TUGAS AKHIR – IF184802

RANCANG BANGUN MODUL SINGLE SIGN-ON MYITS BERBASIS OPENID CONNECT UNTUK APLIKASI ERP ODOO LOGISTIK TERPUSAT ITS

HANIF NASHRULLAH
05111540000140

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

RANCANG BANGUN MODUL SINGLE SIGN-ON MYITS BERBASIS OPENID CONNECT UNTUK APLIKASI ERP ODOO LOGISTIK TERPUSAT ITS

HANIF NASHRULLAH
05111540000140

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

FINAL PROJECT - IF184802

DESIGN AND DEVELOPMENT OF MYITS SINGLE SIGN-ON MODULE BASED ON OPENID CONNECT FOR ODOO ERP APPLICATION OF ITS CENTRAL LOGISTICS

HANIF NASHRULLAH
05111540000140

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

Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN MODUL SINGLE SIGN-ON MYITS BERBASIS OPENID CONNECT UNTUK APLIKASI ERP ODOO LOGISTIK TERPUSAT ITS

TUGAS AKHIR

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

Oleh:
Hanif Nashrullah
NRP: 05111540000140

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Abdul Munif, S.Kom., M.Sc.
NIP. 198608232015041004

Abdul Munif
.....
(Pembimbing)

SURABAYA
MEI 2020

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN MODUL SINGLE SIGN-ON MYITS BERBASIS OPENID CONNECT UNTUK APLIKASI ERP ODOO LOGISTIK TERPUSAT ITS

Nama Mahasiswa : Hanif Nashrullah
NRP : 05111540000140
Departemen : Informatika FTIK - ITS
Dosen Pembimbing : Abdul Munif, S.Kom., M.Sc.

Abstrak

Kemajuan teknologi yang pesat akhir-akhir ini, telah memunculkan berbagai inovasi di berbagai bidang. Salah satu contoh dari inovasi tersebut adalah SSO (*Single Sign-On*). Dengan SSO, pengguna dapat melakukan ontetikasi untuk *login* ke dalam suatu sistem tanpa harus berganti-ganti akun. Teknologi autentikasi dengan SSO yang umum digunakan saat ini adalah OpenID Connect. OpenID Connect adalah protokol autentikasi yang berjalan berdasarkan OAuth 2.0. Saat ini ITS telah memiliki sistem SSO sendiri yaitu MyITS, namun belum semua sistem di lingkungan ITS terintegtasi dengan sistem SSO tersebut, seperti aplikasi ERP (*Enterprise Resource Planning*) Odoo milik Sub-Direktorat Logistik ITS.

Pada Tugas Akhir kali ini, telah dibangun suatu modul khusus untuk aplikasi Odoo yang memungkinkan pengguna untuk *login* kedalam sistem melalui sistem SSO MyITS. Modul khusus ini ditujukan kepada aplikasi Odoo 10 milik Sub-Direktorat Logistik ITS. Modul ini dibangun dan disesuaikan dengan sistem SSO dan versi Odoo yang sedang diimplementasikan saat ini.

Modul khusus ini menghubungkan akun pada aplikasi Odoo milik Sub-Direktorat Logistik ITS dengan akun pengguna yang terdaftar di dalam sistem SSO MyITS. Administrator aplikasi Odoo pada Sub-Direktorat Logistik ITS yang memiliki wewenang

untuk memasang modul ini dapat melakukan manajemen *provider* SSO dan *user*. Setelah administrator menyediakan layanan SSO pada suatu *user*, *user* tersebut dapat *login* kedalam sistem Odoo menggunakan SSO MyITS dengan menekan tombol "Masuk dengan MyITS" pada halaman `Login.` *User* kemudian diarahkan ke SSO MyITS dan melakukan *login* dengan akun MyITS. Jika *login* berhasil, MyITS kemudian mengarahkan kembali bersamaan dengan sebuah kode khusus untuk *user* tersebut. Kode tersebut digunakan oleh modul untuk autentikasi lebih lanjut. Jika *user* berhasil terautentikasi, *user* akan diberikan *session* oleh sistem Odoo.

Pengujian modul dilakukan dengan pengujian 3 skenario simulasi yang berbeda dan pengujian performa dengan pengukuran waktu pemrosesan autentikasi. Dari ketiga skenario pengujian simulasi, modul berhasil memproses autentikasi dan *user* berhasil *login* sesuai dengan akun masing-masing. Pada pengujian performa, modul berhasil memproses autentikasi dalam waktu kurang dari 5 detik. Namun lama pemrosesan autentikasi tetap bergantung pada kecepatan *provider* memproses autorisasi dan kecepatan koneksi jaringan.

Kata kunci: Single Sign-On, Odoo, OpenID Connect, Autentikasi

DESIGN AND DEVELOPMENT OF MYITS SINGLE SIGN-ON MODULE BASED ON OPENID CONNECT FOR ODOO ERP APPLICATION OF ITS CENTRAL LOGISTICS

Student Name	: Hanif Nashrullah
Registration Number	: 05111540000140
Department	: Informatics FTIK - ITS
Supervisor	: Abdul Munif, S.Kom., M.Sc.

Abstract

With the current fast-forwarding technology development, many innovations have been emerged in many aspects. One of those innovations is SSO (Single Sign-On). Using SSO, a user could authenticate him/herself for a login attempt without switching accounts. Currently, the most commonly used SSO authentication technology is OpenID Connect. OpenID Connect is an authentication layer that works based on OAuth 2.0. Today, ITS has its own SSO system that called MyITS, nevertheless not all systems in ITS have been integrated with it. One of those unintegrated systems is Odoo ERP (Enterprise Resource Planning) application by ITS Central Logistics.

In this Final Project, a specific Odoo application module has been developed that enables users to login into system through SSO MyITS. This module is intended for Odoo 10 application belongs to ITS Central Logistics. This module is developed and synchronized based on currently implemented Odoo version and SSO system.

This specific module will connect user's account in Odoo belongs to ITS Central Logistics with the user's account in SSO MyITS system. The administrator in Odoo application belongs to ITS Central Logistics will have privilege to install this module and

then manage user's data and provider's configurations. After the administrator enables SSO service to a user, that user could login into system using SSO MyITS by clicking "Masuk dengan MyITS" in Login page. The user then redirected to SSO MyITS login page and then login using MyITS account. If the user successfully logged in, MyITS would redirect the user back to Odoo with an authorization code. That code will be used by the module for further authentication process. If the user is authenticated, Odoo system will grant a session to the user.

Testing on the module is done by using 3 different scenarios for simulation testing and authentication process time measuring on performance testing. From each scenario on simulation testing, the module successfully authenticated users and each user successfully logged in according on their respective accounts. On performance testing, the module could process user's authentication in less than 5 seconds. However, how much time needed by the module to authenticate users still depends on provider's speed on authorization processing and network speed.

Keywords: Single Sign-On, Odoo, OpenID Connect, Authentication

KATA PENGANTAR

Puji syukur penulis sampaikan kepada Tuhan Yang Maha Esa karena atas segala rahmat dan hidayah-Nya, penulis dapat menyelesaikan Tugas Akhir ini yang berjudul :

"RANCANG BANGUN MODUL SINGLE SIGN-ON MYITS BERBASIS OPENID CONNECT UNTUK APLIKASI ERP ODOO LOGISTIK TERPUSAT ITS"

Pengerjaan Tugas Akhir ini penulis lakukan demi memenuhi salah satu syarat untuk menyelesaikan pendidikan Sarjana di Departemen Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember. Penulis berharap dengan selesainya Tugas Akhir ini, penulis dapat memberikan manfaat kepada pihak-pihak yang terkait maupun tidak terkait.

Dalam menempuh masa perkuliahan dan pengerjaan Tugas Akhir ini, penulis telah menerima banyak bantuan dan dukungan dari berbagai pihak baik secara langsung maupun tidak langsung. Maka dari itu, penulis ingin menyampaikan banyak terima kasih kepada:

1. Allah SWT atas segala nikmat, rahmat, hidayah, kemudahan, dan kesehatan-Nya selama ini.
2. Bapak, ibu, kakak saya Desy, dan keluarga penulis yang telah memberikan banyak perhatian dan dukungan baik materi maupun psikologi kepada penulis sehingga penulis mendapatkan motivasi selama masa perkuliahan maupun pengerjaan Tugas Akhir ini.
3. Bapak Abdul Munif, S.Kom., M.Sc. sebagai dosen pembimbing yang telah bersedia meluangkan waktu dan memberikan bimbingan, ilmu, dan nasihat kepada penulis selama pengerjaan Tugas Akhir ini.
4. Bapak Hatma Suryotrisongko S.Kom., M.Eng. selaku kepala Sub-Div Logistik Terpusat ITS saat penulis mengajukan proposal Tugas Akhir ini yang telah memberikan kesempatan kepada penulis serta

memberikan informasi yang diperlukan untuk mengembangkan modul khusus ini.

5. Keluarga besar Departemen Informatika ITS baik dosen maupun karyawan yang telah memberikan ilmu dan dukungan selama penulis kuliah di Departemen Informatika ITS.
6. Albertus Tommy Halim Putra, Benito Danneswara Widyatama, dan Barep Bimo Pangestu yang telah menjadi sahabat penulis baik di perkuliahan, UKM, hobi, dan kehidupan sehari-hari serta memberikan dukungan maupun bantuan kepada penulis dan juga telah berbaik hati meminjamkan akun MyITS untuk pengujian Tugas Akhir ini.
7. Teman-teman seperhobian baik dari grup ITS JADDICT, *fandom* BanG Dream, *fandom* Hololive, maupun lainnya yang telah menemani penulis baik saat *event* maupun diluar *event* dan telah memberikan dukungan psikologis kepada penulis selama ini.
8. Teman-teman UKM IFLS yang selama ini bersama-sama dengan penulis menghadapi susah-senang dalam menjalankan suatu organisasi, menyelenggarakan suatu *event*, serta memberikan dukungan dan pengalaman berharga kepada penulis selama berkuliah di ITS.
9. Teman-teman penulis dari Jakarta yang telah menemani penulis saat di Jakarta dan memberikan dukungan psikologis kepada penulis.
10. Teman-teman angkatan 2015 di Departemen Teknik Informatika ITS yang telah menjalani masa perkuliahan bersama-sama dengan penulis dan memberikan dukungan dan bantuan kepada penulis.
11. Serta pihak-pihak lain yang tidak dapat penulis sebutkan satu per satu yang telah memberikan dukingan dan bantuan selama penulis menjalani perkuliahan dan pengerjaan Tugas Akhir ini.

Penulis mohon maaf apabila masih ditemukan kesalahan dan kekurangan pada Tugas Akhir ini. Maka dari itu penulis sangat mengharapkan kritik dan saran yang membangun sebagai pembelajaran bagi penulis agar menjadi lebih baik lagi. Semoga dengan Tugas Akhir ini penulis dapat memberikan manfaat kepada penulis sendiri, pihak-pihak yang terkait maupun tidak terkait, dan juga pembaca.

Surabaya, Mei 2020

Hanif Nashrullah

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab I dibahas mengenai latar belakang, tujuan, rumusan masalah, batasan permasalahan, metodologi, serta sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Seiring berjalannya waktu, perkembangan teknologi semakin maju dan pesat. Ide-ide baru memunculkan inovasi baru hingga mencapai tingkat teknologi saat ini. Dengan aplikasi-aplikasi yang telah dikembangkan sebagai realisasi dari inovasi-inovasi tersebut, manusia dapat melakukan pekerjaan dan aktivitasnya secara lebih mudah dan cepat.

Banyak aplikasi yang telah dikembangkan untuk membantu kerja, salah satunya adalah Odoo. Odoo merupakan suatu aplikasi yang menyediakan fitur-fitur manajerial untuk suatu badan ataupun perusahaan. Dengan fitur-fitur yang tersedia serta modul yang *customizable*, Odoo telah membantu banyak pekerjaan dalam bidang industri dan bisnis.

Saat ini telah banyak perusahaan dan badan yang menggunakan Odoo untuk membantu kerja badan/perusahaan tersebut. Salah satu badan yang menggunakan Odoo di lingkungan ITS adalah Sub-Direktorat Logistik ITS. Aplikasi Odoo yang digunakan oleh badan tersebut telah dikostumisasi agar dapat terintegrasi dengan proses bisnis di Logistik Terpusat ITS. Aplikasi tersebut telah membantu operasional Logistik Terpusat ITS dan merealisasikan sistem *paperless* yang mudah dan praktis.

ITS telah memiliki sistem untuk *login* secara terpusat menggunakan akun yang terdaftar di MyITS. MyITS merupakan sistem SSO (*Single Sign-On*) yang menggunakan basis OpenID Connect sebagai *authentication layer* untuk sistem-sistem di lingkungan ITS. Saat Tugas Akhir ini dibuat, aplikasi Odoo yang digunakan oleh Sub-Direktorat Logistik ITS masih.menggunakan akun tersendiri yang terpisah dari SSO MyITS.

Pada Tugas Akhir ini, penulis mengusulkan pembuatan modul yang dapat mengintegrasikan akun pengguna aplikasi Odoo Logistik Terpusat ITS dengan akun yang terdaftar di MyITS sehingga pengguna aplikasi Odoo Logistik Terpusat ITS dapat *login* ke aplikasi melalui sistem SSO MyITS.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana desain dan spesifikasi dari modul SSO MyITS untuk aplikasi Odoo Logistik Terpusat ITS?
2. Bagaimana proses bisnis dalam modul SSO MyITS untuk aplikasi Odoo Logistik Terpusat ITS?
3. Bagaimana transaksi data antara SSO MyITS dengan Odoo Logistik Terpusat ITS?
4. Bagaimana aktivitas dan proses bisnis pengguna yang *login* melalui SSO MyITS?
5. Bagaimana kecepatan modul SSO MyITS untuk aplikasi Odoo Logistik Terpusat ITS dalam memproses autentikasi?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Modul ini diimplementasikan dalam lingkungan Odoo Versi 10
2. Modul yang dibangun hanya diimplementasikan untuk aplikasi Odoo Logistik Terpusat ITS.
3. Modul dibangun berdasarkan OpenID Connect sebagai *authentication layer*.
4. Modul menggunakan sistem SSO MyITS.
5. Modul Odoo dan SSO MyITS hanya melakukan pemrosesan *login*.
6. Akun yang dapat *login* ke aplikasi Odoo Logistik Terpusat ITS hanya akun yang terdaftar di sistem SSO

MyITS dan diberi wewenang oleh Sub-Direktorat Logistik ITS.
7. Modul dibangun berdasarkan protokol yang telah ditentukan pada sistem SSO MyITS.
8. Pengujian modul tidak mencakup pengujian *login* lebih dari satu pengguna secara paralel.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan tugas akhir ini adalah membuat modul yang memungkinkan pengguna untuk *login* ke aplikasi Odoo Logistik Terpusat ITS melalui SSO MyITS.

1.5 Manfaat Tugas Akhir

Tugas akhir ini diharapkan dapat memberi manfaat yaitu memudahkan pengguna untuk *login* ke aplikasi Odoo Logistik Terpusat ITS melalui SSO MyITS.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap pertama dalam proses pengerjaan tugas akhir ini adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir ini tersusun dari latar belakang, rumusan dan batasan masalah, tujuan dan manfaat, tinjauan pustaka, ringkasan, metodologi, dan jadwal kegiatan dari Tugas Akhir yang akan dibuat. Pada latar belakang dijabarkan dasar dorongan dalam pengerjaan Tugas Akhir ini. Tinjauan pustaka berisi referensi pendukung dalam pembuatan Tugas Akhir. Ringkasan isi Tugas Akhir menjelaskan secara singkat alur dan gambaran dari tugas akhir yang akan dibuat. Metodologi menjelaskan tahapan penyusunan tugas akhir. Setelah metodologi terdapat jadwal kegiatan pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Tahap kedua adalah studi literatur, yaitu penulis mengumpulkan berbagai referensi yang berkaitan dengan pengerjaan Tugas Akhir. Bentuk-bentuk referensi dapat berupa buku, artiket internet, *scientific paper*, maupun materi pelajaran yang berhubungan dengan Odoo, *Single Sign-On*, OpenID Connect, Python dan XML.

1.6.3 Analisis dan Desain Perangkat Lunak

Tahap ketiga adalah analisis dan desain perangkat lunak. Pada tahap ini dilakukan penggalian kebutuhan dan perancangan dari perangkat lunak yang akan dibuat. Selain itu dilakukan peninjauan mengenai bagaimana interaksi antara modul Odoo yang akan dibuat, aplikasi Odoo Logistik Terpusat ITS, dan SSO MyITS.

1.6.4 Implementasi Perangkat Lunak

Tahap keempat adalah implementasi perangkat lunak, dimana rancangan dan hasil analisis yang telah didapatkan dari tahap sebelumnya diimplementasikan menjadi modul Odoo. Modul Odoo dibuat dengan menggunakan XML dan Python.

1.6.5 Pengujian dan Evaluasi

Setalah modul Odoo selesai dibuat, dilakukan tahap pengujian dan evaluasi. Pengujian dilakukan dengan memasang modul Odoo tersebut pada aplikasi Odoo dan kemudian diujicoba menggunakan akun yang terdaftar di SSO MyITS. Hasil pengujian kemudian digunakan untuk evaluasi dan *debugging*.

1.6.6 Penyusunan Buku Tugas Akhir

Tahap yang terakhir adalah penyusunan buku tugas akhir yang menjelaskan dasar teori, metodologi yang digunakan, serta hasil dari implementasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Manfaat
    f. Metodologi
    g. Sistematika Penulisan
2. Dasar Teori
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan

Buku Tugas Akhir ini disusun sebagai dokumentasi dan laporan mengenai Tugas Akhir yang telah dikerjakan. Dengan penyusunan buku Tugas Akhir ini, diharapkan pembaca dapat memahami proses pembuatan Tugas Akhir yang dikerjakan oleh penulis serta memberi wawasan bagi yang tertarik untuk mengembangkan lebih lanjut. Sistematika penulisan dari buku Tugas Akhir ini secara garis besar adalah sebagai berikut:

Bab I Pendahuluan

Pada bab I dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, serta sistematika penulisan dari buku Tugas Akhir ini.

Bab II Dasar Teori

Pada bab II dijelaskan mengenai beberapa teori yang berkaitan dengan Tugas Akhir. Teori-teori ini dijadikan sebagai penunjang dalam proses pengerjaan Tugas Akhir ini.

Bab III Analisis dan Perancangan

Pada bab III dibahas mengenai lingkungan dimana modul akan berjalan serta perancangan dari sistem dan modul yang akan diimplementasikan. Perancangan meliputi *parameter*, penyimpanan data, serta alur proses dari modul yang diimplementasikan.

Bab IV Implementasi

Pada bab IV dijelaskan mengenai implementasi modul berdasarkan analisis dan perancangan yang telah dilakukan sebelumnya. Implementasi yang ditunjukkan berupa *source code* disertai dengan penjelasan.

Bab V Pengujian dan Evaluasi

Pada bab V dijelaskan mengenai ruang lingkup pengujian yang dilakukan dengan kondisi dan alur bisnis tertentu serta hasil dari pengujian yang dilakukan.

Bab VI Kesimpulan dan Saran

Pada bab VI dijelaskan mengenai kesimpulan dari hasil pengujian yang dilakukan sebelumnya serta saran mengenai pengembangangan modul untuk kedepannya.

BAB II
DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini.

2.1 Autentikasi

Dalam dunia komputer, autentikasi merujuk kepada suatu proses untuk memverifikasi identitas suatu orang, perangkat atau aplikasi. Dalam proses autentikasi, pengguna, perangkat, ataupun aplikasi membuktikan identitas mereka dengan memberikan informasi yang diperlukan untuk mengidentifikasi siapa mereka dan apakah benar mereka yang mengaksesnya. Contoh umum proses autentikasi adalah saat pengguna melakukan *login* ke suatu *website* dengan memberikan *username* dan *password*. Selain itu juga terdapat metode autentikasi lainnya, seperti dengan memberikan empat atau enam kode angka. [1]

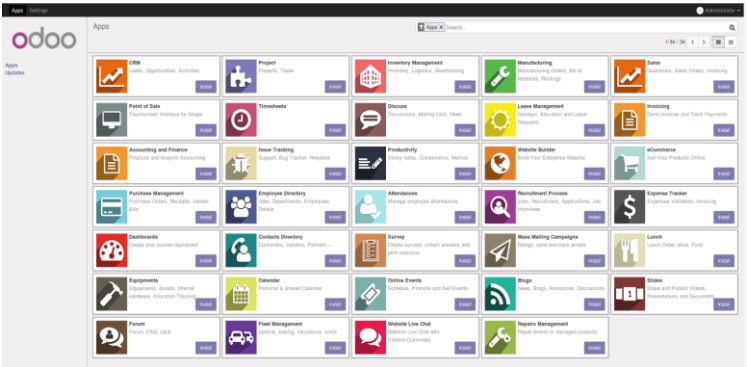
Autentikasi dapat dibagi menjadi tiga kategori berdasarkan jenis informasi yang diberikan, atau disebut *factor*. Kategori-kategori autentikasi adalah sebagai berikut:

- ***Single-Factor Authentication***
  Single-Factor Authentication bergantung kepada sebuah kata sandi sebagai informasi untuk memvalidasi pengguna, umumnya bersamaan dengan *username*.
- ***Two-Factor Authentication***
  Two-Factor Authentication (atau yang umumnya disingkat 2FA) bergantung kepada sebuah informasi tambahan selain *username* dan *password*. Contohnya adalah *website* atau aplikasi keuangan yang memvalidasi pengguna dengan *username* dan *password* yang kemudian memerlukan pengguna untuk memasukkan 6 digit angka atau PIN yang hanya diketahui oleh pengguna.
- ***Multi-Factor Authentication***
  Multi-Factor Authentication memerlukan dua atau lebih *factor* dengan kategori yang berbeda-beda. Contohnya

adalah sistem rumah sakit dimana pengguna perlu memasukkan *username* dan *password*, lalu memberikan kode keamanan dan sidik jari. [2]

2.2 Odoo

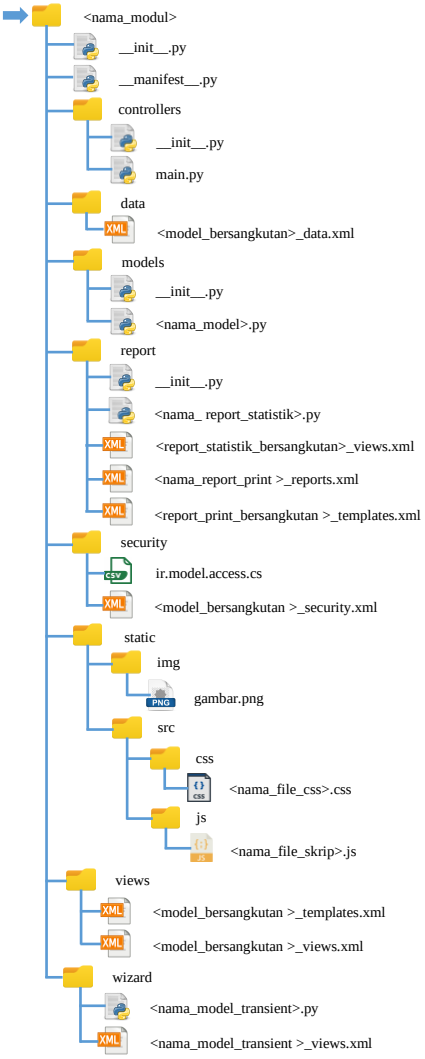
Odoo adalah *Enterprise Resource Planning* (ERP) *software* yang terdiri dari kumpulan aplikasi bisnis yang dibangun berdasarkan *framework* OpenObject. Odoo memiliki modul-modul yang dapat di *install* sesuai dengan kebutuhan pengguna [3]. Odoo dibangun menggunakan bahasa pemrograman Python dan menggunakan basis data PostgreSQL sebagai tempat penyimpanan data. Odoo bersifat *open source* dan modul dapat dikembangkan berdasarkan kebutuhan pengguna. Pengembangan modul menggunakan XML (*Extensible Markup Language*) dan bahasa pemrograman Python [4]. Gambar 2.1 adalah tampilan Odoo 10 setelah instalasi.



Gambar 2.1 Tampilan Odoo 10

Pengembangan dalam Odoo pada umumnya dalam bentuk pembuatan modul. Odoo mengikuti aksitektur MVC (*Model-View-Controller*). *Model* sebagai bagian untuk mendefinisikan struktur data. *View* sebagai bagian untuk mengatur *user interface*. Sedangkan *Controller* sebagai bagian untuk menjalankan logika bisnis dari aplikasi [4].

Secara umum struktur modul pada Odoo versi 10 dapat digambarkan seperti pada Gambar 2.2 berikut:



Gambar 2.2 Struktur Modul pada Odoo 10

Modul Odoo disusun dengan ditata berdasarkan direktori-direktori. Setiap direktori memiliki peran tersendiri dalam menjalankan proses bisnis. Fungsi tiap direktori tersebut adalah sebagai berikut:

- **`data`** : penyimpanan data bawaan atau data *demo*.
- **`models`** : pendefinisian struktur data.
- **`controllers`** : menjalankan logika bisnis, menerima dan merespon *request*.
- **`views`** : mengatur *user interface* dan *templates*.
- **`static`** : menyimpan *assets* dari halaman *web*, seperti CSS, Javascript, gambar, dan lain-lain.
- **`report`** : membuat *report*.
- **`wizard`** : menampung model-model *transient* dan *view*-nya.
- **`security`** : mengatur pembatasan akses pada modul [5].

