ក្រសួងអប់រំ យុវជន និងកីឡា

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និងទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វករ

ប្រធានបទ : ប្រព័ន្ធប្រមូលផ្តុំ និងគ្រប់គ្រងព័ត៌មានអ្នកប្រើប្រាស់

និស្សិត : រដ្ឋា ជាប្រាវីត

ឯកទេស : វិស្វកម្មព័ត៌មានវិទ្យា និងទំនាក់ទំនង

គ្រូទទួលបន្ទុក : លោក ណុប តារម្យ

ឆ្នាំសិក្សា : ២០២៣ - ២០២៤

MINISTERE DE L'EDUCATION,
DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIND D'ETUDES

Titre : **Système de collecte et de gestion des informations sur les utilisateurs**

Etudiant : **ROTHA Dapravith**

Spécialité : **Génie Informatique et Communication**

Tuteur de stage : **M. NOP Phearum**

Année scolaire : **2023-2024**

ក្រសួងអប់រំ យុវជន និងកីឡា

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និងទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វករ

របស់និស្សិត: រដ្ឋា ជាប្រាថីត

កាលបរិច្ឆេទការពារនិក្ខេបបទ: ថ្ងៃទី ១០ ខែ កក្កដា ឆ្នាំ ២០២៤

អនុញ្ញាតឲ្យការពារគម្រោង

នាយកវិទ្យាស្ថាន:_____

ថ្ងៃទី ខែ ឆ្នាំ ២០២៤

ប្រធានបទ : ប្រព័ន្ធប្រមូលផ្តុំ និងគ្រប់គ្រងព័ត៌មានអ្នកប្រើប្រាស់

សហគ្រាស : ក្រសួងសេដ្ឋកិច្ច និង ហិរញ្ញវត្ថុ

ប្រធានដេប៉ាតឺម៉ង់ : លោក ឡៃ ហេង_____

សាស្ត្រាចារ្យដឹកនាំគម្រោង : លោក ណុប តារម្យ _____

អ្នកទទួលខុសត្រូវក្នុងសហគ្រាស : លោក ង៉ុន សីហា_____

រាជធានីភ្នំពេញ ឆ្នាំ២០២៤

MINISTERE DE L'EDUCATION, DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIN D'ETUDES

DE M. ROTHA Dapravith

Date de soutenance: le 10 Juillet 2024

« Autorise la soutenance du mémoire »

Directeur de l'Institut :_____

Phnom Penh, 2024

Titre : **Système de collecte et de gestion des informations sur les utilisateurs**

Etablissement du stage : **Ministère de l'Economie et des Finances**

Chef du département : **M. LAY Heng** _____

Tuteur de stage : **M. NOP Phearum** _____

Responsable de l'établissement : **M. VON Seyha** _____

PHNOM PENH, 2024

ACKNOWLEDGEMENTS

This report would not have been published without the guidance and assistance of several people who always provide a good solution and extended their valuable assistance in the preparation and implementation of this project.

To begin with, I would like to express our deepest gratitude to my parents who always provide me with all kinds of support me everything not only giving me an opportunity to study, money and times but also telling me being a good human resource in the society, being person who have morality and responsibility.

Firstly, I would like to appreciate my gratitude to **H.E. Dr. PO Kimtho,** General Director of Institute of Technology of Cambodia, for his good management of the institute and his good cooperation with the partner universities at the local, regional, and international levels, to enhance the quality of the training of engineers and senior technicians.

Secondly, I wish to express in advance thanks to **Mr. LAY Heng**, Head of the Department of Information Technology and Communication at the Institute of Technology of Cambodia, for his good management and is an integral part of the teaching.

Thirdly, I would like to express appreciation to **Mr. NOP Phearum**, lecturer at ITC and as my academic supervisor, for the time he has devoted to me during this period for giving the solutions concerning internship and having answered our questions with participation in the progress of ourwork. We also greet of him for his remarks and corrections which allow me to write our final dissertation.

Fourthly, I would like to greatly thank to **Mr. VON Seyha,** my organization advisor and project manager at **Legal Council of Ministry of Economy and Finance**, who has giving me a lot of new knowledge, advice, work experience in these three months during I worked there.

Additionally, all lecturer in the department Information and Communication Engineering for knowledge they gave us during our studies at the Institute of Technology of Cambodia.

Finally, I would like to thank all the people who participated and helped us to write this report and build this project until I can complete it successfully.

មូលន័យសង្ខេប

ក្នុងអំឡុងពេលបីខែនៃការចុះកម្មសិក្សារបស់ខ្ញុំនៅ **ក្រុមប្រឹក្សានីតិកម្មនៃក្រសួងសេដ្ឋកិច្ចនិងហិរញ្ញវត្ថុ** រាប់ចាប់ពីថ្ងៃទី **១៩ ខែកុម្ភៈ ដល់ថ្ងៃទី ៣១ ខែឧសភា ឆ្នាំ២០២៤** ខ្ញុំបានធ្វើលើគម្រោងមួយដែលមានឈ្មោះថា**ប្រព័ន្ធប្រមូលផ្តុំនិងគ្រប់គ្រងព័ត៌មានអ្នកប្រើប្រាស់**។គោលបំណងនៃការអភិវឌ្ឍន៍ប្រព័ន្ធនេះ គឺដើម្បីតាមដានការចូលប្រើប្រាស់គណនីរបស់អ្នកប្រើប្រាស់ ងាយស្រួលក្នុងការគ្រប់គ្រង និង ប្រមូលផ្តុំនូវទិន្នន័យរបស់អ្នកប្រើប្រាស់នៅក្នុងប្រព័ន្ធតែមួយ ហើយប្រព័ន្ធនេះក៏ពង្រឹងសុវត្ថិភាពការពារទិន្នន័យអ្នកប្រើប្រាស់។

ចំពោះគម្រោងនេះគឺធ្វើឱ្យអ្នកប្រើប្រាស់អាចចូលប្រើប្រាស់ប្រព័ន្ធដោយប្រើបច្ចេកវិទ្យា **Keycloak** ដែលអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់ផ្សៀងផ្គាត់ចូល និងចាកចេញពីគណនីតែមួយដើម្បីចូលទៅកាន់កម្មវិធីផ្សេងៗបានតាមតម្រូវការ ប្រព័ន្ធក៏បានធ្វើការរក្សាទុក និងប្រមូលផ្តុំទិន្នន័យទាំងអស់នៅក្នុងប្រព័ន្ធនេះ។

ក្នុងនិក្ខេបបទនេះ ខ្ញុំបានពន្យល់អំពីប្រព័ន្ធ **Single Sign-On (SSO)** ដែលអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់ចូលប្រើប្រាស់ប្រព័ន្ធផ្សេងៗដោយប្រើគណនីតែមួយ។ ដំណើរការរបស់ប្រព័ន្ធ **SSO** រួមមានការចូលតាម **Identity Provider (IdP)** ដែលបញ្ជាក់អត្តសញ្ញាណអ្នកប្រើប្រាស់ និងផ្តល់ **Access Token** ឬ **ID Token** សម្រាប់ចូលសេវាកម្មផ្សេងៗដោយមិនចាំបាច់បញ្ចូលព័ត៌មានចូលប្រើប្រាស់ម្តងទៀត។

បន្ទាប់មក ខ្ញុំបាននិយាយអំពីដំណើរការស្ម័យប្រវត្តិកម្ម ក្នុងការគ្រប់គ្រងគម្រោងដោយបញ្ចូនប្រភពកូដ ទៅ **GitLab** និង **SourceTree** ដើម្បីដាក់ប្រើប្រាស់នៅលើ **Digital Ocean Server**។ ខ្ញុំបានប្រើ **Nginx** សម្រាប់បង្ហោះគេហទំព័រ និងគ្រប់គ្រងផ្នែកខាងមុខដោយ **Angular, ReactJS** និង **API** ដោយប្រើ **NestJS**។ ជាបន្ថែមទៀត ខ្ញុំក៏ប្រើ **MongoDB** និង **MySQL** សម្រាប់ផ្ទុកទិន្នន័យ និង **Certbot Let's Encrypt** សម្រាប់សុវត្ថិភាព។ លើសពីនេះ ខ្ញុំក៏បានប្រើ **Docker, PM2 Server, Portainer.io, Ngrok,** និង **Jenkins** សម្រាប់ដំណើរការផ្សេងៗ និង **CI/CD** ដោយប្រើ Jenkins ដើម្បីធានាបានថាប្រព័ន្ធដំណើរការយ៉ាងរលូននិង **Telegram bot** សម្រាប់ជាសារជូនដំណឹងបន្ទាប់ពីដំណើរការនៃ CI/CD បានបញ្ចប់។

RÉSUMÉ

Durant mon stage de trois mois au Conseil juridique du ministère de l'Économie et des Finances, du 19 février au 31 mai 2024, j'ai travaillé sur un projet appelé Système d'authentification unique. L'objectif principal de ce système était de suivre l'accès aux comptes des utilisateurs, de rationaliser la gestion et de consolider les données des utilisateurs dans un système unifié, tout en renforçant la sécurité des données. J'ai mis en œuvre la technologie Keycloak pour permettre aux utilisateurs de s'authentifier et de se déconnecter à partir d'un seul compte, donnant accès à plusieurs applications selon les besoins, et géré le stockage et l'agrégation de toutes les données.

Dans le cadre de cette thèse, j'explique le système d'authentification unique (SSO) qui permet aux utilisateurs d'accéder à différents systèmes à l'aide d'un seul compte. Le processus SSO implique une connexion par l'intermédiaire d'un fournisseur d'identité (IdP) qui vérifie les identités des utilisateurs et émet des jetons d'accès, permettant d'accéder à différents services sans avoir à ressaisir les informations de connexion. J'aborde ensuite le processus DevOps, qui consiste à envoyer le code source à GitLab et Source Tree pour le déployer sur Digital Ocean. J'ai utilisé Nginx pour l'hébergement web, géré le front-end avec Angular et ReactJS, et l'API avec NestJS. En outre, j'ai utilisé MongoDB et MySQL pour le stockage des données et Certbot Let's Encrypt pour la sécurité.

Enfin, j'ai utilisé Docker, PM2 Server, Portainer.io, Ngrok et Jenkins pour diverses tâches opérationnelles et mis en œuvre le CI/CD avec Jenkins pour assurer le bon fonctionnement du système, avec un bot Telegram pour les notifications après le processus.

ABSTRACT

During my three-month internship at the Legal Council of the Ministry of Economy and Finance, from February 19 to May 31, 2024, I worked on a project called the Single Sign-On System. The primary objective of this system was to track user account access, streamline management, and consolidate user data into a unified system. Additionally, the system enhanced the security of user data protection.

For this project, I implemented Keycloak technology to allow users to authenticate and log out from a single account, providing access to multiple applications as needed. The system also managed the storage and aggregation of all data.

In this context of the thesis, I will explain the Single Sign-On (SSO) system, which allows users to access different systems using a single account. The SSO process involves logging in through an identity provider (IdP) that verifies user identities and issues access tokens, or ID tokens, enabling access to various services without the need to repeatedly enter login information.

Following this, I discuss the DevOps process, which involved sending source code to GitLab and Source Tree for deployment on Digital Ocean. I used Nginx for web hosting, managed the front end with Angular and ReactJS, and handled the API with NestJS. Additionally, I used MongoDB and MySQL for data storage and Certbot Let's Encrypt for security.

Furthermore, I utilized Docker, PM2 Server, Portainer.io, Ngrok, and Jenkins for various operational tasks. I implemented Continuous Integration/Continuous Deployment (CI/CD) using Jenkins to ensure smooth system operations, with a Telegram bot providing notifications after the CI/CD process was completed.

LIST OF ABBREVIATION

API	:	Application Programming Interface
CD	:	Continuous Delivery/Deployment
CI	:	Continuous Integration
DevOps	:	Development and Operation
DNS	:	Domain Name System
HTTP	:	Hypertext Transfer Protocol
HTTPS	:	Hypertext Transfer Protocol Secures
ITC	:	Institute of Technology of Cambodia
JSON	:	JavaScript Object Notation
LCMEF	:	Legal Council of Ministry of Economy and Finance
MFA	:	Multi-Factor Authentication
NOSQL	:	Non-Relational Structure Query Language
OAuth2.0	:	Open Authorization 2.0
OIDC	:	OpenID Connect
OS	:	Operating System
RBAC	:	Role-Based Access Control
SAML	:	Security Assertion Markup Language
SQL	:	Structure Query Language
SSH	:	Secure Shell
SSL	:	Secure Socket Layer
SLO	:	Single Logout
SSO	:	Single Sign-On
UI	:	User Interface
UML	:	User Modeling Language
VM	:	Virtual Machine
YAML	:	YAML Ain’t Markup Language
OTP	:	One-Time Password
2FA	:	Two-Factors Authentication

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENT

INTRODUCTION

Every fifth-year engineering student at the Department of Information and Communication Engineering (DICE) of the Institute of Technology of Cambodia (ITC) must do an internship for at least three months at any public or private company. After finishing this internship, all students must submit a report and make a slide presentation about their internship to the defense in front of the chairman and juries. according to the requirements of the department and school to get an engineering degree before graduation.

During my internship at the Ministry of Economy and Finance (MEF), I had the opportunity to work as a software developer and DevOps engineer on a project titled **'Single Sign-On Service Management System (SSO) and DevOps'**. The project aimed to simplify the user experience by enabling users to log in once and access multiple applications without having to re-enter their credentials each time. This also enhances security by reducing password fatigue and decreasing administrative tasks related to password management. The last can apply with web client application for secure user authentication with **OAuth 2.0**.

One of my duties involves overseeing DevOps practices for all projects submitted by developers in the version control system using GitLab and SourceTree. This is achieved through CI/CD integration pipelines with GitLab to configure webhooks and Jenkins for automated deployment and release of projects to the server. We utilize Docker to implement this and configure it with Nginx to secure the domain name for both the frontend and API projects with Certbot.

Furthermore, we utilize PM2 server to initiate and operate any applications on a virtual machine running Ubuntu on Digital Ocean. In the final step, it is necessary to configure the Telegram bot to send alerts for every event that occurs, enabling the tracking of all activities provided by the Telegram channel server via the bot.

I. GENERAL PRESENTATION

1.1. Presentation of the Organization

1.1.1. History of the Organization

Figure 1: LC-MEF Logo

The Legal Council of the Ministry of Economy and Finance was established by Prâkas No. 290 SHV.BRK dated March 15, 2013 on Establishing of the Legal Council of the Ministry of Economy and Finance, which was successively revised and was finally modified by Sub-decree No. 75 ANKR.BK dated May 25, 2017 on the Modification of Sub-decree No. 488 ANKR.BK dated October 16, 2013 on Organizing and functioning of the Ministry of Economy and Finance, and of Sub-decree No. 76 ANKR.BK dated June 13, 2018 on Organizing and functioning of the Legal Council of the Ministry of Economy and Finance, in order to ensure the legality, consistency and harmonization of the drafting of legal texts, as well as to ensure the effectiveness and efficiency of the management of the implementation of legal texts falling within the field of competence of the Ministry of the Economy and Finance and to participate in the execution of the Public Financial Management Reform Program (PFMRP).

1.1.2. Vision and Mission

- **Vision:** To make the national budget management more effective and efficient.
- **Mission:** To ensure the sustainability, transparency, and effectiveness of public financial management and public service delivery.

1.1.3. Address and Contact

- Address: St. 92, Sangkat Wat Phnom, Khan Daun Penh, Phnom Penh
- Tel: (+885) 97 252 8358
- Email: legalcouncilmef@gmail.com

- Website: https://www.legalcouncilmef.com/en
- Facebook: https://www.facebook.com/LegalCouncilMEF/
- Telegram: https://t.me/lcmef
- Location: Shown in *Figure 2*

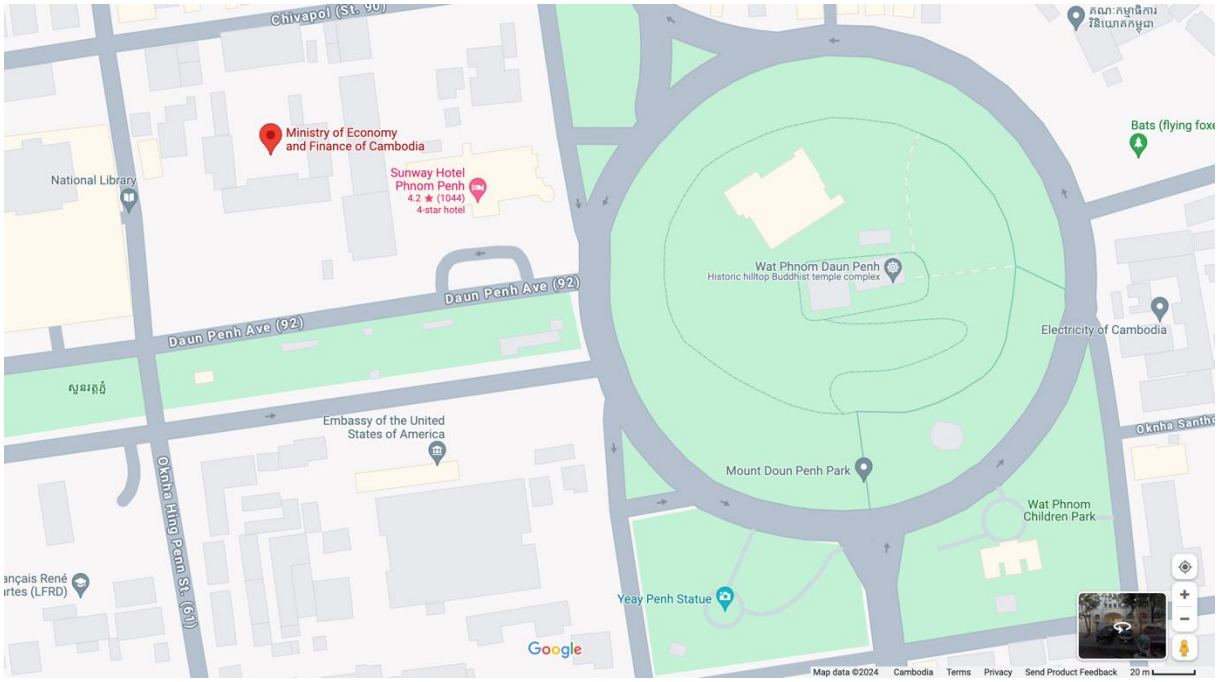

Figure 2: MEF's location

1.2. Internship Project Presentation

For the internship during three months at the Legal Council of Ministry of Economy and Finance, I worked on a project called "**Single Sign-On System and DevOps**", which is an identification method that enable users to login into multiple web client application with one set of credentials. In this project I used third party library service called "**Keycloak**" is an open-source software solution that provides single sign-on access to applications and services. I used DevOps to enable software development teams to release higher-quality software applications at a faster deployment software to production which used by CI/CD integration.

1.2.1. Team Project Structure

The internship and project development were guided and assisted by:

- Academic Supervisor: **Mr. NOP Phearum**
- Project Advisor: **Mr. VON Seyha**
- Developer and DevOps Internship: **Mr. ROTHA Dapravith**

1.2.2. Problematic of SSO

The problem of without Single Sign-On (SSO), users must remember multiple passwords, increasing the risk of security breaches and password fatigue. There are some problems including:

- **Poor user experience issues**: users must sign in separately for each system, leading to a frustrating and time-consuming experience and repeat login to any software application.
- **Password Fatigue**: managing multiple passwords for various systems is cumbersome and error-prone, increasing security risks due to potential password reuse or weak passwords.
- **Fragmented user data**: storing user information across different platforms complicates data security, increasing the risk of data breaches.
- **Data analysis difficulty**: It is difficult to carry out thorough data analysis and get useful insights when user data is dispersed across platforms.
- **IT management complexity**: managing multiple user databases increases IT workload, especially for tasks like password resets and updating access permissions, which are time-consuming and prone to errors.

