRNUlJoZGdX
MVpEdDdnMXV3USJ9.RB-iFvzpkQ_gUzg0eutoviViCKyLugjVYfVqdjDZ63U1MZR
Z-KcUNSsBjCVptc-QdljCSNCUyULIzT2R5Nmg4Q&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2In0.eyJ
pc3MiOiJzNkJoZFJrcXQzIiwic3ViIjoiczZCaGRSa3F0MyIsImF1ZCI6Imh0dHB
zOi8vc2VydmVyLmV4YW1wbGUuY29tIiwianRpIjoiY2NfMVhzc3NmLTJpOG8yZ1B
6SUprMSIsImlhdCI6MTUzNzgxOTQ4NiwiZXhwIjoxNTM3ODE5Nzc3fQ.PWb_VMzU
IbD_aaO5xYpygnAlhRIjzoc6kxg4NixDuD1DVpkKVSBbBweqgbDLV-awkDtuWnyF
yUpHqg83AUV5TA
```

Gambar 2.4 Signed Authentication Request

Berdasarkan gambar Gambar 2.4, ditunjukkan bahwa semua *request* parameter diletakkan dalam parameter *encode* berisi *signed JWT* dari parameter-parameter yang dibutuhkan. Berikut adalah *payload* dari parameter *request*.

```
{
 "iss": "s6BhdRkqt3",
 "aud": "https://server.example.com",
 "exp": 1537820086,
 "iat": 1537819486,
 "nbf": 1537818886,
 "jti": "4LTCqACC2ESC5BWCnN3j58EnA",
 "scope": "openid email example-scope",
 "client_notification_token": "8d67dc78-7faa-4d41-aabd-67707b374255",
 "binding_message": "W4SCT",
 "login_hint_token": "eyJraWQiOiJsdGFjZXNidyIsImFsZyI6IkVTMjU2I
   n0.eyJzdWJfaWQiOnsic3ViamVjdF90eXBlIjoicGhvbmUiLCJwaG9uZSI6I
   isxMzMwMjgxODAwNCJ9fQ.Kk8jcUbHjJAQkRSHyDuFQr3NMEOSJEZc85VfER
   74tX6J9CuUllr89WKUHUR7MA0-mWlptMRRhdgW1ZDt7g1uwQ"
}
```


Gambar 2.5 Payload Signed JWT Authentication Request

2.5.1.2. User Code

CIBA mendukung mekanisme “user code” untuk mencegah otentikasi yang tidak diminta muncul di perangkat milik *End-User* (AD).

“user code” adalah hal privat / rahasia yang diketahui oleh user akan tetapi bukan *password* dapat berupa pin. Ketika OP mendukung fitur ini maka RP harus meminta *End-User* tentang “user code” nya untuk memulai alur CIBA. Hal ini mencegah client atau user yang tidak diinginkan untuk memulai alur CIBA.

OP mendefinisikan dukungan terhadap “user code” pada parameter backchannel_user_code_parameter_supported

2.5.2. Authentication Response

Setelah RP melakukan *authentication request*, RP akan mendapatkan *authentication response*, *authentication response* memiliki *body* yang berisi:

- auth_req_id (*required*)
  berupa *unique identifier* yang digunakan sebagai penanda dari *request* yang dibuat oleh RP ke OP.
- expires_in (*required*)
  waktu dalam detik yang merepresentasikan *auth_req_id* akan kedaluarsa.

- interval (*optional*)
  jumlah minimum waktu dalam detik untuk RP melakukan token *request* ke OP. Parameter interval hanya berlaku jika RP terdaftar dalam mode *poll*. Jika parameter tidak disediakan maka nilai default nya adalah 5.

Berikut adalah contoh dari *authentication response* pada Gambar 2.6

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

{
  "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1",
  "expires_in": 120,
  "interval": 2
}
```

Gambar 2.6 Authentication Response

2.5.3. Mendapatkan User Consent

OP akan mendelegasikan permintaan *consent* kepada AD untuk mendapatkan persetujuan dari *End-User*. Cara untuk mengirim dan mendapatkan persetujuan oleh *End-User* dapat dilakukan dengan berbagai cara, salah satunya adalah mengirimkan permintaan melalui aplikasi *mobile* berupa otentikator.

2.5.4. Mode Pengambilan Token

Setelah itu token telah dibuat dan siap diambil oleh RP, terdapat 3 metode yang dapat digunakan oleh RP untuk mengambil token, yaitu *Poll*, *Ping* dan *Push* mode. Ketersediaan mode yang dapat digunakan oleh RP bergantung kepada parameter backchannel_token_delivery_modes_supported yang didefinisikan oleh OP. Secara umum, perbedaan cara pengambilan token dijelaskan pada Tabel 2.1.

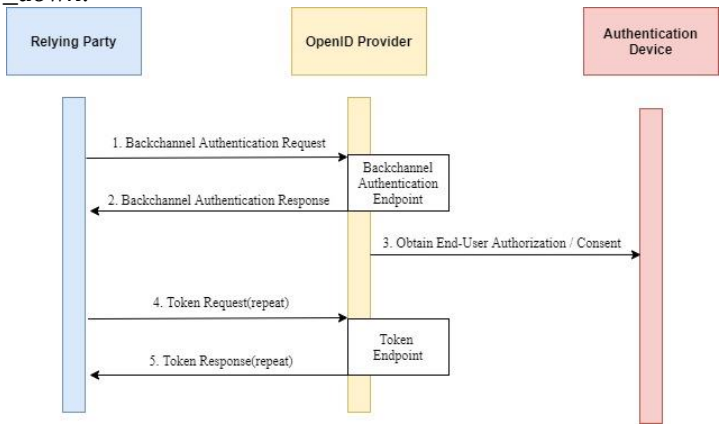
Mode	Deskripsi

Poll	Pengambilan dilakukan dengan *Polling* token endpoint
Ping	Pengambilan dilakukan dengan menunggu notifikasi dari OP
Push	Token diberikan secara langsung oleh OP.

Tabel 2.1 Perbedaan Mode Pengambilan Token

2.5.4.1. Poll Mode

Didalam *Poll* Mode, setelah mendapat *response* dari OP (langkah nomor 2) pada Gambar 2.7, OP akan mendelegasikan *End-User Consent* ke AD. Secara bersamaan, RP dapat mengirimkan *request* secara berulang menuju *Token Endpoint.* Jika telah mendapat *consent* dari *End-User*, maka *Token Endpoint* akan mengirimkan *response* berupa Token kepada RP. Jika OP masih belum mendapatkan *Consent* dari *End-User* maka OP akan mengirimkan *response authorization_pending*. Apabila RP mengirim *request* terlalu cepat dan *request* sebelumnya masih belum dapat diselesaikan maka OP akan mengirimkan *response slow_down.*



Gambar 2.7 Poll Mode

- *token request*
  Pada mode *poll*, RP akan terus meminta token pada *token_endpoint* dengan interval yang diberikan pada

backchannel_authentication_response (langkah nomor 2) pada Gambar 2.7. Pada mode *poll* juga dapat mengimplementasikan *long polling* dimana OP akan melakukan response hanya ketika OP telah mendapat *consent* dari AD atau saat *response timeout*. Waktu yang direkomendasikan untuk RP melakukan *long polling* ke OP sebelum *timeout* adalah selama 30 detik. Pada saat melakukan *long polling* RP tidak boleh mengirim *request* dengan *auth_req_id* yang sama, sehingga RP diperbolehkan mengirim *request* selanjutnya ketika *request* sebelumnya telah mendapat response dari OP. *token request* untuk mode poll ditunjukkan pada Gambar 2.8.

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

grant_type=urn%3Aopenid%3Aparams%3Agrant-type%3Aciba
  &auth_req_id=1c266114-a1be-4252-8ad1-04986c5b9ac1
```

Gambar 2.8 Poll Token Request

Parameter yang terdapat pada *poll token request* adalah:

- grant_type (required)
  nilai harus *urn:openid:params:grant-type:ciba*
- auth_req_id (required)
  adalah *unique identifier* yang didapat dari *authentication response* (langkah nomor 2) pada Gambar 2.7

- *token response*
  OP akan mengirimkan token ke RP sebagai response dari *token request.* Setelah mendapat token, maka *auth_req_id* tidak dapat digunakan lagi atau telah tidak valid. Berikut adalah *successful token response* pada Gambar 2.9

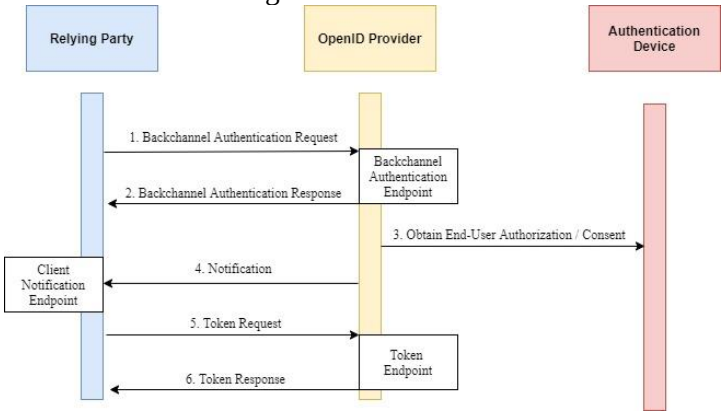
```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "access_token": "G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN",
 "token_type": "Bearer",
 "refresh_token": "4bwc0ESC_IAhflf-ACC_vjD_ltc11ne-8gFPfA2Kx16",
 "expires_in": 120,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2NzcyNiJ9.eyJpc3MiOiJo
   dHRwczovL3NlcnZlci5leGFtcGxlLmNvbSIsInN1YiI6IjI0ODI4OTc2MTAwMSIs
   ImF1ZCI6InM2QmhkUmtxdDMiLCJlbWFpbCI6ImphbmVkb2VAZXhhbXBsZS5jb20i
   LCJleHAiOjE1Mzc4MTk4MDMsImlhdCI6MTUzNzgxOTUwM30.aVq83mdy72ddIFVJ
   LjlNBX-5JHbjmwK-Sn9Mir-blesfYMceIOw6u4GOrO_ZroDnnbJXNKWAg_dxVynv
   MHnk3uJc46feaRIL4zfHf6Anbf5_TbgMaVO8iczD16A5gNjSD7yenT5fslrrW-NU
   _vtmi0s1puoM4EmSaPXCR19vRJyWuStJiRHK5yc3BtBlQ2xwxH1iNP49rGAQe_LH
   fW1G74NY5DaPv-V23JXDNEIUTY-jT-NbbtNHAxnhNPyn8kcO2WOoeIwANO9BfLF1
   EFWtjGPPMj6kDVrikec47yK86HArGvsIIwk1uExynJIv_tgZGE0eZI7MtVb2UlCw
   DQrVlg"
}
```


Gambar 2.9 Successful Poll Token Response

2.5.4.2. Ping Mode

Didalam *Ping* Mode, setelah mendapat *response* dari OP (langkah nomor 2) pada Gambar 2.10, OP akan mendelegasikan *End-User Consent* ke AD, setelah *End-User* mengirimkan *consent* ke OP, maka OP akan menghubungi RP melalui *client motification endpoint*, lalu RP akan melakukan *token request* melalui *token endpoint* dan OP akan mengirimkan token ke RP.



Gambar 2.10 Ping Mode

- *client notification endpoint*
  merupakan endpoint yang disediakan oleh RP dan akan di *hit* oleh OP saat *End-User* telah mengirimkan *consent*. *Request* yang dikirimkan oleh OP ke RP ditunjukkan pada Gambar 2.11

```
POST /cb HTTP/1.1
Host: client.example.com
Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
Content-Type: application/json

{
 "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1"
}
```

Gambar 2.11 Ping Client Notification Endpoint

Parameter yang berada pada *client notification request* adalah:

- auth_req_id
  adalah *unique identifier* yang menandakan bahwa *auth_req_id* tersebut telah dapat diambil token-nya.

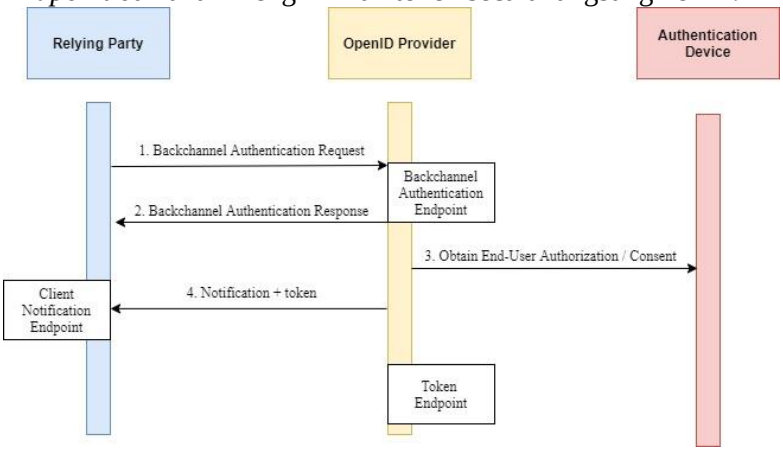
OP juga menambahkan header *Authorization bearer* berisi *client_notification_token* yang dibuat oleh RP saat melakukan *authentication_request* (langkah nomor 1) pada Gambar 2.10

- *token request*
  Setelah mendapat notifikasi dari OP, maka RP akan melakukan validasi terhadap *Authorization bearer* yang dikirimkan OP di header *request*-nya untuk memastikan bahwa OP yang mengirimkan adalah OP yang sesuai dengan *authentication_request* yang dilakukan oleh RP (langkah nomor 1) pada Gambar 2.10.
  token request untuk mode *ping* ditunjukkan pada Gambar 2.8
- *token response*

token response untuk mode *ping* sama dengan mode *poll* yang ditunjukkan pada Gambar 2.9.

2.5.4.3. Push Mode

Didalam *Push* Mode, setelah mendapat response dari OP (langkah nomor 2) pada Gambar 2.12, OP akan mendelegasikan *End-User Consent* ke AD, setelah *End-User* mengirimkan *consent* ke OP, maka OP akan menghubungi RP melalui *Client Notification Endpoint* dan akan mengirimkan token secara langsung ke RP.



Gambar 2.12 Push Mode

- *client_notification_endpoint*
  merupakan endpoint yang disediakan oleh RP dan akan di *hit* oleh OP saat *End-User* telah mengirimkan *consent*. Pada mode *push*, token juga di-ikutsertakan pada parameter *request*-nya. *Request* yang dikirimkan oleh OP ke RP ditunjukkan p*ada* Gambar 2.13

```
POST /cb HTTP/1.1
Host: client.example.com
Authorization: Bearer 8d67dc78-7faa-4d41-aabd-67707b374255
Content-Type: application/json

{
 "auth_req_id": "1c266114-a1be-4252-8ad1-04986c5b9ac1",
 "access_token": "G5kXH2wHvUra0sHlDy1iTkDJgsgUO1bN",
 "token_type": "Bearer",
 "refresh_token": "4bwc0ESC_IAhflf-ACC_vjD_ltc11ne-8gFPfA2Kx16",
 "expires_in": 120,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2NzcyNiJ9.eyJpc3MiOiJ
   odHRwczovL3NlcnZlci5leGFtcGxlLmNvbSIsInN1YiI6IjI0ODI4OTc2MTAwMS
   IsImF1ZCI6InM2QmhkUmtxdDMiLCJlbWFpbCI6ImphbmVkb2VAZXhhbXBsZS5jb
   20iLCJleHAiOjE1Mzc4MTk4MDMsImlhdCI6MTUzNzgxOTUwMywiYXRfaGFzaCI6
   Ild0MGtWRlhNYWNxdm5IZXlVMDAwMXciLCJ1cm46b3Blbmlk0nBhcmFtczpqd3Q
   6Y2xhaW06cnRfaGFzaCI6InNIYWhDdVNwWENSZzVta0REdnZyNHciLCJ1cm46b3
   BlbmlkOnBhcmFtczpqd3Q6Y2xhaW06YXV0aF9yZXFfaWQiOiIxYzI2NjExNC1hM
   WJlLTQyNTItOGFkMS0wNDk4NmM1YjlhYzEifQ.SGB5_a8E7GjwtoYrkFyqOhLK6
   L8-Wh1nLeREwWj30gNYOZW_ZB2mOeQ5yiXqeKJeNpDPssGUrNo-3N-CqNrbmVCb
   XYTwmNB7IvwE6ZPRcfxFV22oou-NS4-3rEa2ghG44Fi9D9fVURwxrRqgyezeD3H
   HVIFUnCxHUou3OOpj6aOgDqKI4Xl2xJ0-kKAxNR8LljUp64OHgoS-UO3qyfOwIk
   IAR7o4OTK_3Oy78rJNT0Y0RebAWyA81UDCSf_gWVBp-EUTI5CdZ1_odYhwB9OWD
   W1A22Sf6rmjhMHGbQW4A9Z822yiZZveuT_AFZ2hi7yNp8iFPZ8fgPQJ5pPpjA7u
   dg"
}
```

Gambar 2.13 Push Client Notification Endpoint

Parameter yang berada pada *client notification request* adalah:

- *auth_req_id*
- *access_token*
- *token_type*
- *refresh_token*
- *expires_in*
- *id_token*

Di dalam id_token terdapat claims yang ditunjukkan pada Gambar 2.14

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "email": "janedoe@example.com",
  "exp": 1537819803,
  "iat": 1537819503,
  "at_hash": "Wt0kVFXMacqvnHeyU0001w",
  "urn:openid:params:jwt:claim:rt_hash": "sHahCuSpXCRg5mkDDvvr4w",
  "urn:openid:params:jwt:claim:auth_req_id":
    "1c266114-a1be-4252-8ad1-04986c5b9ac1"
}
```


Gambar 2.14 ID Token Push Client Notification Endpoint

Claims dari id token berisi sebagai berikut:

- iss
  issuer claims penyedia layanan *authorization* / OP
- aud
  audience claims berisi identifier dari RP
- sub
  berisi identifier dari *End-User*
- email
  berisi alamat email dari user *End-User*
- exp
  waktu kedaluarsa berupa *lifetime* dari *id_token*
- iat
  waktu *signed authentication request* dibuat.
- at_hash
  merupakan hash value dari access token
- urn:openid:params:jwt:claim:rt_hash
  merupakan hash value dari refresh token
- urn:openid:params:jwt:claim:auth_req_id
  berisi auth_req_id

RP harus memastikan bahwa access token yang diterima pada push notification sama dengan access_token di dalam id token.

2.6. Public Key dan Private Key

Private key dan *public key* merupakan pasangan kunci untuk proses enkripsi dan dekripsi. Keduanya selalu berpasangan, jika data di-enkripsi dengan *private key* maka hanya bisa men-*decrypt* menggunakan *public key* atau sebaliknya. *Private key* bersifat rahasia dan hanya diketahui oleh pemiliknya. Sedangkan *public key* tidak bersifat rahasia dan siapapun boleh memilikinya.

2.7. JSON Web Tokens (JWT)

JWT merupakan suatu string yang merepresentasikan *claims* sebagai sebuah JSON object yang dikodekan [5]. JWT memiliki 3 komponen didalamnya, yaitu:

- Header
  Teridiri dari 2 bagian, yaitu tipe token yaitu JWT dan algoritma yang digunakan beruka HMAC, SHA256 atau RSA. Dan bagian ini menggunakan Base64Url encode.
- Payload
  Payload berisi *claims* yaitu data-data yang dikirimkan dari pihak 1 dengan pihak lainnya.
- Signature
  Merupakan *encoding* dari bagian *header*, *payload* dan sebuah *secret* dan di enkripsi menggunakan algoritma pada bagian *header*.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Gambar 2.15 JWT Signature

Bentuk dari JWT adalah sebagai berikut:

header.payload.signature

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

2.8. Bearer Authentication

Bearer Authentication atau *Token Authentication* merupakan skema otentikasi HTTP yang melibatkan *security token* yang disebut *bearer token* cara menggunakan *Bearer Authentication* yaitu dengan menambahkan Authorization : Bearer <token> ke dalam HTTP header. *Bearer Authentication* seharusnya hanya dapat digunakan melalui HTTPS (SSL). Klien menggunakan skema otentikasi *Bearer* untuk mengirimkan *access token* [6].

2.9. RESTful API

RESTful API/ REST API merupakan implementasi API (*Application Programming Interface*). REST (*Representational 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.

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/1, /user/1/photos, /user/1/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.HTTP
3. Response Code
   HTTP *Response Code* adalah kode standarisasi dalam menginformasikan hasil request kepada klien.

4. Format Response
   Setiap *request* yang dilakukan klien akan menerima data response dari server, respon tersebut biasanya berupa data XML ataupun JSON. Setelah mendapatkan data *response* tersebut barulah klien bisa menggunakannya dengan cara memparsing data tersebut dan diolah sesuai kebutuhan.