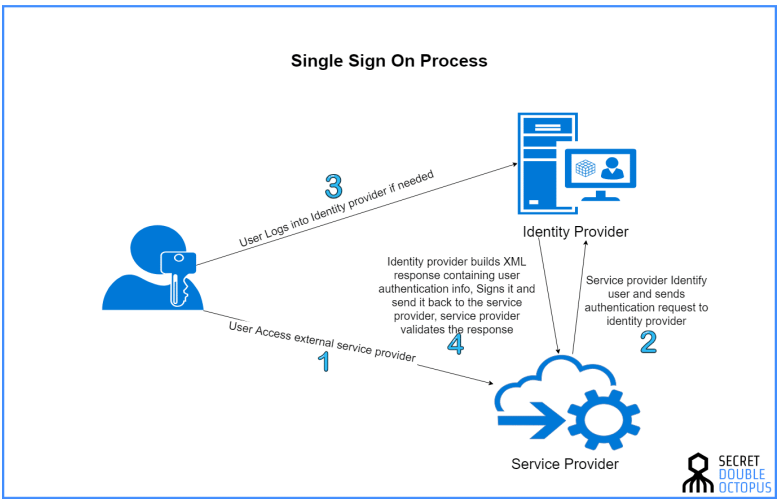
2.3 *Single Sign-On* (SSO)

Single Sign-On (SSO) adalah sebuah proses yang memberikan pengguna akses ke berbagai layanan dengan melalui hanya satu proses autentikasi. Proses ini juga termasuk autentikasi terhadap semua layanan yang diperbolehkan untuk digunakan oleh pengguna. Dengan SSO, pengguna tidak perlu melakukan autentikasi berulang-ulang. Prinsip dasar dari SSO adalah:

1. Tahap pertama adalah mengakses sebuah penyedia layanan lalu *login* melalui penyedia layanan autentikasi yang terhubung dengan penyedia layanan tersebut.
2. Ketika pengguna menggunakan sebuah layanan baru, layanan tersebut akan melakukan *redirect* ke penyedia layanan autentikasi untuk memastikan apakah pengguna sudah *login*.
3. Layanan autentikasi kemudian memberikan OTP (*One-time password*) ke layanan baru tersebut.

4. Layanan baru tersebut kemudian melakukan verifikasi menggunakan OTP yang diberikan, dan jika OTP terverifikasi maka pengguna diperbolehkan masuk [6].

Pada Gambar 2.3 dijelaskan mengenai proses dalam *Single Sign-On.*



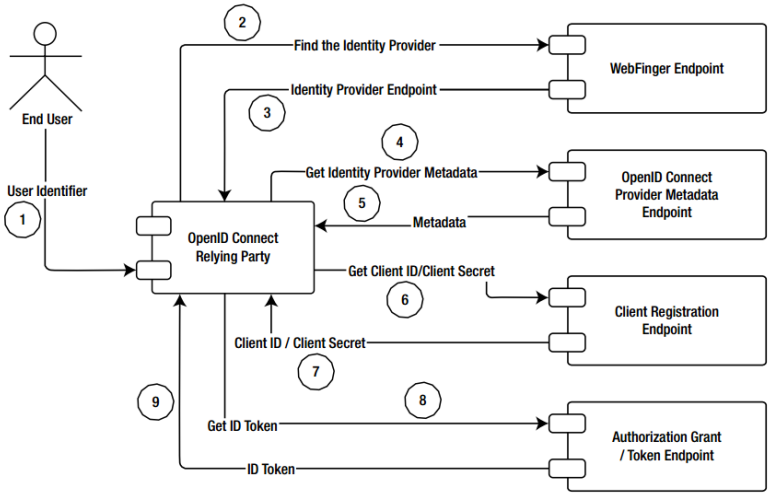
Gambar 2.3 Proses dalam *Single Sign-On* [7]

2.4 OpenID Connect

OpenID adalah protokol autentikasi yang memungkinkan pengguna untuk *login* ke *website* melalui akun terautentikasi pada penyedia layanan OpenID. Dengan OpenID, pengguna hanya cukup menggunakan satu akun untuk *login* ke *website* yang berbeda-beda. *Website* yang menngunakan autentikasi OpenID terhubung dengan penyedia layanan autentikasi OpenID untuk mendapatkan informasi akun pengguna.

OpenID Connect merupakan generasi ke-3 dari OpenID. OpenID Connect mengusung *identity layer* diatas OAuth 2.0. Untuk melakukan autentikasi menggunakan OpenID Connect ke suatu *website*, pihak *website* tersebut perlu mengetahui *authorization server* yang digunakan oleh pengguna. Setalah

pengguna terotorisasi, *authorization server* mengirim JWT (*JSON Web Token*) atau yang disebut *ID token* yang digunakan oleh pihak website untuk mengautentikasi pengguna. Gambar 2.4 menjelaskan alur cara kerja dari OpenID Connect [8].



Gambar 2.4 Alur Cara Kerja OpenID Connect [8]

2.5 Python

Python merupakan sebuah bahasa pemrograman komputer. Program yang ditulis menggunnakan bahasa Python disimpan dalam file berformat ".py". Untuk menjalankan sebuah program Python, digunakan program Python Interpreter. Walaupun dijalankan dengan interpreter, sebenarnya program Python dikompilasi terlebih dahulu. Hasil dari kompilasi program Python yang berupa *byte code* disimpan dalam file berformat ".pyc". Kemudian, *Python Management System* (PMC) mengeksekusi *byte code* tersebut. *Python Virtual Machine* (PVM) merupakan mesin runtime dan bagian dari sistem Python. Pada Python, *byte code* yang dijalankan disimpan di memori komputer dan dibuang ketika program selesai dijalankan. Hal ini menjadikan proses eksekusi

program pada Python berjalan cepat [9]. Gambar 2.5 adalah logo bahasa pemrograman Python.

Gambar 2.5 Logo Bahasa Pemrograman Python [10].

2.6 XML

XML (*Extensible Markup Language*) adalah sebuah bahasa untuk menyimpan, memberi bentuk, melindungi, dan melabelkan data. *Markup* pada XML membagi sebuah dokumen menjadi kontainer informasi terpisah-pisah yang disebut dengan elemen. Sebuah elemen dapat menampung elemen lain, sehingga dapat terbentuk struktur hirarki yang tidak ambigu yang menyimpan segala bentuk informasi tambahan. Dokumen XML terdiri dari sebuah elemen paling luar yang menampung elemen-elemen lain. Informasi tambahan mengenai dokumen dapat ditambahkan dibagian paling atas dokumen [11].

2.7 HTML

HTML (*HyperText Markup Language*) adalah sebuah bahasa yang digunakan dalam pembuatan halaman *web* untuk menampilkan berbagai macam informasi melalui aplikasi penjelajah internet/*browser* dengan menggunakan pemformatan hiperteks yang disimpan dalam bentuk file dengan pengkodean ASCII sehingga bisa memberikan tampilan yang saling terintegrasi. HTML telah digunakan secara luas dan dijadikan standar dalam internet yang diatur oleh *World Wide Web Consortium* (W3C) untuk membuat halaman *web*. HTML berawal dari sebuah bahasa yang sebelumnya umum digunakan dalam

dunia percetakan dan penerbitan yaitu SGML (*Standardized General Markup Language*).

HTML adalah sebuah bahasa *markup* dan bukan bahasa pemrograman. HTML terdiri dari perintah-perintah yang terstruktur dengan menggunakan *tag*. *Tag* yang digunakan pada HTML tidak *case-sensitive* sehingga tidak masalah untuk menggunakan huruf besar maupun kecil, namun penulisannya memiliki aturan tersendiri. Aturan dalam penulisan *tag* dalam HTML adalah sebagai berikut:

- Tiap *tag* harus berpasangan, kecuali untuk beberapa *tag* yang dapat ditulis secara tunggal.
- Sebuah *Tag* dapat ditempatkan di dalam *tag* lain diantara *tag* awal dan *tag* akhir.
- Penulisan setiap *tag* tidak boleh saling tumpang tindih satu sama lain [12].

2.8 CSS

CSS (*Cascading Style Sheet*) adalah sebuah bahasa yang digunakan untuk mengatur tatanan dan desain dari suatu halaman *web*. Dengan CSS, pengguna dapat memperindah tampilan halaman *web* seperti dengan memberikan desain pada *headings, borders, banner,* maupun konten dari halaman *web* itu sendiri sehingga tampilan *web* dapat sebagus majalah. CSS membantu mempersingkat proses dalam mendesain halaman *web.*

CSS berbeda dengan HTML, dimana HTML memberikan struktur dari halaman *web* dengan mengorganisasi informasi yang ada kedalam bagian-bagian atau *tag* seperti *headers*, *footer*, dan lain-lain. CSS bekerjasama dengan HTML untuk memberikan desain yang ditampilkan melalut *web browser*. Dengan kata lain, CSS berkaitan tentang merubah atau memperindah tampilan dari HTML.

Penggunaan CSS memberikan beberapa manfaat, yaitu:

- Membutuhkan lebih sedikit ruang dibadingkan dengan *formatting* langsung di HTML.

- Mempermudah dalam pembaharuan atau perubahan desain.
- Memberiksn lebih banyak opsi dalam proses *formatting* dibandingkan dengan langsung di HTML.
- Dapat memberikan pengaturan pengulangan *background* [13].

2.9 JavaScript

JavaScript adalah sebuah bahasa pemrograman interpretatif yang memiliki kemampuan *object-oriented* (OO). Berdasarkan sintaks, JavaScript menyerupai bahasa C, C++, dan Java, namun penggunannya tidak seketat bahasa-bahasa tersebut, dimana tipe dari variabel tidak harus ditentukan. JavaScript bukanlah Java, dimana Java dikembangkan oleh Sun Microsystem, sedangkan JavaScript dikembangkan oleh Netscape.

JavaScript umumnya digunakan di *web browser*, dimana skrip berkaitan dengan objek dalam halaman *web* sehingga pengguna dapat berinteraksi denga skrip. Dengan JavaScript, pengguna dapat mengatur *web browser*, dan mengubah konten dokumen yang tampil di halaman *web*. Dalam hal ini, JavaScript diletakkan di dalam halaman *web* HTML dan dijalankan di computer *client*, yang umumnya disebut *client-side* JavaScript.

Core dari JavaScript dan tipe data bawaannya telah menjadi bagian dari standar internasional, dan kompabilitas dalam berbagai macam penggunaan relatif sangat bagus. Terdapat bagian dari *client-side* JavaScript yang telah distandardisasi secara formal maupun bagian yang penggunaanya spesifik pada *browser* tertentu. Kemampuan lintas *browser* sering menjadi perhatian penting dalam pengembangan aplikasi *web* yang menggunakan *client-side* JavaScript [14].

2.10 Penelitian Terkait

Pada penelitian sebelumnya, dilakukan pengimplementasian pemrosesan autentikasi menggunakan satu faktor dan banyak

faktor berbasis Webauthn pada aplikasi SSO MyITS. Penelitian tersebut bertujuan untuk meningkatkan keamanan data pengguna dengan mengimplementasikan protokol Webauthn pada sistem SSO MyITS. Protokol Webauthn tersebut digunakan sebagai komplemen dari protokol OpenID Connect yang telah digunakan pada sistem SSO MyITS saat penelitian tersebut dilakukan [15].

Hubungan penelitian tersebut dengan Tugas Akhir ini adalah penelitian tersebut berfokus pada pengembangan protokol autentikasi Webauthn melengkapi protokol OpenID Connect yang sudah digunakan pada sistem SSO MyITS. Pada penelitian tersebut, pengguna diberikan pilihan untuk melakukan autentikasi pada sistem SSO MyITS dengan metode *Single-Factor Authentication* menggunakan protokol OpenID Connect atau dengan metode *Double-Factor Authentication* menggunakan protokol Webauthn. Sedangkan pada Tugas Akhir ini, protokol OpenID Connect yang diterapkan pada sistem SSO MyITS digunakan sebagai dasar perancangan dan pengembangan Modul SSO MyITS untuk aplikasi ERP Odoo Logistik Terpusat ITS.

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan hal-hal yang berkaitan dengan analisis dan perancangan dari sistem dan modul SSO MyITS untuk Odoo 10 milik Sub-Direktorat Logistik ITS.

3.1 Deskripsi Secara Umum

Modul yang dikembangkan akan menghubungkan sistem Odoo milik Sub-Direktorat Logistik ITS dan SSO MyITS. Odoo milik Sub-Direktorat Logistik ITS dapat diakses di http://erp.logistik.its.ac.id/ dan SSO MyITS dapat diakses di https://my.its.ac.id/. Sistem Odoo utama yang digunakan oleh Sub-Direktorat Logistik ITS adalah Odoo versi 10. Saat berkomunikasi dengan pihak Sub-Direktorat Logistik ITS, dalam Odoo mereka telah terpasang modul-modul sebagai berikut:

- `eCommerce`
- `Inventory`
- `Sales`
- `Purchase`
- `Accounting`

Modul-modul berikut akan ikut dipasang untuk diikutsertakan dalam ujicoba modul SSO MyITS.

Pada aplikasi Odoo, administrator berhak mengatur konfigurasi modul SSO MyITS dan menghubungkan akun Odoo milik pengguna pada sistem dengan akun MyITS. Modul SSO MyITS akan menerima *request* dari pengguna untuk melakukan *login* melalui SSO MyITS. Kemudian pengguna akan diarahkan ke halaman SSO MyITS untuk melakukan *login* menggunakan akun MyITS miliknya. Jika berhasil, maka SSO MyITS akan memberikan kode autorisasi yang dapat digunakan oleh modul SSO MyITS untuk proses autentikasi lebih lanjut. Jika berhasil terautentikasi, maka pengguna akan diberikan *session* sesuai dengan akun yang bersangkutan.

3.2 Analisis Kebutuhan

Dalam modul yang dibuat, terdapat kebutuhan-kebutuhan yang harus dipenuhi. Kebutuhan-kebutuhan ini perlu diperhatikan sebagai acuan dalam mengembangan modul Odoo SSO MyITS agar modul dapat berfungsi sesuai dengan ekspektasi. Kebutuhan-kebutuhan tersebut dibagi menjadi dua, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan berkaitan dengan fungsi dan proses dalam modul yang harus dipenuhi. Kebutuhan fungsional dari modul SSO MyITS yang akan dibangun dijelaskan pada Tabel 3.1 sebagai berikut.

Tabel 3.1 Kebutuhan Fungsional Modul SSO MyITS

Kode Kebutuhan	Deskripsi Kebutuhan
FR-01	Menerima *request* pengguna untuk *login* melalui SSO MyITS
FR-02	Menyimpan *query* dan lokasi halaman pada *browser* sebelum *login* dan menampilkan kembali saat berhasil *login*
FR-03	*Request* autorisasi kepada SSO MyITS
FR-04	*Request* JWT kepada SSO MyITS
FR-05	*Request* informasi pengguna kepada SSO MyITS
FR-06	Menerima dan memproses *respond* dari SSO MyITS berdasarkan *request* yang diberikan sebelumnya
FR-07	Mengauntentikasi pengguna
FR-08	Penyimpanan dan manajamen SSO MyITS pada tiap akun Odoo pengguna
FR-09	Penyimpanan dan manajemen konfigurasi *provider* SSO MyITS pada sistem Odoo

FR-10	Penyimpanan informasi pengguna dari SSO MyITS ke basis data *user* pada sistem Odoo

3.2.2 Kebutuhan Non-Fungsional

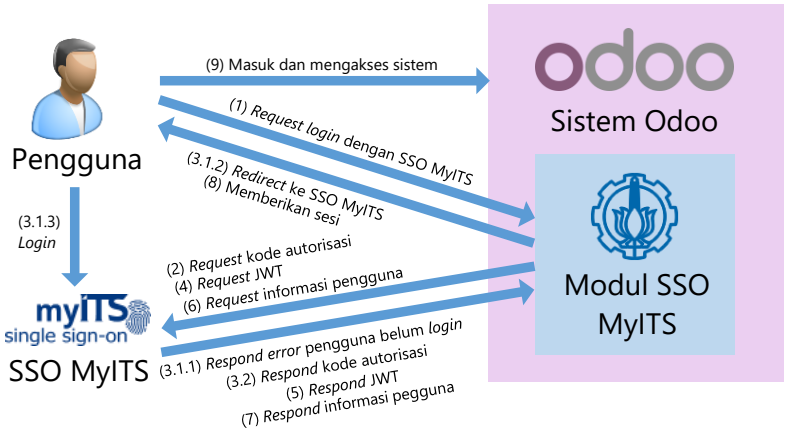
Kebutuhan non-fungsional merupakan kebutuhan yang berkatan dengan batasan layanan atau fungsi serta karakteristik dari modul. Kebutuhan non-fungsional dari modul SSO MyITS yang akan dibangun dijelaskan pada Tabel 3.2 sebagai berikut.

Tabel 3.2 Kebutuhan Non-Fungsional Modul SSO MyITS

Kode Kebutuhan	Deskripsi Kebutuhan
NFR-01	*Login* dengan SSO MyITS dapat diakses pengguna di halaman `login` pada sistem Odoo
NFR-02	Pengaturan modul dan pengguna SSO MyITS dapat diakses oleh Administrator
NFR-03	Modul dapat berjalan di jaringan ITS maupun di luar ITS
NFR-04	Modul dapat diakses lebih dari satu pengguna SSO MyITS
NFR-05	Modul dapat beroperasi 24 jam sehari

3.3 Perancangan Sistem

Dalam rancangan sistem SSO MyITS yang akan dibangun, pengguna, server Odoo, dan *provider* SSO MyITS akan saling terhubung saat melakukan autentikasi. Interaksi antara pengguna, sistem Odoo, dan SSO MyITS dapat digambarkan seperti pada Gambar 3.1 sebagai berikut.



Gambar 3.1 Interaksi Antara Pengguna, Sistem Odoo, dan SSO MyITS

Penjelasan dari interaksi antara pengguna, Sistem Odoo, serta SSO MyITS pada Gambar 3.1 adalah sebagai berikut :

1. Pengguna mengirim *request* untuk *login* ke Sistem Odoo melalui SSO MyITS.
2. Sistem Odoo mengirim *request* kode autorisasi ke SSO MyITS

3.1. Jika pengguna belum login di SSO MyITS :

- 3.1.1. SSO MyITS merespon kode *error* pengguna belum *login* di SSO MyITS kepada Sistem Odoo.
- 3.1.2. Sistem Odoo melakukan *redirect* pengguna ke SSO MyITS untuk melakukan *login*.
- 3.1.3. Pengguna melakukan *login* di SSO MyITS. Jika pengguna berhasil *login*, lanjut ke tahap 3.2.

3.2. Jika pengguna sudah *login* di SSO MyITS, SSO MyITS memberikan kode autorisasi kepada Sistem Odoo.

4. Sistem Odoo mengirim *request* JSON Web Token (JWT) kepada SSO MyITS menggunakan kode autorisasi
5. SSO MyITS memberikan JWT kepada Sistem Odoo

6. Setelah JWT divalidasi, Sistem Odoo mengirim *request* informasi pengguna ke SSO MyITS menggunakan *access token* dari JWT.
7. SSO MyITS memberikan informasi pengguna kepada Sistem Odoo.
8. Setelah memverifikasi pengguna menggunakan informasi pengguna dari SSO MyITS, Sistem Odoo memberikan sesi kepada pengguna.
9. Pengguna dapat masuk dan menggunakan Sistem Odoo.

Interaksi antara modul sebagai *client* SSO MyITS dan *provider* SSO MyITS akan mengikuti protokol OpenID Connect. Setiap *server* OpenID Connect perlu mempublikasi konfigurasinya dengan suatu mekanisme yang disebut OpenID Connect Discovery. Konfigurasi OpenID Connect dari SSO MyITS dapat diakses di https://my.its.ac.id/.well-known/openid-configuration. Konfigurasi OpenID Connect diberikan dalam bentuk JSON. Konfigurasi yang diberikan saat Tugas Akhir ini dibuat dijelaskan dalam Tabel 3.3 sebagai berikut.

Tabel 3.3 Konfigurasi OpenID Connect pada SSO MyITS Saat Tugas Akhir Ini Dibuat

Field	*Value*
`issuer`	`https:\/\/my.its.ac.id`
`authorization_endpoint`	`https:\/\/my.its.ac.id\/authorize`
`token_endpoint`	`https:\/\/my.its.ac.id\/token`
`userinfo_endpoint`	`https:\/\/my.its.ac.id\/userinfo`
`revocation_endpoint`	`https:\/\/my.its.ac.id\/revoke`
`end_session_endpoint`	`https:\/\/my.its.ac.id\/signout`

check_session_iframe	https:\/\/my.its.ac.id\/check_session_iframe
jwks_uri	https:\/\/my.its.ac.id\/.well-known\/jwks.json
response_types_supported	code
subject_types_supported	public
id_token_signing_alg_values_supported	RS256
token_endpoint_auth_methods_supported	client_secret_basic, client_secret_post
claims_supported	-
frontchannel_logout_supported	1
frontchannel_logout_session_supported	1
backchannel_logout_supported	1
backchannel_logout_session_supported	1

Saat pengguna meminta *request* untuk melakukan *login* melalui SSO MyITS, modul akan melakukan permintaan autorisasi dengan cara *redirect* ke alamat `authorization_endpoint` dan memberikan beberapa *query*. *Query* tersebut akan digunakan *provider* SSO MyITS sebagai parameter mengenai identitas, status, dan *scope* dari informasi yang diminta oleh *client* SSO MyITS. Macam-macam *query* yang diberikan dijelaskan dalam Tabel 3.4 sebagai berikut.

Tabel 3.4 *Query* yang Diberikan Saat Melakukan *Authorization Request* [16].

Query	Deskripsi
`prompt`	Parameter mengenai hal yang diminta untuk dilakukan kepada *provider*, nilainya dapat berupa `none` atau `login`
`response_type`	Jenis respon yang diberikan oleh *provider*.
`redirect_uri`	Alamat umpan balik dimana respon akan diberikan.
`client_id`	ID unik dari *client* yang terdaftar di SSO
`nonce`	Sebuah parameter *string* acak sebagai parameter sesi dalam *ID token* untuk mencegah *replay attack*.
`state`	Sebuah parameter *string* acak yang digunakan untuk mengidentifikasi *state* antara *request* dan *response* untuk mencegah *Cross-Site Request Forgery* (CSRF).
`scope`	Parameter megenai informasi-informasi yang diberikan *provider* berkaitan dengan *access token* yang didapatkan *client*. Minimal `openid.`

Jenis-jenis informasi yang diminta *client* diatur dalam *scope*. Tiap *client* memiliki *client ID*, *client secret*, dan akses ke beberapa *scope*, yang diberikan oleh administrator SSO MyITS. Dalam Tugas Akhir ini, *scope* yang digunakan dijelaskan di dalam Tabel 3.5 sebagai berikut.

Tabel 3.5 *Scope* dan Informasi yang Diberikan

Scope	Informasi yang diberikan
`openid`	ID dari pengguna berupa kode unik (`sub`)

email	Email utama (email), status verifikasi email utama (email_verified), email alternatif (alternate_email), stastus verifikasi email alternative (alternate_email_verified)
phone	Nomor telepon (phone), status verifikasi nomor telepin (phone_verified)
profile	Alamat gambar profil (picture), status kunci akun SSO (locked), nama panjang (name), bahasa utama (locale), jenis kelamin (gender), zona waktu (zoneinfo), stasus aktivasi akun (enabled), waktu terakhir profil akun diperbaharui (updated_at), tanggal lahir (birthdate), nomor identitas pengguna (reg_id), status suspensi akun (suspended), status pernah suspense (has_suspended), nama panggilan (nickname)

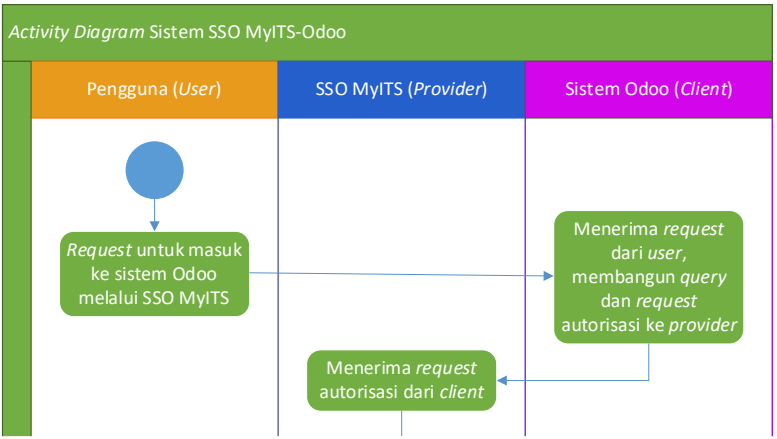
Saat melakukan *request* autorisasi pertama kali, nilai prompt ditentukan sebagai None. Jika pengguna belum melakukan *login* di SSO MyITS, maka *provider* akan memberikan respon dalam bentuk *request GET* bersama dengan kode error (error) dan deskripsi error (error_description) bahwa login diperlukan (login_required) ke redirect_uri. Setelah itu *client* akan meminta *request* autorisasi kembali dengan nilai prompt ditentukan sebagai login. Setelah itu pengguna akan diarahkan ke halaman authorization_endpoint untuk melakukan *login.*

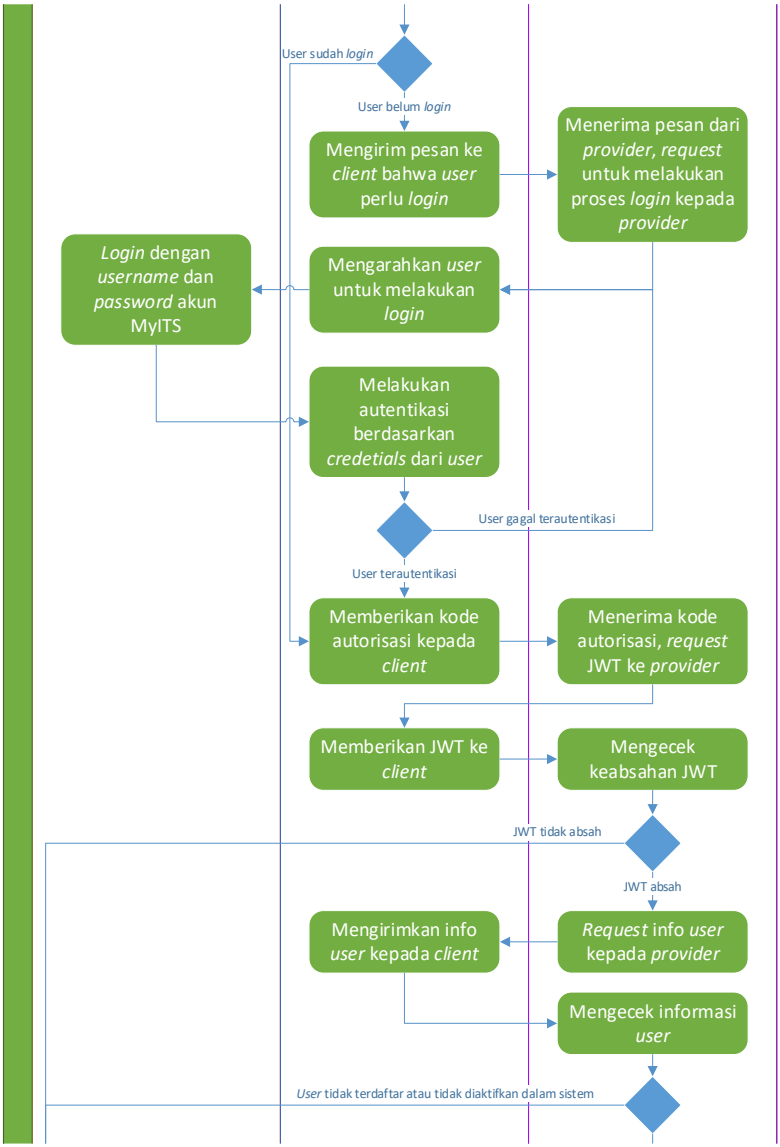
Jika pengguna berhasil *login*, maka *provider* akan melakukan *redirect* ke redirect_uri bersama dengan respon berupa kode autorisasi yang digunakan *client* untuk meminta JWT. *Client* menerima kode autorisasi dari *provider* SSO MyITS. Kemudian c*lient* melakukan *request* JWT dengan memberikan kode autorisasi, *client id*, *client secret*, alamat respon, dan metode autentikasi yang digunakan kepada *token endpoint* SSO MyITS.