1.2.3. Objective of SSO

The core objectives of implementing a Single Sign-On (SSO) are as follows:

- **Simplified IT administration:** centralizes user management and access control, reducing administrative overhead and improving security management**.**
- **Enhanced security:** minimizes the number of passwords users need to remember and manage, reducing the chances of security breaches and enabling the adoption of stronger authentication mechanisms.
- **Unified access:** allows users to access multiple applications with a single set of credentials, streamlining the login process and enhancing security across platforms.
- **Improved user experience**: Users enjoy a seamless login process across platforms, boosting efficiency and satisfaction.

1.2.4. Problematic Without DevOps

The problem without concept DevOps practice include:

- **Slow development cycle**: in the past, without the streamlined processes of DevOps, companies and organizations often faced long development cycles due to separate development and operations teams.
- **Silos between development and operation teams**: traditional models have separate development and operations teams, causing miscommunication, delays, and a lack of shared responsibility.
- **Unreliable environment**: differences in development, testing, and production environments can lead to code working in development but failing once deployed to release production on server.
- **Poor scalability**: handling increased loads or scaling operations is challenging without the infrastructure-as-code and automated scaling strategies provided by DevOps practices.

1.2.5. Objective of DevOps

The main objective for apply concept DevOps are:

- **Accelerated development cycles:** frequent updates and faster product releases through streamlined processes. make deployment of production easier and automate with DevOps methodology.
- **Enhanced software production quality**: continuous testing and integration improve software quality by catching bugs early. Furthermore, delivering software to the market faster and meets the requirements of clients.
- **Improved scalability, stability, and reliability**: consistent infrastructure and streamlined deployment processes reduce errors and make the system more reliable.
- **Faster time resolution, Enhanced innovation, and monitoring**: enhanced monitoring and collaboration in DevOps lead to faster issue resolution and more innovation, by automating routine tasks and enabling a focus on new features.
- **Version control:** utilize version control systems to track changes to code, configurations, and infrastructure.

1.3. Planning Project Table

After getting the main problem of the project, the schedule was conducted and divided overall the project's duration, three months in total (twelve weeks). Time divided according to various parts of the project.

Table 1: Planning table

Tasks	Weeks											
	1	2	3	4	5	6	7	8	9	10	11	12
Learn New Technologies												
Analyze Project Requirement & Database Design												
Study and Implement DevOps												
Implementation SSO Key-cloak												
Maintenance and Testing												
Deployment												
Report												

1.4. Detailed DevOps Project Description

1.4.1. Scope of Study

In this chapter of detailed DevOps project for my tasks, I had a specific role as DevOps assigned by project manager and advisor. While my team were focused on web development. I used DevOps method which is CI and CD to automate the whole deployment process serve by Jenkins file and need to enable notified message alert by telegram channel serve by telegram bot. also I have implemented with docker for containerized application and define container configuration to ensure consistency and portability across different environment in project.

1.4.2. Limitation of Study

During our project, my advisor provided a droplet server from Digital Ocean and a domain name to set up our DevOps tasks. I was tasked with deploying our frontend and API projects using Nginx on this domain, as outlined in our GitLab repository. My main responsibilities included managing the "main" branch for production releases through a CI/CD pipeline using Jenkins, configuring credentials and a Telegram bot for deployment notifications. I instructed the development team to avoid making direct changes to the "main" branch and to work on their feature branches instead. They would merge their updates into the "develop-1.0" branch, which I monitored and reviewed. My role was to ensure these changes were properly merged from "develop-1.0" into "main" for production deployment.

1.4.3. Hardware and Operating System Used

This project is conducted by using some devices with operating systems such as:

- **Local Server**: personal computer model MacBook Pro Chip Apple M1 (Memory: 16 GB, SSD: 494.38 GB)
- **Server**: Ubuntu 23.10 (GNU/Linux 6.5.0-28-generic x86_64) in droplet Digital Ocean was chosen as the primary operation systems for the server due to its stability and compatibility with software stacks.

II. PROJECT ANALYSIS, AND DESIGN

2.1. Case Study of Requirement

2.1.1. Functional Requirement of SSO

Functional requirements are important requirements needed to operate the system and must be implemented. It is necessary to define these functional requirements to respond to the needs of the system and the main objective of the project. Below are the requirements needed to be done in this project:

- **Admin** functionalities:

Table 2: Admin functionalities

Modules	Functionalities	Description
Authentication	Login, Logout	Admins can securely log in and logout accounts in the system through the application interface.
Realms Management	Create, Import, Export Realms	Manage the creation, importation, and exportation of realms to organize and isolate configurations.
Customization	Customize Pages, Emails, Themes	Customize the appearance of login pages, email templates, and themes to align with branding requirements.
Key algorithms Management	Add Providers and Keys	Add authentication providers and specify cryptographic keys for required algorithms.
Events Handling	Manage Events	Set up and manage listeners for various user and admin events within the system.
Localization	Manage Localization	Configure and manage language settings for user accounts, enhancing user experience across regions.

Security Management	Implement Security Policies	Enforce security policies including content security policies and brute force detection mechanisms.
Session Management	Manage Sessions	Control session settings for SSO, client sessions, offline sessions, and login specifics.
Token Management	Create, Update, Delete Tokens	Handle access, refresh, and action tokens for managing user authentication and access controls.
Client Management	Manage Clients and Client Scopes	Configure, update, and delete client profiles, and manage scopes and adapter configurations.
Roles Management	Add Realm Roles	Assign and configure realm roles to users to define authentication and authorization levels.
Profile Management	Manage Profile Policies	Set and enforce policies related to client profiles within the system.
User Registration	User Registrations	Administer the creation, updating, and deletion of user registrations within the system.
Account Management	Manage User Accounts	Create, update, and delete user accounts, managing all aspects of user information.
Credential Management	Set User Passwords	Manage credentials, specifically setting and resetting passwords for user accounts.
Group Management	Manage User Groups	Organize users into groups for simplified management and targeted policy application.

Authentication Flow Management	Create, Update, Bind flow for authentication flows	Design and implement specific authentication flows and steps for user verification processes.
Identity Providers	Manage Identity Providers and Federations	Configure external identity providers and manage federation settings to streamline user access.

- **User** functionalities:

Table 3: User functionalities

Modules	Functionality	Description
Authentication	Self-registration User Account	Users can independently create their own accounts, initiating their access to the system without immediate admin intervention. This facilitates a user-driven approach to account management.
	Login	Users can securely log in to the system to access connected services and applications, ensuring secure and authenticated access.
	Logout	Users can securely log out from the system, ensuring their session is safely ended and reducing the risk of unauthorized access.
Email Verification	Send Email Verification	Following registration, users can send a verification email to confirm their account, activating the account only after admin approval. This adds an extra layer of security and verification.
Account Management	Manage Account	Users can manage their account settings and preferences, providing control over privacy settings and personal preferences.

Profile Management	Manage User Profile	Users can edit and update their profile information within the system, ensuring personal data is current and accurate.
Access Management	Views All Applications and Resources	Users can view and access all applications and resources they are authorized to use, enhancing the user experience by simplifying access to multiple services.
Identity Provider Management	Adjust Settings for Identity Providers	Users can set up and modify their identity provider settings for services like Google or Facebook, offering flexibility in how they choose to authenticate.
Security Management	Define and Enforce Security Guidelines	Users are subject to security guidelines enforced across the system for consistent security, helping to maintain a secure environment for all users.
Activity Monitoring	Track Activity and System Health	Users can view logs and reports related to their activities and overall system usage, allowing for self-monitoring and proactive management of potential issues.
Personal Information account	Update Personal Information	Users can update their personal details such as name, email, and contact information, which is essential for maintaining accurate communication channels.
Password Management	Update Password	Users can change their password as part of routine security practices, which is crucial for maintaining the security integrity of their accounts.
Device Management	View Device Activity and Linked Accounts	Users can monitor which devices are linked to their account and review device activity logs, providing insights into account access patterns and potential security threats.

Two-Factor-Authentication	Setup with OTP Code via Google Authenticator	Users can enhance security by setting up two-factor authentication using OTP codes from apps like Google Authenticator, adding an additional layer of security to their login process.

2.1.2. Non-Functional Requirement of SSO

A non-functional requirement is the least important requirement for a system to be implemented, but it should also exist in the project. This requirement can help the system to improve the performance and quality of the project. Below is the non-functional requirement of this project:

- **Performance:** the system should respond quickly to login requests, even during busy times, to keep user frustration low and productivity high.
- **Scalability:** many users are joining or as demand grows. so, the system should easily expand without users noticing any slowdown or issues. This ensures a smooth experience regardless of how large or busy your organization gets.
- **Availability:** the SSO system needs to be up and running all the time, minimizing any disruptions or downtime. This constant availability is crucial for users who rely on the system to access multiple applications for their daily work.
- **Reliability:** users depend on the SSO system to function correctly every time; it should consistently handle authentication requests without errors.
- **Security:** protecting user data and credentials is paramount for SSO. the system must include robust encryption, secure connections, and adherence to best security practices.
- **Maintenance:** the system should be easy to manage and update, which requires a clear, modular architecture and straightforward maintenance procedures to facilitate updates and changes with minimal service interruption.
- **Usability:** The user interface and overall user experience should be intuitive and straightforward, minimizing user input errors and ensuring compatibility across different platforms and devices**.**

2.1.3. Functional Requirement of DevOps

Functional requirements are crucial tasks that the system must do to work properly. They must be carried out to meet the system's needs and the main goal of the project. Here are the things that need to be done in DevOps project:

- **Version Control:** integrating GitLab with SourceTree enhances version control by combining GitLab's robust repository management and CI/CD capabilities with SourceTree's intuitive graphical interface, making complex Git operations accessible to all team members.
- **Containerization:** implement a built-in container registry for efficient Docker image management and ensure seamless integration with container orchestration platforms to facilitate automated deployment and scaling of containerized applications.
- **Web server configuration:** automate Nginx setup for frontend and API projects, ensuring SSL, caching, load balancing, secure headers, and Certbot security.
- **Continuous Integration (CI):** set up a Gitlab server CI to automate source code build and test update code. ensure that CI pipelines in Jenkins script file run unit tests and other automated tests or validate change.
- **Continuous Deployment (CD):** create automated deployment pipelines to push code updates to different environments such as development or production seamlessly. Additionally, incorporate blue-green deployments or canary releases to minimize the impact of changes during the deployment process.
- **Infrastructure as Code (IaC):** utilize Jenkins for infrastructure-as-code to automate provisioning and management of infrastructure. Store infrastructure code in version control for change tracking and reproducibility.
- **Monitoring Services:** Implement monitoring services to regularly check and assess the performance of the website on servers. And set up alerts for potential errors or crashes.

2.1.4. Non-Functional Requirement of DevOps

Non-functional requirements might not grab the spotlight, but they play a crucial role in making the system better. They help enhance how well the system works and improve the overall quality of the project. Here are the essential non-functional requirements for this project:

- **Performance**: optimize code and data retrieval methods to reduce latency and improve response times, ensuring efficient resource utilization and enhancing user experience.
- **Scalability:** use Docker for containerization to easily deploy and scale applications, enabling seamless horizontal scaling to handle increased loads without downtime during CI/CD process.
- **Reliability:** implement automated testing pipelines with Jenkins to ensure code quality and stability, while using PM2 for process management to maintain high availability and resilience.
- **Security:** we decide who go to see what happen in Gitlab for version control to track changes securely during CI/CD process.
- **Network:** we decide how much internet speed is needed for things to run smoothly.
- **Telegram for alerting:** make sure that alert message in telegram work fast and reliably, and we consider how quickly notified should be sent.

2.2. System and Design

2.2.1. Physical Architecture of SSO

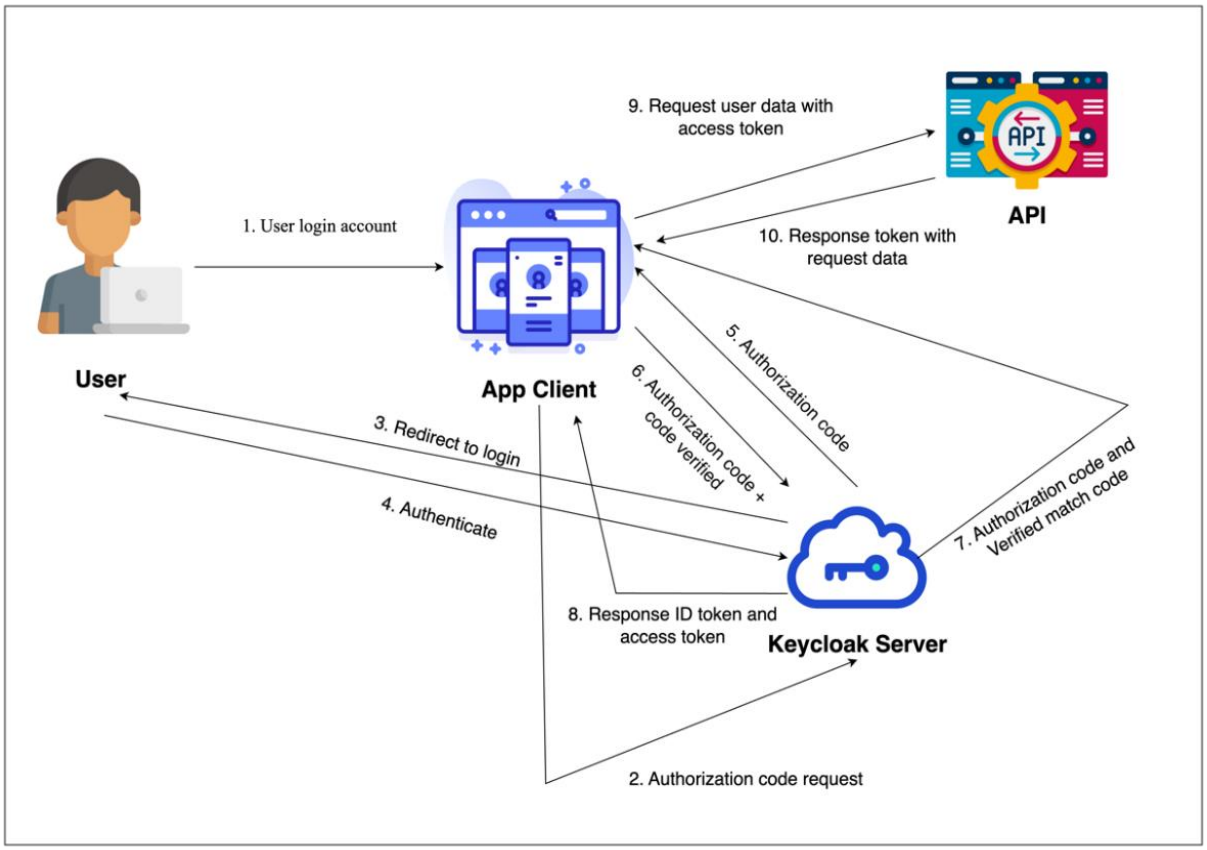

Figure 3: Physical architecture of sso

The user logs into an SSO-integrated application, which directs them to Keycloak's login page. After entering their credentials, Keycloak validates them and sends an authorization code to the client application. The client application sends this code back for verification. Keycloak then issues ID and access tokens, which the client uses to request user data. The client receives the requested data and any necessary tokens.

2.2.2. Logical Architecture of SSO

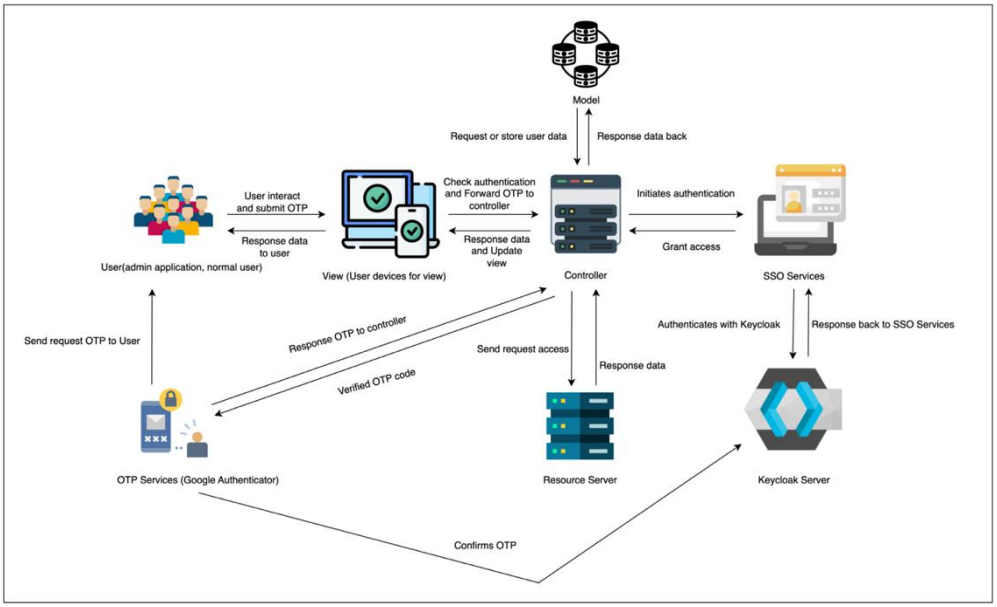

Figure 4: Logical architecture of sso

The logical architecture diagram of an SSO system with Keycloak consists of user, view, controller, SSO Services, Keycloak Server, Resource Server, and OTP Services. The user interacts with the web client's application via browser, while the view provides a graphical interface. The controller handles user authentication, authorization, and response view paths. OTP Services enhance security services by providing an additional layer of authentication.

2.3. DevOps Infrastructure Design

2.3.1. Physical Architecture of DevOps

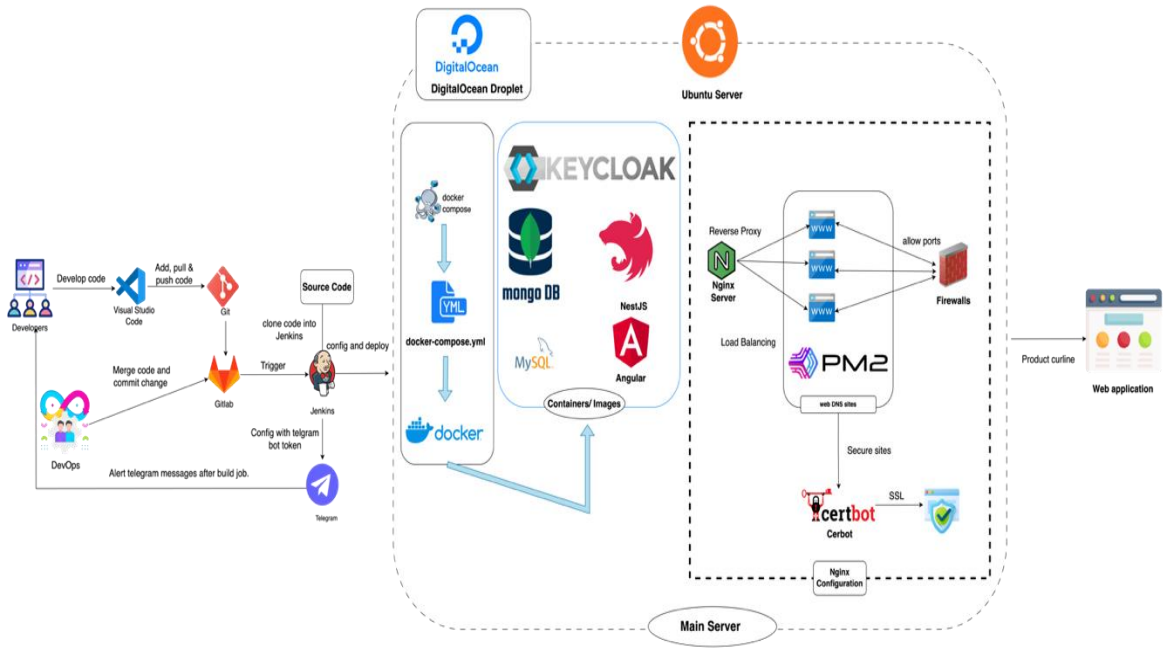

Figure 5: Physical architecture of DevOps

The deployment process begins with developers committing updated code to GitLab. Docker and Docker Compose containerize the application, ensuring consistency across environments. A Jenkins CI/CD pipeline triggers code changes, runs automated tests, and builds Docker images. These images are pushed to Docker Hub and then deployed on a Digital Ocean server using Docker Compose. Nginx is set up as a reverse proxy and load balancer. Security is ensured with firewall configurations and SSL/TLS certificates managed by Certbot. PM2 handles process management and monitoring, keeping the application operational and managing logs.