2.10. Phalcon

Phalcon merupakan *framework* PHP bersifat *open source*. *Phalcon* ditulis sebagai ekstensi PHP menggunakan bahasa pemrograman C yang membuatnya berbeda dari *framework* yang lain. Phalcon menggunakan prinsip-prinsip MVC dan dikembangkan oleh Phalcon Team. *Framework* Phalcon memiliki karakteristik sebagai berikut:

- Semua komponen ditulis dalam bahasa pemrograman C.
- Ada berbagai versi untuk sistem operasi popular: Linux, Windows, dan macOS.
- Kinerja tinggi dan biaya sumber daya *server* rendah.
- Menurut tes, Phalcon adalah salah satu *framework* PHP tercepat.
- Interaksi dengan basis data diimplementasikan dalam bahasa C menggunakan teknologi ORM.

Dalam kemampuan *routing*, Phalcon memiliki dua mode *routing* yaitu mode MVC dan *match only* mode yang secara otomatis mencoba menemukan *controller* dan metodenya berdasarkan input URL. Mode MVC memungkinkan *programmer* untuk secara manual mengkonfigurasi *route* dan mengarahkan *request* ke *controller* dan metode yang sesuai. Router dapat dikonfigurasi dengan kode PHP atau menggunakan anotasi yang ditulis langsung pada *controller*.

Kerangka Phalcon memiliki *template* sendiri yang disebut "Volt". Volt ditulis menggunakan Bahasa C dan dikompilasi bersama dengan Phalcon sebagai ekstensi PHP. Volt juga mengubah semua template ke kode PHP sehingga memungkinkan

untuk menggunakan kode PHP (*raw* PHP) mentah di dalam *template*. Untuk penggunaan kembali kode yang sudah ditulis, Volt menawarkan berbagai mekanisme. Kode dapat dibagi menggunakan operator {% blok %}. Fungsionalitas dapat digunakan misalnya untuk membuat *file* master tunggal dan kemudian cukup memasukkan *template* yang lain.

2.11. Microsoft SQL Server

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) yang merupakan produk dari Microsoft. Bahasa *query* utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase. Microsoft SQL Server banyak digunakan pada dunia bisnis, pendidikan, dan juga pemerintahan sebagai solusi penyimpanan data.

Microsoft SQL Server dan Sybase/ASE dapat berkomunikasi lewat jaringan menggunakan protokol TDS (*Tabular Data Stream*). Selain itu, *Microsoft SQL Server* juga mendukung ODBC (*Open Database Connectivity*) dan mempunyai *driver* JDBC untuk bahasa pemrograman Java. Fitur lain dari SQL Server yaitu kemampuannya untuk membuat basis data *mirroring* dan *clustering*.

Untuk implementasi bahasa pemrograman PHP dengan database Microsoft SQL Server, harus dilakukan konfigurasi sesuai sistem operasi yang digunakan. Setiap sistem operasi yang mendukung Microsoft SQL Server memiliki *driver* tersendiri yang harus dipasangkan. Secara umum langkah yang harus dilakukan adalah:

1. Mengunduh ekstensi *Microsoft SQL Server* untuk *PHP*.
2. Mengaktifkan ekstensi *pdo_sqlsrv* pada konfigurasi *PHP*.
3. Melakukan instalasi *Microsoft ODBC Drive*r yang digunakan untuk menjalankan dan memberikan koneksi suatu aplikasi melalui sistem manajemen *database*.

2.12. Redis

Redis adalah *open source No SQL database*. Yaitu *database* yang tidak memiliki struktur dan menyimpan datanya dalam bentuk *Key-Value. Redis* disimpan didalam memori sehingga dalam proses *read write* sangatlah cepat dan biasa digunakan sebagai *cache database.*

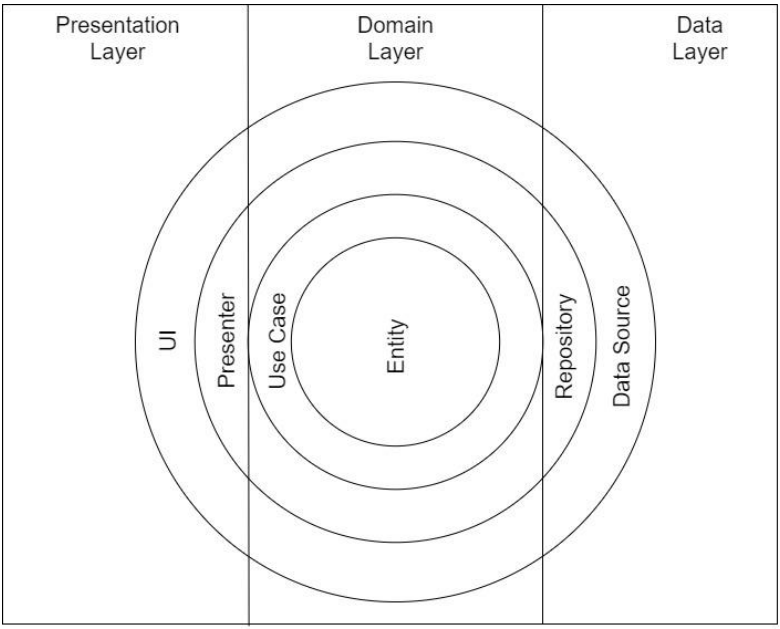
2.13. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) adalah *service* yang disediakan oleh Firebase untuk melakukan komunikasi melalui beberapa platform yang berbeda. Salah satu fitur yang digunakan adalah *Instanst Messaging* yaitu fitur yang memungkinkan mengirimkan pesan melalui *Cloud* dan diterima oleh target melalui *Push Notification* secara *real time.*

2.14. Clean Architecture

Clean Architecture merupakan sebuah arsitektur sistem dimana setiap komponen kode keterikatan yang sangat minimal atau dapat dikatakan sebagai independen, dalam arsitektur ini terdapat 3 *layer* yang terpisah, yaitu *Entities, Presentation Layer, Domain Layer,* dan *Data Layer*. *Presentation Layer* berisi *User Interface* yang dikoordinasi oleh *Presenter/ViewModels* yang mengeksekusi 1 atau lebih *Use cases*. *Presentation Layer* dependen kepada *Domain Layer*. *Domain Layer* adalah lapisan terdalam dari arsitektur dan tidak memiliki dependensi dengan lapisan lainnya. Berisi *Entities, Use cases,* dan *Repository Interfaces*. *Use case* dapat mengkombinasikan dara dari 1 atau lebih dari *Repository Interfaces. Data Layer* berisi implementasi repositori dan 1 atau berbagai *Data Source*. Repositori bertanggung jawab mengkoordinasi data dari berbagai *Data Sources*. *Data Layer* dependen terhadap *Domain Layer*. Contoh struktur *Clean Architecture* terdapat pada Gambar 2.16.

Clean Architecture digunakan untuk membangun *client* atau *Relying Party* agar dapat mengimplementasi alur *Client Initiated Backchannel Authentication.*



Gambar 2.16 Contoh Struktur Clean Architecture

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Bab analisis dan perancangan berisi pembahasan analisis dan perancangan alur *Client Initiated Backchannel Authentication*. Hasil dari proses ini berupa diagram yang 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

Tahap 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

Direktorat Pengembangan Sistem Informasi (DPTSI) menangani pelayanan keluhan terhadap masalah pada akun serta layanan milik *civitas academic* pada Institut Teknologi Sepuluh Nopember, salah satunya adalah jika terjadi masalah pada akun milik mahasiswa, mahasiswa akan melaporkan masalah dan pihak DPTSI akan melakukan pengecekan kepada akun tersebut dengan *Impersonate,* masalah muncul ketika melakukan *Impersonate*, pihak DPTSI dapat melihat dan melakukan perubahan pada akun yang di-*Impersonate*. Di lain pihak, mahasiswa tidak tau bahwa akunnya sedang diakses pihak DPTSI, sehingga muncul permasalahan tentang keamanan, yaitu untuk mencegah pihak DPTSI melakukan penyalahgunaan terhadap akun mahasiswa terkait.

Sistem saat ini belum dapat menangani masalah tersebut, maka dari itu dibangun sistem menggunakan CIBA untuk mendapatkan persetujuan dari mahasiswa terlebih dahulu sebelum melakukan *Impersonate.* Pada protokol CIBA, *Impersonate* hanya dapat dilakukan ketika pemilik akun atau dalam kasus ini adalah mahasiswa telah menyetujui permintaan *Impersonate,* sedangkan jika pemilik akun yang asli tidak menyetujui permintaan tersebut, maka pihak DPTSI tidak dapat melanjutkan proses *Impersonate* dalam artian tidak dapat mengakses pemilik akun mahasiswa, sehingga dalam proses *Impersonate,* dapat dipastikan bahwa pihak DPTSI dapat melihat dan melakukan perubahan pada akun mahasiswa atas persetujuan mahasiswa terkait.

3.1.2. Deskripsi Umum Sistem

Alur *Client Initiated Backchannel Authentication* (CIBA) dibangun menggunakan 2 bagian penting, yaitu *Server* dan *Client*, di dalam *server* dijelaskan tentang bagaimana *server* menangani permintaan CIBA dan melakukan manajemen *access token* untuk alur CIBA. Bagian *server* diimplementasi oleh Adistya Azhar dalam tugas akhirnya yang berjudul "Implementasi Protokol *Client Initiated Bachchannel Authentication (CIBA)* di Sisi Server pada myITS *Single Sign-On".* Sedangkan pada bagian *client* menjelaskan bagaimana cara *client* untuk memulai alur CIBA. Bagian *client* akan dibahas pada tugas akhir ini, secara utuh alur ciba dijelaskan pada Gambar 2.1 dimana bagian *server* adalah pekerjaan milik Adistya dan bagian *client pekerjaan* pada tugas akhir ini.

Untuk menggunakan *Client Initiated Backchannel Authentication*, OP harus mendefinisikan *backchannel authentication endpoint* yang digunakan sebagai *endpoint* untuk RP melakukan CIBA *request*, *backchannel token delivery modes supported* yang digunakan untuk mendefinisikan mode yang di support oleh OP, disini terdapat 3 mode yang dapat digunakan, yaitu *POLL*, *PING*, dan *PUSH.*

Dari segi RP, RP harus mendefinisikan *backchannel token delivery mode* untuk mendaftarkan mode apa yang akan digunakan untuk mendapatkan *token* dari OP, jika RP mendaftarkan dengan mode *PING* atau *PUSH* maka RP juga harus mendefinisikan *backchannel client notification endpoint* sebagai sarana penghubung antara OP dan RP untuk mengirimkan *token.*

Untuk memulai alur CIBA, RP akan meminta *backchannel authentication request* kepada OP, setelah diverifikasi maka RP akan mendapatkan *backchannel authentication response* dari OP. pada saat yang bersamaan OP akan mengirimkan *user consent* ke AD. Dimana AD adalah perangkat otentikator milik *End-User* yang telah terotentikasi bahwa perangkat tersebut benar-benar miliknya. Lalu *End-User* akan mengirimkan *consent* nya ke OP. jika *consent* nya *"No"* maka OP tidak akan membuatkan *token* untuk RP dan akan mengirimkan response error saat RP meminta *token request* sedangkan jika *consent* nya *"Yes"* maka OP akan membuatkan *token* yang dapat diambil oleh RP dengan metode-metode yang telah didefinisikan oleh OP pada *backchannel token delivery modes supported*.

Setelah RP mendapatkan *token* maka token tersebut dapat digunakan sebagai persetujuan untuk melakukan berbagai jenis proses bisnis yang memerlukan persetujuan *End-User*.

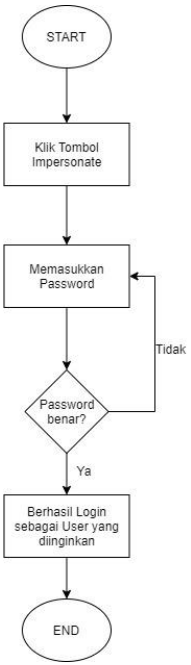
3.1.3. Studi Kasus Sistem Impersonate pada Security Management

Alur *Client Initiated Backchannel Authentication* dapat digunakan untuk berbagai keperluan yang di dalam prosesnya membutuhkan persetujuan *End-User.* Salah satunya adalah sistem *Impersonate* yang terdapat pada aplikasi *Security Management* yang dipelihara oleh DPTSI dan digunakan untuk mengetahui kendala dan error yang terjadi pada akun *End-User* dengan cara seolah-olah login sebagai *End-User* tersebut, dengan demikian pihak DPTSI akan dengan cepat mengetahui error yang terjadi pada akun *End-Use*r. Dalam studi kasus ini, MyITS SSO berperan sebagai OP, MyITS Security Management berperan sebagai RP,

MyITS Authenticator berperan sebagai AD, dan Perangkat milik DPTSI berperan sebagai CD.

3.1.4. Perbandingan Sistem Impersonate *Security Management* dengan CIBA dan tanpa CIBA

Melalui tahapan analisis proses bisnis pada sistem *Impersonate* pada *Security Management* tanpa alur CIBA dan dengan alur CIBA maka didapatkan perbandingan proses bisnis pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Sistem Impersonate Di Security Management Tanpa CIBA

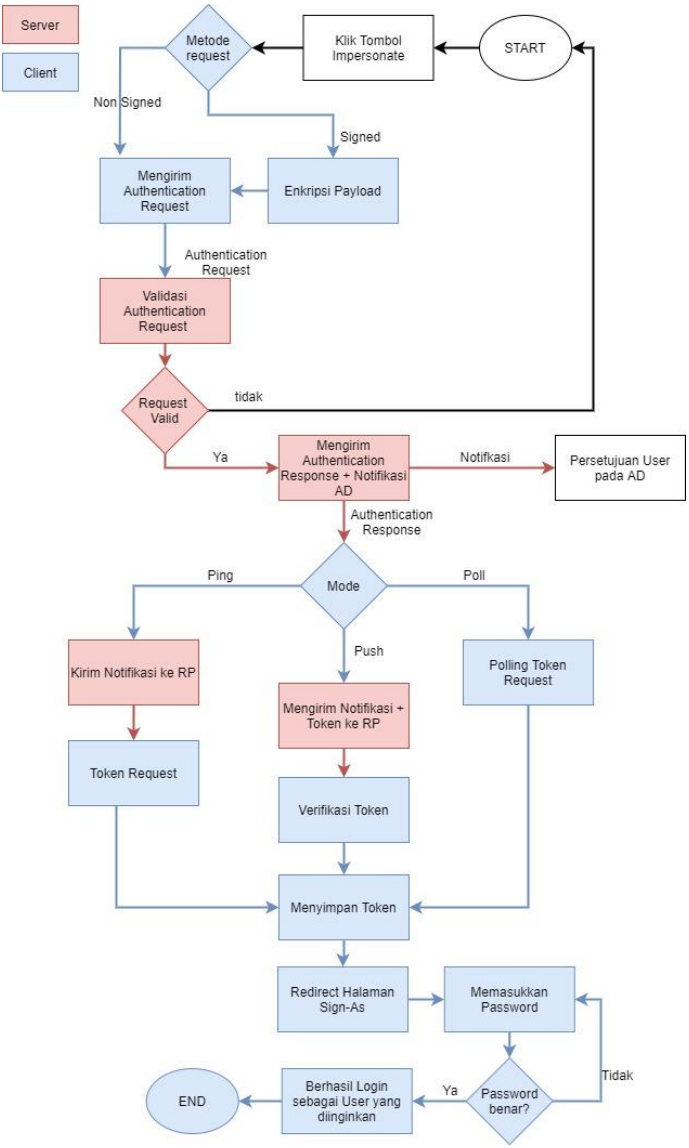
Pada sistem *Impersonate* pada aplikasi *Security Management* yang saat ini, setelah memasukkan password dari user yang akan melakukan impersonate, dan password sesuai maka akan diotentikasi oleh *OpenID Connect* dan berhasil masuk ke sistem

sebagai user yang dituju. Pengguna yang di *Impersonate* tidak tahu bahwa akunnya sedang diakses oleh pihak DPTSI, sehingga dapat memungkinkan terjadinya penyalahgunaan akun, sehingga diperlukan mekanisme agar saat melakukan *Impersonate* pengguna dapat mengetahui bahwa akunnya sedang diakses oleh pihak DPTSI.

Pada sistem *Impersonate* di aplikasi *Security Management* yang telah diimplementasikan alur CIBA, ditunjukkan pada Gambar 3.2, ketika user menekan tombol *Impersonate*, maka MyITS Security Management akan memulai proses alur CIBA. Ketika memulai *Authentication Request*, *Authentication Response* dan *Request Consent* akan sama dari ketiga mode (*Poll*, *Ping*, dan *Push*). Hal yang berbeda adalah ketika mengambil token dari OP. Perbandingan sistem saat sebelum dan sesudah diimplementasikan alur CIBA dapat dilihat pada Tabel 3.1.

Sistem	**Cara melakukan *Impersonate***
Sistem sebelum implementasi alur CIBA	Input Password
Sistem setelah implementasi alur CIBA	Meminta persetujuan user yang akan di *Impersonate*, lalu Input Password.

Tabel 3.1 Perbandingan Sistem Sebelum dan Sesudah Diimplementasikan Alur CIBA



Gambar 3.2 Sistem Impersonate Di Security Management Menggunakan CIBA

3.1.5. 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 mengakomodasi kebutuhan. Diharapkan dengan adanya spesifikasi ini dapat menyesuaikan kebutuhan pengguna terkait dengan sistem otentikasi aplikasi *myITS SSO*.

3.1.6. Kebutuhan Fungsional

Spesifikasi 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 3.2.

No	Kebutuhan Fungsional	Deskripsi
1	Melakukan *CIBA Request* mode *Poll*	Melakukan permintaan otentikasi dan memulai alur CIBA pada mode *Poll*.
2	Melakukan *CIBA Request* mode *Ping*	Melakukan permintaan otentikasi dan memulai alur CIBA pada mode *Ping*.
3	Melakukan *CIBA Request* mode *Push*	Melakukan permintaan otentikasi dan memulai alur CIBA pada mode *Push*.
4	Mengubah Aplikasi klien yang mendukung alur CIBA	Melakukan pengubahan kepada aplikasi klien yang mendukung penggunaan alur CIBA

Tabel 3.2 Kebutuhan Fungsional

3.1.7. Kasus Penggunaan

Bagi Bagian ini menjelaskan analisa dan perancangan kasus-kasus penggunaan yang terdapat pada fitur perangkat lunak. Kasus penggunaan tersebut ditunjukkan pada Gambar 3.3**Error! Reference source not found.** dan tiap kasus penggunaan akan disertakan spesifikasi dan diagram aktivitasnya. Diagram kasus penggunaan dapat dilihat pada Tabel 3.3.

Gambar 3.3 Diagram Kasus Penggunaan

Aktor	Penjelasan
Super Administrator	Merupakan pegawai dan dosen ataupun tendik yang memiliki peran menjadi super administrator aktif yang ada pada lingkungan Institut Teknologi Sepuluh Nopember

Tabel 3.3 Aktor Aplikasi Security Management

3.1.7.1. Kasus Penggunaan Melakukan *CIBA Request* mode *Poll*

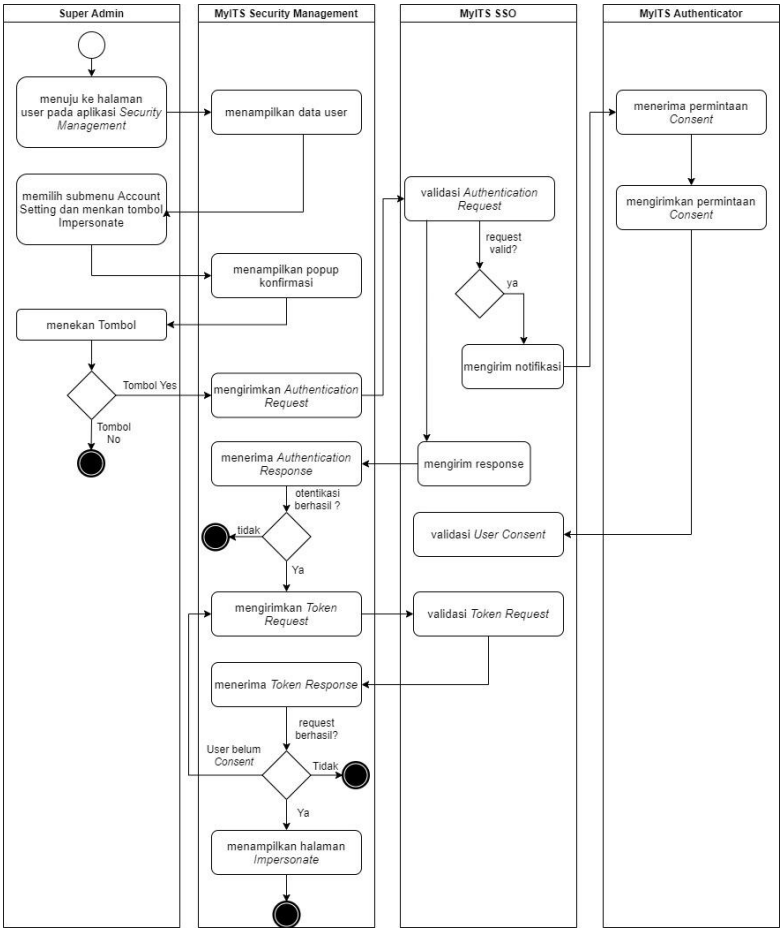
Pada kasus penggunaan melakukan CIBA *Request* mode *Poll*, aktor akan menginisiasi alur *Client Initiated Backchannel Authentication* dengan melakukan impersonate kepada akun user yang dituju. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.4 dan diagram aktivitas pada Gambar 3.4.