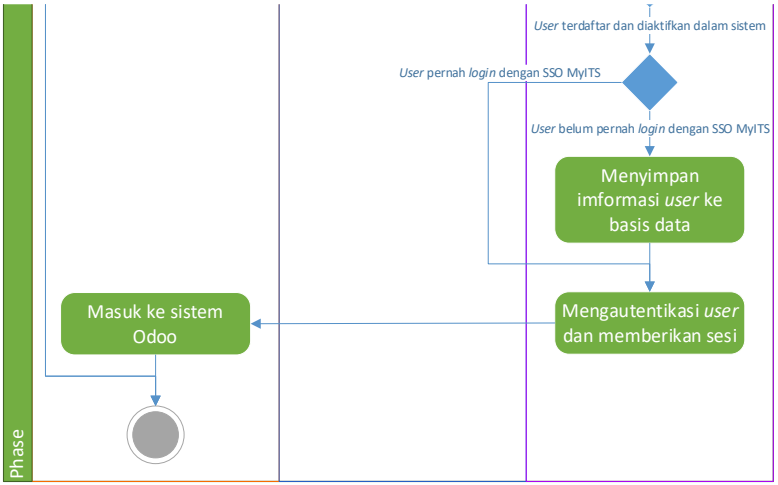
Dalam Tugas Akhir ini, metode autentikasi yang digunakan pada *token endpoint* adalah `client_secret_basic`.

Jika *client* berhasil mendapatkan JWT, client akan memverifikasi ID Token pada JWT tersebut. ID Token diverifikasi dengan memastikan apakah *signature*, *provider*, *client*, *nonce* serta masa berlaku JWT sesuai. Jika JWT terverifikasi, *client* akan mengirim *request* informasi mengenai pengguna yang bersangkutan dengan memberikan *access token* dari JWT dan skema kepada *userinfo endpoint* SSO MyITS. Jika *request* berhasil, *provider* akan memberikan informasi mengenai pengguna sesuai dengan *scope* yang diminta sebelumnya. Informasi mengenai pengguna tersebut digunakan oleh *client* untuk mengautentikasi pengguna. Jika pengguna terautentikasi pada *client* (sistem Odoo), maka pengguna akan diberikan sesi dan diarahkan unuk masuk ke dalam sistem.

Alur proses dalam sistem autentikasi melalui SSO MyITS seperti yang dijelaskan sebelumnya dapat digambarkan pada *Activity Diagram* seperti Gambar 3.2 berikut.

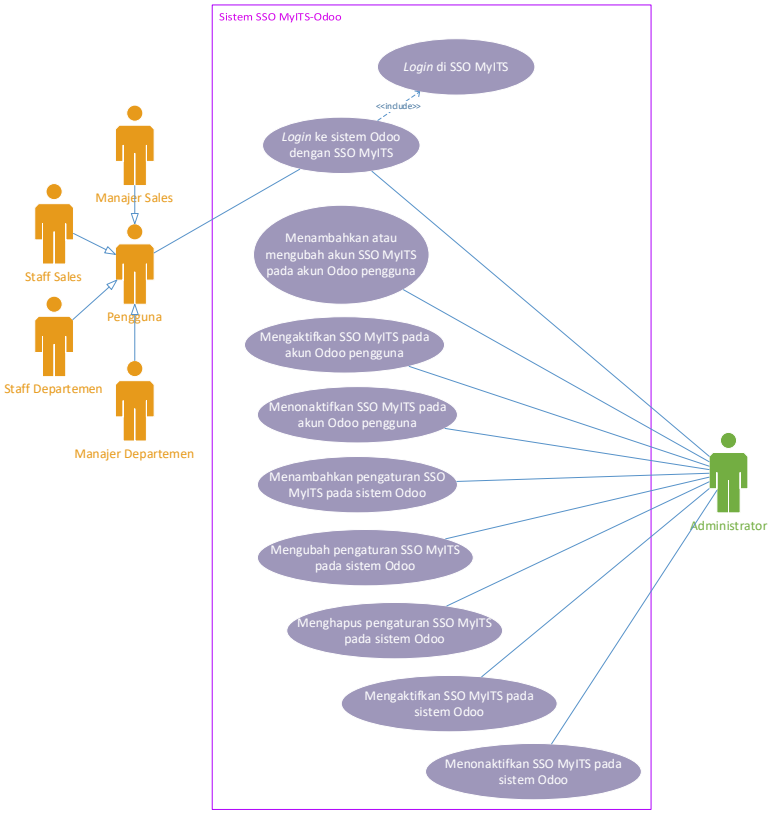



User sudah *login*
User belum *login*
Mengirim pesan ke *client* bahwa *user* perlu *login*
Menerima pesan dari *provider*, *request* untuk melakukan proses *login* kepada *provider*
Login dengan *username* dan *password* akun MyITS
Mengarahkan *user* untuk melakukan *login*
Melakukan autentikasi berdasarkan *credetials* dari *user*
User gagal terautentikasi
User terautentikasi
Memberikan kode autorisasi kepada *client*
Menerima kode autorisasi, *request* JWT ke *provider*
Memberikan JWT ke *client*
Mengecek keabsahan JWT
JWT tidak absah
JWT absah
Mengirimkan info *user* kepada *client*
Request info *user* kepada *provider*
Mengecek informasi *user*
User tidak terdaftar atau tidak diaktifkan dalam sistem



Gambar 3.2 *Activity Diagram* Sistem SSO MyITS – Odoo

Dalam sistem SSO MyITS – Odoo ini, terdapat beberapa aktor yang berperan, yaitu pengguna dan administrator. Aktor pengguna dapat dibagi menjadi Staff Departemen, Manajer Departemen, Staff Sales, dan Manajer Sales. Aktor-aktor pengguna ini memiliki nomor identitas yang terdaftar di SSO MyITS dan hanya bisa mengakses Odoo jika aktor-aktor pengguna tersebut didaftarkan oleh aktor administrator/pihak yang diberi wewenang. Setiap aktor yang terlibat dijelaskan perannya pada *Use Case Diagram* yang ditunjukkan pada Gambar 3.3 berikut.

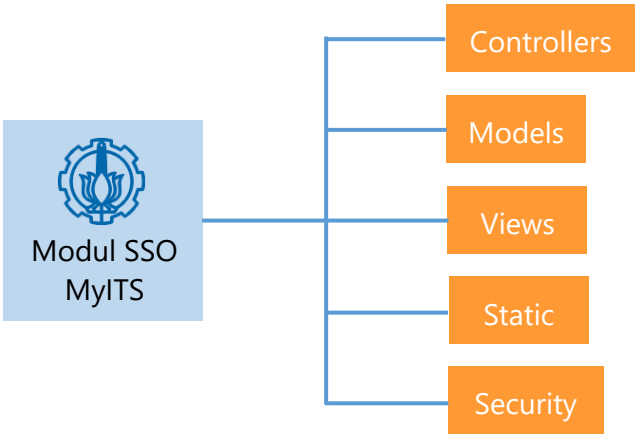


Gambar 3.3 *Use Case Diagram* Dari Sistem SSO MyITS – Odoo

3.4 Perancangan Modul

Perancangan modul dilakukan berdasarkan fungsi dan proses bisnis dalam sistem yang perlu dijalankan oleh modul. Perancangan modul ditujukan agar mempermudah pengembangan modul, memberi gambaran mengenai modul yang akan dibangun, serta mempermudah jika terdapat pembaharuan.

Modul Odoo SSO MyITS dibangun berdasarkan spesifikasi Odoo versi 10 yang saat ini digunakan di ERP Odoo Logistik Terpusat ITS. Modul yang dirancang dibagi menjadi beberapa bagian mengikuti struktur standar modul Odoo 10. Desain dari modul tersebut dibagi menjadi bagian *controllers*, *models*, *views*, *static*, dan *security*. Bagian-bagian dari modul dijabarkan pada Gambar 3.4 sebagai berikut



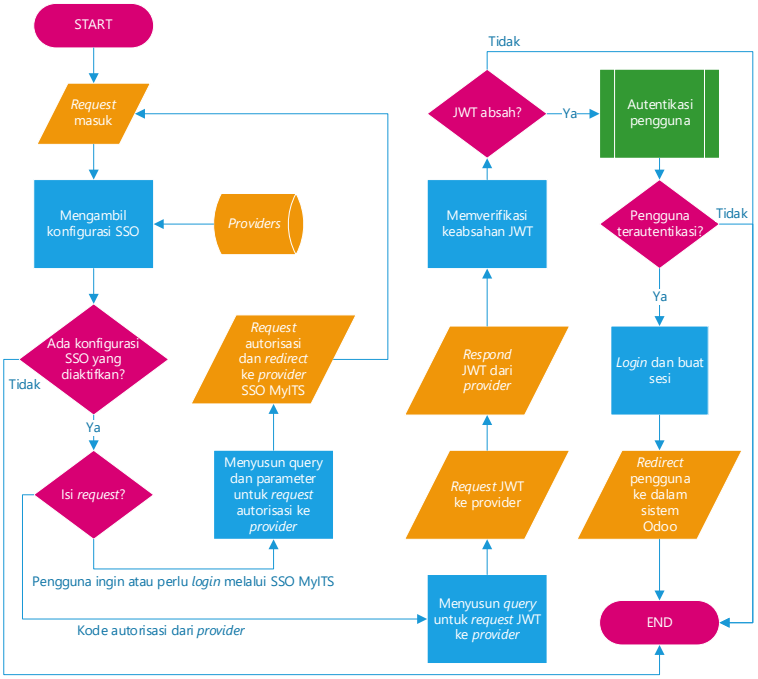
Gambar 3.4 Bagian-bagian Dari Rancangan Modul

3.4.1 *Controllers*

Bagian *controllers* adalah bagian yang mengatur interaksi antara pengguna dan *provider* SSO MyITS. Bagian ini juga melakukan verifikasi keabsahan data yang diterima dari *provider*. *Controllers* memiliki fungsi sebagai berikut:

- Menerima dan memproses *request* dari pengguna untuk *login* dengan SSO MyITS
- Mengirim *request* autorisasi dan JWT ke *provider* SSO MyITS
- Menerima, memverifikasi dan memproses *respond* dari *provider*
- Menjalankan prosedur autentikasi pada sistem Odoo

Algoritma dari *controller* dijabarkan menggunakan *flowchart* yang ditunjukkan pada Gambar 3.5 sebagai berikut.



Gambar 3.5 *Flowchart* Dari Algoritma *Controller*

Jika *controller* menerima *request* dari pengguna untuk *login* melalui SSO MyITS, *controller* mengirim *request* autorisasi ke *provider*. Jika pengguna belum login di SSO MyITS, maka pengguna diarahkan untuk melakukan *login* dengan akun MyITS. Jika berhasi *login*, maka *controller* akan menerima kode autentikasi yang kemudian digunakan untuk mendapatkan JWT. JWT tersebut kemudian diverifikasi dan digunakan untuk autentikasi pengguna. Prosedur autentikasi pengguna akan ditempatkan di model. Jika pengguna berhasil terautentikasi, maka *controller* akan melakukan *login* dan meberikan sesi kepada

pengguna, kemudian mengarahkan pengguna masuk ke dalam sistem Odoo.

3.4.2 *Models*

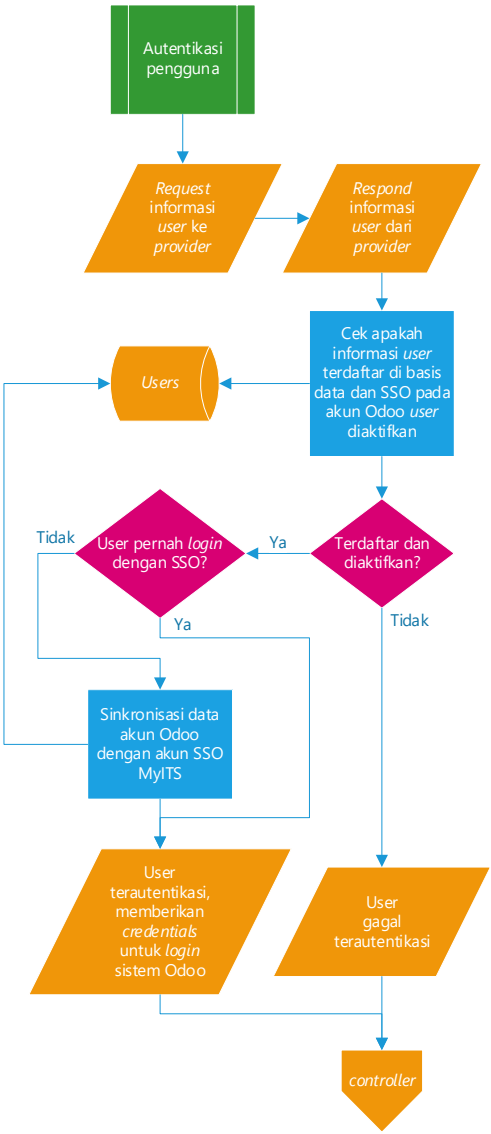
Bagian *models* adalah bagian dimana struktur data dalam modul didefinisikan. *Models* juga dapat ditambahkan fungsi kustom untuk manajemen data maupun hal lainnya sesuai dengan kebutuhan. Dalam modul SSO MyITS yang dibangun, digunakan dua model yaitu model *user* dan model *provider*.

3.4.2.1 Model *User*

Model *user* mendefinisikan struktur data *user* yang berkaitan dengan proses bisnis dan fungsionalitas dalam sistem SSO MyITS – Odoo. Model ini akan melengkapi model *user* bawaan yang sudah ada di dalam sistem Odoo dengan cara *inheritance*. Setidaknya data *user* dan formatnya yang perlu didefinisikan adalah sebagai berikut:

- Nomor identitas pengguna yang terdaftar di SSO MyITS (*String*)
- ID unik dari SSO MyITS sebagai identitas akun (*String*)
- Status *login* pertama *user* melalui SSO MyITS (*Boolean*)
- Status aktivasi SSO MyITS pada *user* (*Boolean*)

Selain pendefinisian data *user*, model ini juga memerlukan beberapa fungsi tambahan untuk proses autentikasi *user* yang diperlukan oleh *controller* dan sinkronisasi data *user* dari akun MyITS dengan akun Odoo. Algoritma dalam prosedur autentikasi *user* dalam model *user* dijabarkan dalam bentuk *flowchart* pada Gambar 3.6 sebagai berikut.



Gambar 3.6 *Flowchart* Dari Algoritma Autentikasi Pengguna pada Model *User*

Saat prosedur autentikasi dijalankan, *request* informasi *user* dikirimkan ke *provider* SSO MyITS. Setelah informasi *user* didapatkan, nomor identitas dari informasi *user* yang diterima digunakan untuk pengecekan *user* pada basis data. Jika nomor identitas ditemukan dan SSO pada akun Odoo bersangkutan ditemukan diaktifkan, maka *user* terautentikasi dalam sistem Odoo. Jika *user* belum pernah login dengan SSO sebelumnya, maka data akun Odoo akan disinkronkan dengan data *user* yang diterima dari *provider* SSO MyITS. Setelah *user* berhasil terautentikasi, *credentials* diberikan dan proses dikembalikan ke *controller*.

3.4.2.2 Model *Provider*

Model *provider* mendefinisikan struktur data mengenai konfigurasi SSO MyITS pada sistem Odoo sebagai acuan dalam proses bisnis dan fungsionalitas sistem SSO MyITS – Odoo. Model ini akan menjadi model tersendiri dan khusus diterapkan pada modul SSO MyITS Odoo. Terdapat beberapa konfigurasi SSO MyITS yang perlu dsimpan pada basis data, yaitu :

- *Client ID* (*String*)
- *Client Secret* (*String*)
- URL *provider* (*String*)
- *Scope* yang digunakan (*String*)
- Status aktivasi konfigurasi (*Boolean*)

Hanya satu konfigurasi yang dapat diaktifkan dalam satu waktu. Konfigurasi ini digunakan selama proses autentikasi dijalankan di dalam *controller*. Jika tidak ada konfigurasi yang diaktifkan, dapat dikatakan SSO MyITS dalam sistem Odoo dimatikan.

3.4.3 *Views*

Bagian *views* adalah bagian yang memberikan dan mengatur tampilan UI (*User Interface*) dari modul. Pada modul SSO MyITS Odoo ini, *views* akan mengatur tampilan model *user* dan model

provider. Sebuah *view* juga akan ditambahkan pada halaman `login` Odoo untuk memudahkan akses pengguna yang ingin *login* melalui SSO MyITS.

3.4.3.1 *View User*

View untuk model *user* akan memberikan dan mengatur UI dari model *user*. Karena model *user* pada modul ini melakukan *inherit* ke model *user* bawaan sistem, *view* ini juga akan ditempatkan di dalam *view user* bawaan sistem. *View* ini akan ditampilkan dalam bentuk *form* di sebuah tab khusus SSO MyITS di tampilan *form* pengaturan *user*. Pengaturan *user* diakses oleh administrator di menu `Settings` > `Users` > `Users`. Tampilan dari *view* ini kurang lebih dapat digambarkan pada Gambar 3.7 berikut.

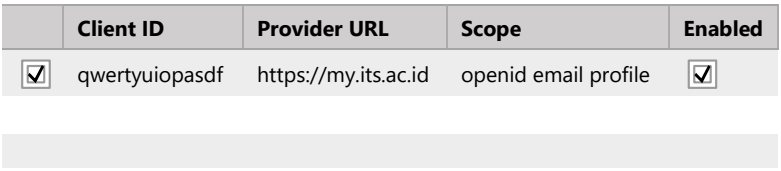
Gambar 3.7 Perkiraan Tampilan *View User*

Data yang ditampilkan adalah nomor identitas pengguna, status aktivasi SSO MyITS pada akun Odoo (`Enabled`) dan status *login* pertama *user* melalui SSO MyITS (`Verified`). ID unik dari SSO MyITS tidak ditampilkan karena data tersebut cukup sensitif dan digunakan dalam proses autentikasi.

3.4.3.2 *View Provider*

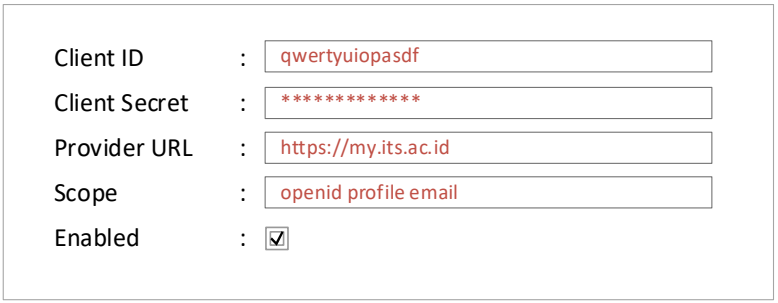
View untuk model *provider* akan memberikan dan mengatur tampilan UI dari model *provider*. *View* ini nantinya akan diberikan

menu sendiri yang diletakkan di `Settings` > `Users`. *View pirovider* akan bisa ditampilkan dalam bentuk *form* dan *tree*. Tampilan *tree* dari *view provider* kurang lebih dapat digambarkan seperti pada Gambar 3.8 berikut.



Gambar 3.8 Perkiraan Tampilan *Tree* pada *View Provider*

Sedangkan tampilan *form* dari *view provider* kurang lebih dapat digambarkan seperti pada Gambar 3.9 berikut.

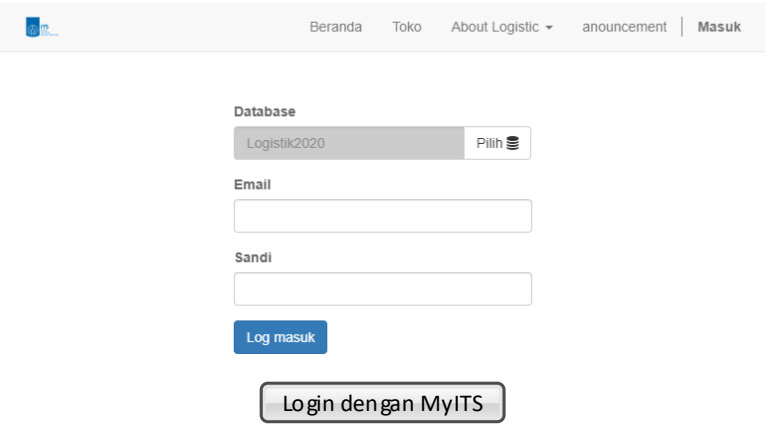


Gambar 3.9 Perkiraan Tampilan *Form* pada *View Provider*

Data yang ditampilkan di *tree view* adalah c*lient ID*, URL *provider*, *scope*, dan status aktivasi konfigurasi *provider* (`Enabled`). Pada tampilan *form view*, semua data yang ditampilkan di *tree view* juga ikut ditampilkan ditambah dengan *client secret*. *Client secret* diperlakukan seperti *password* pada tampilan *form*.

3.4.3.3 *View Login*

View login tidak terhubung dengan model, namun mendapatkan informasi mengenai konfigurasi *provider* melalui *controller*. *View login* berfungsi untuk mempermudah pengguna mengirim *request* ke modul SSO MyITS Odoo untuk *login* ke sistem Odoo melalui SSO MyITS. *View login* ini akan diletakkan di dalam *view login* bawaan sistem Odoo. Untuk mempermudah akses pengguna, *view* ini akan menambahkan sebuah tombol di halaman `login` yang jika ditekan akan mengirim *request login* melalui SSO MyITS kepada modul SSO MyITS Odoo. Tampilan dan lokasi tombol tersebut kurang lebih dapat digambarkan seperti pada Gambar 3.10 berikut.



Gambar 3.10 Perkiraan Tampilan dan Lokasi Tombol yang ditambahkan *View Login*

3.4.4 *Static*

Bagian *static* berfungsi untuk menyimpan *web assets* yang digunakan oleh modul. Contoh beberapa *asset* yang akan digunakan modul ini adalah ikon modul, deskripsi modul, file CSS, dan font yang digunakan CSS.

3.4.5 *Security*

Bagian *Security* berfungsi untuk mengatur pembatasan hak akses suatu model kepada grup pengguna tertentu. Pada modul SSO MyITS Odoo ini, *security* akan memberi batasas akses pada modul *provider* sehingga modul *provider* hanya dapat diakses oleh administrator.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini, dijelaskan berbagai hal mengenai lingkungan implementasi dimana Tugas Akhir ini dibuat, serta *tools* apa saja yang digunakan selama pengembangan modul SSO MyITS Odoo.

4.1 Lingkungan Implementasi

Lingkungan implementasi menjelaskan segala sesuatu mengenai kondisi dan konteks dimana Tugas Akhir ini dibuat. Lingkungan implementasi pada Tugas Akhir ini dapat dibagi menjadi tiga bagian, yaitu lingkungan implementasi perangkat keras, lingkungan implementasi perangkat lunak, dan lingkungan implementasi sistem.

4.1.1 Lingkungan Implementasi Perangkat Keras

Lingkungan implementasi perangkat keras menjelaskan keadaan-keadaan fisik dimana Tugas Akhir ini dibuat. Lingkungan implementasi perangkat keras pada Tugas Akhir ini dijelaskan pada Tabel 4.1 sebagai berikut.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat Keras	Detail
Komputer	ASUS X455LF
Prosesor	Intel® Core™ i5 5200U
VGA	Nvidia® GeForce® 930M
RAM	DDR3 4096MB
HDD	500 GB
Input Device	*Mouse, Keyboard*
Output Device	Monitor LCD 12.0 inch

4.1.2 Lingkungan Implementasi Perangkat Lunak

Lingkungan impementasi perangkat lunak menjelaskan hal-hal yang berkaitan dengan perangkat lunak yang terkait dan *tools* yang digunakan dalam pengembangan modul SSO MyITS Odoo. Lingkungan implementasi perangkat lunak pada Tugas Akhir ini dijelaskan pada Tabel 4.2 di bawah ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Perangkat Lunak	Detail
ERP	Odoo v10
Basis Data	PostgreSQL 10.5
Sistem Operasi	Linux Ubuntu 18.04
Python *library* tambahan	PyJWT, Requests
IDE	Sublime Text
Browser	Google Chrome, Chromium, Mozilla Firefox

4.1.3 Lingkungan Implementasi Sistem

Lingkungan inpementasi sistem menjelaskan konteks megenai keadaan sistem dimana proses pengembangan modul SSO MyITS Odoo dilakukan. Lingkungan inpementasi sistem dapat dijelaskan sebagai pada Tabel 4.3 dibawah ini.