2.3.2. Logical Architecture of DevOps

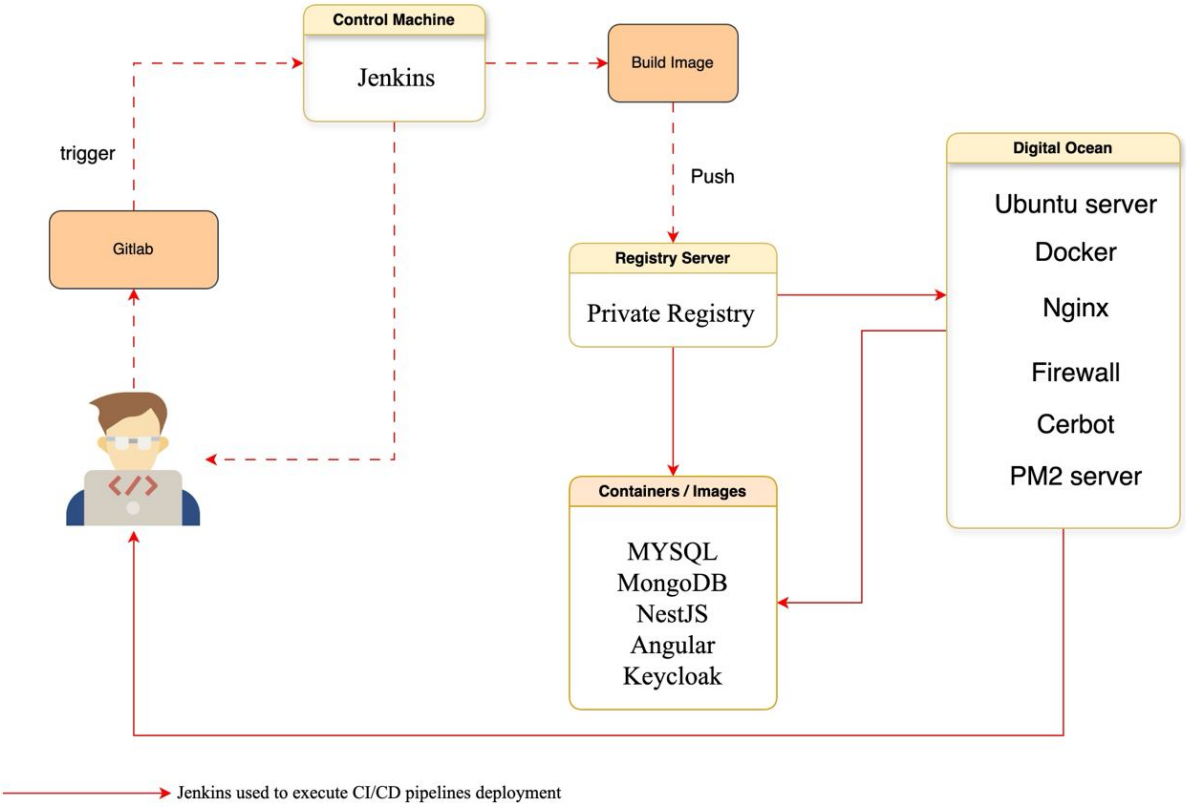

Figure 6: Logical architecture of DevOps

The Figure above describes the workflow how I operate. The DevOps workflow in the diagram starts with GitLab, where a code trigger initiates Jenkins. Jenkins automates the process by building Docker images which are then pushed to a private registry. From the registry, the images are deployed on a Digital Ocean server configured with Ubuntu. This server uses Docker to manage containers and Nginx for handling web traffic. Security is maintained with firewall rules and SSL via Certbot, while PM2 is used to manage and keep the applications running smoothly.

2.4. Project Analysis

2.4.1. Use Case Diagram

A use case diagram is a diagram that shows the user roles or types in the system to confirm specific what functionalities can use in any roles. Each use case diagram represents one feature use case of the system, and each module contains much functionality, which is described in detail in the following parts.

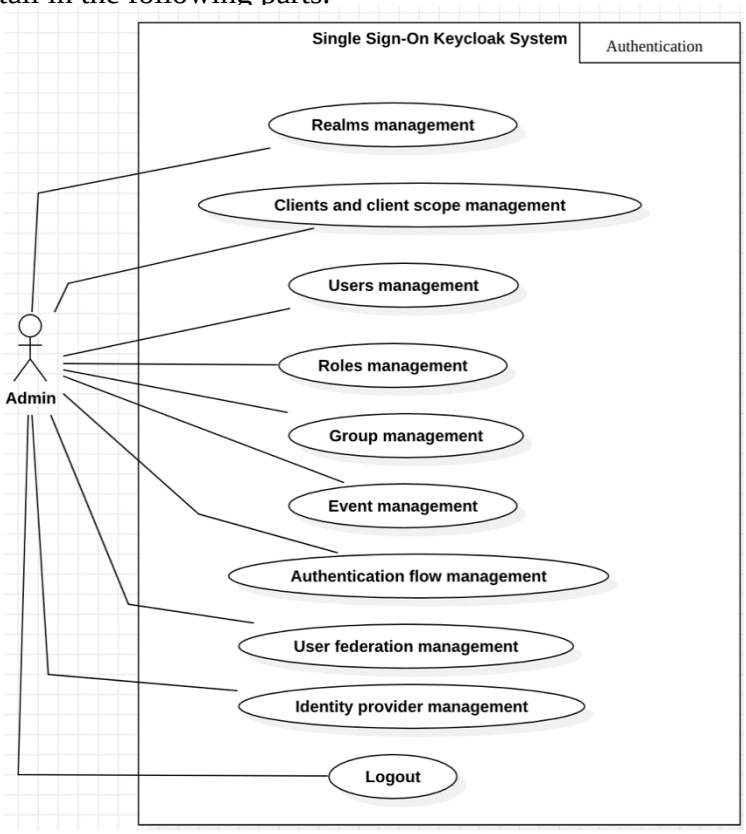

Figure 7: Admin use case diagram

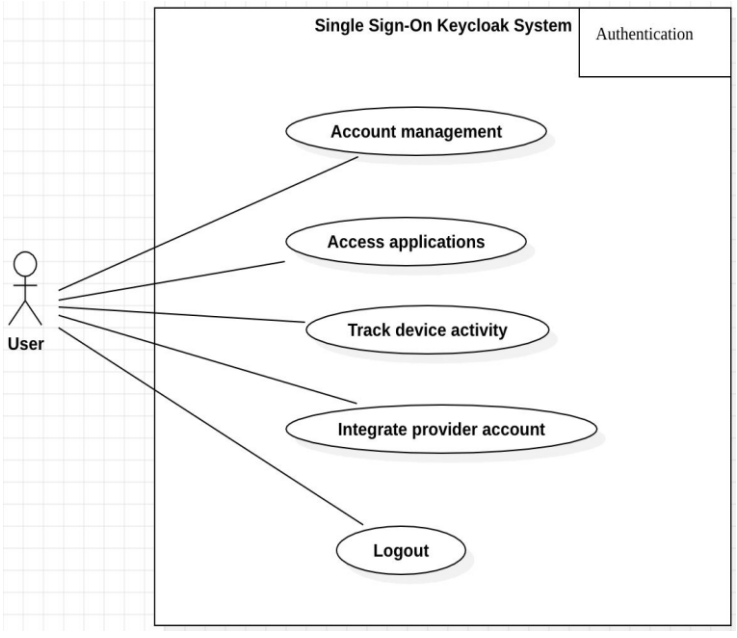

Figure 8: User role use case

2.4.2. Database Schema

This section provides a simple introduction to the details of database design. Inside, you will discover a clear picture of our application's database, showing how different pieces of information are connected and explaining what's inside each table.

2.4.3. Overall database schema

This is the overall database schema in our system that makes the system work. The figure below shows all the tables that we use.

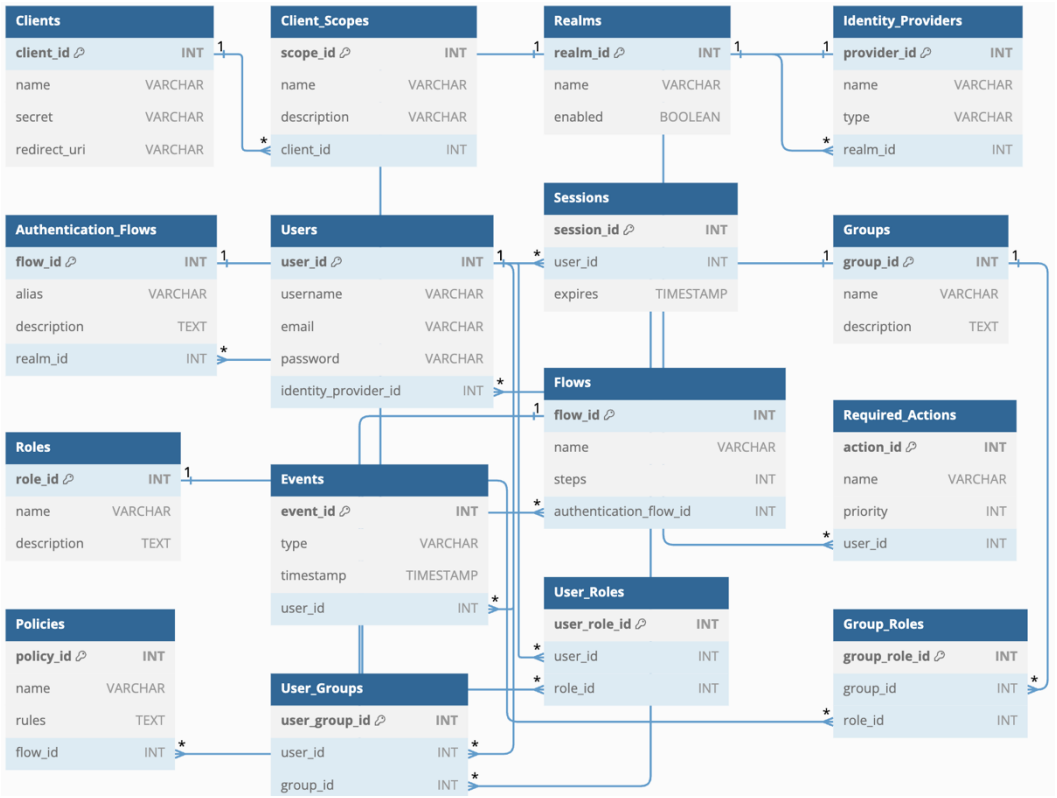

Figure 9: Database schema

2.4.4. Entity Relation Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database schema. These diagrams provide a clear and intuitive means to conceptualize database structure, making them essential for effective data modeling.

- **Entities:** which are represented by rectangles. An entity is an object or concept.

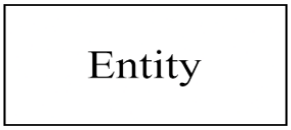

Figure 10: Entity model

- **Actions:** which are represented by diamond shapes, It is illustrating how two entities share information or have relationship connect within the database.

Figure 11: Action model

An Entity Relationship diagram (ERD) shows how different parts of data in a database schema are connected. In simple terms it means that shows the relationships between these data components include attributes in each table of database. ER diagram can help developers to understand the logical layout arrangement of databases, as shown in the figure.

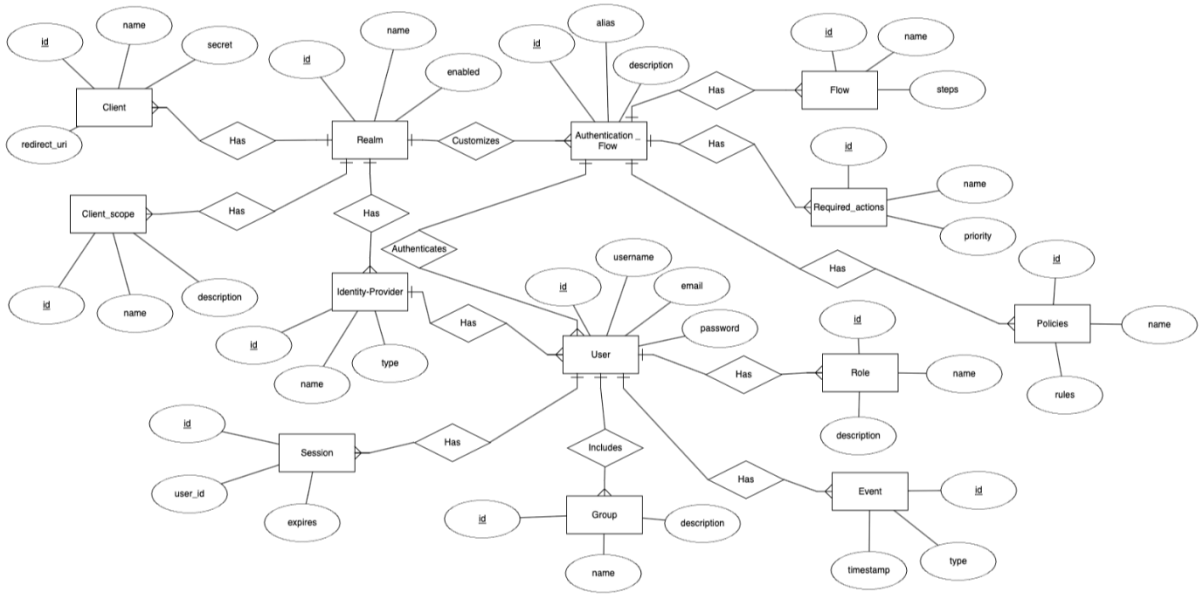

Figure 12: ER Diagram

2.5. Activity Diagram

An activity diagram is a type of UML diagram that visually represents the workflow or sequence of actions within a system, highlighting decision points, parallel processing, and the overall control flow. It is used in projects to demonstrate how different parts of a system interact, especially useful for modeling business processes and complex functionalities in software development.

2.5.1. Role Admin

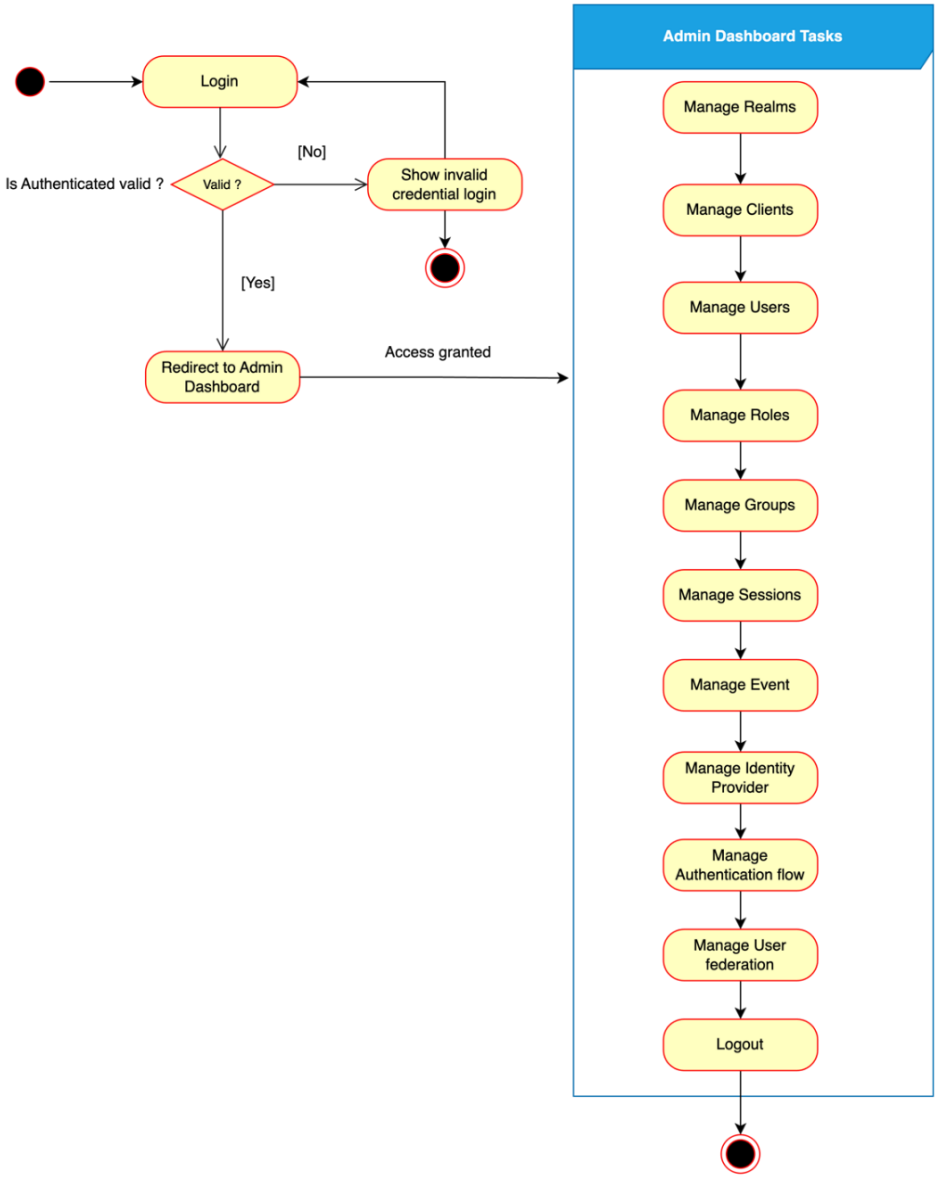

Figure 13: Activity diagram of admin role

In Figure is an activity diagram for admin role in the system I implemented, I will describe the overall step include:

- **First,** the administrator logs in to the system by providing their credentials. This step is crucial for accessing the administrative functionalities of Keycloak.
- **Second,** the system verifies these credentials. If the credentials are invalid, the system immediately notifies the administrator by displaying an invalid credential login message. This step prevents unauthorized access and ensures that only valid administrators can proceed.

- **Third,** once the credentials are confirmed to be valid, the administrator is redirected to the admin dashboard. This dashboard is the control center where all the administrative tasks are performed.
- **Fourth,** within the dashboard, the administrator can perform various tasks such as managing realms, clients, users, roles, and more.
- **Finally,** after managing the necessary configurations and ensuring the system is running smoothly, the administrator can log out of the system, securely ending their session.

