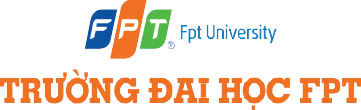


Table of Contents



|  |  |
| --- | --- |
| **SWR302** | |
| **Group Members** | Vũ Văn Đạt – HE141205  Lê Long Dũng - SE04960 Nguyen Hai Dang - SE05366 |
| **Supervisor** | Lecturer: Pham Ngoc Ha |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| **Vehicle Plate Tracking in Car Parking** |

MINISTRY OF EDUCATION AND TRAINING

- Hanoi, <Dec>/<2020> -

[Acknowledgement 4](#_bookmark0)

[Definition and Acronyms 4](#_bookmark1)

1. [Project Introduction 5](#_bookmark2)
   1. [Overview 5](#_bookmark3)
   2. [Product Background 5](#_bookmark4)
   3. [Existing Systems 5](#_bookmark5)

[Advantages 6](#_bookmark6)

[Disadvantages 6](#_bookmark7)

* 1. [Business Opportunity 8](#_bookmark8)
  2. [Software Product Vision 8](#_bookmark9)
  3. [Project Scope & Limitations 8](#_bookmark10)

1. [Project Management Plan 10](#_bookmark11)
   1. [Overview 10](#_bookmark12)
   2. [Management Approach 14](#_bookmark13)
   3. [Master Schedule 16](#_bookmark14)
   4. [Project Organization 17](#_bookmark15)
   5. [Project Communication 17](#_bookmark16)
   6. [Configuration Management 18](#_bookmark17)
2. [Software Requirement Specification 19](#_bookmark18)
   1. [Overall Description 19](#_bookmark19)
   2. [User Requirements 20](#_bookmark20)

[2.2 Use-case list 22](#_bookmark21)

* 1. [Functional Requirements 59](#_bookmark22)

[3.3 Entity Relationship Diagram 69](#_bookmark23)

1. [Software Design Description 70](#_bookmark24)
   1. [Overall Description 70](#_bookmark25)
   2. [System Architecture Design 70](#_bookmark26)
   3. [System Detailed Design 80](#_bookmark27)
   4. [Data & Database Design 128](#_bookmark28)
2. [Software Testing Documentation 128](#_bookmark29)
   1. [Overall Description 128](#_bookmark30)

[2 Test Plan 133](#_bookmark31)

1. [Test Process Model 138](#_bookmark32)
2. [Test Cases 138](#_bookmark33)
3. [Test Result 138](#_bookmark34)
4. [Test Report 140](#_bookmark35)
5. [Release Package & User Guides 146](#_bookmark36)
   1. [Installation Guides 146](#_bookmark37)

3. User Manual **Error! Bookmark not defined.**

**Acknowledgement**

Foremost, we would like to express our sincere gratitude to our supervisor: Mr. Phan Duy Hung for the continuous support of our project development, for his patience, motivation, enthusiasm and immense knowledge. His guidance helped us in all the time of this project. We could not have imagined having a better advisor and mentor for our project. Besides our supervisor, we would like to thank all of our friends who listen to projects and give more suggestions for us, our teammates who have done their best to make to complete the Applying IOT for managing household gas cylinder project.

In addition, we would also like to thanks the instructors at FPT University for all the classes. The instructors have used the experiences and enthusiasm to give us during this period of four years to get here today.

Last but not least, we would like to thank FPT University for giving us this precious opportunity to constantly study and improve ourselves. What we learned through this project will be the basis for us to work well after graduation.

**Definition and Acronyms**

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| BA | Business Analysis |
| BR | Business Rule |
| ERD | Entity Relationship Diagram |
| GUI | Graphical User Interface |
| PM | Project Manager |
| SDD | Software Design Description |
| SPM P | Software Project Management Plan |
| SRS | Software Requirement Specification |
| UAT | User Acceptance Test |
| UC | Use Case |
| API | Application Program Interface |
| IOT | Internet of things |

1. **Project Introduction**
   1. **Overview**
   2. Project Information

* Project name: **Vehicle Plate Tracking in Car Parking**
* Project code: **Vehicle-Car**
* Project group name: **SWR302\_Topic6**
* Product type: IOT, Website and Mobile Application.
  1. Project Team

1. Supervisor

|  |  |  |  |
| --- | --- | --- | --- |
| **Full Name** | **Email** | **Phone Number** | **Title** |
| Pham Ngoc Ha | [HungPD2@fe.edu.vn](mailto:HungPD2@fe.edu.vn) | 0975597339 | Lecturer |

1. Team Members

|  |  |  |  |
| --- | --- | --- | --- |
| **Full Name** | **Email** | **Mobile** | **Role** |
| Nguyen Huu Phuong | [PhuongNHSE05047@fpt.edu.vn](mailto:PhuongNHSE05047@fpt.edu.vn) | 0357123633 | Leader |
| Nguyen Hai Dang | [DangNHSE05366@fpt.edu.vn](mailto:DangNHSE05366@fpt.edu.vn) | 0967611312 | Member |
| Tran Ngoc Thanh | [ThanhTNse04960@fpt.edu.vn](mailto:ThanhTNse04960@fpt.edu.vn) | 0378972120 | Member |
| Vu Duy Hung | [HungVDse05582@fpt.edu.vn](mailto:HungVDse05582@fpt.edu.vn) | 0365386210 | Member |
| Nguyen Manh Ha | [HaNMHE130235@fpt.edu.vn](mailto:HaNMHE130235@fpt.edu.vn) | 0944036163 | Member |

## Product Background

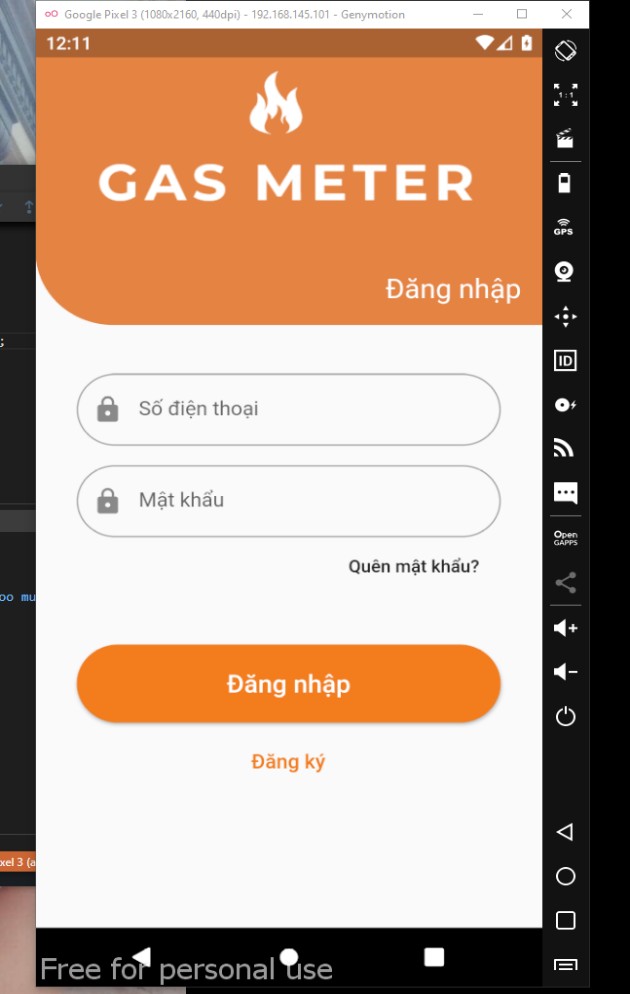
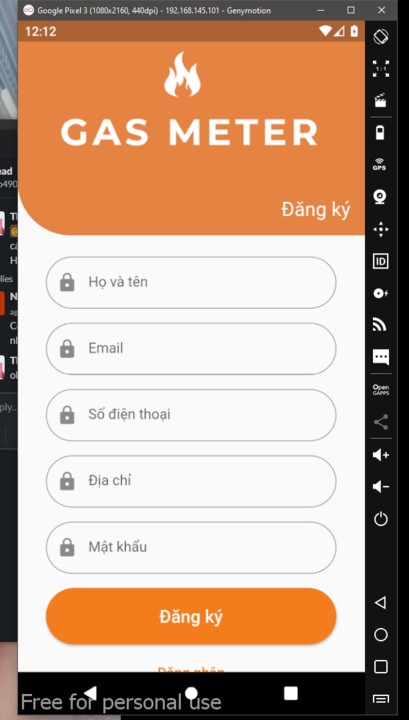
Today, many families are still using gas for cooking and warming during winter. Since they would buy gas at the gas stores, it is essential to have a software that support managing gas exchanges between the store and the customers. Also, there are risks that related to safety while using gas – leaked gas can easily start a fire or explosion. Therefore, we aim to develop a system that manages anything related to manage gas situation between stores and customers.

## Existing Systems

* 1. Mobile application

This software can be downloaded on Google Play.In the future, we can develop on App Store. The mobile app is used by the customers of the gas store. If an user is a new user, he/she can register an account through the app and login. From the app:

* + - User can check the following parameters: gas concentration, temperature, remaining gas, ...
    - Users also receive risk notifications and warnings when there is a fire hazard
    - Update location user through the app.

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| GUI is user-friendly, suppost language: VietNamese. | Available only on Android |
| Freeware, public on Google Play | Need internet to take notification |
| Handy, simple, easy to use anytime, anywhere |  |

* 1. Web application

This software is published publicly on the Internet and it is freeware. It covers bussiness aspects that a gas store would face in a real life situation, such as creating, reading, updating and deleting gas cylinder informations, etc... This software supports only remote access and all data stored on a server. The server uses Database-as-a-Service (DaaS), which brings many benefits such as Security, High Availability, less configuration. That’s a good option and example for us to consider using DaaS.

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| GUI is user-friendly, support language: Vietnamese | Only support remote access (Web), no support for offline use (Desktop apps) |
| There is always an employee flow the website 24/7 | Only support via hotline, not via facebook, zalo, etc… |
| As long as the internet is available, it can be accessed at all times. | Low security, easy to attack by hackers |



* 1. Backend system

Backend system is made base on Spring Boot. Spring Boot is a project developed by JAV (java language) in Spring framework ecosystem. It helps our programmers to simplify the process of programming an application with Spring, focusing only on developing business for the application.

* 1. Embedded system

Backend system is made base on Arduino. Arduino is an open-source electronics platform based on easy-to-use hardware and software. It’s simple and accessible user experience, bring many benefits such as high Availability, less configuration. That’s a good option and example for us to consider using Adruino.

## Business Opportunity

Today every apartment in Vietnam has at least one gas stove. So this is a good opportunity for the gas trading industry in Vietnam to develop.

However, the safety issue when using the gas stove is a very important issue for users. As for businesses, they need to manage the amount of gas consumed as well as their customers.

To solve this problem, an IoT system will help users keep track of indicators such as temperature in the room, amount of gas, ... There is also a web service that helps stores to manage customer management as well as the amount of gas sold.

## Software Product Vision

With the Applying IOT for managing household gas cylinder system (Gas-Meter), we aim to optimize the operational structure of managing gas cylinder for each customer, both versions of the software brings the best result and experience to our customer. Specifically, in the Web Application version, the software assists management functions of gas cylinder which are currently using by customers for admin from head to toe while the Mobile Application version provides to the customers the ability of knowing the situation of his/her current gas so that he/she could contact the store when there is any problem. The system supports both the stores and the customers of the stores in terms of managing problem with gas cylinder.

## Project Scope & Limitations

Gas-Meter is not system give me information about gas cylinder, temperature environment, warning when there is fire or explosion on the phone has a web-based management system that helps the gas- store operate easily and flexibly.

6.1 Major Features

* Mobile application:
  + Role: User
    - Allows users to view and change personal information
    - Allows users to monitor device information such as: gas concentration, remaining gas, ambient temperature, and device battery
    - Send notification for user.
* Web application:
  + Role: Store
    - Allows Store to manage list user information that belongs to your store
    - Allows Store to manage list contract information that belongs to your store
    - Allows Store to manage list broken device information that belongs to your store
    - Allows Store to view map show location of devices and it’s status
  + Role: Admin
    - Allows Admin to manage list store.
    - Allows Admin to manage list user information in each store
    - Allows Admin to manage list contract information in each store
    - Allows Admin to manage list all broken device information
    - Allows Admin to view map show location of devices and it’s status

# Project Management Plan

## Overview

* 1. WBS & Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **WBS Item** | **Complexit y** | **Est. Effort**  **(man- days)** |
| ***1*** | ***Pre-Initiating Phase*** |  | ***3*** |
| 1.1 | Select project manager | Simple | 0.5 |
| 1.2 | Determine the role of team members | Simple | 0.5 |
| 1.3 | Choose topic | Simple | 1.5 |
| 1.4 | Register Capstone Project | Simple | 0.5 |
| ***2*** | ***Initiating Phase*** |  | ***5*** |
| 2.1 | Kick off | Simple | 0.5 |
| 2.2 | Identify stakeholders | Medium | 1 |
| 2.3 | Meet instructor | Simple | 1 |
| 2.4 | Develop project introduction document | Simple | 2.5 |
| ***3*** | ***Planning Phase*** |  | ***14*** |
| 3.1 | Define scope | Simple | 1.5 |
| 3.2 | Choose working model process | Simple | 1.5 |
| 3.3 | Select tools and techniques | Simple | 1.5 |
| 3.4 | Develop project schedule | Medium | 3 |
| 3.5 | Develop risk management plan | Medium | 3 |
| 3.6 | Organize project resources | Simple | 1.5 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3.7 | Develop software project management plan | Simple | 2 |
| ***4*** | ***Executing*** |  | ***299*** |
| **4.1** | **Iteration 1: Develop main functions** |  | **171** |
| 4.1.1 | Analysis | Complex | 5 |
| 4.1.2 | Requirements |  | 10 |
| 4.1.2.1 | Software Requirement Specifications | Medium | 5 |
| 4.1.2.2 | Software Design Requirements | Medium | 5 |
| 4.1.3 | Design |  | 10 |
| 4.1.3.1 | Design UI | Medium | 5 |
| 4.1.3.2 | Design database | Medium | 5 |
| 4.1.4 | Coding |  | 105 |
| 4.1.4.1 | Back-end | Complex | 30 |
| 4.1.4.2 | Web Application | Complex | 30 |
| 4.1.4.3 | Mobile Application | Complex | 30 |
| 4.1.4.4 | Fix bugs | Medium | 15 |
| 4.1.5 | Testing |  | 39 |
| 4.1.5.1 | Create test cases for unit test | Medium | 5 |
| 4.1.5.2 | Unit test | Medium | 10 |
| 4.1.5.3 | Create test cases for integration test | Medium | 3 |
| 4.1.5.4 | Integration test | Medium | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.1.5.5 | Create test cases for system test | Medium | 3 |
| 4.1.5.6 | System test | Medium | 5 |
| 4.1.5.7 | Create test cases for acceptance test | Medium | 3 |
| 4.1.5.8 | Acceptance test | Medium | 5 |
| 4.1.6 | Evaluation | Simple | 2 |
| **4.2** | **Iteration 2: Develop all functions** |  | **128** |
| 4.2.1 | Analysis | Complex | 5 |
| 4.2.2 | Requirements |  | 10 |
| 4.2.2.1 | Software Requirement Specifications | Medium | 5 |
| 4.2.2.2 | Software Design Requirements | Medium | 5 |
| 4.2.3 | Design |  | 8 |
| 4.2.3.1 | Design UI | Medium | 3 |
| 4.2.3.2 | Design database | Medium | 5 |
| 4.2.4 | Coding |  | 80 |
| 4.2.4.1 | Back-end | Complex | 30 |
| 4.2.4.2 | Web Application | Complex | 30 |
| 4.2.4.3 | Mobile Application | Complex | 10 |
| 4.2.4.4 | Fix bugs | Medium | 10 |
| 4.2.5 | Testing |  | 23 |
| 4.2.5.1 | Create test cases for unit test | Medium | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.2.5.2 | Unit test | Medium | 5 |
| 4.2.5.3 | Create test cases for integration test | Medium | 2 |
| 4.2.5.4 | Integration test | Medium | 3 |
| 4.2.5.5 | Create test cases for system test | Medium | 2 |
| 4.2.5.6 | System test | Medium | 3 |
| 4.2.5.7 | Create test cases for acceptance test | Medium | 2 |
| 4.2.5.8 | Acceptance test | Medium | 3 |
| 4.2.6 | Evaluation | Simple | 2 |
| ***5*** | ***Monitoring and Controlling*** |  | ***11*** |
| 5.1 | Meet instructor | Simple | 1 |
| 5.2 | Progress Report | Simple | 10 |
| ***6*** | ***Closing*** |  | ***48*** |
| 6.1 | Complete final report | Medium | 25 |
| 6.2 | Prepare for final project presentation | Medium | 20 |
| 6.3 | Present final project | Medium | 2 |
| 6.4 | Project completed | Simple | 1 |

***Total Estimated Effort (man-days) 380***

* 1. Project Objectives

Objectives:

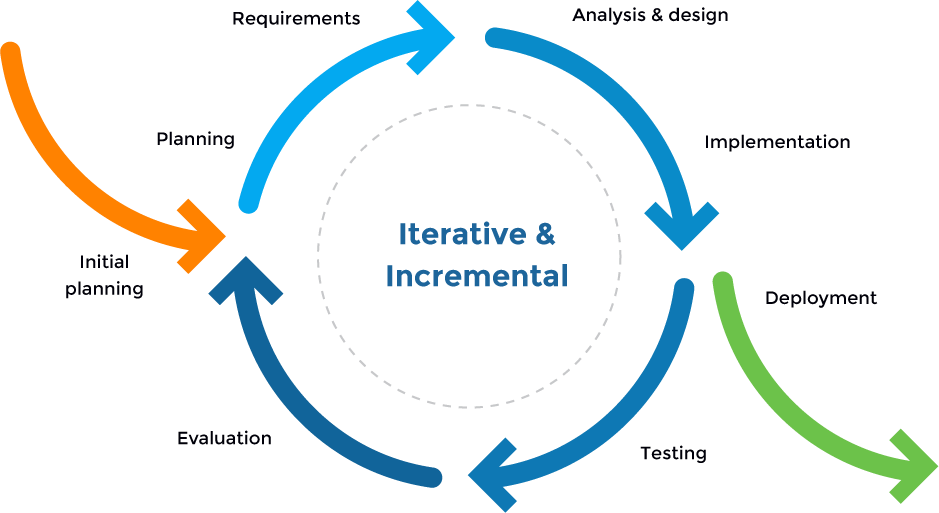
* This project must be finished by 12/20/2020.
* All team members have to follow the task assigned by the PM.
* All team members give their best effort to complete the project.
* Project team members learn new knowledge, new technology
  1. Project Risks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Risk Description** | **Impact** | **Possibility** | **Response Plans** |
| 1 | Project team misunderstand the requirements | High | Low | Have regular meetings to make sure the team understand the requirements and know what they need to do |
| 2 | Team members lack technical skills and knowledge | High | Medium | Coding sub-team leader finds out to train the skills that members are lacking, team members study by themselves |
| 3 | Source code is conflicted | Medium | High | Use backup version, discuss with other members and continue to work |
| 4 | Team members does not complete project on time | High | Medium | Ask for support measures from supervisor |
| 5 | One or more team members abandon the project | High | Low | Find out the reasons, persuade them and reallocate tasks |

## Management Approach

* 1. Project Process

After careful research, the project will apply **Iterative & Incremental Model** to the system development process.



In an Iterative & Incremental model, initially, a partial implementation of a total system is constructed so that it will be in a deliverable state. Increased functionality is added. Defects, if any, from the prior delivery are fixed and the working product is delivered. The process is repeated until the entire product development is completed. The repetitions of these processes are called iterations. At the end of every iteration, a product increment is delivered.

The reasons for the project to choose this model are:

* + - You can develop prioritized requirements first.
    - Initial product delivery is faster.
    - Customers get important functionality early.
    - Lowers initial delivery cost.
    - Each release is a product increment, so that the customer will have a working product at hand all the time.
    - Customers can provide feedback to each product increment, thus avoiding surprises at the end of development.
    - Requirements changes can be easily accommodated.
  1. Quality Management
     + Defect Prevention:
       - Once a defect is found, the related person in charge should be notified immediately.
       - The defect must be assessed carefully such as “how bad is the defect?”, “how long does it take to fix the defect?”.
       - The deadline for fixing defects must be clearly stated.
     + Reviewing:
       - The person in charge must be honest and show no favor over any member. If something goes wrong, that person must notify the person who takes responsibility for that defect.
       - The defect must be logged on Bug Tracking software with detail such as priority.
       - The person who takes responsibility for the found defects must take actions accordingly.
     + Unit Testing:
       - The person in charge must prepare test cases carefully and accurately. The test cases must match well with the system.
       - The defect must be logged on Bug Tracking software with detail such as priority.
       - The person who takes responsibility for the found defects must take action accordingly.
     + Integration Testing
       - The person in charge must prepare test cases carefully and accurately. The test cases must match well with the system and architecture design.
       - The defect must be logged on Bug Tracking software with detail such as priority.
       - The person who takes responsibility for the found defects must take action accordingly.
       - Internal modules within the system work smoothly.
     + System Testing
       - The person in charge must prepare test cases carefully and accurately. The test cases must match well with the system and system design.
       - The defect must be logged on Bug Tracking software with detail such as priority.
       - The person who takes responsibility for the found defects must take action accordingly.
       - System testing test cases cover the entire system functionality and the communication under development with external systems.
     + Acceptance Testing
       - The person in charge must prepare test cases carefully and accurately. The test cases must match well with the system and business requirements.
       - The person who takes responsibility for the found defects must take action accordingly.
       - The defect must be logged on Bug Tracking software with detail such as priority.
       - If there are customers, customers should take part in Acceptance testing.
       - The test should cover non-functional issues such as load and performance defects.
  2. Training Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Training Area** | **Participants** | **When, Duration** | **Waiver Criteria** |
| Bitbucket, Source Tree | All team members | 14/09/2020  (1 day) | Mandatory |
| Angular 8 (Front-end for Web) | All team members | 21/09/2020 -  27/09/2020  (1 week) | Mandatory |
| Spring Boot (Back-end for Web and Mobiles) | All team members | 28/09/2020 -  04/10/2020  (1 week) | Mandatory |
| Fluter (for Android) | Nguyễn Hải Đăng | 05/10/2020 -  08/10/2020  (4 days) | Mandatory |
| Arduino IDE(for Embedded system) | Nguyễn Hữu Phương | 01/10/2010 -  08/10/2020  (1 week) | Mandatory |
| Coding Convention & Bug Logging Convention | All team members | 10/10/2020  (1 day) | Mandatory |

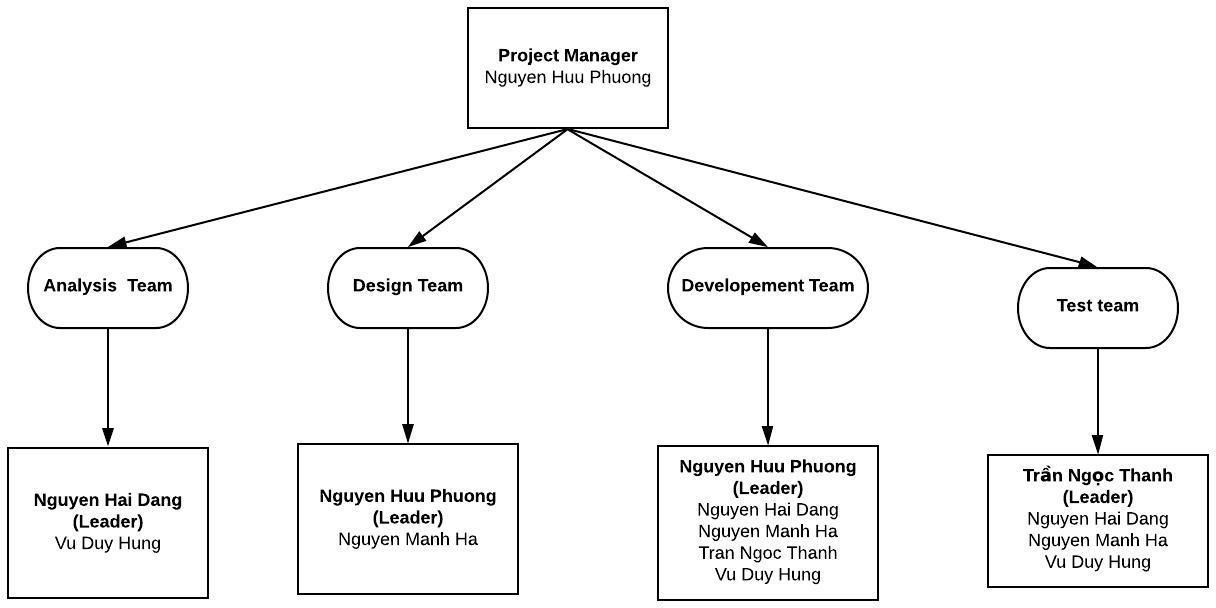
## Master Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Deliverable** | **Due Date** | **Deliverable Scope** |
| 1 | Project Plan | 10/05/2020 | Project Schedule, Risk Management Plan, Resources Management Plan |
| 2 | SRS | 10/12/2020 | SRS |
| 3 | Design | 10/26/2020 | Architecture Design, Detailed Design, Database |
| 4 | Iteration 1 | 11/09/2020 | Code & Unit test, System test cases |
| 5 | Iteration 2 | 11/23/2020 | Code & Unit test, System test cases |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | UAT Package | 11/30/2020 | Codes, System test reports |
| 7 | Final Package | 12/15/2020 | Final Codes & documents, User manual |

## Project Organization

* 1. Team & Structures



* 1. Roles & Responsibilities

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| Project Manager | Nguyen Huu Phuong |
| Analysis Leader | Nguyen Hai Dang |
| Analysis Member | Vu Duy Hung |
| Design Leader | Nguyen Huu Phuong |
| Design Member | Nguyen Manh Ha |
| Technical Leader | Nguyen Huu Phuong |
| Test Leader | Tran Ngoc Thanh |
| Test Member | Nguyen Hai Dang, Nguyen Manh Ha, Vu Duy Hung |

## Project Communication

* 1. Communication Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Communicatio n Item** | **Who/ Target** | **Purpose** | **When, Frequency** | **Type, Tool, Method(s)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Weekly Meeting | Team members  Supervisor | Review members’ work achievements during the week and report the project’s progress and status. | Every Monday | Face to Face |
| Daily Meeting | Team members | Report the progress that members achieved each day. | Daily | Messenger,  Slack, Microsoft Team ,  Gmail |
| Unscheduled Meeting | Team members | Occurred when there's a critical problem that need to be resolved immediately |  | Google Meet |

* 1. External Interface

1. FU Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Contact Person (name, position)** | **Contact address (email, telephone)** | **Responsibility** |
| Supervisor | Phan Duy Hung | [HungPD2@fe.edu.vn](mailto:HungPD2@fe.edu.vn) 0975597339 | * Give instruction to project team * Review deliverables * Supervise project status |

1. Customer Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Contact Person (name, position)** | **Contact address (email, telephone)** | **Responsibility** |
| Supervisor | Phan Duy Hung | [HungPD2@fe.edu.vn](mailto:HungPD2@fe.edu.vn) 0975597339 | * Give instruction to project team * Review deliverables * Supervise project status |

## Configuration Management

* 1. Tools & Infrastructures

|  |  |
| --- | --- |
|  |  |
| **Programming languages** | Angular8, Java Spring Boot. |
| **Framework** | Spring Boot, Flutter |
| **API** | Java Spring Boot |
| **DBMS** | MySQL |
| **IDEs/Editors** | Arduino IDE, Visual Studio Code, Intellij IDE |
| **UML tools** | Lucid Chart, Visual Paradigm, draw.io |

|  |  |
| --- | --- |
| **Version Control** | Source Tree |
| **Deployment server** | Google Cloud Platform, centOS 7 |
| **Project management tool** | Bitbucket |

* 1. Document Management

We use Bitbucket and Source Tree as our primary tool for sharing, editing and version control of our documents, along with Status Reports with-in each document.

It allows us to see what is changed in the documents and who is responsible for that change, reverse and simultaneously compare between versions of the document.

* 1. Source Code Management

For version control of our source code, we use Bitbucket. It tracks the changes team members make to files, so we have a record of what has been done, and we can revert to specific versions should we ever need to. Bitbucket also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

Bitbucket also features branches that provide an isolated environment for every change to our codebase. When a team member wants to start working on something—no matter how big or small— he creates a new branch. This ensures that the master branch always contains production-quality code.

# Software Requirement Specification

## Overall Description

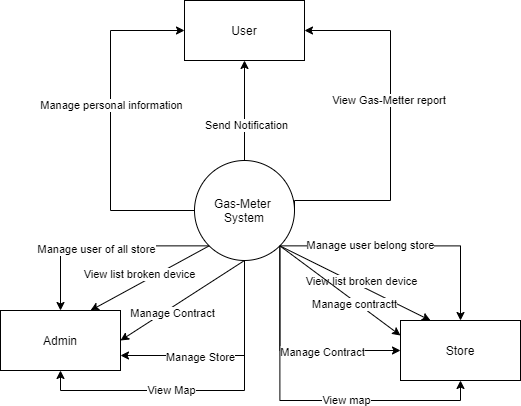
* 1. Product Overview

Figure : Functional overview of Gas-Metter

Gas-Meter is a system that helps users to monitor their home gas tank warning device. It also helps store and admin manage objects such as customers or contracts. The context diagram below illustrates the external entities and system interfaces for release 1.0. The system is expected to develop a number of other functions such as: chatting, sending store advertise to users, eventually becoming a gas management system and gas store service.