Tabel 3.4 Spesifikasi Kasus Penggunaan Melakukan *CIBA Request* mode *Poll*

Komponen	Deskripsi
Nama	Melakukan *CIBA Request* mode *Poll*
Nomor	UC-001
Deskripsi	Kasus penggunaan ini digunakan untuk melakukan CIBA *Request* menggunakan mode *Poll*
Tipe	Fungsional
Aktor	Super Administrator
Kondisi Awal	Aktor telah terotentikasi dan memiliki peran sebagai Super Administrator, MyITS Security Management terdaftar pada mode *Poll*

Komponen	Deskripsi
Kondisi Akhir	Sistem mendapatkan *token*
Alur Normal	1. Aktor menuju ke halaman user yang akan dituju pada aplikasi *Security Management*. 2. *Browser* menampilkan data user yang akan dituju. 3. Aktor memilih submenu *Account Actions* dan menekan tombol *Impersonate User*. 4. *Browser* menampilkan *Popup* konfirmasi aksi *impersonate.* 5. Aktor menekan tombol *Yes.* 6. Sistem melakukan *Authentication Request.* 7. MyITS SSO mengirimkan response. *Authentication Request* berhasil. 8. MyITS SSO mengirimkan *Request Consent* ke MyITS Authenticator. 9. Sistem melakukan *Token Request.* 10. MyITS SSO mengirimkan response. *Token Request* berhasil. 11. Browser me-*redirect* ke halaman *sign-as.*
Alur Alternatif	- 10.a. User belum melakukan *Consent.* 10.a.1. Kembali ke point 9.
Eksepsi	- 5.a. Aktor menekan tombol *No.* - 7.a. MyITS SSO mengirimkan response *Authentication Request* gagal. - 10.b. User menolak CIBA *request.* 10.b.1. Browser menampilkan *Popup* CIBA ditolak. - 10.c. User belum melakukan *Consent* sampai *request* kedaluarsa.

Komponen	Deskripsi
	10.c.1. Browser menampilkan *Popup* CIBA *expired*.



Gambar 3.4 Diagram Aktivitas Penggunaan Melakukan *CIBA Request* Mode *Poll*

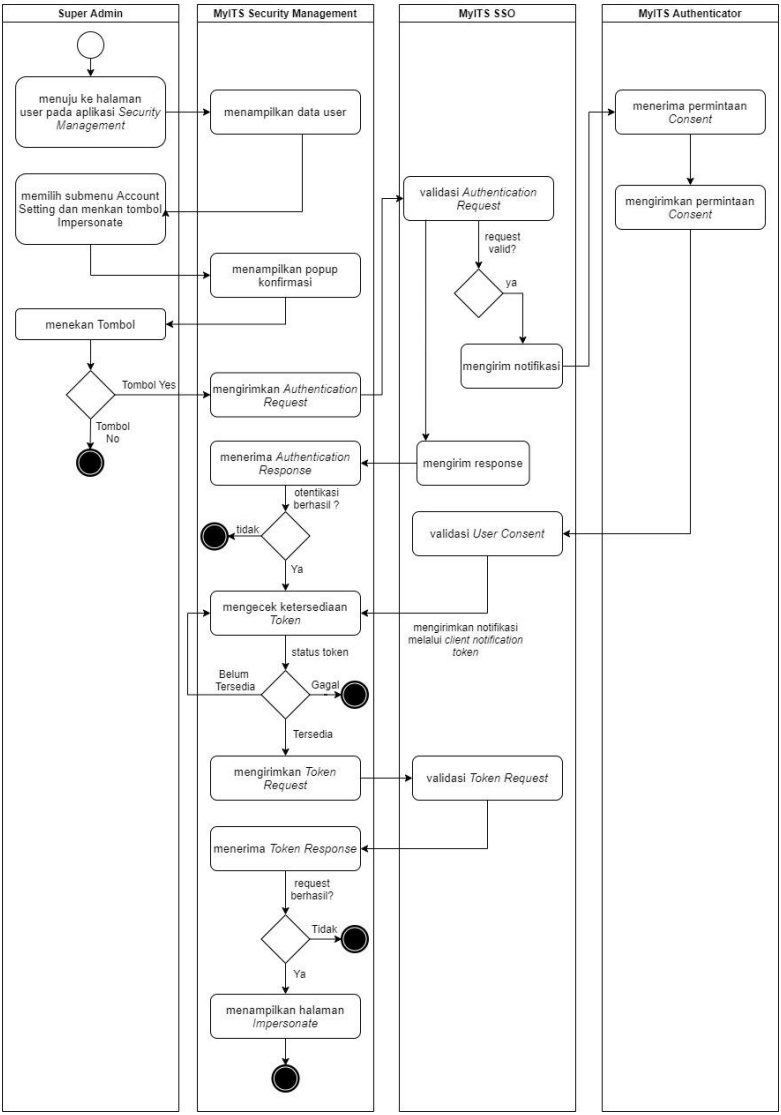
3.1.7.2. Kasus Penggunaan Melakukan *CIBA Request* mode *Ping*

Pada kasus penggunaan melakukan CIBA *Request* mode *Ping*, aktor akan menginisiasi alur *Client Initiated Backchannel Authentication* dengan melakukan impersonate kepada akun user yang dituju. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.5 dan diagram aktivitas pada Gambar 3.5.

Tabel 3.5 Spesifikasi Kasus Penggunaan Melakukan CIBA Request mode Ping

Komponen	Deskripsi
Nama	Melakukan *CIBA Request* mode *Ping*
Nomor	UC-002
Deskripsi	Kasus penggunaan ini digunakan untuk melakukan CIBA *Request* menggunakan mode *Ping*
Tipe	Fungsional
Aktor	Super Administrator
Kondisi Awal	Aktor telah terotentikasi dan memiliki peran sebagai Super Administrator, MyITS Security Management terdaftar pada mode *Ping*
Kondisi Akhir	Sistem mendapatkan *token*
Alur Normal	1. Aktor menuju ke halaman user yang akan dituju pada aplikasi *Security Management*. 2. *Browser* menampilkan data user yang akan dituju.

Komponen	Deskripsi
	3. Aktor memilih submenu *Account Actions* dan menekan tombol *Impersonate User*. 4. *Browser* menampilkan *Popup* konfirmasi aksi *impersonate.* 5. Aktor menekan tombol *Yes.* 6. Sistem melakukan *Authentication Request.* 7. MyITS SSO mengirimkan response. *Authentication Request* berhasil. 8. MyITS SSO mengirimkan *Request Consent* ke MyITS Authenticator. 9. Sistem mengecek ketersediaan token. 10. Token tersedia melalui notifikasi dari MyITS SSO. 11. Sistem melakukan *Token Request.* 12. MyITS SSO mengirimkan response. *Token Request* berhasil. 13. Browser me-*redirect* ke halaman *sign-as.*
Alur Alternatif	- 10.a. Token belum tersedia 10.a.1 Kembali ke poin 9
Eksepsi	- 5.a. Aktor menekan tombol *No.* - 7.a. MyITS SSO mengirimkan response *Authentication Request* gagal. - 10.b. User belum melakukan *Consent* sampai *request* kedaluarsa. 10.b.1. Browser menampilkan *Popup* CIBA *expired*. - 10.c. User menolak CIBA *request.* 10.c.1. Browser menampilkan *Popup* CIBA ditolak.



Gambar 3.5 Diagram Aktivitas Penggunaan Melakukan *CIBA Request* Mode *Ping*

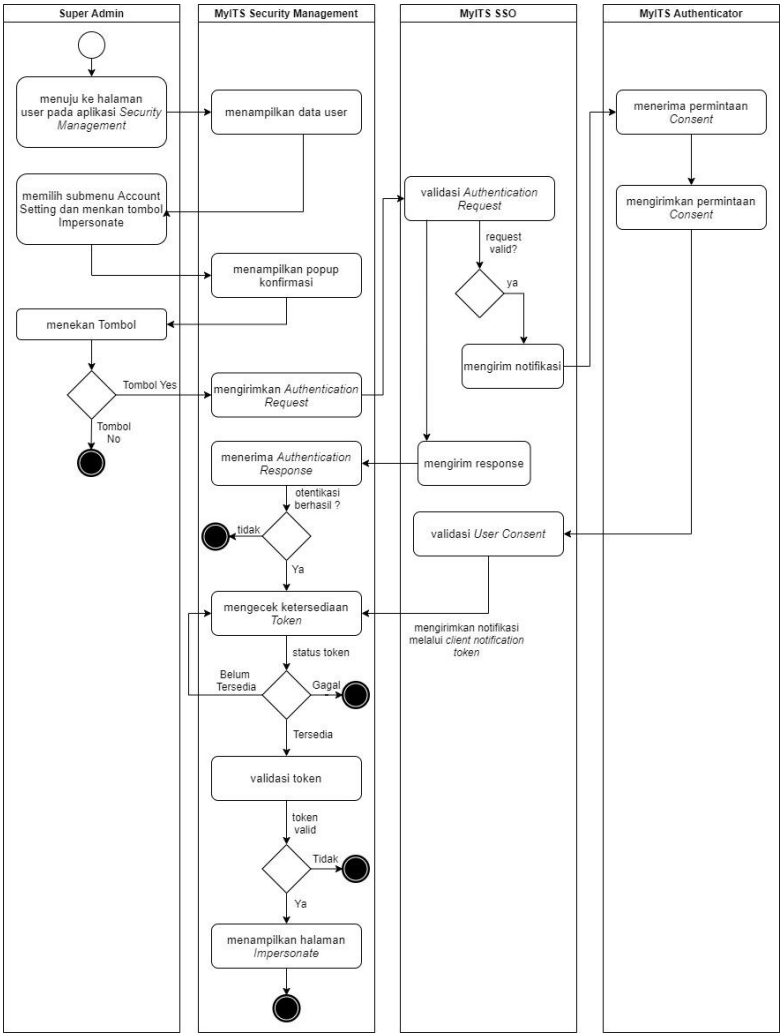
3.1.7.3. Kasus Penggunaan Melakukan *CIBA Request* mode *Push*

Pada kasus penggunaan melakukan CIBA *Request* mode *Push*, aktor akan menginisiasi alur *Client Initiated Backchannel Authentication* dengan melakukan impersonate kepada akun user yang dituju. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.6 dan diagram aktivitas pada Gambar 3.6.

Tabel 3.6 Spesifikasi Kasus Penggunaan Melakukan *CIBA Request* mode *Push*

Komponen	Deskripsi
Nama	Melakukan *CIBA Request* mode *Push*
Nomor	UC-003
Deskripsi	Kasus penggunaan ini digunakan untuk melakukan CIBA *Request* menggunakan mode *Push*
Tipe	Fungsional
Aktor	Super Administrator
Kondisi Awal	Aktor telah terotentikasi dan memiliki peran sebagai Super Administrator, MyITS Security Management terdaftar pada mode *Push*
Kondisi Akhir	Sistem mendapatkan *token*
Alur Normal	1. Aktor menuju ke halaman user yang akan dituju pada aplikasi *Security Management*. 2. *Browser* menampilkan data user yang akan dituju. 3. Aktor memilih submenu *Account Actions* dan menekan tombol *Impersonate User*.

Komponen	Deskripsi
	4. *Browser* menampilkan *Popup* konfirmasi aksi *impersonate.* 5. Aktor menekan tombol *Yes.* 6. Sistem melakukan *Authentication Request.* 7. MyITS SSO mengirimkan response. *Authentication Request* berhasil. 8. MyITS SSO mengirimkan *Request Consent* ke MyITS Authenticator. 9. Sistem mengecek ketersediaan token. 10. Token tersedia melalui notifikasi dari MyITS SSO. 11. Sistem mengecek kevalidan *Token.* 12. Browser me-*redirect* ke halaman *sign-as.*
Alur Alternatif	- 10.a. Token belum tersedia. 10.a.1 Kembali ke poin 9.
Eksepsi	- 5.a. Aktor menekan tombol *No.* - 7.a. MyITS SSO mengirimkan response *Authentication Request* gagal. - 10.b. User belum melakukan *Consent* sampai *request* kedaluarsa. 10.b.1. Browser menampilkan *Popup* CIBA *expired*. - 10.c. User menolak CIBA *request.* 10.c.1. Browser menampilkan *Popup* CIBA ditolak. - 11.a.1. Token Tidak Valid.



Gambar 3.6 Diagram Aktivitas Penggunaan Melakukan *CIBA Request* Mode *Push*

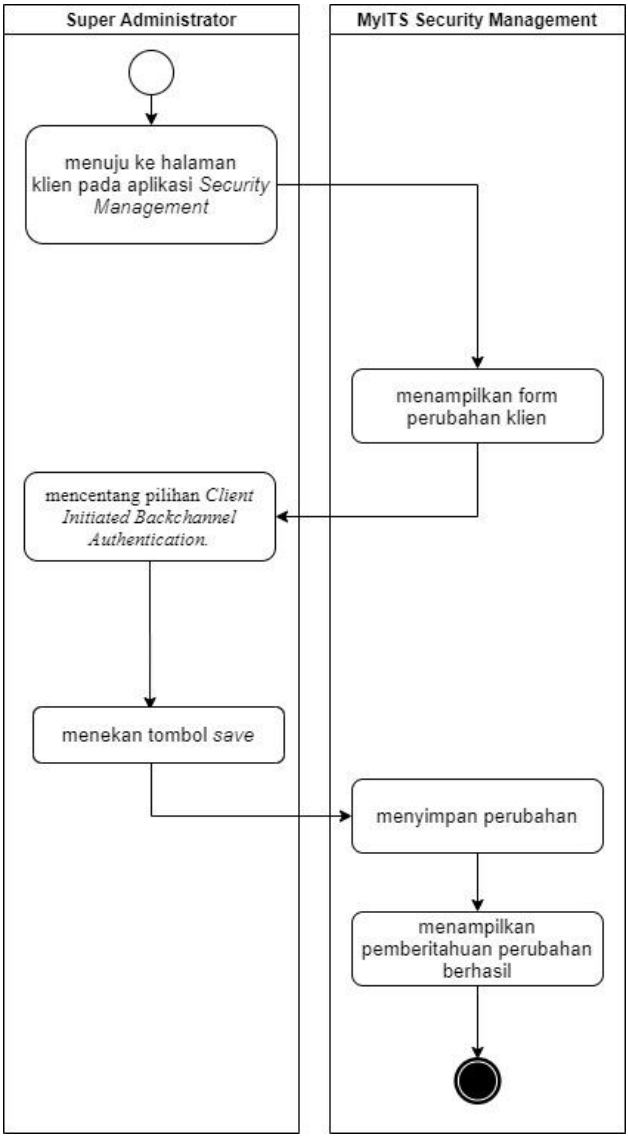
3.1.7.4. Kasus Penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA

Pada kasus penggunaan Mengubah Aplikasi klien yang mendukung alur CIBA, aktor akan memilih ketersediaan aplikasi klien yang mendukung alur CIBA dengan melakukan perubahan pada menu aplikasi klien. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.7 dan diagram aktivitas pada Gambar 3.7. Klien pada Tabel 3.7 adalah MyITS *Security Management*. Super Administrator akan melakukan perubahan pada klien dengan mencentang Client Initiated Backchannel Authentication pada form dan memilih mode yang tersedia (Poll, Ping, dan Push). Hal itu menunjukkan bahwa klien akan menggunakan alur CIBA.

Tabel 3.7 Spesifikasi Kasus Penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA

Komponen	Deskripsi
Nama	Mengubah Aplikasi klien yang mendukung alur CIBA
Nomor	UC-004
Deskripsi	Kasus penggunaan ini digunakan untuk melakukan perubahan ketersediaan aplikasi klien pada alur CIBA
Tipe	Fungsional
Aktor	Super Administrator
Kondisi Awal	Klien tidak mendukung alur CIBA
Kondisi Akhir	Sistem melakukan perubahan pada aplikasi klien sehingga klien mendukung alur CIBA

Komponen	Deskripsi
Alur Normal	1. Aktor menuju ke halaman klien yang akan diubah pada aplikasi *Security Management*. 2. *Browser* menampilkan form perubahan klien. 3. Aktor mencentang pada pilihan *Client Initiated Backchannel Authentication*. 4. Aktor menekan tombol *Save* 5. Sistem menyimpan perubahan 6. Sistem menampilkan pemberitahuan perubahan berhasil
Alur Alternatif	-
Eksepsi	



Gambar 3.7 Diagram Aktivitas Penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA

3.2. Perancangan

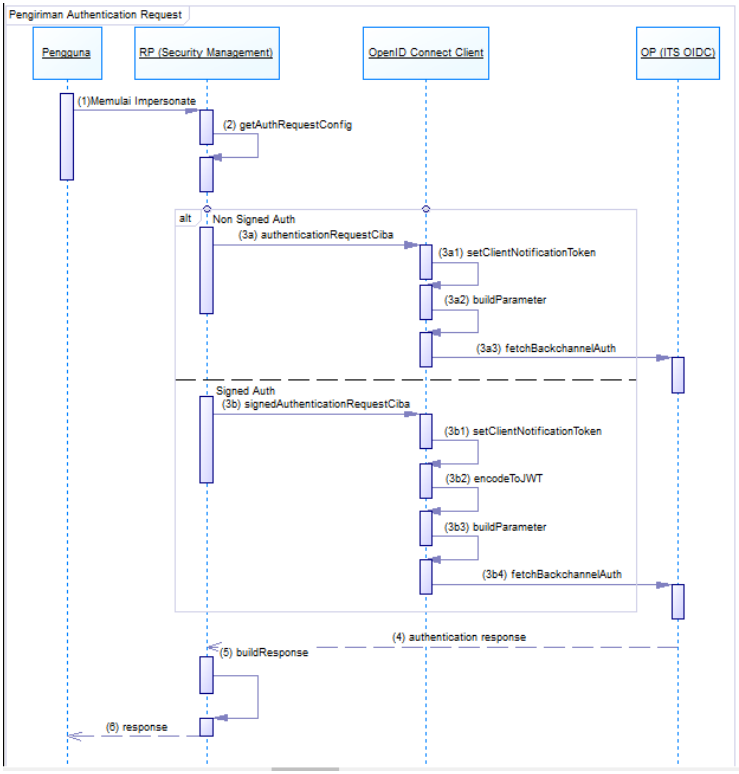
Pada sub bab perancangan akan dijelaskan mengenai arsitektur sistem yang digunakan, perancangan diagram kelas, perancangan basis data, dan perancangan antarmuka.

3.2.1. Perancangan Arsitektur Sistem

Arsitektur sistem yang digunakan dalam tugas akhir ini yaitu menggunakan kerangka kerja (*framework*) *Phalcon* dengan desain arsitektur *clean architecture*. *Presentation Layer* terdiri dari *View*, *View Model* dan *Controller*. *Domain Layer* terdiri dari *Use Case*, *Entity* dan *Repository Interface*. *Data Layer* terdiri dari *Repository*.

3.2.2. Perancangan Sistem Pengiriman *Authentication Request*

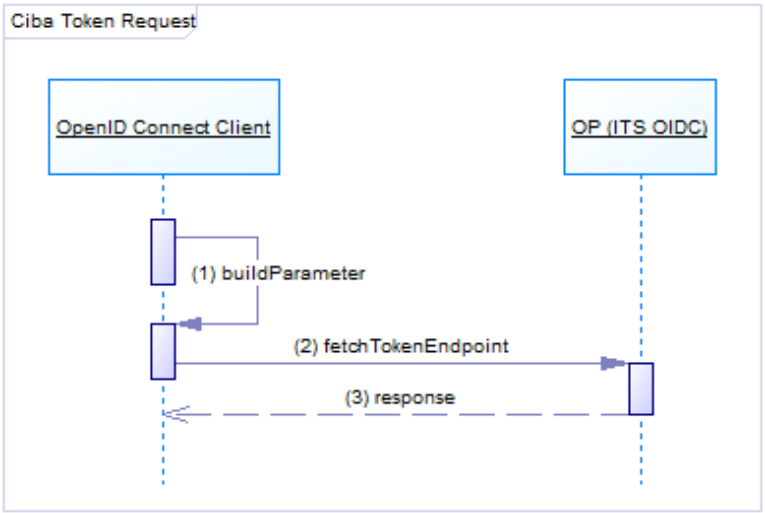
Sistem pengiriman *Authentication Request* memiliki *flow* yang digunakan dapat dilihat pada Gambar 3.8.