2.5.2. Role User

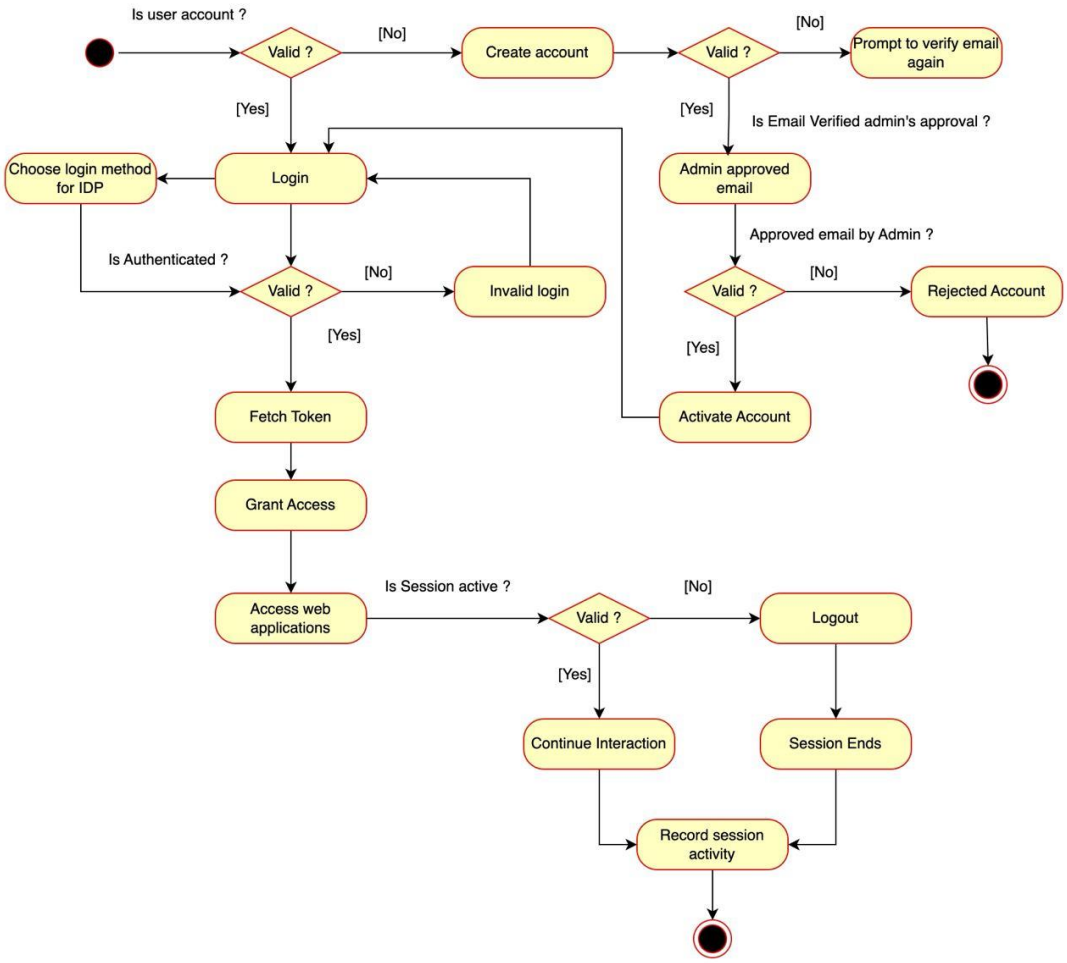

Figure 14: Activity diagram of user role

In Figure shown an activity diagram for user role in the system I implemented, hence is the details step about the overall step in the system include:

- **First,** the system checks if the user already has an account. If not, the user must create an account and verify their email.
- **Second,** if the email verification requires administrative approval, the system checks for the admin's approval. If the email isn't verified or is rejected by the admin, the account is not activated. Otherwise, the account is activated successfully.
- **Third,** the user proceeds to log in by selecting the appropriate identity provider (IDP) login method. If the login credentials are invalid, the system will prompt an invalid login message.
- **Fourth,** after successful authentication, the system fetches a token and grants access to the user.
- **Fifth,** the user gains permission to access web applications as required.
- **Sixth,** the system checks if the user's session is still active.
- **Finally,** the session activity is recorded, and the workflow is complete.

2.6. Sequential Diagram

A sequence diagram is a type of UML diagram that shows the interactions between objects or components in a system over time. Software developers and business professionals use it to understand system requirements or document existing workflow processes.

2.6.1. User Authentication

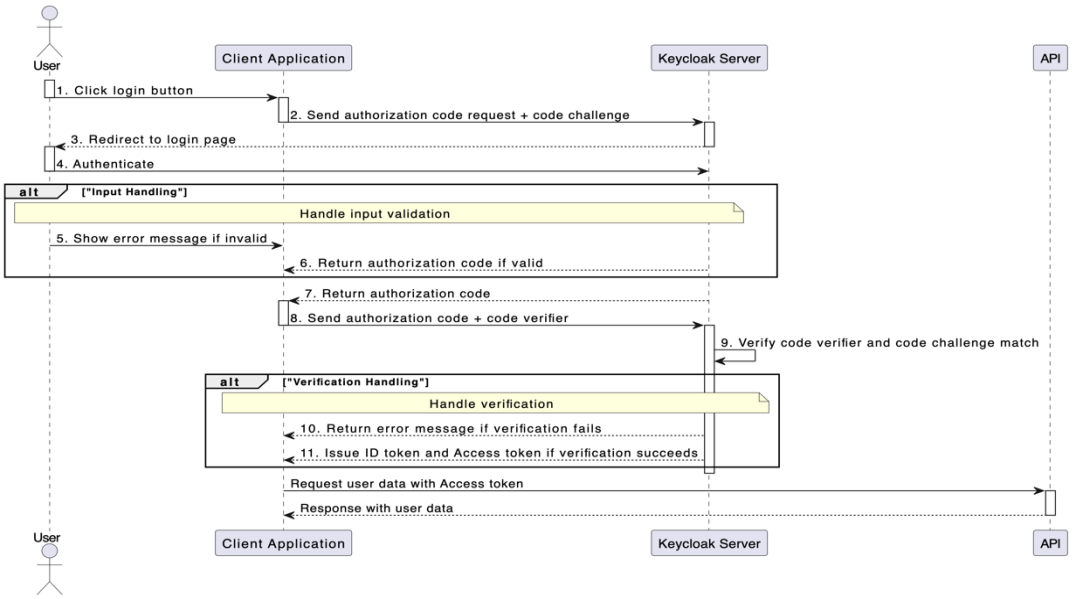

Figure 15: Sequential diagram of authentication

The sequence diagram illustrates user authentication with Keycloak. It starts with the user clicking the login button, which triggers the client to send an authorization code request. The user is then redirected to the login page for authentication. Once authenticated, Keycloak verifies the authorization code and issues ID and access tokens. The client uses these tokens to request user data from the API.

2.6.2. User Authorization

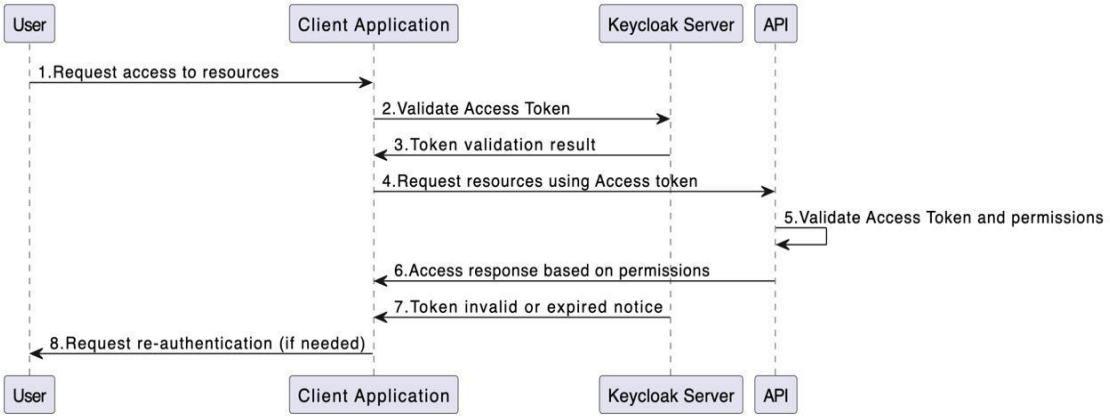

Figure 16: Sequential diagram of authorization

The diagram illustrates user authorization for accessing resources. It involves requesting access through the client application, sending an access token to Keycloak for validation, re-validating the token, and notifying the client if the token is invalid or expired.

2.7. Choice of Technology and Tools

2.7.1. Technologies

Keycloak [1] is an open-source Identity and Access Management (IAM) tool which was developed by Red Hat. Keycloak allows users to authenticate and authorize applications securely. offering features like single sign-on, identity brokering and social login, user federation, admin console, account management and standard protocols provides support for OpenID Connect, OAuth 2.0 and SAML.

Figure 17: Keycloak's logo

Angular [2] is a powerful framework from Google used for building dynamic web applications with TypeScript. I picked Angular because it has everything needed built-in, making development smoother and more efficient. I used it to create robust applications and integrate them with Key cloak for secure single sign-on. Its CLI made project setup and deployment a breeze, perfect for large-scale projects.

Figure 18: Angular logo

Docker [3] is an open platform for developing, shipping, and running applications in standardized units called containers, which include all necessary components like libraries, system tools, code, and runtime. I used Docker for building images and running containers, ensuring consistency and efficiency. While Portainer.io complements Docker with a lightweight, web-based UI for managing containers, images, networks, and volumes, featuring an intuitive interface and role-based access control.

Figure 19: Docker logo

Nginx [4] is an open-source web server software used for reverse proxy known for handling high load balancing traffic efficiently compared to other web servers like Apache is high configurable, I used Nginx as a web server for config DNS with project that I managed and secured website use cerbot and Let's encrypt with SSL certificate encryption.

Figure 20: Nginx logo

Jenkins [5] is an open-source automation server vital for CI/CD processes in software development. It enables teams to automate building, testing, delivery, and deployment of software applications. Jenkins supports extensive automation through pipelines, integrating with tools like GitLab and providing notifications via Telegram, streamlining the entire development lifecycle. using Blue Ocean with Jenkins, I efficiently monitored CI/CD processes, quickly identifying, and resolving issues. Its streamlined interface greatly enhances the user experience compared to the classic Jenkins interface, making it a valuable tool for DevOps.

Figure 21: Jenkins logo

MYSQL [6] is an open-source relational database management system (RDBMS) based on SQL (Structured Query Language). It is widely used for managing and organizing structured data and supports various data types. Also, it is favored for its reliability, scalability, and compatibility with all major hosting providers, making it ideal for a wide range of web applications.

Figure 22: MySQL logo

MongoDB [7] is a NoSQL database management system characterized by its document-oriented data paradigm. It is valued for its flexibility, scalability, and ease of handling semi-structured and unstructured data, making it suitable for modern online applications.

Figure 23: MongoDB logo

PM2 [8] is a process manager for Node.js applications that enhances uptime by automatically restarting apps and balancing loads. It is particularly effective in production environments due to its robust monitoring and management features.

Figure 24: PM2 logo

YAML (YAML Ain't Markup Language) [9] YAML is a human-readable format used in CI/CD workflows like GitLab CI, Ansible, Kubernetes, and Docker Compose for defining pipelines and automating processes. Integrating YAML with Shell Script enhances automation by defining complex tasks simply and effectively.

Figure 25: YAML logo

2.7.2. Tools

Visual Studio Code [10] is a standalone source code editor that runs on Windows, macOS, and Linus. The top pick for Website development, with more extensions support. I used it because it works fast for my computer, and it can create and compile projects similar to other tools.

Figure 26: Visual studio code logo

SSH (Secure Shell) [11] is a network protocol that enables secure remote access to computers. It uses strong encryption to ensure that data transmitted between the user and the remote server is safe. My experience with SSH has been positive, as it provides a reliable and secure way to manage remote systems. Compared to Telnet, which sends data in plaintext, SSH offers superior security by encrypting all data, making it the preferred choice for secure remote operations.

Figure 27: SSH key logo

Terminus [12] is a cross-platform SSH client supporting SSH, Mosh, and Telnet protocols, enhancing remote device management. It enables team collaboration and offers command history and secure syncing across devices. With a modern interface, it outperforms traditional clients like PuTTY in managing multiple connections and organizing servers.

Figure 28: Termius logo

SourceTree [13] is a graphical user, which is a distributed version management system. It provides a visual interface to manage and track changes in source code, as well as collaborate with others on software development projects. I chose SourceTree because of its simple interface and powerful Git integration. It simplifies difficult version control procedures and allows you to easily visualize branching, merging, and commit history.

Figure 29: SourceTree logo

GitLab [14] is a web-based Git repository manager that supports code collaboration, project management, and development lifecycle automation. It offers continuous integration, delivery, containerization, and security testing. GitLab manages public and private repositories securely and integrates with tools like Jira, Jenkins, and Kubernetes. Its user-friendly interface and versatile deployment options make it a popular choice for development teams.

Figure 30: GitLab logo

Digital Ocean [15] is a cloud infrastructure provider offering a high-performance virtual server called “droplet”. My experience has been positive, with straightforward setup and efficient application scaling. Compared to AWS, Digital Ocean is less complex and more cost-effective, though it lacks some advanced features. Using an Ubuntu server on Digital Ocean enhances ease of use and supports scalable deployment.

Figure 31: Digital Ocean logo

Postman [16] is an API platform for developers to design, build, test, and iterate their APIs. It provides a graphical user interface to test HTTP requests and validate responses. I use Postman to test project APIs after building each functionality and before integrating with the front-end. With Swagger configuration, importing the API description link in Postman generates all the requests for easy testing.

Figure 32: Postman logo

Name.com [17] is a user-friendly domain registration service known for its simplicity and excellent customer support. I use it for registering domain names due to its straightforward interface. Compared to competitors like GoDaddy, Name.com offers a more streamlined experience without aggressive upselling, making it ideal for basic domain registration needs.

Figure 33: Name.com logo

Draw.io [18] is free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams, I use this tool because it is a free software application for making flowcharts and another diagram.

Figure 34: Draw.io logo

Telegram [19] supports bots for automated tasks via Bot Father, which helps create and manage them. I use a Telegram bot to receive CI/CD pipeline updates from Jenkins. Bot Father provides a token for integration, sending updates directly to my Telegram account. This method offers quick, efficient monitoring of build and deployment statuses, providing instant feedback and improving workflow compared to email.

Figure 35: Telegram logo

III. PROJECT IMPLEMENTATION

In this chapter, I will detail the implementation of the project during the internship. and I explain how to set up tools and technology, project implementation, and installment.

3.1. Project Implementation of SSO System

3.1.1. Project Setup Environment and Installation

In this report, I will describe the installation of Keycloak with two methods to set up the environment keycloak on locally and the server by using docker-compose and daemon application manual installation.

I. Daemon applications (Manual setup)

To install the requirements and set up the Keycloak project, follow these steps:

1.Download Keycloak from Keycloak's official website.

2.Extract the file using tar or an unzip utility.

3.Run the Keycloak server by executing **standalone.sh**.

4.Access the Keycloak admin console at http://localhost:8080/auth/admin.

For more detailed configuration, refer to the Keycloak documentation.

II. Keycloak with Docker

To set up the Keycloak with docker you can follow these steps:

1. Ensure Docker and Docker Compose are installed on your server.

2. Create a docker-compose.yml file using the official Keycloak Docker image.

3. Config port mappings.

4. Start keycloak server **docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:24.0.5 start-dev**.

5. To access the Keycloak admin console, visit http://localhost:8080/auth/admin on your web browser or use your configured domain name if applicable.

For detailed steps, see the Keycloak Docker guide.

3.1.2. Configuration

In the configuration section, to configure Keycloak with a MySQL database, navigate to the configuration directory **/opt/keycloak-24.0.4/conf** and edit the **keycloak.conf** file to set the database parameters. place the **MySQL JDBC driver** in the **/opt/keycloak-24.0.4/lib** directory. Finally, start the Keycloak server using the command **/opt/keycloak-24.0.4/bin/kc.sh start**. This setup integrates MySQL as Keycloak's database. Here is the step I followed to configure the **keycloak.conf** file with MYSQL database and other configuration as shows in figure 36:

keycloak.conf

```
# Basic settings for running in production. Change accordingly before deploying the server.
# Database
# The database vendor.
db=mysql

# The username of the database user.
db-username=keycloak

# The password of the database user.
db-password=keycloak

# The full database JDBC URL. If not provided, a default URL is set based on the selected database vendor.
db-url=jdbc:mysql://localhost:3306/keycloak

# Observability
# If the server should expose healthcheck endpoints.
#health-enabled=true

# If the server should expose metrics endpoints.
#metrics-enabled=true

# HTTP
http-enabled=true
hostname-strict-https=false
hostname-strict-backchannel=true
hostname-url=https://sso.itcintern.xyz
hostname-admin-url=https://sso.itcintern.xyz

# The file path to a server certificate or certificate chain in PEM format.
#https-certificate-file=${kc.home.dir}conf/server.crt.pem

# The file path to a private key in PEM format.
#https-certificate-key-file=${kc.home.dir}conf/server.key.pem

# The proxy address forwarding mode if the server is behind a reverse proxy.
proxy=edge

# Do not attach route to cookies and rely on the session affinity capabilities from reverse proxy
#spi-sticky-session-encoder-infinispan-should-attach-route=false

# Hostname for the Keycloak server.
#hostname=myhostname

http-port=8082

KEYCLOAK_ADMIN=admin
KEYCLOAK_ADMIN_PASSWORD=admin

KC_LOG=file
KC_LOG_FILE=/var/log/keycloak/keycloak.log
```

Figure 36: Config database in keycloak

For detailed more configuration, refer to Setup Keycloak Server.

keycloak-docker > Dockerfile > ...

ROTHA DAPRAVITH, 2 months ago | 1 author (ROTHA DAPRAVITH)

```
# Use the base Keycloak image
FROM quay.io/keycloak/keycloak:24.0.3 as builder

# Enable health and metrics support
ENV KC_HEALTH_ENABLED=true
ENV KC_METRICS_ENABLED=true

# Configure a database vendor
ENV KC_DB=mysql

# make working directory
WORKDIR /opt/keycloak

# for demonstration purposes only, please make sure to use proper certificates in production instead
RUN keytool -genkeypair -storepass password -storetype PKCS12 -keyalg RSA -keysize 2048 -dname "CN=server" \
-alias server -ext "SAN:c=DNS:localhost,IP:127.0.0.1" -keystore conf/server.keystore

RUN /opt/keycloak/bin/kc.sh build

FROM quay.io/keycloak/keycloak:24.0.3 as production

WORKDIR /opt/keycloak

COPY --from=builder /opt/keycloak/ /opt/keycloak/

# start dev server
ENTRYPOINT ["/opt/keycloak/bin/kc.sh"]
```

Figure 37: Dockerfile config keycloak

In the Figure above, I will go through the configuration key cloak with docker, to use Keycloak, specify the Docker image and set environment variables for health checks, metrics, and MySQL connections. Generate SSL keys with key tool, run kc.sh build, and transfer configuration files to the runtime stage. Define the entry point for the Docker container in development mode.

3.1.3. Authentication

The implementation of authentication in the SSO system, which includes features like login and logout.