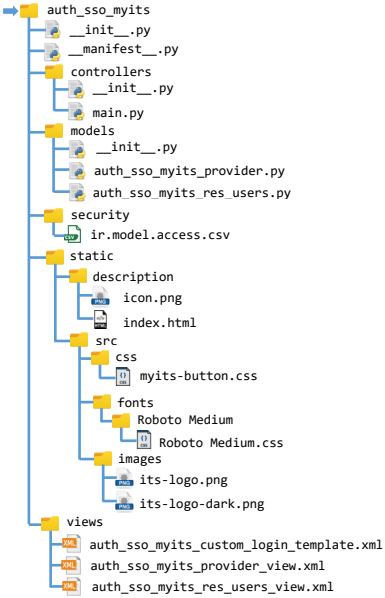
Tabel 4.3 Lingkungan Implementasi Sistem

Keadaan Sistem	Detail
URL *Client*	http://localhost:8069
Akses SSO MyITS pada *client*	http://localhost:8069/auth_sso_myits/signin
URL *Provider*	https://dev-my.its.ac.id , *provider* ini adalah *provider* khusus yang digunakan dalam

	pengembangan sistem-sistem di ITS yang berhubungan dengan MyITS. Secara fungsional sama dengan https://my.its.ac.id.
Scope yang disediakan *provider*	openid email integra phone profile resource

4.2 Implementasi Modul SSO MyITS Odoo

Pada bagian ini dijelaskan mengenai implementasi dari Modul SSO MyITS Odoo. Modul ini dibagi menjadi beberapa bagian sesuai dengan rancangan yang dilakukan sebelumnya, ditambahkan dengan insialisasi modul Modul ini diberikan nama teknikal `auth_sso_myits`. Struktur dari modul `auth_sso_myits` dapat dijelaskan seperti pada Gambar 4.1 sebagai berikut.



Gambar 4.1 Struktur Modul SSO MyITS Odoo

4.2.1 Implementasi Inisialisasi Modul

Implementasi inisialisasi modul terdiri dari `__init__.py` dan `__manifest__.py`. `__init__.py` berfungsi untuk mendefinisikan bagian *controllers* dan *models* dari modul ini. `__manifest__.py` berfungsi untuk mendefiniskan berbagai informasi mengenai modul ini. Implementasi `__init__.py` ditunjukkan pada lampiran Kode Sumber S.1, dan implementasi `__manifest__.py` ditunjukkan pada lampiran Kode Sumber S.2.

4.2.2 Implementasi *Controllers*

Implementasi c*ontrollers* terdiri dari `__init__.py` dan `main.py`. `__init__.py` berfungsi untuk mendefinisikan *controller* (`main.py`) , sedangkan `main.py` berfungsi menerima *request* di alamat akses SSO MyITS pada *client* (http://<url_client>/auth_sso_myits/signin) serta menjalankan fungsi-fungsi dari *controllers*. Selain itu `main.py` juga mengatur informasi yang ditampilkan pada halaman `login` dengan cara melakukan *override method* dari modul `web` bawaan Odoo.

Controllers mengirim *request* JWT kepada *provider* SSO MyITS dengan menggunakan *method* `post` dari *Python library* `requests`. Setelah menerima *respond* berupa JWT dari *provider*, JWT tersebut kemudian diverifikasi *signature*-nya menggunakan *method* `__verify_jwt_signature`. *Method* tersebut menggunakan *method* yang ada di *Python library* `PyJWT`. Kemudian *provider, client, nonce,* dan masa berlaku JWT diverifikasi pada method `__verify_jwt_claims`. Implementasi `__init__.py` dari *controllers* ditunjukkan pada lampiran Kode Sumber S.3, sedangkan implementasi `main.py` dari *controllers* ditunjukkan pada lampiran Kode Sumber S.4.

4.2.3 Implementasi *Models*

Implementasi *models* bertujuan untuk mendefinisikan struktur data serta menjalankan beberapa fungsi seperti yang telah

dirancang sebelumnya. Implementasi *models* terdiri dari `auth_sso_myits_provider.py` sebagai implementasi dari model *provider*, `auth_sso_myits_res_users.py` sebagai implementasi dari model *user* dan `__init__.py` untuk menginisialisasi kedua model tersebut. Implementasi `__init__.py` dari model ditunjukkan pada lampiran Kode Sumber S.5.

4.2.3.1 Implementasi Model *User*

Implementasi model *user* ditempatkan pada file `auth_sso_myits_res_users.py`. Model ini mendefinisikan data yang berkaitan dengan SSO MyITS pada tiap *user*. Karena data ini berkaitan dengan *user*, maka model ini akan melakukan *inherit* dari model `res.users` yang menjadi model *user* bawaan sistem. Model ini juga memanajemen data *user* serta memproses autentikasi dengan mengunakan beberapa *method*. Data pada model ini didefinisikan sebagai *field* yang dijelaskan pada Tabel 4.4 sebagai berikut.

Tabel 4.4 *Field* dari Model *User*

Field	Tipe	Detail
`auth_sso_myits_identity_no`	`Char`	Nomor identitas *user* yang terdaftar di SSO MyITS
`auth_sso_myits_openid_identifier`	`Char`	ID unik dari SSO MyITS sebagai identitas akun
`auth_sso_myits_enabled`	`Boolean`	Status aktivasi SSO MyITS pada *user*
`auth_sso_myits_verified`	`Boolean`	Status *login* pertama *user* melalui SSO MyITS

Pada `auth_sso_myits_res_users.py`, juga terdapat beberapa *method* tambahan untuk proses autentikasi dan penyimpanan data *user* dari SSO MyITS ke *user* Odoo. *Method* ini

akan dijalankan saat *controller* melakukan proses autentikasi. Selain itu, model akan memastikan tiap *user* memiliki `auth_sso_myits_identity_no` dan `auth_sso_myits_openid_identifier` yang unik untuk mencegah kerancuan data serta membatasi akses.

Model user mengirim *request* informasi *user* kepada *provider* SSO MyITS menggunakan *method* `post` dari *Python library* `requests` yang ditempatkan di *method* `__request_user_info`. Informasi *user* kemudian diverifikasi dan disimpan menggunakan *method* `__auth_login`. Implementasi a`uth_sso_myits_res_users.py` dari model *provider* ditunjukkan pada lampiran Kode Sumber S.6.

4.2.3.2 Implementasi Model *Provider*

Implementasi model *provider* ditempatkan pada file a`uth_sso_myits_provider.py`. Model ini mendefinisikan data konfigurasi SSO MyITS pada *client* serta memanajemen data tersebut mengunakan beberapa *method*. Data didefinisikan sebagai *field* yang dijelaskan pada Tabel 4.5 sebagai berikut.

Tabel 4.5 *Field* dari Model *Provider*

Field	Tipe	Detail
`client_id`	`Char`	*Client ID* milik *clent*
`client_secret`	`Char`	*Client Secret* milik *client*
`provider_url`	`Char`	URL dari *provider*
`scope`	`Char`	*Scope* yang digunakan *client*
`enabled`	`Boolean`	Status aktivasi konfigurasi

Pada a`uth_sso_myits_provider.py`, juga terdapat *method* tambahan untuk memastikan hanya satu konfigurasi yang aktif pada satu waktu. Ketike *create* dan *write* pada model

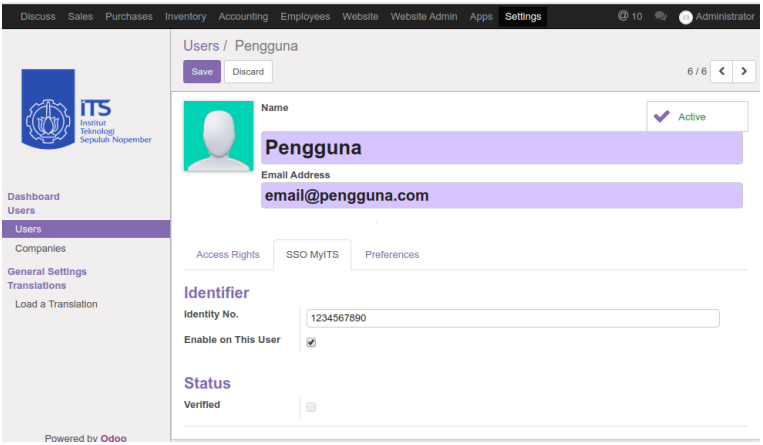
dijalankan, *method* tersebut akan memeriksa jika ada lebih dari satu konfigurasi yang aktif. Jika ditemukan, maka konfigurasi yang lama aktif akan dimatikan menggunakan *method* `_disable_active_provider`. Selain itu, *method* tersebut juga memastikan beberapa *scope* tertentu yang dibutuhkan modul telah dimasukkan. Implementasi a`uth_sso_myits_provider.py` dari model *provider* ditunjukkan pada lampiran Kode Sumber S.7.

4.2.4 Implementasi *Views*

Implementasi *views* ditujukan untuk memberikan tampila dari model dan menambahkan UI pada halaman `login` agar mempermudah akses pengguna untuk mengakses *login* melalui SSO MyITS. Implementasi *views* terdiri dari `auth_sso_myits_res_users_view.xml` sebagai implementasi *view user*, `auth_sso_myits_provider_view.xml` sebagai implementasi *view provider,* dan `auth_sso_myits_custom_login_template.xml` sebagai implementasi *view login*.

4.2.4.1 Implementasi *View User*

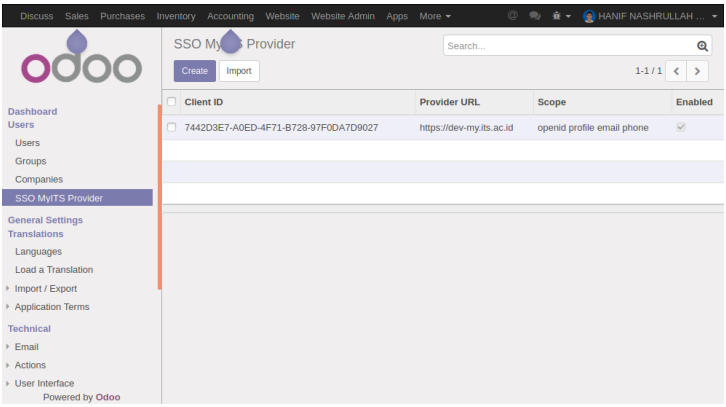
Implementasi dari *view user* ditujukan untuk memberikan tampilan dari model *user* berdasarkan desain yang dilakukan sebelumnya. *View user* diimplementasikan di file `auth_sso_myits_res_users_view.xml` yang isinya ditunjukkan pada lampiran Kode Sumber S.8. Tampilan dari *view user* ditunjukkan pada Gambar 4.2.



Gambar 4.2 Tampilan dari *View User*

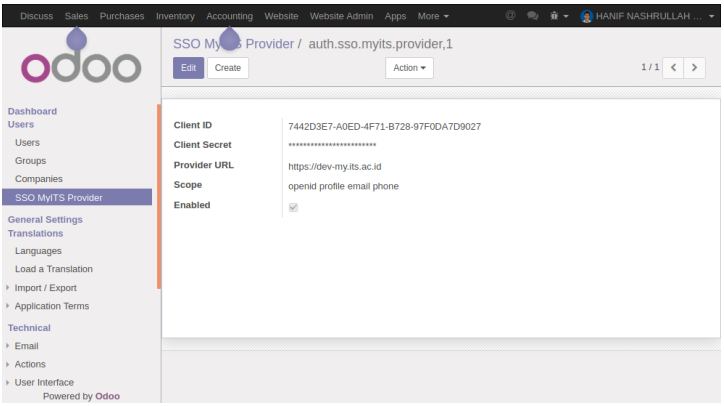
4.2.4.2 Implementasi *View Provider*

Implementasi dari *view provider* ditujukan untuk memberikan tampilan *tree* dan *form* dari model *provider* berdasarkan desain yang dilakukan sebelumnya. *View provider* diimplementasikan di file `auth_sso_myits_provider_view.xml` yang isinya ditunjukkan pada lampiran Kode Sumber S.9. *View provider* dapat diakses dengan mengaktifkan *debug mode* pada `Settings`. Tampilan *tree* dari *view provider* ditunjukkan pada Gambar 4.3.



Gambar 4.3 Tampilan *Tree* Dari *View Provider*

Sedangkan tampilan *form* dari *view provider* ditunjukkan pada Gambar 4.4.

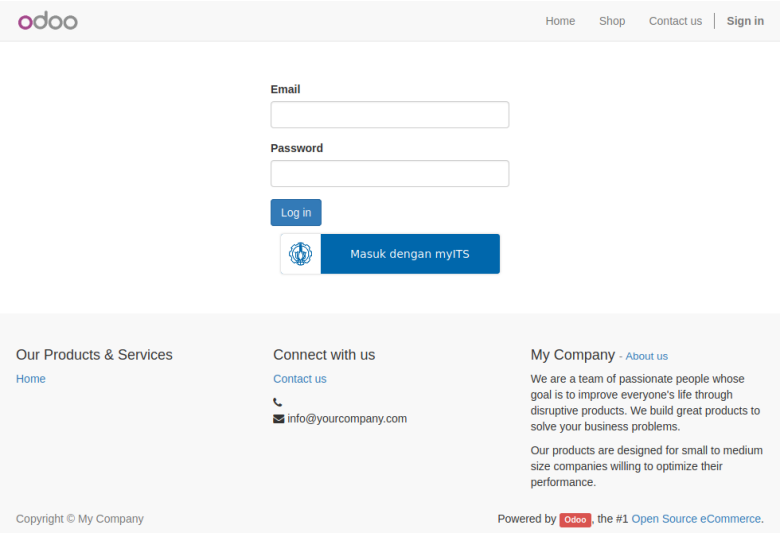


Gambar 4.4 Tampilan *Form* Dari *View Provider*

4.2.4.3 Implementasi *View Login*

Implementasi dari *view login* ditujukan untuk memberikan tampilan UI berupa tombol pada halaman `login` untuk

mempermudah pengguna mengakses SSO MyITS pada *client* di alamat http://<url_client>/auth_sso_myits/signin. *View login* diimplementasikan di file `auth_sso_myits_custom_login_template.xml` yang isinya ditunjukkan pada lampiran Kode Sumber S.10. Tampilan tombol dan lokasinya ditunjukkan pada Gambar 4.5.



Gambar 4.5 Tampilan dari *View Login*

4.2.5 Implementasi *Static*

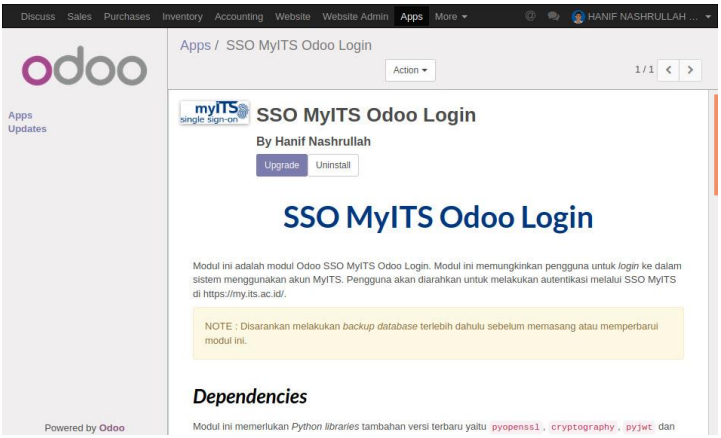
Implementasi *static* di modul ini ditujukan untuk menyediakan hal-hal yang berkaitan dengan modul, seperti deskripsi modul dan *web assets.* Deskripsi dari modul ditempatkan pada *folder* `description`, sedangkan *web assets* ditempatkan pada folder `src`.

Pada folder `description`, terdapat logo modul yang diberi nama `icon.png`, dan deskripsi modul yang diimplementasikan di

`index.html`. Ikon modul yang digunakan ditampilkan sebagai pada Gambar 4.6.

Gambar 4.6 Ikon dari Modul SSO MyITS Odoo

Implementasi dari `index.html` ditunjukkan pada lampiran Kode Sumber S.11. Penampilan dari deskripsi modul ditunjukkan pada menu `Apps` seperti pada Gambar 4.7 berikut.



Gambar 4.7 Tampilan Deskripsi Dari Modul

Selain itu, terdapat *style* yang digunakan oleh *view login. Style* ini berguna untuk mengatur desain tombol *login* melalui SSO MyITS di halaman `login`. *Style* ini diimplementasikan di `src/css/myits-button.css` dan isinya ditunjukkan pada lampiran Kode Sumber S.12. CSS tersebut juga menggunakan gambar yang ditempatkan di di `src/images` dan desain *font* Roboto Medium yang ditempatkan di `src/fonts/Roboto Medium`.

4.2.6 Implementasi *Security*

Implementasi *security* berguna untuk membatasi akses pada model tertentu kepada grup *user* tertentu. Pada modul ini, *security* membatasi model *provider* (`auth_sso_myits_provider`) agar hanya dapat diakses oleh administrator. *Security* diimplementasikan pada file `ir.model.access.csv`. Implementsasi dari `ir.model.access.csv` ditunjukkan pada lampiran Kode Sumber S.13.

4.3 Pemasangan dan Penggunaan Modul

Untuk menggunakan modul SSO MyITS Odoo, modul perlu dipasang terlebih dahulu. Setelah modul berhasil terpasang, administrator dapat mengatur konfigurasi *provider* dan *user*.

4.3.1 Pemasangan Modul

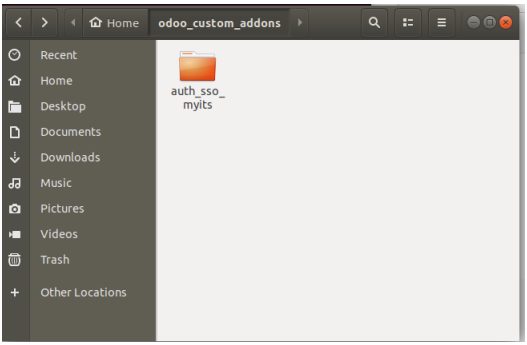
Langkah-langkah dalam pemasangan modul SSO MyITS Odoo adalah sebagai berikut:

1. Pastikan Python *library* `pyopenssl`, `cryptography`, `pyjwt`, dan `requests` terpasang dan terbaharui. Pemasangan dapat menggunakan `pip`.

```
anip@anip:/opt/odoo/odoo-10.0$ pip install -U pyopenssl cryptography pyjwt requests
```

Gambar 4.8 Pemasangan dan Pembaharuan Python *Library*

2. Letakkan modul di sebuah direktori khusus modul Odoo.



Gambar 4.9 Meletakkan Modul Di Direktori Model Khusus Odoo

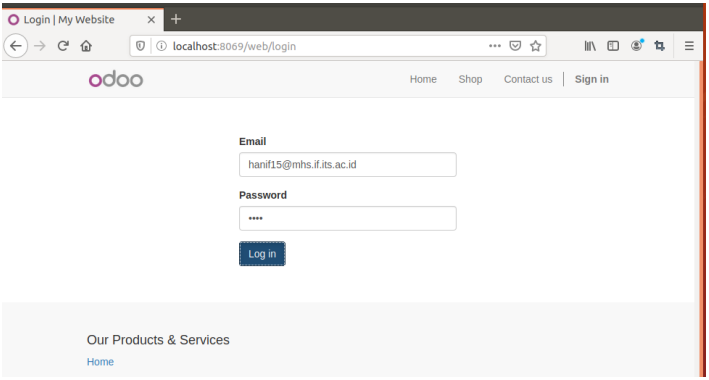
3. Buka file konfigurasi Odoo, lalu pastikan direktori dimana modul SSO MyITS Odoo disimpan terdaftar di `addons_path`. Jika tidak ada, maka dapat ditambahkan dengan memberi koma (,) lalu alamat direktori modul tersebut.

Gambar 4.10 File Konfigurasi Odoo

4. Jalankan Odoo dengan file konfigurasi tersebut.

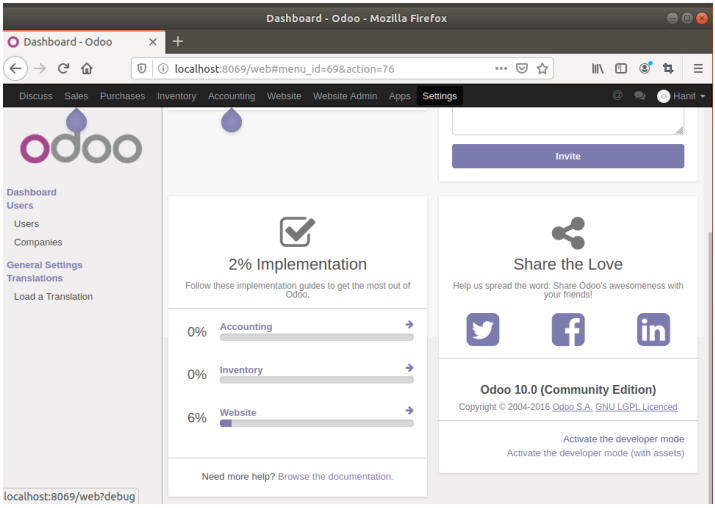
Gambar 4.11 Menjalankan Odoo Dengan Konfigurasi

5. Buka *browser*, buka halaman `login` lalu *login* sebagai administrator.



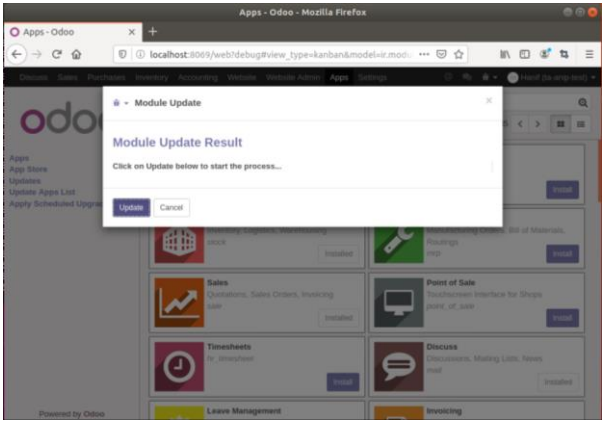
Gambar 4.12 *Login* Sebagai Administrator

6. Aktifkan mode `debug` dengan mengakses `Settings` > `Activate the developer mode`.



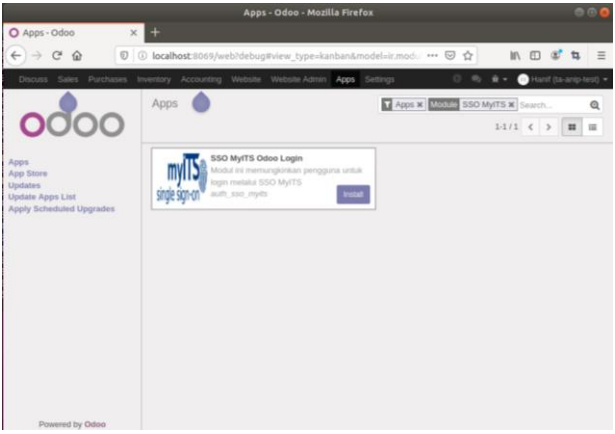
Gambar 4.13 Aktifkan Mode Debug

7. Buka `Apps`, kemudian lakukan pembaharuan daftar modul dengan menekan `Update Apps List`, lalu pada *popup* `Module Updates` tekan `Update`.



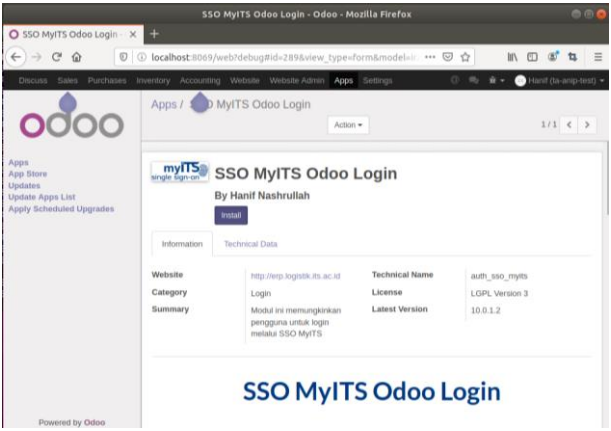
Gambar 4.14 *Update* Daftar *Modul*

8. Setelah selesai pembaharuan daftar modul, cari modul SSO MyITS, lalu pilih modul tersebut.



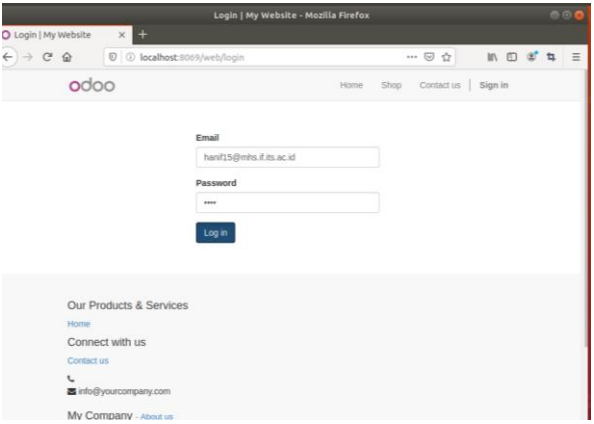
Gambar 4.15 Cari modul Odoo SSO MyITS

9. Perhatikan deskripsi dari modul, kemudian tekan `install`.



Gambar 4.16 *Install* Modul SSO MyITS

10. Setelah berhasil terpasang, sesi semua *user* akan dihapus. Jika ingin melakukan pengaturan selanjutnya, kembali *login* menggunakan akun administrator.