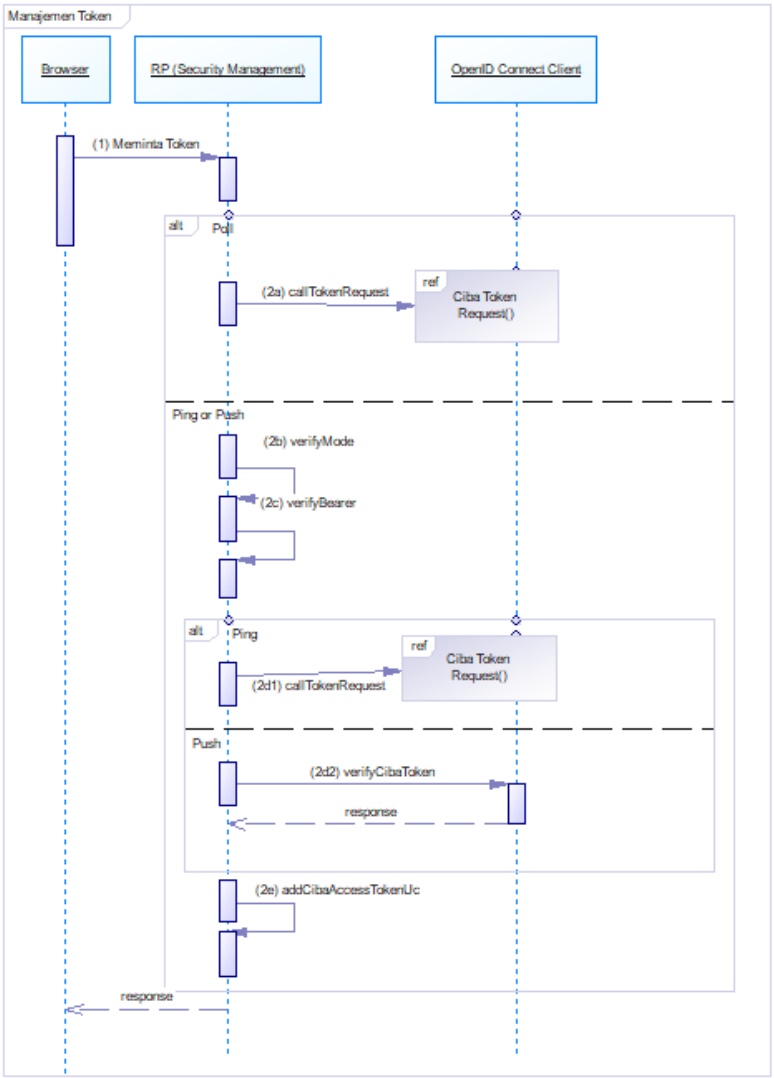
Gambar 3.8 Sequence Diagram Pengiriman *Authentication Request*

3.2.3. Perancangan Sistem Manajemen Token

Sistem manajemen Token memiliki beberapa *flow*. *Flow* yang digunakan dapat dilihat pada Gambar 3.9 dan Gambar 3.10.



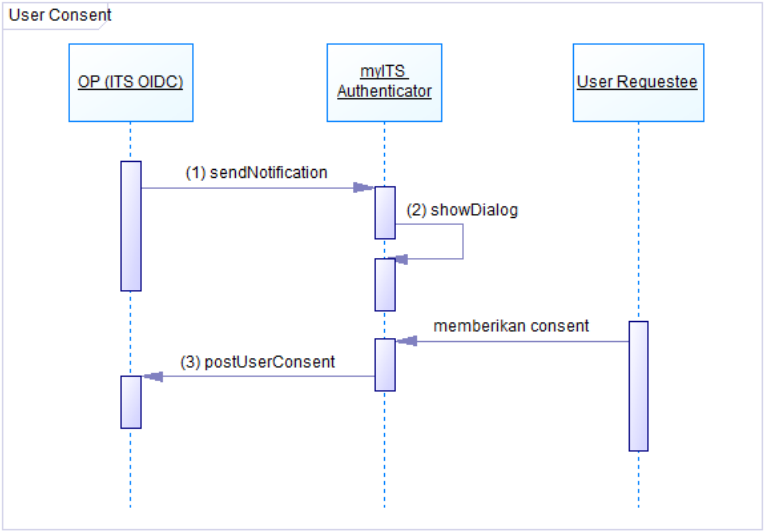
Gambar 3.9 Sequence Diagram Token Request



Gambar 3.10 Sequence Diagram manajemen Token

3.2.4. Perancangan Sistem User Consent Pada Authorization Device

Sistem menerima notifikasi dan melakukan *User Consent* memiliki *flow* yang digunakan dapat dilihat pada Gambar 3.11.



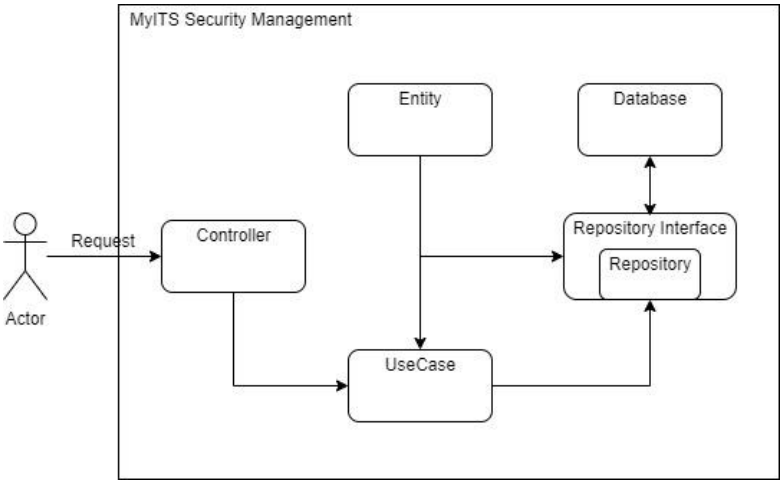
Gambar 3.11 Sequence Diagram User Consent Pada Authorization Device

3.2.5. Perancangan Desain Clean Architecture

Sistem akan menerapkan *clean architecture* sebagai desain arsitekturnya. Didalam *clean architecture*, terdapat beberapa komponen sebagai berikut:

- *Presentation Layer*
- *Domain Layer*
- *Data Layer*

Secara umum, *clean architecture* akan menerima request melalui *controller*, lalu *controller* akan memanggil *usecase* untuk menjalankan fungsionalitasnya. Usecase akan memanggil *repository* yang telah mengimplementasi *repository interface*. Lalu *repository* akan berinteraksi dengan *database Entity* dapat diakses melalui *usecase* ataupun *repository*. Skema clean architecture ditunjukkan pada Gambar 3.12.



Gambar 3.12 perancangan skema clean architecture

Di dalam *Presentation Layer*, akan diterapkan kelas-kelas sebagai berikut:

- **CibaController**
  adalah *controller* yang bertujuan untuk mengatur dan penghubung dengan *entity* pada *domain layer*.
- **ClientController**
  adalah *controller* yang bertujuan untuk mengatur klien (MyITS Security Management). Terutama untuk mengatur ketersediaan klien untuk alur CIBA.

Pada *Domain Layer*, akan diterapkan kelas-kelas sebagai berikut:

- **Entity**
  Entitas pada alur CIBA akan digambarkan melalui kelas *entity*
  a. CibaAccessToken
  b. ClientNotificationToken
  c. Client
- **Repository Interface**
  Interface akan digunakan untuk manajemen atas *entity*, digunakan interface sehingga implementasi dari repositorynya lebih dinamis.
  a. CibaAccessTokenRepositoryInterface
  b. ClientNotificationTokenRepositoryInterface
  c. ClientRepositoryInterface
- **UseCase**
  Usecase akan melakukan fungsionalitasnya sesuai usecase yang tersedia, meskipun tidak selalu fungsionalitas pada *usecase diagram* akan dijalankan pada 1 kelas *usecase.*
  a. AddCibaAccessToken
  b. GetCibaAccessToken
  c. AddClientNotificationToken
  d. FindClientNotificationToken
  e. FindClient
  f. EditClient

Pada Data Layer, akan diterapkan kelas-kelas sebagai berikut:

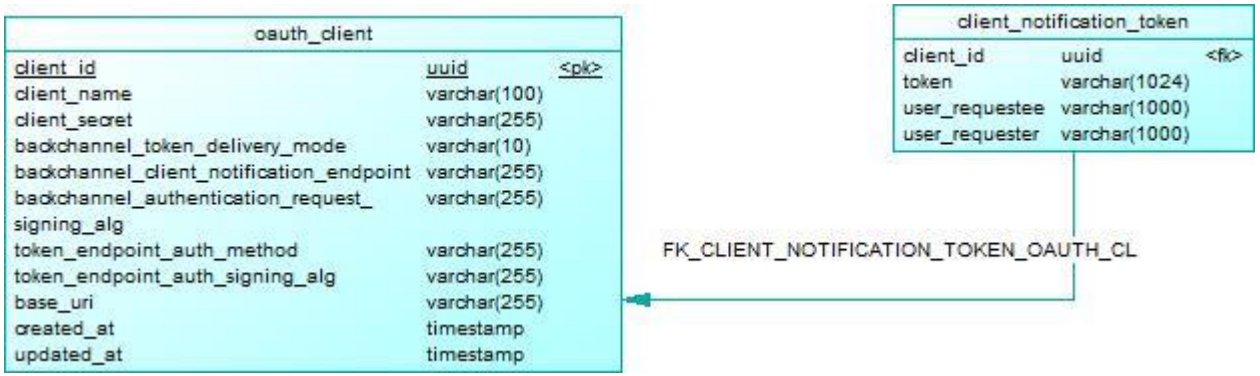
- **Repository**
  Kelas *repository* akan mengimplementasi kelas *repository interface*, *repository* akan melakukan manajemen pada *entity.*
  a. CibaAccessTokenRepository
  b. ClientNotificationTokenRepository
  c. ClientRepository

3.2.6. Perancangan Basis Data

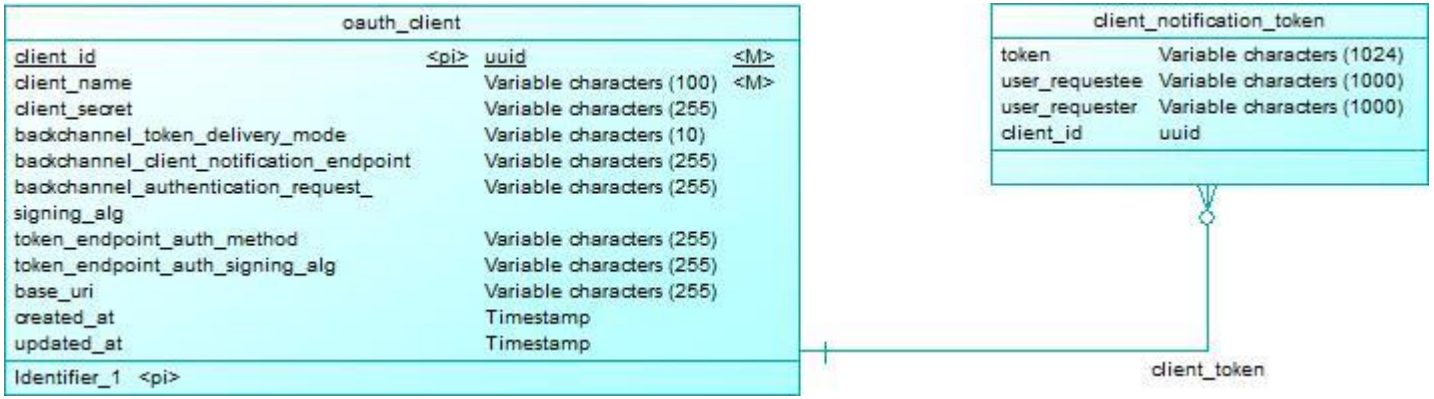
Dalam membuat suatu aplikasi berupa sistem informasi, diperlukan analisis kebutuhan berupa perancangan basis data. Basis data yang digunakan nantinya adalah SQL Server yang dipilih karena mudah digunakan dan mendukung penggunaan *Universally Unique Identifiers* (selanjutnya disebut UUID).

Rancangan basis data ditampilkan dalam bentuk *Conceptual Data Model* (CDM) ditunjukan pada Gambar 3.14, dan *Physical Data Model* (PDM) ditunjukan pada Gambar 3.13 yang dibagi menjadi beberapa bagian untuk mempermudah menampilkan relasi antar objek dan penjelasan masing-masing objek pada diagram.

Diagram *CIBA Client* berisi tabel-tabel yang digunakan untuk proses alur *Client Initiated Backchannel Authentication*. Masing-masing tabel pada diagram akan dijelaskan sebagai berikut:



Gambar 3.13 Diagram PDM CIBA Flow



Gambar 3.14 Diagram CDM CIBA Flow

3.2.6.1. Tabel OAuth Client

Tabel *oauth_client* adalah tabel yang menyimpan informasi tentang klien (RP). Masing-masing klien akan dilayani oleh satu *Provider*. Dalam CIBA *client*, klien digunakan untuk mengidentifikasi penggunaan *CIBA request*, *request* antara masing-masing klien harus dibedakan dengan masing-masing *requester* atau user yang meminta otentikasi menggunakan alur CIBA. *Field backchannel_token_delivery_mode* menyimpan informasi pemberian *token* dari MyITS SSO ke MyITS Security Management. *Field backchannel_client_notification_endpoint* menyimpan url *client notification* untuk mengirimkan notifikasi dari MyITS SSO ke MyITS Security Management setelah pemberian *consent*. *Field backchannel_authentication_request_signing_alg* menyimpan algoritma yang digunakan untuk pembentukan Signed *Authentication Request*. *Field token_endpoint_auth_method* menyimpan informasi tentang metode otentikasi yang digunakan oleh klien.

3.2.6.2. Tabel Client Notification Token

Tabel *client_notification_token* berisi tentang daftar *client_notification_token* yang telah dibuat oleh klien. *Field* user*_requester* adalah pengguna yang memegang kendali CD. *Field user_requestee* adalah pengguna yang memegang kendali MyITS Authenticator.

3.2.7. Perancangan Antarmuka

Pada bagian ini akan dibahas mengenai rancangan antarmuka bagi pengguna untuk memenuhi kasus penggunaan yang sudah dirancang, peracangan antarmuka terdiri dari 2 bagian, yaitu antarmuka pada *myITS Security Management* dan pada aplikasi *myITS Authenticator*.

3.2.7.1. Halaman User Detail myITS Security Management

Halaman *User Detail* digunakan pada kasus penggunaan melakukan melakukan *Authentication Request* (UC-001). Halaman ini berisi detail dari pengguna, dalam hal ini digunakan untuk melakukan *Impersonate* untuk memulai proses CIBA. Rancangan antarmuka dapat dilihat pada Gambar 3.15 dan Tabel 3.8.

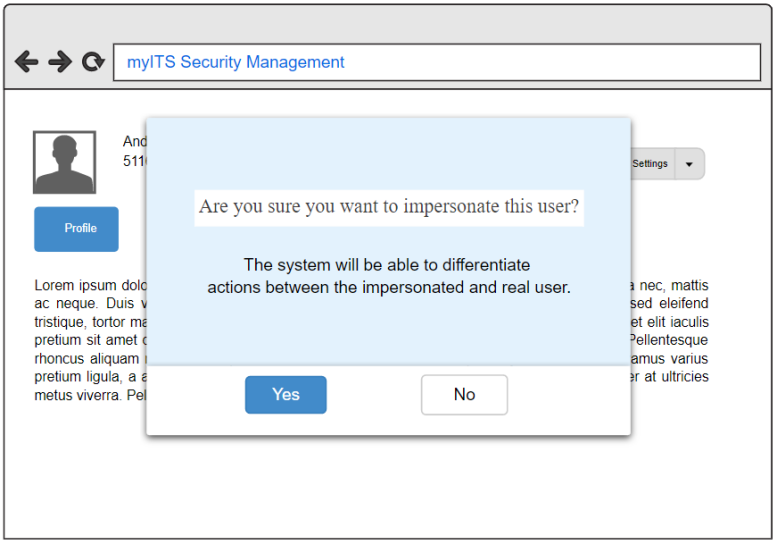
Gambar 3.15 Rancangan Antarmuka User Detail

Tabel 3.8 Penjelasan Antarmuka User Detail

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*Account settings*	*Dropdown*	Menampilkan dropdown berisi pilihan action dari user	*Button Clicked*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
2	*Impersonate button*	*Button*	Tombol aksi untuk memulai alur impersonate, berada dalam komponen dropdown account settings	*Button Clicked*

Selain itu, terdapat pula rancangan dan penjelasan terkait *impersonate button* yang akan menampilkan popup setelah tombol ditekan yang dapat dilihat Gambar 3.16 dan Tabel 3.9.

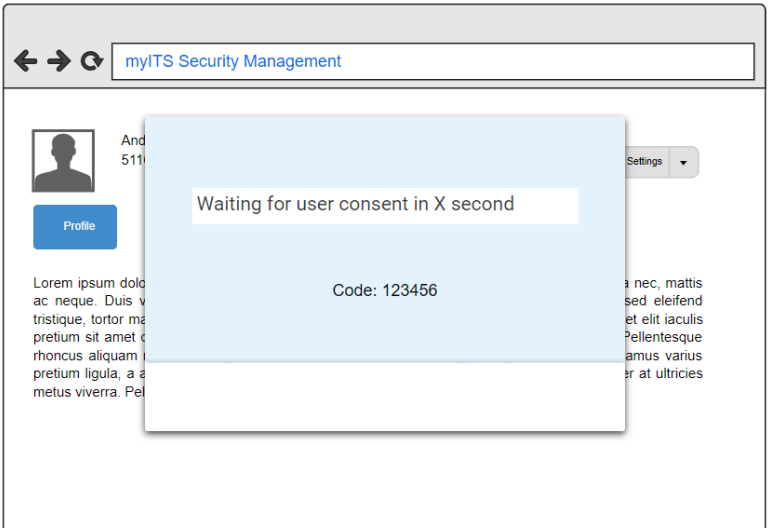


Gambar 3.16 Rancangan Antarmuka Popup Impersonate

Tabel 3.9 Penjelasan Antarmuka Popup Impersonate

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*promptImpersonate*	*Prompt*	Menampilkan prompt agar pengguna memilih aksi untuk memulai proses impersonate.	*Popup*
2	*yesButton*	*Button*	Tombol memulai proses impersonate.	*Button Clicked*
3	*noButton*	*Button*	Tombol membatalkan proses impersonate.	*Button Clicked*

Terdapat pula rancangan dan penjelasan terkait *impersonate button* yang akan menampilkan popup menunggu *consent* dari user pada Gambar 3.17 dan Tabel 3.10.



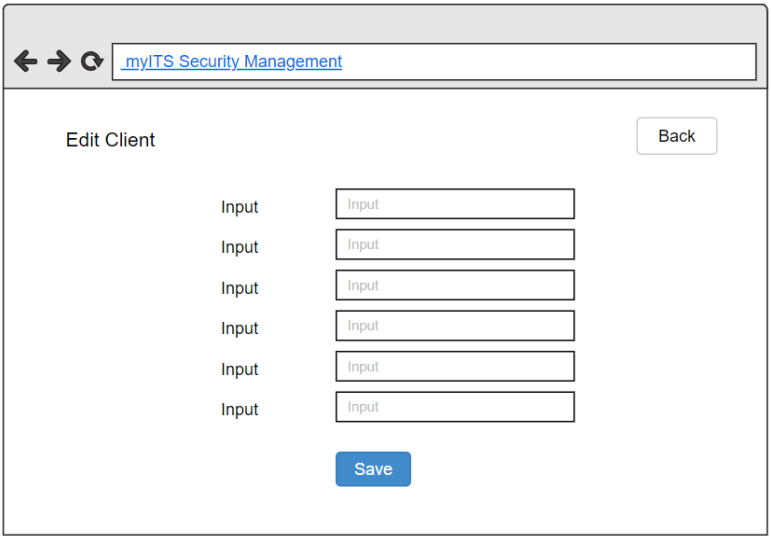
Gambar 3.17 Rancangan Antarmuka Popup *Waiting User Consent*

Tabel 3.10 Penjelasan Antarmuka Popup *Waiting User Consent*

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*promptImpersonate*	*Prompt*	Menampilkan prompt menunggu proses *user consent*.	*Popup*
2	*X Text*	*Text*	Waktu kedaluarsa dari CIBA *request*	*Text*
3	*Code Text*	*Text*	Binding message yang mucul pada device user yang di impersonate.	*Test*

3.2.7.2. Halaman Edit Client myITS Security Management

Halaman *edit client* digunakan pada kasus penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA (UC-004). Halaman ini berisi detail dari klien, dan digunakan untuk mengubah detail dari klien. Rancangan antarmuka dapat dilihat pada Gambar 3.18 dan Tabel 3.11.