1. Login process flowchart

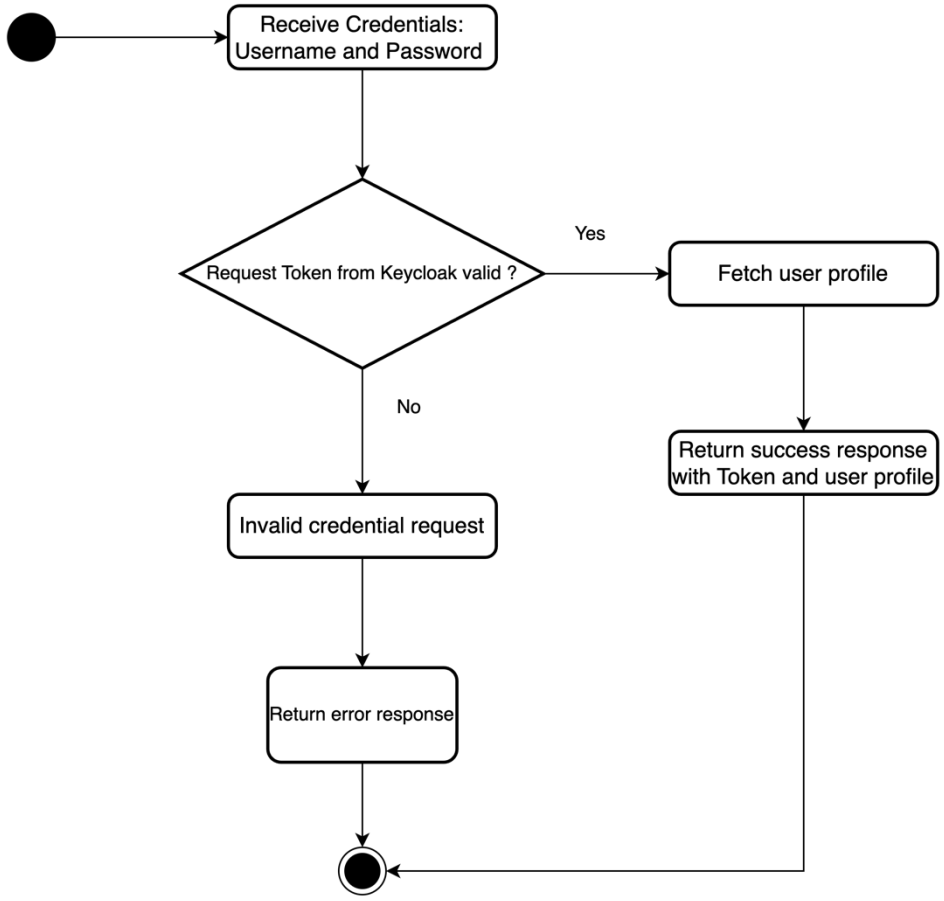

Figure 38: Login flowchart

In Figure shown about the process of authentication for login process are include:

- **First,** the process begins with login when the user submits their credentials like username and password.
- **Second,** if the user entered correct credentials with condition **"yes"** it will redirect fetch user profile and stored access token with return success response message then ending the process.
- **Third,** else not, if the user entered incorrect credentials with condition **"No"** it will show invalid credentials request and return an error response and not saved access token because it is wrong validating credentials before ending the process.

2. Logout process workflow

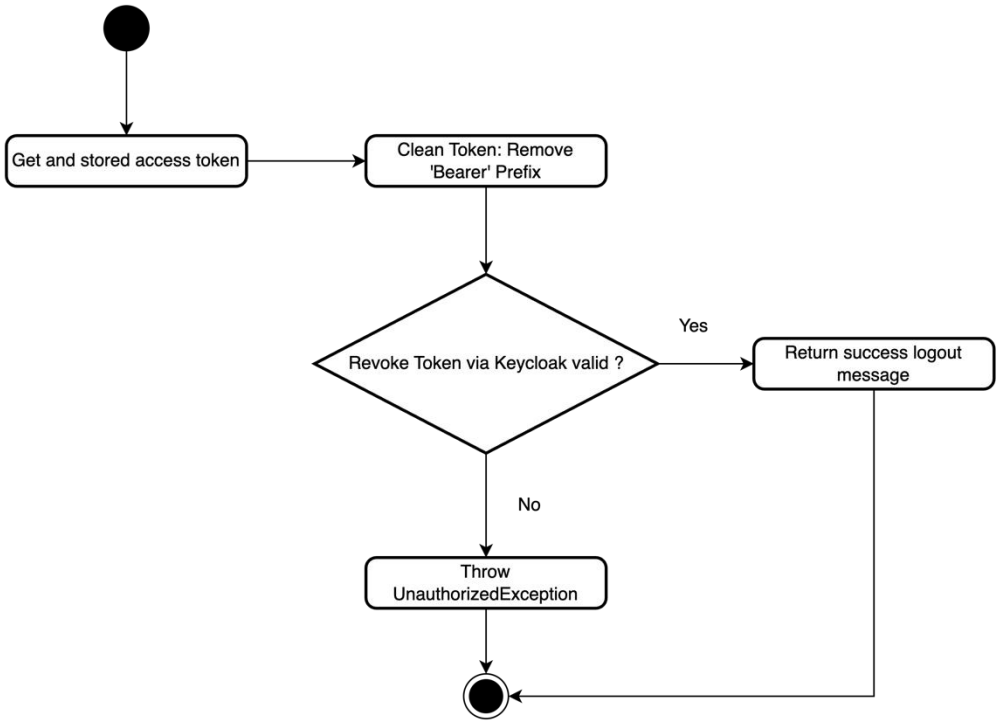

Figure 39: Logout flowchart

In Figure above shown about the logout process flowchart which describes steps:

- **First,** the process begins with keycloak system getting and store access token which is required for authentication.
- **Second,** it cleans the token by removing the 'Bearer' prefix. The system removes this prefix using the replace method, resulting in a cleaned token stored in only Token.
- **Third,** it attempts to revoke the token via Keycloak by calling the revoke method on the keycloakClientService.client with the cleaned token.
- **Fourth,** If the token is successfully revoked without any errors, the system returns an object with the message **'Logout success'**, indicating that the logout process was successful.
- **Otherwise,** it throws an Unauthorized Exception if the token revocation fails, or an error occurs before ending the process.

3.1.4. Authorization

Token validation is the process in which a system verifying that a digital token is authentic and valid, crucial for secure authentication. Below is the figure of flowchart process for token validation.

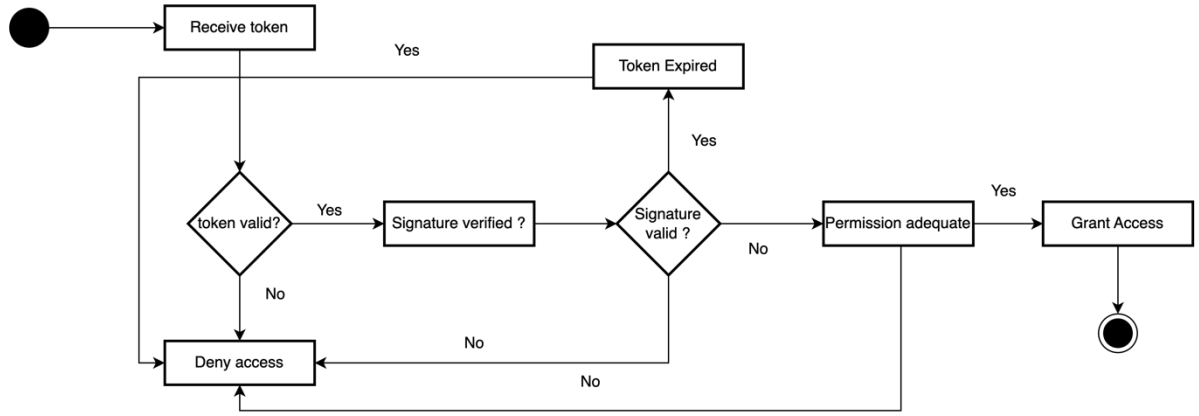

Figure 40: Token validation

Let’s break down each step-in flowchart above:

- **First,** the system receives the token from client’s authentication. normally, embedded in the HTTP authorization header of request.
- **Second,** the system verifies the token's validity by confirming that it is correctly formatted and remains unchanged.
- **Third,** the system verifies the signature of the token by using issuer’s public key to confirm its authenticity and that it has not been alerted.
- **Fourth,** the system checks if the token is expired. This involves checking the token's expiration time to ensure it's still active.
- **Then,** the system suggests a second check to confirm if the signature is valid. This might be an extra step for added security or could be a redundancy in the diagram.
- **After that,** the system evaluates if the permissions in the token are adequate for what the user is trying to do.
- **Finally,** if all checks are passed (valid token, valid signature, not expired, adequate permissions), the system grants access to the user. If any check fails, access is denied.

3.1.5. Deployment

For deployment section, I use docker for containerization, nginx for config web server, Jenkins build automated CI/CD deployment and notified messages via telegram and start server via pm2 server and logs for each application to track status.

3.2. DevOps project implementation

3.2.1. Version Control

Git flow is a set of recommendations for developers to follow when using Git, which acts as a central repository for a project and allows for easy to understand and progress between multiple branches. Below is the figure of Gitlow work process:

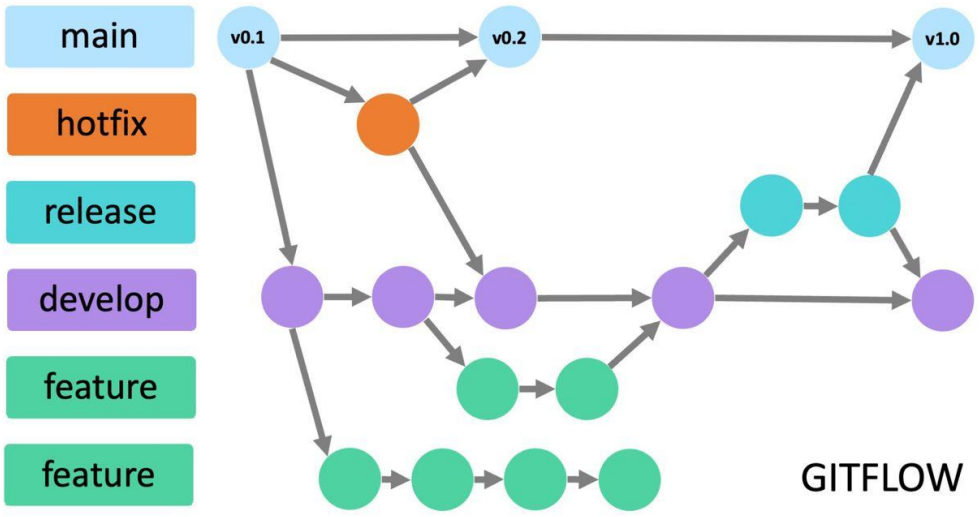

Figure 41: Git flow process

Here is the overall flow of Gitlow step is:

- Create a develop branch from main.
- Create feature branches from develop and merge completed features back into develop.
- Create a release branch from develop and merge it back into both develop and main when complete.
- For issues in main, create a hotfix branch from main and merge it back into both develop and main once resolved.

For more detailed information, refer to Gitflow Workflow.

Here is the Gitlow that I implemented in SourceTree application shows in figure 42

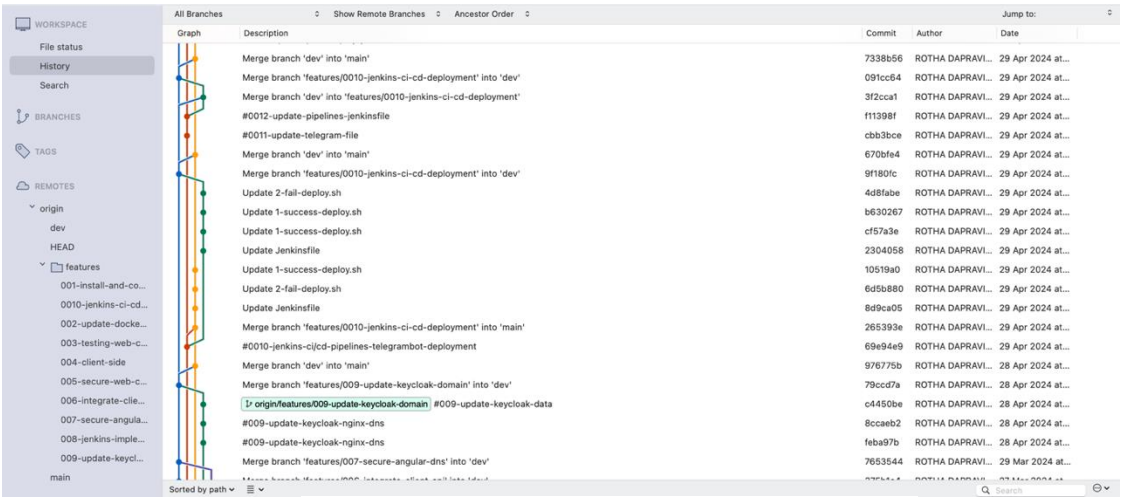

Figure 42: Implement Git flow on SourceTree

3.2.2. Service Provider (Digital Ocean)

For project deployment, I chose Digital Ocean to buy a service and config any projects on server. To set up Digital Ocean, start by creating an account or logging in. Next, create a "Droplet" and configure its settings. Set up backups, monitoring, and SSH keys for secure access. Choose a hostname and create the droplet. Connect to your droplet via SSH or the console. Finally, install the necessary software and prepare your application.

For detailed configuration steps, please refer to DigitalOcean documentation.

3.2.3. SSH key

To access your SSH key on a server, you can follow these steps:

- Connect using ssh your_username@your_server_ip.
- Enter your SSH Key Passphrase.
- Access the ~/.ssh directory.
- List files with ls and view the public key with cat.

For more details set up configure ssh key, refer to DigitalOcean's SSH setup guide.

3.2.4. Build Docker Images

Docker uses a client-server architecture. The docker client talks to the Docker daemon, which is used to build, run, and distribute the Docker containers. The Docker client and daemon communicate using a REST API, over UNIX sockets, or a network interface.

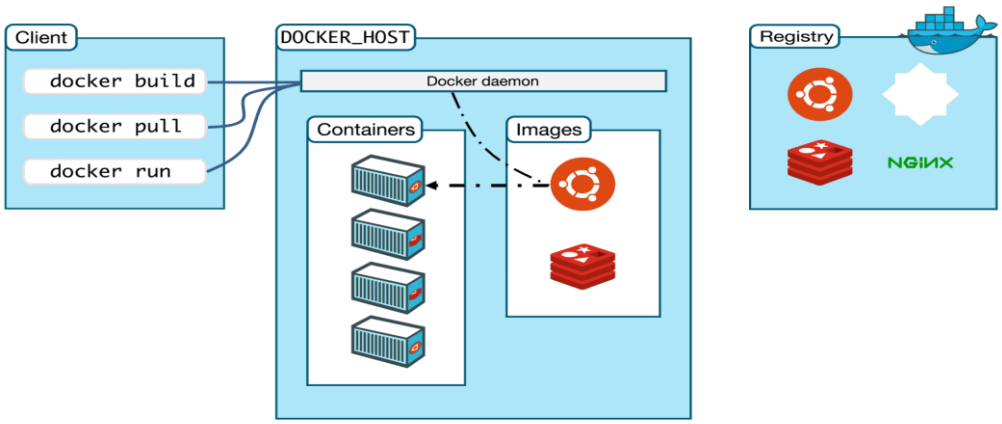

Figure 43: Docker Architecture diagram

Docker Daemon manages Docker objects like images and containers, while Docker Clients provide a CLI for interaction. Docker Host offers a full environment to run applications, and Docker Registry stores images. Docker Images are read-only templates built from Docker files, defining application dependencies and processes.

3.2.5. Nginx

First, update the OS and install Nginx, then adjust the firewall. Next, configure server blocks, manage Nginx commands (start, stop, restart), and test the configuration. Finally, access the server via a web browser to verify the setup. For more Nginx configuration please refer to Nginx configuration.

3.2.6. Jenkins

Jenkins can be installed on various environments including Docker, Kubernetes, Linux, macOS, and Windows. This guide focuses on setting up Jenkins on Linux using the Long-Term Support (LTS) release.

- Prerequisite: hardware requirements minimum 256 MB RAM, 1 GB drive space (10 GB if using Docker) and recommended: 4 GB+ RAM, 50 GB+ drive space although 10 GB is a recommended minimum if running Jenkins as a Docker container.

  1. Installation Steps

- **Install Java**: ensure Java is installed and set up environment variables.

- **Add Jenkins Repository:** Add Jenkins repository and import GPG keys.
- **Install Jenkins:** Update package index and install Jenkins.
- **Start Jenkins**: Enable Jenkins to start on boot, start the service, and check status.

2. Initial Setup

- **Unlock Jenkins:** Access Jenkins at http://localhost:8080 and enter the initial admin password.
- **Install Plugins:** Install suggested or specific plugins.
- **Create Admin User:** Set up the first admin user.
- **Configure System:** Set the Jenkins URL and add environment variables.
- **Manage Plugins and Credentials:** Install necessary plugins and add credentials for services.

3. Create new Jobs in Jenkins

- Click "New Item" on the Jenkins Dashboard.
- Enter a job name and select the "Pipeline" job type.
- Configure job settings, such as Build Triggers and pipeline scripts.
- Apply and save the configuration.

For detailed commands and configurations, refer to the references link official Jenkins documentation.

3.2.7. Jenkins CI/CD pipeline

Figure illustrates how to start a Jenkins pipeline that runs on any available agent. The environment block sets key variables: registry for the Docker registry path, registry Credential for Docker authentication, docker Image and docker Image for the Docker image name, and securely retrieves the Telegram bot token and chat ID using Jenkins credentials. This centralizes configuration and enhances security.

Jenkinsfile

ROTHA DAPRAVITH, 7 days ago | 1 author (ROTHA DAPRAVITH)

```
pipeline {
    agent any
    environment {
        registry = "dapravith99/lc-sso-system" // Docker registry
        registryCredential = 'dapravith99-dockerhub' // Jenkins credential ID for Docker
        gitCredentialsId = 'gitlab' // Ensure this ID matches the one in Jenkins
        Telegram_BotToken = credentials('Telegram_BotToken')
        Telegram_CHATID = credentials('Telegram_CHATID')
        dockerImage = ''
        IMAGE_TAG = "dapravith99/lc-sso-system:${env.BUILD_NUMBER}"
        DOCKER_COMPOSE_PATH = "${env.WORKSPACE}/docker-compose"
    }
```

Figure 44: Environment Jenkins pipelines

Figure, these stages in the Jenkins pipeline define the process from starting the pipeline, cloning the repository, updating documentation, building the application, and finally building the Docker image. Each stage is clearly defined, with placeholder commands where necessary, making it easy to customize and expand the pipeline according to project requirements.

Jenkinsfile

```
        steps {
            echo "Welcome to start Jenkins!"
        }
    }

    stage('Clone Project') {
        steps {
            git branch: 'main',
                credentialsId: "${env.gitCredentialsId}",
                url: 'https://gitlab.com/VonSeyha/lc-sso-system.git'
            echo "Project was cloned!"
        }
    }

    stage('Build Docker Image') {
        steps {
            script {
                dockerImage = docker.build("dapravith99/lc-sso-system:${env.BUILD_NUMBER}")
                echo "Build docker images successfully ~!"
            }
        }
    }

    stage('Deploy Docker Compose') {
        steps {
            script {
                sh "$DOCKER_COMPOSE_PATH up -d"
                echo "Docker Compose has been deployed."
            }
        }
    }

    stage('Push Docker Image') {
        steps {
            script {
                docker.withRegistry('', registryCredential) {
                    dockerImage.push()
                    echo "Docker image pushed successfully: ${env.IMAGE_TAG}"
                }
            }
        }
    }
```

Figure 45: Logic stages pipelines

The figure illustrates setting up an "on commit" pipeline in GitLab for Jenkins integration:

- Log in to GitLab and open the project.
- Go to "Settings" > "Webhooks."
- Add Jenkins webhook URL and token.
- Set the trigger condition and add the webhook.
- Test the setup and verify the status code.