* 1. Business Rules

|  |  |
| --- | --- |
| **No** | **Description** |
| **B01** | Phone number cannot be empty |
| **B02** | Phone number must have from 10 digits. |
| **B03** | Email cannot be empty. |
| **B04** | Each email may be used for only one account. |
| **B05** | Password cannot be empty |
| **B06** | Password must contain at least 6 characters |
| **B06** | Admin can manage broken device, store and User. |
| **B07** | Store manage contract, broken device and user. |

## User Requirements

* 1. ***Use Case Diagram***

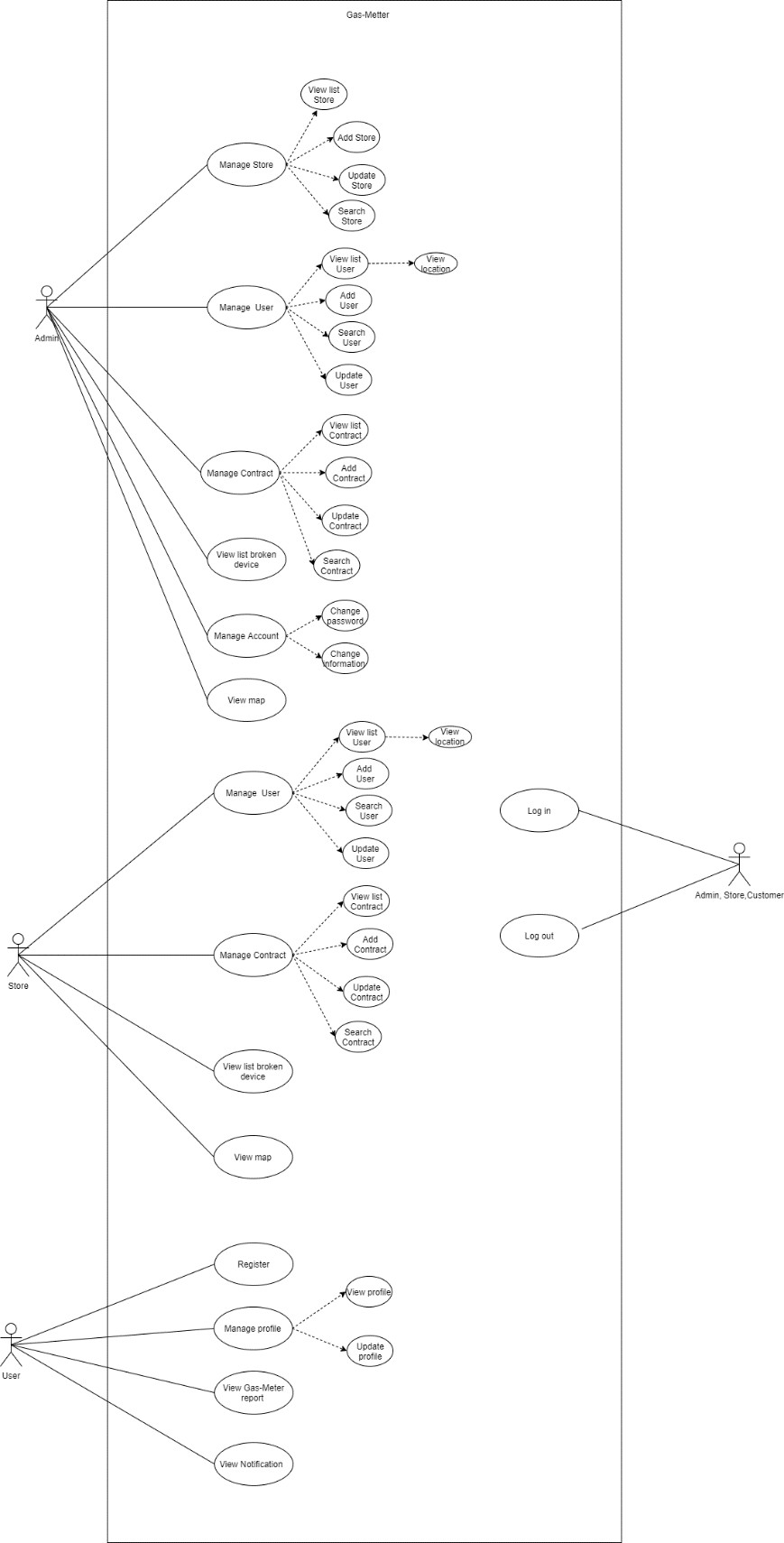


Figure 2.1: Use-case diagram

### Use-case list

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Actor** | **Function** | **Glossary** |
| UC-01 | User | Login | User Login into system. |
| UC-02 | User | Logout | User Logout system. |
| UC-03 | User | Register | User or gas store’s employee register account gas-meter. |
| UC-04 | User | View profile | User view his or her profile |
| UC-05 | User | Edit profile | User edit his or her profile |
| UC-06 | User | View gas-report | User can view gas-report include: gas concentration, device battery, weight gas  cylinder. |
| UC-07 | User | View notification | User can view notification |
| UC- 08 | Store | Login | Employee of store Login into system |
| UC-09 | Store | Logout | Employee of store Logout into system |
| UC-10 | Store | View list User | Store view list User belong store |
| UC-11 | Store | Create User | Store create new user in store |
| UC-12 | Store | Search User | Store can search User |
| UC-13 | Store | Update User | Store edit information of User |
| UC-14 | Store | View location of user | Store view location of user in map |
| UC-15 | Store | View list contract | Store can view list device in store. |
| UC-16 | Store | Add contract | Store create contract |
| UC-17 | Store | Edit contract | Store edit information of contract |
| UC-18 | Store | Search contract | Store search contract |
| UC- 19 | Store | View list broken device | View list broken device |
| UC-20 | Store | View map | Store view location of device belong of store  and status of them |
| UC-21 | Admin | Login | Admin Login in system |
| UC-22 | Admin | Logout | Admin Logout the system |
| UC-23 | Admin | View list store | Admin can view list store |
| UC-24 | Admin | Create Store | Admin can create store |
| UC-25 | Admin | Search Store | Admin search store |
| UC-26 | Admin | Update Store | Admin update store |

|  |  |  |  |
| --- | --- | --- | --- |
| UC-27 | Admin | View list User | Admin view list User in all store |
| UC-28 | Admin | Create User | Admin create new user. |
| UC-29 | Admin | Update User | Admin update one User |
| UC-30 | Admin | Search User | Admin update user. |
| UC-31 | Admin | View location user | View location of user in map |
| UC-32 | Admin | View list contract | Admin can view list all list contract for each  store. |
| UC-33 | Admin | Create a contract | Admin create a contract |
| UC-34 | Admin | Update a contract | Admin update a contract |
| UC-35 | Admin | Search contract | Admin search contract |
| UC-36 | Admin | View list broken- device | Admin view list broken device of all store |
| UC-37 | Admin | View map | Admin view location of all device and status of  them |
| UC-38 | Admin | Changed Password | Admin changed password. |
| UC39 | Admin | Changed information Admin | Admin change information. |

Table 2.2: Use-case list

2.2.1 User

* + - 1. ***Login***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-01 | | |
| **Use Case Name** | Login to System | | |
| **Creator** | ThanhTN | **Date Created** | 2020/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | User | | |
| **Description** | Login an account using email and password | | |
| **Pre-conditions** | * User can access the system. * User has already registered an account. * User is currently not Loged into the system. | | |
| **Post-conditions** | * User is Loged into the system. * User is redirected to Homepage. | | |

|  |  |
| --- | --- |
| **Normal Flow** | 1. From the app. 2. The system loads the Login page. 3. User enters “Email” and “Mật khẩu”. 4. User clicks "Đăng nhập" button. 5. The system checks your input data. If User entered true, User will be Login successful. 6. The system redirects User to the Home page. |
| **Alternative Flows** | N/A |
| **Exceptions** | *EXC1: At step 3 of normal flow, User leaves “Email” blank, then proceeds to step 4.*  The system displays the error message "Bạn phải nhâp email. ". User is not Loged in.  *EXC2: At step 3 of normal flow, User leaves “Mật khẩu” blank, then proceeds to step 4.*  The system displays the error message “Bạn phải nhập mật khẩu". User is not Loged in  *EXC3: At step 3 of normal flow, User enters invalid Login credentials, then proceeds to step 4.*  The system displays the error message "Số điện thoại hay mật khẩu không  chính xác. Hãy nhập lại". User is not Loged in. |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | Email cannot empty  Password cannot be empty |
| **Other Information** | N/A |

* + - 1. ***Logoutde***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-02 | | |
| **Use Case Name** | Logout | **Date Created** | 2020/10/10 |
| **Creator** | ThanhTN | **Last Updated** | 2019/10/10 |
| **Version** | 1.0 |  |  |
| **Actor** | User | | |
| **Description** | Logout of the account | | |

|  |  |
| --- | --- |
| **Pre-conditions** | * User accesses to the system. * User is currently Loged in. |
| **Post-conditions** | User is Loged out of the system. |
| **Normal Flow** | 1. In “Trang chủ Gas Meter” screen, select the menu with the three-tile icon in the upper right corner. 2. User clicks "Đăng xuất". 3. The system will redirect User to the Home page. |
| **Alternative Flows** | N/A |
| **Exceptions** | N/A. |
| **Priority** | Medium |
| **Frequency of Use** | Medium |
| **Business Rules** | N/A |

* + - 1. ***Register***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-03 | | |
| **Use Case Name** | Register | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | User | **Secondary Actors** | N/A |
| **Description** | Create a new account | | |
| **Pre-conditions** | * User can access the system * User is currently not Loged in. | | |
| **Post-conditions** | * The account is added to the system. * User is automatically Loged into the system. * User is redirected to Login page. | | |

|  |  |
| --- | --- |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Đăng ký". 2. The system will load the Register page. 3. User fills information into the required form. 4. User clicks "Đăng kí". |
| **Alternative Flows** | N/A |
| **Exceptions** | *EXC1: At step 3 of normal flow, User leaves “Họ và tên” blank, then proceeds to step 4.*  The system displays the error message "Hãy nhập họ và tên”. User is not registered  *EXC2: At step 3 of normal flow, User leaves email blank, then proceeds to step 4.*  The system displays the error message " Hãy nhập Email ". User is not registered  *EXC3: At step 3 of normal flow, User leaves “Số điện thoại” blank, then proceeds to step 4.*  The system displays the error message "Hãy nhập số điện thoại". User is not registered  *EXC4: At step 3 of normal flow, User leaves “Địa chỉ” blank, then proceeds to step 4.*  The system displays the error message "Hãy nhập địa chỉ”. User is not registered  *EXC5: At step 3 of normal flow, User leaves “Mật khẩu” blank, then proceeds to step 4.*  The system displays the error message "Hãy nhập mật khẩu”. User is not registered  *EXC6: At step 3 of normal flow, User enters an invalid “Số điện thoại”, then proceeds to step 4.*  The system displays the error message "Số điện thoại bị sai”. User is not registered.  *EXC7: At step 3 of normal flow, User enters a valid” Số điện thoại” that already exists in the system.*  The system displays the error message "Số điện thoại này đã tồn tại! Mời bạn  nhập số điện thoại khác". User is not registered. |

|  |  |
| --- | --- |
|  | *EXC8: At step 3 of normal flow, User enters a “Mật khẩu” that has below 8 characters.*  The system displays the error message " Mật khẩu phải có ít nhất 8 ký tự". User is not registered.  . |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***View profile***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-04 | | |
| Use Case Name | View Profile | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | User |  |  |
| Description | View personal profile | | |
| Pre-conditions | * User accesses the system. * User is currently Loged in. | | |
| Post-conditions | System displays the User profile page. | | |
| Normal Flow | 1. In “Gas Meter homepage” screen, select the menu with the three-tile icon in the upper right corner. 2. User click”Cá nhân”. 3. The system will redirect User to the profile page. | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |

* + - 1. ***Update profile***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-05 | | |
| **Use Case Name** | Updated Profile | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | User |  |  |
| **Description** | User update personal information | | |
| **Pre-conditions** | Client is on the app mobile Client is Loging in  Client is on their profile page | | |
| **Post-conditions** | The User change their information successfully | | |
| **Normal Flow** | 1. In “Gas Meter homepage” screen, select the menu with the three-tile icon in the upper right corner. 2. User click”Cá nhân”. 3. The system will redirect User to the profile page. 4. User change their profile. 5. The system will redirect User to the Profile page | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | *EXC1: At step 3 of normal flow, User leaves full name blank, then proceeds to step 4.*  The system displays the error message "Họ và tên không được để trống ". User is not changed profile.  *EXC2: At step 3 of normal flow, User leaves email, then proceeds to step 4.*  The system displays the error message "Email’ không được để trống ". User is not changed profile.  *EXC3: At step 3 of normal flow, User leaves “Địa chỉ” blank, then proceeds to step 4.*  The system displays the error message "Địa chỉ không được để trống". User is  not changed profile. | | |

|  |  |
| --- | --- |
|  | *EXC4: At step 3 of normal flow, User leaves “Mật Khẩu” blank, then proceeds to step 4.*  The system displays the error message "Mật khẩu không được để trống ". User  is not changed profile. |
| **Priority** | Medium |
| **Frequency of Use** | Low |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***View Gas-Meter report***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-06 | | |
| **Use Case Name** | View Gas-report | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | User |  |  |
| **Description** | User view gas-report. | | |
| **Pre-conditions** | User is currently Loged in. | | |
| **Post-conditions** | The User view gas-report include: gas concentration, device battery, weight gas cylinder. | | |
| **Normal Flow** | 1. After user Login, the system redirect to gas-report page. | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | *N/A* | | |
| **Priority** | Medium | | |
| **Frequency of Use** | Low | | |
| **Business Rules** | N/A | | |
| **Other Information** | N/A | | |

2.2.1.7 View Notification

|  |  |
| --- | --- |
| **Use Case ID** | UC-07 |
| **Use Case Name** | View Notification |

|  |  |  |  |
| --- | --- | --- | --- |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | User |  |  |
| **Description** | User view notification. | | |
| **Pre-conditions** | User is currently Loged in. | | |
| **Post-conditions** | The User view notification from store. | | |
| **Normal Flow** | 1. In “Gas Meter homepage” screen, select the menu with the three-tile icon in the upper right corner. 2. User click”Thông báo”. 3. System display notification screen. | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | *N/A* | | |
| **Priority** | Medium | | |
| **Frequency of Use** | Low | | |
| **Business Rules** | N/A | | |
| **Other Information** | N/A | | |

* + 1. Store
       1. ***Store Login***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-08 | | |
| **Use Case Name** | Login to System | | |
| **Creator** | ThanhTN | **Date Created** | 2020/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | Store | | |
| **Description** | Login an account using email and password | | |
| **Pre-conditions** | * Store can access the system. * Store has already registered an account. * Store is currently not Loged into the system. | | |
| **Post-conditions** | * Store is Loged into the system. * Store is redirected to Homepage. | | |

|  |  |
| --- | --- |
| **Normal Flow** | 1. Store access website management. 2. The system loads the Login page. 3. Store enters “Email” and “Mật khẩu”. 4. Store clicks "Đăng nhập" button. 5. The system checks your input data. If User entered true, User will be Login successful. 6. The system redirects Store to the Home page. |
| **Alternative Flows** | N/A |
| **Exceptions** | *EXC1: At step 3 of normal flow,* Store *leaves “Email” blank, then proceeds to step 4.*  The system displays the error message "Bạn phải nhâp Email". Store is not Loged in.  *EXC2: At step 3 of normal flow,* Store *leaves “Mật khẩu”blank, then proceeds to step 4.*  The system displays the error message “Bạn phải nhập mật khẩu". Store is not Loged in  *EXC3: At step 3 of normal flow,* Store *enters invalid Login credentials, then proceeds to step 4.*  The system displays the error message "Email hoặc mật khẩu không chính xác.  Hãy nhập lại". Store is not Loged in. |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | Store name cannot be empty  Password cannot be empty |
| **Other Information** | N/A |

* + - 1. ***Store Logout***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-09 | | |
| **Use Case Name** | Logout | **Date Created** | 2020/10/10 |
| **Creator** | ThanhTN | **Last Updated** | 2019/10/10 |
| **Version** | 1.0 |  |  |
| **Actor** | Store | | |
| **Description** | Logout of the account | | |

|  |  |
| --- | --- |
| **Pre-conditions** | * Store accesses to the system. * Store is currently Loged in. |
| **Post-conditions** | Store is Loged out of the system. |
| **Normal Flow** | 1. From the website header, Store moves cursor above the top right corner, with the store's personal icon. 2. Store clicks "Đăng xuất". 3. The system will redirect User to the Home page. |
| **Alternative Flows** | N/A |
| **Exceptions** | N/A. |
| **Priority** | Medium |
| **Frequency of Use** | Medium |
| **Business Rules** | N/A |

2.2.2.3 View list user belong store

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-10 | | |
| Use Case Name | View List User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | View list User in store | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -User belong store. | | |
| Post-conditions | System displays list User belong store. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách khách hàng” button. 2. The system will redirect store list User page. | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |

|  |  |
| --- | --- |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Add user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-11 | | |
| Use Case Name | Add User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Add new user in store | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -User belong store. | | |
| Post-conditions | * The User is added to the system. * User is automatically Loged into the system. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách khách hàng” button. 2. The system will redirect store list user page. 3. Store click”Thêm mới” icon in the header. 4. The system redirect register page. 5. Store input data in page. 6. Store click”Thêm mới” in register page. 7. Store input data User. 8. Store click “Đồng ý”. 9. System send redirect to list User page. | | |
| Alternative Flows | N/A | | |
| Exceptions | *EXC1: At step 5 of normal flow, User leaves “Họ và tên” blank, then proceeds to step 6.*  The system displays the error message "Hãy nhập họ và tên”. User is not registered  *EXC2: At step 5 of normal flow, User leaves email blank, then proceeds to step 6.*  The system displays the error message " Hãy nhập Email ". User is not | | |

|  |  |
| --- | --- |
|  | registered  *EXC3: At step 5 of normal flow, User leaves “Số điện thoại” blank, then proceeds to step 6.*  The system displays the error message "Hãy nhập số điện thoại". User is not registered  *EXC4: At step 5 of normal flow, User leaves “Địa chỉ” blank, then proceeds to step 6.*  The system displays the error message "Hãy nhập địa chỉ”. User is not registered  *EXC5: At step 5 of normal flow, User leaves “Mật khẩu” blank, then proceeds to step 6.*  The system displays the error message "Hãy nhập mật khẩu”. User is not registered  *EXC6: At step 5 of normal flow, User enters an invalid “Số điện thoại”, then proceeds to step 6.*  The system displays the error message "Số điện thoại bị sai”. User is not registered.  *EXC7: At step 5 of normal flow, User enters a valid” Số điện thoại” that already exists in the system.*  The system displays the error message "Số điện thoại này đã tồn tại! Mời bạn nhập số điện thoại khác". User is not registered.  *EXC8: At step 5 of normal flow, User enters a “Mật khẩu” that has below 8 characters.*  The system displays the error message " Mật khẩu phải có ít nhất 8 ký tự". User is not registered. |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Update user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-12 | | |
| **Use Case Name** | Update User | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Update information User complete. | | |
| **Pre-conditions** | * Store can access the system * Store is currently not Loged in. | | |
| **Post-conditions** | * The information of User is update to the system. * System redirect to list User page | | |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Danh sách khách hàng". 2. The system will load the List User page. 3. Store click the User want to update 4. Store click “Update” icon 5.System load update page 6.Store input information 5. After Store input data, Store click “Thay đổi” 6. System display messenger ”Bạn có muốn update thông tin” 9.System redirect to list User page. | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | *EXC1: At step 7 of normal flow, store leaves “Họ và tên” blank, then proceeds to step 8.*  The system displays the error message "Hãy nhập họ và tên”. User is not registered  *EXC2: At step 7 of normal flow, store leaves email blank, then proceeds to step 8.*  The system displays the error message " Hãy nhập Email ". User is not registered  *EXC3: At step 7 of normal flow, store leaves “Số điện thoại” blank, then proceeds to step 8.*  The system displays the error message "Hãy nhập số điện thoại". User is not registered  *EXC4: At step 7 of normal flow, store leaves “Địa chỉ” blank, then proceeds to step 8.*  The system displays the error message "Hãy nhập địa chỉ”. User is not registered  *EXC5: At step 8 of normal flow, User click “Đồng ý”, then proceeds to step 9.* | | |

|  |  |
| --- | --- |
|  | The system is changed. User is changed. System display new list User  *EXC6: At step 8 of normal flow, User click “Từ Chối”, then proceeds to step 9.*  *The system is not changed. The system redirect in list User page* |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***Search user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-13 | | |
| Use Case Name | Search User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Search User in store by name. | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -User belong store. | | |
| Post-conditions | System displays list User belong store after search. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách khách hàng” button. 2. The system will redirect list User page of store. 3. Store click”Tìm kiếm” field in the header.   5. Store input name in search field. 6.System display result after search. | | |
| Alternative Flows | N/A | | |
| Exceptions | *Exc1: Store leave blank “Tìm kiếm” field, then click search Expected: The system display all list User.*  *Exc2: Store input special character, html, javascript…*  *Expected: The system display alert: “Bạn nhập sai”* | | |
| Priority | Medium | | |

|  |  |
| --- | --- |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***View location of user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-14 | | |
| Use Case Name | View location of user | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Display location device on map. | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -User belong store. | | |
| Post-conditions | System displays location of user on map. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách khách hàng” button. 2. The system will redirect list User page of store. 3. Store find user want to view location.   5. Store click “Vị trí” button of user . 6.System display location of user. | | |
| Alternative Flows | N/A | | |
| Exceptions | *N/A* | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |

* + - 1. ***View list contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-15 | | |
| Use Case Name | View List Contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |

|  |  |  |  |
| --- | --- | --- | --- |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | View list Contract belong store | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -Contract belong store. | | |
| Post-conditions | System displays list device belong store. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách hợp đồng” button. 2. The system will redirect store list contract page. | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |

* + - 1. ***Add new contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-16 | | |
| Use Case Name | Add Contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Store add contract to system | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in. | | |
| Post-conditions | Add new contract in system. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách hợp đồng ” button. 2. The system will redirect store list contract page. 3. Store click”Thêm mới” icon in the header. 4. The system redirect register page. | | |

|  |  |
| --- | --- |
|  | 1. Store input data in page. 2. Store click”Thêm mới” in register page. 3. System display message”Bạn có muốn thêm hợp đồng này”. 4. Store click “Đồng ý”. 5. System send redirect to list contract page. |
| Alternative Flows | N/A |
| Exceptions | *N/A* |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

2.2.2.10 Update contract

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-17 | | |
| **Use Case Name** | Update contract | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Update information Device complete. | | |
| **Pre-conditions** | * Store can access the system * Store is currently not Loged in. | | |
| **Post-conditions** | * The information of contract is update to the system. * System redirect to list contract page | | |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Danh sách hơp đồng". 2. The system will load the List contract page. 3. Store click the contract want to update 4. Store click “Sửa” button. 5.System load update page 6.Store input information 5. After Store input data, Store click “Thay đổi” 6. System display messenger ”Bạn có muốn update thông tin” 9.System redirect to list contracts page. | | |

|  |  |
| --- | --- |
| **Alternative Flows** | N/A |
| **Exceptions** | *N/A* |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***Search contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-18 | | |
| Use Case Name | Search contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Search contract. | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in. | | |
| Post-conditions | System displays list contract belong store after search. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách hợp đồng” button. 2. The system will redirect list contract page of store. 3. Store click”Tìm kiếm” field.   5. Store input data in search field. 6.System display result after search. | | |
| Alternative Flows | N/A | | |
| Exceptions | *Exc1: Store leave blank “Tìm kiếm” field, then click search Expected: The system display all list contract.*  *Exc2: Store input special character, html, javascript…*  *Expected: The system display alert: “Bạn nhập sai”* | | |

|  |  |
| --- | --- |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***View list broken device***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-19 | | |
| **Use Case Name** | View list broken device | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Display list broken device belong store | | |
| **Pre-conditions** | * Store can access the system * Store is currently not Loged in. | | |
| **Post-conditions** | - Display list broken device of store | | |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Danh sách thiết bị hỏng". 2. The system will load the list broken device page. | | |
| **Alternative Flows** | N/A | | |

* + - 1. ***View map***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-20 | | |
| **Use Case Name** | View map | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Display location of device in the map and status of them. | | |
| **Pre-conditions** | - Store can access the system | | |

|  |  |
| --- | --- |
|  | - Store is currently not Loged in. |
| **Post-conditions** | - Display list broken device of store |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Bản đồ". 2. The system will load the location of device one map. |
| **Alternative Flows** | N/A |

* + 1. Admin
       1. ***Login***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-21 | | |
| **Use Case Name** | Admin to System | | |
| **Creator** | ThanhTN | **Date Created** | 2020/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Actor** | Admin | | |
| **Description** | Login an account | | |
| **Pre-conditions** |  | | |
| **Post-conditions** | * Admin is Loged into the system. * Admin is redirected to Homepage. | | |
| **Normal Flow** | 1. From the website , Admin go to Login website. 2. The system loads the Login page. 3. Admin enters “Tên đăng nhập” and “Mật khẩu”. 4. Admin clicks "Đăng nhập" button. 5. The system checks your input data. If admin entered true, admin will be Login successful. 6. The system redirects admin to the Home page. | | |
| **Alternative Flows** | N/A | | |

|  |  |
| --- | --- |
| **Exceptions** | *N/A* |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** |  |
| **Other Information** | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| ***2.1.3.2 Logout*** | | | |
| **Use Case ID** | UC-22 | | |
| **Use Case Name** | Logout | **Date Created** | 2020/10/10 |
| **Creator** | ThanhTN | **Last Updated** | 2019/10/10 |
| **Version** | 1.0 |  |  |
| **Actor** | Admin | | |
| **Description** | Logout of the account | | |
| **Pre-conditions** | * Admin accesses to the system. * Admin is currently Loged in. | | |
| **Post-conditions** | Admin is Loged out of the system. | | |
| **Normal Flow** | 1. From the website header, User moves cursor above the top right corner, with the Admin's personal icon. 2. Admin clicks "Đăng xuất". 3. The system will redirect Admin to the Home page. | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | N/A. | | |
| **Priority** | Medium | | |
| **Frequency of Use** | Medium | | |
| **Business Rules** | N/A | | |
| ***2.1.3.3 View list store*** | | | |

|  |  |
| --- | --- |
| **Use Case ID** | UC-23 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | View list store | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | View list | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | System displays list store. | | |
| Normal Flow | 1. From the web header, Admin click”Danh sách Cửa Hàng” button. 2. The system will redirect store list store page. | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |

* + - 1. ***Add store***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-24 | | |
| Use Case Name | Add User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Create new store | | |
| Pre-conditions |  | | |
| Post-conditions | * The Store is added to the system. * Store is automatically Loged into the system. | | |

|  |  |
| --- | --- |
| Normal Flow | 1. From the web header, Admin click”Danh sách Cửa hàng” button. 2. The system will redirect store list store page. 3. Admin click ”Thêm mới” icon . 4. The system redirect register store page. 5. Admin input data in page. 6. Admin click ”Thêm mới” in register store page. 7. Admin input data store. 8. Admin click “Đồng ý”. 9. System send redirect to list store page. |
| Alternative Flows | N/A |
| Exceptions | *N/A* |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Update store***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-25 | | |
| **Use Case Name** | Update User | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Update information User complete. | | |
| **Pre-conditions** | - Admin can access the system | | |
| **Post-conditions** | * The information of store is update to the system. * System redirect to list store page | | |

|  |  |
| --- | --- |
| **Normal Flow** | 1. From the homepage, the Admin clicks on button "Danh sách Cửa hàng". 2. The system will load the List Store page. 3. Admin click the store want to update 4. Admin click “Update” icon 5.System load update page 6.Admin input information 5. After Admin input data, Admin click “Thay đổi” 6. System display messenger “Bạn có muốn update thông tin” 9.System redirect to list store page. |
| **Alternative Flows** | N/A |
| **Exceptions** | *N/A* |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***Search store***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-26 | | |
| Use Case Name | Search Store | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | Search Store. | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | System displays list store. | | |

|  |  |
| --- | --- |
| Normal Flow | 1. From the web header, Admin click”Danh sách cửa hàng” button. 2. The system will redirect store list store page. 3. Admin click”Tìm kiếm” icon .   5. Admin input name in search field. 6.System display result after search. |
| Alternative Flows | N/A |
| Exceptions | *Exc1: Admin leave blank “Tìm kiếm” field, then click search Expected: The system display all list User.*  *Exc2: Admin input special character, html, javascript…*  *Expected: The system display alert: “Bạn nhập sai”* |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***View list user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-27 | | |
| Use Case Name | View List User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | View list User | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | System displays list User. | | |
| Normal Flow | 1. From the web header, Admin click”Danh sách khách hàng” button. 2. The system will redirect list user page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to view customer. 4. System display list customer belong to store. | | |
| Alternative Flows | N/A | | |
| Exceptions | N/A | | |

|  |  |
| --- | --- |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Add user***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-28 | | |
| Use Case Name | Add User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Create new user. | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | * Admin is added to the system. * Store is automatically Loged into the system. | | |
| Normal Flow | 1. From the web header, Admin click”Danh sách khách hàng” button. 2. The system will redirect list user page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to create user. 4. System display list customer belong to store. 5. Admin click “Thêm mới ”icon. 6.System display register user page. 6. Admin input data in system. 7. After input data, Admin click “Thêm” button. System display message”Bạn có muốn them người dùng này” 8. Admin click “Đồng ý”. System display new list user belong store. New user   add to system. | | |
| Alternative Flows | N/A | | |
| Exceptions | *N/A* | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |

|  |  |
| --- | --- |
| Other Information | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| ***2.1.3.9 Search user*** | | | |
| **Use Case ID** | UC-29 | | |
| Use Case Name | Search User | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Search User. | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | System displays list User after search. | | |
| Normal Flow | 1. From the web header, Admin click”Danh sách khách hàng” button. 2. The system will redirect list User page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to search user. 4. System display list user belong store which admin choose. 5. Admin input data in search field. 6.System display result after search. | | |
| Alternative Flows | N/A | | |
| Exceptions | *Exc1: Admin leave blank “Tìm kiếm” field, then click search Expected: The system display all list User.*  *Exc2: Admin input special character, html, javascript…*  *Expected: The system display alert: “Bạn nhập sai”* | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |
| ***2.1.3.10Update user*** | | | |

|  |  |
| --- | --- |
| **Use Case ID** | UC-30 |
| **Use Case Name** | Update User |

|  |  |  |  |
| --- | --- | --- | --- |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Update information User complete. | | |
| **Pre-conditions** | * Admin can access the system * Admin is currently not Loged in. | | |
| **Post-conditions** | * The information of User is update to the system. * System redirect to list User page | | |
| **Normal Flow** | 1. From the homepage, Admin clicks on button "Khách hàng". 2. The system will load the List User page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to update user. 4. Admin click the User want to update 5.Admin click “Update” icon   6.System load update page 7.Admin input information   1. After Store input data, Admin click “Thay đổi” 2. User is update. System redirect to list User page. | | |
| **Alternative Flows** | N/A | | |
| **Exceptions** | *N/A* | | |
| **Priority** | High | | |
| **Frequency of Use** | High | | |
| **Business Rules** | N/A | | |
| **Other Information** | N/A | | |

* + - 1. ***View location of user***

|  |  |
| --- | --- |
| **Use Case ID** | UC-31 |
| Use Case Name | View location of user |

|  |  |  |  |
| --- | --- | --- | --- |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Display location of device on map and status of device. | | |
| Pre-conditions | * Store accesses the system. * Store is currently Loged in.   -User belong store. | | |
| Post-conditions | System displays location device on map. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách khách hàng” button. 2. The system will redirect list User page of store. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to view user. System display list user belong that store. 4. Admin find user want to view location. 5. Admin click “Vị trí ” button of user . 6.System display location of user on map. | | |
| Alternative Flows | N/A | | |
| Exceptions | *N/A* | | |
| Priority | Medium | | |
| Frequency of Use | High | | |
| Business Rules | N/A | | |
| Other Information | N/A | | |

* + - 1. ***View list contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-32 | | |
| Use Case Name | View List Contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Store |  |  |
| Description | View list Contract each store | | |
| Pre-conditions | * Contract accesses the system. * Admin is currently Loged in. | | |

|  |  |
| --- | --- |
|  |  |
| Post-conditions | System displays list device belong store. |
| Normal Flow | 1. From the web header, Store click”Danh sách hợp đồng” button. 2. The system will redirect store list contract page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to view contract. 4. System display list contract belong that store which user choose. |
| Alternative Flows | N/A |
| Exceptions | N/A |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Add contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-33 | | |
| Use Case Name | Add Contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Admin add contract to system | | |
| Pre-conditions | - Admin is currently Loged in. | | |
| Post-conditions | Add new contract in system. | | |
| Normal Flow | 1. From the web header, Store click”Danh sách hợp đồng ” button. 2. The system will redirect store list contract page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to add new contract. System display list contract belong store what admin choose. 4. Admin click”Thêm mới” icon . | | |

|  |  |
| --- | --- |
|  | 1. The system display register page. 2. Admin input data in page. 3. Admin click”Thêm mới” in register page. 4. System display message”Bạn có muốn thêm hợp đồng này”. 5. Admin click “Đồng ý”. 6. System send redirect to list contract page of store what admin choose. |
| Alternative Flows | N/A |
| Exceptions | *N/A* |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***Update contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-34 | | |
| **Use Case Name** | Update contract | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Admin | **Secondary Actors** | N/A |
| **Description** | Update information Device complete. | | |
| **Pre-conditions** | * Store can access the system * Store is currently not Loged in. | | |
| **Post-conditions** | * The information of contract is update to the system. * System redirect to list contract page | | |
| **Normal Flow** | 1. From the homepage, the Admin clicks on button "Danh sách hơp đồng". 2. The system will load the List contract page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to add update contract. System display list contract belong store what admin choose. 4. Admin find the contract want to update 5. Admin click “Sửa” icon 6.System load update page | | |

|  |  |
| --- | --- |
|  | 1. Admin input information 2. After Admin input data, Store click “Thay đổi” 3. System display messenger ”Bạn có muốn update thông tin” 4. System redirect to list contracts page of store what admin choose. User update success. |
| **Alternative Flows** | N/A |
| **Exceptions** | *N/A* |
| **Priority** | High |
| **Frequency of Use** | High |
| **Business Rules** | N/A |
| **Other Information** | N/A |

* + - 1. ***Search contract***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-35 | | |
| Use Case Name | Search contract | | |
| Creator | ThanhTN | **Date Created** | 2019/10/10 |
| Version | 1.0 | **Last Updated** | 2019/10/10 |
| Actor | Admin |  |  |
| Description | Search contract. | | |
| Pre-conditions | * Admin accesses the system. * Admin is currently Loged in. | | |
| Post-conditions | System displays list contract belong store after search. | | |

|  |  |
| --- | --- |
| Normal Flow | 1. From the web header, Amin click”Danh sách hợp đồng” button. 2. The system will redirect list contract page of store. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to add update contract. System display list contract belong store what admin choose. 4. Admin click”Tìm kiếm” field. 5. Admin input data in search field. 6.System display result after search. |
| Alternative Flows | N/A |
| Exceptions | *Exc1: Store leave blank “Tìm kiếm” field, then click search Expected: The system display all list contract.*  *Exc2: Store input special character, html, javascript…*  *Expected: The system display alert: “Bạn nhập sai”* |
| Priority | Medium |
| Frequency of Use | High |
| Business Rules | N/A |
| Other Information | N/A |

* + - 1. ***View list broken device***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-36 | | |
| **Use Case Name** | View list broken device | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Display list broken device belong store | | |
| **Pre-conditions** | * Admin can access the system * Admin is currently not Loged in. | | |
| **Post-conditions** | - Display list broken device of store | | |

|  |  |
| --- | --- |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Danh sách thiết bị hỏng". 2. The system will load the list broken device page. 3. On the left side of the website have a slide bar which display list store. Admin click “store” want to view list broken device. 4. System display list broken device belong store which Admin ch |
| **Alternative Flows** | N/A |

* + - 1. ***View Map***

|  |  |  |  |
| --- | --- | --- | --- |
| **3 Use Case ID** | UC-37 | | |
| **Use Case Name** | View map | | |
| **Creator** | ThanhTN | **Date Created** | 2019/10/10 |
| **Version** | 1.0 | **Last Updated** | 2019/10/10 |
| **Primary Actor** | Store | **Secondary Actors** | N/A |
| **Description** | Display location of device in the map and status of them. | | |
| **Pre-conditions** | * Store can access the system * Store is currently not Loged in. | | |
| **Post-conditions** | - Display list broken device of store | | |
| **Normal Flow** | 1. From the homepage, the User clicks on button "Bản đồ". 2. The system will load the location of device and store one map. Display information of them if admin hover it. | | |
| **Alternative Flows** | N/A | | |