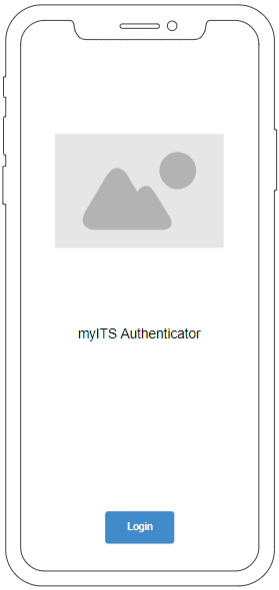
Gambar 3.18 Rancangan Antarmuka Edit Client

Tabel 3.11 Penjelasan Antarmuka Edit Client

No	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Jenis Masukan/Keluaran
1	*Back*	*Button*	Tombol kembali ke detail aplikasi	*Button Clicked*
2	*Save*	*Button*	Tombol simpan	*Button Clicked*
3	*Form Input*	*Input*	Sebagai Input form	-

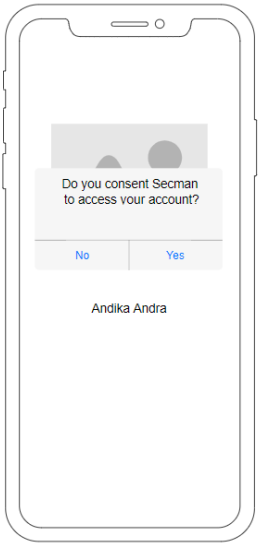
3.2.7.3. MyITS Authenticator

Layout myITS *Authenticator* berisi tampilan dari aplikasi myITS Authenticator yang digunakan sebagai perangkat otentikator oleh user. Rancangan antarmuka dapat dilihat pada Gambar 3.19.



Gambar 3.19 Rancangan Antarmuka myITS Authenticator

Selain itu jika terdapat permintaan *consent* dari aplikasi maka akan muncul *prompt*. Rancangan antarmuka dapat dilihat pada Gambar 3.20.



Gambar 3.20 Rancangan Antarmuka myITS Authenticator request consent

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem sesuai dengan analisis dan perancangan proses otentikasi aplikasi *myITS SSO* dengan alur *Client Initiated Backchannel Authentication*.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti yang ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi

Perangkat	Spesifikasi
Perangkat keras	Prosesor: Intel® Core™ i5 2.3GHz Memori: 8192 MB
Perangkat lunak	Sistem Operasi: Ubuntu 18.04 LTS Perangkat Pengembang: Phalcon, Visual Studio Code, DBeaver, Perangkat Perancang Diagram: StarUML, Draw.io Perangkat Database: Microsoft SQL Server 2017

4.2. Implementasi Clean Architecture

Implementasi Sistem yang dibuat memiliki lapisan-lapisan yang direpresentasikan dalam kelas, *Presentation Layer* terdiri dari *View* sebagai lapisan antarmuka pengguna, *View Model* dan *Controller* sebagai tempat untuk menerima *request* yang dikirim oleh aplikasi *client* dan mengirim balik *response*. *Domain Layer* terdiri dari *Use Case* menyimpan *Application Logic* yang mengorkestrasikan alur dari sebuah logika bisnis, *Entity* merupakan gabungan dari *Business Rule* yang merupakan *Domain*

dari aplikasi dan *Repository Interface* sebagai interface dari *Repository*. *Data Layer* terdiri dari *Repository* sebagai tempat untuk melakukan pengelolaan terhadap basis data. Pada tugas akhir ini, *Clean Architecture* diterapkan dengan menambahkan kelas *entity, usecase, repository interface* dan *controller* pada aplikasi MyITS Security management.

4.2.1. Implementasi Kelas *Entity*

Implementasi *entity* dalam aplikasi berada pada package atau folder Models dimana terdapat kelas-kelas *entity*.

4.2.1.1. Entity CIBA Access Token

Entity CIBA Access Token merepresentasikan *Access Token* yang diterima oleh MyITS Security Management sebagai bentuk persetujuan dari *User Requestee* atau user yang diminta persetujuannya. Penggunaan entity ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace MyIts\Secman\Domain\Models;
4.
5.  class CibaAccessToken
6.  {
7.      protected $authReqId;
8.      protected $accessToken;
9.      protected $expiryTime;
10.     public function __construct($authReqId, $accessToken,
    $expiryTime = 30)
11.     {
12.         $this->authReqId = $authReqId;
13.         $this->accessToken = $accessToken;
14.         $this-> expiryTime = $expiryTime;
15.     }
16.     public function setAuthReqId($authReqId)
17.     {
18.         $this->authReqId = $authReqId;
19.     }
```

```
20.     public function setAccessToken($accessToken)
21.     {
22.         $this->accessToken = $accessToken;
23.     }
24.     public function setExpiryTime($expiryTime)
25.     {
26.         $this->expiryTime = $expiryTime;
27.     }
28.     public function getAuthReqId()
29.     {
30.         return $this->authReqId
31.     }
32.     public function getAccessToken()
33.     {
34.         return $this->accessToken;
35.     }
36.     public function getExpiryTime()
37.     {
38.         return $this->expiryTime;
39.     }
40. }
```

Kode Sumber 4.1 Entity CibaAccessToken.php

4.2.1.2. Entity Client Notification Token

Entity Client Notification Token merepresentasikan Client *Notification Token* yang dibuat oleh MyITS Security Management sebagai penanda CIBA *request* yang dimiliki oleh MyITS Security Management. Penggunaan entity ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace MyIts\Secman\Domain\Models;
4.
5.  class ClienNotificationToken
6.  {
7.      protected $clientId;
8.      protected $token;
9.      protected $requester;
```

```
10.     protected $requestee;
11.
12.     public function __construct($clientId, $token,
    $requester = NULL, $requestee = NULL)
13.     {
14.         $this->clientId = $clientId;
15.         $this->token = $token;
16.         $this->requester = $requester;
17.         $this->requestee = $requestee;
18.     }
19.     public function setClientId($clientId)
20.     {
21.         $this->clientId = $clientId;
22.     }
23.     public function setRequester($requester)
24.     {
25.         $this-> requester = $requester;
26.     }
27.     public function setRequestee($requestee)
28.     {
29.         $this->requestee = $requestee;
30.     }
31.     public function setToken($token)
32.     {
33.         $this->token = $token;
34.     }
35.     public function getCLientId()
36.     {
37.         return $this->clientId
38.     }
39.     public function getRequester()
40.     {
41.         return $this->requester;
42.     }
43.     public function getRequestee()
44.     {
45.         return $this->requestee;
46.     }
47.     public function getToken()
48.     {
49.         return $this->$token;
50.     }
51. }
```

Kode Sumber 4.2 Entity ClientNotificationToken.php

4.2.1.3. Entity Client

Entity Client merepresentasikan Client sebagai MyITS Security Management. Penggunaan entity ini terdapat pada kasus penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA (UC-004).

```
1.  <?php
2.
3.  namespace MyIts\Secman\Domain\Models;
4.
5.  class CLient
6.  {
7.      protected $clientId;
8.      protected $bcAuthAlg;
9.      protected $bcTokenDeliveryMode;
10.     protected $bcClientNotificationEndp;
11.     protected $bcUserCodeParameter;
12.     protected $tokenEndpAuthMethod;
13.     protected $tokenEndpAuthigningAlg
14.
15.     public function __construct($clientId,
            $bcAuthAlg,
            $bcTokenDeliveryMode,
            $bcClientNotificationEndp,
            $bcUserCodeParameter,
            $tokenEndpAuthMethod,
            $tokenEndpAuthMethod)
16.     {
17.         $this->clientId = $clientId;
18.         $this->bcAuthAlg = $bcAuthAlg;
19.         $this->bcTokenDeliveryMode = $bcTokenDeliveryMode;
20.         $this->bcClientNotificationEndp =
    $bcClientNotificationEndp;
21.         $this->bcUserCodeParameter = $bcUserCodeParameter;
22.         $this->tokenEndAuthMethod = $tokenEndAuthMethod;
23.         $this->tokenEndAuthigningAlg =
    $tokenEndAuthigningAlg;
24.     }
25.     public function setCLientId($clientId)
26.     {
```

```
27.         $this->clientId = $clientId;
28.     }
29.     public function setBcAuthAlg($bcAuthAlg)
30.     {
31.         $this->bcAuthAlg = $bcAuthAlg;
32.     }
33.     public function setBcTokenDeliveryMode($bcTokenDeliver
    yMode)
34.     {
35.         $this->bcTokenDeliveryMode = $bcTokenDeliveryMode;
36.     }
37.     public function setBcClientNotificationEndp($bcClientN
    otificationEndp)
38.     {
39.         $this->bcClientNotificationEndp =
    $bcClientNotificationEndp;
40.     }
41.     public function setBcUserCodeParameter($bcUserCodePara
    meter)
42.     {
43.         $this->bcUserCodeParameter = $bcUserCodeParameter;
44.     }
45.     public function setTokenEndpAuthMethod($tokenEndpAuthM
    ethod)
46.     {
47.         $this->tokenEndpAuthMethod = $tokenEndpAuthMethod;
48.     }
49.     public function setTokenEndpAuthigningAlg($tokenEndpAu
    thigningAlg)
50.     {
51.         $this->tokenEndpAuthigningAlg =
    $tokenEndpAuthigningAlg;
52.     }
53.     public function getCLientId()
54.     {
55.         return $this->clientId;
56.     }
57.     public function getBcAuthAlg()
58.     {
59.         return $this->bcAuthAlg;
60.     }
61.     public function getBcTokenDeliveryMode()
62.     {
```

```
63.         return $this->bcTokenDeliveryMode;
64.     }
65.     public function getBcClientNotificationEndp()
66.     {
67.         return $this->bcClientNotificationEndp;
68.     }
69.     public function getBcUserCodeParameter()
70.     {
71.         return $this->bcUserCodeParameter;
72.     }
73.     public function getTokenEndpAuthMethod()
74.     {
75.         return $this->tokenEndpAuthMethod;
76.     }
77.     public function getTokenEndpAuthigningAlg()
78.     {
79.         return $this->tokenEndpAuthigningAlg;
80.     }
81. }
```

Kode Sumber 4.3 Entity Client.php

4.2.2. Implementasi Kelas *UseCase*

Implementasi *usecase* dalam aplikasi berada pada package atau folder usecases dimana terdapat kelas-kelas *usecase* yang merepresentasikan *Application Logic* dari aplikasi.

4.2.2.1. UseCase AddCibaAccessToken

UseCase AddCibaAccessToken digunakan untuk menambahkan atau menyimpan *CIBA access token*. Penggunaan *usecase* ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace MyIts\Secman\UseCases\AddCibaAccessToken;
4.
5.  use Exception;
6.  use MyIts\Secman\Domain\Models\CibaAccessToken;
```

```
7.  use MyIts\Secman\Domain\Contracts\Repositories\
    CibaAccessTokenRepositoryInterface;
8.
9.  class AddCibaAccessToken
10.
11.     protected $repository;
12.     public function __construct(CibaAccessTokenRepositoryI
    nterface $repository)
13.     {
14.         $this->repository = $repository;
15.     }
16.     public function handle(AddCibaAccessTokenRequest
    $request)
17.     {
18.         $accessToken = new CibaAccessToken(
                    $request->getAuthReqId(),
                    $request->getData(),
                    $request->getExpiryTime()
            );
19.         try {
20.             return $this->repository-
    >saveAccessToken($accessToken);
21.         } catch (Exception $exception) {
22.             Throw new Exception(
                    $exception->getMessage()
                );
23.         }
24.     }
25. }
```

Kode Sumber 4.4 Usecase AddCibaAccessTokenUseCase.php

4.2.2.2. UseCase GetCibaAccessToken

UseCase GetCibaAccessToken digunakan untuk mendapatkan *CIBA access token* yang diminta. Penggunaan *usecase* ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace MyIts\Secman\UseCases\GetCibaAccessToken;
```

```
4.
5.  use Exception;
6.  use MyIts\Secman\Domain\Contracts\Repositories\
    CibaAccessTokenRepositoryInterface;
7.
8.  class GetCibaAccessToken
9.  {
10.     protected $repository;
11.     public function __construct(CibaAccessTokenRepositoryI
    nterface $repository)
12.     {
13.         $this->repository = $repository;
14.     }
15.     public function handle(GetCibaAccessTokenRequest
    $request)
16.     {
17.         try {
18.             return $this->repository-
    >getAccessToken($request->getAuthReqId());
19.         } catch (Exception $exception) {
20.             Throw new Exception(
                    $exception->getMessage()
                );
21.         }
22.     }
23. }
```

Kode Sumber 4.5 Usecase GetCibaAccessTokenUseCase.php

4.2.2.3. UseCase AddClientNotificationToken

UseCase AddClientNotificationToken digunakan untuk menambah atau menyimpan *client notification token* yang telah dibuat oleh MyITS Security Management. Penggunaan *usecase* ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace
    MyIts\Secman\UseCases\AddClientNotificationToken;
4.
```

```
5.  use Exception;
6.  use MyIts\Secman\Domain\Models\ClientNotificationToken;
7.  use MyIts\Secman\Domain\Contracts\Repositories\
    ClientNotificationTokenRepositoryInterface;
8.
9.  class AddClientNotificationTokenUseCase
10. {
11.     protected $repository;
12.     protected const MESSAGE_SUCCESS = 'Successfuly added
    client notification token.';
13.     public function __construct(ClientNotificationTokenRep
    ositoryInterface $repository)
14.     {
15.         $this->repository = $repository;
16.     }
17.     public function handle(AddCLientNotificationTokenReque
    st $request)
18.     {
19.         $clientNotificationToken = new
    CLientNotificationToken(
                $request->getClientId(),
                $request->getRequester(),
                $request->getRequestee(),
                $request->getToken()
            );
20.         try {
21.             $this->repository-
    >create($clientNotificationToken);
22.             return new AddCLientNotificationTokenResponse
    (self::MESSAGE_SUCCESS);
23.         } catch (Exception $exception) {
24.             return new AddCLientNotificationTokenResponse
    ($exception->getMessage(), true);
25.         }
26.     }
27. }
```

Kode Sumber 4.6 Usecase AddClientNotificationTokenUseCase.php

4.2.2.4. UseCase FindClientNotificationToken

UseCase FindClientNotificationToken digunakan untuk mencari *client notification token* yang diminta. Penggunaan

usecase ini terdapat pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003).

```
1.  <?php
2.
3.  namespace
    MyIts\Secman\UseCases\FindClientNotificationToken;
4.
5.  use Exception;
6.  use MyIts\Secman\Domain\Contracts\Repositories\
    ClientNotificationTokenRepositoryInterface;
7.
8.  class FindClientNotificationTokenUseCase
9.  {
10.     protected $repository;
11.     public function __construct(ClientNotificationTokenRep
    ositoryInterface $repository)
12.     {
13.         $this->repository = $repository;
14.     }
15.     public function handle(FindCLientNotificationTokenRequ
    est $request)
16.     {
17.         $clientNotificationToken = new
    CLientNotificationToken(
                $request->getClientId(),
                $request->getToken()
            );
18.         $clientNotificationTokenData = $this->repository-
    >find($clientNotificationToken);
19.         return new FindCLientNotificationToken
    ($clientNotificationTokenData);
20.     }
21. }
```

Kode Sumber 4.7 Usecase FindClientNotificationTokenUseCase.php

4.2.2.5. UseCase FindClient

```
1.  <?php
2.
3.  namespace MyIts\Secman\UseCases\FindClient;
```

```
4.
5.  use Exception;
6.  use
    MyIts\Secman\Domain\Contracts\Repositories\ClientsReposito
    ryInterface;
7.
8.  class FindClient
9.  {
10.     protected $repository;
11.     public function __construct(ClientRepositoryInterface
    $repository)
12.     {
13.         $this->repository = $repository;
14.     }
15.     public function handle(FindCLientRequest $request)
16.     {
17.         $client =  $this->repository->findClient($request-
    >getClientId());
18.         $response = new FindClientResponse();
19.         $response->setBcAuthAlg($client->getBcAuthAlg());
20.         $response->setBcTokenDeliveryMode($client-
    >getBcTokenDeliveryMode());
21.         $response->setBcClientNotificationEndp($client-
    >getBcClientNotificationEndp());
22.         $response->setBcUserCodeParameter($client-
    >getBcUserCodeParameter());
23.         $response->setTokenEndpAuthMethod($client-
    >getTokenEndpAuthMethod());
24.         $response->setTokenEndpAuthigningAlg($client-
    >getTokenEndpAuthigningAlg());
25.         Return $response;
26.     }
27. }
```

Kode Sumber 4.8 Usecase FindClientUseCase.php

4.2.2.6. UseCase EditClient

```
1.  <?php
2.
3.  namespace MyIts\Secman\UseCases\EditClient;
4.
5.  use Exception;
```

```
6.  use
    MyIts\Secman\Domain\Contracts\Repositories\ClientsReposito
    ryInterface;
7.  class EditClient
8.  {
9.      protected $repository;
10.     public function __construct(ClientRepositoryInterface
    $repository)
11.     {
12.         $this->repository = $repository;
13.     }
14.     public function handle(EditCLientRequest $request)
15.     {
16.         $bcAuthAlg = $request->getBcAuthAlg();
17.         $bcTokenDeliveryMode = $request-
    >getBcTokenDeliveryMode();
18.         $bcClientNotificationEndp = $request-
    >getBcClientNotificationEndp();
19.         $bcUserCodeParameter = $request-
    >getBcUserCodeParameter();
20.         $tokenEndpAuthMethod = $request-
    >getTokenEndpAuthMethod();
21.         $tokenEndpAuthigningAlg = $request-
    >getTokenEndpAuthigningAlg();
22.         try {
23.             $response = $this->repository->editClient(
                    $clientId,
                    $bcAuthAlg,
                    $bcTokenDeliveryMode,
                    $bcClientNotificationEndp,
                    $bcUserCodeParameter,
                    $tokenEndpAuthMethod,
                    $tokenEndpAuthigningAlg);
24.             return new EditClientResponse("update
    success");
25.         } catch (\Exception $exception) {
26.             return new EditClientResponse(
                    $exception->getMessage(), true);
27.     }
28. }
```

Kode Sumber 4.9 Usecase EditClientUseCase.php

4.2.3. Implementasi Kelas *Repository Interface*

Implementasi *repository interface* dalam aplikasi berada pada package atau folder repositories dimana terdapat kelas-kelas *repository interface*.

4.2.3.1. Repository Interface Ciba Access Token.

```
1.  <?php
2.
3.  use MyIts\Secman\Domain\Models\CibaAccessToken;
4.
5.  namespace MyIts\Secman\Domain\Contracts\Repositories;
6.
7.  interface CibaAccessTokenRepositoryInterface
8.  {
9.      public function saveAccessToken(CibaAccessToken $cibaA
    ccessToken);
10.     public function getAccessToken($authReqId);
11. }
```

Kode Sumber 4.10 Repository Interface CibaAccessTokenRepositoryInterface.php

4.2.3.2. Repository Interface Client Notification Token

```
1.  <?php
2.
3.  use MyIts\Secman\Domain\Models\ClientNotificationToken;
4.
5.  namespace MyIts\Secman\Domain\Contracts\Repositories;
6.
7.  interface ClientNotificationTokenRepositoryInterface
8.  {
9.      public function create(ClientNotificationToken
    $clientNotificationToken);
10.     public function find(ClientNotificationToken
    $clientNotificationToken);
11. }
```

Kode Sumber 4.11 Repository Interface ClientNotificationTokenRepositoryInterface.php

4.2.3.3. Repository Interface Client

```php
<?php

use MyIts\Secman\Domain\Models\Client;

namespace MyIts\Secman\Domain\Contracts\Repositories;

interface ClientNotificationTokenRepositoryInterface
{
    public function findCLient($clientIdentifier);
    public function editCLient(Client $client);
}
```

Kode Sumber 4.12 Repository Interface ClientRepositoryInterface.php

4.2.4. 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 sub bab ini, akan dijelaskan fungsi-fungsi umum pada kelas *repository*. Kelas *repository* akan mengimplementasi kelas *repository interface* pada bagian 4.2.3

4.2.4.1. Repository Ciba Access Token

Kelas ini berfungsi untuk menambah *ciba access token* dan mendapatkan *ciba access token*. Kelas ini mengimplemen CibaAccessTokenRepositoryInterface.