Jenkinsfile

```
    post {
        success {
            script {
                // Use the sh command to run the script and ensure the path is correct
                sh 'chmod +x ./1-success-deploy.sh'
                sh './1-success-deploy.sh'
            }
        }

        failure {
            script {
                // Use the sh command to run the script and ensure the path is correct
                sh 'chmod +x ./2-failure-deploy.sh'
                sh './2-failure-deploy.sh'
            }
        }
    }

}

```

Figure 46: Alert messages action via telegram

The Jenkins file snippet configures post-build activities. If the build is successful, it runs **1-success-deploy.sh.** If the build fails, it runs **2-failure-deploy.sh**. This ensures the appropriate script is executed based on the build outcome.

1-success-deploy.sh

```
#!/bin/bash

Telegram_CHATID=${Telegram_CHATID}
Telegram_BotToken=${Telegram_BotToken}

L="------------------------------------------------------------"

# Setting the timezone for the date command
export TZ=Asia/Phnom_Penh

DATE="$(date '+%Y-%m-%d')"
TIME="$(date '+%I:%M:%S %p')"

# Use the adjusted DATE and TIME for the commit log
Log=$(git log -n 1 --pretty=format:"<b>COMMITER</b>: %cN %n<b>DATE</b>: ${DATE} %n<b>TIME</b>: ${TIME} %n<b>MESSAGE</b>: %s")

Server="<b>Server</b>: Development-API-System"

Domain_Name="<b>Domain Name</b>: https://api.adms.itcintern.xyz/api"

Port="<b>Port in Production Server</b>: 4400"

Database_Name="<b>Database Name</b>: lc-adms-db"

MongoDB_Port="<b>MongoDB Port</b>: 27017"  # Assuming default MongoDB port

# Success message with domain name, port, database name, and MongoDB port included
MSG="${L}%0A<b>PROJECT</b>: LC-Book-Inventory-API %0A<b>APPLICATION</b>: API Swagger App %0A<b>STATUS</b>: Success%0A<b>VERSION</b>: ${BUILD_NUMBER}%0A${L}%0A${Log}%0A${L}%0A${Server}%0A${L}%0A${Domain_Name}%0A${L}%0A${Port}%0A${L}%0A${Database_Name}%0A${L}%0A${MongoDB_Port}%0A${L}"

if [ -z "${Log}" ]; then
    echo "Commit log is empty."
    exit 1
else
    curl -s -X POST "https://api.telegram.org/bot${Telegram_BotToken}/sendMessage" -d chat_id="${Telegram_CHATID}" -d text="${MSG}" -d parse_mode="HTML"
fi

```

Figure 47: Deploy messages sh file

To configure Bot Father to notified message in telegram bot followed these steps below:

- First, create a new bot on Telegram using the "Bot Father". Search for "Bot Father" in Telegram, then use the **/newbot** command to create a new bot.
- Second, after creating the bot, start a chat with it and send any message.
- Third, retrieve your Chat ID by calling the Telegram API: replace **<YourBotToken>** in https://api.telegram.org/bot<YourBotToken>/getUpdates with the actual token you received from Bot Father.

  For more details information on Telegram bot configuration, please visit the Telegram Bot API documentation.

3.2.8. Portainer.io

Portainer.io is a tool for managing Docker environments. Follow these steps:

- First, ensure Docker is installed.
- Second, pull the Portainer image from Docker Hub and create a data volume.
- Third, run Portainer in a Docker container.
- Next, access the Portainer web interface at http://your_server_ip:9000 or https://your_domain_name.
- Finally, create an admin user and log in to manage Docker resources.

For detailed command-line instructions, refer to the Portainer documentation.

The figure below is displays about the sample portainer.io web dashboard application user interface.

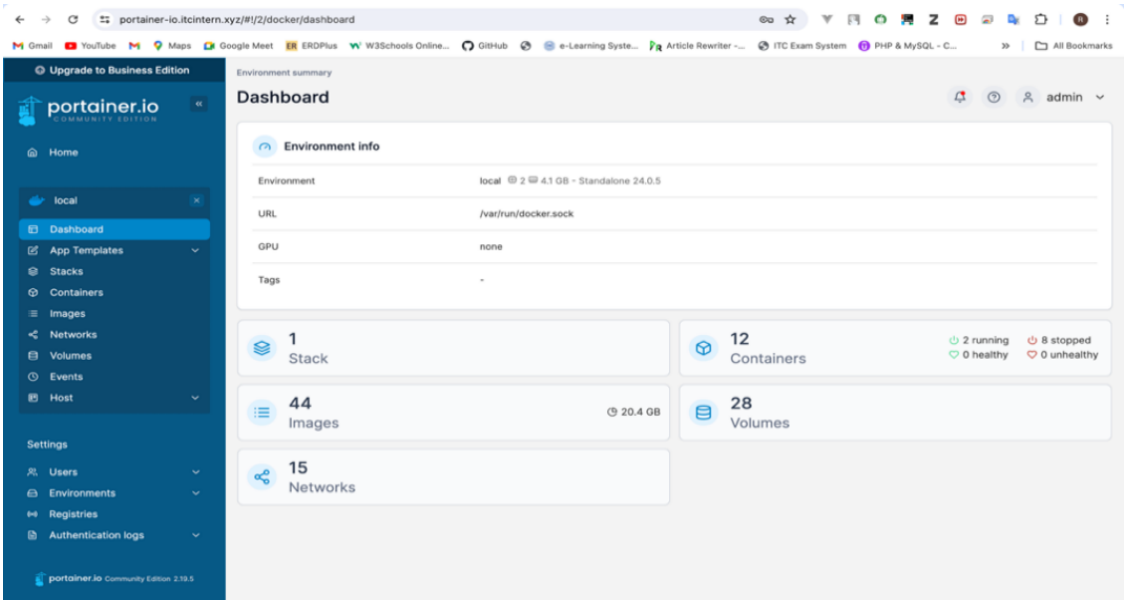

Figure 48: Portainer.io web dashboard

IV. CONCLUSION

4.1. Completed and Uncompleted Tasks

4.1.1. Single Sign-On System tasks

Table 4: SSO system tasks

Tasks	Status	Responsible by
Install and Configure Keycloak server	**Completed**	ROTHA Dapravith
User Authentication	**Completed**	ROTHA Dapravith
Role-Based Access Control (RBAC)	**Completed**	ROTHA Dapravith
User Registration	**Completed**	ROTHA Dapravith
User Profile Management	**Completed**	ROTHA Dapravith
Administrator Authentication	**Completed**	ROTHA Dapravith
Session Management	**Completed**	ROTHA Dapravith
Deployment	**Completed**	ROTHA Dapravith
Integrate SSO system with another Web Client	**Uncompleted**	ROTHA Dapravith

4.1.2. DevOps tasks

Table 5: DevOps overall tasks

Tasks	Status	Responsible by
Manage Version Control	**Completed**	ROTHA Dapravith
CI/CD Integration with Gitlab and Jenkins	**Completed**	ROTHA Dapravith
Implementation with Docker	**Completed**	ROTHA Dapravith
Config automation deployment	**Completed**	ROTHA Dapravith
Notified Message to Telegram	**Completed**	ROTHA Dapravith
Configure domain name with Nginx	**Completed**	ROTHA Dapravith
Implementation with PM2 Server	**Completed**	ROTHA Dapravith

4.2. Strong Point

After I worked and developed on this project, I found some strong points in this system including:

- Keycloak provides centralized management of user identities across all applications.
- Ensures robust security with features like brute force attack protection, strict password policies, and two-factor authentication.
- Seamlessly integrates with protocols such as OAuth 2.0, SAML, and OpenID Connect.
- Allows for detailed role-based access control.

Here are the strong points for DevOps which is:

- Packages applications and their dependencies into containers for consistent performance across different environments.
- Automates the integration and deployment process, making code updates faster and more reliable.

4.3. Weak Point

The system has developed with several strong points, however, there are still some points that need improvement including:

- Complex setup, especially for intricate security needs
- High resource demand for large user bases and high traffic
- Steep learning curve for new administrators and developers.

There are some weak points for DevOps which is:

- Complex management of Docker, Nginx, and Jenkins pipelines
- High system resource usage from multiple containers and pipelines
- Regular maintenance and updates needed for security and performance, increasing administrative workload.

4.4. Difficulties

During my internship, I encountered several challenges such as:

- Faced challenges due to lack of prior work experience.
- Implemented advanced security measures and troubleshooted code issues.
- Completed project on time, expanding knowledge within a tight schedule.
- Internship enriched technical skills and adaptability to new technologies.

4.5. Experiences

During my internship, which lasted for three months and half, I had the opportunity to gain valuable experiences and improve my skills in various areas such as:

- Familiar with new technologies and adapt to workplace environments.
- Familiarity with cloud computing services like Digital Ocean and applying integrated concept DevOps practice into Development processes.
- Learning about CICD pipelines to apply with real practice work for integrating any system projects.

4.6. Perspectives

If I had more available time to progress on this project, I want to fill out the lack of any features for improve this project. There are functionalities that I should do in this project:

- Apply to Microservices with SSO system.
- Enhance security and improve user experience.
- Learn new DevOps tools like Ansible, Kubernetes, Grafana and cloud computing services like AWS.
- Completed with undone tasks.

4.7. Summary

During my three-and-a-half-month internship at **LC-MEF**, I integrated Single Sign-On (SSO) with **Keycloak** and implemented DevOps practices. I configured **Keycloak** for seamless authentication across multiple applications, enhancing security and user experience. Using Docker, I containerized applications and set up Nginx as a reverse proxy for efficient traffic management.

Furthermore, I automated CI/CD pipelines with Jenkins, streamlining build, test, and deployment processes. Additionally, I integrated a Telegram bot for real-time build and deployment notifications. This project enhanced my technical skills and soft skills, including teamwork and time management. I am grateful to **LC-MEF** and the **GIC** department for their support and guidance.

REFERENCES

[1] Keycloak. (n.d). Retrieved May 24, 2024, from https://www.keycloak.org/

[2] Angular. (n.d). Retrieved May 24, 2024, from https://angular.io/

[3] Docker. (n.d). Retrieved May 24, 2024, from https://www.docker.com/

[4] Nginx. (n.d). Retrieved May 24, 2024, from https://nginx.org/

[5] Jenkins. (n.d). Retrieved May 24, 2024, from https://www.jenkins.io/

[6] MYSQL. (n.d). Retrieved May 24, 2024, from https://www.mysql.com/

[7] MongoDB. (n.d). Retrieved May 24, 2024, from https://www.mongodb.com/

[8] PM2. (n.d). Retrieved May 24, 2024, from https://pm2.keymetrics.io/

[9] YAML. (n.d). Retrieved May 24, 2024, from https://yaml.org

[10] Visual Studio Code. (n.d). Retrieved May 24, 2024, from https://code.visualstudio.com

[11] SSH. (n.d). Retrieved May 24, 2024, from https://www.openssh.com/

[12] Termius. (n.d). Retrieved May 24, 2024, from https://termius.com

[13] SourceTree. (n.d). Retrieved May 24, 2024, from https://www.sourcetreeapp.com

[14] GitLab. (n.d). Retrieved May 24, 2024, from https://gitlab.com

[15] Digital Ocean. (n.d). Retrieved May 24, 2024, from https://www.digitalocean.com