* + - 1. ***Changed password***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-38 | | |
| **Use Case Name** | Logout | **Date Created** | 2020/10/10 |
| **Creator** | ThanhTN | **Last Updated** | 2019/10/10 |
| **Version** | 1.0 |  |  |

|  |  |
| --- | --- |
| **Actor** | Admin |
| **Description** | Logout of the account |
| **Pre-conditions** | * Admin accesses to the system. * Admin is currently Loged in. |
| **Post-conditions** | Admin is Loged out of the system. |
| **Normal Flow** | 1. From the website header, User moves cursor above the top right corner, with the Admin's personal icon. 2. Admin clicks "Đổi mật khẩu". 3. The system will dislay change password page. 4. Admin change password. 5. System send redirect to homepage. Password changed complete. |
| **Alternative Flows** | N/A |
| **Exceptions** | N/A. |
| **Priority** | Medium |
| **Frequency of Use** | Medium |
| **Business Rules** | N/A |

* + - 1. ***Changed Admin information***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-39 | | |
| **Use Case Name** | Logout | **Date Created** | 2020/10/10 |
| **Creator** | ThanhTN | **Last Updated** | 2019/10/10 |
| **Version** | 1.0 |  |  |
| **Actor** | Admin | | |
| **Description** | Logout of the account | | |
| **Pre-conditions** | * Admin accesses to the system. * Admin is currently Loged in. | | |

|  |  |
| --- | --- |
| **Post-conditions** | Admin is Loged out of the system. |
| **Normal Flow** | 1. From the website header, User moves cursor above the top right corner, with the Admin's personal icon. 2. Admin clicks "Thay đổi thông tin". 3. The system will dislay change password page. 4. Admin change information . 5. System send redirect to homepage. Information changed complete. |
| **Alternative Flows** | N/A |
| **Exceptions** | N/A. |
| **Priority** | Medium |
| **Frequency of Use** | Medium |
| **Business Rules** | N/A |

## Functional Requirements

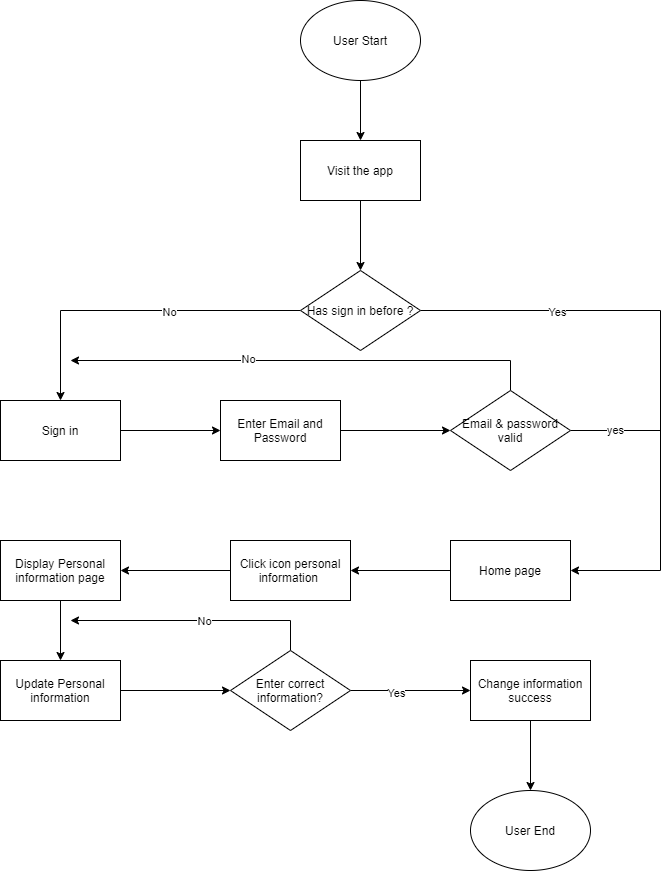
* 1. System Functional Overview

###### Screen Authorization

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **User** | **Store** | **Admin** |
| Register | ✓ |  |  |
| View User information | ✓ | ✓ | ✓ |
| Update user information | ✓ | ✓ | ✓ |
| View Gas-meter report | ✓ | ✓ | ✓ |

|  |  |  |  |
| --- | --- | --- | --- |
| View notification | ✓ |  |  |
| View list user |  | ✓ | ✓ |
| Add new user |  | ✓ | ✓ |
| Search user |  | ✓ | ✓ |
| View location of user |  | ✓ | ✓ |
| View list store |  |  | ✓ |
| Add new store |  |  | ✓ |
| Update Store |  |  | ✓ |
| Search Store |  |  | ✓ |
| View Contract |  | ✓ | ✓ |
| Create Contract |  | ✓ | ✓ |
| Update Contract |  | ✓ | ✓ |
| Search Contract |  | ✓ | ✓ |
| View map |  | ✓ | ✓ |
| View list broken device |  | ✓ | ✓ |
| Change Password |  |  | ✓ |

* 1. System Flowchart
     1. Update profile



Firgure 3.2.1 : Update profile flowchart

* + 1. View Gas-Meter report

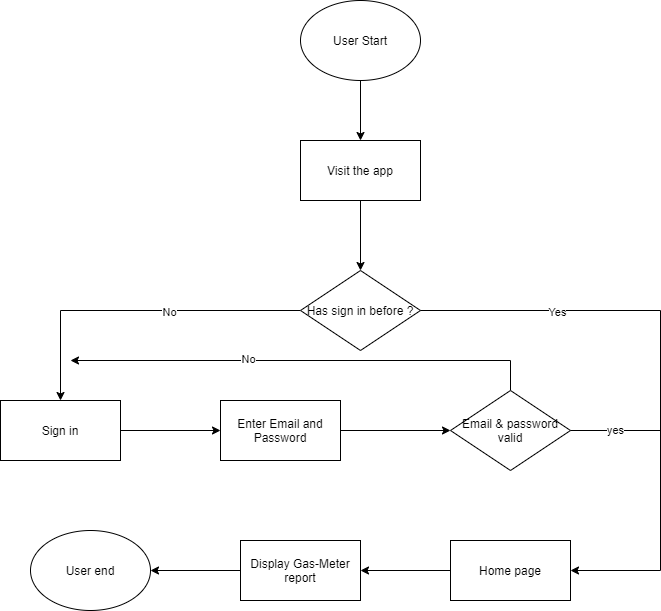


Figure 3.2.2: View Gas-Meter report flow chart

* + 1. View notification

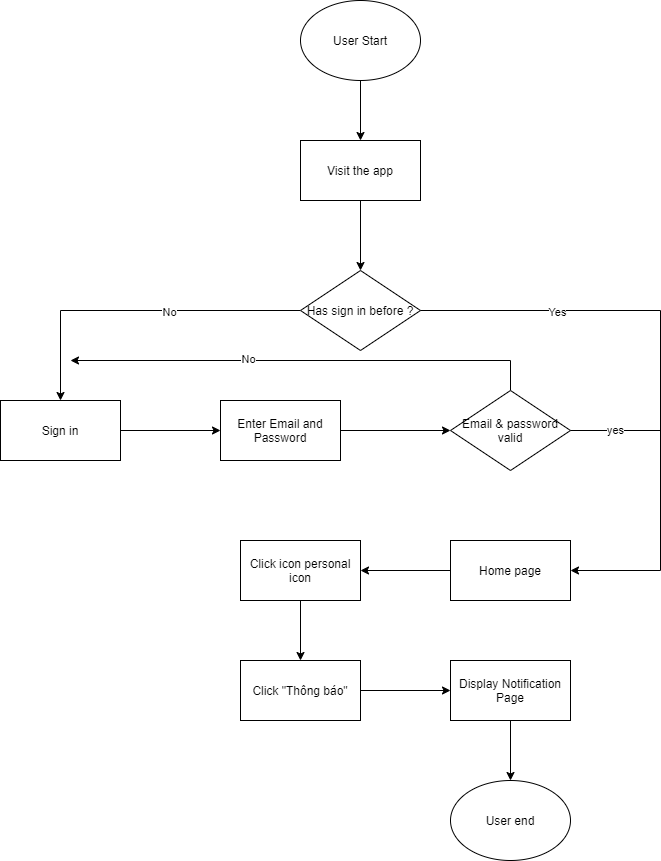


Figure 3.2.3 : View Notification Flow chart

* + 1. Manage User

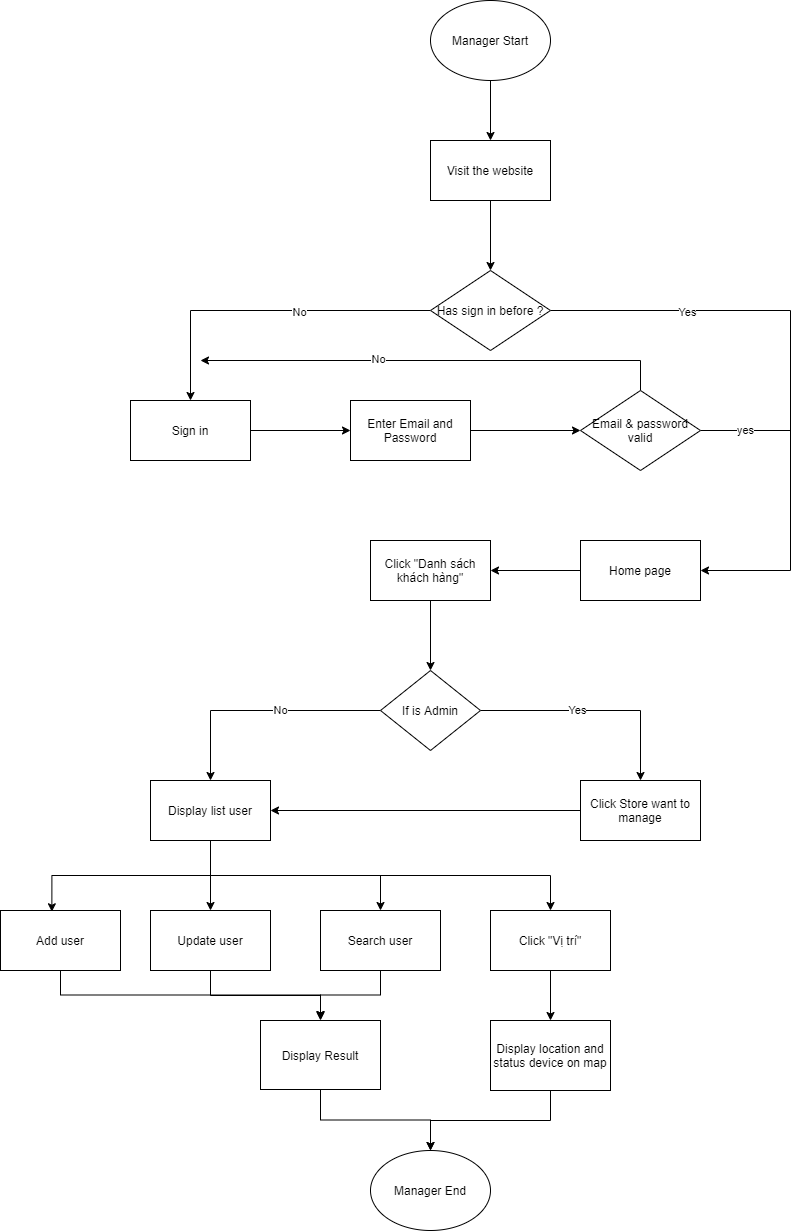


Figure 3.2.4: Manage user flow chart

* + 1. Manage Contract

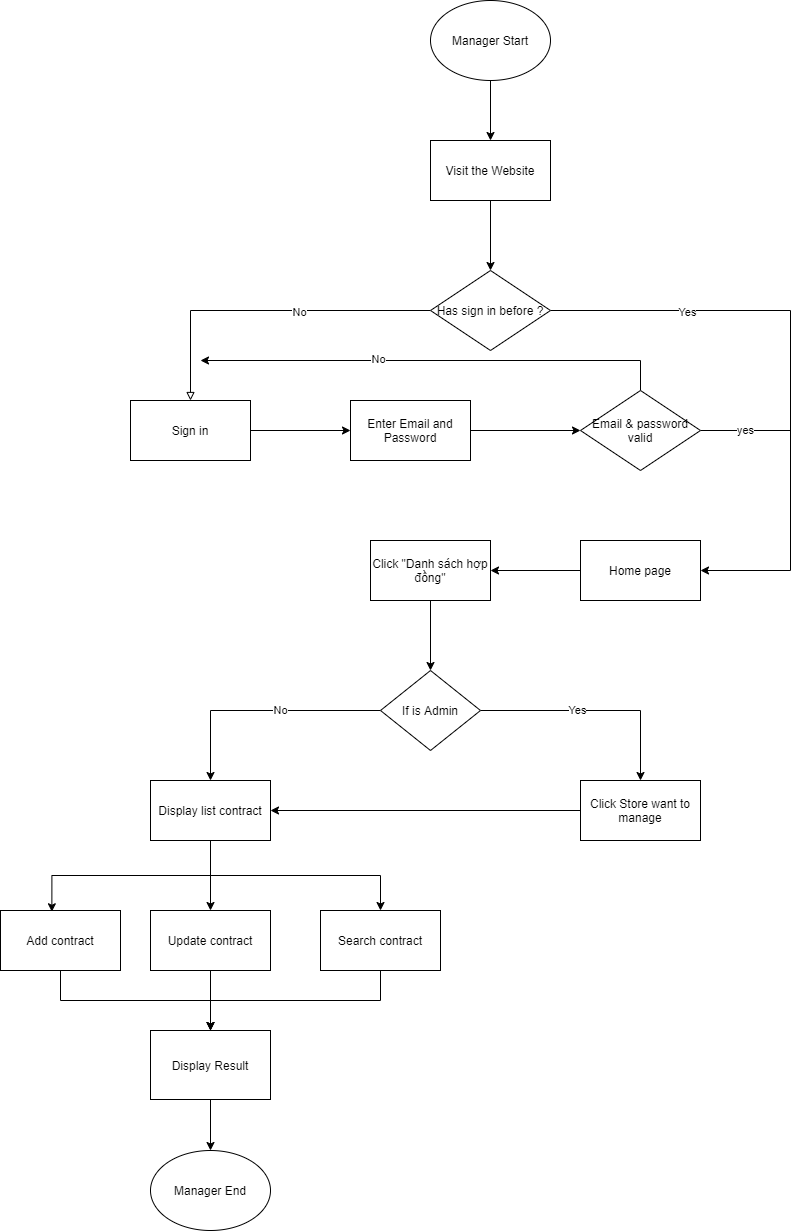


Figure 3.2.5: Manage contract flow chart

* + 1. View list broken device

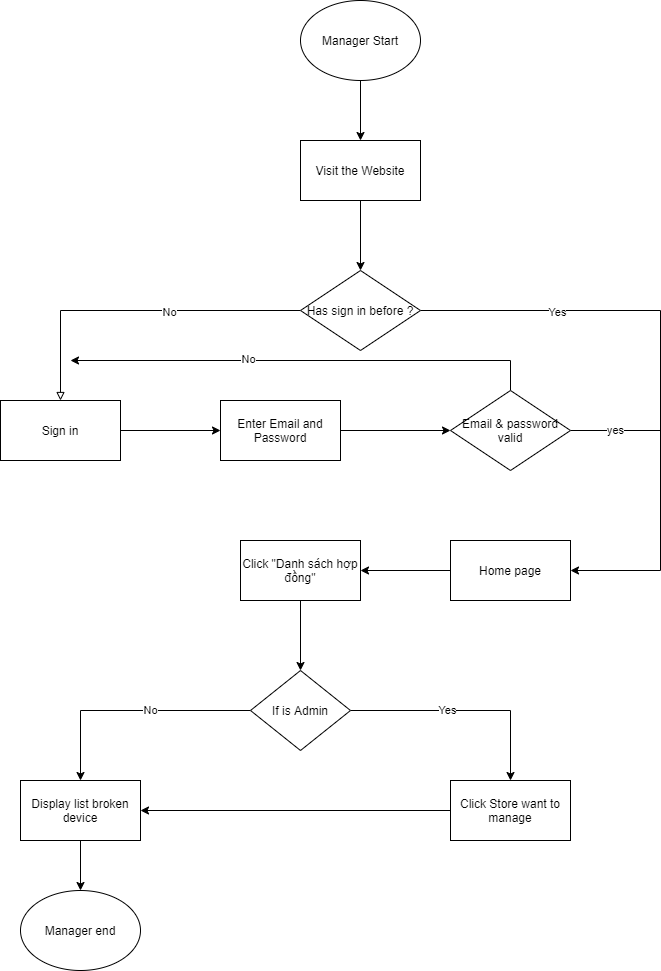


Figure 3.2.6 : View list broken device flow chart

* + 1. View map



Figure3.2.7 : View map flow chart.

* + 1. Change Admin password



Figure 3.2.8 : Change Admin flow chart

## Entity Relationship Diagram

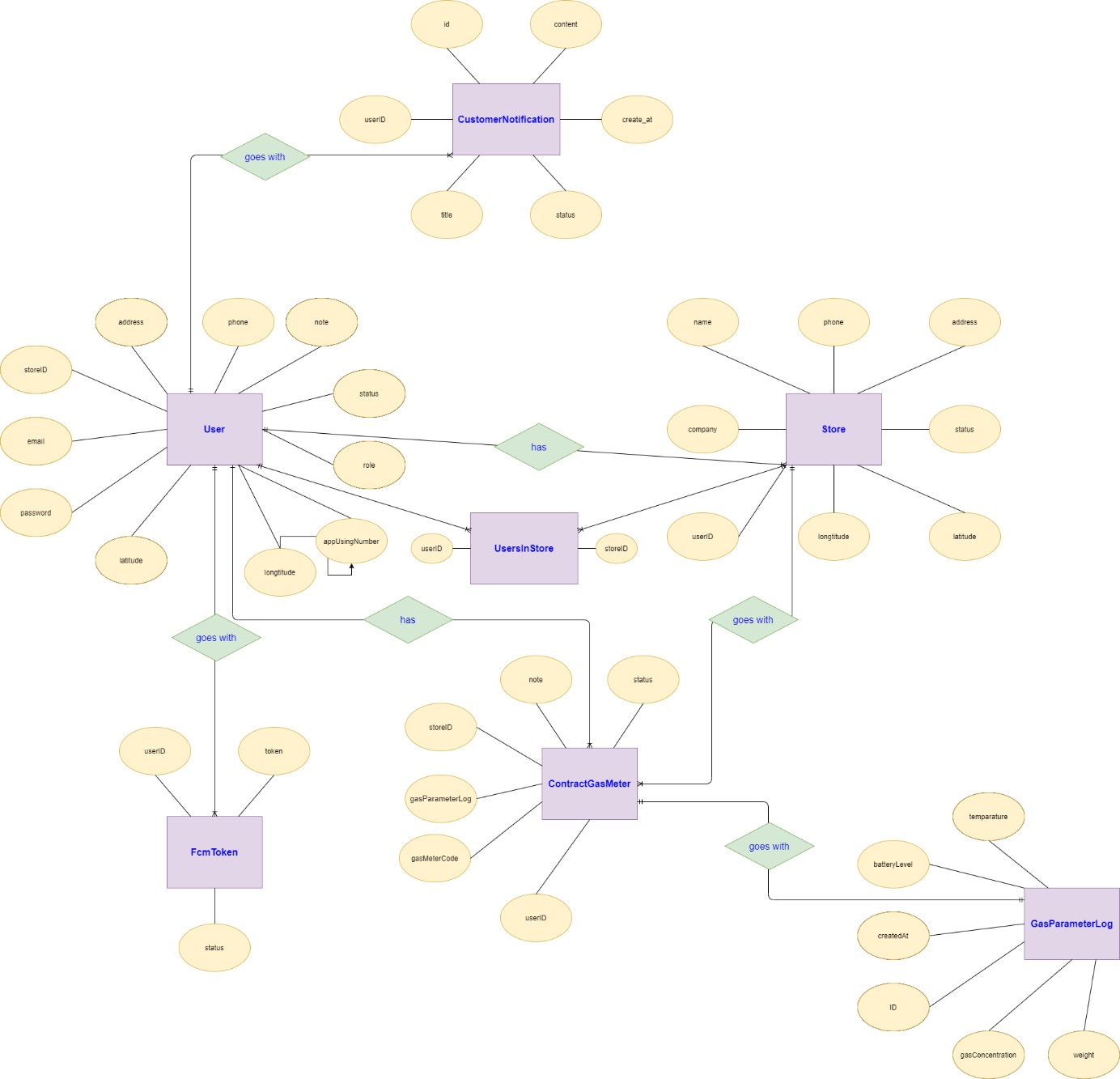


Figure 3.3: Entity Relationship Diagram

# Software Design Description

## Overall Description

This document's purpose is to describe the technical design of the GasMeter application. This will give the developer team the overview of the system’s architecture so they can understand how the system should be implemented and the way the system works. This document will consist of:

* System Architectural Design
* Detailed System Design
* Sequence Diagrams
* Description of all class
* User Interface Designs
* Database Design
  1. Assumptions

This system is designed to base on these following assumptions:

* + - Web-servers run on CentOS which supports Internet connection.
    - Chrome web browser.
    - Database using MySQL.
    - Users with phone devices running Android operating system.
  1. Design Constraints
* End-user’s Environment: Windows, Android.
* Support languages: Vietnamese.
* Web application and mobile application must be responsive and snappy.
* The user’s phone must have a stable connection to the int ernet.
  1. [Technology Suggestion]
* Database: MySQL.
* Web Application: Angular 8.
* Mobile Application: Flutter.
* Server: Java Spring Boot.

## System Architecture Design

* 1. Overall Architecture



**Figure 2.1:** System Architecture

* 1. System Architecture Explanation
     1. Flutter



**Figure 2.2.1** Flutter

Flutter is new Google SDK for mobile devices that helps developers and designers quickly build apps for mobile devices (Android, iOS). In our project, we use flutter to develop mobile application.

* + 1. Firebase



**Figure 2.2.2:** Firebase

Firebase is an excellent backend option for app development. The Google-owned platform has evolved over the years to power lots of apps with cutting-edge features. The platform offers many tools and services that allow developers to perform tasks faster and more efficiently. Firebase handles the backend hassles, so the developers have more time to create excellent frontend features for their apps. In our project, firebase The firebase server is where the data will be stored: user’s profile, account, Gas-meter information … It updates data from the sensor controller or from the controller module. With the changes measured by the sensor, the controller will make the comparison. If there is an abnormality, it will notify by sending the message to the clients. When firebase receives a request to retrieve data, it will retrieve the user's data and display it on the screen of the application.

* + 1. Spring Boot



**Figure 2.2.3** Spring Boot.

Spring boot is an outstanding project in the Spring Framework ecosystem, which is the fastest way to create a standalone REST service. Spring boot simplifies the configuration, in particular, Spring Boot self-configures all by providing specific behaviors. In our project, we use Java Spring Boot for designing backend system.

* + 1. MySQL



**Figure 2.2.4:** MySQL

MySQL is the world's most popular open source database management system (RDBMS). MySQL is fast, stable, easy to use, and works on a variety of operating systems. In our project, we use MySQL for database management.

* + 1. Angular



#### Figure 2.2.5: Angular

Angular is a javascript framework developed by google to build Single Page Applications (SPA) using JavaScript, HTML and TypeScript. Angular provides built-in features for animations, http services, and has features like auto-complete, navigation, toolbar, menus, and more. Code is written in TypeScript, compiled into JavaScript, and displayed similarly in the browser. In our project, we use Angular to develop front-end.

* + 1. HTML5 + SCSS/CSS3



**Figure 2.2.6:** HTML5+SCSS/CSS3

HTML (Hypertext Markup Language) is a platform that helps users to design website elements, structure pages, categories or design applications... So, the main function of this platform is to create layouts, and website format. CSS (Cascading Style Sheets) is a simple mechanism for adding styles (colors, fonts, ...) to web documents. The layout of our website is designed by HTML5.

* + 1. Hibernate



**Figure 2.2.7:** Hibernate

Hibernate framework is an open source, lightweight ORM (Object Relational Mapping) solution. Hibernate helps to simplify the development of java applications to interact with a database. ORM tools simplify data creation, data manipulation, and data access. It is a programming technique for mapping an object to the data stored in a database.

* + 1. CentOS



**Figure 2.2.9:** CentOS

CentOS is a premium operating system supported by its own community. Because of its resemblance to RHEL, CentOS is a perfect programming environment and is one of the

dominant Linux distributions in the world of Linux. CentOS provides a stable environment. As a result, CentOS delivers an enterprise-grade server experience. The Red Hat sponsored operating system uses the same source code as found in RHEL. CentOS uses the RPM package manager. We use centOS for server.

* + 1. Google Cloud Platform

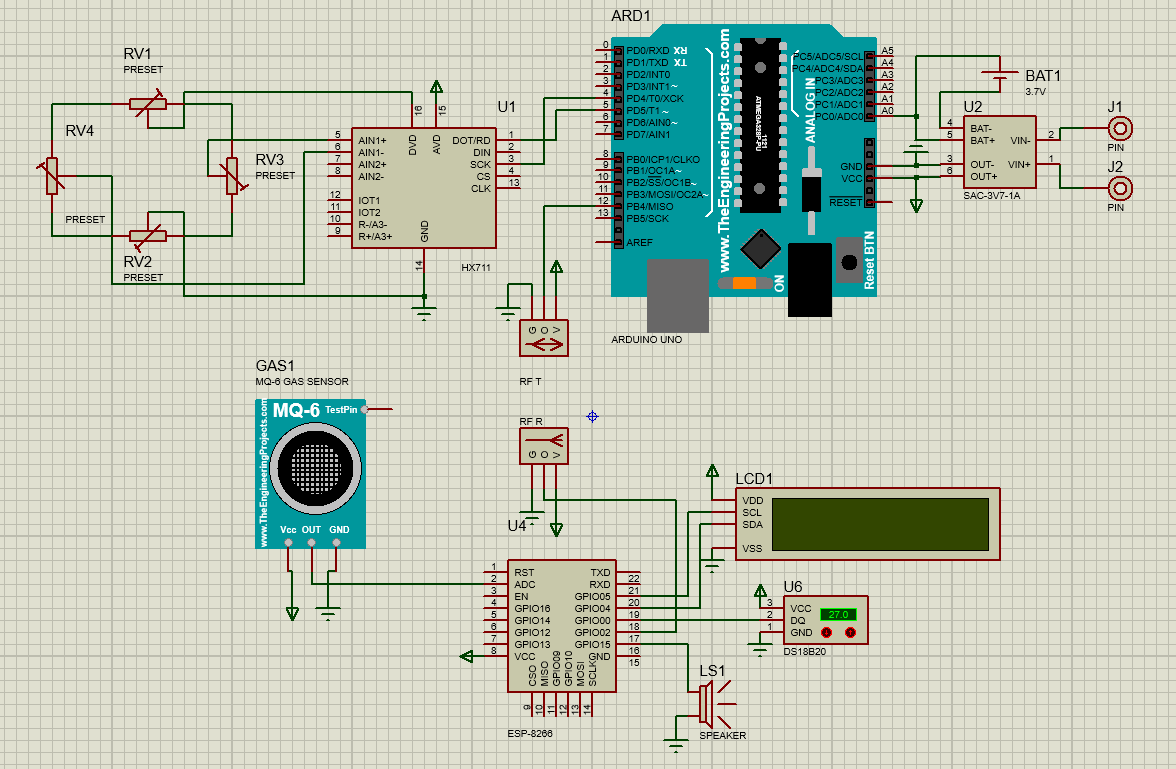


**Figure 2.2.9:** Google Cloud Platform

*An instance is a virtual machine (VM) hosted on Google's infrastructure. You can create an instance by using the Google Cloud Console, the gcloud command-line tool, or the Compute Engine API.*

You can consult here: *https://cloud.google.com/compute/docs/instances*

2.2.11 Embedded Device



**Figure 2.2.11:** Embedded Device

Modular designs such as device under gas cylinder, device on the wall and some protective devices, increase connectivity described in the picture above.

Using Esp32 as a processor because it has a wifi connection function, supports firebase data warehouse connection, helps control and transfer data information to users conveniently, but the price of esp32 is cheap and easy to buy.

##### Hardware devices

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Image*** | ***Information*** |
| Node MCU ESP8266 |  | * CPU: Xtensa Dual-Core 32- bit LX6 microprocessor * Wireless connectivity:   o Wi-Fi: [802.11](https://en.wikipedia.org/wiki/IEEE_802.11) b/g/n/e/i  o Bluetooth: v4.2 BR/EDR and BLE   * USB – 1x micro USB port for power and programming * Misc – BOOT and EN buttons, red (power) and blue (GPIO2) LEDs * Power Supply – 5V via USB or Vin pin * Dimensions – 51.4 x 28.3 mm * ***Used for getting parameter to send to firebase*** |
| Arduino Uno R3 |  | * Microcontroller: ATmega328P Clock Speed: 16MHz * Operating Voltage: 5V * Digital I/O: 14 * Analog Inputs: 6 * Flash Memory: 32 KB (ATmega328P) 0.5 KB is used by the bootloader * EEPROM: 1 KB (ATmega328P) * SRAM: 2KB (ATmega328P) * ***Used for reading signals, managing sensors and***   ***controlling motors*** |

|  |  |  |
| --- | --- | --- |
| LoadCell 50Kg LC3434- 50 |  | * Size: 34mmX34mm. * Load capacity: 50kg. * ***Feature:*** Gravity sensor for electronic scales. |
| [Loadcell](https://www.datasheets360.com/search/results?query=ds18b20&se=ggka&cid=paidsearch&gclid=EAIaIQobChMIwPu5t4qs3AIVAQAAAB0BAAAAEAAYACAAEgJVzfD_BwE) sensor WITH 24  bit ADC – HX711 |  | * Operating voltage: 2.7 ~ 5VDC * Current consumption: <1.5 mA * Sampling rate: 10 - 80 SPS (customized) * Resolution: 24 bit ADC * Voltage resolution: 40 mV * Dimensions: 38 x 21 x 10 mm * ***Feature:***   ADC 24bit Loadcell HX711 ADC converter circuit is used to read resistance value changed from Loadcell sensor (usually very small can not read directly by VĐK) with 24bit ADC resolution and switch to 2- wire communication (Clock) and Data) to submit data to the Microcontroller, suitable for use with load cells in  weighing applications. |

|  |  |  |
| --- | --- | --- |
| 1-cell backup battery charging circuit - 18650  3.7v  battery |  |  |
| 433Mhz RF  Transceiv er |  | Aplication:   * Automated Meter Reading (AMR) * Wireless sensor * Industrial Automation * The control of traffic signal * Wireless handheld terminal * Remote control and monitoring * The management of cars * Wire Replacement * Oil and Gas Detection. * The control of robot |
| Gas Sensor Module LPG /  Butane / Propan MQ6 |  | * Operating source: 5V * Data type: Analog * Wide detection range * Fast response speed and high sensitivity * Simple circuit * Stable when used for a long time * Can detect flammable LPG gases such as methane, propane. * **Feature:**   LPG / Butane / Propan Gas Sensor Module MQ6 is a highly and widely applicable module.  The module detects low |

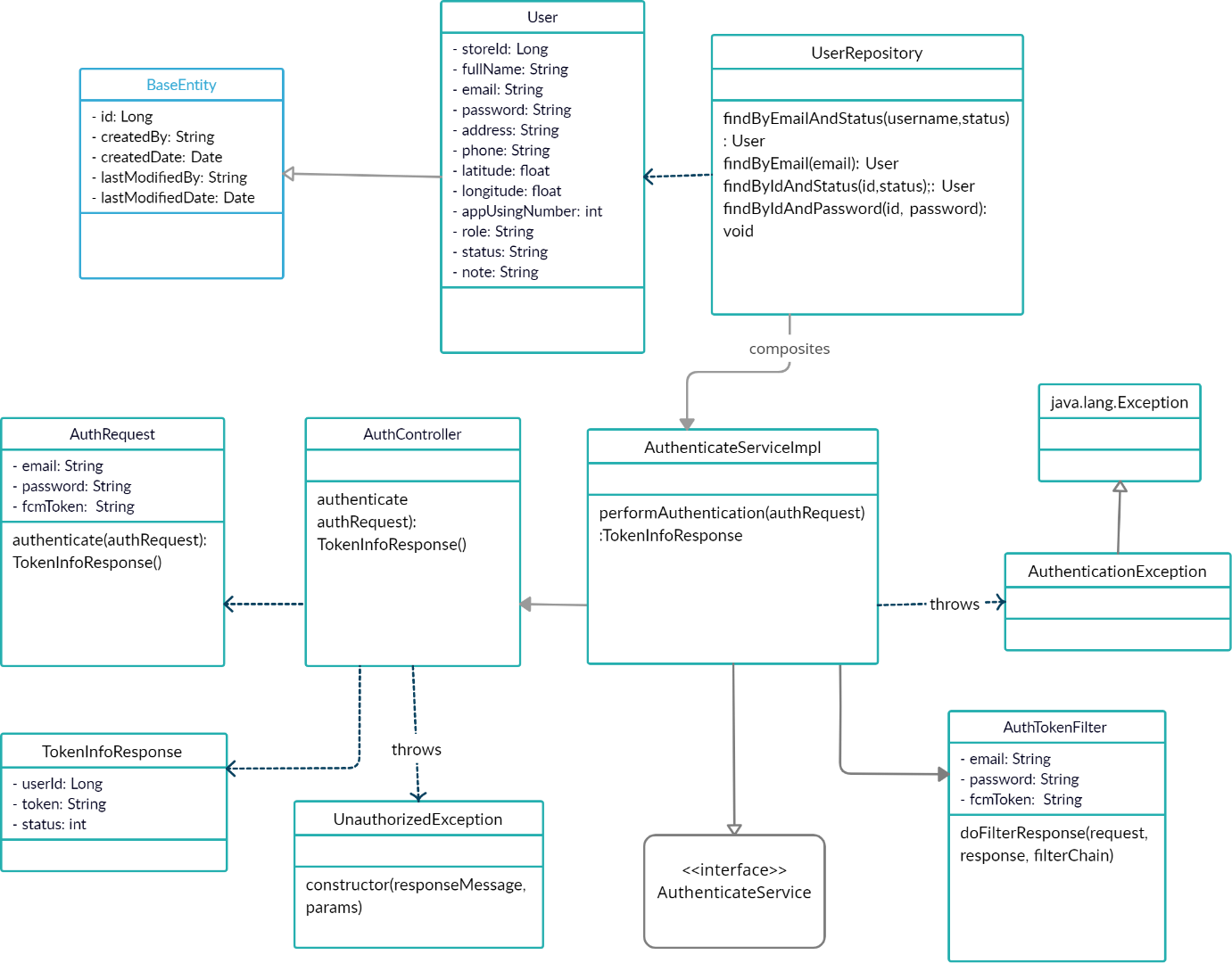


|  |  |  |
| --- | --- | --- |
|  |  | concentrations of flammable gases, is good for warning and firefighting applications, helps to detect gas leaks and can detect some toxic gases in low concentration. |
| Buzz 5V |  | Function is to create sound |
| LCD  screen 0802 |  | * Operating voltage: 5VDC * Screen: 8 lines, 2 characters * Dimensions: 58.0 x 32.0 x   14.0 mm   * Font size: 2.45 x 5.00mm * Dot size: 0.45 x 0.50m * **Feature:**   Blue 0802 5V LCD  display 2 lines of 8 characters each, LCD monitor with contrast  adjustment, integrated green backlight. |