```
public function saveAccessToken(CibaAccessToken)
{

    resultValue = insert(CibaAccessToken)

    IF resultValue is true THEN
        RETURN true
    ELSE
        RETURN false
}

```

```
12. public function getAccessToken(id)
13. {
14.     result = get(id)
15.
16.     IF result is not empty THEN
17.         RETURN result
18.     ELSE
19.         RETURN false
20. }
```

Kode Sumber 4.13 Pseudocode Kelas Repository CibaAccessToken

4.2.4.2. Repository Client Notification Token

Kelas ini berfungsi untuk menambah *client notification token* dan mendapatkan *client notification token*. Kelas ini mengimplemen ClientNotificationTokenRepositoryInterface.

```
1.  public function create(ClientNotificationToken)
2.  {
3.
4.      resultValue = insert(ClientNotificationToken)
5.
6.      IF resultValue is true THEN
7.          RETURN true
8.      ELSE
9.          RETURN false
10. }
11.
12. public function find(ClientNotificationToken)
13. {
14.     result = get(ClientNotificationToken.id)
15.
16.     IF result is not empty THEN
17.         RETURN result
18.     ELSE
19.         RETURN false
20. }
```

Kode Sumber 4.14 Pseudocode Kelas Repository ClientNotificationToken

4.2.4.3. Repository Client

Kelas ini berfungsi untuk mengubah *client* dan mendapatkan *client*. Kelas ini mengimplemen ClientRepositoryInterface.

```
1.  public function editClient(Client)
2.  {
3.
4.      resultValue = edit(Client)
5.
6.      IF resultValue is true THEN
7.          RETURN true
8.      ELSE
9.          RETURN false
10. }
11.
12. public function findCLient(id)
13. {
14.     result = get(id)
15.
16.     IF result is not empty THEN
17.         RETURN result
18.     ELSE
19.         RETURN false
20. }
```

Kode Sumber 4.15 Pseudocode Kelas Repository Client

4.2.5. Implementasi Kelas *Controller*

Kelas *controller* berisikan fungsi-fungsi yang digunakan untuk menangkap *request* dan mengirimkan *response* baik berupa string JSON kepada *browser* maupun berupa tampilan antarmuka dengan memproses terlebih dahulu di dalam kelas *controller*. Kelas-kelas *controller* berada pada *package* atau folder *Controllers*.

4.2.5.1. Controller CIBA

Kelas Controller CIBA digunakan untuk menangkap dan menangani *request* yang digunakan untuk mendaftarkan dan melakukan *ciba request* dengan berbagai *mode.*

```php
<?php

namespace MyIts\Secman\Controllees\Web;

class CibaController
{
    public function clientNotificationAction()
    public function authenticationRequestActionCiba()
    public function signedCibaRequestAction()
    public function exposeClientPublicKeyForOpAction()
    public function checkAccessTokenAction ()
    public function requestTokenAction()
}
```

Kode Sumber 4.16 Pseudocode Fungsi Kelas Controller CibaController

4.2.5.2. Controller Client

Kelas Controller Client digunakan untuk menangkap dan menangani *request* yang digunakan untuk mengubah dan mencari *client*.

```php
<?php

namespace MyIts\Secman\Controllees\Web;

class ClientController
{
    public function findClientAction()
    public function editClientAction()
}
```

Kode Sumber 4.17 Pseudocode Fungsi Kelas Controller CibaController

4.2.6. Implementasi Pengiriman Authentication Request

Proses mengirimkan *authentication request* terjadi pada controller dan pada library *OpenID Connect Client*. Pada sisi klien saat mengirimkan *Authentication Request* tidak disertakan *user_code* karena MyITS SSO tidak mensupport penggunaan *User Code*.

```php
<?php

Method POST
Param : 'user_id'
public function authenticationRequestActionCiba()
{
    $clientId = $this->config->openid->clientId;
    $userId = $this->request->getPost('user_id');
    $client_notification_token = $this->getUuid();

    // menyimpan client notification token yang telah dibuat dan untuk mengambil token. Ref: Kode Sumber 4.6
    $request = new AddClientNotificationTokenRequest(
        $clientId,
        $this->getAuthUser()->getId(),
        $userId,
        $client_notification_token);

    $this->addClientNotificationTokenUc->handle($request);


    // membaca config, apakah request dikirim dalam bentuk signed atau tidak
    $signedRequest = $this->config->openid->signed_request;
    $privateKey = $this->config->openid->ciba_private_key;
    $kid = $this->config->openid->ciba_kid;
    $alg = $this->config->openid->ciba_key_alg;

    try {
        if ($signedRequest) {
            // memanggil method signedAuthenticationRequestCiba pada OpenID Connect Client. Ref: Kode Sumber 4.20
            $response = $this->oidcClient->signedAuthenticationRequestCiba($client_notification_token, $userId, $privateKey, $kid, $alg, 60);
        } else {
            // memanggil method authenticationRequestCiba pada OpenID Connect Client. Ref: Kode Sumber 4.19
```

```
28.             $response = (array)$this->oidcClient-
    >authenticationRequestCiba($client_notification_token,
    $userId, 60);
29.         }
30.     } catch (OpenIDConnectClientException $e) {
31.         return $this->send($e->getMessage(), 400);
32.     }
33.
34.     $bindingMessage = $this->oidcClient
    >getBindingMessage();
35.     $now = time();
36.     $expiresInSecond = ($response['expires_in'] - $now);
37.
38.     // mengeluarkan response kepada browser
39.     return $this->send(array_merge(
            $response, [
                'binding_message' => $bindingMessage,
                'expires_in_second' => $expiresInSecond
            ]));
40. }
```

Kode Sumber 4.18 Fungsi Melakukan Authentication Request pada CibaController

```
1.  <?php
2.
3.  public function authenticationRequestCiba($client_notifica
    tion_token, $user_id, $requested_expiry = 5)
4.  {
5.      $ciba_auth_endpoint = $this-
    >getProviderConfigValue("backchannel_authentication_endpoi
    nt");
6.      $notification_token = $client_notification_token);
7.      $binding_message = $this->generateBindingMessage();
8.
9.      $headers = [];
10.     $auth_methods_supported = true;
11.
12.     $auth_params = array(
13.         'scope' => 'openid',
14.         'client_notification_token' => $
    notification_token,
15.         'login_hint' => $user_id,
```

```
16.         'binding_message' => $binding_message,
17.         'requested_expiry' => $requested_expiry
18.     );
19.
20.     // menggunakan basic auth sebagai header
21.     if ($token_endpoint_auth_methods_supported)) {
22.         $headers = ['Authorization: Basic ' .
    base64_encode($this->clientID . ':' . $this-
    >clientSecret)];
23.     }
24.
25.     $auth_params = http_build_query($auth_params, null,
    '&');
26.
27.     // menghubungi server untuk memanggil authentication
    request endpoint
28.     $response = $this->fetchURL($ciba_auth_endpoint,
    $auth_params, $headers);
29.     $this->setAuthReqId($response);
30.
31.     if ($this->getBcTokenDeliveryMode() === 'poll') {
32.         $this->setAllowedToPoll(true);
33.     }
34.
35.     return json_decode($response);
36. }
```

Kode Sumber 4.19 Fungsi Melakukan Authentication Request pada OpenID Connect Client

```
1.  <?php
2.
3.  public function signedAuthenticationRequestCiba($client_no
    tification_token, $user_id, $private_key, $kid, $alg =
    'RS256', $requested_expiry = 5)
4.  {
5.      $this->timeOut = 500;
6.      $now = time();
7.
8.      $ciba_auth_endpoint = $this-
    >getProviderConfigValue("backchannel_authentication_endpoi
    nt");
9.      $notification_token = $client_notification_token);
```

```
10.     $binding_message = $this->generateBindingMessage();
11.
12.     $headers = [];
13.     $auth_methods_supported = true;
14.
15.     $requiredClaims = [
16.         'aud' => $this->getProviderURL(),
17.         'iss' => $this->getClientID(),
18.         'exp' => $now + $requested_expiry,
19.         'iat' => $now ,
20.         'nbf' => $now,
21.         'jti' => $this->generateRandString()
22.     ];
23.
24.     $auth_request_parameters = [
25.         'scope' => 'openid',
26.         'client_notification_token' =>
    $notification_token,
27.         'login_hint' => $user_id,
28.         'binding_message' => $binding_message,
29.         'requested_expiry' => $requested_expiry,
30.     ];
31.
32.     $jwtClaims = array_merge($auth_request_parameters,
    $requiredClaims);
33.
34.     // request di encode menjadi JWT
35.     $jwt = $this->encode($jwtClaims, $private_key, $kid,
    $alg);
36.
37.     $auth_params = [ 'request' => $jwt ];
38.
39.     // menggunakan basic auth sebagai header
40.     if ($token_endpoint_auth_methods_supported)) {
41.         $headers = ['Authorization: Basic ' .
    base64_encode($this->clientID . ':' . $this-
    >clientSecret)];
42.     }
43.
44.     $auth_params = http_build_query($auth_params, null,
    '&');
45.
```

```
46.     // menghubungi server untuk memanggil authentication
    request endpoint
47.     $response = $this->fetchURL($ciba_auth_endpoint,
    $auth_params, $headers);
48.     $this->setAuthReqId($response);
49.
50.     if ($this->getBcTokenDeliveryMode() === 'poll') {
51.         $this->setAllowedToPoll(true);
52.     }
53.
54.     return json_decode($response);
55. }
```

Kode Sumber 4.20 Fungsi Melakukan Signed Authentication Request pada OpenID Connect Client

4.3. Implementasi User Consent Authentication Device

Implementasi *Authentication Device* dilakukan pada *project* yang berbeda, pengimplementasian menggunakan bahasa pemrograman Java.

4.3.1. Implementasi Listener Firebase Cloud Messaging

Sistem notifikasi yang diterapkan dalam MyITS Authenticator menggunakan *Firebase Cloud Messaging*. *Firebase Cloud Messaging* dipilih karena memiliki layanan pengiriman notifikasi secara gratis, dan cara pengimplementasian lebih mudah karena memiliki dokumentasi yang jelas. Pemberian notifikasi pada *Firebase Cloud Messaging* berdasarkan *topics*, dengan adanya *topics*, MyITS SSO dapat mengirimkan notifikasi kepada *topics* tersebut dan hanya pemilik *topics* tersebut yang dapat menerima pesan tersebut. Konvensi *topics* yang dimiliki oleh MyITS SSO adalah “ciba_{USER_ID}”.

```
1.  Public class AppFirebaseMessagingService  extends
    FirebaseMessagingService {
2.     private static final String TAG = "FirebaseMSG";
3.     private static final String CHANNEL = "ciba:channel";
4.     private static final String AUTH_REQ_ID_KEY =
    "cibaAuthReqId";
5.     private static final String PREF_NAME = "cibaSession";
```

```
6.
7.      @override
8.      public void onNewToken(String token) {
9.         FirebaseInstanceId.getInstance().
           getInstanceId().
           addOnCompleteListener(new
           OnCompleteListener<InstanceIdResult>() {
10.           @override
11.           public void onComplete(
              @NonNull Task<InstanceIdResult> task) {
12.              if (!task.isSuccessful()) {
13.                 return;
14.              }
15.           }
16.        });
17.     }
18.
19.     @override
20.     public void onMessageReceived(
        RemoteMessage remoteMessage) {
21.        super.onMessageReceived(remoteMessage);
22.
23.        // mengecek apakah tersedia data dari notifikasi
    Firebase
24.        if (remoteMessage.getData().size() > 0) {
25.           Map<String, String> data =
                 remoteMessage.getData();
26.           String authReqId = data.get("auth_req_id");
27.           String title = data.get("title");
28.           String message = data.get("message");
29.           String bindingMessage =
                 data.get("binding_message");
30.
31.           saveAuthReqId(authReqId);
32.
33.           NotificationManager notificationManager =
              (NotificationManager)getApplicationContext()
              .getSystemService(NOTIFICATION_SERVICE);
34.
35.           Final int notificationId = new
              Random().nextInt(60000);
36.
```

```
37.          Intent intent = new Intent(this,
    RequestConsentActivity.class);
38.          intent.addCategory(Intent.CATEGORY_LAUNCHER);
39.          intent.setAction(Intent.ACTION_MAIN);
40.          intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
41.          intent.putExtra("title", title);
42.          intent.putExtra("message", message);
43.          intent.putExtra("binding_message",
    bindingMessage);
44.
45.          // membuat Instance push notifikasi
46.          NotificationCompat.Builder builder = new
             NotificationCompat.Builder(
47.             getApplicationContext(), CHANNEL)
48.             .setSmallIcon(R.mipmap.ic_launcher)
49.             .setContentTitle(title)
50.             .setContentText(message)
51.             .setAutoCancel(true)
52.             .setSound(RingtoneManager.getDefaultUri(
                   RingtoneManager.TYPE_NOTIFICATION));
53.
54.          builder.setPriority(
                NotificationCompat.PRIORITY_HIGH);
55.             notificationManager.notify(notificationId,
                builder.build());
56.
57.          // memulai activity baru, yaitu pop-up user
    consent
58.          startActivity(intent);
59.       }
60.    }
61.
62.     // menyimpan authreqid pada SharedPreferences
63.     private void saveAuthReqId(String authReqId) {
64.         SharedPreferences sharedPref =
            getSharedPreferences(CIBA_PREF_NAME,
    MODE_PRIVATE);
65.         SharedPreferences.Editor editor =
    sharedPref.edit();
66.         editor.putString(CIBA_AUTH_REQ_ID_KEY, authReqId);
67.         editor.commit();
68.     }
69. }
```

Kode Sumber 4.21 Class Implementasi Firebase Cloud Message Service

4.3.2. Implementasi Pemberian User Consent

Pemberian *consent* dilakukan pada *User Consent Activity*. *Activity* tersebut akan memanggil *endpoint* yang akan mengirimkan *consent* menuju MyITS SSO.

```
1.      private void setupDialog() {
2.          button buttonNo = findViewById(R.id.button_no);
3.          button buttonYes = findViewById(R.id.button_yes);
4.
5.          mApiInterface = ApiClient.getClient().
            create(ApiClientInterface.class);
6.
7.          SharedPreferences sharedPref =
            getSharedPreferences(CIBA_PREF_NAME, MODE_PRIVATE);
8.
9.      String authReqId = sharedPref.
        getString(CIBA_AUTH_REQ_ID_KEY, null);
10.
11.     buttonNo.setOnClickListener(v -> {
12.         Call<UserConsentResponse>
            postUserConsentCall = mApiInterface.
            postUserConsent(authReqId, "deny");
13.         postUserConsentCall.enqueue(new Callback
            <UserConsentResponse>() {
14.             @Override
15.             public void onResponse(
                Call<UserConsentResponse> call,
                Response<UserConsentResponse> response) {
16.                 finish();
17.             }
18.             @Override
19.             public voidonFailure(
                Call<UserConsentResponse> call, Throwable t) {
20.                 finish();
21.             }
22.         });
23.     });
24.
25.     buttonYes.setOnClickListener(v -> {
```