[16] Postman. (n.d). Retrieved May 24, 2024, from https://www.postman.com

[17] Name.com. (n.d). Retrieved May 24, 2024, from https://www.name.com

[18] Draw.io. (n.d). Retrieved May 24, 2024, from https://app.diagrams.net/

[19] Telegram. (n.d). Retrieved May 24, 2024, from https://telegram.org/

APPENDIX

Appendix A: Screenshot of Single Sign-On System with Keycloak

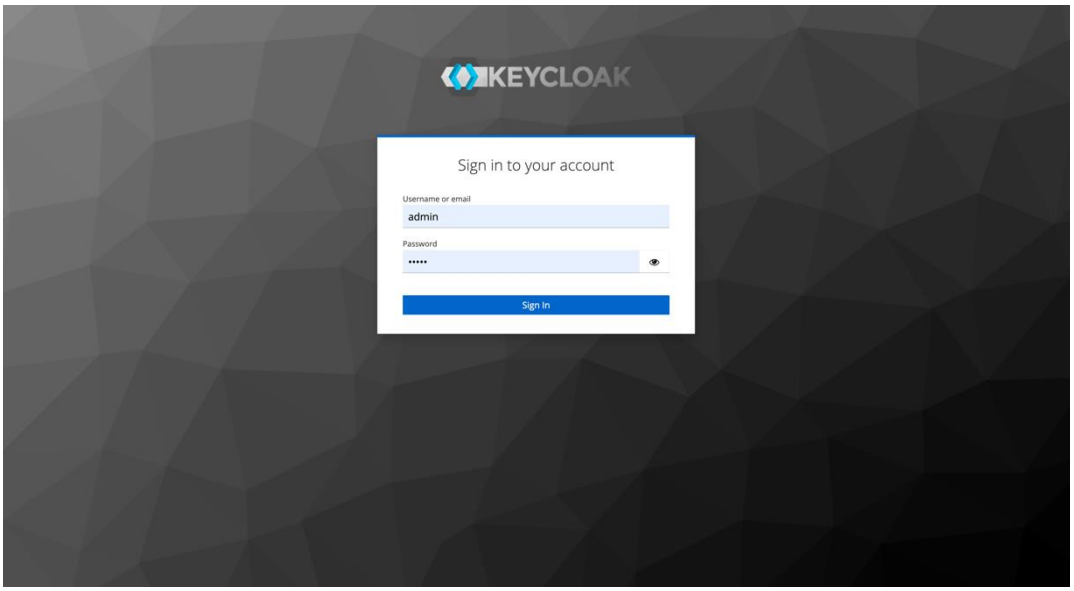

Figure 49: Admin login page

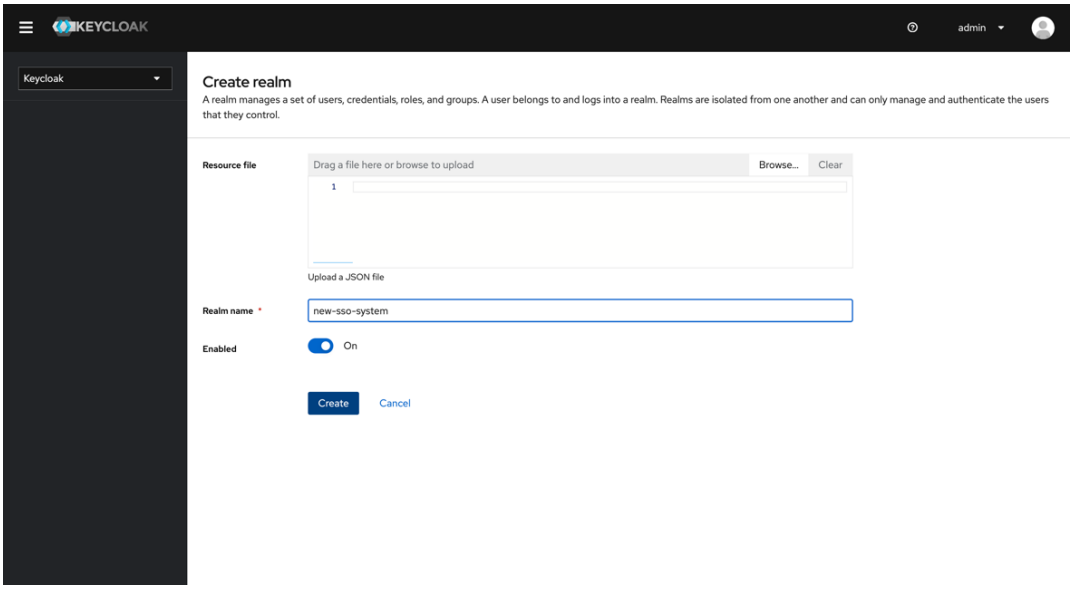

Figure 50: Create realms in keycloak

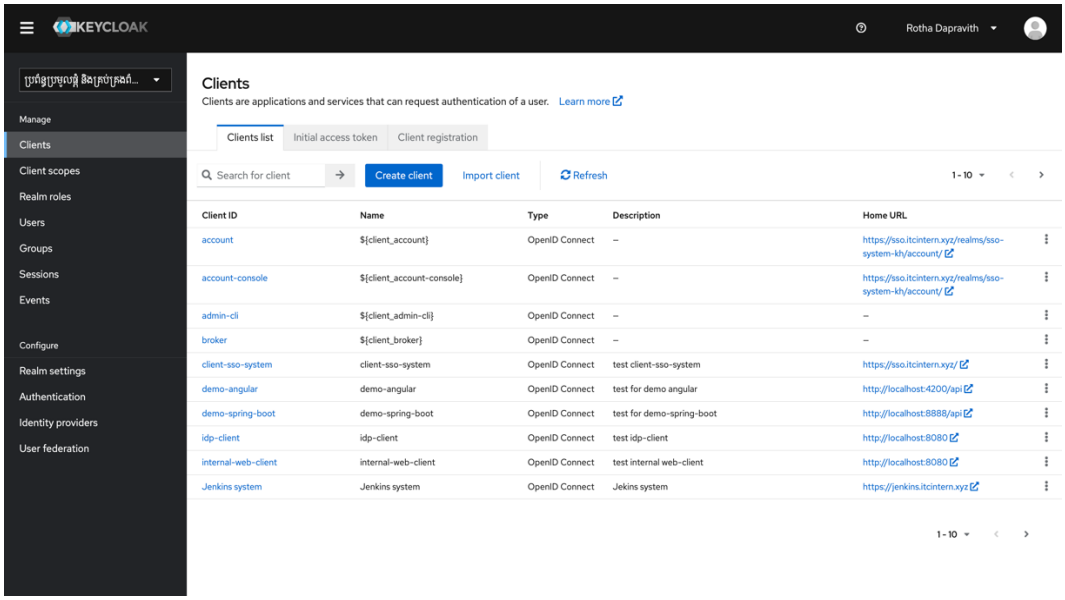

Figure 51: List of clients

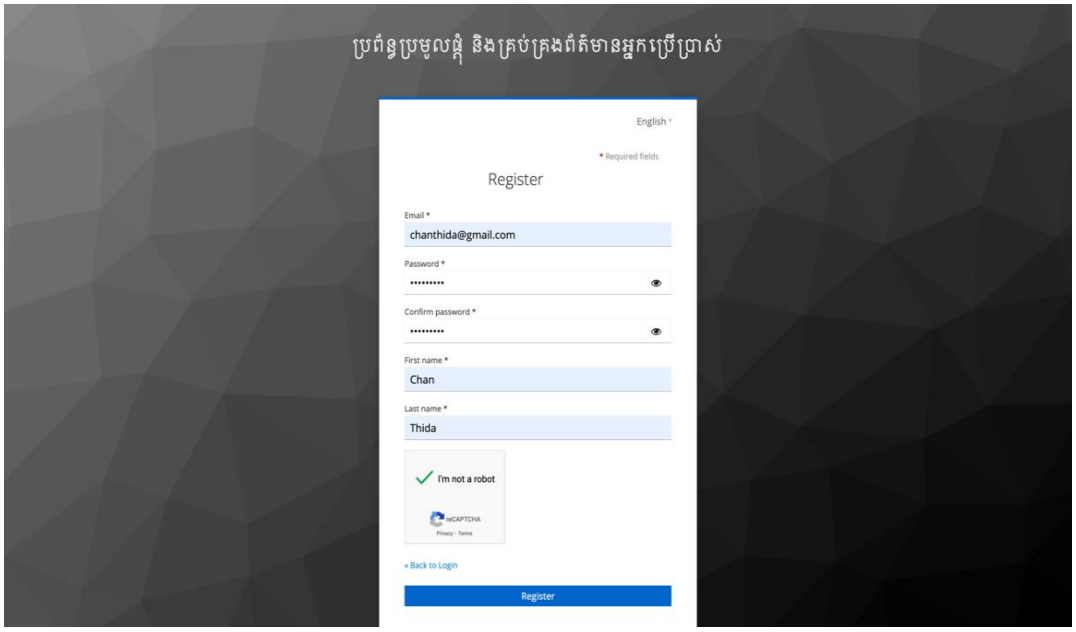

Figure 52: User registration page

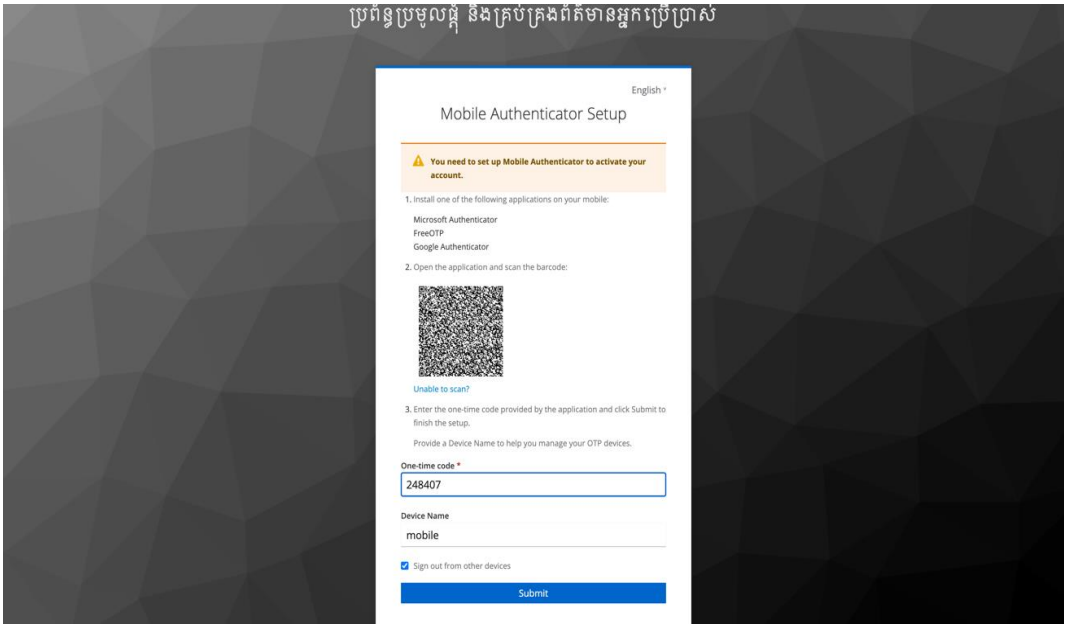

Figure 53: Verify screen with 2FA

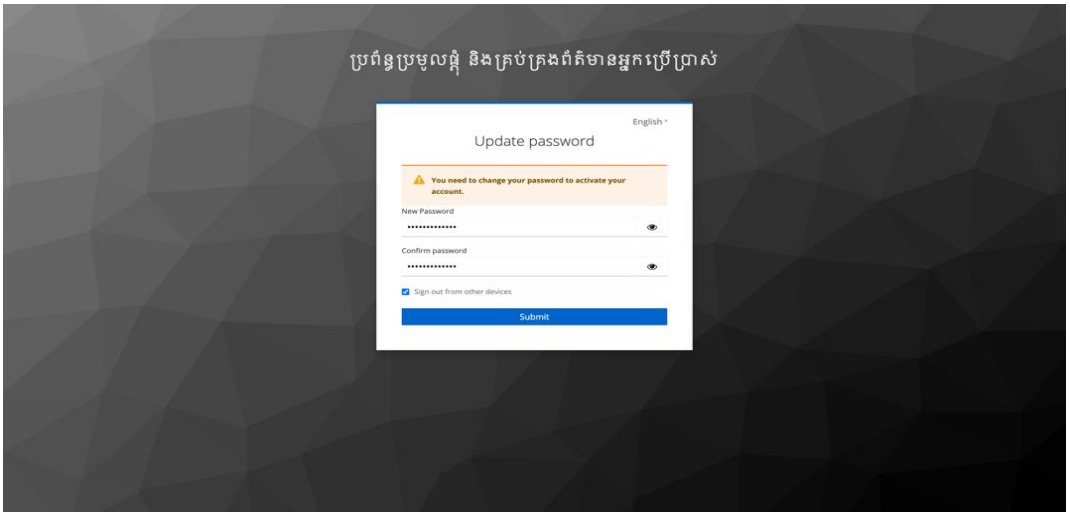

Figure 54: Update password screen

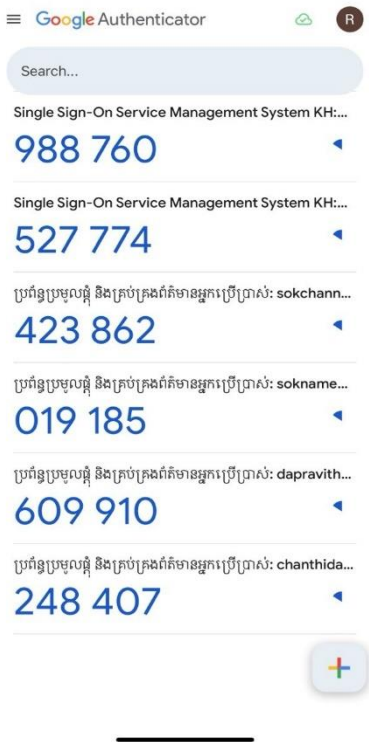

Figure 55: OTP code screen

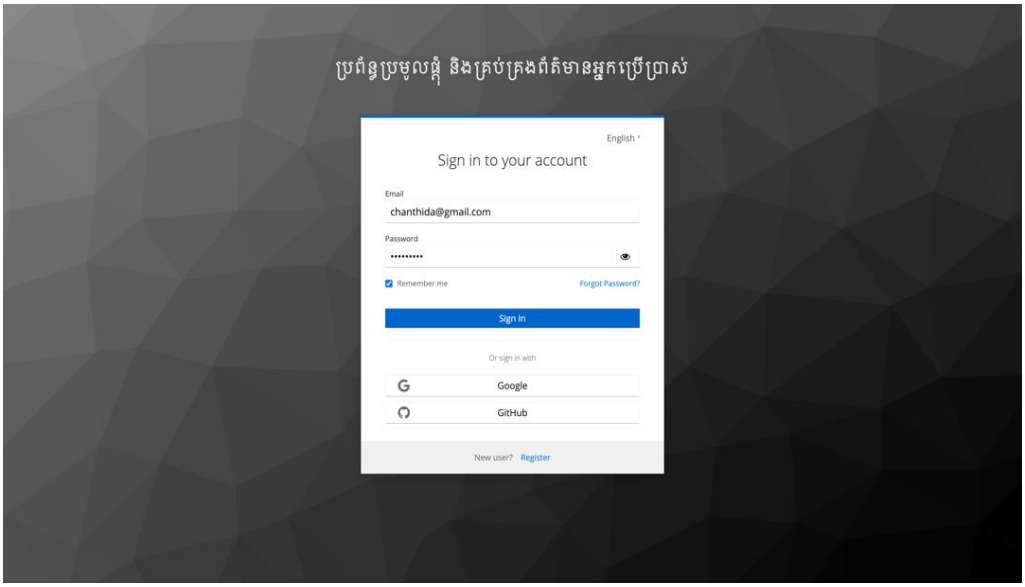

Figure 56: Client login page

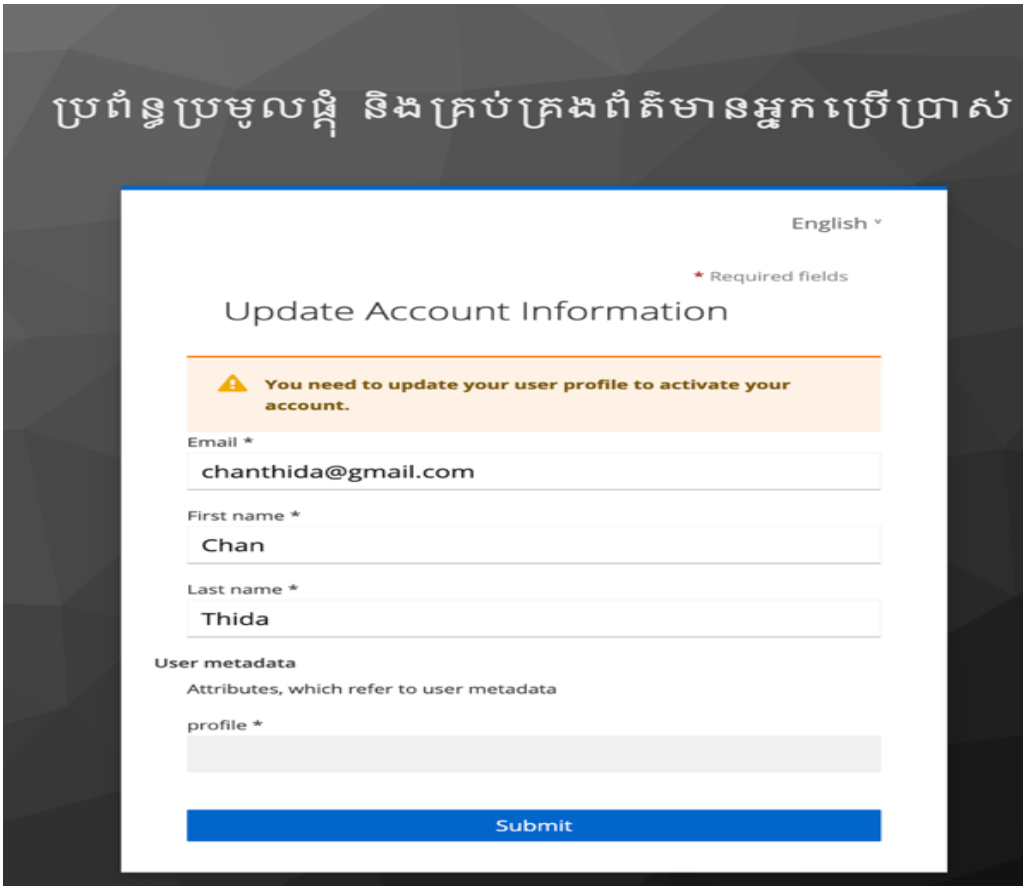

Figure 57: Update user account

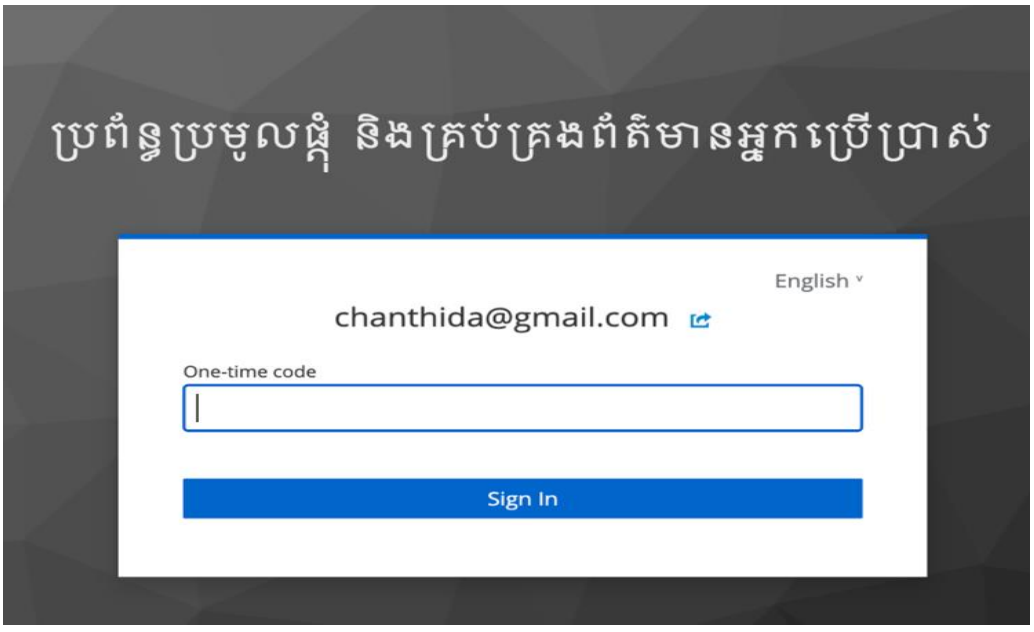

Figure 58: Input OTP code

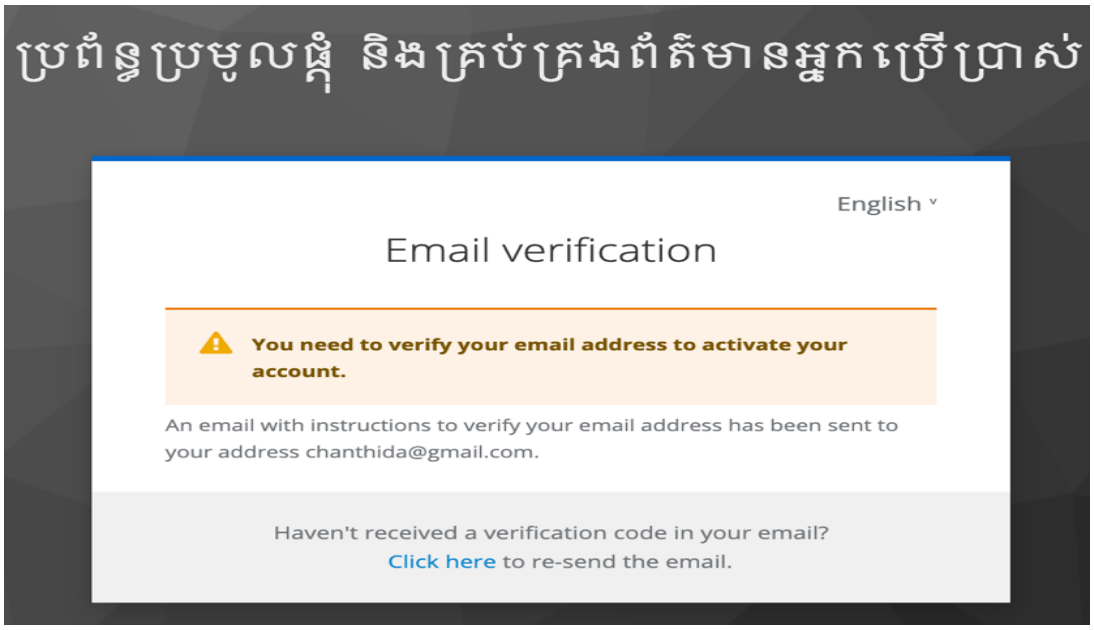

Figure 59: Email verified page

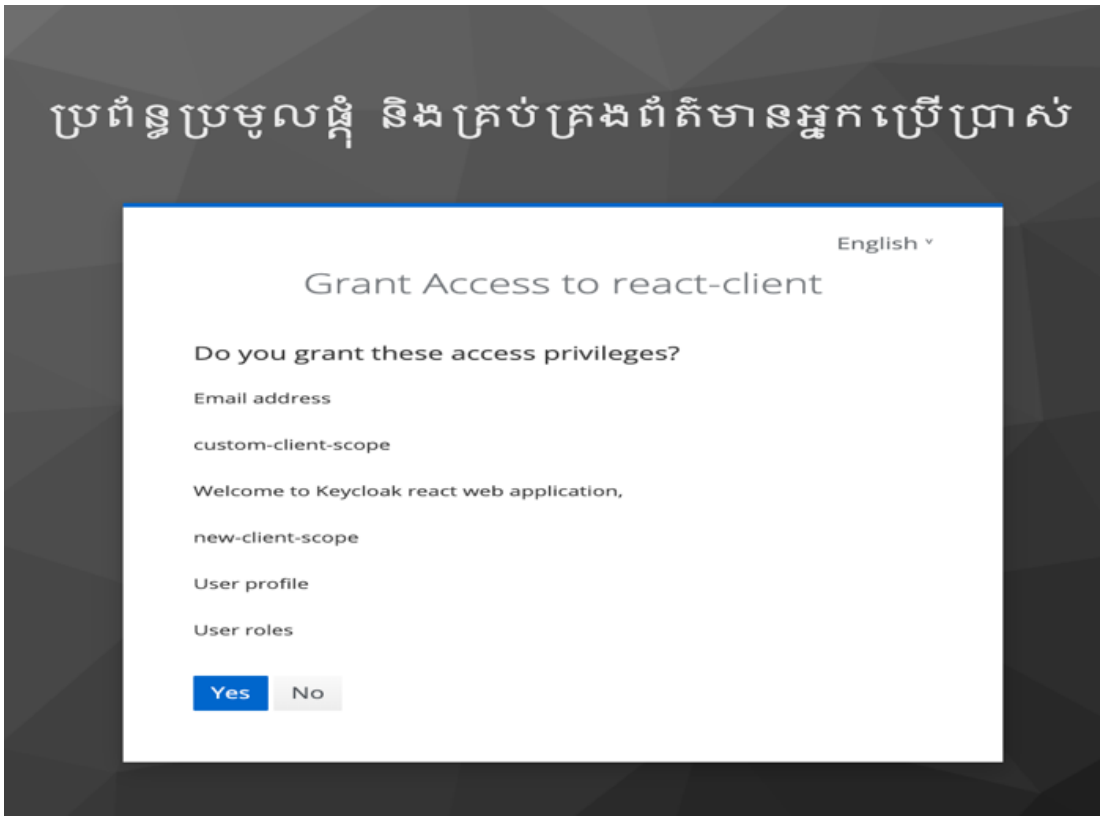

Figure 60: Grant access action to client apps

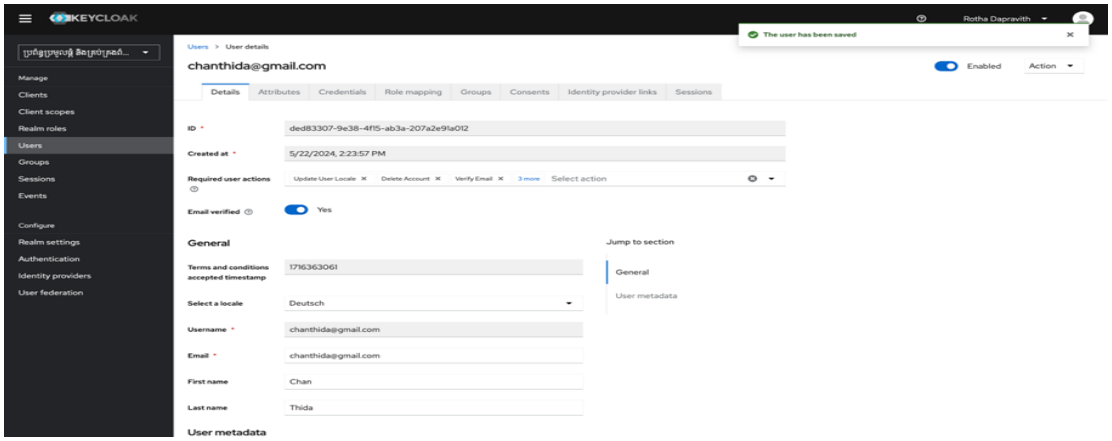

Figure 61: Admin approval email

Figure 62: Access to client web apps

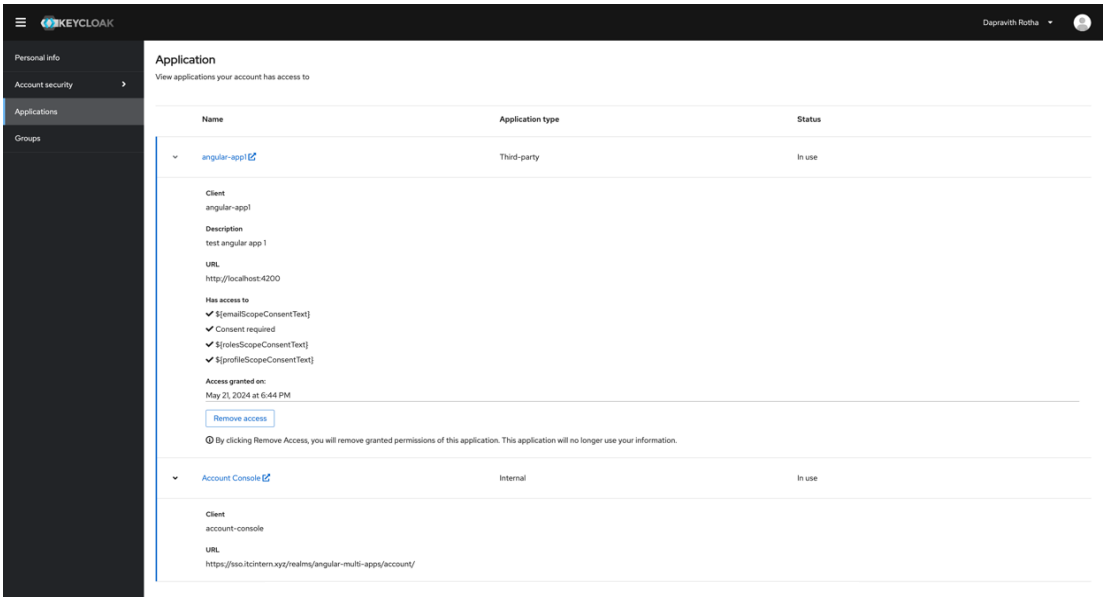

Figure 63: View applications in user account

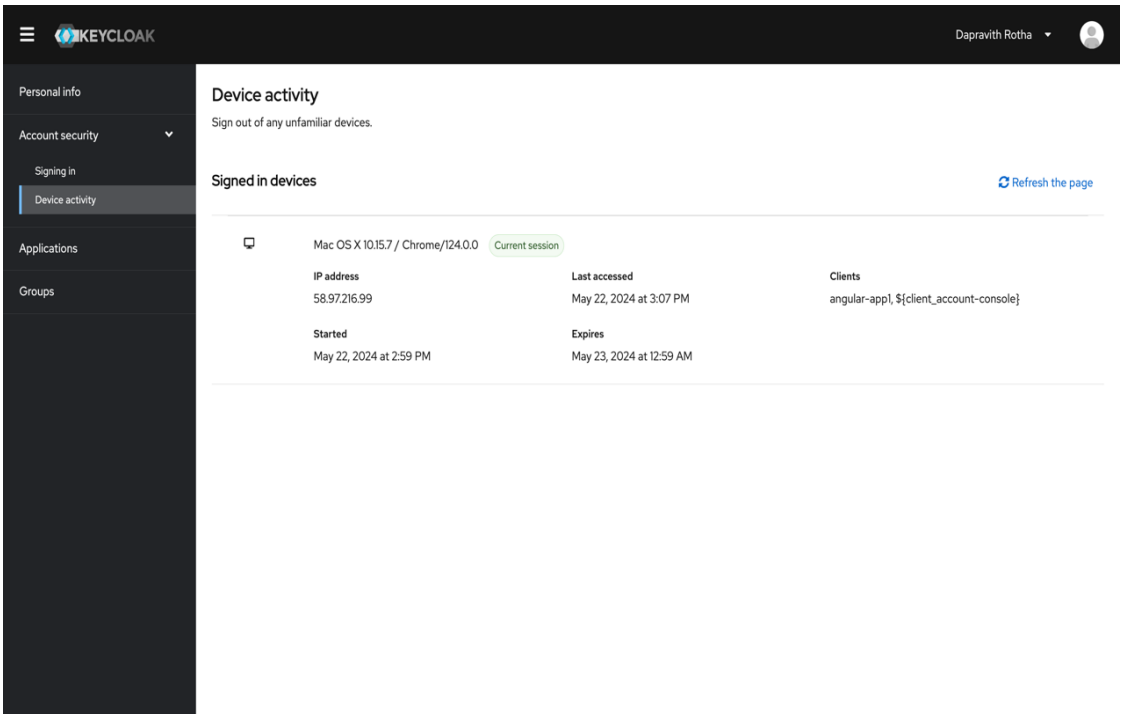

Figure 64: Device activity

Appendix B: Screenshot of DevOps

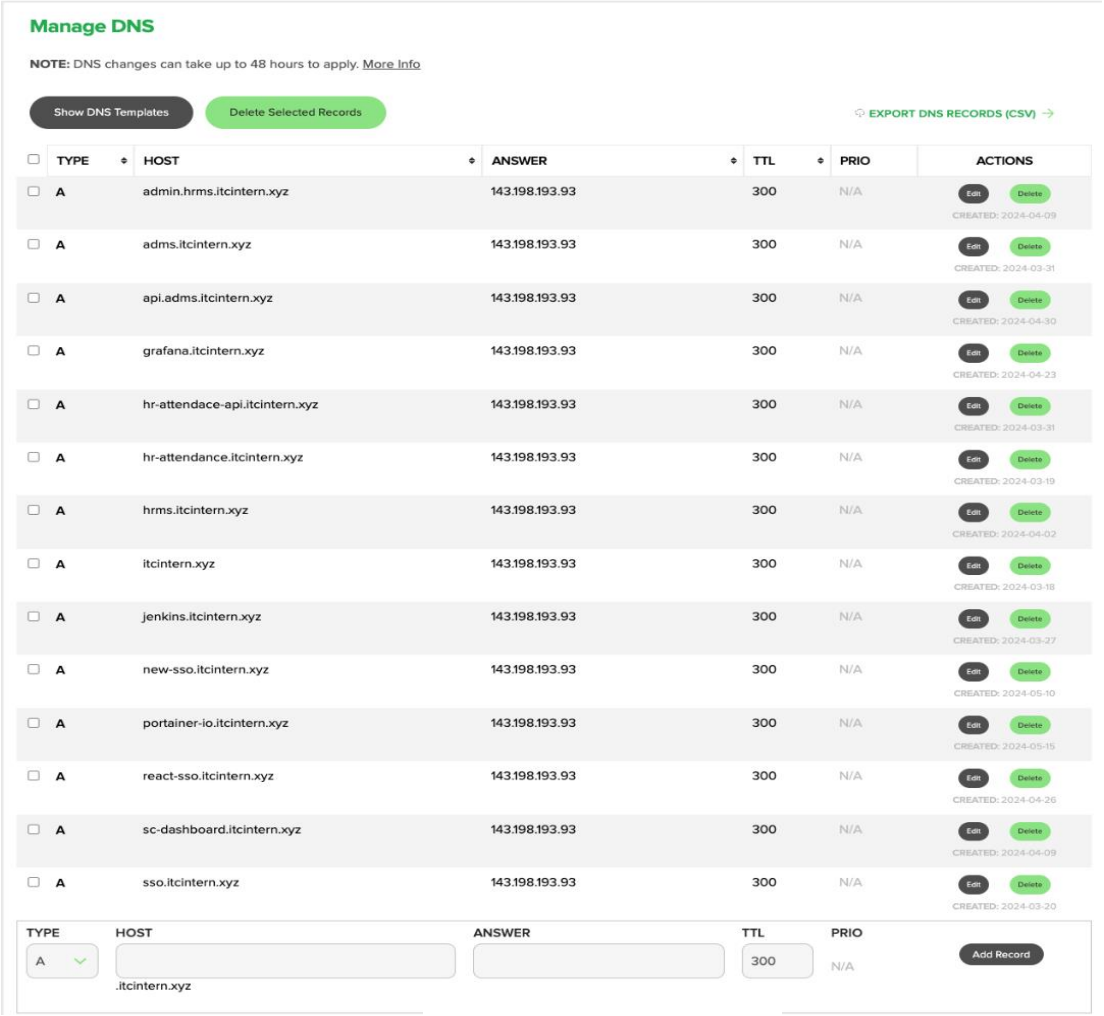

Figure 65: List of all DNS

```
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80                         ALLOW       Anywhere
801                        ALLOW       Anywhere
8080                       ALLOW       Anywhere
8081                       ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
Nginx Full                 ALLOW       Anywhere
802                        ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
27017                      ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443                        ALLOW       Anywhere
4100                       ALLOW       Anywhere
8082                       ALLOW       Anywhere
4200                       ALLOW       Anywhere
4300                       ALLOW       Anywhere
8083                       ALLOW       Anywhere
9082                       ALLOW       Anywhere
9080                       ALLOW       Anywhere
9082/tcp                   ALLOW       Anywhere
8443                       ALLOW       Anywhere
8081/tcp                   ALLOW       Anywhere
2379                       ALLOW       Anywhere