## System Detailed Design

* 1. Login

1. Class Diagram

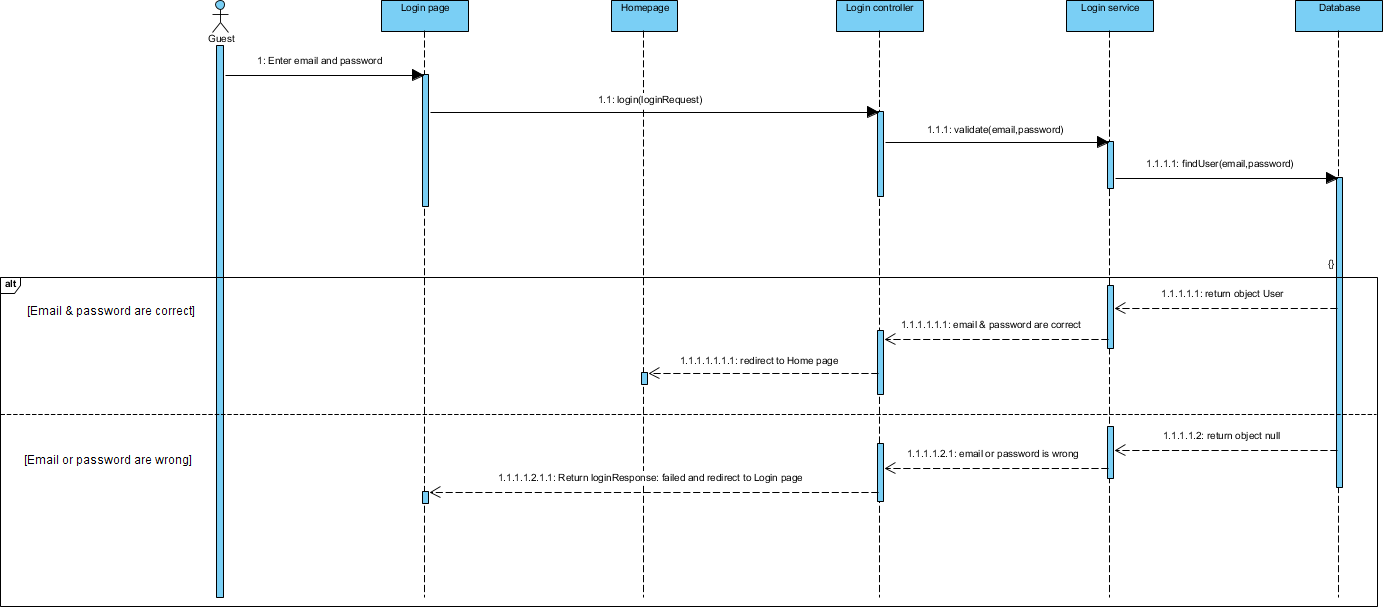


1. Class Specification

**Figure 3.1.1** Login Class Diagram.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to login process. |
| 3 | AuthRequest | Authenticate(authRequest): TokenInfoResponse () | Data request from client send to server. |
| 4 | AuthController | authenticate authRequest() TokenInfoResponse() | Direction of flow for api. |
| 5 | AuthenticateServiceIplm | performAuthentication(authRequest): TokenInfoResponse | Contains authentication methods detail. |

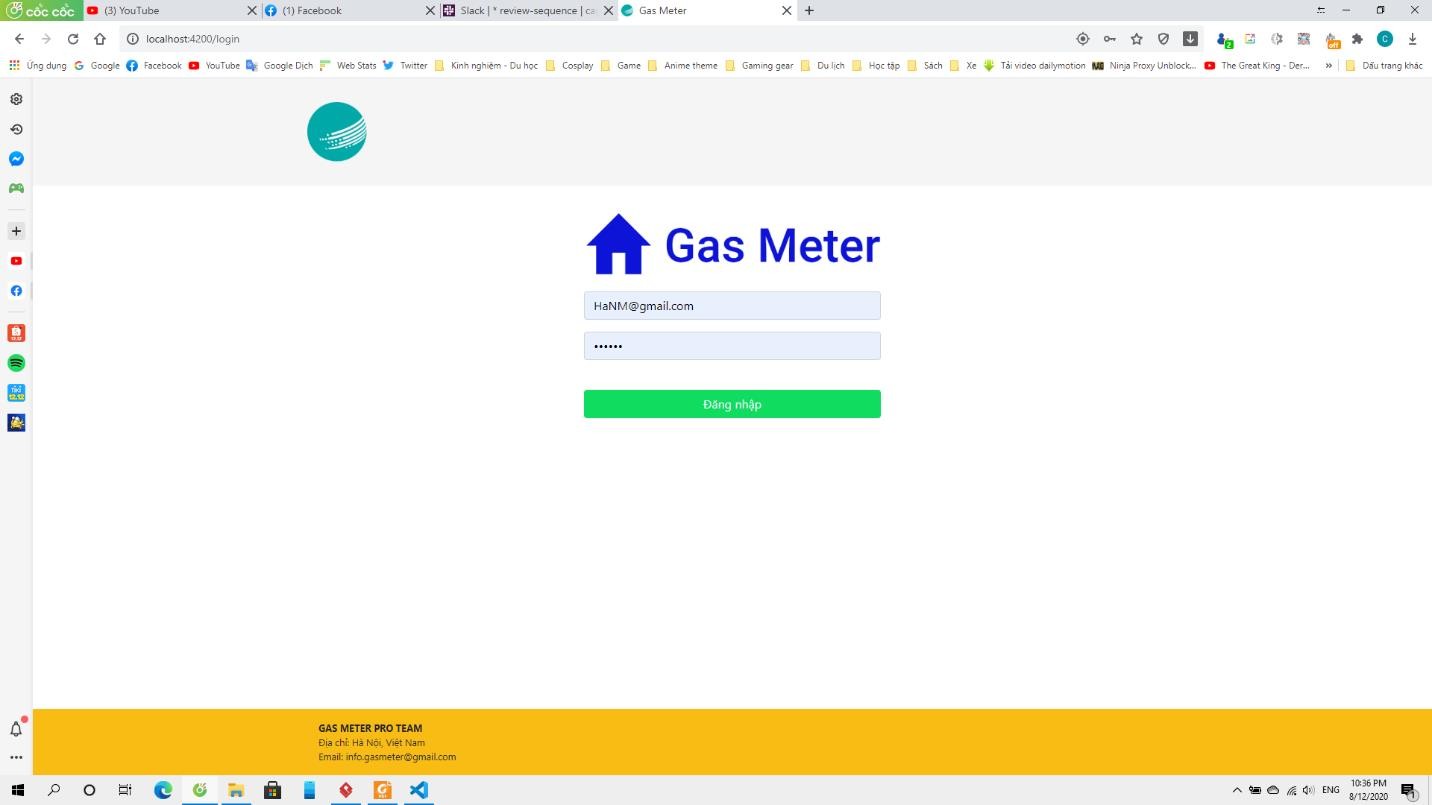
|  |  |  |  |
| --- | --- | --- | --- |
| 6 | UserRepository | findByEmailAndStatus(username, status): User | The method is used to check the database table if there is any user who has the information that. |
| findByEmail(email): User | The method is used to check the database table if there is any user who has the information that. |
| findByIDAndStatus(id, status): User | The method is used to check the database table if there is any user who has the information that. |
| findByIDAndPassword(id,password): void | The method is used to check the database table if there is any user who has the information that. |
| 7 | AuthTokenFilter | doFilterResponse(request, response,filterChain) | Filter user by role |
| 8 | TokenInfoResponse |  | Return information of user with a jwt token for authentication. |
| 9 | AuthentiactionException |  | Handle exception when authentication. |
| 10 | UnauthentiactionException |  | Handle exception except authenticationException. |
| 11 | <<interface>> AuthenticateService |  | Interface to define methods of authentication. |

1. Sequence Diagram(s)
2. Design

**Figure 3.1.1** Login Sequence Diagram.



**Firgure 3.1.2:** Login Screen.

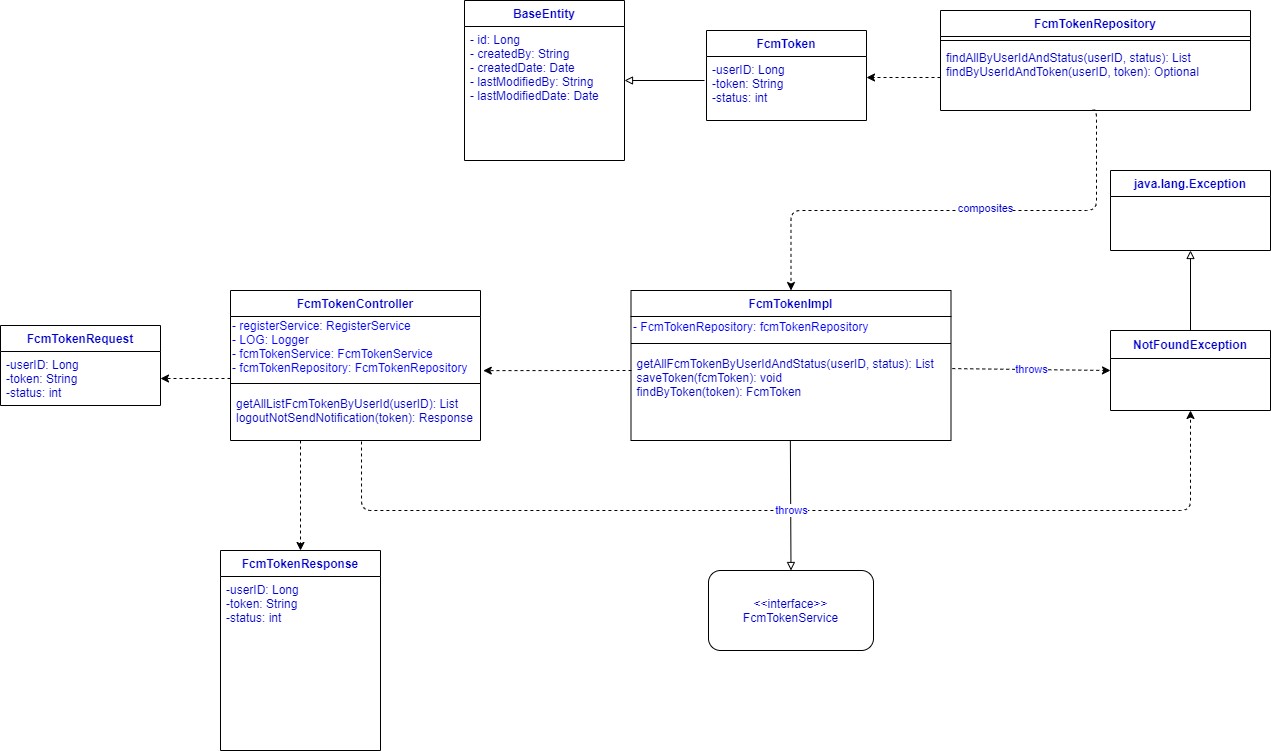


**Firgure 3.1.3:** Manager Login.

* 1. Logout

1. Class Diagram

*[This part presents the class diagram for the relevant feature]*

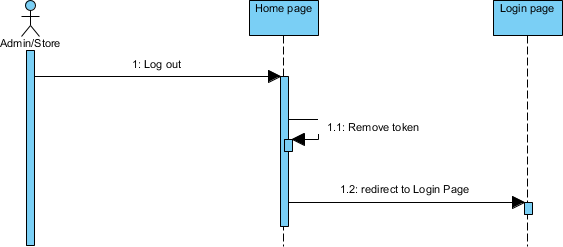


1. Class Specification:

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | FcmToken |  | FcmToken class, which is related to log out process. |
| 3 | FcmTokenRepository | findAllByUserIdAndStatus (userID, status): List | Method to find token by userID if this user is active. |
| saveToken(fcmToken): void | Get token by userID. |
| 4 | FcmTokenImpl | getAllFcmTokenByUserIdAndStatus(user ID, status): List | Get all FCM Token in system. |
| saveToken(fcmToken): void | Save token |
| findByToken(token): FcmToken | Find by token |
| 5 | FcmTokenController | getAllListFcmTokenByUserId(userID): List | Get all Fcm tokens by userID |
| logoutNotSendNotification(token):  Response | Find and update status for  found tokens. |
| getOneUser(ID): List | Get one user by userID. |
| updateUser(id,userRequest, token): ResponseEntity | Update data of user. |
| 6 | FcmTokenRequest |  | Request data from client. |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | FcmTokenResponse |  | Response data from server. |
| 8 | <<interface>> FcmTokenService |  | Interface to define method of user. |

1. Sequence Diagram(s)



c. Design

* 1. Register

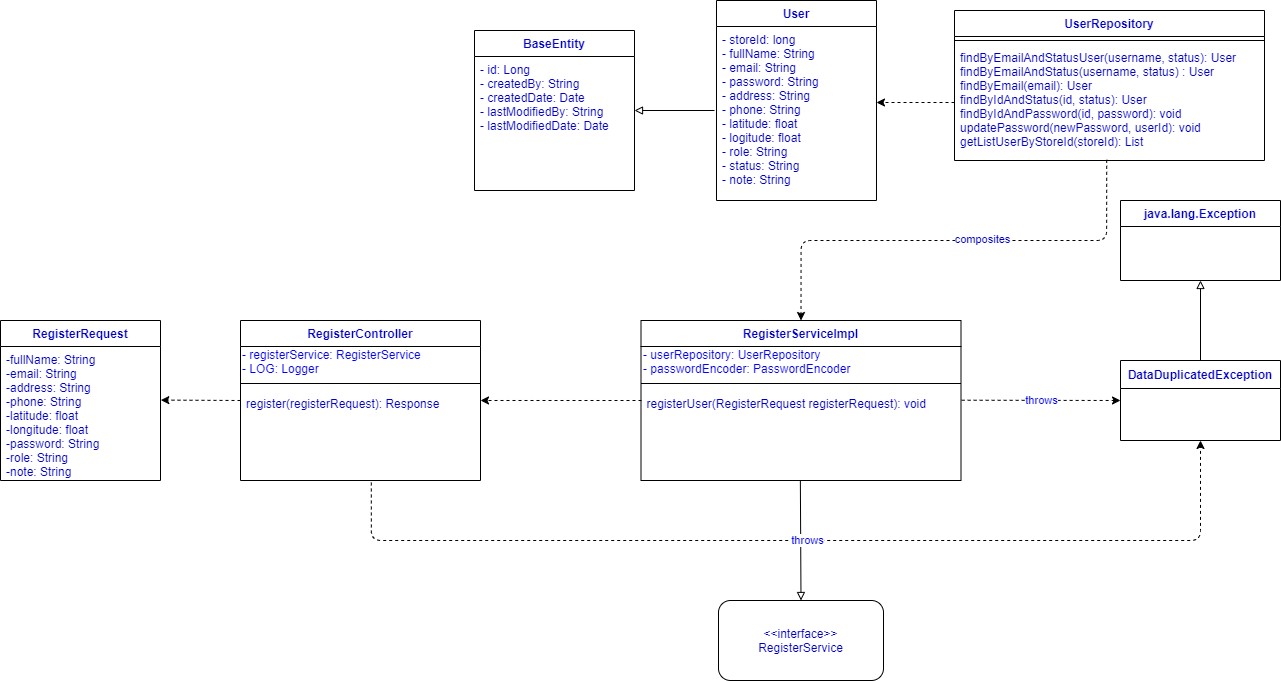
1. Class Diagram

**Firgure 3.2.1:** Logout Sequence Diagram.



**Firgure 3.2.2:** Logout Screen.

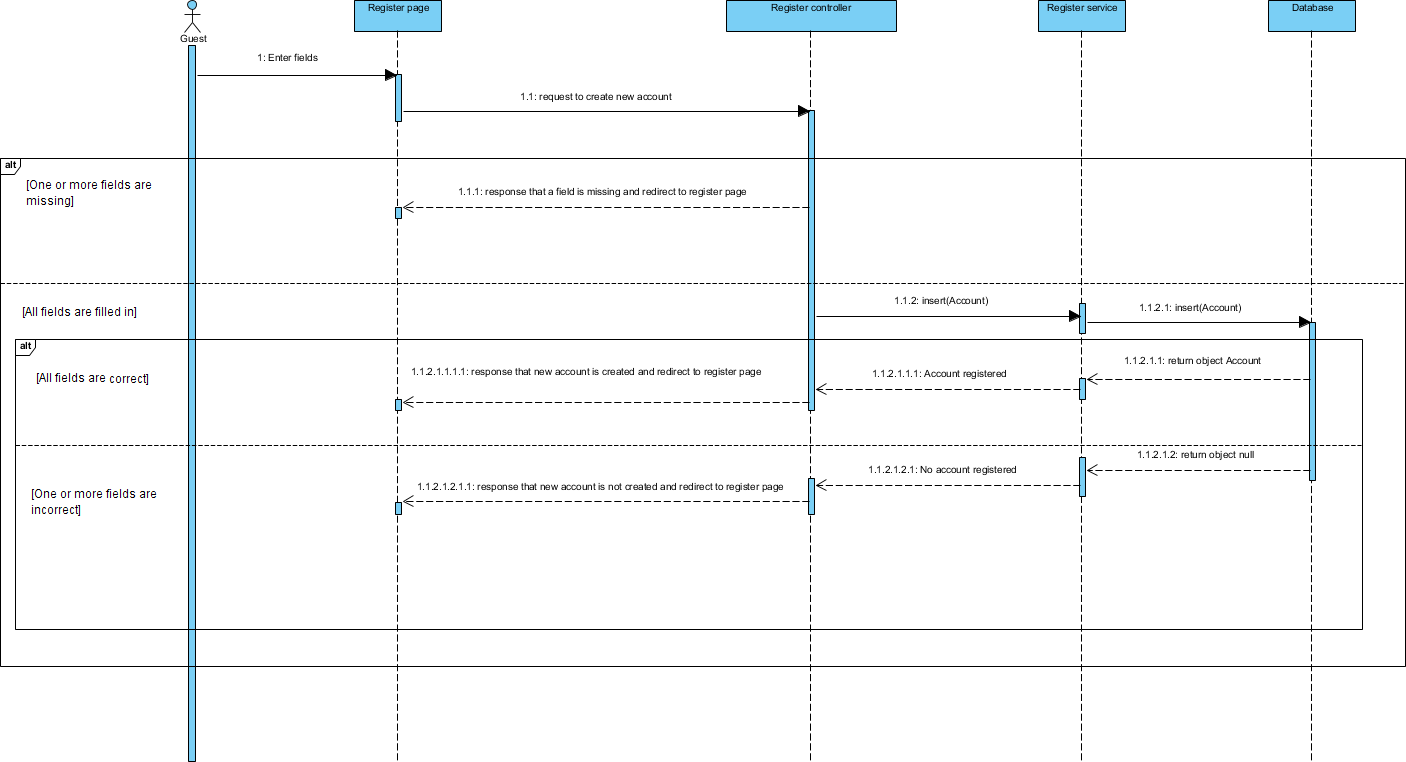
*[This part presents the class diagram for the relevant feature]*



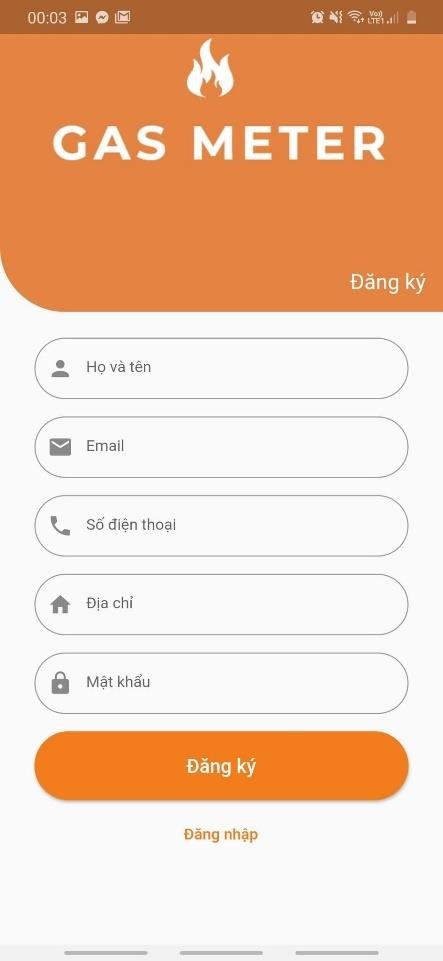
1. Class Specification

*[Provide the description for each class, including both Class Attributes and Class Methods information. Those can be in the table format as below]*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to login process. |
| 3 | UserRepository | findByEmailAndStatus( username, status): User | Method to find user by email if this user is active. |
| findByEmail(email): User | Method to find user by email. |
| searchUser(Strign text): List | Search user by key word. |
| updatePassword (newPassword,userID): void | Method to update password of user. |
| getListUserByStoreID(storeID): List | Get list user by store(User of this store). |
| 4 | RegisterServiceImpl | registerUser(RegisterRequest registerRequest) | Method to register user. |
| 5 | RegisterController | register(registerRequest): Response | Response that user is registered or not. |
| 6 | RegisterRequest |  | Request data from client. |
| 8 | <<interface>> RegisterService |  | Interface to define method of RegisterServiceImpl. |

1. Sequence Diagram(s)
2. Design

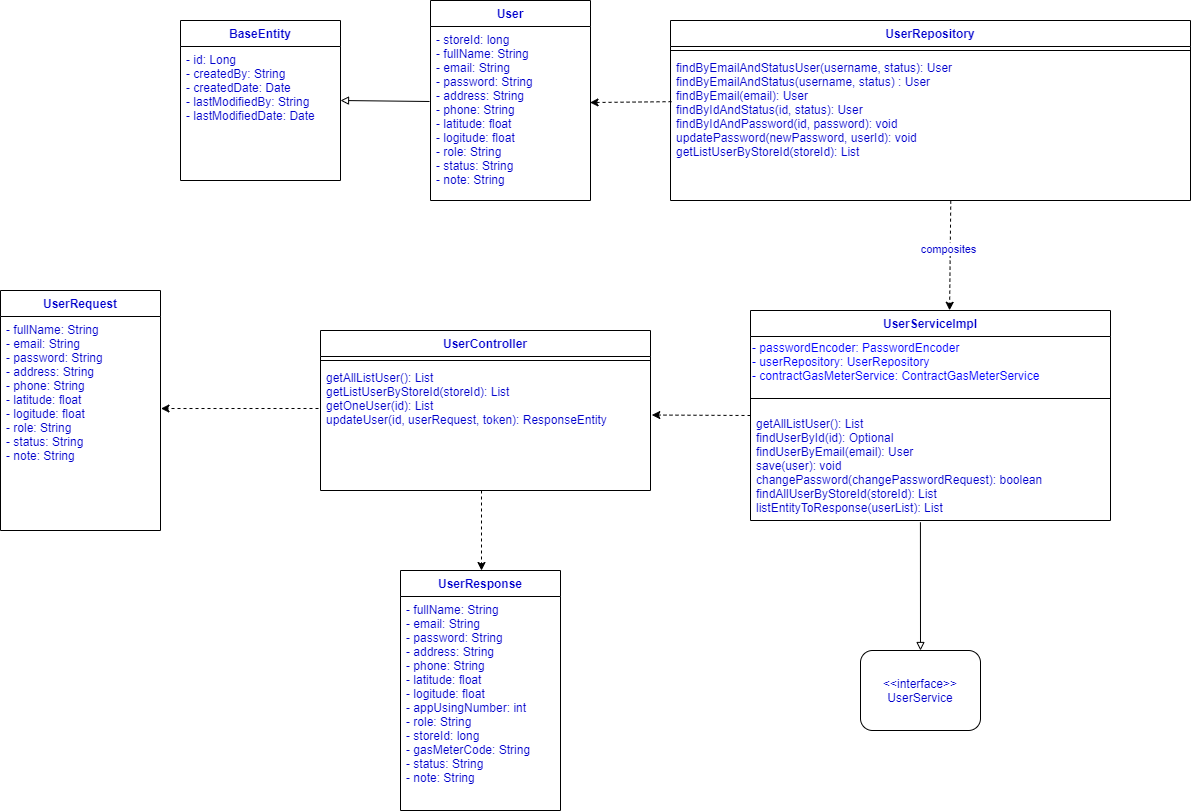
**Firgure 3.3.1:** Register Sequence Diagram.



**Firgure 3.3.2:** Register Screen.

* 1. Update Profile

1. Class Diagram

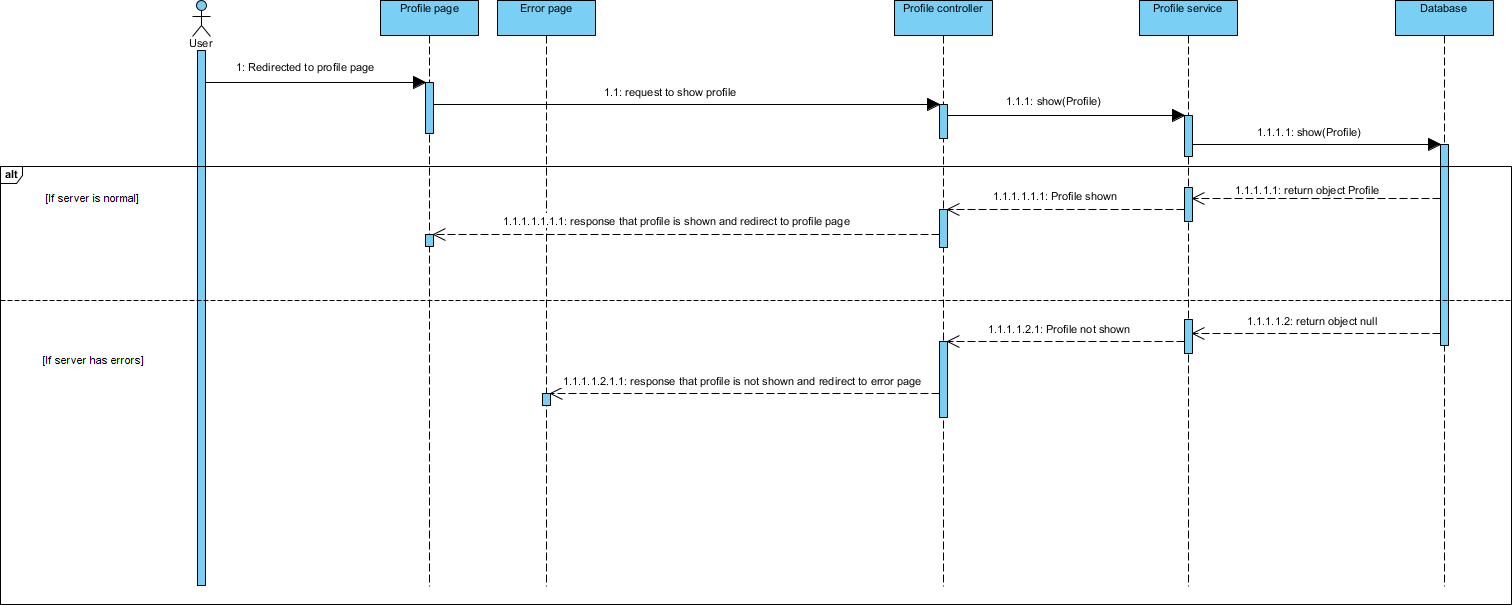


**Firgure 3.9.1:** Update User Class Diagram.

1. Class Specification

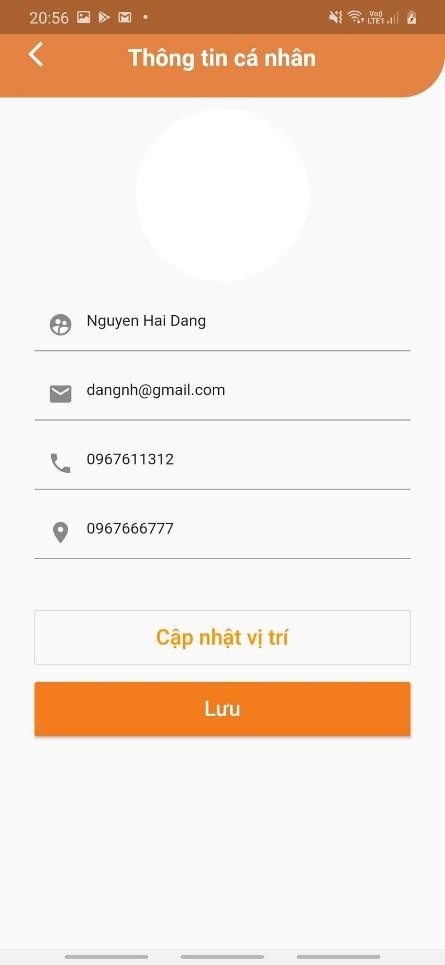
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to update process. |
| 3 | UserRepository | findByEmailAndStatus(username, status): User | Method to find user by email if this user is active. |
| findByEmail(email): User | Method to find user by email. |
| searchUser(Strign text): List | Search user by key word. |
| updatePassword (newPassword,userID): void | Method to update password of user. |
| getListUserByStoreID(storeID): List | Get list user by store(User of this store). |
| getAllListUser(): List | Get all user in system. |
| findUserByID: Optional | Get user by userID. |
| 4 | UserServiceImpl | findUserByEmail(email) : User | Get user by email. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | save(user): void | Add user to database. |
| changePassword(changePassrequest): boolean | Change password of user. |
| findAllUserByStoreID(storeID): List | Get all user by storeID. |
| listEntityToResponse(userList): List | Convert user entity to response data. |
| getAllListUser():List | Get all user in system. |
| getListUserByStoreID(storeID): List | Get all user by storeID. |
| 5 | UserController | getOneUser(id): List | Get one user by userID. |
| updateUser(id,userRequest, token): ResponseEntity | Update datta of user. |
|  | Request data from client . |
|  | Response data from server. |
| 6 | <<interface>> UserService |  | Interface to define method of user. |