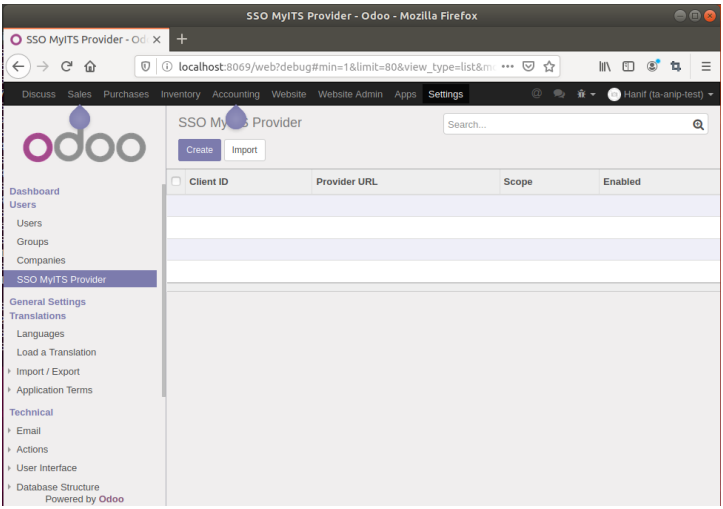
Gambar 4.17 Login Kembali Sebagai Administrator

4.3.2 Pengaturan *Provider*

Langkah-langkah dalam konfigurasi *provider* pada modul SSO MyITS Odoo adalah sebagai berikut:

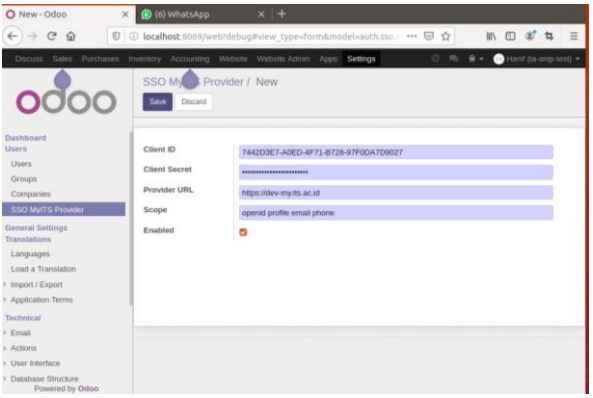
1. *Login* menggunakan akun administrator, seperti pada Gambar 4.12.

2. Untuk mengakses konfigurasi *provider*, buka `Setings` > `Users` > `SSO MyITS Provider`. Untuk membuat konfigurasi baru, tekan `Create` , sedangkan untuk mengunakan atau mengatur konfigurasi yang sudah ada, pilih konfigurasi lalu tekan `Edit`.



Gambar 4.18 Akses Konfigurasi *Provider*

3. Masukkan konfigurasi *client ID*, *client secret*, URL *provider*, dan *scope*. Beri centang pada `Enabled` untuk mengaktifkan konfigurasi. Untuk menonaktifkan konfigurasi, hilangkan centang pada `Enabled`. Jika sudah, klik `Save`.

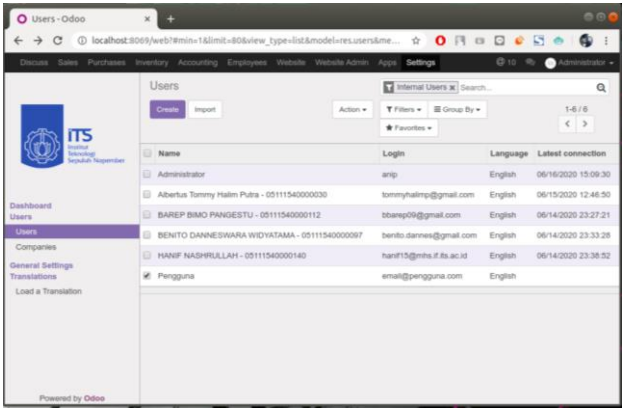


Gambar 4.19 Konfigurasi *Provider*

4.3.3 Pengaturan *User*

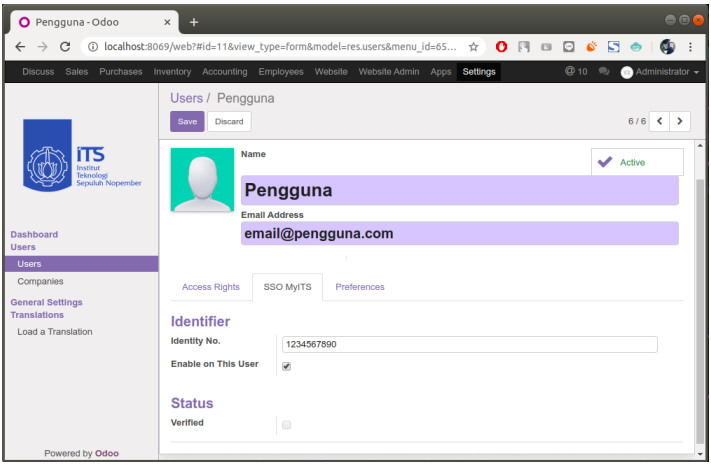
Langkah-langkah dalam konfigurasi *user* pada modul SSO MyITS Odoo adalah sebagai berikut:

1. *Login* menggunakan akun administrator, seperti pada Gambar 4.12.
2. Buka `Setings` > `Users` > `Users`, lalu pilih *user*.



Gambar 4.20 Akses *User*

3. Klik `Edit`, lalu pilih *tab* `SSO MyITS`. Masukkan nomor identitas milik *user* dan beri centang `Enabled` untuk mengaktifkan SSO MyITS pada *user* tersebut. Untuk menonaktifkan SSO MyITS pada *user*, hilangkan tanda centang pada `Enabled`. Untuk penggantian akun SSO MyITS, langsung ganti `Identity No.` dengan nomor identitas *user* yang baru. Jika sudah, klik `Save`. `Verified` akan tercentang saat *user* berhasil *login* dengan SSO MyITS pertama kali.

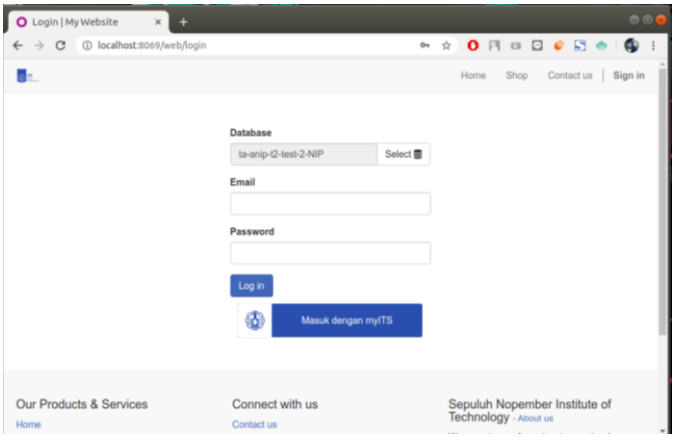


Gambar 4.21 Konfigurasi SSO MyITS Pada *User*

4.3.4 *Login* Dengan SSO MyITS

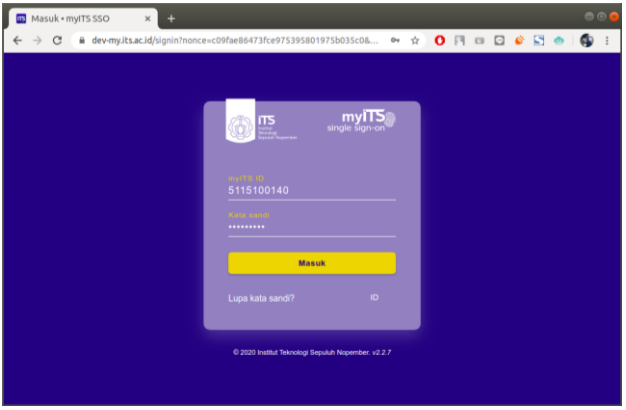
Langkah-langkah yang dilakukan *user* untuk *login* dengan SSO MyITS Odoo adalah sebagai berikut:

1. Pada halaman `login`, tekan tombol `Masuk dengan MyITS`. *User* akan di *redirect* ke *provider* SSO MyITS.



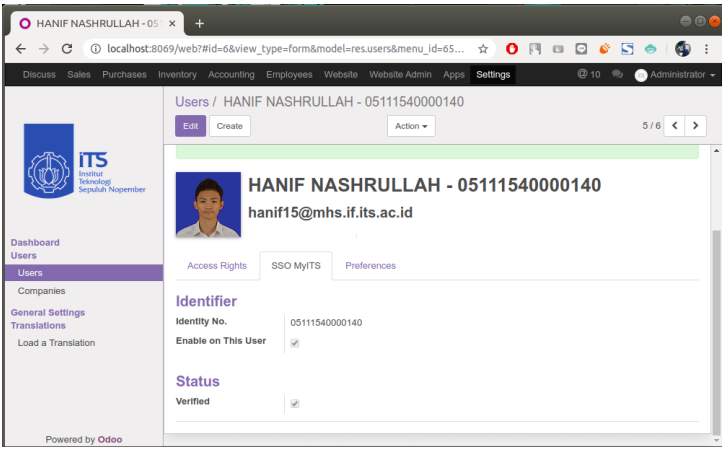
Gambar 4.22 *User Login* Melalui SSO MyITS

2. *Login* dengan akun MyITS. Jika berhasil, *user* akan diarahkan kembali ke *client*.



Gambar 4.23 *Login* Dengan Akun MyITS

3. Setelah diarahkan kembali ke *client*, modul akan melakukan proses autentikasi untuk *user*. Jika berhasil terautentikasi, maka *user* diberikan sesi serta diarahkan untuk masuk sistem. Jika *user* melakukan *login* dengan SSO MyITS untuk pertama kali, maka `Verified` akan tercentang dan beberapa data dari akun MyITS disimpan di profil akun user Odoo. Gambar 4.24 menunjukkan data *user* setelah *login* pertama kali yang dapat dilihat oleh administrator.



Gambar 4.24 *Profil User Odoo* Setelah *Login* Dengan SSO MyITS.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini, dijelaskan mengenai proses pengujian yang dilakukan serta evaluasi hasil pengujian.

5.1 Lingkungan Pengujian

Lingkungan pengujian modul SSO MyITS dijelaskan pada Tabel 5.1.

Tabel 5.1 Lingkungan Pengujian Modul SSO MyITS

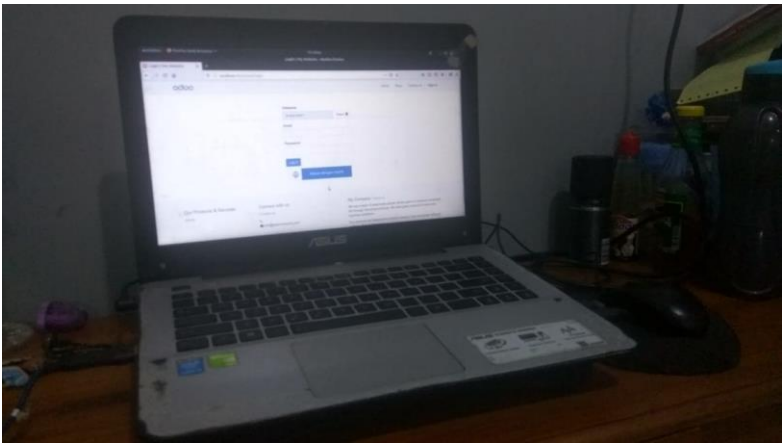
Jenis Lingkungan Pengujian	Detail
Perangkat Keras	• Komputer ASUS X455LF • Prosesor Intel® Core™ i5 5200U • VGA Nvidia® GeForce® 930M • RAM DDR3 4096MB • HDD 500 GB
Perangkat Lunak	• ERP Odoo v10 • Basis Data PostgreSQL 10.5 • Sistem Operasi Linux Ubuntu 18.04 • *Browser*

	Google Chrome, Chromium, Mozilla Firefox
Sistem	• *Provider* https://dev-my.its.ac.id • *Scope* openid profile email phone

Pengujian dilakukan dengan menggunakan koneksi jaringan di lingkungan ITS dan di luar ITS. Pada lingkungan ITS, pengujian dilakukan menggunakan koneksi jaringan Informatics_Wifi di Departemen Informatika ITS. Gambar lingkungan pengujian pada Depertemen Informatika ITS dapat dilihat pada Gambar 5.1.

Gambar 5.1 Lingkungan Pengujian Pada Depertemen Informatika ITS

Sedangkan untuk lingkungan pengujian di luar jaringan koneksi ITS menggunakan koneksi MyRepublic. Gambar lingkungan pengujian di luar jaringan ITS dapat dilihat pada Gambar 5.2.



Gambar 5.2 Lingkungan Pengujian Di Luar Jaringan ITS

5.2 Pengujian

Pengujian dilakukan dengan tujuan mengetahui keberhasilan sistem memproses autentikasi serta mengetahui performa sistem yang telah dibangun. Pengujian keberhasilan sistem dilakukan dengan melakukan simulasi dalam beberapa macam skenario, sedangkan pengujian performa dilakukan dengan pengukuran kecepatan proses autentikasi sistem.

5.2.1 Pengujian Simulasi

Pada pengujian simulasi, dilakukan pengujian pada sistem dengan menggunakan beberapa skenario. Skenario-skenario tersebut dijelaskan pada Tabel 5.2 sebagai berikut.

Tabel 5.2 Skenario-skenario Pada Pengujian Simulasi

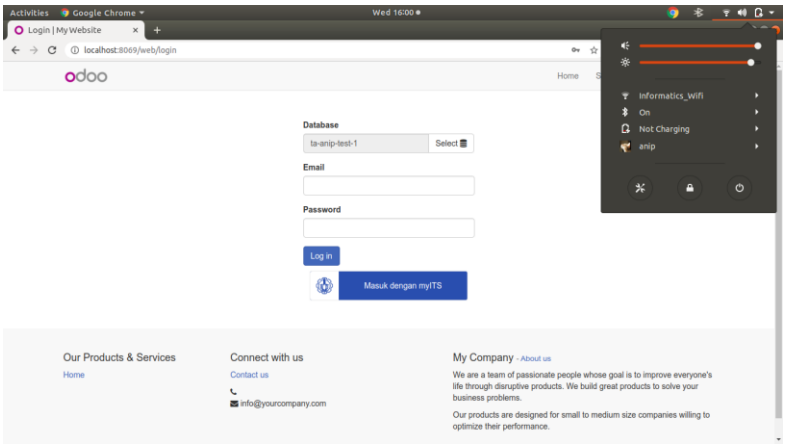
Skenario	Tujuan
Login ke sistem	Mengetahui keberhasilan sistem melakukan proses autentikasi

Proses bisnis pembelian barang dengan dan tanpa login melalui SSO MyITS	Mengetahui keberhasilan sistem dalam proses bisnis pembelian barang saat pengguna *login* dengan maupun tanpa melalui SSO MyITS
Login dengan tiga akun berbeda	Mengetahui keberhasilan sistem melakukan proses autentikasi pada akun yang berbeda-beda

5.2.1.1 Skenario *Login* Ke Sistem

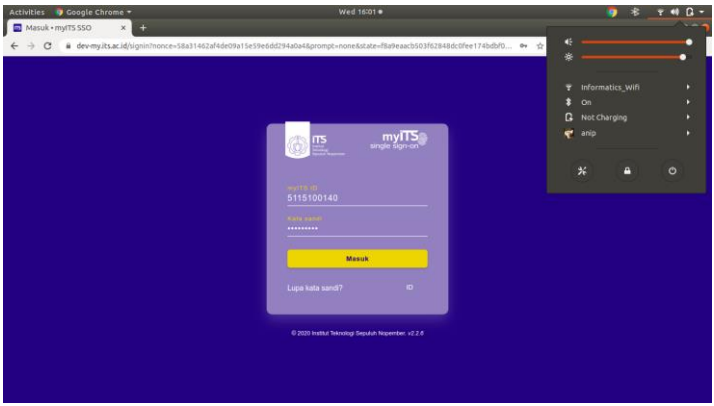
Pada simulasi ini, pengguna melakukan *login* dengan SSO MyITS melalui halaman `login` pada Odoo. Dalam simulasi ini, koneksi yang digunakan adalah koneksi jaringan ITS (Informatics_Wifi). Tahap-tahap yang dilakukan dalam simulasi dengan skenario *login* ke sistem adalah sebagai berikut:

1. *User* mengakses halaman `login` Odoo, kemudian menekan tombol `Masuk dengan MyITS.`



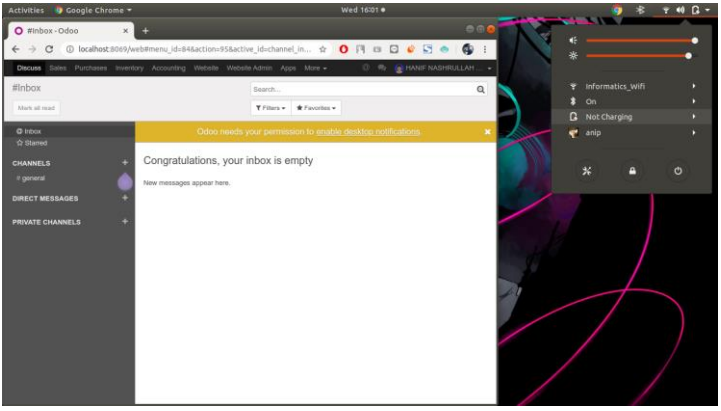
Gambar 5.3 *User* Mengakses Halaman `Login` Odoo

2. *User* diarahkan ke *provider* SSO MyITS, kemudian melakukan *login* dengan memasukkan *username* dan *password*, kemudian menekan tombol `Masuk`.



Gambar 5.4 *User* Melakukan *Login* Di SSO MyITS

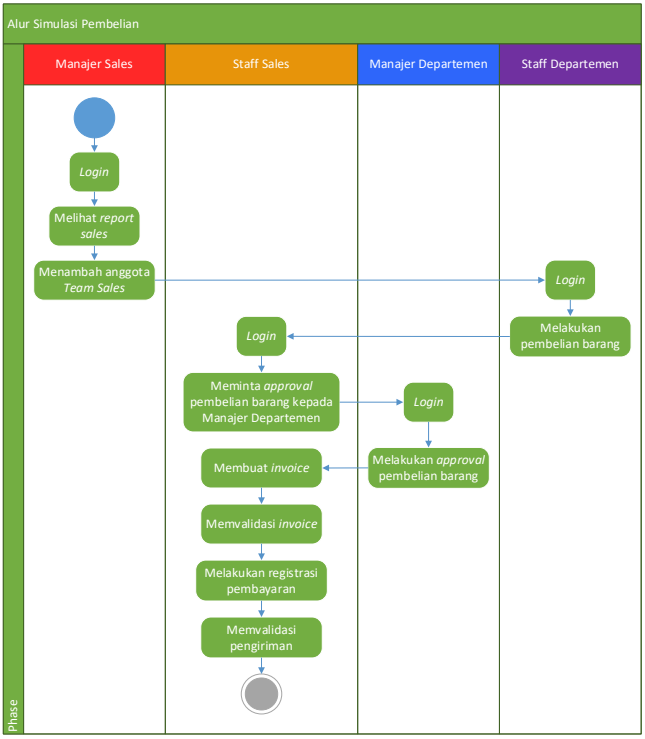
3. *User* diarahkan kembali ke sistem Odoo, kemudian masuk ke sistem Odoo.



Gambar 5.5 *User* Diarahkan Kembali Ke Sistem Odoo Dan Masuk Ke Sistem Odoo

5.2.1.2 Skenario Proses Bisnis Pembelian Barang Dengan Dan Tanpa *Login* Melalui SSO MyITS

Pada simulasi ini, pengguna terlibat dalam proses bisnis pembelian barang yang mungkin terjadi di ERP Odoo Logistik Terpusat ITS. Pengguna dibagi menjadi 4, yaitu Manajer Sales, Staff Sales, Manajer Departemen, dan Staff Departemen. Manajer Sales mengakses *report sales* dan memanajemen *Sales Team*. Staff Sales melakukan pemrosesan pembelian barang. Staff Departemen melakukan pembelian barang. Sedangkan Manajer Departemen melakukan *approval* pembelian barang. Alur simulasi proses bisnis pembelian barang dapat ditunjukkan dengan *activity diagram* pada Gambar 5.6 berikut.

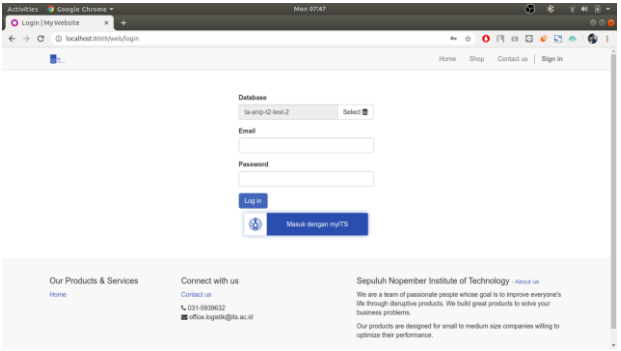
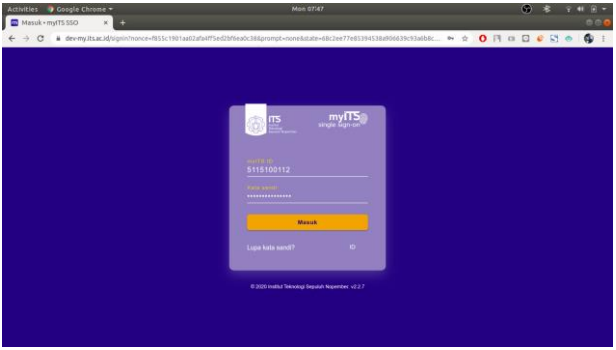


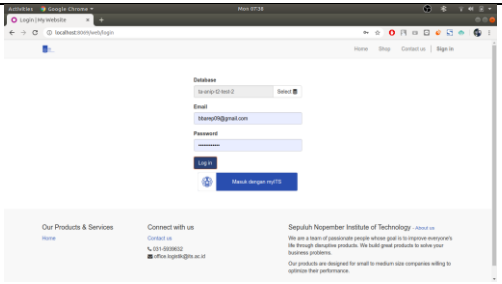
Gambar 5.6 Alur Simulasi Proses Bisnis Pembelian Barang

Dalam simulasi ini, koneksi yang digunakan adalah koneksi jaringan luar ITS (MyRepublic). Simulasi dilakukan dengan keadaan pengguna *login* dengan dan tanpa SSO MyITS. Tahap-tahap yang dilakukan dalam simulasi proses bisnis pembelian barang dijelaskan sebagai berikut:

1. Manajer Sales melakukan *login*

Tabel 5.3 Manajer Sales Melakukan *Login*

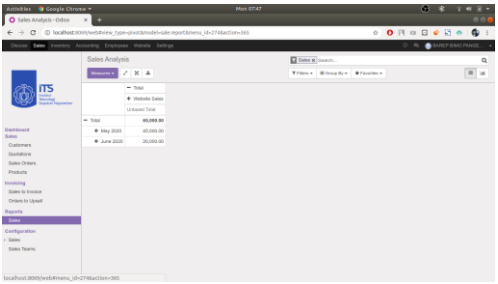
Dengan SSO MyITS	a. Menekan tombol `Masuk dengan MyITS`  **Gambar 5.7 Manajer Sales Menekan Tombol `Masuk dengan MyITS`** b. *Login* dengan akun SSO MyITS  **Gambar 5.8 Manajer Sales *Login* Dengan Akun SSO MyITS**

Tanpa SSO MyITS	

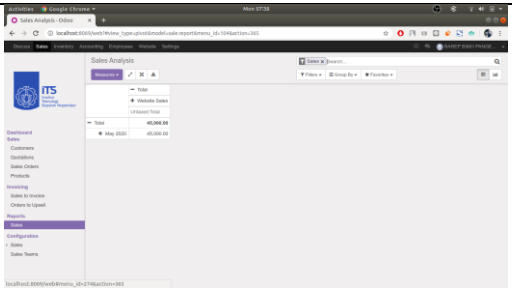
Gambar 5.9 Manajer Sales *Login* Tanpa Akun SSO MyITS

2. Manajer Sales melihat *report sales*

Tabel 5.4 Manajer Sales Melihat *Report Sales*

Dengan SSO MyITS	

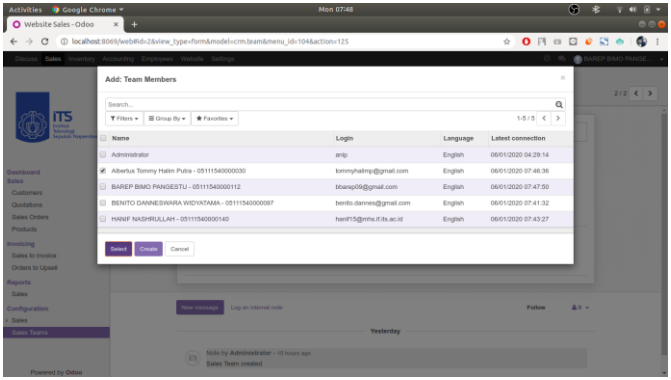
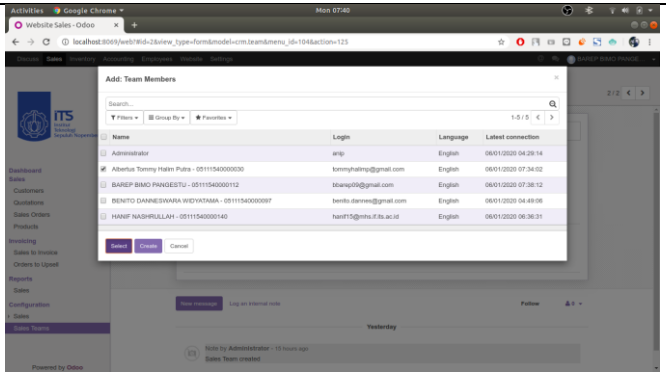
Gambar 5.10 Manajer Sales (Dengan SSO MyITS) Melihat *Report Sales*

Tanpa SSO MyITS	

Gambar 5.11 Manajer Sales (Tanpa SSO MyITS) Melihat *Report Sales*

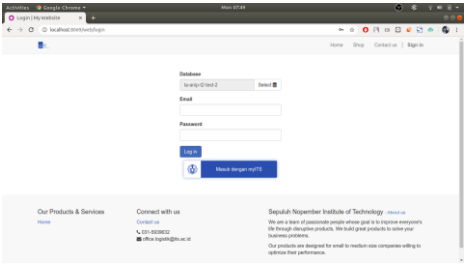
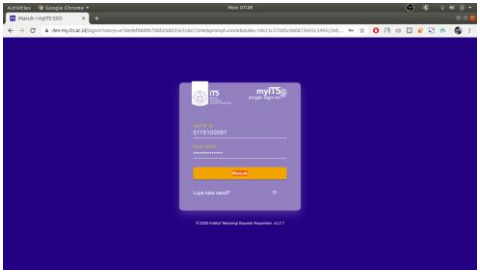
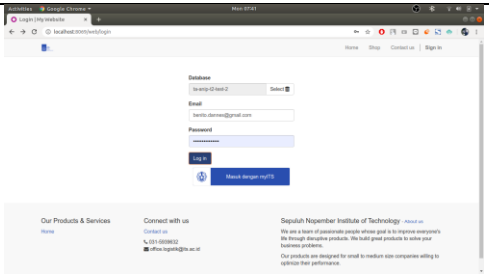
3. Manajer Sales menambah anggota *Team Sales*