```
26.        Call<UserConsentResponse>
           postUserConsentCall = mApiInterface.
           postUserConsent(authReqId, "accept");
27.        postUserConsentCall.enqueue(new Callback
           <UserConsentResponse>() {
28.        @Override
29.        public void onResponse(
           Call<UserConsentResponse> call,
           Response<UserConsentResponse> response) {
30.           finish();
31.        }
32.        @Override
33.        public void onFailure(
           Call<UserConsentResponse> call, Thorwable t) {
34.              finish();
35.        }
36. });
```

Kode Sumber 4.22 Class Implementasi User Request Activity

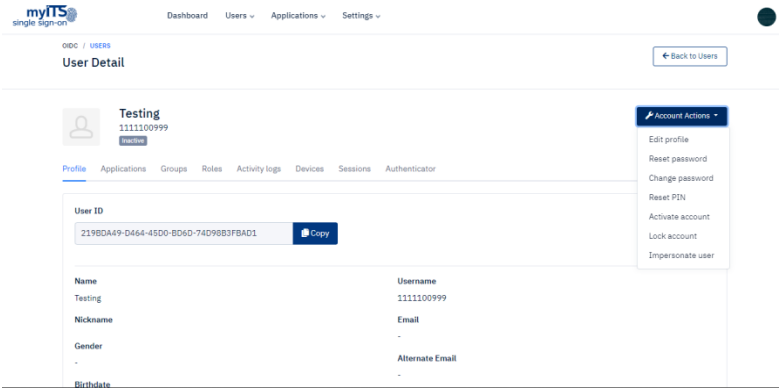
4.4. Implementasi Antarmuka Pengguna

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

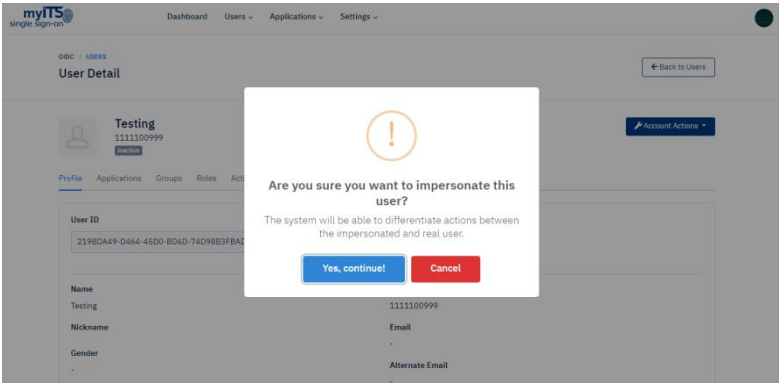
4.4.1. Halaman User Detail myITS Security Management

Halaman *user detail* digunakan pada kasus penggunaan Melakukan *CIBA Request* mode *Poll* (UC-001), Melakukan *CIBA Request* mode *Ping* (UC-002), Melakukan *CIBA Request* mode *Push* (UC-003). Halaman ini berisi detail user yang akan diimpersonate. Implementasi terkait antarmuka *user detail* dapat dilihat pada Gambar 4.1, Gambar 4.2, Gambar 4.3, Gambar 4.4

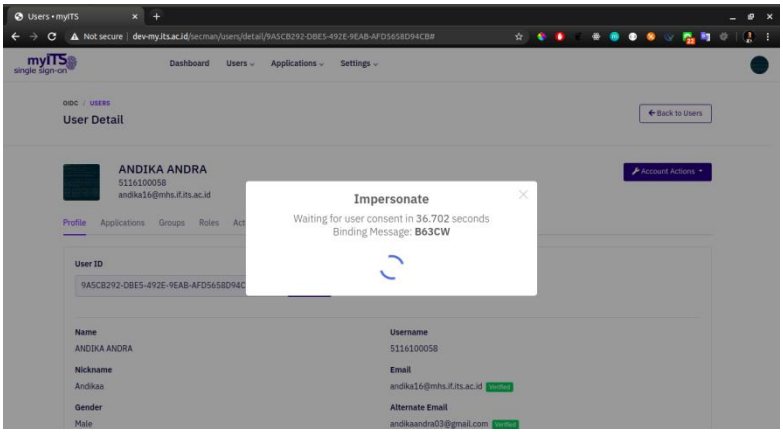
dan Gambar 4.5.



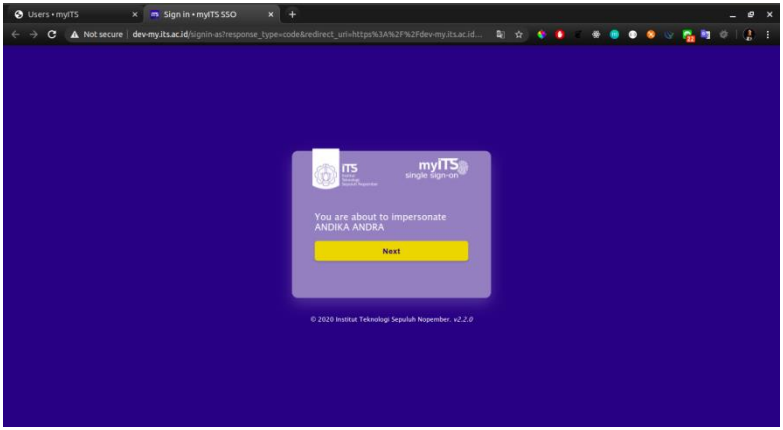
Gambar 4.1 Implementasi Halaman User Detail myITS Security Management



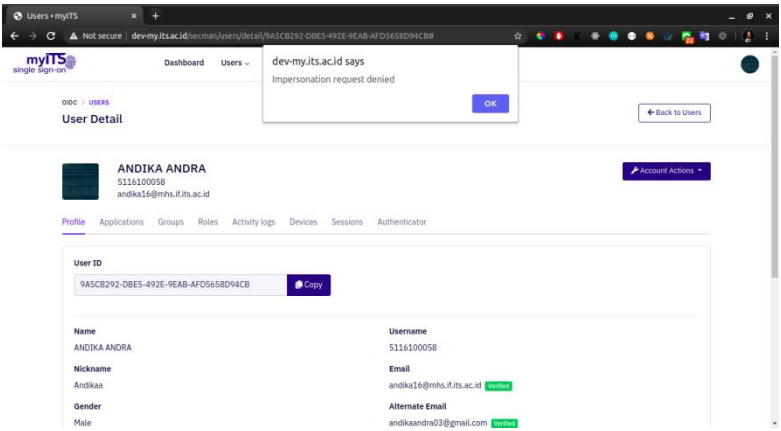
Gambar 4.2 Implementasi Halaman User Detail myITS Security Management Verify Prompt



Gambar 4.3 Implementasi Halaman User Detail myITS Security Management PopUp Ciba Status



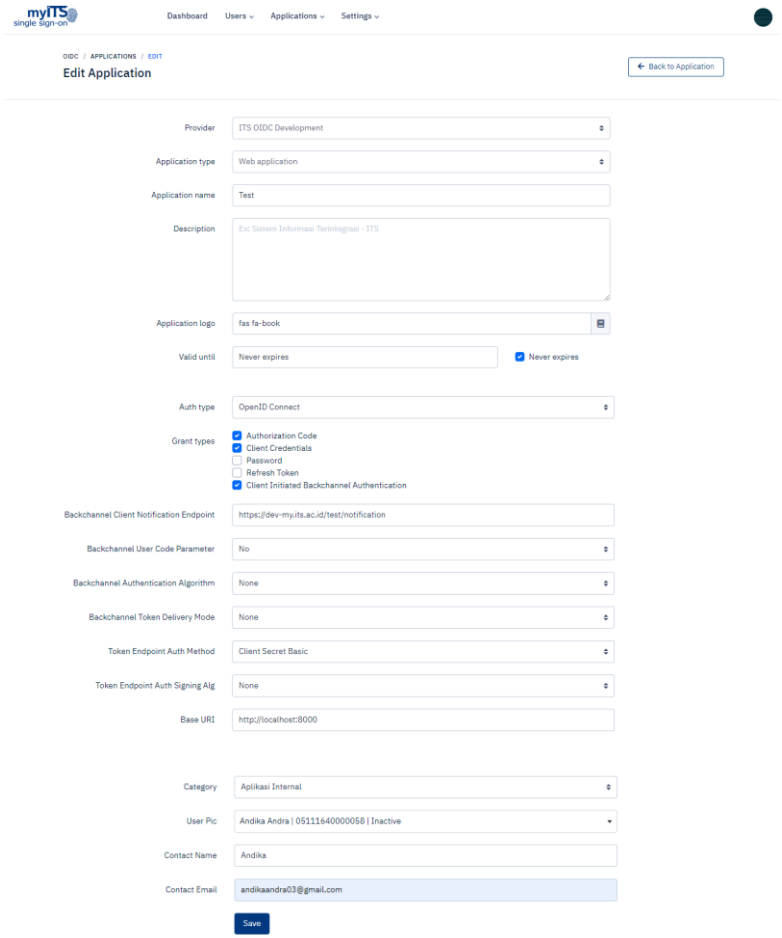
Gambar 4.4 Implementasi Halaman User Detail myITS Security Management Ciba Accepted



Gambar 4.5 Implementasi Halaman User Detail myITS Security Management Ciba Denied

4.4.2. Halaman Edit Client myITS Security Management

Halaman *edit client* akan digunakan pada kasus penggunaan Mengubah Aplikasi Klien yang Mendukung Alur CIBA (UC-004). Pada halaman ini menampilkan data klien yang akan diubah. Implementasi terkait antarmuka *edit client* dapat dilihat pada Gambar 4.6.



Gambar 4.6 Implementasi Halaman Edit Client myITS Security

4.4.3. MyITS Authenticator

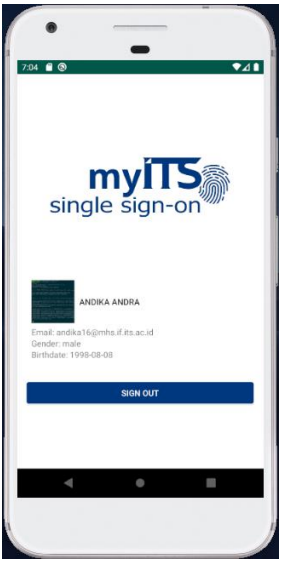
myITS Authenticator digunakan oleh user yang diminta *consent* nya melalui MyITS Authenticator. Implementasi terkait antarmuka myITS Authenticator dapat dilihat pada Gambar 4.7, Gambar 4.8, Gambar 4.9 dan Gambar 4.10.

Gambar 4.7 Implementasi myITS Authenticator Menu Utama

Pada menu utama myITS Authenticator, terdapat logo myITS SSO dan tombol untuk login, dan jika ditekan akan diarahkan untuk membuka *browsable concent* yang akan membuka laman myITS SSO.

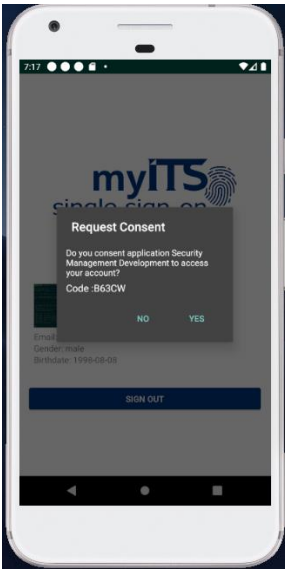
Gambar 4.8 Implementasi myITS Authenticator Halaman Login

Menu akan muncul ketika telah menekan tombol login, laman yang ditampilkan adalah laman dari myITS SSO.



Gambar 4.9 Implementasi myITS Authenticator Halaman User

Setelah berhasil login maka *browsable content* yang berisi laman myITS SSO akan hilan dan akan muncul menu utama setelah login, terdapat tombol logout dan berisi informasi tentang pengguna yang telah login.



Gambar 4.10 Implementasi myITS Authenticator Popup Consent

Setalah MyITS SSO mengirimkan notifikasi maka akan muncul *popup* tentang permintaan consent dari *user requester*.

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Bab ini membahas uji coba dan evaluasi terhadap perangkat lunak yang telah dikembangkan dari implementasi alur *Client Initiated Backchannel Authentication*.

5.1. Lingkungan Uji Coba

Lingkungan uji coba adalah kombinasi antara perangkat keras dan perangkat lunak yang digunakan untuk melakukan uji coba. Pengujian dilakukan dengan menggunakan tiga buah lingkungan pengujian yaitu lingkungan ujicoba *platform Windows*, lingkungan uji coba *platform macOS*, lingkungan uji coba *platform Android*. Adapun rincian dari masing-masing lingkungan pengujian tersebut secara berturut-turut ditunjukkan pada Tabel 5.1, Tabel 5.2 dan Tabel 5.3.

Tabel 5.1 Lingkungan Uji Coba *Platform Windows*

Spesifikasi	Deskripsi
CPU	Intel® Core™ i5-6200U CPU @ 2.30GHz
RAM	8.00 GB
Sistem Operasi	Windows 10 Enterprise 64-bit
Browser	Chrome versi 80, Firefox versi 74, Edge versi 17

Tabel 5.2 Lingkungan Uji Coba *Platform macOS*

Spesifikasi	Deskripsi
CPU	Intel® Core™ i7 3.30GHz
RAM	8.00 GB
Sistem Operasi	Mojave 14.1
Browser	Chrome versi 66 dan 67, Firefox versi 59 dan 60, Safari 12.1

Tabel 5.3 Lingkungan Uji Coba *Platform Android*

Spesifikasi	Deskripsi
Smartphone	Xiaomi Mi4
CPU	Quad-core 2.5GHz
RAM	2.00 GB
Sistem Operasi	Android 6.0.1 (Marshmallow)
Browser	Chrome for android versi 80, Firefox for android versi 68

5.2. Skenario Pengujian

Pada bagian ini akan dibahas mengenai proses uji coba yang digunakan. Pengujian dilakukan dengan metode *black box* untuk menguji masing-masing fungsionalitas yang sudah dirancang pada sistem. Metode *black box* merupakan metode pengujian perangkat lunak yang memeriksa fungsionalitas dari suatu perangkat lunak tanpa memandang struktur internalnya.

Pada proses uji coba, pengujian dilakukan dengan menjalankan serangkaian perintah terhadap sistem yang dilakukan oleh 5 mahasiswa dan selanjutnya akan disebut sebagai kasus pengujian. Kasus pengujian ini berkorelasi dengan kasus-kasus penggunaan dan kebutuhan fungsional yang sebelumnya sudah dirancang dan dijelaskan pada Bab III.

5.2.1. Kasus Pengujian Melakukan CIBA Request Mode Poll

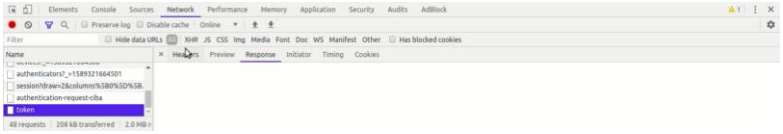
Pada kasus uji ini dilakukan untuk menguji apakah pengguna dapat Melakukan *CIBA Request* mode *Poll*.

Tabel 5.4 Kasus Pengujian Melakukan *CIBA Request* mode *Poll*

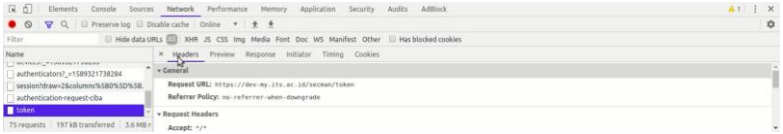
ID	UJ-001
Nama Skenario Pengujian	Fungsionalitas Melakukan *CIBA Request* mode *Poll*

ID	UJ-001
Nama	Pengujian Melakukan *CIBA Request* mode *Poll*
Tujuan Pengujian	Menguji apakah sistem sudah mampu untuk mendeteksi serta melakukan *error handling* saat pengguna melakukan *CIBA Request* mode *Poll*
Skenario 1	*Membatalkan CIBA Request*
ID Skenario	UJ-001A
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”No”.
Hasil yang diharapkan	*Browser* akan menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil yang diperoleh	*Browser* menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil Pengujian	Berhasil
Skenario 2	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator belum melakukan consent*
ID Skenario	UJ-001B
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. Pemegang MyITS Authenticator Mendapatkan Notifikasi
Hasil yang diharapkan	*Browser* akan tetap melakukan *request* dan menunggu response dari MyITS SSO
Hasil yang diperoleh	*Browser* tetap melakukan *request* dan menunggu response dari MyITS SSO sampai mendapatkan response.
Hasil Pengujian	Berhasil
Hasil	Gambar **5.1**, Gambar **5.2**

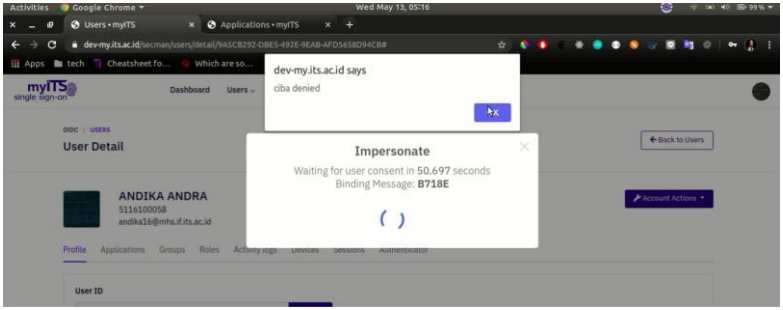
ID	UJ-001
Skenario 3	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator tidak menyetujui consent*
ID Skenario	UJ-001C
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. User pemegang MyITS Authenticator melakukan *consent* “NO”
Hasil yang diharapkan	*Browser* akan menampilkan pop-up bahwa CIBA telah ditolak
Hasil yang diperoleh	*Browser* menampilkan pop-up bahwa CIBA telah ditolak
Hasil Pengujian	Berhasil
Hasil	Gambar **5.3**
Skenario 4	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator menyetujui consent*
ID Skenario	UJ-001D
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. User pemegang MyITS Authenticator melakukan *consent* “YES”
Hasil yang diharapkan	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil yang diperoleh	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil Pengujian	Berhasil
Hasil	Gambar **5.4**



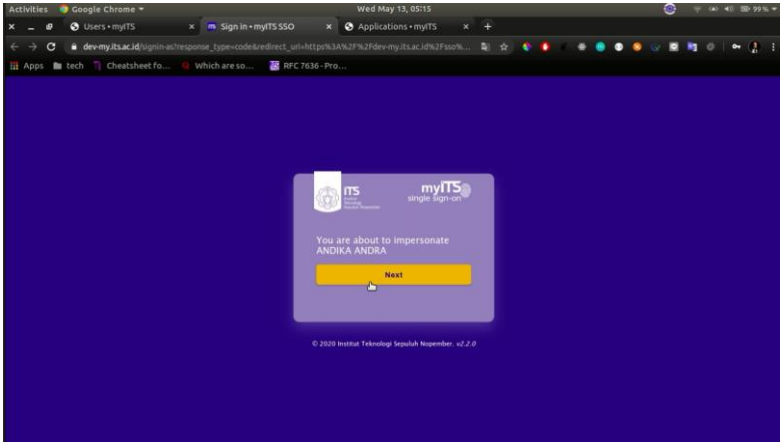
Gambar 5.1 Proses Request Token Mode Poll (1)



Gambar 5.2 Proses Request Token Mode Poll (1)



Gambar 5.3 Proses CIBA Request Mode Poll Ditolak



Gambar 5.4 Proses CIBA Request Mode Poll Diterima

5.2.2. Kasus Pengujian Melakukan CIBA Request Mode Ping

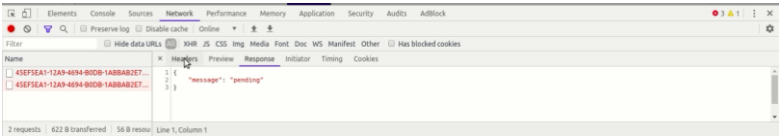
Pada kasus uji ini dilakukan untuk menguji apakah pengguna dapat Melakukan *CIBA Request* mode *Ping*.

Tabel 5.5 Kasus Pengujian Melakukan *CIBA Request* mode *Ping*

ID	UJ-002
Nama Skenario Pengujian	Fungsionalitas Melakukan *CIBA Request* mode *Ping*
Nama	Pengujian Melakukan *CIBA Request* mode *Ping*
Tujuan Pengujian	Menguji apakah sistem sudah mampu untuk mendeteksi serta melakukan *error handling* saat pengguna melakukan *CIBA Request* mode *Ping*
Skenario 1	*Membatalkan CIBA Request*
ID Skenario	UJ-002A
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”No”.
Hasil yang diharapkan	*Browser* akan menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil yang diperoleh	*Browser* menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil Pengujian	Berhasil
Skenario 2	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator belum melakukan consent*
ID Skenario	UJ-002B
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”.

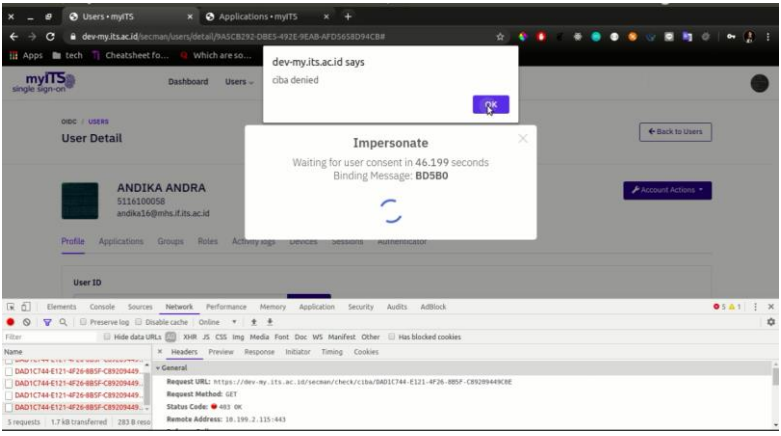
ID	UJ-002
	3. Pemegang MyITS Authenticator Mendapatkan Notifikasi
Hasil yang diharapkan	*Browser* akan tetap melakukan *request* dan menunggu *response* dari MyITS SSO
Hasil yang diperoleh	*Browser* tetap melakukan *request* dan menunggu response dari MyITS SSO sampai mendapatkan *response*.
Hasil Pengujian	Berhasil
Hasil	Gambar **5.5**, Gambar **5.6**
Skenario 3	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator tidak menyetujui consent*
ID Skenario	UJ-002C
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. User pemegang MyITS Authenticator melakukan *consent* “NO”
Hasil yang diharapkan	*Browser* akan menampilkan pop-up bahwa CIBA telah ditolak
Hasil yang diperoleh	*Browser* menampilkan pop-up bahwa CIBA telah ditolak
Hasil Pengujian	Berhasil
Hasil	Gambar **5.7**
Skenario 4	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator menyetujui consent*
ID Skenario	UJ-002D
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”.

ID	UJ-002
	3. User pemegang MyITS Authenticator melakukan *consent* “YES”
Hasil yang diharapkan	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil yang diperoleh	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil Pengujian	Berhasil
Hasil	Gambar **5.8**

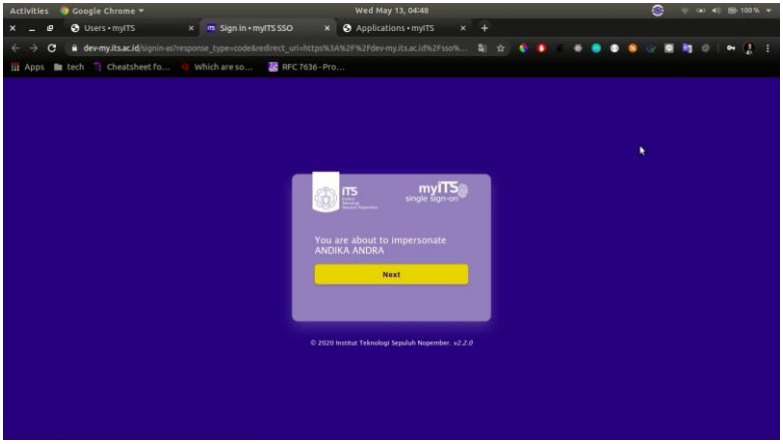


Gambar 5.5 Proses Request Token Mode Ping (1)

Gambar 5.6 Proses Request Token Mode Ping (2)



Gambar 5.7 Proses CIBA Request Mode Ping Ditolak



Gambar 5.8 Proses CIBA Request Mode Ping Diterima

5.2.3. Kasus Pengujian Melakukan CIBA Request Mode Push

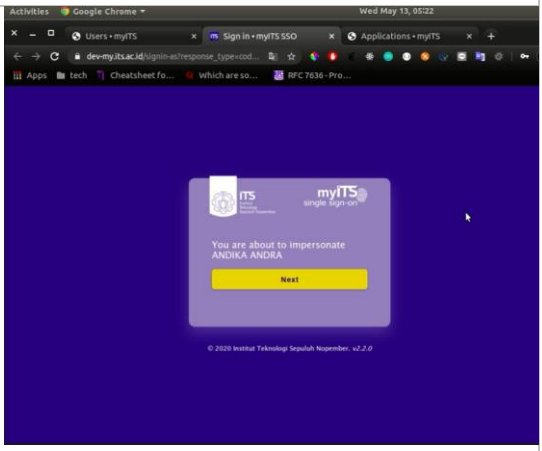
Pada kasus uji ini dilakukan untuk menguji apakah pengguna dapat Melakukan *CIBA Request* mode *Push.*

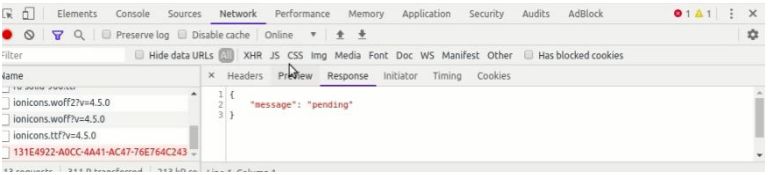
Tabel 5.6 Kasus Pengujian Melakukan *CIBA Request* mode *Push*

ID	UJ-003
Nama Skenario Pengujian	Fungsionalitas Melakukan *CIBA Request* mode *Push*
Nama	Pengujian Melakukan *CIBA Request* mode *Push*
Tujuan Pengujian	Menguji apakah sistem sudah mampu untuk mendeteksi serta melakukan *error handling* saat pengguna melakukan *CIBA Request* mode *Push*
Skenario 1	*Membatalkan CIBA Request*
ID Skenario	UJ-003A
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”No”.

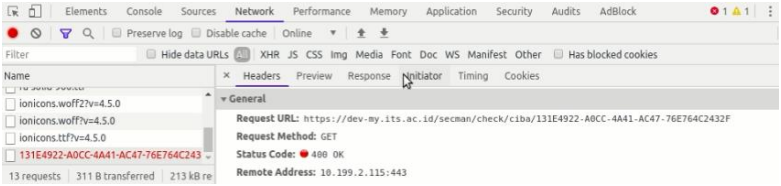
ID	UJ-003
Hasil yang diharapkan	*Browser* akan menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil yang diperoleh	*Browser* menutup prompt dan tidak melakukan proses *CIBA Request*.
Hasil Pengujian	Berhasil
Skenario 2	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator belum melakukan consent*
ID Skenario	UJ-003B
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. Pemegang MyITS Authenticator Mendapatkan Notifikasi
Hasil yang diharapkan	*Browser* akan tetap melakukan *request* dan menunggu *response* dari MyITS SSO