1. Sequence Diagram(s)

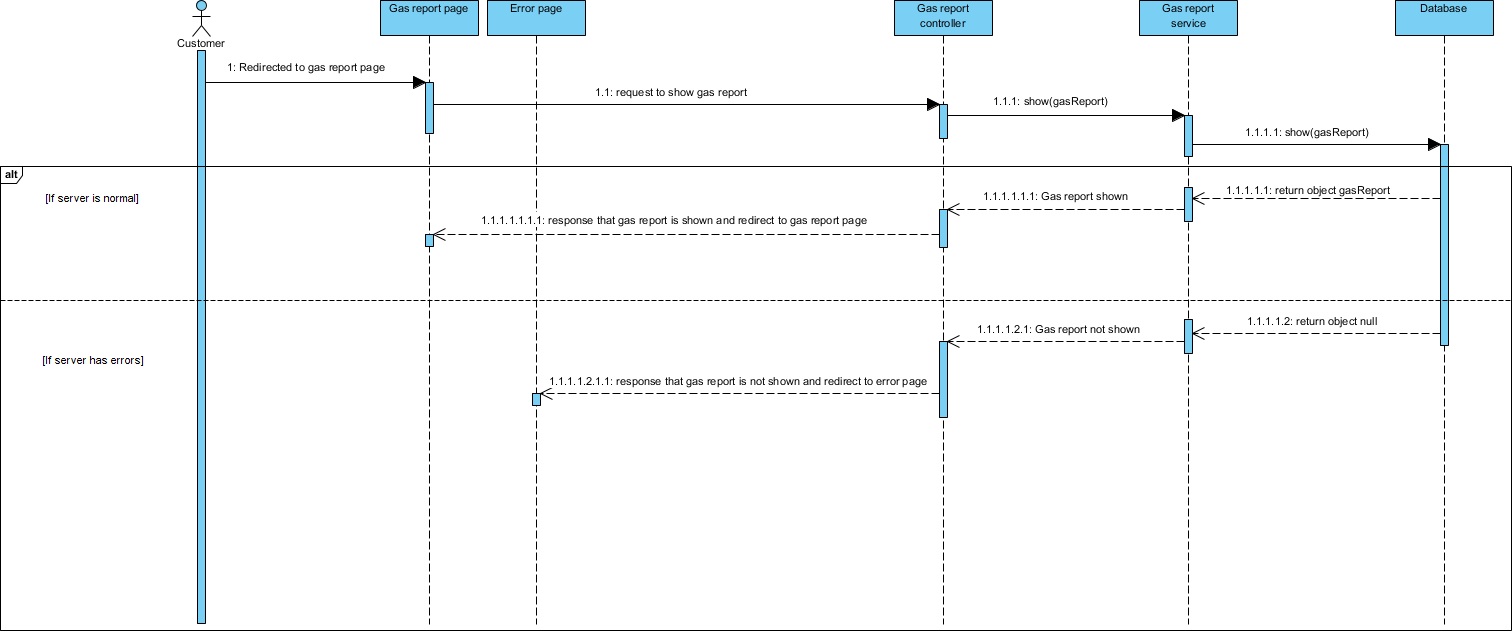
**Firgure 3.4.1:** View Profile Sequence Diagram.

c. Design:



**Firgure 3.4.2:** Update Profile Screen.

* 1. View Gas-report

1. Sequence Diagram(s)

**Firgure 3.5.1:** View Gas-report Sequence Diagram.

1. Design:

* 1. View Notification

**Firgure 3.5.2:** View Gas-report Screen.

*[Provide the detailed design for the feature <Feature Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

1. Class Diagram

*[This part presents the class diagram for the relevant feature]*

1. Class Specification

*[Provide the description for each class, including both Class Attributes and Class Methods information. Those can be in the table format as below]*

b1. XYZ Class

*[Provide the detailed description for the class methods]*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | <method name> | <Description of the method, including the inputs, outputs & internal method processing> |
|  |  |  |

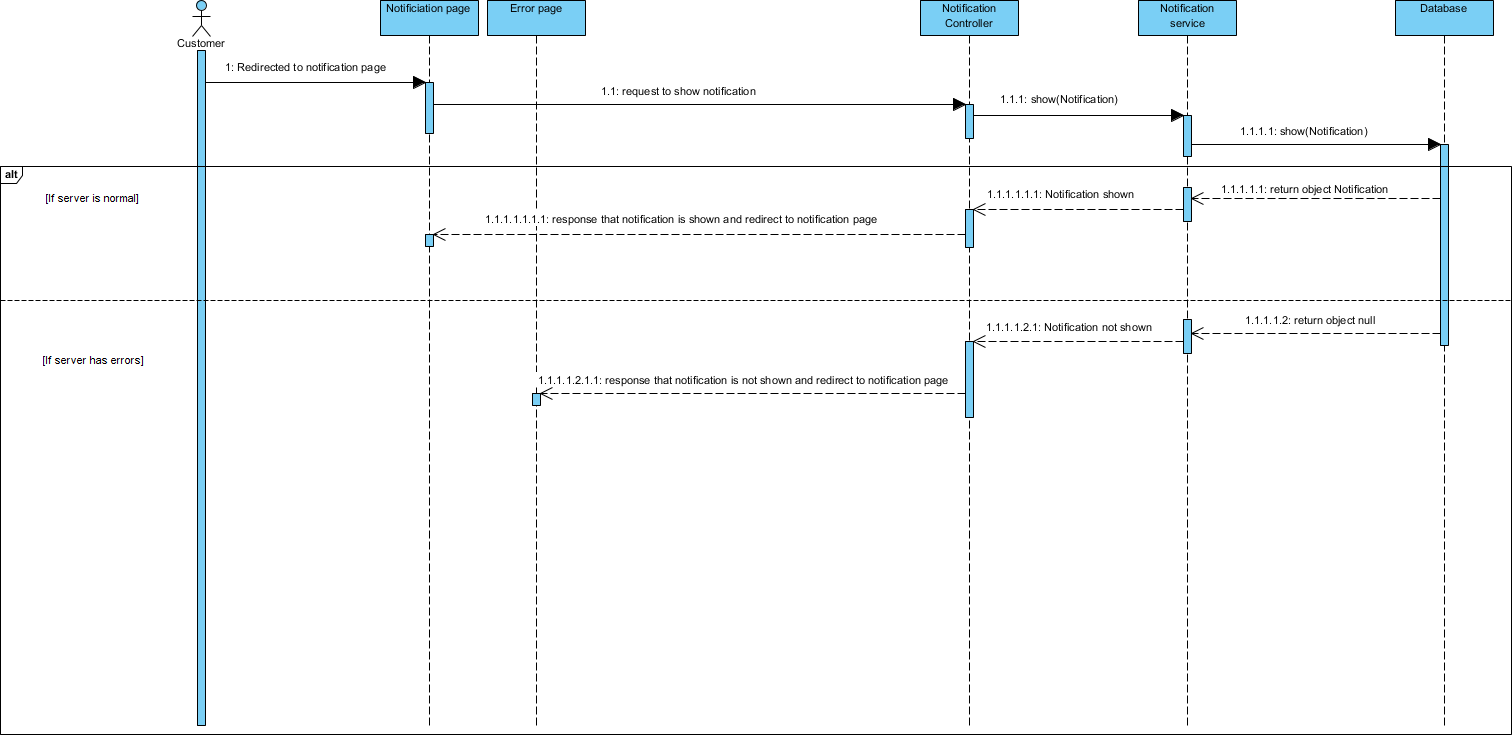
b2. ABC Class

***Class Methods***

*[Provide the detailed description for the class methods]*

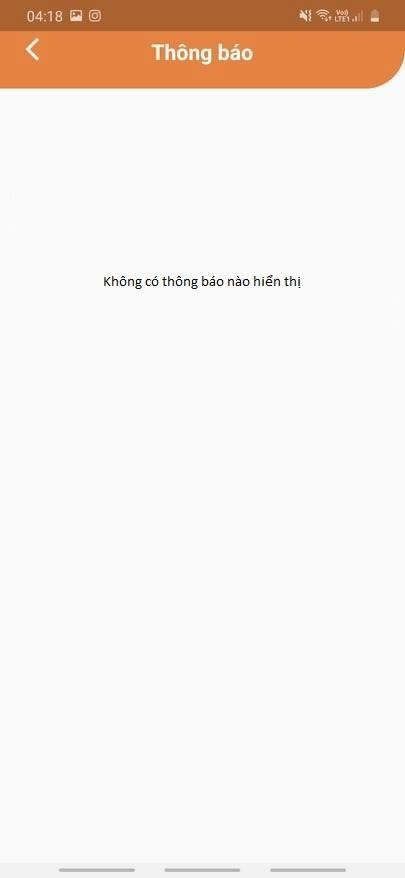
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | <method name> | <Description of the method, including the inputs, outputs & internal method processing> |
|  |  |  |

b3. …

1. Sequence Diagram(s)

**Firgure 3.6.1:** View Notification Sequence Diagram.

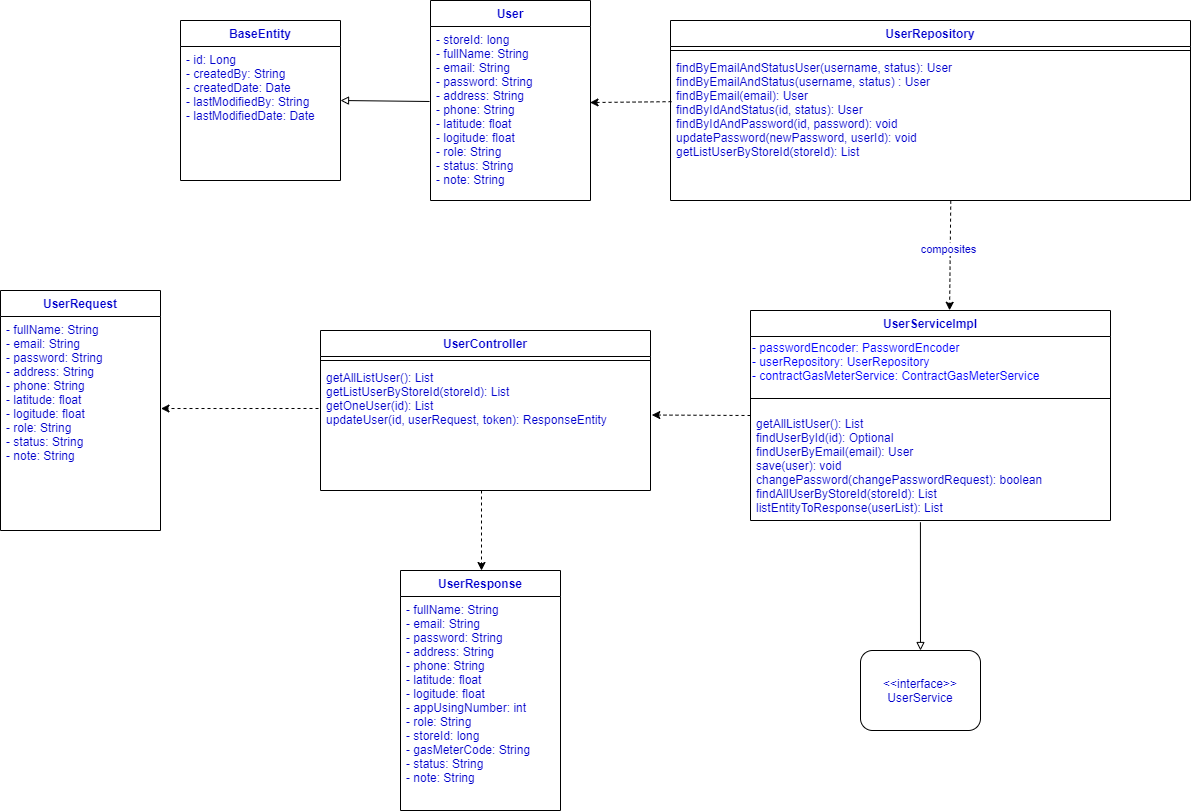
1. Design



**Firgure 3.6.2:** View Notification Design.

* 1. View List - user

1. Class Diagram



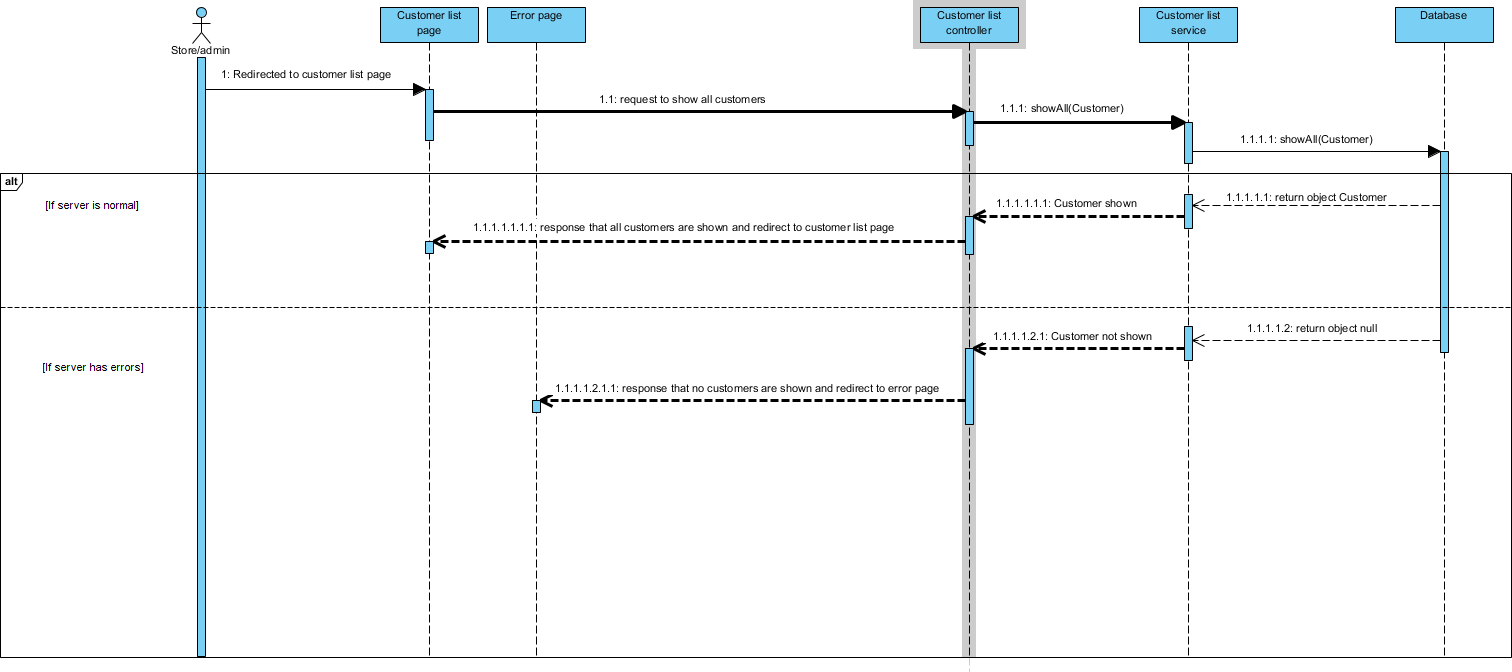
1. Class Specification

**Firgure 3.7.1:** View List-user Class Diagram.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to login process. |
| 3 | UserRepository | findByEmailAndStatus( username, status): User | Method to find user by email if this user is active. |
| findByEmail(email): User | Method to find user by email. |
| searchUser(Strign text): List | Search user by key word. |
| updatePassword (newPassword,userID): void | Method to update password of user. |
| getListUserByStoreID(storeID): List | Get list user by store(User of this store). |
| 4 | UserServiceImpl | getAllListUser(): List | Get all user in system. |
| findUserByID: Optional | Get user by userID. |
| findUserByEmail(email) : User | Get user by email. |

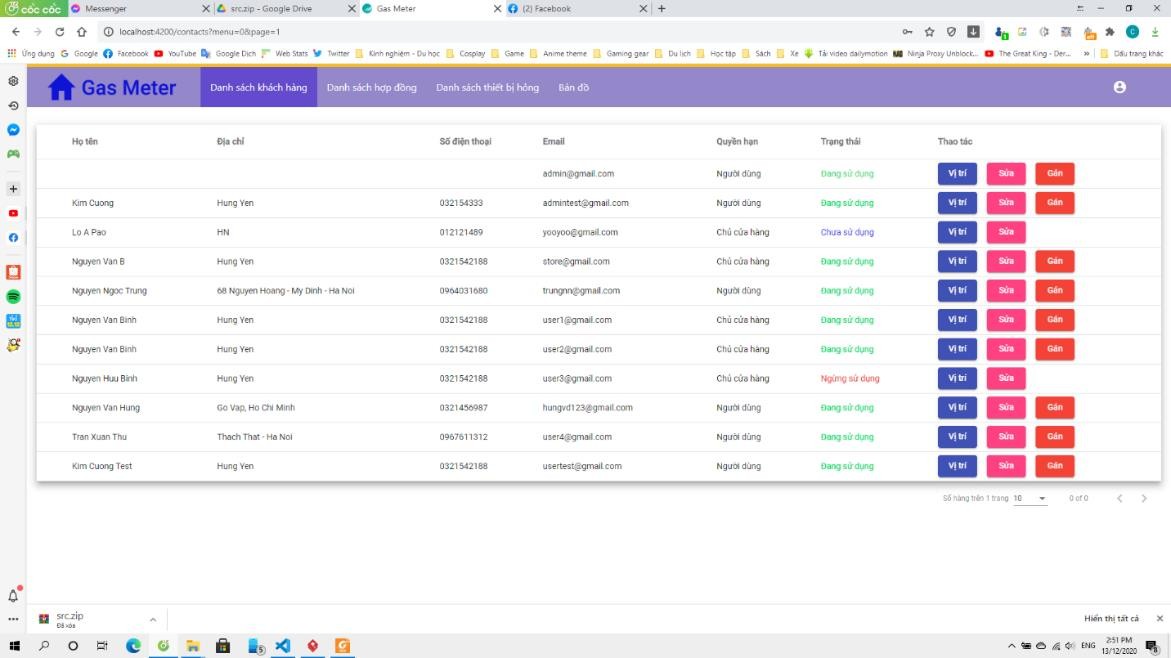
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | save(user): void | Add user to database. |
| changePassword(changePassrequest): boolean | Change password of user. |
| findAllUserByStoreID(storeID): List | Get all user by storeID. |
| listEntityToResponse(userList): List | Convert user entity to response data. |
| 5 | UserController | getAllListUser():list | Get all user in system. |
| getListUserByStoreID(storeID): List | Get all user by storeID. |
| getOneUser(id): List | Get one user by userID. |
| updateUser(id,userRequest, token): ResponseEntity | Update datta of user. |
| 6 | UserRequest |  | Request data from client . |
| 7 | UserResponse |  | Response data from server. |
| 8 | <<interface>> UserService |  | Interface to define method of user. |

1. Sequence Diagram(s)



**Firgure 3.7.2:** View List-user Sequence Diagram.

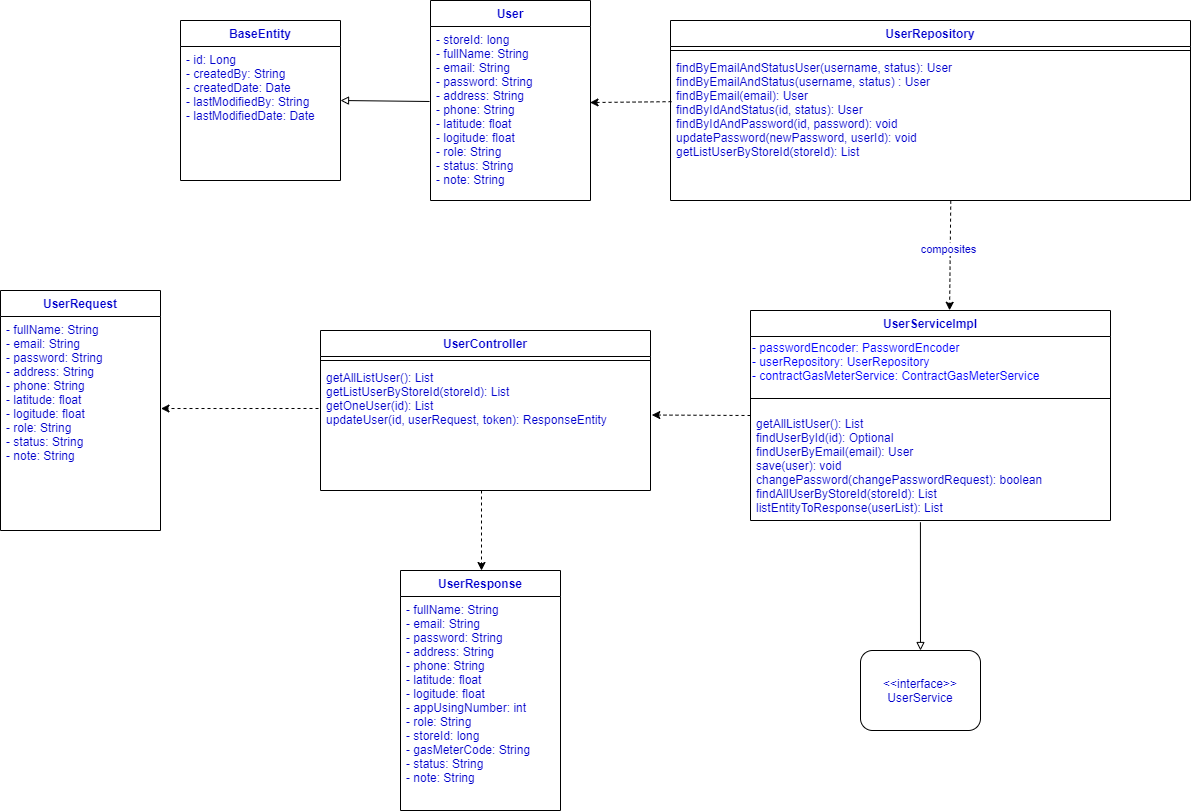
1. Design:



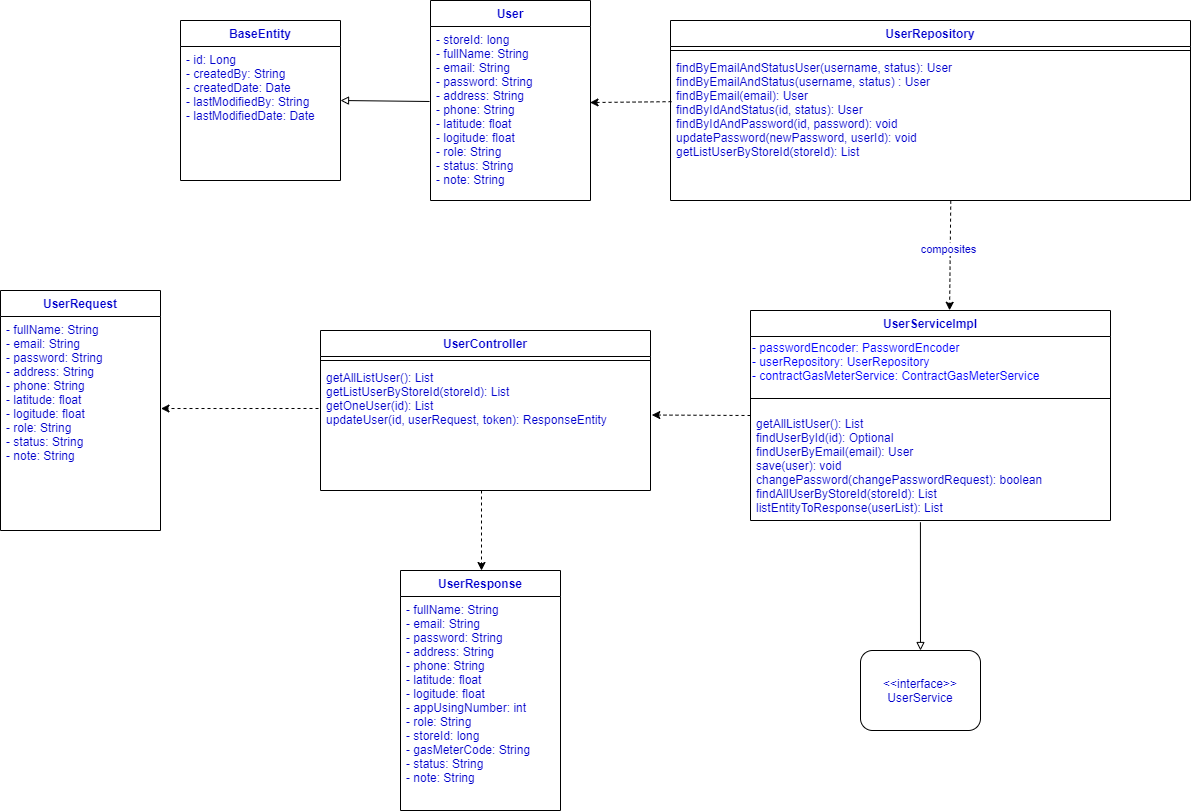
**Firgure 3.7.3:** View List-user Sequence Diagram.

* 1. Search User

1. Class Diagram



**Firgure 3.8.1:** Search User by A Class Diagram.

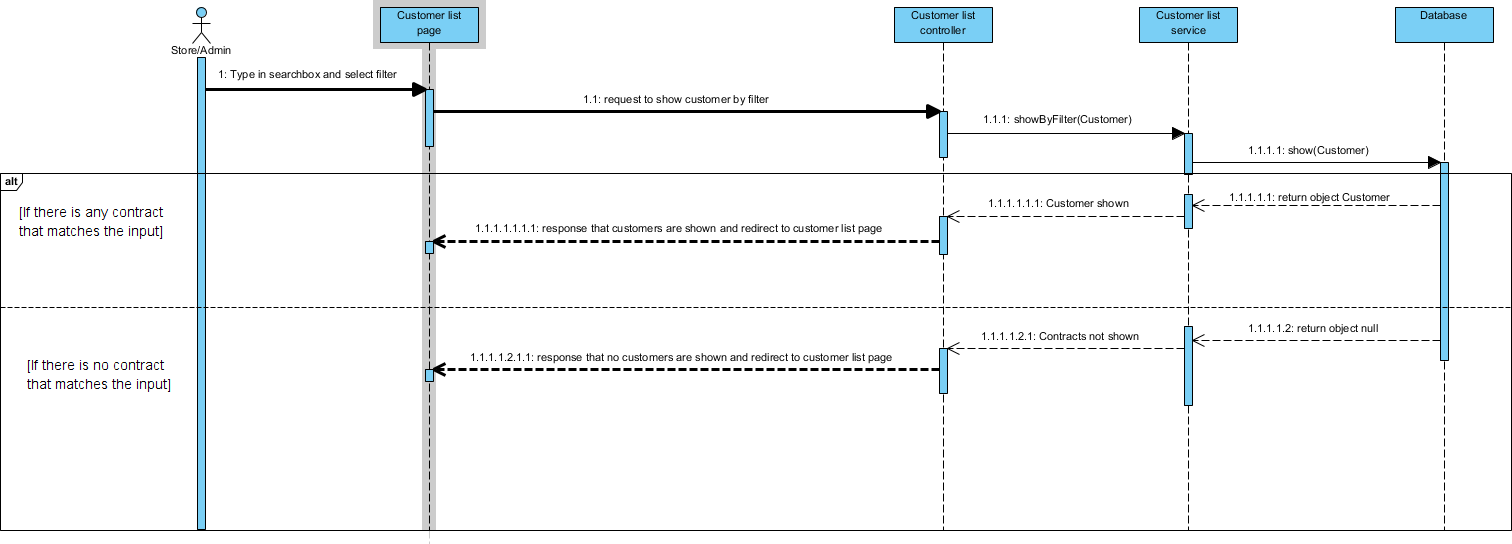


1. Class Specification

**Firgure 3.8.2:** Search User by storeID Class Diagram.

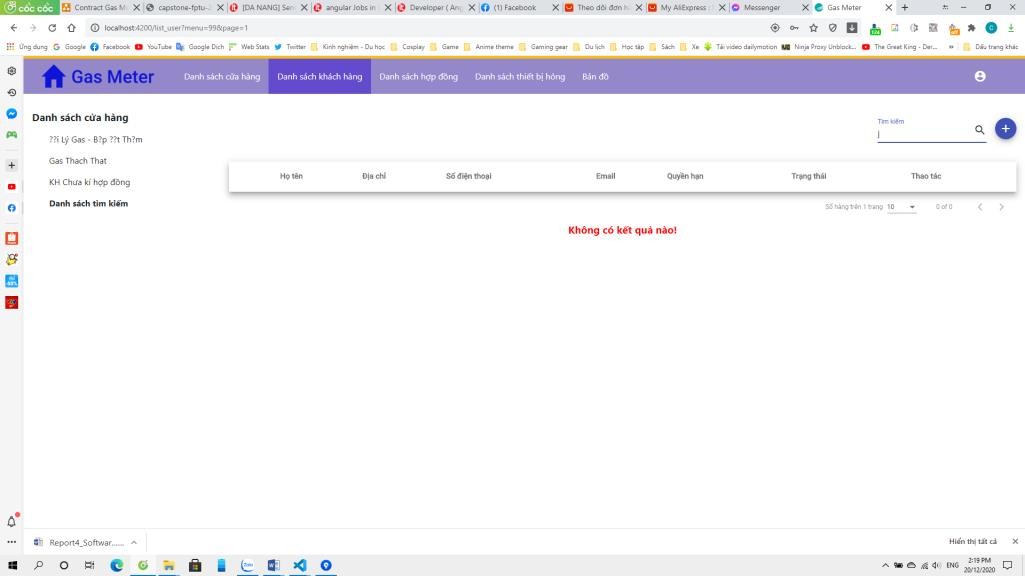
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to search process. |
| 3 | UserRepository | findByEmailAndStatus( username, status): User | Method to find user by email if this user is active. |
| findByEmail(email): User | Method to find user by email. |
| searchUser(Strign text): List | Search user by key word. |
| updatePassword (newPassword,userID): void | Method to update password of user. |
| getListUserByStoreID(storeID): List | Get list user by store(User of this store). |
| 4 | UserServiceImpl | getAllListUser():List | Get all user in system. |
| findUserByID: Optional | Get user by userID. |
| findUserByEmail(email) : User | Get user by email. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | save(user): void | Add user to database. |
| changePassword(changePassrequest): boolean | Change password of user. |
| findAllUserByStoreID(storeID):List | Get all user by storeID. |
| listEntityToResponse(userList): List | Convert user entity to response data. |
| 5 | UserController | getAllListUser():list | Get all user in system. |
| getListUserByStoreID(storeID): List | Get all user by storeID. |
| getOneUser(ID): List | Get one user by userID. |
| updateUser(id,userRequest, token): ResponseEntity | Update data of user. |
| 6 | UserRequest |  | Request data from client. |
| 7 | UserResponse |  | Response data from server. |
| 8 | <<interface>> UserService |  | Interface to define method of user. |

1. Sequence Diagram(s)

**Firgure 3.8.3:** Search User Sequence Diagram.

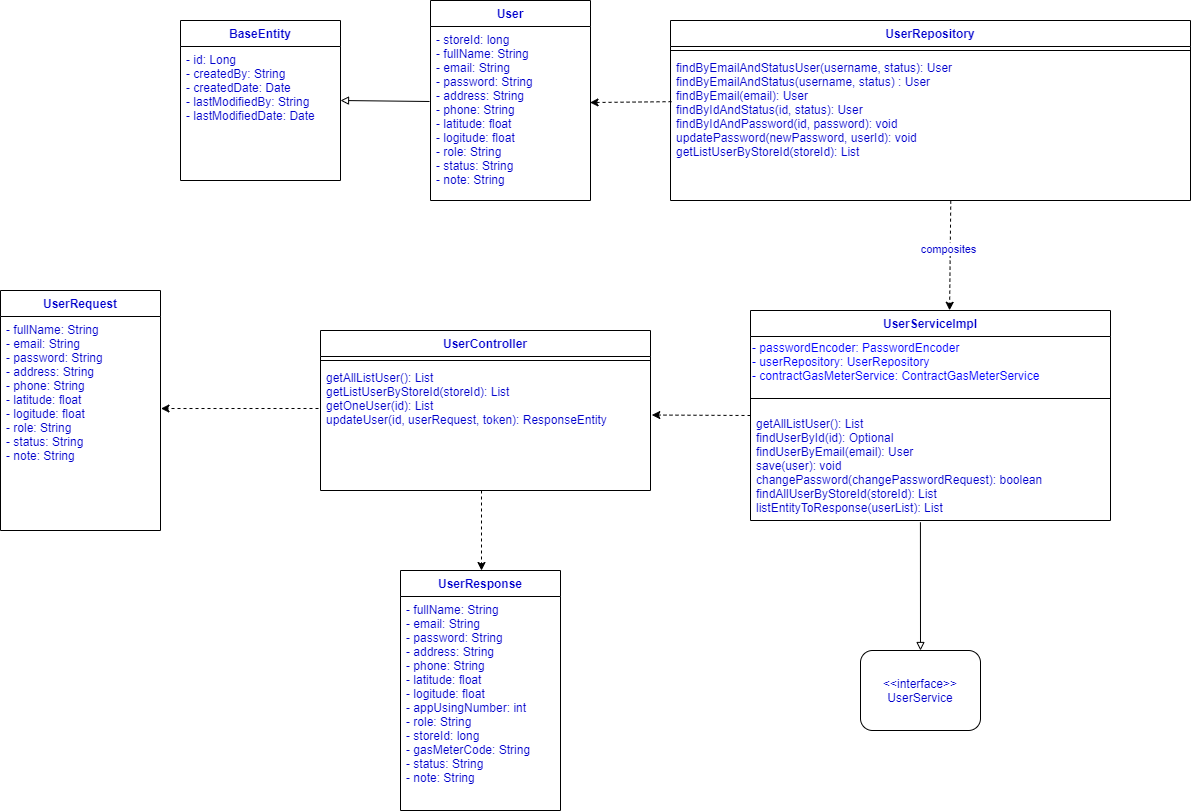
1. Design:



**Firgure 3.8.4:** Search User Screen.

* 1. Update User

1. Class Diagram



**Firgure 3.9.1:** Update User Class Diagram.

1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |

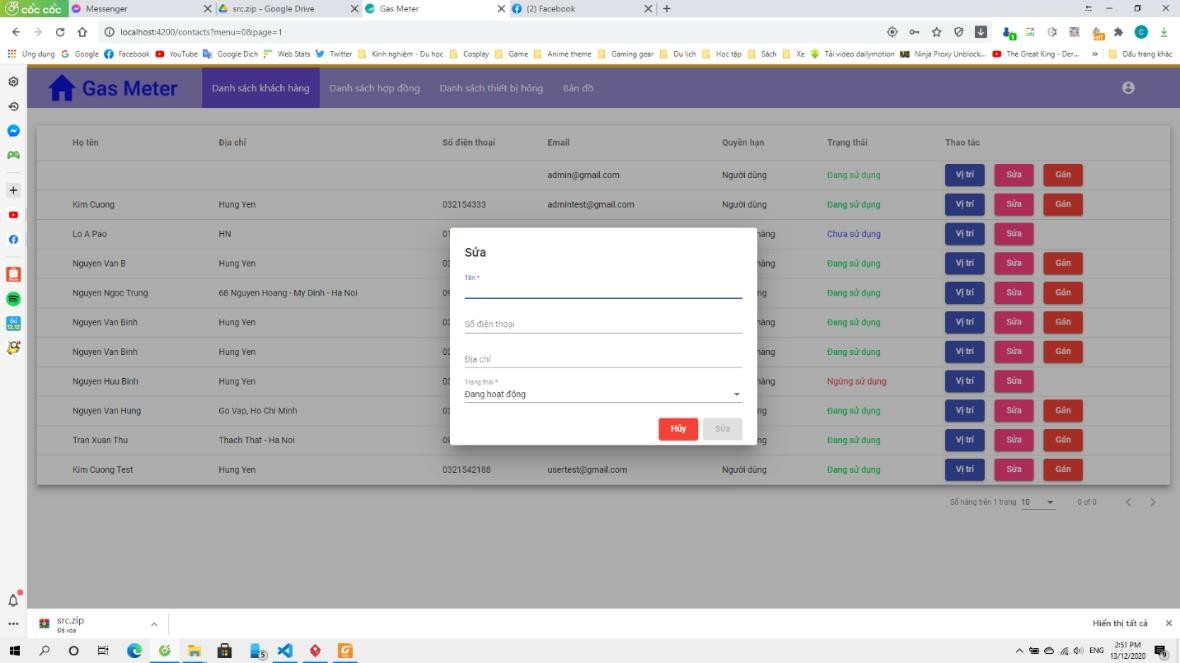
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | User |  | User class, which is related to update process. |
| 3 | UserRepository | findByEmailAndStatus(username, status): User | Method to find user by email if this user is active. |
| findByEmail(email): User | Method to find user by email. |
| searchUser(Strign text): List | Search user by key word. |
| updatePassword (newPassword,userID): void | Method to update password of user. |
| getListUserByStoreID(storeID): List | Get list user by store(User of this  store). |
| getAllListUser(): List | Get all user in system. |
| findUserByID: Optional | Get user by userID. |
| 4 | UserServiceImpl | findUserByEmail(email) : User | Get user by email. |
| save(user): void | Add user to database. |
| changePassword(changePassrequest): boolean | Change password of user. |
| findAllUserByStoreID(storeID): List | Get all user by storeID. |
| listEntityToResponse(userList): List | Convert user entity to response data. |
| getAllListUser():List | Get all user in system. |
| getListUserByStoreID(storeID): List | Get all user by storeID. |
| 5 | UserController | getOneUser(id): List | Get one user by userID. |
| updateUser(id,userRequest, token): ResponseEntity | Update datta of user. |
|  | Request data from client . |
|  | Response data from server. |
| 6 | <<interface>> UserService |  | Interface to define method of user. |

1. Sequence Diagram(s)



1. Design

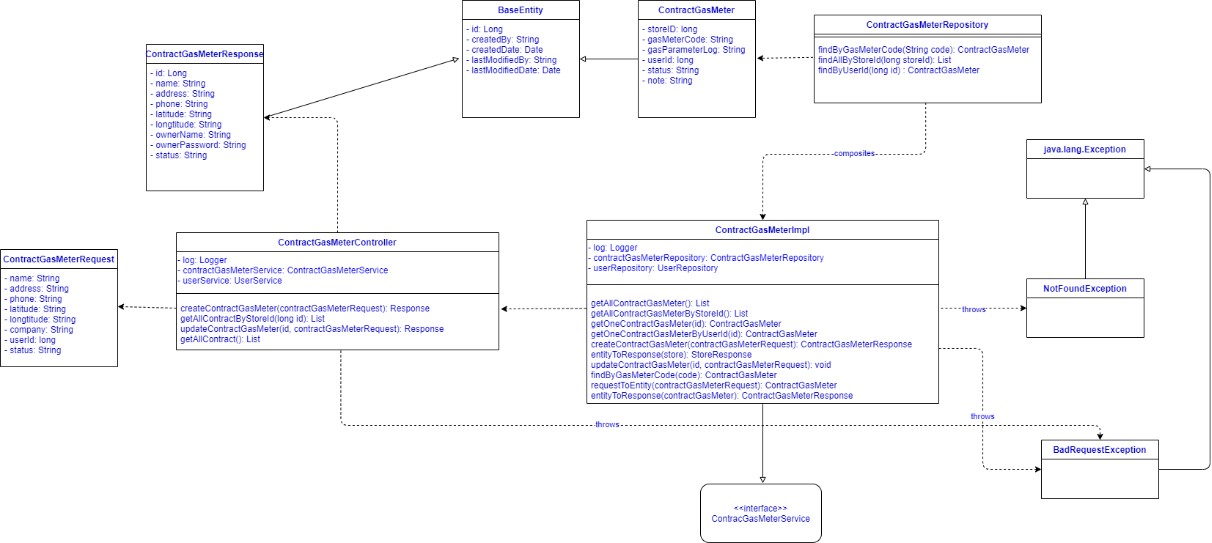
**Firgure 3.9.2:** Update User Sequence Diagram.



**Firgure 3.9.1:** Update User Screen.

* 1. View list contract

1. Class Diagram

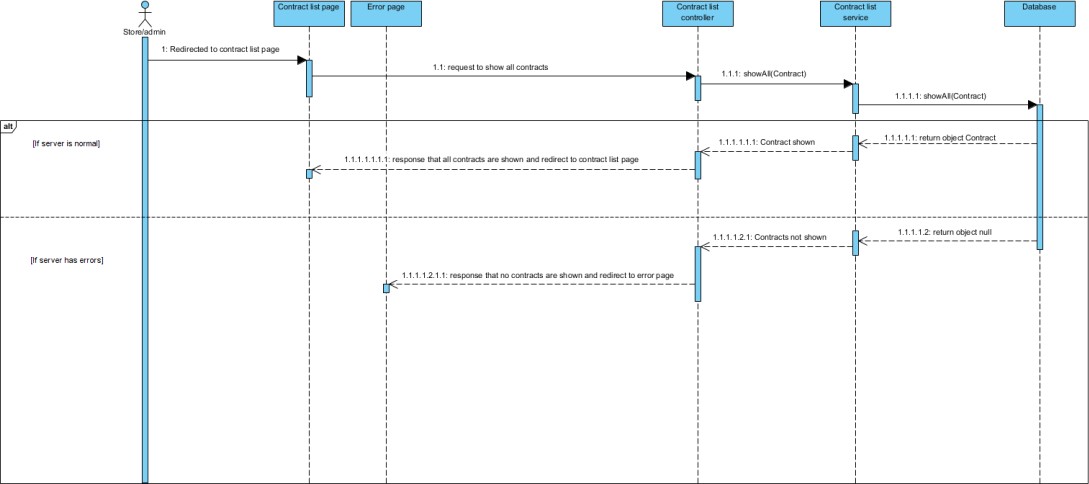


**Firgure 3.10.1:** View List Contract Class Diagram.

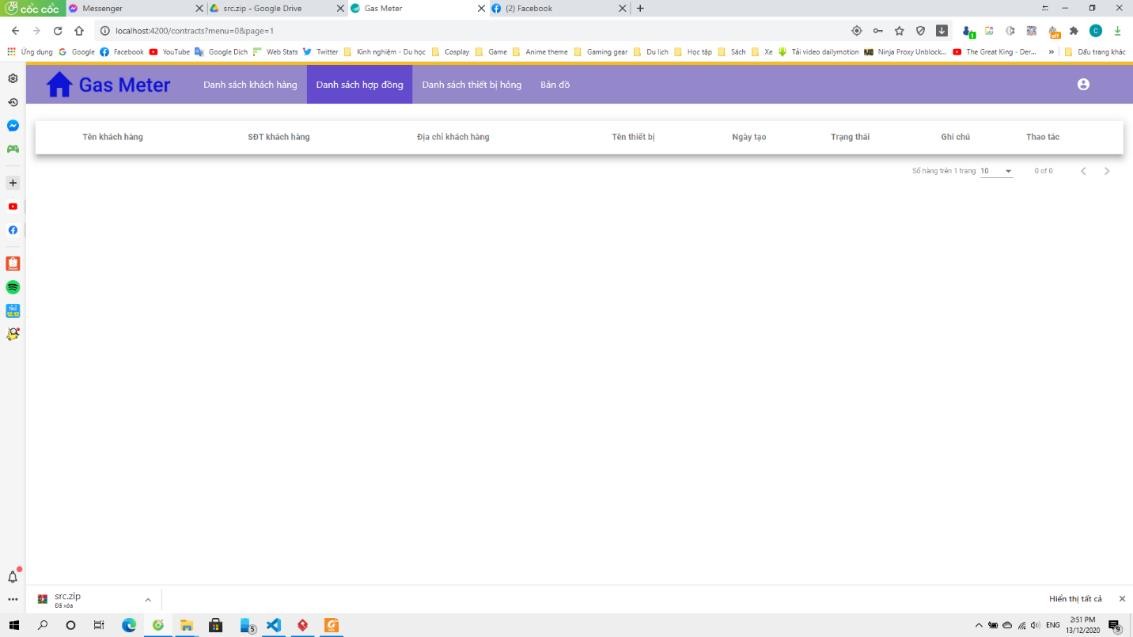
1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many other entity class. |
| 2 | ContractGasmeterResponse |  | Contract response data send from server. |
| 3 | ContractGasMeter |  | ContractGasMeter class which is related to view contract process. |
| 4 | ContractGasMeterRepository | findByGasMeterCode(String code): ContractGasMeter | Method to find contract by gasMeterCode. |
| findAllByStoreID(long storeID): List | Method to find contract by storeID. |
| findByUserID (long ID): ContractGasMeter | Method to find contract by userID. |
| 5 | ContractGasMeterRequest |  | Contract request data from client. |
| 6 | ContractGasMeterController | createContractGasMeter(contractGasMeter Request): Response | Method to create Contract. |
| getAllContractByStoreID(long id): List | Method to get list Contract by storeID. |
| updateContractGasMeter(id, contractGasMeterRequest): Response | Method to update data of Contract. |
| getAllContract(): List | Method to get All Contract. |
| 7 | ContractGasMeterImpl | getAllContractGasMeter(): List | Method to get All Contract from database. |
| getAllContractGasMeterByStoreID(): List | Method to get list Contract by storeID from database. |
| getOneContractGasMeter(id): ContractGasMeter | Method to get One Contract by id from database. |
| getOneContractGasMeterByUserID(id): ContractGasMeterResponse | Method to get One Contract by userID from database. |
| createContractGasMeter(contractGasMeter Request): | Method to add a new Contract to database. |
| entityToResponse(ContractGasMeter): ContractGasMeterResponse | Convert entity ContractGasMeter to response data. |
| updateContractGasMeter(id, contractGasMeterRequest): void | Update data of Contract from database. |
| findByGasMeterCode(code): ContractGasMeter | Find Contract by gas meter code. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | requestToEntity(contractGasMeterRequest): ContractGasMeter | Convert request data to entity. |
| 8 | <<interface>> ContractGasmeterService |  | Interface to define method. |
| 9 | NotFoundException |  | Exception when contract not found. |
| 10 | BadRequestException |  | Exception when request is invalid. |

1. Sequence Diagram(s)
2. Design

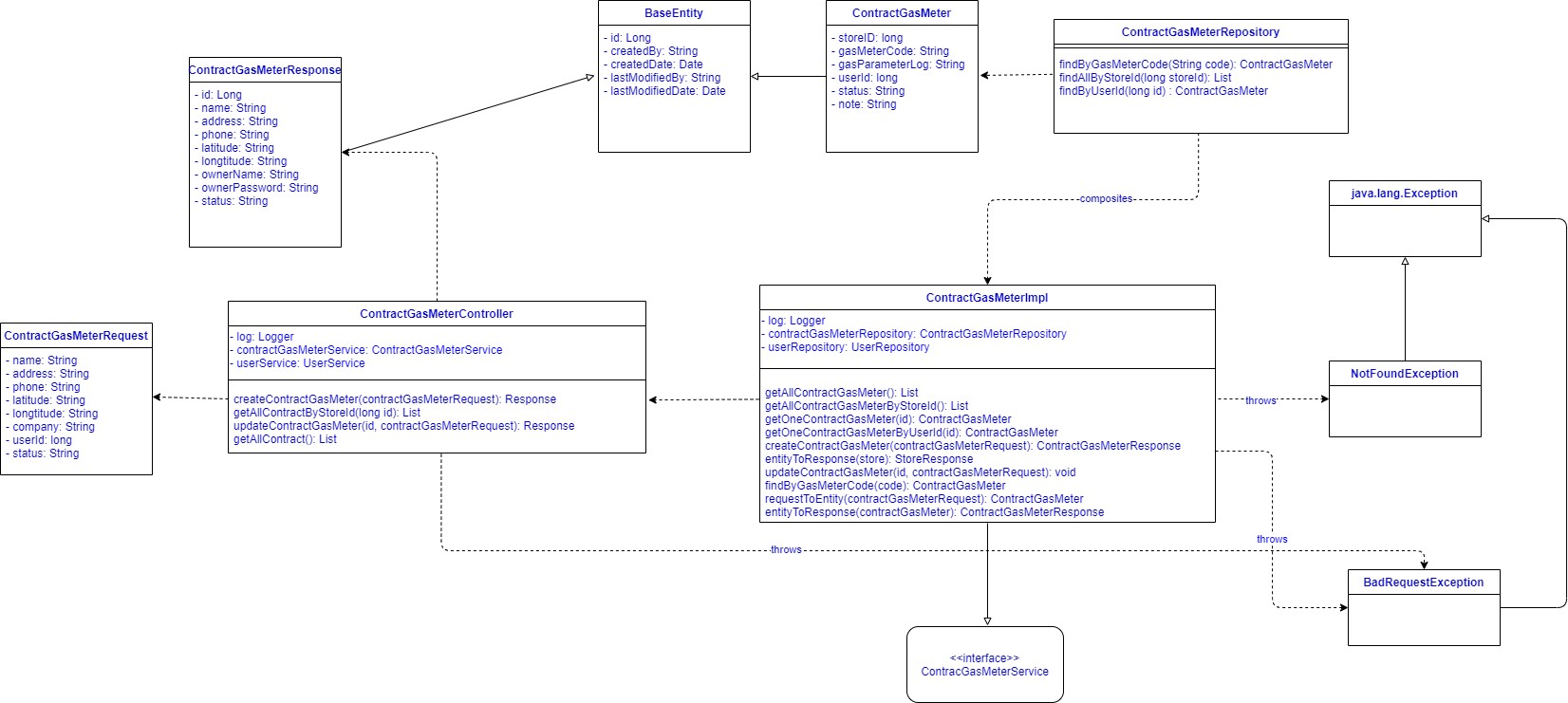
**Firgure 3.10.2:** View List Contract Sequence Diagram.



**Firgure 3.10.3:** View List Contract Screen.

* 1. Create Contract

1. Class Diagram

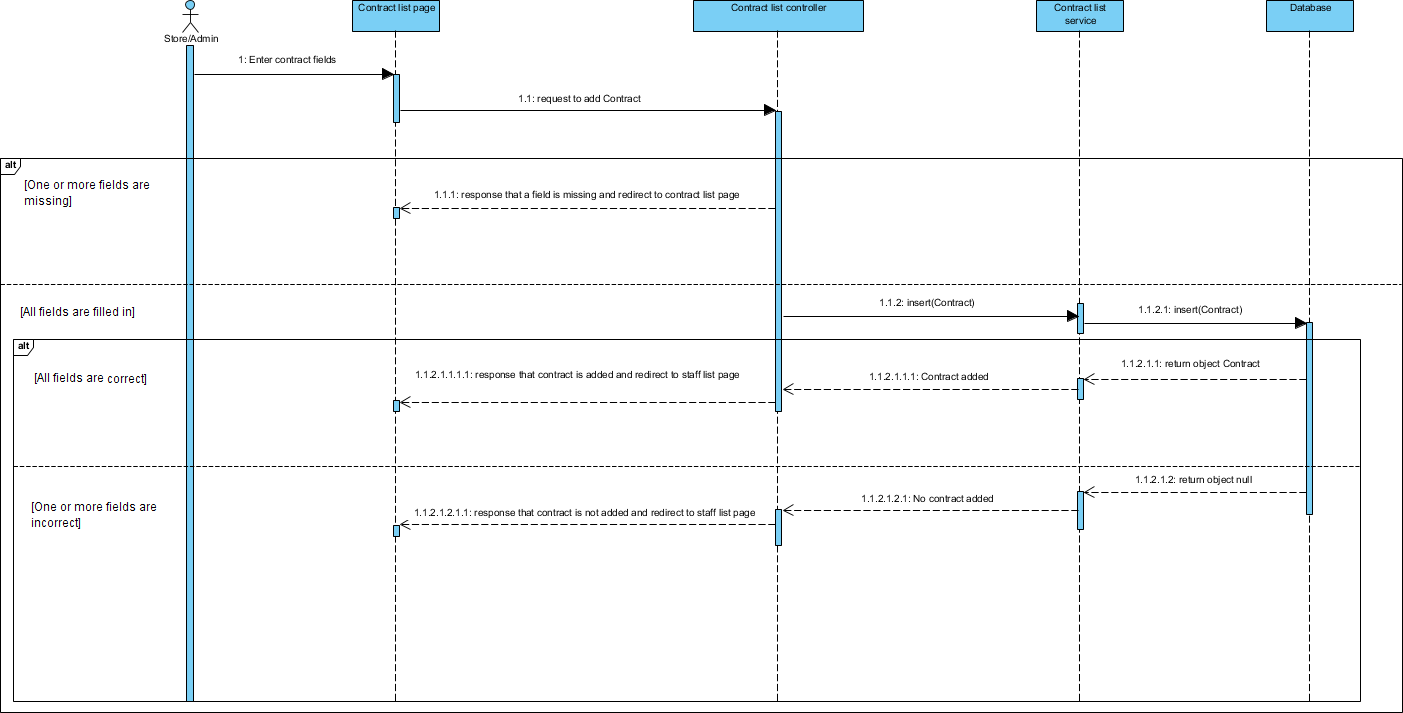


1. Class Specification

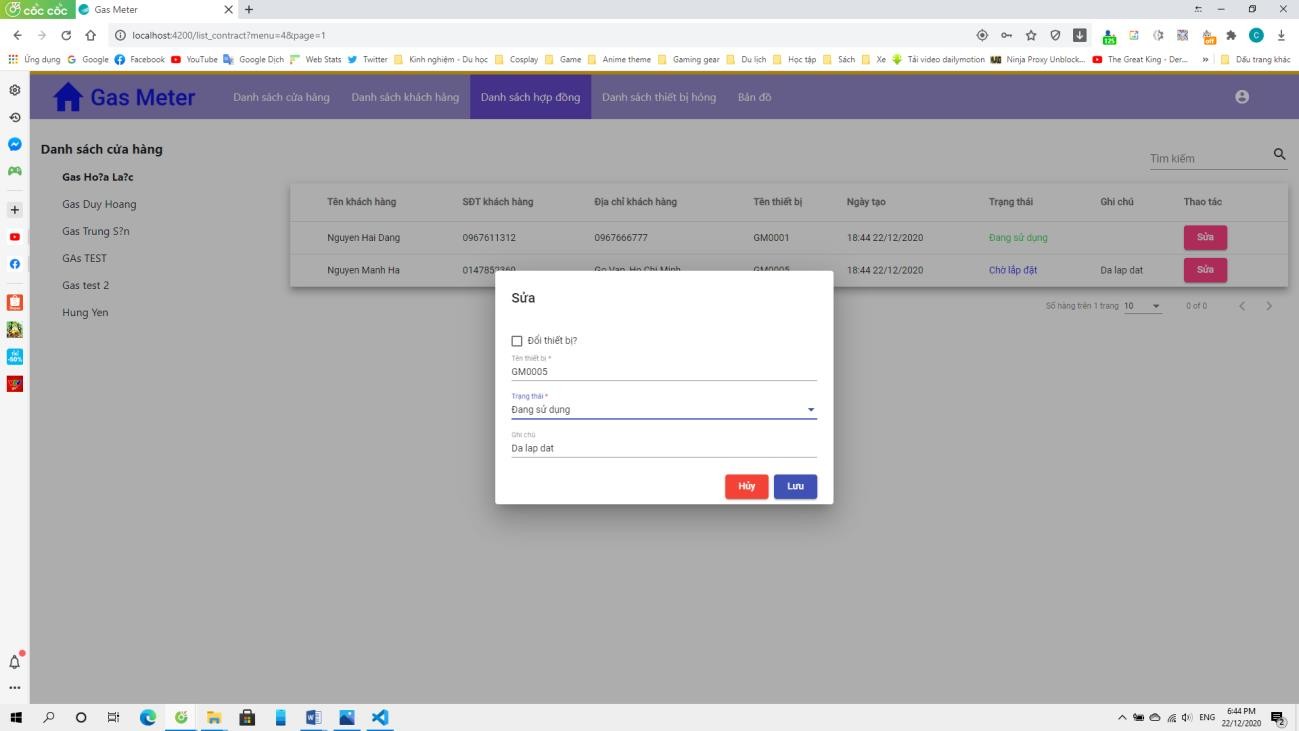
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity,  which was inherited by many other entity class |
| 2 | ContractGasmeterResponse |  | Contract response data send from server |
| 3 | ContractGasMeter |  | ContractGasMeter class, which is related to create  contract process |
| 4 | ContractGasMeterRepository | findByGasMeterCode(String code): ContractGasMeter | Method to find contract by gasMeterCode |
| findAllByStoreID(long storeID):  List | Method to find contract by  storeID |
| findByUserID(long id)  : ContractGasMeter | Method to find contract by userID |
| 5 | ContractGasMeterRequest |  | Contract request data from  client |
| 6 | ContractGasMeterController | createContractGasMeter(contractG asMeterRequest): Response | Method to create Contract |
| getAllContractByStoreID(long id):  List | Method to get list Contract by  storeID |
| updateContractGasMeter(id, contractGasMeterRequest): Response | Method to update data of Contract |
| getAllContract(): List | Method to get All Contract |
| 7 | ContractGasMeterImpl | getAllContractGasMeter(): List | Method to get All Contract from database |
| getAllContractGasMeterByStoreID (): List | Method to get list Contract by storeID from database |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | getOneContractGasMeter(id): ContractGasMeter | Method to get One Contract by id from database |
| getOneContractGasMeterByUserI D(id): ContractGasMeterResponse | Method to get One Contract by userID from database |
| createContractGasMeter(contractG asMeterRequest): | Method to add a new Contract to database |
| entityToResponse(ContractGasMet er): ContractGasMeterResponse | Convert entity ContractGasMeter to response data |
| updateContractGasMeter(id,  contractGasMeterRequest): void | Update data of Contract from  database |
| findByGasMeterCode(code): ContractGasMeter | Find Contract by gas meter code |
| requestToEntity(contractGasMeter  Request): ContractGasMeter | Convert request data to entity |
| 8 | <<interface>> ContractGasmeterService |  | Interface to define method |
| 9 | NotFoundException |  | Exception when contract not  found |
| 10 | BadRequestException |  | Exception when request is invalid |

1. Sequence Diagram(s)

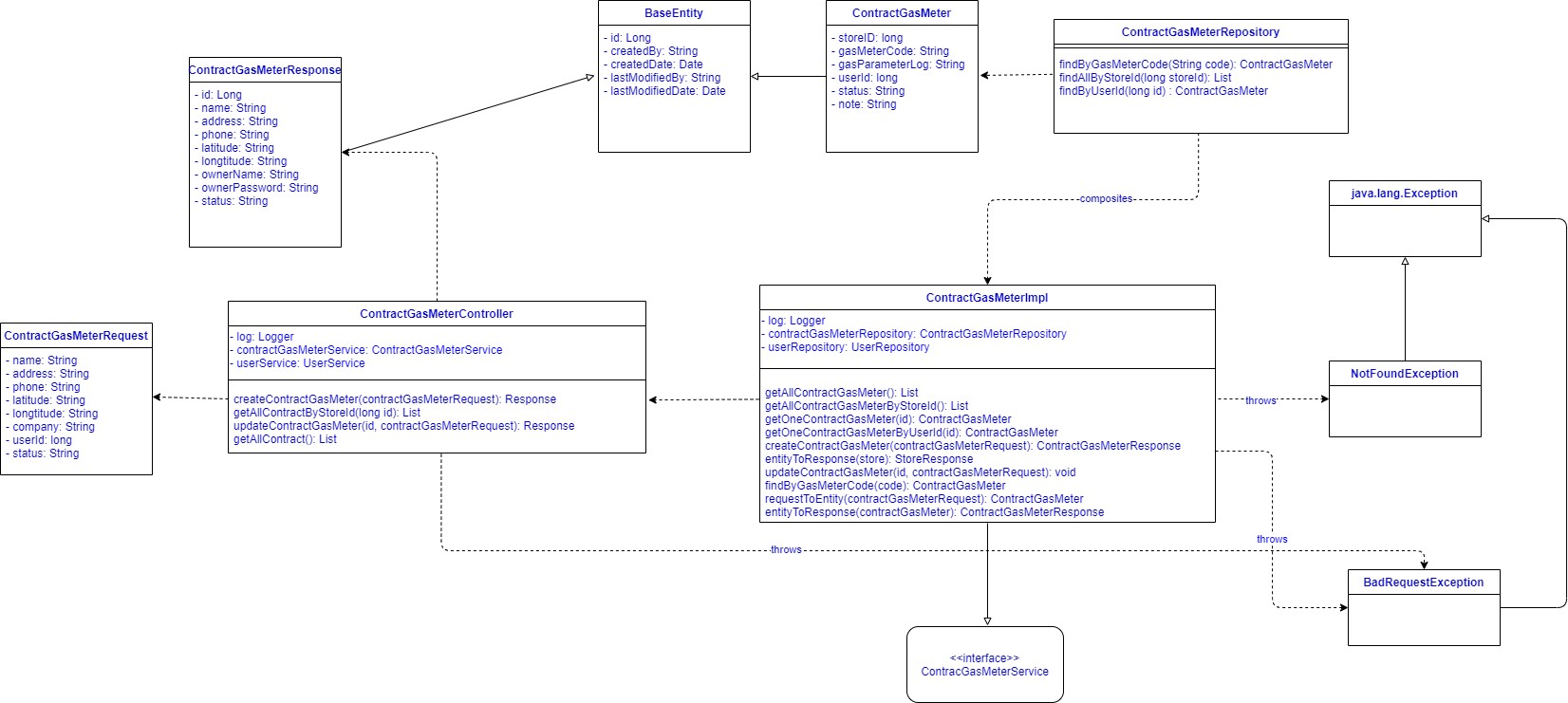


1. Design



3.14 Edit contract

1. Class Diagram



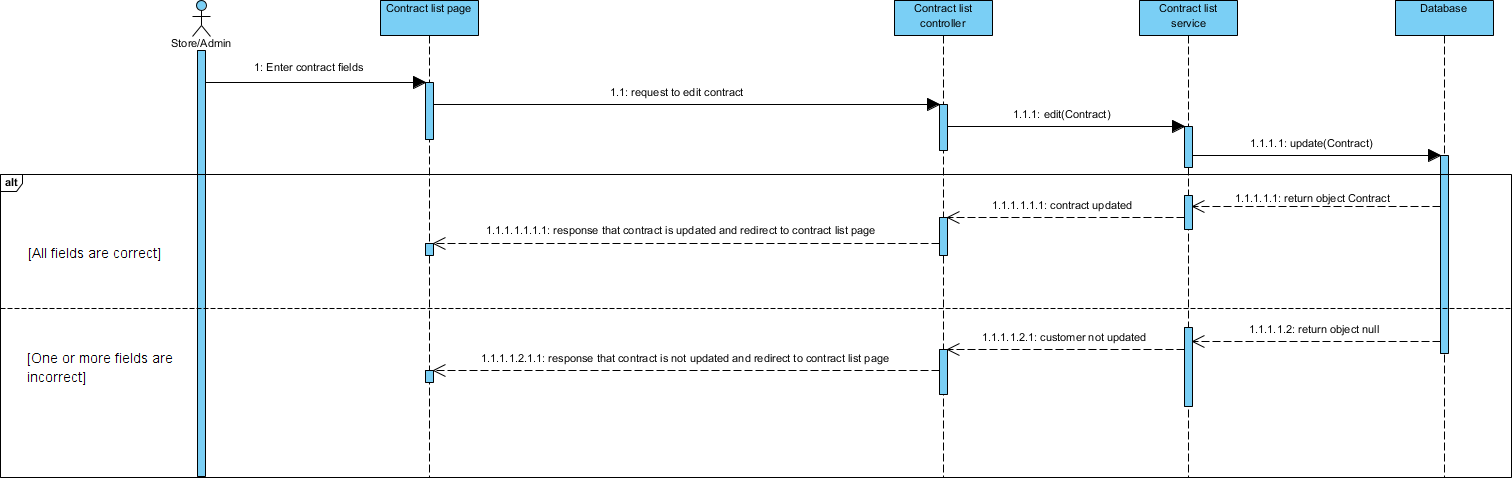
Firgure : Delete contract digram

1. Class Specification

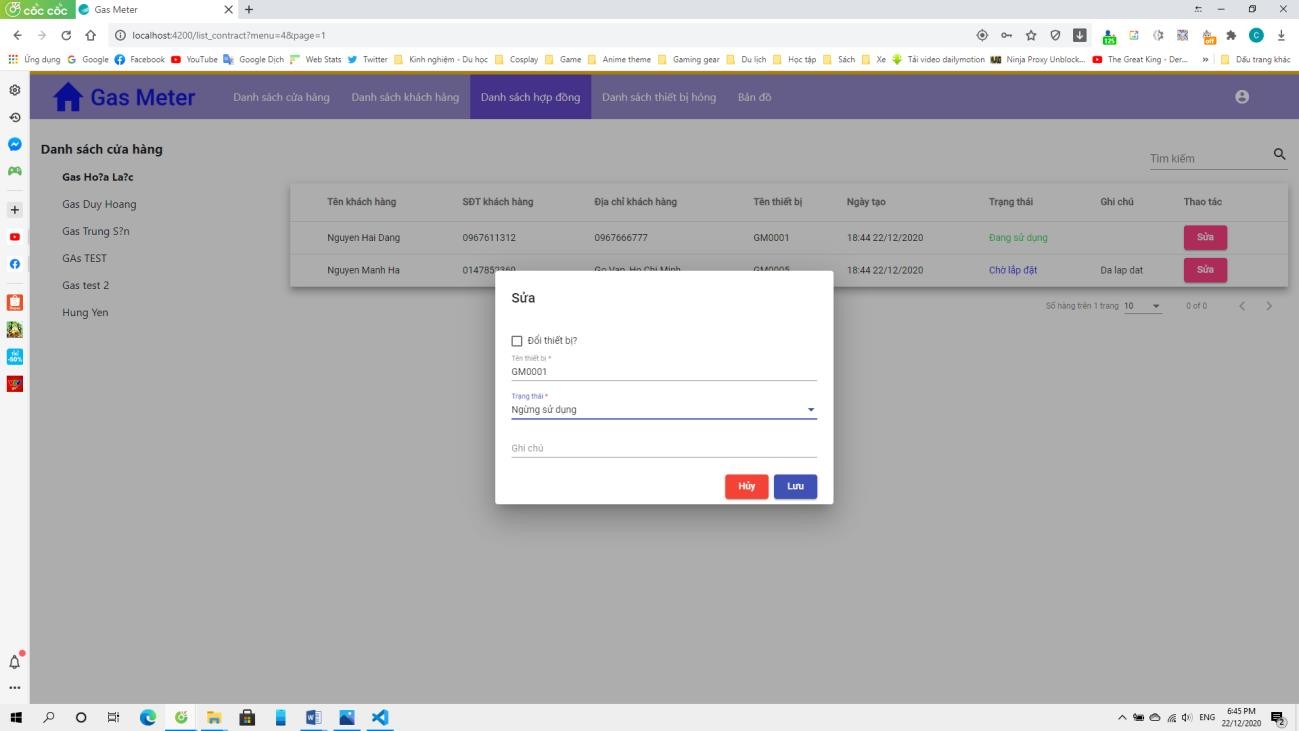
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many  other entity class |
| 2 | ContractGasmeterResponse |  | Contract response data send from server |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | ContractGasMeter |  | ContractGasMeter class, which is related to edit  contract process |
| 4 | ContractGasMeterRepository | findByGasMeterCode(String code): ContractGasMeter | Method to find contract by gasMeterCode |
| findAllByStoreID(long storeID):  List | Method to find contract by  storeID |
| findByUserID(long id)  : ContractGasMeter | Method to find contract by userID |
| 5 | ContractGasMeterRequest |  | Contract request data from  client |
| 6 | ContractGasMeterController | createContractGasMeter(contractG asMeterRequest): Response | Method to create Contract |
| getAllContractByStoreID(long id): List | Method to get list Contract by storeID |
| updateContractGasMeter(id, contractGasMeterRequest): Response | Method to update data of Contract |
| getAllContract(): List | Method to get All Contract |
| 7 | ContractGasMeterImpl | getAllContractGasMeter(): List | Method to get All Contract from database |
| getAllContractGasMeterByStoreID (): List | Method to get list Contract by storeID from database |
| getOneContractGasMeter(id): ContractGasMeter | Method to get One Contract by id from database |
| getOneContractGasMeterByUserI D(id): ContractGasMeterResponse | Method to get One Contract by userID from database |
| createContractGasMeter(contractG asMeterRequest): | Method to add a new Contract to database |
| entityToResponse(ContractGasMet er): ContractGasMeterResponse | Convert entity ContractGasMeter to response data |
| updateContractGasMeter(id,  contractGasMeterRequest): void | Update data of Contract from  database |
| findByGasMeterCode(code): ContractGasMeter | Find Contract by gas meter code |
| requestToEntity(contractGasMeter  Request): ContractGasMeter | Convert request data to entity |
| 8 | <<interface>> ContractGasmeterService |  | Interface to define method |
| 9 | NotFoundException |  | Exception when contract not  found |
| 10 | BadRequestException |  | Exception when request is invalid |