Tabel 5.5 Manajer Sales Menambah Anggota *Team Sales*

Dengan SSO MyITS	 **Gambar 5.12 Manajer Sales (Dengan SSO MyITS) Menambah Anggota *Team Sales***
Tanpa SSO MyITS	 **Gambar 5.13 Manajer Sales (Tanpa SSO MyITS) Menambah Anggota *Team Sales***

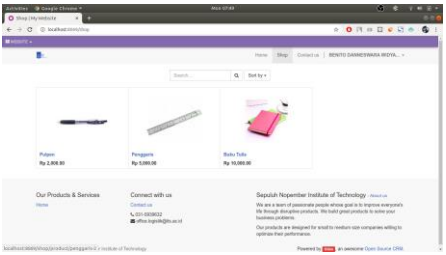
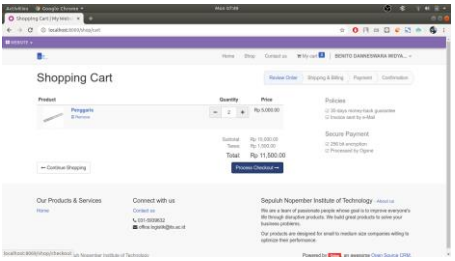
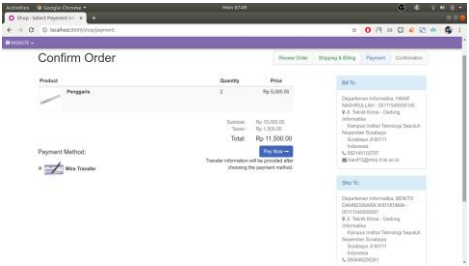
4. Staff Departemen melakukan *login*

Tabel 5.6 Staff Departemen Melakukan *Login*

Dengan SSO MyITS	a. Menekan tombol `Masuk dengan MyITS`  **Gambar 5.14 Staff Departemen Menekan Tombol `Masuk dengan MyITS`** b. *Login* dengan akun SSO MyITS  **Gambar 5.15 Staff Departemen *Login* Dengan Akun SSO MyITS**
Tanpa SSO MyITS	 **Gambar 5.16 Staff Departemen *Login* Tanpa Akun SSO MyITS**

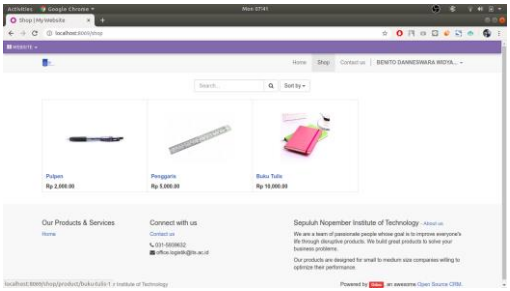
5. Staff Departemen melakukan pembelian barang

Tabel 5.7 Staff Departemen Melakukan Pembelian Barang

Dengan SSO MyITS	a. Memilih barang yang dibeli  **Gambar 5.17 Staff Departemen (Dengan SSO MyITS) Memilih Barang Yang Dibeli** b. Menentukan jumlah dan *checkout*  **Gambar 5.18 Staff Departemen (Dengan SSO MyITS) Menentukan Jumlah Dan *Checkout*** c. Mengkonfirmasi pembelian  **Gambar 5.19 Staff Departemen (Dengan SSO MyITS) Mengkonfirmasi Pembelian**

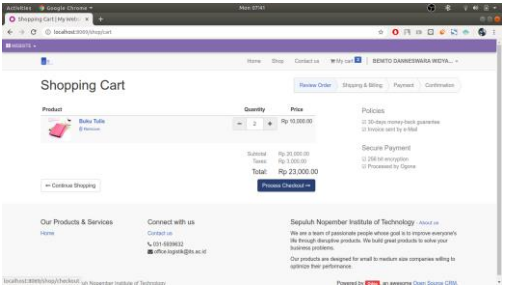
Tanpa SSO MyITS

a. Memilih barang yang dibeli



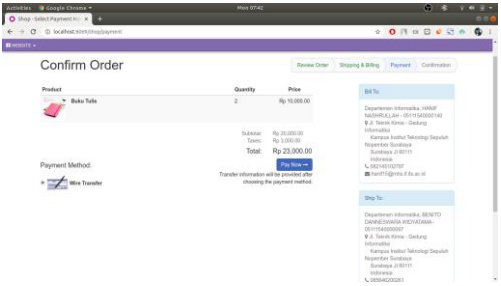
Gambar 5.20 Staff Departemen (Tanpa SSO MyITS) Memilih Barang Yang Dibeli

b. Menentukan jumlah dan *checkout*



Gambar 5.21 Staff Departemen (Tanpa SSO MyITS) Menentukan Jumlah Dan *Checkout*

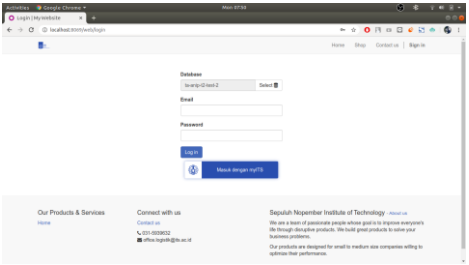
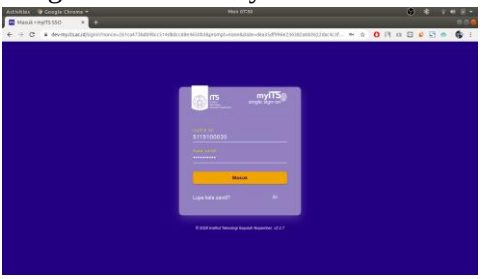
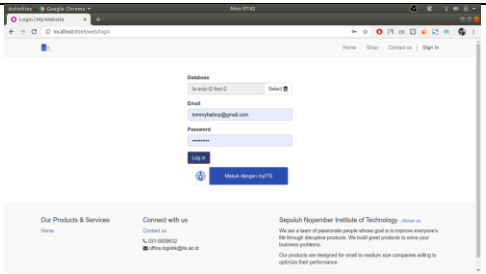
c. Mengkonfirmasi pembelian



Gambar 5.22 Staff Departemen (Tanpa SSO MyITS) Mengkonfirmasi Pembelian

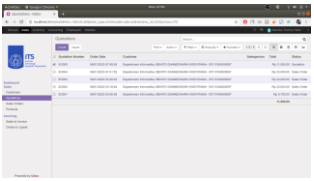
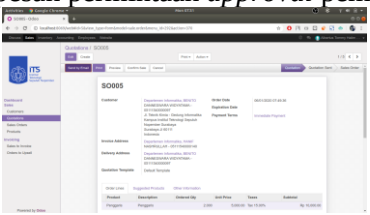
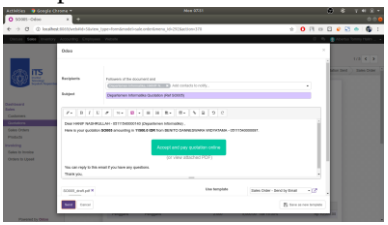
6. Staff Sales melakukan *login*

Tabel 5.8 Staff Sales Melakukan *Login*

<table>
<tr>
<td>Dengan SSO MyITS</td>
<td>
a. Menekan tombol <code>Masuk dengan MyITS</code>

Gambar 5.23 Staff Sales Menekan Tombol <code>Masuk dengan MyITS</code>
b. Login dengan akun SSO MyITS

Gambar 5.24 Staff Sales Login Dengan Akun SSO MyITS
</td>
</tr>
<tr>
<td>Tanpa SSO MyITS</td>
<td>

Gambar 5.25 Staff Sales Login Tanpa Akun SSO MyITS
</td>
</tr>
</table>

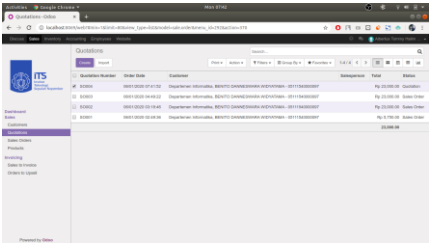
7. Staff Sales meminta *approval* pembelian barang kepada Manajer Departemen

Tabel 5.9 Staff Sales Meminta *Approval* Pembelian Barang Kepada Manajer Departemen

Dengan SSO MyITS	a. Memilih *quotation*  **Gambar 5.26 Staff Sales (Dengan SSO MyITS) Memilih *Quotation*** b. Membuat pesan permintaan *approval* pembelian barang  **Gambar 5.27 Staff Sales (Dengan SSO MyITS) Membuat Pesan Permintaan *Approval* Pembelian Barang** c. Mengirim pesan permintaan *approval* pembelian barang ke Manajer Departemen  **Gambar 5.28 Staff Sales (Dengan SSO MyITS) Mengirim Pesan Permintaan *Approval* Pembelian Barang Ke Manajer Departemen**

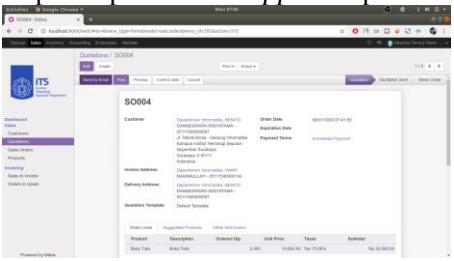
Tanpa SSO MyITS

a. Memilih *quotation*



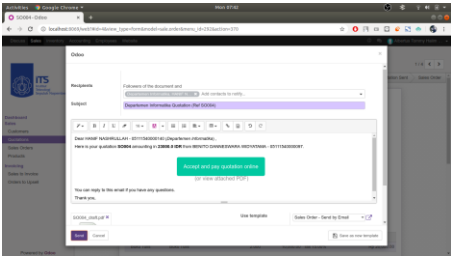
Gambar 5.29 Staff Sales (Tanpa SSO MyITS) Memilih *Quotation*

b. Membuat pesan permintaan *approval* pembelian barang



Gambar 5.30 Staff Sales (Tanpa SSO MyITS) Membuat Pesan Permintaan *Approval* Pembelian Barang

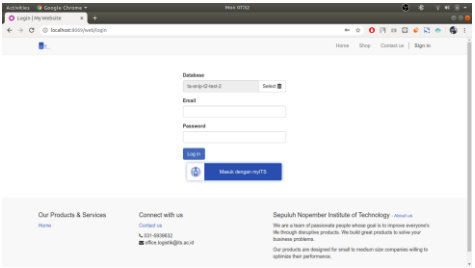
c. Mengirim pesan permintaan *approval* pembelian barang ke Manajer Departemen

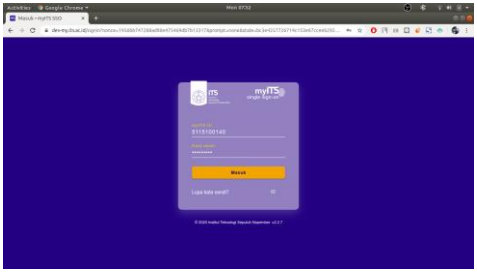


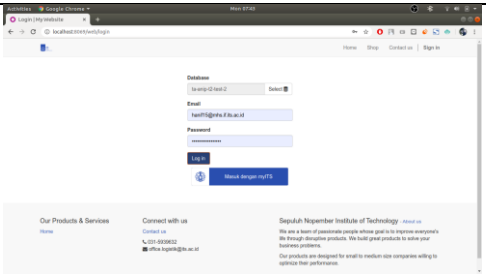
Gambar 5.31 Staff Sales (Tanpa SSO MyITS) Mengirim Pesan Permintaan *Approval* Pembelian Barang Ke Manajer Departemen

8. Manajer Departemen melakukan *login*

Tabel 5.10 Manajer Departemen Melakukan *Login*

<table>
<tr>
<td>Dengan SSO MyITS</td>
<td>
a. Menekan tombol <code>Masuk dengan MyITS</code>


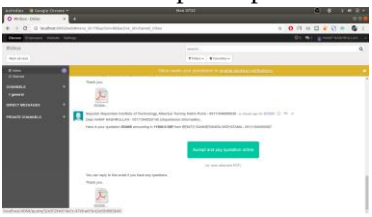
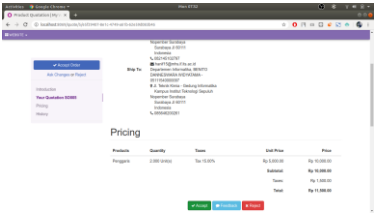
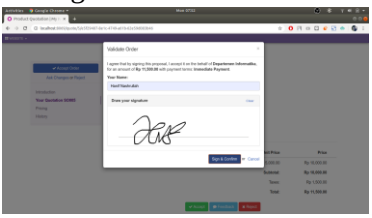
Gambar 5.32 Manajer Departemen Menekan Tombol <code>Masuk dengan MyITS</code>
b. Login dengan akun SSO MyITS


Gambar 5.33 Manajer Departemen Login Dengan Akun SSO MyITS
</td>
</tr>
<tr>
<td>Tanpa SSO MyITS</td>
<td>


Gambar 5.34 Manajer Departemen Login Tanpa Akun SSO MyITS
</td>
</tr>
</table>

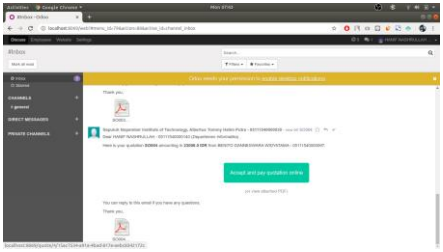
9. Manajer Departemen melakukan *approval* pembelian barang

Tabel 5.11 Manajer Departemen Melakukan *Approval* Pembelian Barang

Dengan SSO MyITS	a. Menerima dan membuka pesan permintaan *approval*  **Gambar 5.35 Manajer Departemen (Dengan SSO MyITS) Menerima Dan Membuka Pesan Permintaan *Approval*** b. *Accept* permintaan pembelian barang  **Gambar 5.36 Manajer Departemen (Dengan SSO MyITS) *Accept* Permintaan Pembelian Barang** c. Menandatangani dan mengkonfirmasi *approval* pembelian barang  **Gambar 5.37 Manajer Departemen (Dengan SSO MyITS) Menandatangani Dan Mengkonfirmasi *Approval* Pembelian Barang**

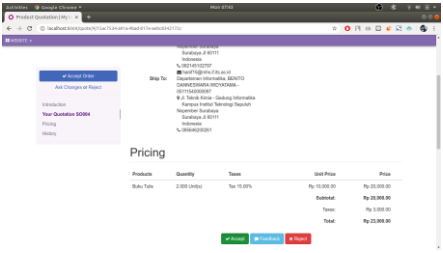
Tanpa SSO MyITS

a. Menerima dan membuka pesan permintaan *approval*



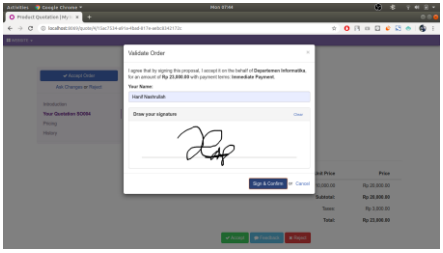
Gambar 5.38 Manajer Departemen (Tanpa SSO MyITS) Menerima Dan Membuka Pesan Permintaan *Approval*

b. *Accept* permintaan pembelian barang



Gambar 5.39 Manajer Departemen (Tanpa SSO MyITS) *Accept* Permintaan Pembelian Barang

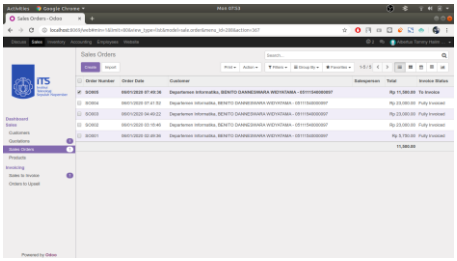
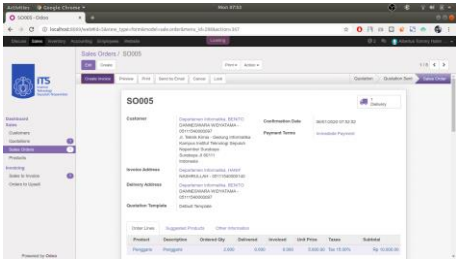
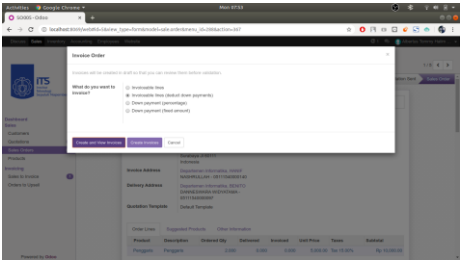
c. Menandatangani dan mengkonfirmasi *approval* pembelian barang



Gambar 5.40 Manajer Departemen (Tanpa SSO MyITS) Menandatangani Dan Mengkonfirmasi *Approval* Pembelian Barang

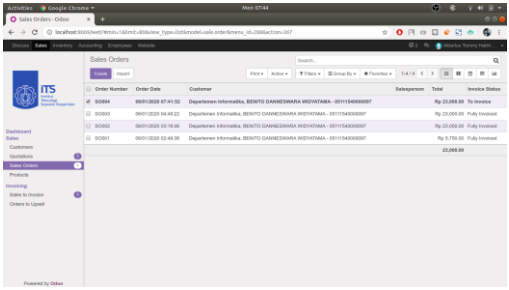
10. Staff Sales membuat *invoice*

Tabel 5.12 Staff Sales Membuat *Invoice*

Dengan SSO MyITS	a. Memilih *sales order*  **Gambar 5.41 Staff Sales (Dengan SSO MyITS) Memilih *Sales Order*** b. Membuat *invoice*  **Gambar 5.42 Staff Sales (Dengan SSO MyITS) Membuat *Invoice*** c. Memilih jenis *invoice*  **Gambar 5.43 Staff Sales (Dengan SSO MyITS) Memilih Jenis *Invoice***

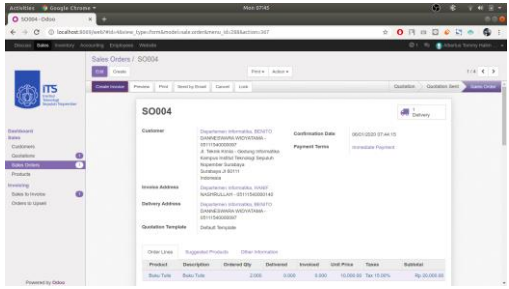
Tanpa SSO MyITS

a. Memilih *sales order*



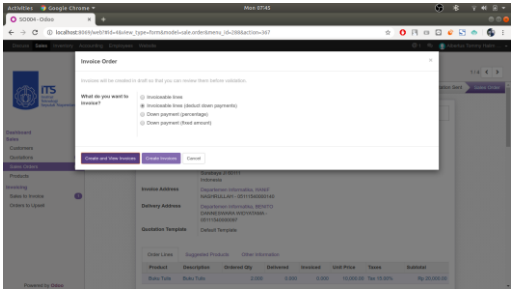
Gambar 5.44 Staff Sales (Tanpa SSO MyITS) Memilih *Sales Order*

b. Membuat *invoice*



Gambar 5.45 Staff Sales (Tanpa SSO MyITS) Membuat *Invoice*

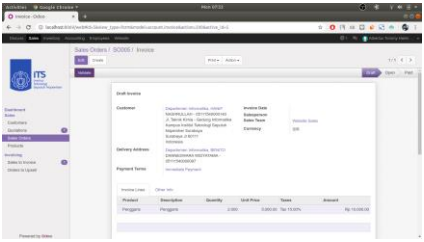
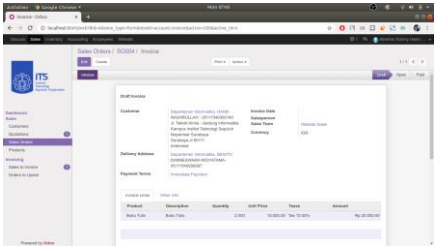
c. Memilih jenis *invoice*



Gambar 5.46 Staff Sales (Tanpa SSO MyITS) Memilih Jenis *Invoice*

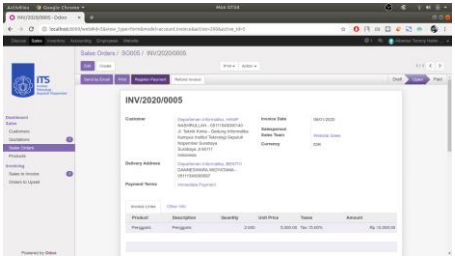
11. Staff Sales memvalidasi *invoice*

Tabel 5.13 Staff Sales Memvalidasi *Invoice*

Dengan SSO MyITS	 **Gambar 5.47 Staff Sales (Dengan SSO MyITS) Memvalidasi *Invoice***
Tanpa SSO MyITS	 **Gambar 5.48 Staff Sales (Tanpa SSO MyITS) Memvalidasi *Invoice***

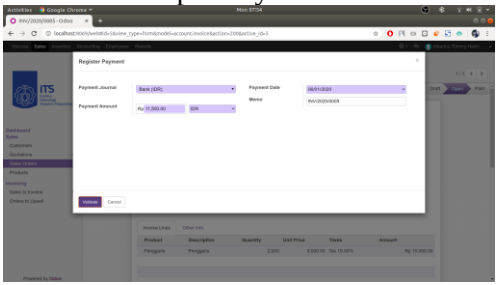
12. Staff Sales melakukan registrasi pembayaran

Tabel 5.14 Staff Sales Melakukan Registrasi Pembayaran

Dengan SSO MyITS	a. Melakukan registrasi pembayaran pada *invoice*  **Gambar 5.49 Staff Sales (Dengan SSO MyITS) Melakukan Registrasi Pembayaran Pada *Invoice***

Dengan SSO MyITS

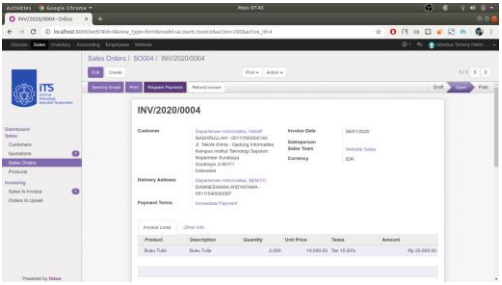
b. Memasukkan detail pembayaran



Gambar 5.50 Staff Sales (Dengan SSO MyITS) Memasukkan Detail Pembayaran

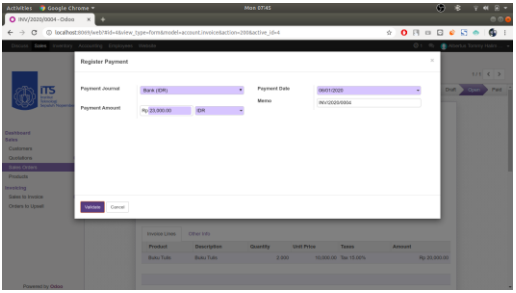
Tanpa SSO MyITS

a. Melakukan registrasi pembayaran pada *invoice*



Gambar 5.51 Staff Sales (Tanpa SSO MyITS) Melakukan Registrasi Pembayaran Pada *Invoice*

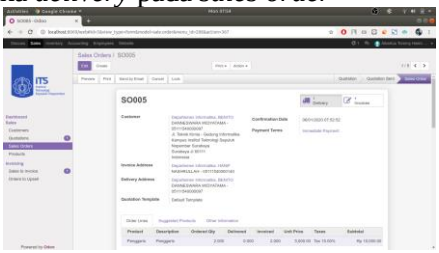
b. Memasukkan detail pembayaran

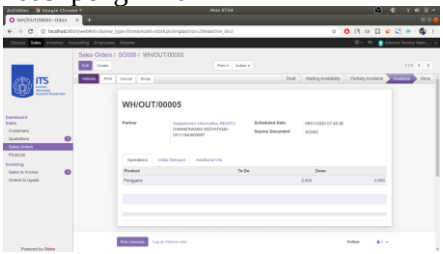


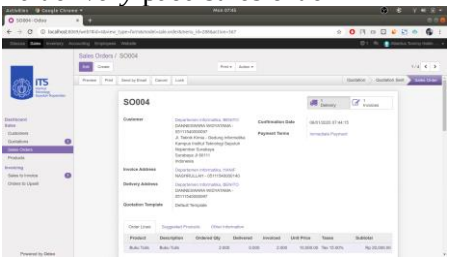
Gambar 5.52 Staff Sales (Tanpa SSO MyITS) Memasukkan Detail Pembayaran

13. Staff Sales memvalidasi pengiriman

Tabel 5.15 Staff Sales Memvalidasi Pengiriman

<table>
<tr>
<td>Dengan SSO MyITS</td>
<td>
a. Membuka delivery pada sales order


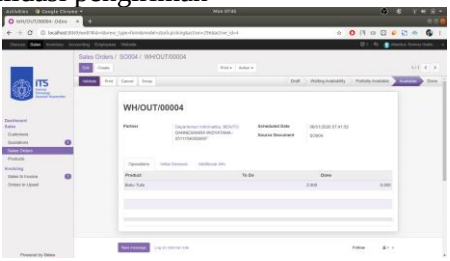
Gambar 5.53 Staff Sales (Dengan SSO MyITS) Membuka Delivery Pada Sales Order
b. Memvalidasi pengiriman


Gambar 5.54 Staff Sales (Dengan SSO MyITS) Memvalidasi Pengiriman
</td>
</tr>
<tr>
<td>Tanpa SSO MyITS</td>
<td>
a. Membuka delivery pada sales order


Gambar 5.55 Staff Sales (Tanpa SSO MyITS) Membuka Delivery Pada Sales Order
</td>
</tr>
</table>