Hasil yang diperoleh	*Browser* tetap melakukan *request* dan menunggu *response* dari MyITS SSO sampai mendapatkan *response*.
Hasil Pengujian	Berhasil
Hasil	Gambar **5.9**, Gambar **5.10**
Skenario 3	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator tidak menyetujui consent*
ID Skenario	UJ-003C
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. User pemegang MyITS Authenticator melakukan *consent* “NO”

ID	UJ-003
Hasil yang diharapkan	*Browser* akan menampilkan pop-up bahwa CIBA telah ditolak
Hasil yang diperoleh	*Browser* menampilkan pop-up bahwa CIBA telah ditolak
Hasil Pengujian	Berhasil
Hasil	Gambar **5.11**
Skenario 4	*Sistem melakukan CIBA Request, user pemegang MyITS Authenticator menyetujui consent*
ID Skenario	UJ-003D
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. User pemegang MyITS Authenticator melakukan *consent* “YES”
Hasil yang diharapkan	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil yang diperoleh	*Browser* akan mengarahkan halaman menuju halaman *sign-as*
Hasil Pengujian	Berhasil
Hasil	

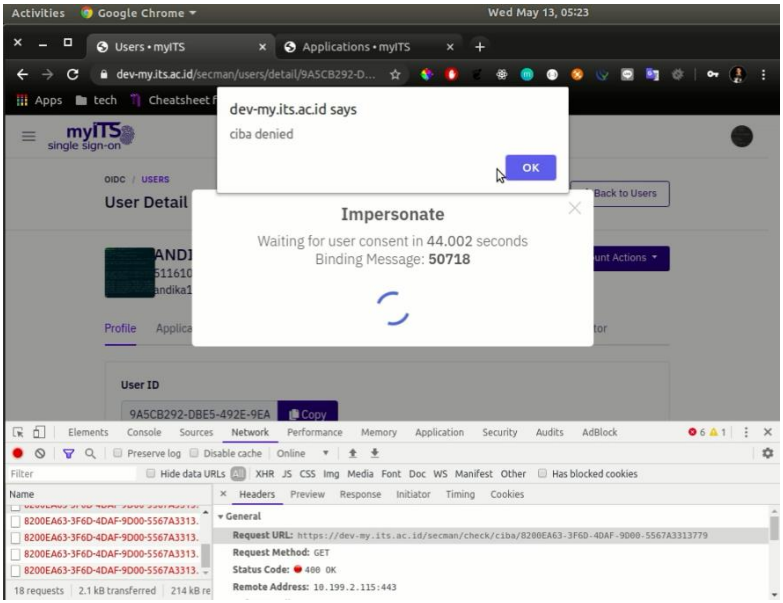
ID	UJ-003
	 Gambar 5.12



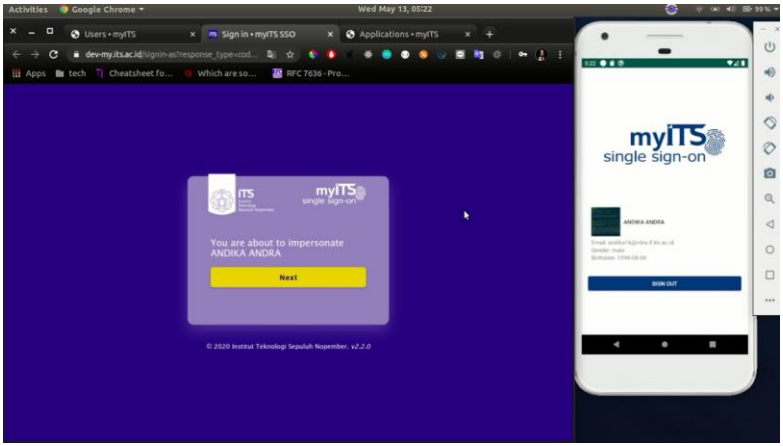
Gambar 5.9 Proses Request Token Mode Push (1)



Gambar 5.10 Proses Request Token Mode Push (2)



Gambar 5.11 Proses CIBA Request Mode Push Ditolak



Gambar 5.12 Proses CIBA Request Mode Push Diterima

5.2.4. Kasus Pengujian Pemegang MyITS Authenticator Tidak Melakukan Consent

Pada kasus uji ini dilakukan untuk menguji jika pemegang MyITS Authenticator tidak melakukan consent sampai waktu yang ditentukan.

Tabel 5.7 Kasus Pengujian Pemegang MyITS Authenticator tidak melakukan *consent*

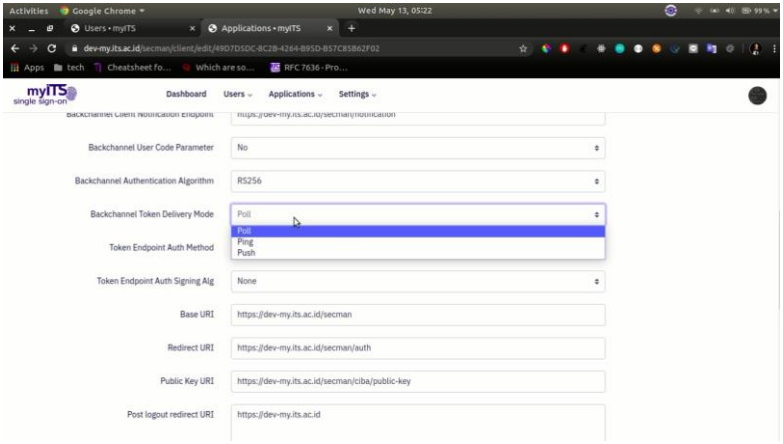
ID	UJ-004
Nama Skenario Pengujian	Fungsionalitas pemegang MyITS Authenticator tidak melakukan consent
Nama	Pengujian Pemegang MyITS Authenticator Tidak Melakukan *Consent*
Tujuan Pengujian	Menguji apakah sistem sudah mampu untuk mendeteksi serta melakukan *error handling* saat pemegang MyITS Authenticator tidak melakukan *consent* samapai waktu yang ditentukan
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih menekan tombol *Impersonate*. 2. Pengguna memilih tombol ”Yes”. 3. Pemegang MyITS Authenticator Mendapatkan Notifikasi
Hasil yang diharapkan	*Browser* akan menutup *prompt* status CIBA *request*, dan tidak melakukan apa-apa
Hasil yang diperoleh	*Browser* menutup *prompt* status CIBA *request*, dan tidak melakukan apa-apa
Hasil Pengujian	Berhasil

5.2.5. Kasus Pengujian Mengubah Aplikasi Klien yang Mendukung Alur CIBA

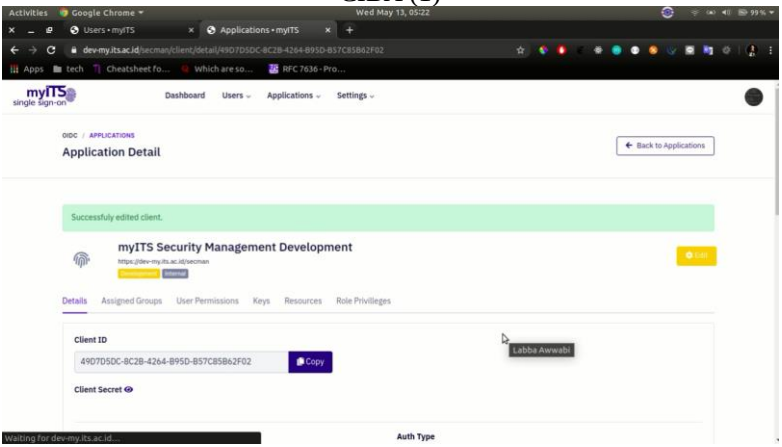
Pada kasus uji ini dilakukan untuk menguji apakah pengguna dapat Mengubah aplikasi klien yang mendukung alur CIBA.

Tabel 5.8 Kasus Pengujian Mengubah Aplikasi Klien Yang Mendukung ALur CIBA

ID	UJ-005
Nama Skenario Pengujian	Fungsionalitas Mengubah Aplikasi Klien Yang Mendukung Alur CIBA
Nama	Pengujian Mengubah Aplikasi Klien Yang Mendukung Alur CIBA
Tujuan Pengujian	Menguji apakah dapat mengubah aplikasi klien yang mendukung alur CIBA
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna memilih klien yang akan diubah. 2. Pengguna menekan tombol *edit*. 3. Pengguna melakukan perubahan pada klien 4. Pengguna menekan tombol *save*
Hasil yang diharapkan	Sistem akan menyimpan perubahan san mengalihkan halaman menuju detail aplikasi klien
Hasil yang diperoleh	Sistem menyimpan perubahan san mengalihkan halaman menuju detail aplikasi klien
Hasil Pengujian	Berhasil



Gambar 5.13 Proses Mengubah Aplikasi Klien Yang Mendukung Alur CIBA (1)



Gambar 5.14 Proses Mengubah Aplikasi Klien Yang Mendukung Alur CIBA (2)

5.2.6. Kasus Pengujian Penerimaan Notifikasi Pada MyITS Authenticator

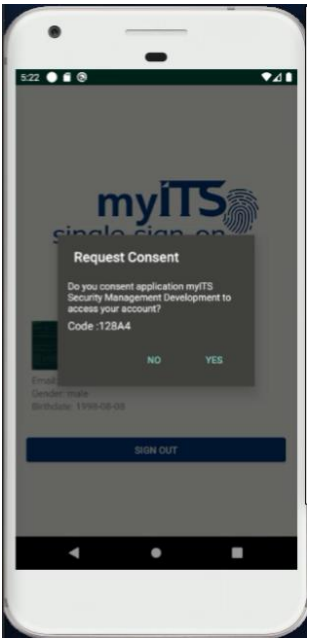
Pada kasus uji ini dilakukan untuk menguji apakah MyITS Authenticator dapat menerima notifikasi

Tabel 5.9 Kasus Pengujian Penerimaan Notifikasi Pada MyITS Authenticator

ID	UJ-006
Nama Skenario Pengujian	Fungsionalitas Penerimaan Notifikasi Pada MyITS Authenticator
Nama	Pengujian Penerimaan Notifikasi Pada MyITS Authenticator
Tujuan Pengujian	Menguji apakah MyITS Authenticator dapat menerima notifikasi dari MyITS SSO
Skenario 1	*Pengguna akan melakukan Impersonate pada Adistya Azhar, akan tetapi Andika Andra login pada MyITS Authenticator*
ID Skenario	UJ-006A
Kondisi Awal	Pengguna sudah terotentikasi

ID	UJ-006
Langkah Pengujian	1. Pengguna melakukan *Impersonate*.
Hasil yang diharapkan	Pada MyITS Authenticator tidak akan muncul notifikasi
Hasil yang diperoleh	MyITS Authenticator tidak muncul notifikasi, karena notifikasi bukan untuk Andika Andra
Hasil Pengujian	Berhasil
Hasil	**Gambar 5.15**
Skenario 2	*Pengguna akan melakukan Impersonate pada Andika Andra, Andika Andra login pada MyITS Authenticator*
ID Skenario	UJ-006B
Kondisi Awal	Pengguna sudah terotentikasi
Langkah Pengujian	1. Pengguna melakukan *Impersonate*.
Hasil yang diharapkan	Pada MyITS Authenticator akan muncul notifikasi permintaan *User Consent*
Hasil yang diperoleh	MyITS Authenticator akan muncul notifikasi permintaan *User Consent*
Hasil Pengujian	Berhasil
Hasil	**Gambar 5.16**

Gambar 5.15 Proses Penerimaan Notifikasi (1)



Gambar 5.16 Proses Penerimaan Notifikasi (2)

5.3. Pengujian Non Fungsional

Pada bagian ini akan dibahas mengenai proses uji coba yang digunakan. Pengujian dilakukan berdasarkan cara sistem bekerja.

5.3.1. Pengujian Kinerja

Pengujian dilakukan dengan mengukur waktu yang diperlukan oleh sistem untuk melakukan *authentication request* untuk mode *Poll*, *Ping*, dan *Push.* Masing masing pengujian ditunjukkan pada

Gambar 5.17 Waktu Response Authentication Request Mode Poll

Gambar 5.18 Waktu Response Authentication Request Mode Ping

Gambar 5.19 Waktu Response Authentication Request Mode Push

Dari pengujian terhadap waktu *response authentication request* pada setiap mode, mode *poll* memiliki performa rata-rata terbaik yaitu dengan 992.55ms, lalu mode *push* dengan waktu 1.02s dan mode *ping* dengan waktu 1.16s. waktu dari ketiga mode dapat dikatakan sama, karena perbedaannya tidak signifikan. Perbedaan dari masing-masing mode hanya terdapat pada pengecekan kondisi (*if statement*) dan keadaan *web server* yang tidak selalu sama setiap saat.

5.4. Evaluasi

Pada Sub bab ini dijelaskan hasil dari pengujian yang dilakukan pada Sub bab sebelumnya. Evaluasi yang tertulis adalah evaluasi fungsional sistem CIBA.

5.4.1. Evaluasi Fungsionalitas Sistem

Evaluasi ini adalah hasil dari pengujian pada sub bab 5.2. Hasil dinyatakan dengan status terpenuhi atau tidak. Hasil uji percobaan aplikasi ditunjukkan pada Tabel 5.10.

Tabel 5.10 Evaluasi Kasus Pengujian Fungsionalitas Sistem

No	Kode Uji	Terpenuhi
1	UJ-001A	✓
2	UJ-001B	✓
3	UJ-001C	✓
4	UJ-001D	✓
5	UJ-002A	✓
6	UJ-002B	✓
6	UJ-002C	✓
8	UJ-002D	✓
9	UJ-003A	✓
10	UJ-003B	✓
11	UJ-003C	✓
12	UJ-003D	✓
13	UJ-004	✓
14	UJ-005	✓
15	UJ-006A	✓
16	UJ-006B	✓

Hasil evaluasi yang ditunjukkan pada Tabel 5.10, semua uji coba yang dilakukan oleh pengguna terpenuhi sesuai skenario uji coba. Pada pengujian dengan kode uji UJ-006B, terkadang *Firebase* tidak mampu mengirim notifikasi secara *real-time*, dikarenakan pada *Firebase* terdapat Batasan untuk akun tidak berbayar atau tidak berlangganan *Firebase Cloud Messaging* sehingga notifikasi tidak dapat diterima oleh pemegang MyITS Authenticator, sehingga pada saat tertentu sistem tidak dapat melanjutkan alur otentikasi, cara untuk menangani hal ini adalah dengan melakukan pengulangan pengiriman notifikasi kepada MyITS Authenticator. Sistem dapat menjalankan alur CIBA sesuai spesifikasi yang telah tertulis pada spesifikasi CIBA. Sistem telah dapat mengimplementasikan alur CIBA dari sisi klien. Sistem juga dapat mengatasi *error* yang terjadi, di dalam implementasi alur CIBA pada sisi klien, jika terjadi *error* maka rangkaian alur CIBA dikatakan berhenti dan dianggap user tidak dapat melakukan *consent*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan yang diperoleh selama pengerjaan tugas akhir ini berdasarkan hasil pengujian dan hal lainnya yang telah dilakukan. Selain itu, juga terdapat beberapa saran terhadap tugas akhir ini untuk pengembangan kedepannya.

6.1. Kesimpulan

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

1. Protokol *Client Initiated Backchannel Authentication* (CIBA) di sisi klien dapat diimplementasikan pada MyITS *Single Sign-On* dengan mengimplementasikan sistem sesuai spesifikasi pada OpenID Connect serta membangun MyITS *Security Management* dengan *clean architecture*.
2. Komunikasi antara *Authentication Device* dengan *Authorization Server* dapat diimplementasikan pada MyITS *Single Sign-On* dengan memanfaatkan layanan *Firebase Cloud Messaging* sebagai protokol untuk mengirim *push notification.*
3. *Client notification endpoint* dapat diimplementasikan pada MyITS *Single Sign-On* sehingga *Authorization Server* dapat berkomunikasi dengan *Consumption Device* dengan membuat *endpoint* baru pada klien sehingga server dapat menghubungi klien untuk mengirimkan notifikasi.
4. *Firebase Cloud Messaging* tidak selalu dapat mengirimkan notifikasi kepada AD secara *real time* dikarenakan adanya batasan penggunaan pada pengguna non berbayar. Maka dari itu diperlukan layanan/sistem pengirim *push notification* lain yang lebih handal untuk komunikasi antara *Authotization Server* dengan *Consumption Device*.

6.2. Saran

Terdapat beberapa saran terkait tugas akhir ini yang diharapkan bisa membuat tugas akhir ini menjadi lebih baik. Saran-saran tersebut antara lain:

1. Kehandalan serta keamanan dari *Authentication Device* perlu ditingkatkan untuk menghindari penyalahgunaan.
2. Performa server pada myITS *Single Sign-On* perlu ditingkatkan untuk menangani *token request* yang terus menerus pada alur CIBA.

DAFTAR PUSTAKA

[1] "Introduction - Client Initiated Backchannel Authentication Flow." [Online]. Available: https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#rfc.section.1. [Accessed: 07-May-2020].

[2] K. Dwiastini, "Implementasi Otentikasi Single Sign On Dan Otorisasi Role Based Access Control Berbasis Standar OpenId Connect," Institut Teknologi Sepuluh Nopember, 2018.

[3] D. Hardt, Ed, "RFC 6749-OAuth2 Authorization Code" 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749#section-1.3.1. [Accessed: 07-May-2020].

[4] M. Anicas, "An Introduction to OAuth 2 | DigitalOcean," 2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/anintroduction-to-oauth-2. [Accessed: 07-May-2020].

[5] M. Jones, "RFC 7519- JSON Web Token (JWT)" 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519. [Accessed: 07-May-2020].

[6] Jones & Hardt, "RFC 6750- Authorization Request Header Field" 2012. [Online]. Available: https://tools.ietf.org/html/rfc6750#section-2.1. [Accessed: 07-May-2020].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis lahir di Ogan Komering Ilir, 8 Agustus 1998. Penulis telah menempuh pendidikan dasar di SDN 01 Klakah Lumajang, kemudian untuk pendidikan menengah pertama di SMPN 01 Sukodono Lumajang dan di jenjang menengah atas di SMAN 02 Lumajang. Sejak kecil, penulis suka dengan hal-hal terkait perkembangan teknologi dan perkembangan komputer. Hal tersebut juga yang mendasari penulis melanjutkan pendidikan sarjana S1 di Jurusan Teknik Informatika, Fakultas Teknologi Elektro dan Informtika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif berorganisasi menjadi Staf Ahli Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS 2018/2019, staf REEVA Schematics ITS 2017/2018, staf Ahli REEVA Schematics ITS 2018/2019, Manager Branding & Communication Duacare 2018-2020, dan Ketua Umum Ikatan Mahasiswa Lumajang di Surabaya 2018/2019.

Penulis dalam menyelesaikan pendidikan S1 mengambil rumpun mata kuliah (RMK) Rekayasa Perangkat Lunak serta memiliki ketertarikan di bidang Sistem dan Manajemen Basis Data, Pemrograman *Web*. Untuk komunikasi, penulis dapat dihubungi melalui surel: andikaandra03@gmail.com