1. Sequence Diagram(s)



1. Design



* 1. Search contract

1. Class Diagram

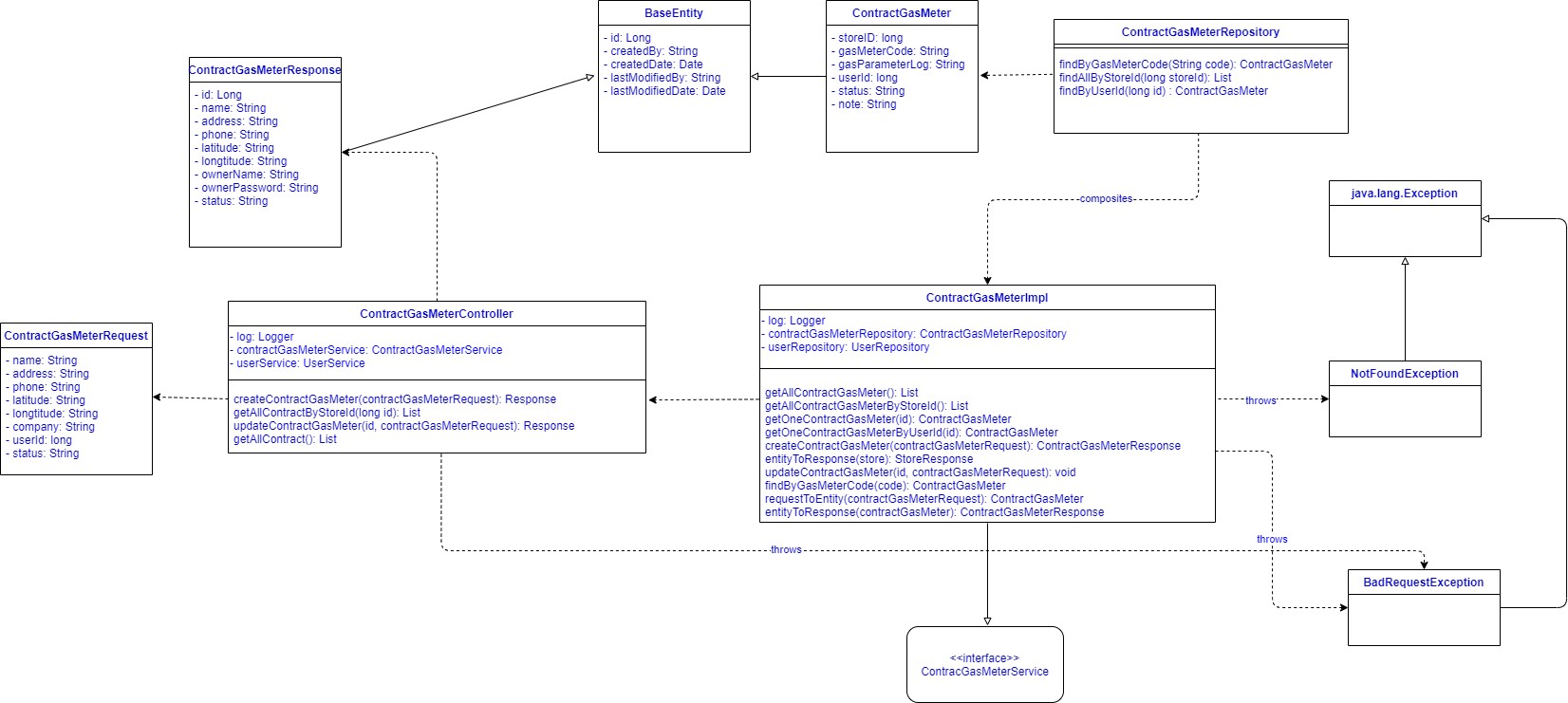


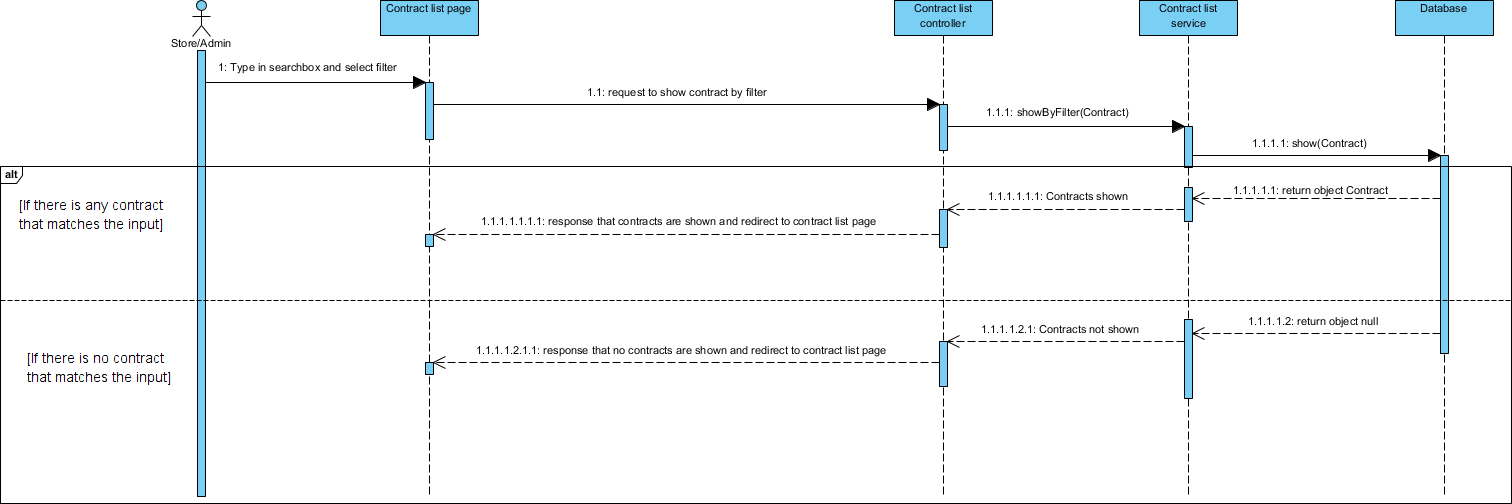
Figure : Search contract class diagram

1. Class Specification

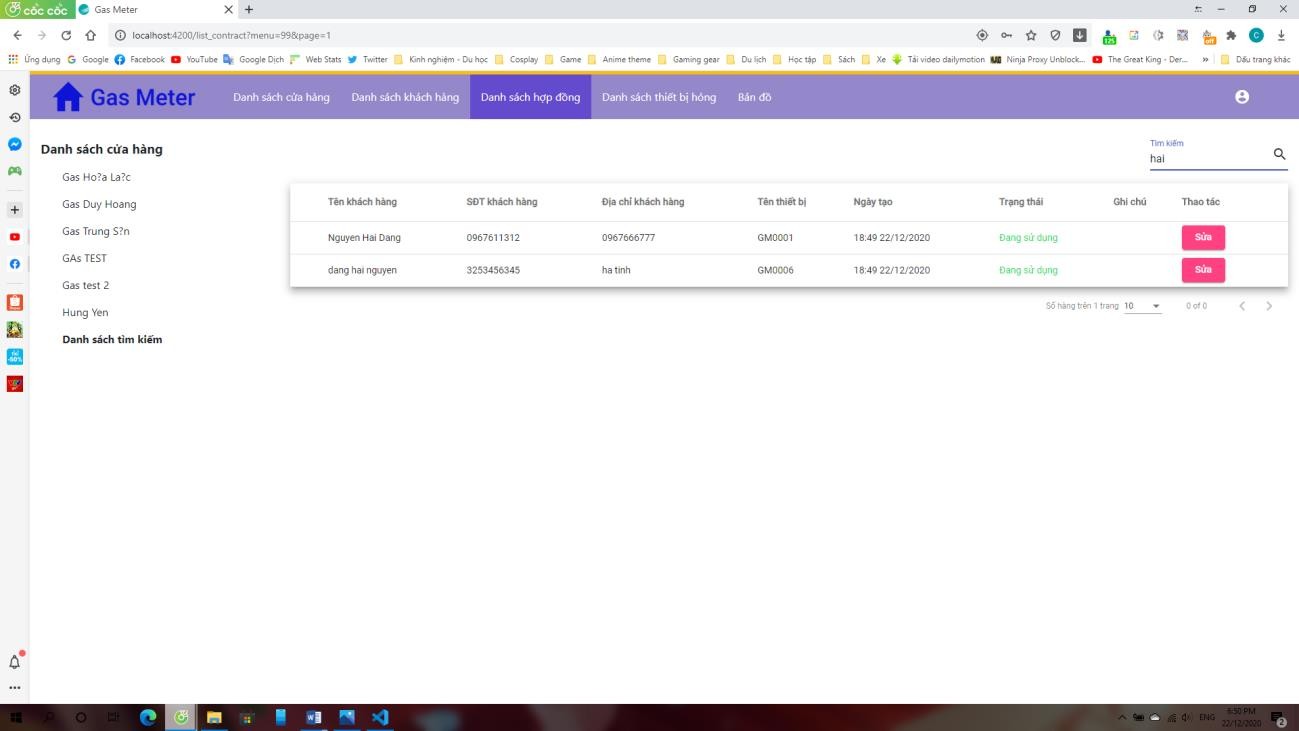
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity,  which was inherited by many other entity class |
| 2 | ContractGasmeterResponse |  | Contract response data send from server |
| 3 | ContractGasMeter |  | ContractGasMeter class, which is related to view  contract process |
| 4 | ContractGasMeterRepository | findByGasMeterCode(String code): ContractGasMeter | Method to find contract by gasMeterCode |
| findAllByStoreID(long storeID):  List | Method to find contract by  storeID |
| findByUserID(long id)  : ContractGasMeter | Method to find contract by userID |
| 5 | ContractGasMeterRequest |  | Contract request data from  client |
| 6 | ContractGasMeterController | createContractGasMeter(contractG asMeterRequest): Response | Method to create Contract |
| getAllContractByStoreID(long id):  List | Method to get list Contract by  storeID |
| updateContractGasMeter(id, contractGasMeterRequest): Response | Method to update data of Contract |
| getAllContract(): List | Method to get All Contract |
| 7 | ContractGasMeterImpl | getAllContractGasMeter(): List | Method to get All Contract from database |
| getAllContractGasMeterByStoreID (): List | Method to get list Contract by storeID from database |
| getOneContractGasMeter(id): ContractGasMeter | Method to get One Contract by id from database |
| getOneContractGasMeterByUserI D(id): ContractGasMeterResponse | Method to get One Contract by userID from database |
| createContractGasMeter(contractG asMeterRequest): | Method to add a new Contract to database |
| entityToResponse(ContractGasMet er): ContractGasMeterResponse | Convert entity ContractGasMeter to response data |
| updateContractGasMeter(id,  contractGasMeterRequest): void | Update data of Contract from  database |
| findByGasMeterCode(code): ContractGasMeter | Find Contract by gas meter code |
| requestToEntity(contractGasMeter  Request): ContractGasMeter | Convert request data to entity |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | <<interface>> ContractGasmeterService |  | Interface to define method |
| 9 | NotFoundException |  | Exception when contract not found |
| 10 | BadRequestException |  | Exception when request is invalid |

1. Sequence Diagram(s)

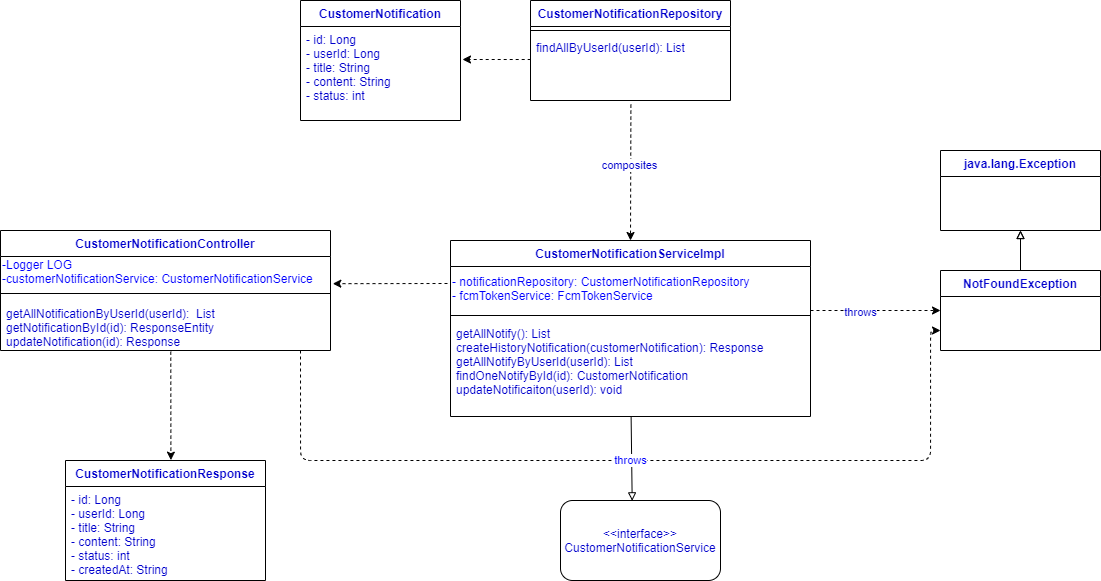


1. Design



* 1. View Notification

1. Class Diagram

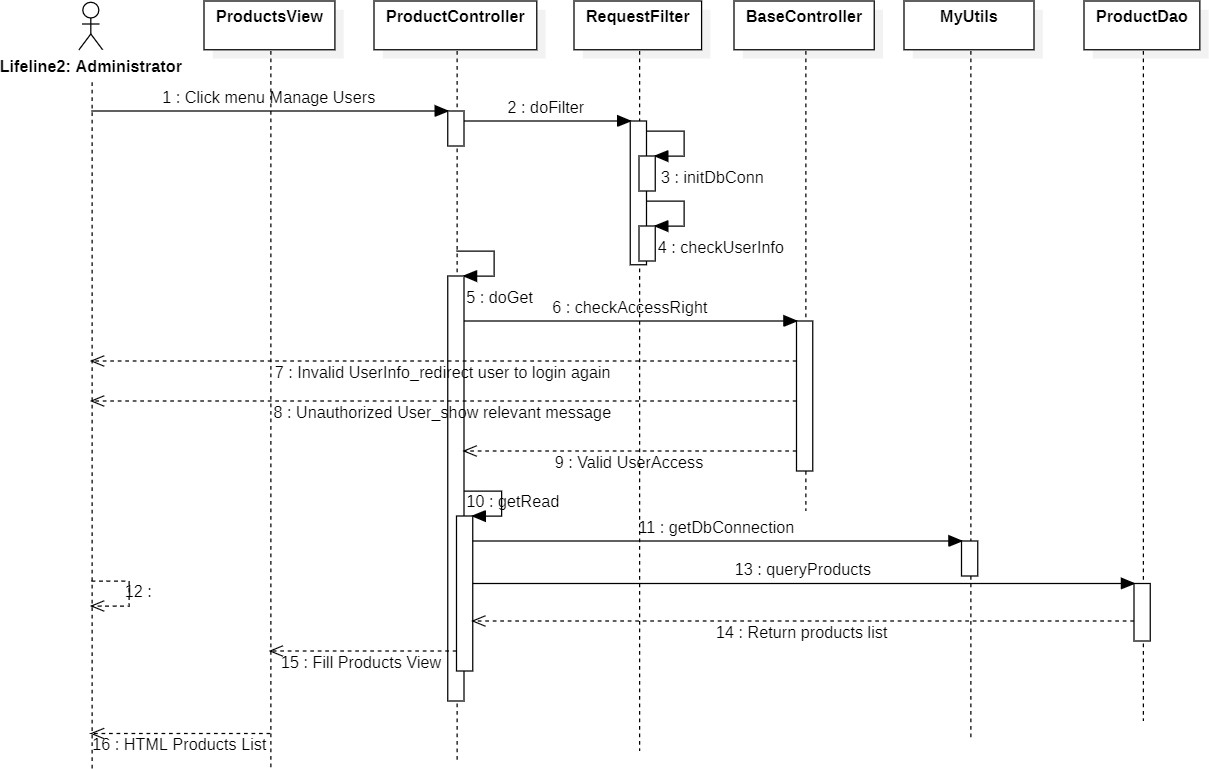


1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | CustomerNotification |  | CustomerNotification class which is related to view customer notifications |
| 2 | CustomerNotification Repository | findAllByUserId(userID): List | Method to find all notifications by userID |

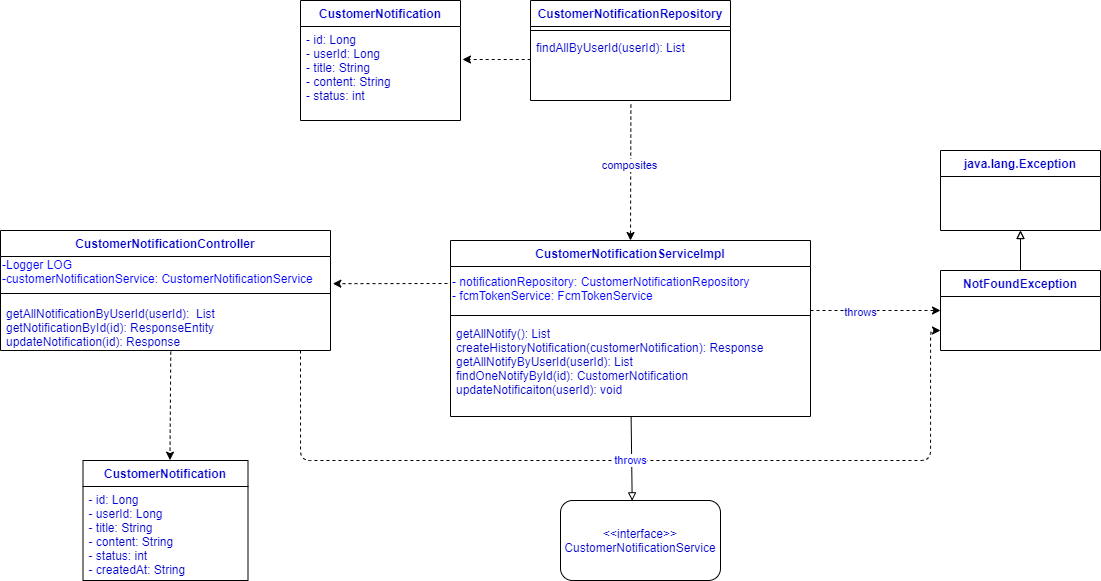
|  |  |  |  |
| --- | --- | --- | --- |
| 3 | CustomerNotificationRes ponse |  | Response data from server |
| 4 | CustomerNotificationCon troller | getAllNotificationByUserId(userId): List | Method to get notifications by userID. |
| getNotificationById(id): ResponseEntity | Method to get notifications by id |
| updateNotification(id): Response | Update notifications by id |
| 5 | CustomerNotificationServ iceImpl | getAllNotify(): List | Method to get all notifications from database. |
| createHistoryNotification(customerNoti fication): Response | Method to generate history notifications. |
| getAllNotifyByUserId(userId): List | Method to get all notifications from database  by userID |
| findOneNotifyById(id): CustomerNotification | Method to get a notification by userID from database. |
| updateNotificaiton(userId): void | Method to update notifications by userID |
| 6 | <<interface>>  CustomerNotificationServ ice |  | Interface to define method. |
| 7 | NotFoundException |  | Exception when contract not found. |

1. Sequence Diagram(s)



* 1. Create Notification

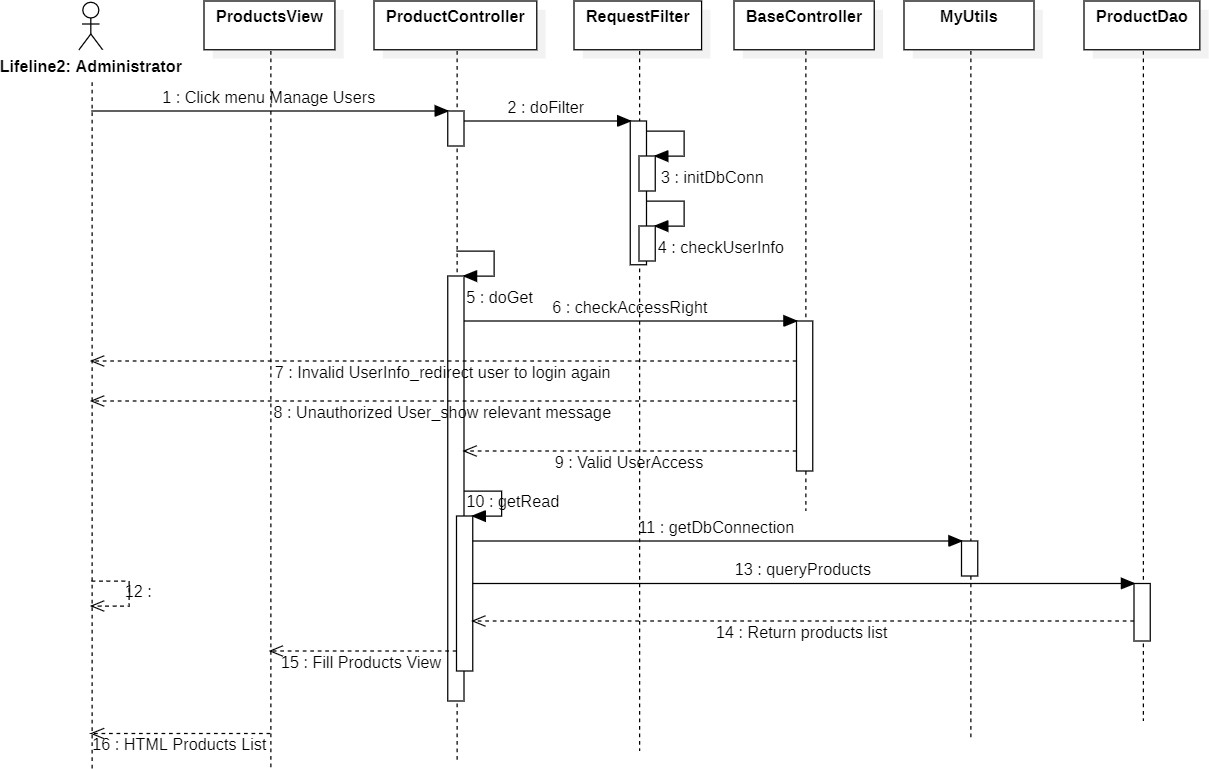
1. Class Diagram



1. Class Specification

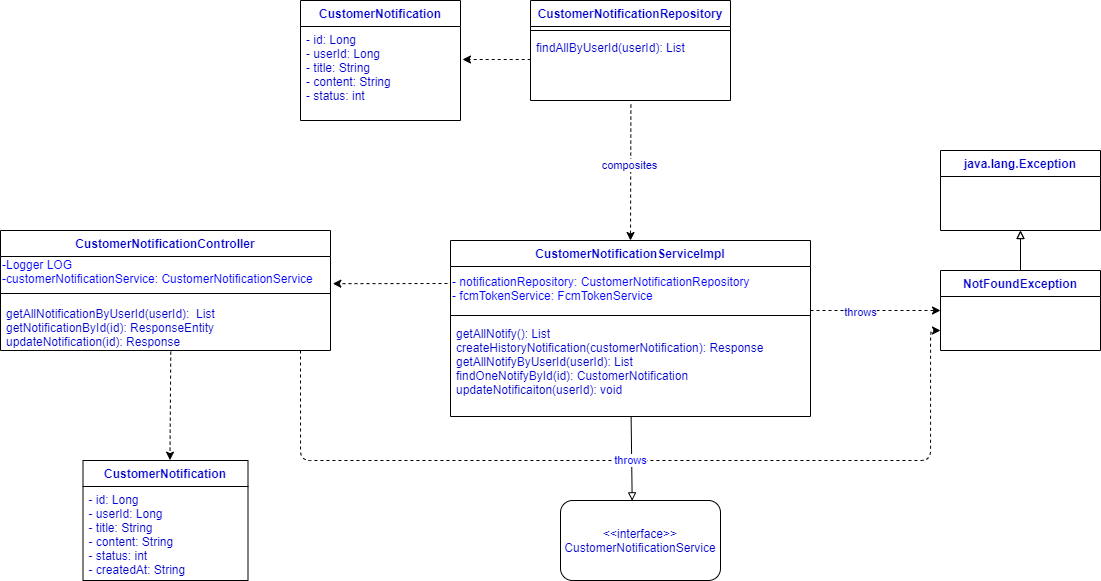
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | CustomerNotification |  | CustomerNotification class which is related to view customer notifications |
| 2 | CustomerNotification Repository | findAllByUserId(userID): List | Method to find all notifications by userID |
| 3 | CustomerNotificationRes ponse |  | Response data from server |
| 4 | CustomerNotificationCon troller | getAllNotificationByUserId(userId): List | Method to get notifications by userID. |
| getNotificationById(id): ResponseEntity | Method to get notifications by id |
| updateNotification(id): Response | Update notifications by id |
| 5 | CustomerNotificationServ iceImpl | getAllNotify(): List | Method to get all notifications from database. |
| createHistoryNotification(customerNoti fication): Response | Method to generate history notifications. |
| getAllNotifyByUserId(userId): List | Method to get all notifications from database  by userID |
| findOneNotifyById(id): CustomerNotification | Method to get a notification by userID from database. |
| updateNotificaiton(userId): void | Method to update notifications by userID |
| 6 | <<interface>>  CustomerNotificationServ ice |  | Interface to define method. |
| 7 | NotFoundException |  | Exception when contract not found. |

1. Sequence Diagram(s)



* 1. Delete Notification

1. Class Diagram

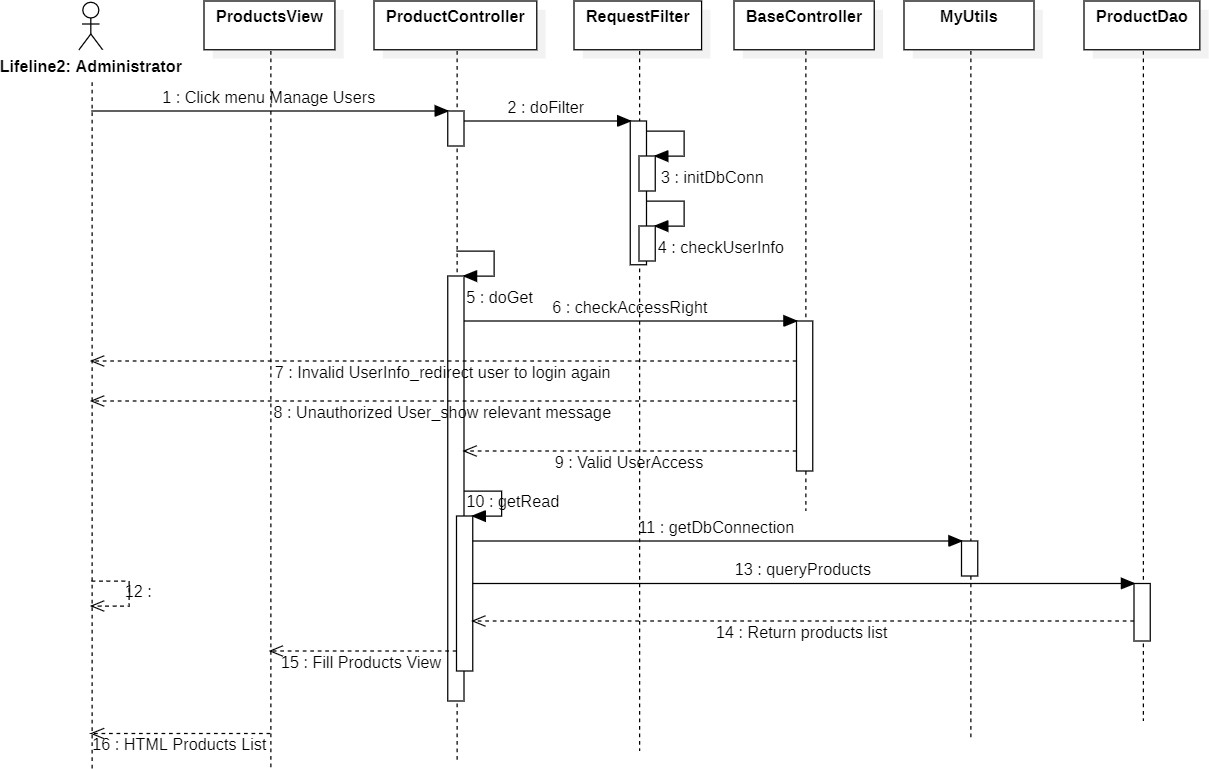


1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |

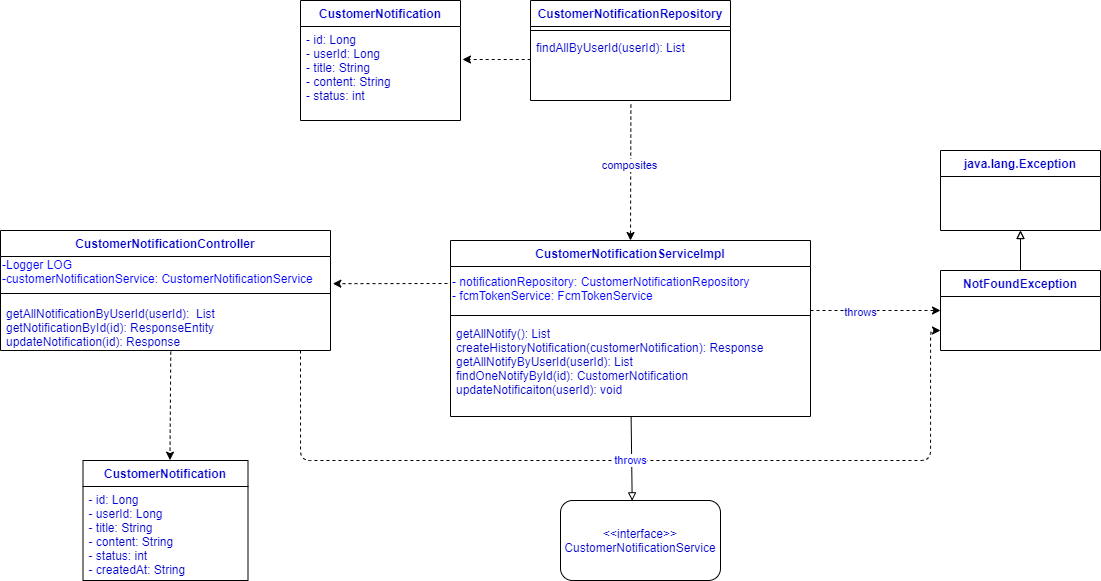
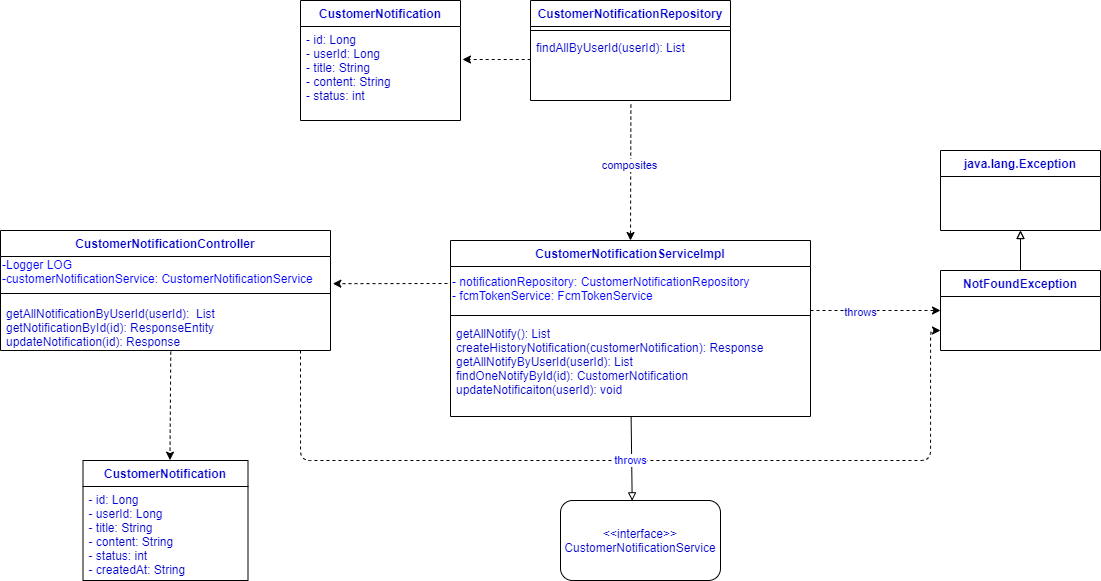
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | CustomerNotification |  | CustomerNotification class which is related to view customer notifications |
| 2 | CustomerNotification Repository | findAllByUserId(userID): List | Method to find all notifications by userID |
| 3 | CustomerNotificationRes ponse |  | Response data from server |
| 4 | CustomerNotificationCon troller | getAllNotificationByUserId(userId): List | Method to get notifications by userID. |
| getNotificationById(id): ResponseEntity | Method to get notifications by id |
| updateNotification(id): Response | Update notifications by id |
| 5 | CustomerNotificationServ iceImpl | getAllNotify(): List | Method to get all notifications from database. |
| createHistoryNotification(customerNoti fication): Response | Method to generate history notifications. |
| getAllNotifyByUserId(userId): List | Method to get all notifications from database  by userID |
| findOneNotifyById(id): CustomerNotification | Method to get a notification by userID from database. |
| updateNotificaiton(userId): void | Method to update notifications by userID |
| 6 | <<interface>> CustomerNotificationServ ice |  | Interface to define method. |
| 7 | NotFoundException |  | Exception when contract not found. |