Tanpa SSO MyITS

b. Memvalidasi pengiriman

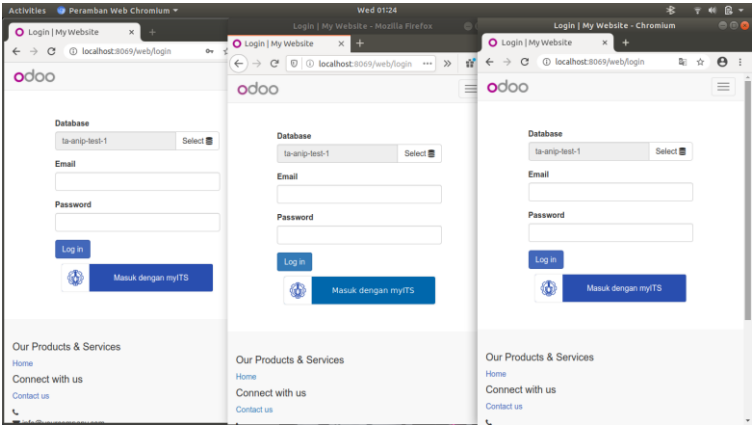


Gambar 5.56 Staff Sales (Tanpa SSO MyITS) Memvalidasi Pengiriman

5.2.1.3 Skenario *Login* Dengan Tiga Akun Berbeda

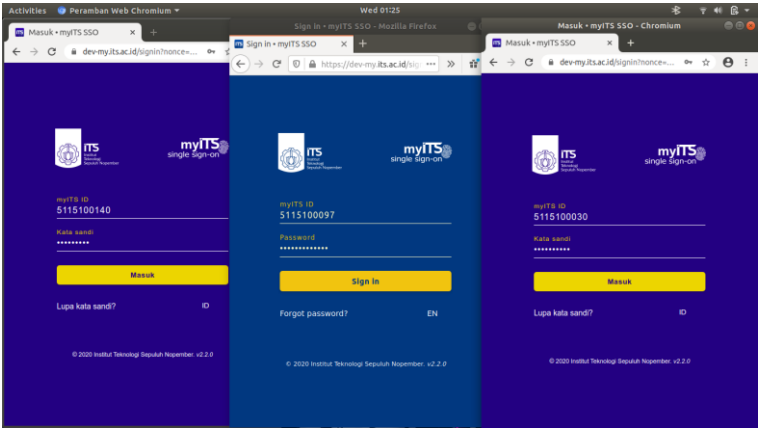
Pada skenario ini, tiga *user* melakukan *login* dengan SSO MyITS melalui halaman `login` pada Odoo seperti skenario pertama. Dalam simulasi ini, koneksi yang digunakan adalah koneksi jaringan luar ITS (MyRepublic). Tahap-tahap yang dilakukan dalam simulasi dengan skenario *login* dengan tiga akun berbeda adalah sebagai berikut:

1. Ketiga *user* mengakses halaman `login` Odoo, kemudian menekan tombol `Masuk dengan MyITS.`



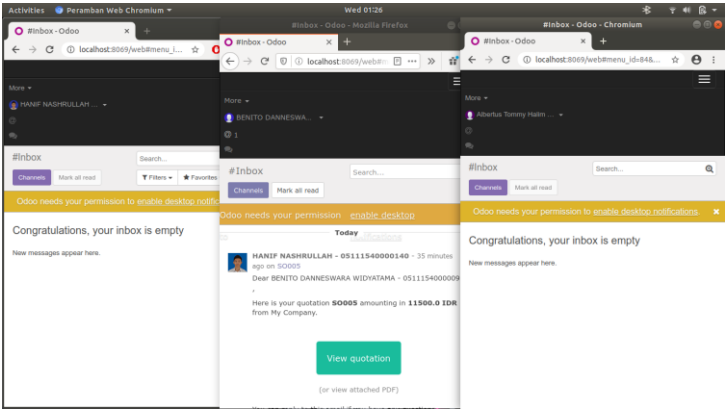
Gambar 5.57 Ketiga *User* Mengakses Halaman `Login` Odoo

2. Ketiga *user* diarahkan ke *provider* SSO MyITS, kemudian melakukan *login* dengan memasukkan *username* dan *password*, kemudian menekan tombol `Masuk.`



Gambar 5.58 Ketiga *User* Melakukan *Login* Di SSO MyITS

3. Ketiga *user* diarahkan kembali ke sistem Odoo, kemudian masuk ke sistem Odoo.



Gambar 5.59 Ketiga *User* Diarahkan Kembali Ke Sistem Odoo Dan Masuk Ke Sistem Odoo

5.2.2 Pengujian Performa

Pada pengujian performa, dilakukan pengukuran waktu yang dibutuhkan sistem untuk melakukan proses autentikasi pengguna. Pengukuran waktu dilakukan dengan satuan detik (s). Perhitungan waktu dimulai saat sistem Odoo menerima *request* pengguna untuk *login* dengan SSO MyITS hingga *user* berhasil masuk ke sistem Odoo. Dalam pengujian ini, koneksi yang digunakan adalah koneksi jaringan luar ITS (MyRepublic). Lingkungan pengujian modul pada pengujian performa sesuai dengan Tabel 5.1. Pada pengujian performa, diasumskan pengguna sudah *login* di *provider* SSO MyITS sehingga pengguna tidak perlu memasukkan *username* dan *password* di SSO MyITS. Akun yang digunakan pada pengujian ini adalah satu akun SSO MyITS yang digunakan berulang-ulang. Tahap-tahap yang dilakukan sistem SSO MyITS – Odoo pada pengujian performa adalah sebagai berikut :

1. Pengguna menekan tombol `Masuk dengan ITS` di halaman *login* sistem Odoo. Pengukuran waktu dimulai saat sistem Odoo menerima *request* pengguna.
2. Sistem Odoo mengirim *request* kode autorisasi pada SSO MyITS.
3. SSO MyITS memberikan kode autorisasi kepada Sistem Odoo.
4. Sistem Odoo mengirim *request* JWT ke SSO MyITS.
5. SSO MyITS mengirim JWT kepada Sistem Odoo.
6. Sistem Odoo memproses JWT dan mengirim *request* informasi pengguna menggunakan *access token* dari JWT.
7. SSO MyITS mengirim informasi pengguna kepada Sistem Odoo
8. Jika pengguna belum pernah *login* melalui SSO MyITS, informasi pengguna disimpan pada akun Odoo pengguna. Namun jika sudah pernah maka langsung ke tahap 9.
9. Sistem Odoo mengautentikasi pengguna dan mengarahkan pengguna untuk masuk ke Sistem Odoo. Pengukuran waktu selesai ketika pengguna berhasil masuk ke sistem Odoo.

10. Untuk percobaan *login* selanjutnya, pengguna melakukan *logout* di Sistem Odoo. Kemudian mengulangi dari tahap 1.

Pengujian dilakukan 3 kali dan pada tiap pengujian dilakukan 5 kali *login,* dengan keadaan pengguna belum pernah *login* melalui SSO MyITS pada percobaan *login* pertama, sehingga data akun Odoo belum disinkronkan dengan data akun SSO MyITS. Pengujian dilakukan 3 kali menggunakan kondisi dan akun yang sama untuk melihat dan membandingkan karakteristik yang terjadi. Hasil dari pengujian performa ditunjukkan pada Tabel 5.16 sebagai berikut.

Tabel 5.16 Hasil Pengujian Performa Dalam Detik (s)

Detik (s)	**Uji ke-1**	**Uji ke-2**	**Uji ke-3**
***Login* ke-1**	4.410	4.397	4.743
***Login* ke-2**	2.506	2.583	2.483
***Login* ke-3**	2.553	2.668	2.742
***Login* ke-4**	2.640	2.473	2.566
***Login* ke-5**	3.038	2.914	2.461

Dari ketiga pengujian tersebut, didapatkan bahwa pemrosesan *login* membutuhkan waktu kurang dari 5 detik, dan pada tiap pengujian, *login* pertama memakan waktu lebih lama dibandingkan *login* berikutnya. Hal ini disebabkan karena pada *login* pertama dilakukan pengambilan dan penyimpanan informasi pengguna pada akun Odoo dari SSO MyITS.

5.3 Evaluasi

Dari hasil pengujian yang telah dilakukan sebelumnya didapat evaluasi yang dijelaskan sebagai berikut:

1. Modul SSO MyITS Odoo berhasil mengautentikasi pengguna sesuai dengan akun SSO MyITS dan akun Odoo pengguna yang bersangkutan.
2. Pengguna yang *login* melalui SSO MyITS berhasil melakukan aktivitas dan proses bisnis seperti pengguna yang *login* secara langsung ke sistem Odoo.

3. Modul Odoo SSO MyITS berhasil memproses autentikasi lebih dari satu pengguna.
4. Modul Odoo SSO MyITS berhasil memproses autentikasi pengguna di jaringan ITS dan di jaringan luar ITS.
5. Dari pengujian performa, modul Odoo SSO MyITS memproses autentikasi kurang dari 5 detik, dimana *login* pertama memakan waktu paling lama dibandingkan dengan *login* kedua dan selanjutnya. Hal ini dikarenakan modul melakukan pengambilan data pengguna dari *provider* SSO MyITS dan menyimpannya ke dalam basis data Sistem Odoo saat pengguna melakukan *login* dengan SSO MyITS untuk pertama kali.

BAB VI
KESIMPULAN

Pada bab ini, dijelaskan kesimpulan yang didapat berdasarkan pengerjaan Tugas Akhir telah dilakukan dan saran untuk pengembangan modul Odoo SSO MyITS.

6.1 Kesimpulan

Dari perancangan, pengembangan, dan pengujian modul Odoo SSO MyITS untuk aplikasi ERP Odoo Logistik Terpusat ITS, dapat diambil kesimpulan yang dijelaskan sebagai berikut:

1. Modul Odoo SSO MyITS berhasil dibangun pada spesifikasi dan desain ERP Odoo Logistik Terpusat ITS, yaitu Odoo versi 10.
2. Modul Odoo SSO MyITS dapat melakukan proses autentikasi dengan melakukan *redirect* pengguna ke SSO MyITS untuk melakukan *login*, *request* JWT kepada SSO MyITS, *request* informasi pengguna ke SSO MyITS, serta memverifikasi pengguna menggunakan informasi pengguna dari SSO MyITS.
3. Modul Odoo SSO MyITS dapat melakukan transaksi data dengan *provider* SSO MyITS dengan menerima kode autorisasi dari *provider* SSO MyITS; *request* JWT dengan memberikan kode autorisasi, *client id*, *client secret*, alamat respon, dan metode autentikasi yang digunakan kepada *token endpoint* SSO MyITS; serta *request* informasi mengenai pengguna yang bersangkutan dengan memberikan *access token* dari JWT dan skema kepada *userinfo endpoint* SSO MyITS.
4. Pengguna yang *login* melalui SSO MyITS dapat melakukan aktivitas dan proses bisnis seperti pengguna yang *login* secara langsung ke sistem Odoo.
5. Modul Odoo SSO MyITS dapat memproses autentikasi kurang dari 5 detik, dimana *login* pertama memakan waktu paling lama dibandingkan dengan *login* kedua dan

selanjutnya. Hal ini dikarenakan modul melakukan pengambilan data pengguna dari *provider* SSO MyITS dan menyimpannya ke dalam basis data Sistem Odoo saat pengguna melakukan *login* dengan SSO MyITS untuk pertama kali.

6.2 Saran

Dari pengerjaan Tugas Akhir yang dilakukan, terdapat saran untuk pengembangan sistem di masa yang akan datang, yaitu meningkatkan jumlah informasi pengguna yang dapat diambil dari akun SSO MyITS dan disimpan pada akun Odoo.

DAFTAR PUSTAKA

[1] P. Christensson, "Authentication Definition," techterms.com, 13 Juli 2018. [Online]. Available: https://techterms.com/definition/authentication. [Diakses 6 Mei 2020].

[2] Auth0, "Application Authentication," [Online]. Available: https://auth0.com/docs/application-auth/current. [Accessed 6 Mei 2020].

[3] G. Moss, Working with Odoo, Birmingham: Packt Publishing Ltd., 2015.

[4] D. Reis, Odoo 10 Development Essentials, Birmingham: Packt Publishing Ltd., 2016.

[5] Odoo S.A., "Odoo Guidelines," [Online]. Available: https://www.odoo.com/documentation/10.0/reference/guidelines.html. [Accessed 7 Mei 2020].

[6] S. Daityari, "Single Sign-On (SSO) Explained," SitePoint, 26 Februari 2014. [Online]. Available: https://www.sitepoint.com/single-sign-on-explained/. [Accessed 6 Mei 2020].

[7] Secret Double Octopus, "SINGLE SIGN ON (SSO)," [Online]. Available: https://doubleoctopus.com/security-wiki/federation-and-sso/single-sign-on/. [Accessed 7 Mei 2020].

[8] P. Siriwardena, Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE, New York City: Apress, 2014.

[9] B. Raharjo, Mudah Belajar Python untuk Aplikasi Desktop dan Web, Bandung: Informatika, 2015.

[10] Python Software Foundation, "The Python Logo," [Online]. Available: https://www.python.org/community/logos/. [Accessed 6 Mei 2020].

[11] E. T. Ray, Learning XML: Creating Self-Describing Data, 2nd Edition, Sebastopol: O'Reilly Media, 2003.

[12] R. Jumardi, WEBSITE STATIS: Konsep dan Praktik HTML - CSS, Ponorogo: Uwais Inspirasi Indonesia, 2019.

[13] D. S. McFarland, CSS: The Missing Manual, Sebastopol: O'Reilly Media Inc., 2006.

[14] D. Flanagan, JavaScript: The Definitive Guide, Fifth Edition, Sebastopol: O'Reilly Media Inc., 2006.

[15] A. Mudhoffar, Implementasi Otentikasi Single-Factor dan Multi-Factor Berbasis Protokol WebAuthn di Aplikasi myITS Single Sign-On, Surabaya: Institut Teknologi Sepuluh Nopember, 2019.

[16] N. Sakimura, J. Bradley, M. . B. Jones, B. d. Medeiros and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1," OpenID Foundation, 8 November 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html. [Accessed 8 Mei 2020].

LAMPIRAN
KODE SUMBER

Kode Sumber S.1 Inisialisasi Modul (`__init__.py`)

```
from . import controllers
from . import models
```

Kode Sumber S.2 *Manifest* Modul (`__manifest__.py`)

```
{
    'name': "SSO MyITS Odoo Login",
    'summary': """
        Modul ini memungkinkan pengguna untuk login melalui SSO MyITS
        """,
    'author': "Hanif Nashrullah",
    'website': "http://erp.logistik.its.ac.id",
    'category': 'Login',
    'version': '1.2',
    'support': 'hn3692@gmail.com',
    'images': ['static/description/icon.png'],
    'depends': ['base', 'web'],
    'external_dependencies': {'python' : ['OpenSSL', 'cryptography', 'requests', 'jwt']},
    'data': [
        'views/auth_sso_myits_custom_login_template.xml',
        'views/auth_sso_myits_provider_view.xml',
        'views/auth_sso_myits_res_users_view.xml',
        'security/ir.model.access.csv',
    ],
    'css': [
        'static/src/css/myits-button.css',
        'static/src/fonts/Roboto Medium/Roboto Medium.css',
    ],
    'installable' : True,
    'application' : True,
}
```

Kode Sumber S.3 Inisialisasi *Controllers* (`__init__.py`)

```
from . import main
```

Kode Sumber S.4 *Controller* (`main.py`)

```
from odoo.addons.web.controllers.main import ensure_db, Home
from odoo.exceptions import AccessDenied, UserError, AccessError
from werkzeug.exceptions import ServiceUnavailable
from odoo import api, http, SUPERUSER_ID
from base64 import b64encode, b64decode
from werkzeug import url_encode
from json import loads, dumps
from hashlib import md5
from uuid import uuid1
from time import time
import logging
import sys

try:
    import jwt
    import requests
except:
    pass

_logger = logging.getLogger(__name__)

class Login(Home):
    def get_provider(self):
        try:
            provider = http.request.env['auth.sso.myits.provider'].sudo().search_read([('enabled', '=', True)])
        except Exception:
            provider = []
        for p in provider:
            p['auth_link'] = "/auth_sso_myits/signin"
        return provider

    @http.route()
    def web_login(self, *args, **kw):
        ensure_db()
        if http.request.httprequest.method == 'GET' and http.request.session.uid and http.request.params.get('redirect'):
            # Redirect if already logged in and redirect param is present
```

```
37.             return http.redirect_with_hash(request.params.get(
    'redirect'))
38.         provider = self.get_provider()
39.         response = super(Login, self).web_login(*args, **kw)
40.         if response.is_qweb:
41.             error = http.request.params.get('auth_error')
42.             if error == '1':
43.                 error = 'User is not authorized in this system
    '
44.             elif error == '2':
45.                 error = http.request.params.get('auth_error_de
    scription')
46.                 if not error:
47.                     error = 'Unknown error occured'
48.             else:
49.                 error = None
50.             response.qcontext['provider'] = provider
51.             if error:
52.                 response.qcontext['error'] = error
53.         return response
54.
55. class Main(http.Controller):
56.     def __init__(self):
57.         base_url = http.request.env['ir.config_parameter'].sud
    o().get_param('web.base.url')
58.         self.__redirect_url = base_url + '/auth_sso_myits/sign
    in'
59.         self.__leeway = 300
60.
61.     def __fetch_url(self, url, post_body = None, header = {}):

62.         try:
63.             if post_body is not None:
64.                 header['Content-Type']= 'application/x-www-
    form-urlencoded'
65.                 output = requests.post(url, data = post_body,
    headers = header, timeout=10, verify = False)
66.             else:
67.                 output = requests.post(url, headers = header,
    timeout=10, verify = False)
68.         except requests.exceptions.Timeout:
69.             desc = ('Connection error with provider : Request
    Time Out')
```

```
70.             raise UserError, desc, sys.exc_info()[2]
71.         except Exception, e:
72.             desc = ('Connection error with provider (%s)' % e)

73.             raise UserError, desc, sys.exc_info()[2]
74.         return output.text
75.
76.     def __get_well_known_config_value(self, provider_url):
77.         well_known_config_url = provider_url.rstrip('/') + '/.
    well-known/openid-configuration'
78.         well_known_json = self.__fetch_url(well_known_config_u
    rl)
79.         try:
80.             well_known = loads(well_known_json)
81.         except Exception as e:
82.             desc = ('Error get well known config value (%s)' %
     e)
83.             raise UserError, desc, sys.exc_info()[2]
84.         if not well_known:
85.             desc = 'Error get well known config value'
86.             raise UserError(desc)
87.         return well_known
88.
89.     def __request_tokens(self, provider, code, well_known):
90.         headers = {}
91.         grant_type = "authorization_code"
92.         token_params = {'grant_type' : grant_type, 'code' : co
    de, 'redirect_uri' : self.__redirect_url, 'client_id' : provid
    er['client_id'], 'client_secret' : provider['client_secret']}

93.         if 'client_secret_basic' in well_known['token_endpoint
    _auth_methods_supported']:
94.             headers['Authorization'] = 'Basic ' + b64encode( p
    rovider['client_id'] + ':' + provider['client_secret'])
95.             token_params.pop('client_secret')
96.         token = self.__fetch_url(well_known['token_endpoint'],
     token_params, headers)
97.         if token and token.strip():
98.             try:
99.                 token = loads(token)
100.            except ValueError:
101.                token = token.split('{')[-1]
102.                token = '{' + token
```

```
103.                 try:
104.                     token = loads(token)
105.                 except Exception:
106.                     desc = 'Error decoding token.'
107.                     raise UserError(desc)
108.             return token
109.         else:
110.             desc = 'Failed to obtain token.'
111.             raise UserError(desc)
112.
113.     def __get_state(self):
114.         try:
115.             return http.request.session.get('openid_connect_st
    ate')
116.         except Exception:
117.             return False
118.
119.     def __decode_jwt(self, jwtok, section = 0):
120.         parts = jwtok.split('.')
121.         return loads(b64decode(parts[section] + "==="))
122.
123.     def __get_key_for_headers(self, keys, header):
124.         for key in keys:
125.             if key['kty'] == 'RSA':
126.                 if 'kid' not in header or key['kid'] == header
    ['kid']:
127.                     return key
128.             else:
129.                 if key['alg'] == header['alg'] and key['kid']
    == header['kid']:
130.                     return key
131.         if 'kid' in header:
132.             desc = 'Unable to find a key for (algorithm, kid):
    '  + header['alg'] +  ', ' + header['kid']
133.         else:
134.             desc = 'Unable to find a key for RSA'
135.         raise UserError(desc)
136.
137.     def __verify_jwt_signature(self, provider, jwtok, jwks_uri
    ):
138.         parts = jwtok.split('.')
139.         header = loads(b64decode(parts[0] + "==="))
140.         try:
```

```
141.             jwks = loads(self.__fetch_url(jwks_uri))
142.         except (TypeError, ValueError) as e:
143.             desc = 'Error decoding JSON from jwks_uri'
144.             raise UserError(desc)
145.         verified = False
146.         if header['alg'] in ['RS256', 'RS384', 'RS512']:
147.             key = self.__get_key_for_headers(jwks['keys'], hea
    der)
148.             public_key = jwt.algorithms.RSAAlgorithm.from_jwk(
    dumps(key))
149.             try:
150.                 jwt.decode(jwtok, key=public_key, audience = p
    rovider['client_id'], algorithms=[header['alg']])
151.             except jwt.InvalidSignatureError:
152.                 verified = False
153.             else:
154.                 verified = True
155.         elif header['alg'] in ['HS256', 'HS384', 'HS512']:
156.             try:
157.                 jwt.decode(jwtok, key=provider['client_secret'
    ],  audience = provider['client_id'], algorithms=[header['alg'
    ]])
158.             except jwt.InvalidSignatureError:
159.                 verified = False
160.             else:
161.                 verified = True
162.         else:
163.             desc = 'No support for signature type: ' + header[
    'alg']
164.             raise UserError(desc)
165.         return verified
166.
167.     def __verify_jwt_claims(self, provider, claims, well_known
    ):
168.         if 'issuer' not in well_known:
169.             well_known['issuer'] = None
170.         return ((claims['iss'] == provider['provider_url'] or
    claims['iss'] == well_known['issuer'] or claims['iss'] == (wel
    l_known['issuer'] + '/'))
171.             and (claims['aud'] == provider['client_id'] or pro
    vider['client_id'] in claims['aud'])
172.             and (claims['nonce'] == http.request.session['open
    id_connect_nonce'])
```

```
173.             and ('exp' not in claims or claims['exp'] >= time(
    ) - self.__leeway)
174.             and ('nbf' not in claims or claims['nbf'] <= time(
    ) + self.__leeway))
175.
176.     def __generate_random_string(self):
177.         return md5(uuid1().hex).hexdigest()
178.
179.     def __set_nonce(self, nonce):
180.         http.request.session['openid_connect_nonce'] = nonce
181.         return nonce
182.
183.     def __set_state(self, state):
184.         http.request.session['openid_connect_state'] = state
185.         return state
186.
187.     def __request_authorization(self, provider, auth_endpoint,
     prompt):
188.         response_type = 'code'
189.         nonce = self.__set_nonce(self.__generate_random_string
    ())
190.         state = self.__set_state(self.__generate_random_string
    ())
191.         auth_params = {
192.             'prompt' : prompt,
193.             'response_type' : response_type,
194.             'redirect_uri' : self.__redirect_url,
195.             'client_id' : provider['client_id'],
196.             'nonce' : nonce,
197.             'state' : state,
198.             'scope' : 'openid'
199.             }
200.         if len(provider['scope']) > 0:
201.             auth_params.update({
202.                 'scope' : ' '.join(provider['scope'].split())

203.                 })
204.         if auth_endpoint.find('?') < 0:
205.             auth_endpoint += '?'
206.         else:
207.             auth_endpoint += '&'
208.         auth_endpoint += url_encode(auth_params)
209.         return http.redirect_with_hash(auth_endpoint)
```

```
210.
211.     def __redirect_session_storage(self, url, query = None):
212.         root = http.request.httprequest.url_root
213.         if url[0] == '/':
214.             root = root[:-1]
215.         url = root + url
216.         url.replace("'", "%27").replace("<", "%3C")
217.         if query:
218.             query = url_encode(query)
219.         else:
220.             query = ""
221.         return ("""
222.         <html>
223.             <head>
224.                 <script>
225.                     var url = '%s';
226.                     var query = '%s';
227.                     if (typeof(Storage) !== "undefined"){
228.                         if (sessionStorage.getItem('query')){
229.                             if(query){
230.                                 query = '&' + query;
231.                             }
232.                             query = sessionStorage.getItem('qu
    ery') + query;
233.                             sessionStorage.removeItem('query')
    ;
234.                         }
235.                         else{
236.                             if(query){
237.                                 query = '?' + query;
238.                             }
239.                         }
240.                         url = url + query;
241.                         if (sessionStorage.getItem('hash')){
242.                             url = url + sessionStorage.getItem
    ('hash');
243.                             sessionStorage.removeItem('hash');

244.                         }
245.                     }
246.                     else{
247.                         if(query){
248.                                 url = '?' + query;
```

```
                        }
                    }
                    window.location.assign(url);
                </script>
            </head>
        </html>""" % (url, query))

    def authenticate(self, provider):
        ensure_db()
        prompt = 'none'
        url_login = '/web/login'
        if not http.request.uid:
            http.request.uid = SUPERUSER_ID
        try:
            if 'error' in http.request.params.keys():
                if http.request.params['error'] == 'login_requ
ired':
                    prompt = 'login'
                else:
                    desc = 'Unknown error from provider.'
                    if 'error_description' in http.request.par
ams.keys():
                        desc = http.request.params['error_desc
ription']
                    raise UserError(desc)
            if 'code' in http.request.params.keys():
                well_known = self.__get_well_known_config_valu
e(provider['provider_url'])
                code = http.request.params['code']
                if 'token_endpoint_auth_methods_supported' not
 in well_known:
                    well_known['token_endpoint_auth_methods_su
pported'] = ['client_secret_basic']
                token_json = self.__request_tokens(provider, c
ode, well_known)
                if 'error' in token_json:
                    desc = 'Unknown error from provider.'
                    if 'error_description' in token_json:
                        desc = token_json['error_description']

                    raise UserError(desc)
                if http.request.params['state'] != self.__get_
state():
```

```
283.                     raise AccessError("Unable to determine sta
    te")
284.                 http.request.session.pop('openid_connect_state
    ')
285.                 if 'id_token' not in token_json:
286.                     raise UserError("User did not authorize op
    enid scope")
287.                 claims = self.__decode_jwt(token_json['id_toke
    n'], 1)
288.                 if not self.__verify_jwt_signature(provider, t
    oken_json['id_token'], well_known['jwks_uri']):
289.                     raise AccessError("Failed to verify signat
    ure")
290.                 if not self.__verify_jwt_claims(provider, clai
    ms, well_known):
291.                     raise AccessError("Failed to verify JWT cl
    aims")
292.                 http.request.session.pop('openid_connect_nonce
    ')
293.                 old_uid = http.request.uid
294.                 user_info_endpoint = well_known['userinfo_endp
    oint']
295.                 credentials = http.request.env['res.users'].su
    do().auth_user(user_info_endpoint, token_json['access_token'])

296.                 uid = http.request.session.authenticate(*crede
    ntials)
297.                 if uid is not False:
298.                     return self.__redirect_session_storage('/w
    eb')
299.                 http.request.uid = old_uid
300.                 raise AccessDenied()
301.             else:
302.                 return self.__request_authorization(provider,
    provider['provider_url'] + '/authorize', prompt)
303.         except AccessDenied:
304.             error = {'auth_error': 1}
305.         except AccessError, e:
306.             _logger.exception("auth_sso_myits: %s" % e)
307.             error = {'auth_error': 1}
308.         except UserError, e:
309.             _logger.exception("auth_sso_myits: %s" % e[0])
```

```
310.             error = {'auth_error': 2, 'auth_error_description'
    : e[0]}
311.         # Error when accessing database
312.         except Exception, e:
313.             _logger.exception("auth_sso_myits: %s" % e[0])
314.             error = {'auth_error': 2, 'auth_error_description'
    : 'An error has occured in server. Please try again.'}
315.         return self.__redirect_session_storage(url_login, erro
    r)
316.
317.     @http.route('/auth_sso_myits/signin', auth='public', type=
    'http')
318.     def auth(self , **kw):
319.         ensure_db()
320.         if http.request.httprequest.method == 'GET' and http.r
    equest.session.uid:
321.             # Redirect if already logged in and redirect param
     is present
322.             return self.__redirect_session_storage('/web')
323.         data_provider = http.request.env['auth.sso.myits.provi
    der'].sudo().search_read([('enabled', '=', True)])
324.         if not data_provider:
325.             _logger.info("No SSO MyITS provider is enabled")
326.             return ServiceUnavailable(description = "SSO MyITS
     is disabled on this system.")
327.         assert len(data_provider) == 1
328.         for provider in data_provider:
329.             return self.authenticate(provider)
```