8020                       ALLOW       Anywhere
3001                       ALLOW       Anywhere
4400                       ALLOW       Anywhere
8082/tcp                   ALLOW       Anywhere
9000                       ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
8000                       ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
8443/tcp                   ALLOW       Anywhere
```


Figure 66: List of ufw port

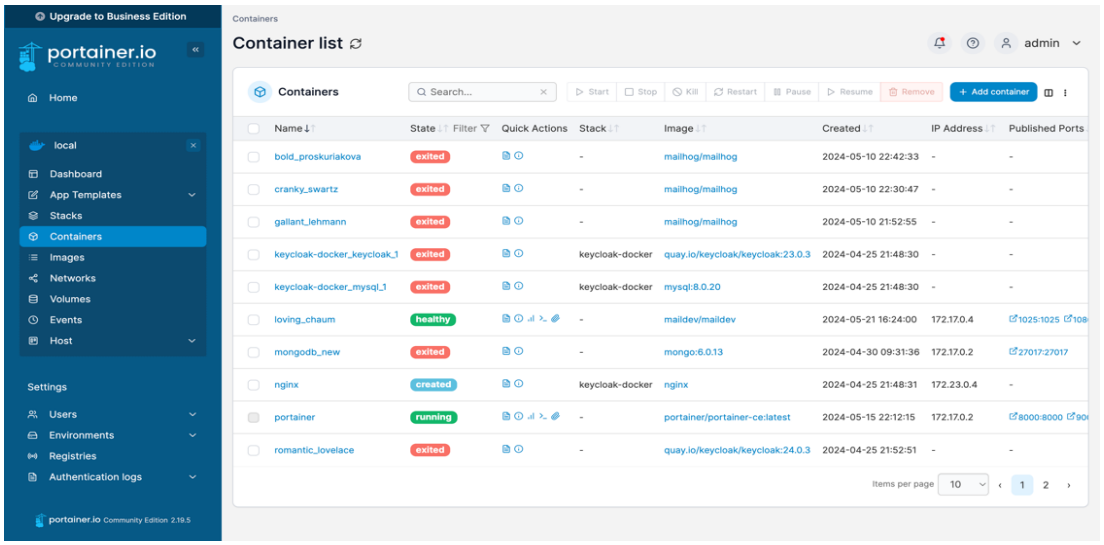

Figure 68: List of all containers

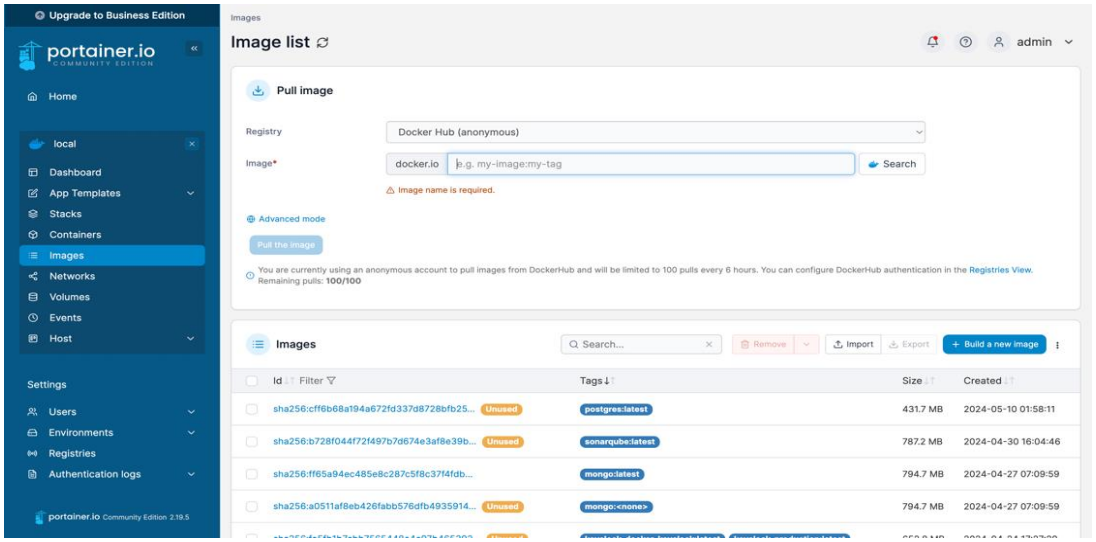

Figure 69: List of all images

```
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:~# cd /etc/nginx/sites-available/
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:/etc/nginx/sites-available# ls -al
total 84
drwxr-xr-x 2 root root 4096 Jun 12 10:53 .
drwxr-xr-x 8 root root 4096 Jun 14 14:40 ..
-rw-r--r-- 1 root root 1945 Apr  9 11:26 admin.hrms.itcintern.xyz
-rw-r--r-- 1 root root 1484 Apr 22 02:38 adms.itcintern.xyz
-rw-r--r-- 1 root root 1035 Mar 27 02:07 angular-sso-app
-rw-r--r-- 1 root root 2135 May 29 02:43 api.adms.itcintern.xyz
-rw-r--r-- 1 root root 6136 May 15 15:56 default
-rw-r--r-- 1 root root 1727 Apr 23 13:05 grafana.itcintern.xyz
-rw-r--r-- 1 root root 1660 Jun  7 02:35 hrms.itcintern.xyz
-rw-r--r-- 1 root root 1104 Apr 22 03:51 jenkins.itcintern.xyz
-rw-r--r-- 1 root root 1656 May 10 15:54 new-sso.itcintern.xyz
-rw-r--r-- 1 root root 8055 Apr 19 03:31 package-lock.json
-rw-r--r-- 1 root root   53 Apr 19 03:31 package.json
-rw-r--r-- 1 root root  669 May 15 15:53 portainer-io.itcintern.xyz
-rw-r--r-- 1 root root  207 Mar 27 04:03 react-client
-rw-r--r-- 1 root root 2700 Apr 28 10:29 react-sso.itcintern.xyz
-rw-r--r-- 1 root root 1893 Apr  9 09:04 sc-dashboard.itcintern.xyz
-rw-r--r-- 1 root root 1573 May 14 02:41 sso.itcintern.xyz
-rw-r--r-- 1 root root 2453 Mar 27 01:52 vithe-commerce.shop
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:/etc/nginx/sites-available#
```

Figure 70: List of nginx DNS

```
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:~# pm2 list
⇆PM2+ activated | Instance Name: server1-7877 | Dash: https://app.pm2.io/#/r/1ehgpw3g3dxu0qm
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
11	hrms_app_api_prod	default	1.0.0	fork	2007960	8D	4	online	0%	98.3mb	root	enabled
9	lc_book_inventory_api_prod	default	1.0.0	fork	289390	11D	8371	online	0%	0b	root	enabled

```
root@server1-1715328957859-s-2vcpu-4gb-120gb-intel-sgp1-01:~#
```

Figure 71: List of pm2

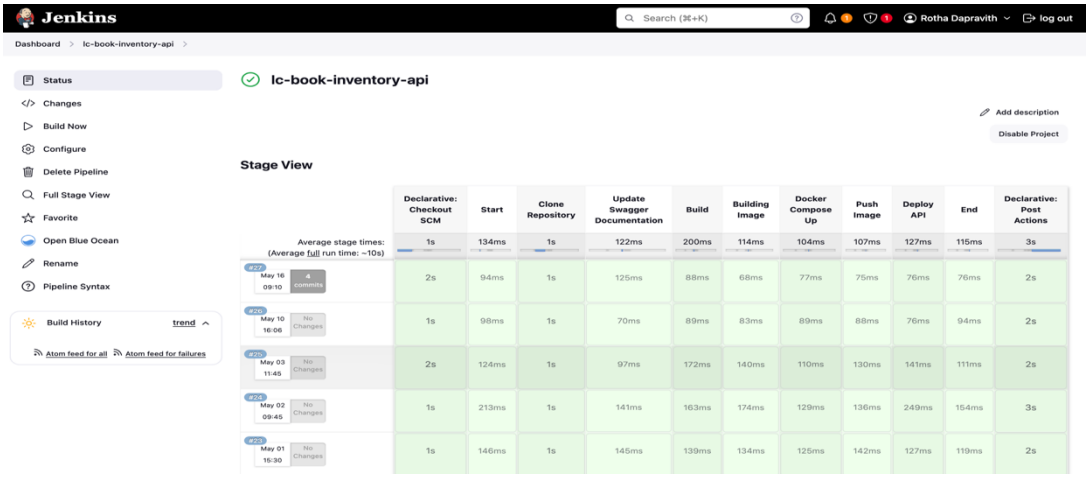

Figure 72: Jenkins stage view

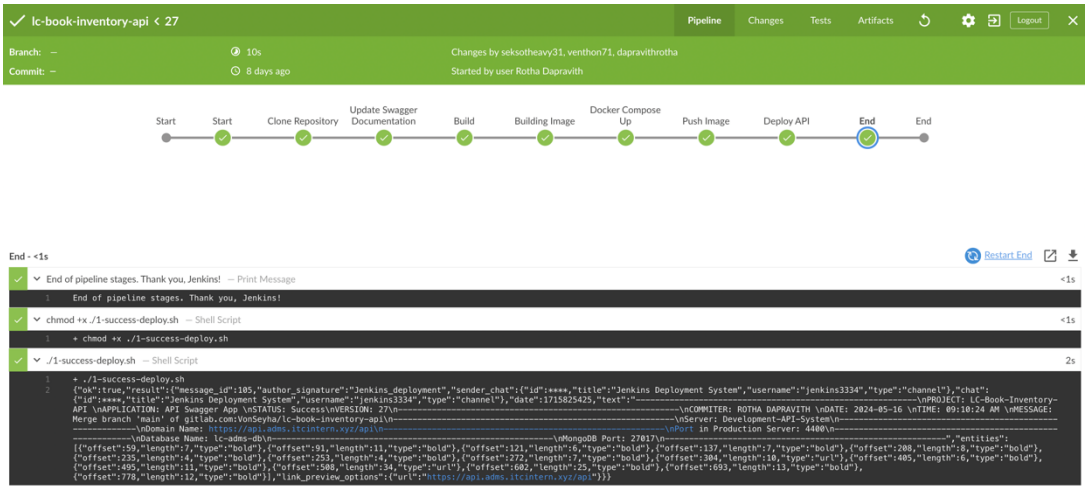

Figure 73: Open blue ocean stage view

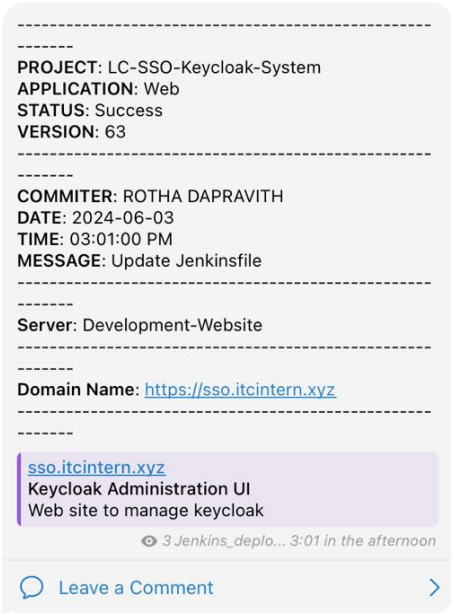

Figure 74: Notified messages actions to telegram