Kode Sumber S.5 Inisialisasi *Models* (`__init__.py`)

```
1.  import auth_sso_myits_provider
2.  import auth_sso_myits_res_users
```

Kode Sumber S.6 Model *User* (`auth_sso_myits_res_users.py`)

```
1.  from odoo.exceptions import AccessDenied, UserError
2.  from passlib.hash import pbkdf2_sha512
3.  from odoo import api, fields, models
4.  from odoo.addons import base
5.  from base64 import b64encode
6.  from json import loads
```

```
import threading
import logging
import sys

_logger = logging.getLogger(__name__)

base.res.res_users.USER_PRIVATE_FIELDS.append('auth_sso_myits_openid_identifier')

try:
    import requests
except:
    pass

class SSOMyITSResUsers(models.Model):
    _inherit = 'res.users'

    auth_sso_myits_identity_no = fields.Char(string='Identity No.', help=" User identity number on SSO MyITS. ", default=None)
    auth_sso_myits_openid_identifier = fields.Char(hidden=True, readonly=True, default=None)
    auth_sso_myits_enabled = fields.Boolean(string='Enable on This User', help="Enable or disable SSO MyITS on this user.", default=False) #Enable or disable provider
    auth_sso_myits_verified = fields.Boolean(string='Verified', help="Checked if user has verify by first login attempt.", readonly=True, default=False) #Enable or disable provider

    _sql_constraints = [
        ('unique_auth_sso_myits_identity_no', 'unique(auth_sso_myits_identity_no)', 'Identity number must be unique.'),
        ('unique_auth_sso_myits_openid_identifier', 'unique(auth_sso_myits_openid_identifier)', 'OpenID identifier must be unique.'),
    ]

    @api.multi
    def write(self, vals):
        if 'auth_sso_myits_identity_no' in vals:
            vals['auth_sso_myits_openid_identifier'] = None
            vals['auth_sso_myits_verified'] = False
        return super(SSOMyITSResUsers, self).write(vals)
```

```
39.
40.     @api.model
41.     def __request_user_info(self, user_info_endpoint, access_t
    oken):
42.         schema = 'openid+profile'
43.         user_info_endpoint += '?schema=' + schema
44.         header = {'Authorization' : 'Bearer ' + access_token}

45.         try:
46.             response = requests.post(user_info_endpoint, heade
    rs = header, timeout=10, verify = False)
47.         except requests.exceptions.Timeout:
48.             desc = ('Connection error with provider : Request
    Time Out')
49.             raise UserError, desc, sys.exc_info()[2]
50.         except Exception, e:
51.             desc = ('Connection error with provider (%s)' % e)

52.             raise UserError, desc, sys.exc_info()[2]
53.         return loads(response.text)
54.
55.     @api.model
56.     def __auth_login(self, user_info):
57.         user = self.search([("auth_sso_myits_identity_no", "="
    , user_info['reg_id']), ("auth_sso_myits_enabled", "=", True)]
    )
58.         if not user:
59.             raise AccessDenied()
60.         assert len(user) == 1
61.         if user.auth_sso_myits_openid_identifier and user.auth
    _sso_myits_verified:
62.             try:
63.                 if not (pbkdf2_sha512.verify(user_info['sub'],
     user.auth_sso_myits_openid_identifier)):
64.                     raise AccessDenied()
65.             except Exception:
66.                 raise AccessDenied()
67.         else:
68.             try:
69.                 hashed_sub = pbkdf2_sha512.hash(user_info['sub
    '])
70.             except AttributeError:
71.                 try:
```

```
                    hashed_sub = pbkdf2_sha512.encrypt(user_info['sub'])
                except Exception:
                    raise AccessDenied()
            except Exception:
                    raise AccessDenied()
            sso_email = user_info.get('email') if user_info.get('email') else user_info.get('alternate_email')
            name = user_info.get('name')
            phone = user_info.get('phone')
            picture_url = user_info.get('picture')
            if picture_url and picture_url.strip():
                try:
                    image = requests.get(picture_url, timeout=10, verify=False)
                    image = b64encode(image.content)
                except requests.exceptions.Timeout:
                    _logger.exception("auth_sso_myits: Failed to fetch profile picture. Use default image instead.")
                    image = None
            else:
                image = None
            lock = threading.RLock()
            with lock:
                try:
                    user.write({'auth_sso_myits_openid_identifier': hashed_sub, 'auth_sso_myits_verified': True})
                    if sso_email and sso_email.strip():
                        if len(self.search([("login", "=", sso_email), ("id", "!=", user.id)])) == 0:
                            user.write({'login': sso_email})
                        user.write({'email': sso_email})
                    if name and name.strip():
                        user.write({'name': name + ' - ' + user_info['reg_id']})
                    if phone and phone.strip():
                        user.write({'phone': phone})
                    elif user.phone:
                        user.write({'phone': None})
                    if image:
                        user.write({'image': image})
                    else:
```

```
107.                         image = user.partner_id._get_default_i
    mage(user.partner_id.type, user.is_company, user.parent_id)
108.                         user.write({'image': image})
109.                     self.env.cr.commit()
110.                 except:
111.                     self.env.cr.rollback()
112.                     raise
113.         return user.login
114.
115.     @api.model
116.     def auth_user(self, user_info_endpoint, access_token):
117.         user_info = self.__request_user_info(user_info_endpoin
    t, access_token)
118.         if not user_info.get('sub'):
119.             raise AccessDenied()
120.         login = self.__auth_login(user_info)
121.         if not login:
122.             raise AccessDenied()
123.         return (self.env.cr.dbname, login, user_info.get('sub'
    ))
124.
125.     @api.model
126.     def check_credentials(self, password):
127.         try:
128.             return super(SSOMyITSResUsers, self).check_credent
    ials(password)
129.         except AccessDenied, e:
130.             res = self.sudo().search([('id', '=', self._uid),
    ("auth_sso_myits_enabled", "=", True), ("auth_sso_myits_verifi
    ed", "=", True)])
131.             if not res:
132.                 raise e
133.             try:
134.                 if not (pbkdf2_sha512.verify(password, res.aut
    h_sso_myits_openid_identifier)):
135.                     raise e
136.             except Exception:
137.                 raise e
138.
139.     def _get_session_token_fields(self):
140.         return super(SSOMyITSResUsers, self)._get_session_toke
    n_fields() | {'auth_sso_myits_openid_identifier'}
```

Kode Sumber S.7 Model *Provider* (`auth_sso_myits_provider.py`)

```
1.  from odoo.exceptions import Warning
2.  from odoo import api, fields, models
3.
4.  class AuthSSOMyITSProvider(models.Model):
5.
6.      _name = 'auth.sso.myits.provider'
7.      _description = 'SSO MyITS provider'
8.      _order = 'client_id'
9.
10.     client_id = fields.Char(string='Client ID', required=True)
      # SSO MyITS Client ID
11.     client_secret = fields.Char(string='Client Secret', requir
    ed=True)  # SSO MyITS Client Secret
12.     provider_url = fields.Char(string='Provider URL', required
    =True, default='https://my.its.ac.id')  # SSO MyITS URL to aut
    henticate users
13.     scope = fields.Char(string='Scope', help='Divide multiple
    scopes by space (" ")', required=True, default='openid profile
     email phone')  # SSO MyITS user data desired to access
14.     enabled = fields.Boolean(string='Enabled') #Enable or disa
    ble provider
15.
16.     @api.model
17.     def create(self, vals):
18.         if not (set(['openid', 'profile', 'email']).issubset(s
    et(vals['scope'].split()))):
19.             raise Warning('Scope must contain atleast "openid
    profile email".')
20.         if vals['enabled'] == True:
21.             self._disable_active_provider()
22.         return super(AuthSSOMyITSProvider, self).create(vals)
23.
24.     @api.multi
25.     def write(self, vals):
26.         self.ensure_one()
27.         if 'scope' in vals and not (set(['openid', 'profile',
    'email']).issubset(set(vals['scope'].split()))):
28.             raise Warning('Scope must contain atleast "openid
    profile email".')
```

```
        if 'enabled' in vals and vals['enabled'] == True:
            self._disable_active_provider()
        return super(AuthSSOMyITSProvider, self).write(vals)

    @api.model
    def _disable_active_provider(self):
        active_provider = self.search([('enabled', '=', True)]
)
        for provider in active_provider:
            provider.write({'enabled': False})
```

Kode Sumber S.8 *View User* (auth_sso_myits_res_users_view.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <record id="view_sso_myits_users_form" model="ir.ui.view">

        <field name="name">res.users.auth.sso.myits</field>
        <field name="model">res.users</field>
        <field name="type">form</field>
        <field name="inherit_id" ref="base.view_users_form"/>

        <field name="arch" type="xml">
            <xpath expr="//page[@name='access_rights']" positi
on="after">
                <page string="SSO MyITS" groups="base.group_sy
stem">
                    <group string="Identifier" name="identifie
r">
                        <field name="auth_sso_myits_identity_n
o"/>
                        <field name="auth_sso_myits_enabled"/>

                    </group>
                    <group string="Status" name="Status">
                        <field name="auth_sso_myits_verified"/
>
                    </group>
```

```
                </page>
            </xpath>
        </field>
    </record>
</odoo>
```

Kode Sumber S.9 *View Provider* (auth_sso_myits_provider_view.xml)

```
<?xml version="1.0"?>
<odoo>
    <record id="view_sso_myits_provider_form" model="ir.ui.view">
        <field name="name">auth.sso.myits.provider.form</field>
        <field name="model">auth.sso.myits.provider</field>
        <field name="arch" type="xml">
            <form string="arch">
                <sheet>
                    <group>
                        <field name="client_id" />
                        <field name="client_secret" password="True" />
                        <field name="provider_url" />
                        <field name="scope" />
                        <field name="enabled" />
                    </group>
                </sheet>
            </form>
        </field>
    </record>
    <record id="view_sso_myits_provider_tree" model="ir.ui.view">
        <field name="name">auth.sso.myits.provider.tree</field>
        <field name="model">auth.sso.myits.provider</field>
        <field name="arch" type="xml">
            <tree string="arch">
                <field name="client_id" />
                <field name="provider_url" />
                <field name="scope" />
                <field name="enabled" />
```

```
            </tree>
        </field>
    </record>
    <record id="action_sso_myits_provider" model="ir.actions.act_window">
        <field name="name">SSO MyITS Provider</field>
        <field name="res_model">auth.sso.myits.provider</field>
        <field name="view_type">form</field>
        <field name="view_mode">tree,form</field>
    </record>
    <menuitem id="menu_sso_myits_provider" name="SSO MyITS Provider"
        parent="base.menu_users" sequence="10"
        action="action_sso_myits_provider" groups="base.group_no_one"/>
</odoo>
```

Kode Sumber S.10 *View Login* (`auth_sso_myits_custom_login_template.xml`)

```
<odoo>
    <data>
        <template id="auth_sso_myits.assets_login" inherit_id="web.assets_frontend">
            <xpath expr="." position="inside">
                <link rel="stylesheet" href="/auth_sso_myits/static/src/css/myits-button.css"/>
            </xpath>
        </template>
        <template id="auth_sso_myits.login" inherit_id="web.login" name="SSO MyITS Login button">
                <xpath expr="//div[contains(@class, 'oe_login_buttons')]" position="after">
                    <div t-foreach="provider" t-as="p">
                            <div class="clearfix myits-button-dark" style="margin-top: 10px; margin-left: auto; margin-right: auto; cursor: pointer;" onclick="ssoRedirect()">
                                <div class="myits-logo"></div>
                                <div class="myits-label">Masuk dengan myITS</div>
                            </div>
```

```
                    </div>
                    <script type="text/javascript">
                        function removeQuery(query, parameter)
    {
                            if (query) {
                                if(query.charAt(0) === '?')
                                {
                                    queryquery = query.substr(
    1);
                                }
                                var param = encodeURIComponent
    (parameter) + '=';
                                var queryquery_array = query.s
    plit(/[&]/g);
                                for (var i = query_array.lengt
    h; i-- > 0;) {
                                    if (query_array[i].lastInd
    exOf(param, 0) !== -1) {
                                        query_array.splice(i,
    1);
                                    }
                                }
                                return (query_array.length > 0
     ? '?' + query_array.join('&') : '');
                            }
                            return query;
                        }

                        function ssoRedirect(){
                            var auth_link = window.location.or
    igin + "<t t-foreach="provider" t-as="p"><t t-
    esc="p['auth_link']"/></t>";
                            if (typeof(Storage) !== "undefined
    "){
                                if (window.location.search){
                                    var q = removeQuery(window
    .location.search, "auth_error")
                                    q = removeQuery(q, "auth_e
    rror_description")
                                    sessionStorage.setItem("qu
    ery", q);
                                }
                                else{
```

```
                                        sessionStorage.removeItem('query');
                                    }
                                    if (window.location.hash){
                                        sessionStorage.setItem("hash", window.location.hash);
                                    }
                                    else{
                                        sessionStorage.removeItem('hash');
                                    }
                                }
                                window.location.assign(auth_link);

                            }
                        </script>
                    </xpath>
                </template>
            </data>
</odoo>
```

Kode Sumber S.11 Deskripsi Modul (`index.html`)

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <h2 class="oe_slogan" style="color: #013880; opacity: 1;"><strong>SSO MyITS Odoo Login</strong></h2>
        <div class="oe_span12">
            <p>
                Modul ini adalah modul Odoo SSO MyITS Odoo Login. Modul ini memungkinkan pengguna untuk <i>login</i> ke dalam sistem menggunakan akun MyITS. Pengguna akan diarahkan untuk melakukan autentikasi melalui SSO MyITS di https://my.its.ac.id/.
            </p>
            <p class="alert alert-warning">
                NOTE : Disarankan melakukan <i>backup database</i> terlebih dahulu sebelum memasang atau memperbarui modul ini.
            </p>
        </div>
        <div class="oe_span12">
```

```
13.             <h3 class="oe_slogan" style="color: #000000; opaci
    ty: 1; margin-top: 16px; margin-bottom: 16px; text-
    align: left;"><i><strong>Dependencies</strong></i></h3>
14.             <p>
15.                 Modul ini memerlukan <i>Python libraries</i> t
    ambahan versi terbaru yaitu <code>pyopenssl</code>, <code>cryp
    tography</code>, <code>pyjwt</code> dan <code>requests</code>.
     Anda dapat memasangnya dengan menjalankan perintah:
16.             </p>
17.             <pre><code>pip install -
    U pyopenssl cryptography pyjwt requests</code></pre>
18.         </div>
19.         <div class="oe_span12">
20.             <h3 class="oe_slogan" style="color: #000000; opaci
    ty: 1; margin-top: 16px; margin-bottom: 16px; text-
    align: left;"><strong>Pengaturan</strong></h3>
21.             <h4 style='font-
    family: "Lato", "Open Sans", "Helvetica", Sans; color: #000000
    ; margin-top: 16px; margin-
    bottom: 16px;'><strong>Pengaturan <i>Provider</i></strong></h4
    >
22.             <p>
23.                 Untuk melakukan pengaturan <i>provider</i>, an
    da perlu mengaktifkan <em>developer mode</em> yang dapat diaks
    es di <em>Settings → Activate the developer mode</em>.
24.             </p>
25.             <ul class="list-unstyled">
26.                 <li>
27.                     <h5 class="oe_slogan" style="color: #00000
    0; opacity: 1; margin-top: 10px; margin-bottom: 10px; text-
    align: left;"><strong>Mengaktifkan <i>Provider</i></strong></h
    5>
28.                     <ol>
29.                         <li>
30.                             Buka <em>Settings → Users → SSO My
    ITS Provider</em>.
31.                         </li>
32.                         <li>
33.                             Klik <em>Create</em> untuk membuat
     pengaturan baru, atau pilih pengaturan yang sudah ada lalu kl
    ik <em>Edit</em>.
34.                         </li>
35.                         <li>
```

```
                            Masukkan <em>Client ID</em>, <em>Client Secret</em>, <em>Provider URL</em>, <em>Scope</em> (minimal "openid profile email"), dan beri centang pada bagian <em>Enabled</em>.
                        </li>
                        <li>
                            Klik <em>Save</em>. Pengaturan <i>provider</i> lain yang sedang aktif akan otomatis dinonaktifkan.
                        </li>
                    </ol>
                </li>
                <li>
                    <h5 class="oe_slogan" style="color: #000000; opacity: 1; margin-top: 10px; margin-bottom: 10px; text-align: left;"><strong>Menonaktifkan <i>Provider</i></strong></h5>
                    <ol>
                        <li>
                            Buka <em>Settings → Users → SSO My ITS Provider</em>.
                        </li>
                        <li>
                            Pilih pengaturan <i>provider</i> yang sedang aktif, lalu klik <em>Edit</em>.
                        </li>
                        <li>
                            Hilangkan centang pada bagian <em>Enabled</em>.
                        </li>
                        <li>
                            Klik <em>Save</em>.
                        </li>
                    </ol>
                </li>
            </ul>
            <h4 style='font-family: "Lato", "Open Sans", "Helvetica", Sans; color: #000000; margin-top: 16px; margin-bottom: 16px;'><strong>Pengaturan <i>User</i></strong></h4>
            <ul class="list-unstyled">
                <li>
```

```
64.                     <h5 class="oe_slogan" style="color: #00000
    0; opacity: 1; margin-top: 10px; margin-bottom: 10px; text-
    align: left;"><strong>Mengaktifkan SSO MyITS pada <i>User</i><
    /strong></h5>
65.                     <ol>
66.                         <li>
67.                             Buka <em>Settings → Users → Users<
    /em>.
68.                         </li>
69.                         <li>
70.                             Pilih <em>User</em> yang ingin dia
    ktifkan, lalu klik <em>Edit</em>.
71.                         </li>
72.                         <li>
73.                             Buka tab <em>SSO MyITS</em>.
74.                         </li>
75.                         <li>
76.                             Masukkan <em>Identity
    No.</em> (nomor identitas pengguna di SSO MyITS) dan beri
    centang pada bagian <em>Enabled on This User</em>.
77.                         </li>
78.                         <li>
79.                             Klik <em>Save</em>. Indikator stat
    us <em>Verified</em> akan otomatis tercentang saat <em>User</e
    m> terhubung dengan akun SSO MyITS setelah berhasil <i>login</
    i> pertama kali.
80.                         </li>
81.                     </ol>
82.                 </li>
83.                 <li>
84.                     <h5 class="oe_slogan" style="color: #00000
    0; opacity: 1; margin-top: 10px; margin-bottom: 10px; text-
    align: left;"><strong>Menonaktifkan SSO MyITS pada <i>User</i>
    </strong></h5>
85.                     <ol>
86.                         <li>
87.                             Buka <em>Settings → Users → Users<
    /em>.
88.                         </li>
89.                         <li>
90.                             Pilih <em>User</em> yang ingin din
    onaktifkan, lalu klik <em>Edit</em>.
91.                         </li>
```

```
92.                             <li>
93.                                 Buka tab <em>SSO MyITS</em>.
94.                             </li>
95.                             <li>
96.                                 Hilangkan centang pada bagian <em>
    Enabled on This User</em>.
97.                             </li>
98.                             <li>
99.                                 Klik <em>Save</em>.
100.                            </li>
101.                        </ol>
102.                </li>
103.                <li>
104.                    <h5 class="oe_slogan" style="color: #00000
    0; opacity: 1; margin-top: 10px; margin-bottom: 10px; text-
    align: left;"><strong>Mengganti akun SSO MyITS pada <i>User</i
    ></strong></h5>
105.                    <ol>
106.                        <li>
107.                            Buka <em>Settings → Users → Users<
    /em>.
108.                        </li>
109.                        <li>
110.                            Pilih <em>User</em> yang ingin dig
    anti, lalu klik <em>Edit</em>.
111.                        </li>
112.                        <li>
113.                            Buka tab <em>SSO MyITS</em>.
114.                        </li>
115.                        <li>
116.                            Ganti <em>Identity No.</em> dengan
    nomor identitas pengguna yang baru.
117.                        </li>
118.                        <li>
119.                            Klik <em>Save</em>. Indikator stat
    us <em>Verified</em> akan otomatis tercentang saat <em>User</e
    m> terhubung dengan akun SSO MyITS yang baru setelah berhasil
    <i>login</i> pertama kali.
120.                        </li>
121.                    </ol>
122.                </li>
123.            </ul>
124.        </div>
```

```
    </div>
</section>
```

Kode Sumber S.12 *Style* Tombol Di *View Login* (myits-button.css)

```
@import"../fonts/Roboto Medium/Roboto Medium.css";
.myits-button{
    background:#fff;
    width:275px;
    height:50px;
    box-shadow:0 0 2px 0 rgba(0,0,0,.25)
    border-radius:3px;
    cursor:pointer;
    overflow:hidden;
    -webkit-transition:all .25s;
    -moz-transition:all .25s;
    -ms-transition:all .25s;
    -o-transition:all .25s;
    transition:all .25s
}
.myits-button:hover{
    box-shadow:0 0 10px rgba(0,103,172,.5)
}
.myits-button .myits-logo{
    background:#0067ac;
    background-image:url("../images/its-logo.png");
    background-repeat:no-repeat;
    background-size:30px 30px;
    background-position:50% 50%;
    width:50px;
    height:50px;
    float:left
}
.myits-button .myits-label{
    font-family:"robotomedium",sans-serif;
    text-align:center;
    color:#0067ac;
    line-height:50px;
    width:225px;
    height:50px;
    float:left
}
```

```
.myits-button-dark{
    background:#0067ac;
    width:275px;
    height:50px;
    box-shadow:0 0 2px 0 rgba(0,0,0,.25);
    border-radius:3px;
    cursor:pointer;
    overflow:hidden;
    -webkit-transition:all .25s;
    -moz-transition:all .25s;
    -ms-transition:all .25s;
    -o-transition:all .25s;
    transition:all .25s
}
.myits-button-dark:hover{
    box-shadow:0 0 10px rgba(0,103,172,.5)
}
.myits-button-dark .myits-logo{
    background:#fff;
    background-image:url("../images/its-logo-dark.png");
    background-repeat:no-repeat;
    background-size:30px 30px;
    background-position:50% 50%;
    width:50px;
    height:50px;
    float:left
}
.myits-button-dark .myits-label{
    font-family:"robotomedium",sans-serif;
    text-align:center;
    color:#fff;
    line-height:50px;
    width:225px;
    height:50px;
    float:left
}
```

Kode Sumber S.13 *Security* (`ir.model.access.csv`)

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_auth_sso_myits_provider,auth_sso_myits_provider,model_auth_sso_myits_provider,base.group_system,1,1,1,1
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Hanif Nashrullah, lahir di Jakarta pada tanggal 27 Juli 1997. Penulis merupakan anak bungsu dari 2 bersaudara. Penulis telah menempuh pendidikan formal di TK Kartika X-13 KPAD Cibubur Jakarta Timur (2002-2003), SDS Kartika XIII-1 KPAD Cibubur Jakarta Timur (2003-2009), SMP Negeri 49 Jakarta (2009-2012), SMA Negeri 39 Jakarta (2012-2015), dan kemudian penulis melanjutkan pendidikan S1 di Departemen Teknik Informatika ITS Surabaya. Selama kuliah, penulis aktif di UKM ITS Foreign Language Society (ITS) dan telah beberapa kali menjadi pengurus yaitu menjadi Staf Divisi Jurnalistik UKM IFLS (2016-2017) dan Staf Ahli Departemen Kominfo UKM IFLS (2017-2018). Penulis juga aktif dalam kepanitiaan di beberapa acara yang diadakan di ITS, diantaranya adalah Panitia Bagian Seminar di Parade KMI 2016, Panitia Divisi Acara INOCHI 2016 dan 2017, Panitia Divisi Acara K-FEST 2016, serta Panitia Divisi Kamjintrans di K-FEST 2017. Penulis dapat dihubungi melalui nomor telepon +6282145102797 dan *email* hn3692@gmail.com.