1. Sequence Diagram(s)



* 1. Search Notification

1. Class Diagram

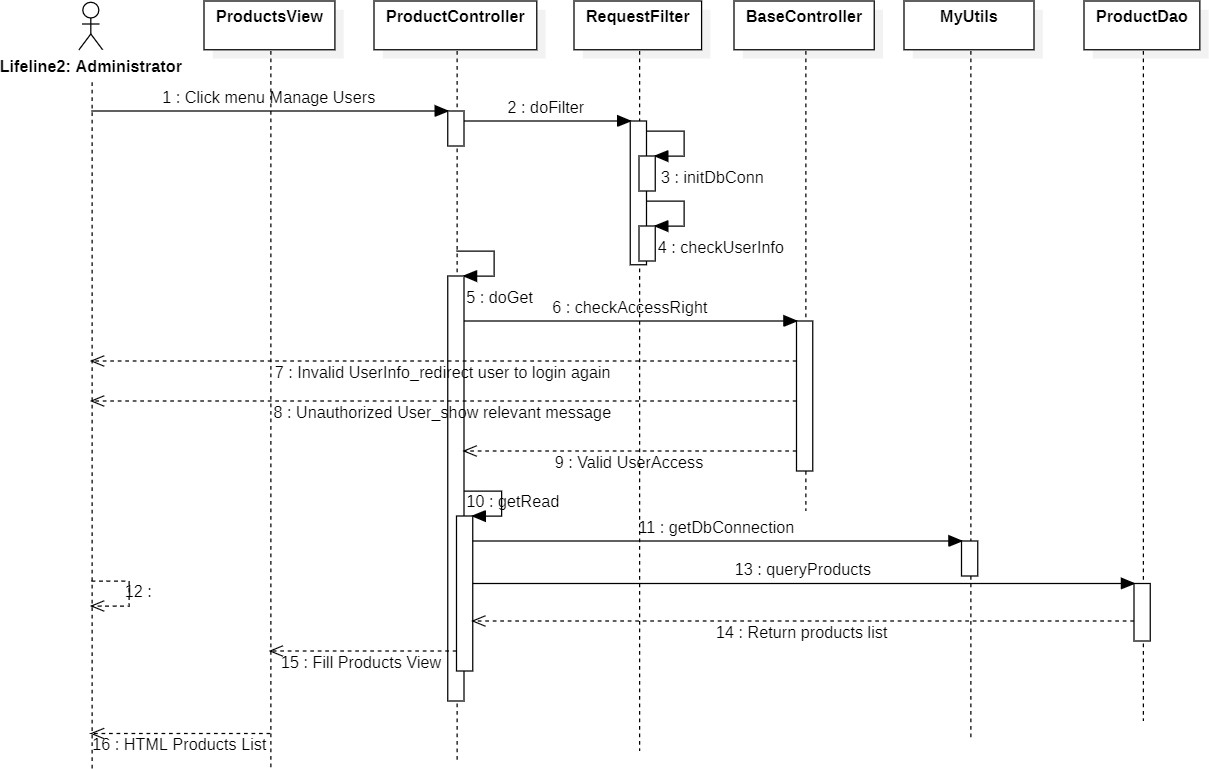


1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | CustomerNotification |  | CustomerNotification class which is related to view customer notifications |
| 2 | CustomerNotification Repository | findAllByUserId(userID): List | Method to find all notifications by userID |
| 3 | CustomerNotificationRes ponse |  | Response data from server |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | CustomerNotificationCon troller | getAllNotificationByUserId(userId): List | Method to get notifications by userID. |
| getNotificationById(id): ResponseEntity | Method to get notifications by id |
| updateNotification(id): Response | Update notifications by id |
| 5 | CustomerNotificationServ iceImpl | getAllNotify(): List | Method to get all notifications from database. |
| createHistoryNotification(customerNoti fication): Response | Method to generate history notifications. |
| getAllNotifyByUserId(userId): List | Method to get all  notifications from database by userID |
| findOneNotifyById(id): CustomerNotification | Method to get a notification by userID from database. |
| updateNotificaiton(userId): void | Method to update notifications by userID |
| 6 | <<interface>> CustomerNotificationServ ice |  | Interface to define method. |
| 7 | NotFoundException |  | Exception when contract not found. |

1. Sequence Diagram(s)



* 1. View list Store

1. Class Diagram

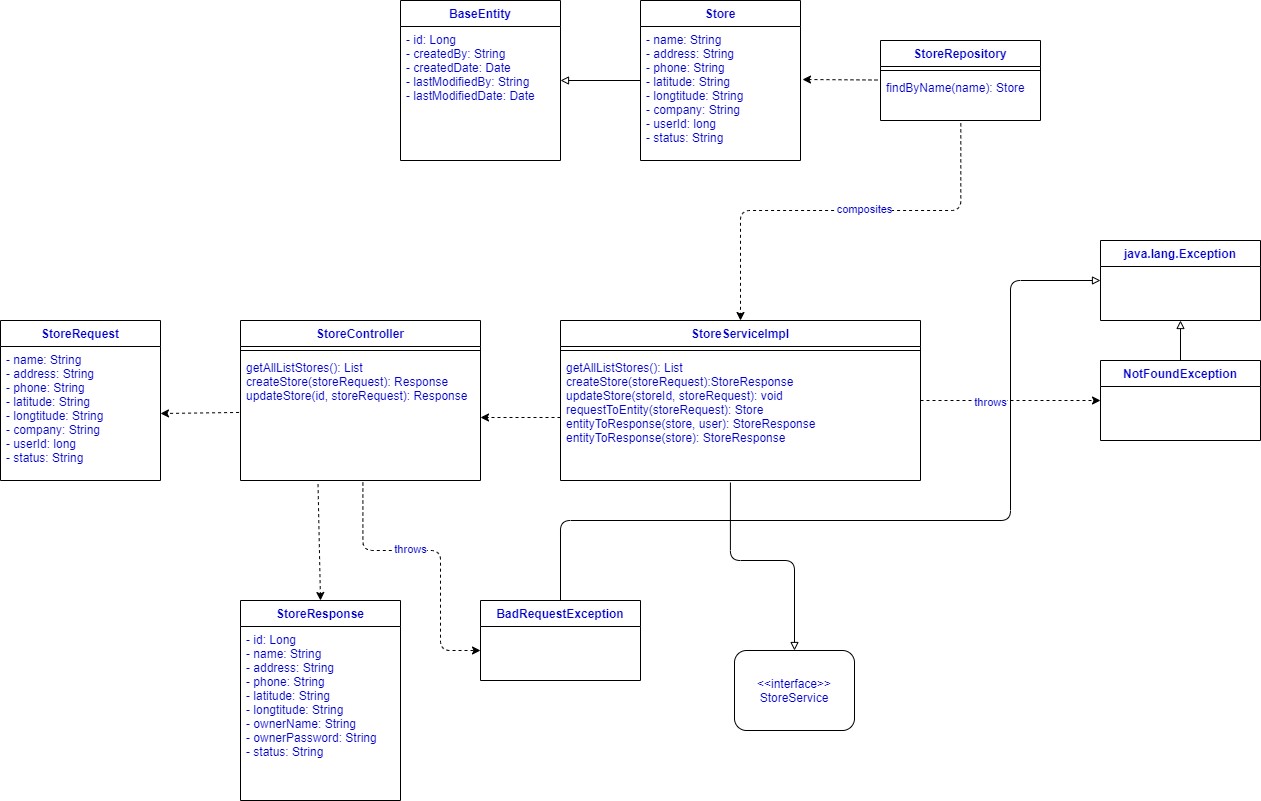
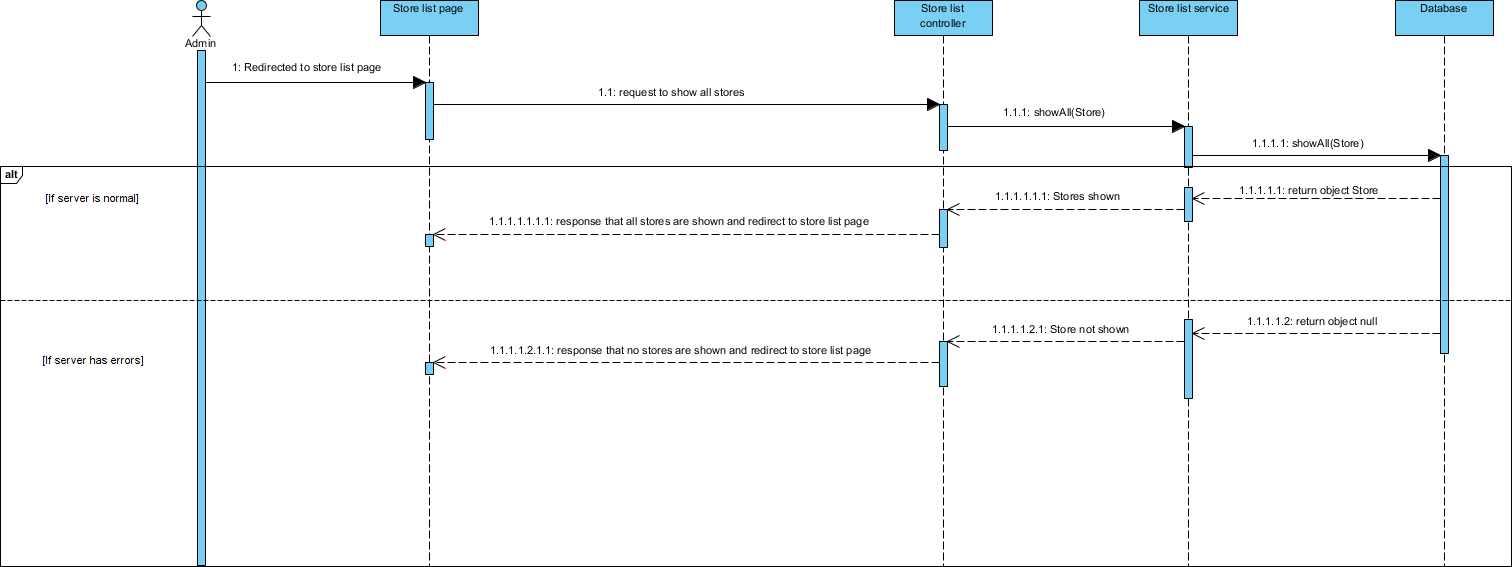


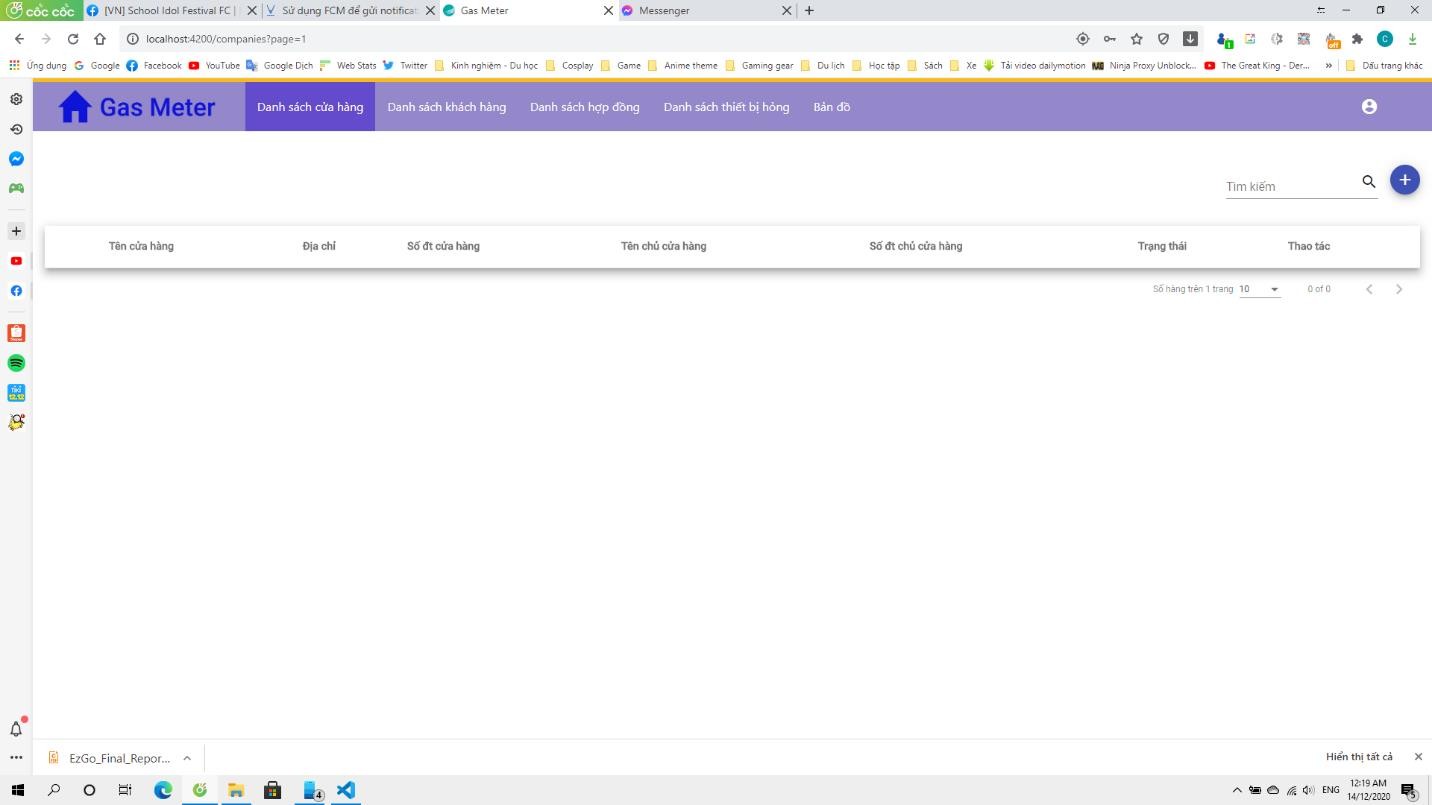
Figure : View list-store class diagram

1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many  other entity class |
| 2 | Store |  | Store class, which is related to view Store process |
| 3 | StoreRepository |  | This class define method  interact with database |
| 4 | StoreRequest |  | Request data from client |
| 5 | StoreController |  | to navigate when the API has called |
| 6 | StoreServiceIpml | getAllListStore():List | Function to get all store |
|  | entityToResponse(store,user): StoreResponse | Convert Store entity to  response data(attach user name and phone number) |
| entityToResponse(store): StoreResponse | Convert Store entity to response data |
| 7 | StoreResponse |  | Response data from server |
| 8 | <<interface>> StoreService |  | Interface to define function |
| 9 | NotFoundException |  | Exception when store not found |
| 10 | BadRequestException |  | Exception when Request is  invalid |

1. Sequence Diagram(s)
2. Design



* 1. Create Store

1. Class Diagram

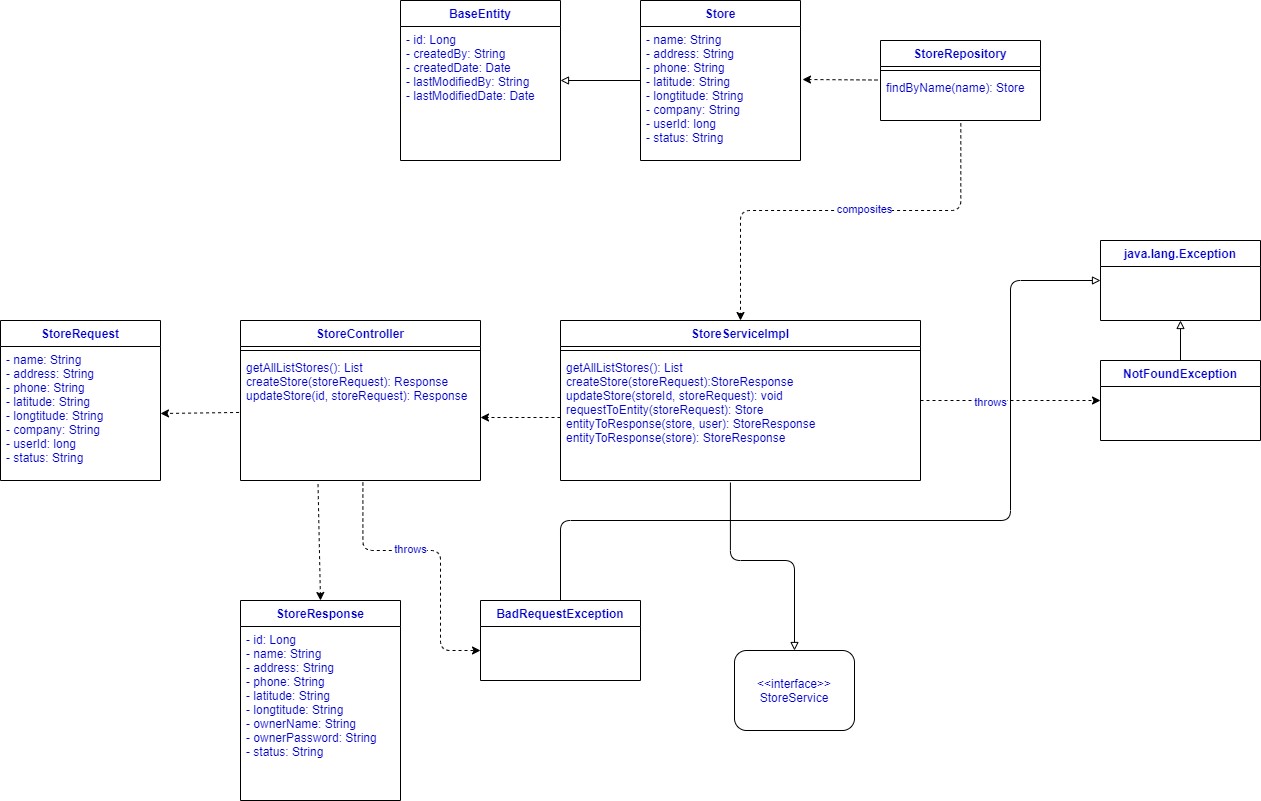
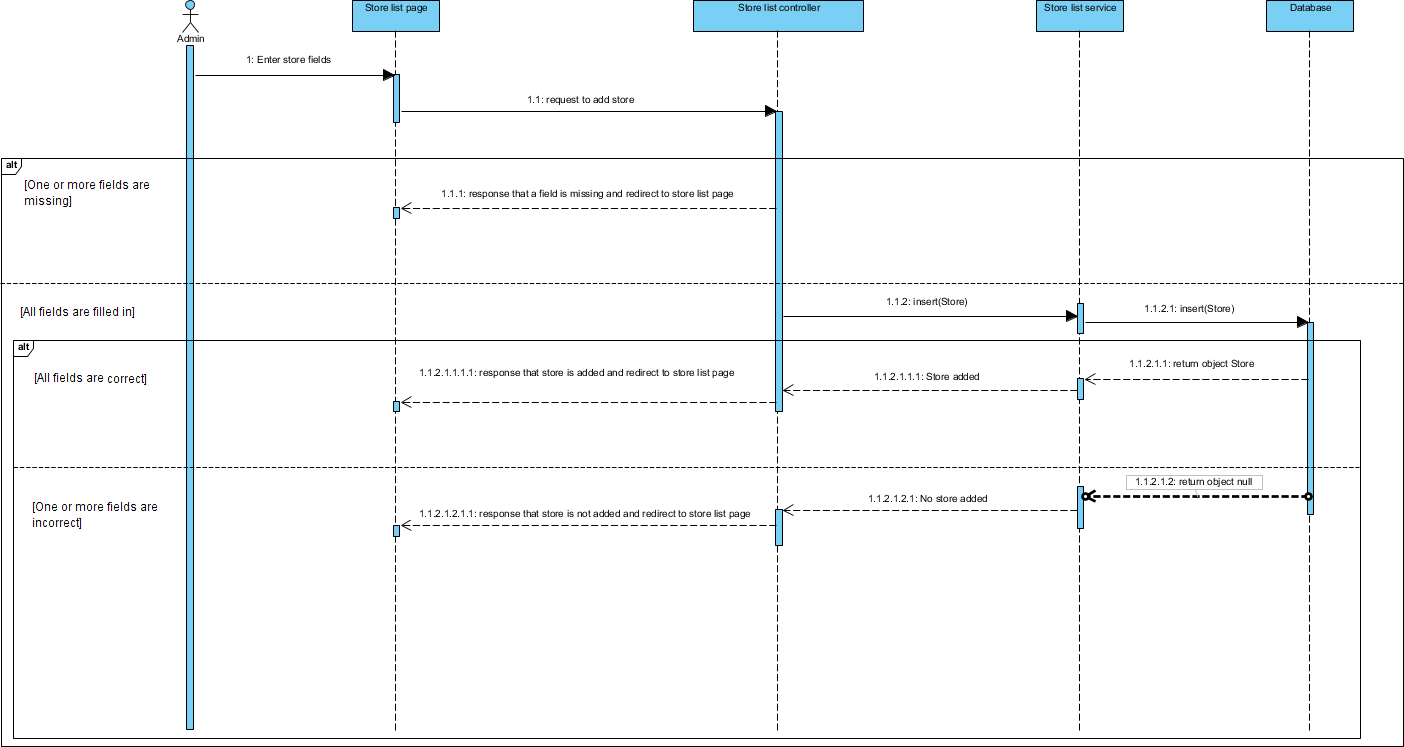


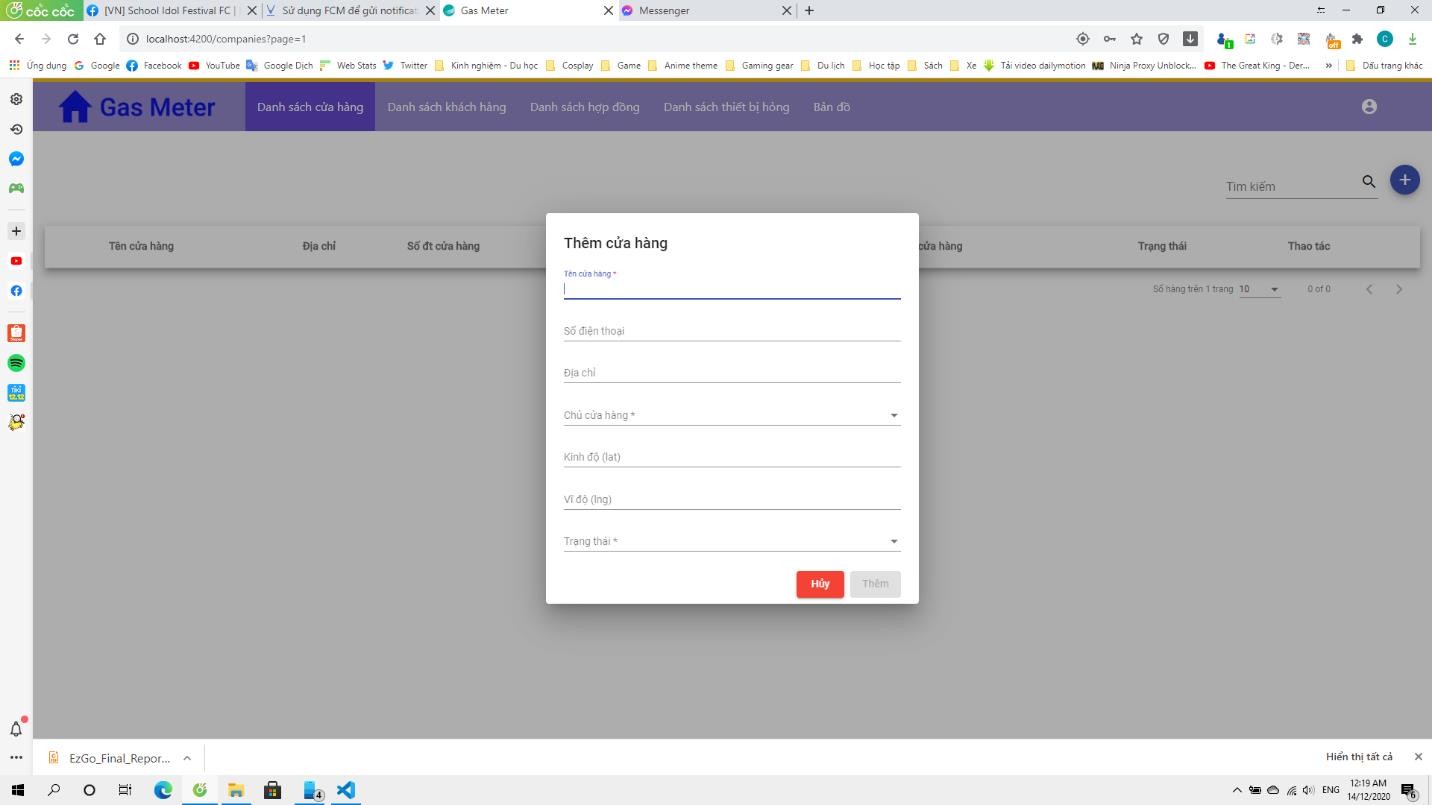
Figure : Create Store class diagram

1. Class Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | BaseEntity |  | The default class of entity, which was inherited by many  other entity class |
| 2 | Store |  | Store class, which is related to view Store process |
| 3 | StoreRepository |  | This class define method  interact with database |
| 4 | StoreRequest |  | Request data from client |
| 5 | StoreController | createStore(storeRequest): response | to navigate when the API has called |
| 6 | StoreServiceIpml | createStore(StoreRequest): StoreResponse | Function to create new store |
| requestToEntity(StoreRequest): Store | Convert request data to entity |
| 7 | StoreResponse |  | Response data from server |
| 8 | <<interface>> StoreService |  | Interface to define function |
| 9 | NotFoundException |  | Exception when store not found |
| 10 | BadRequestException |  | Exception when Request is invalid |

1. Sequence Diagram(s)
2. Design:



* 1. Update Store

1. Class Diagram

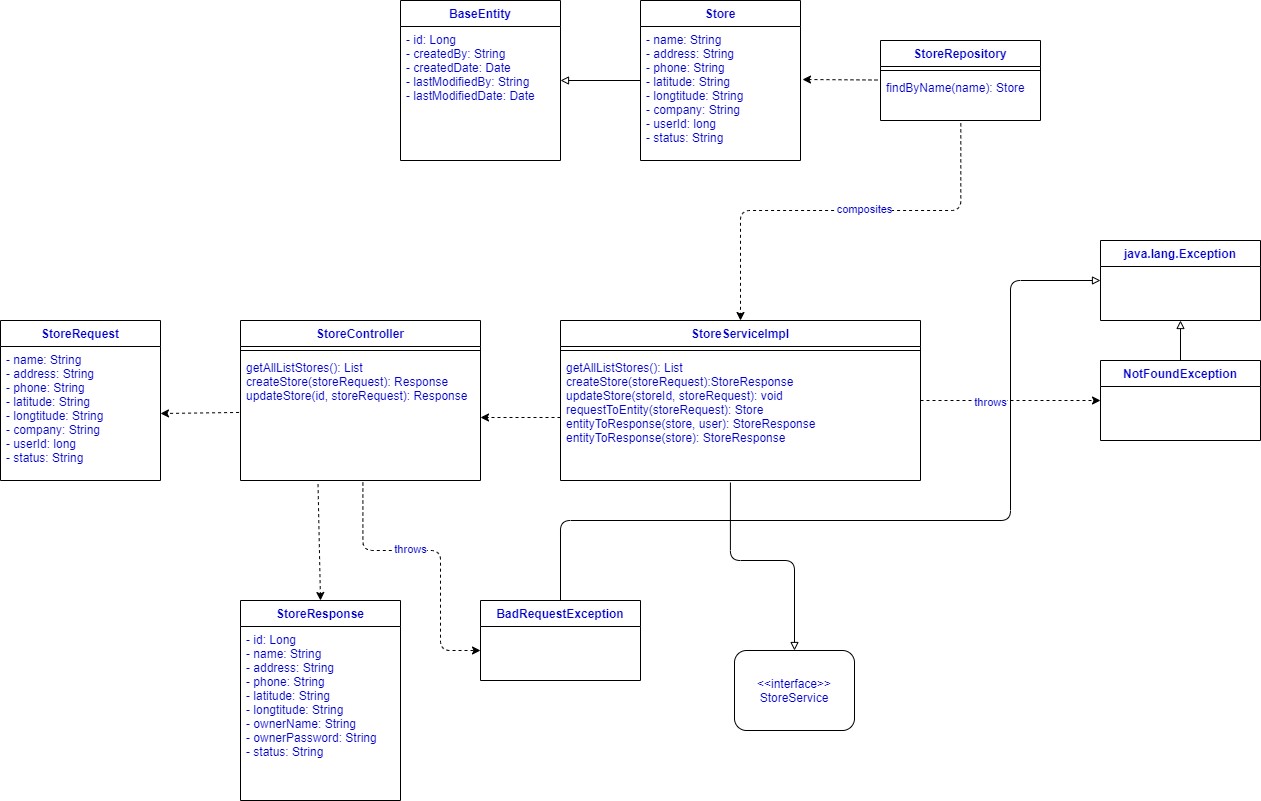
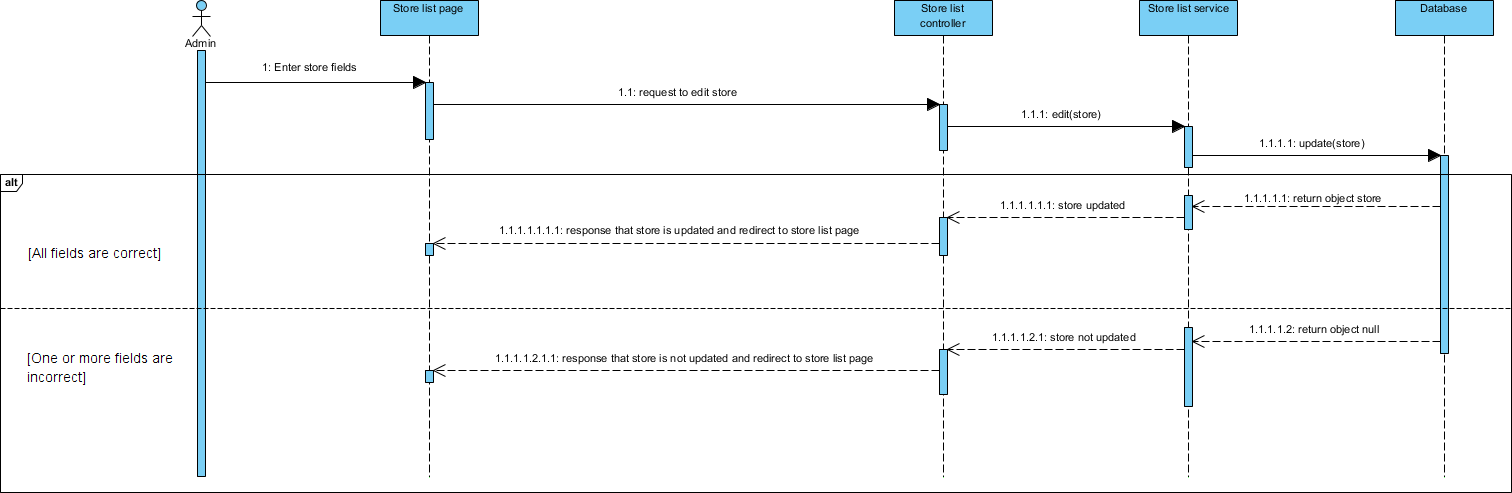


Figure : Update Store class diagram

1. Class Specification

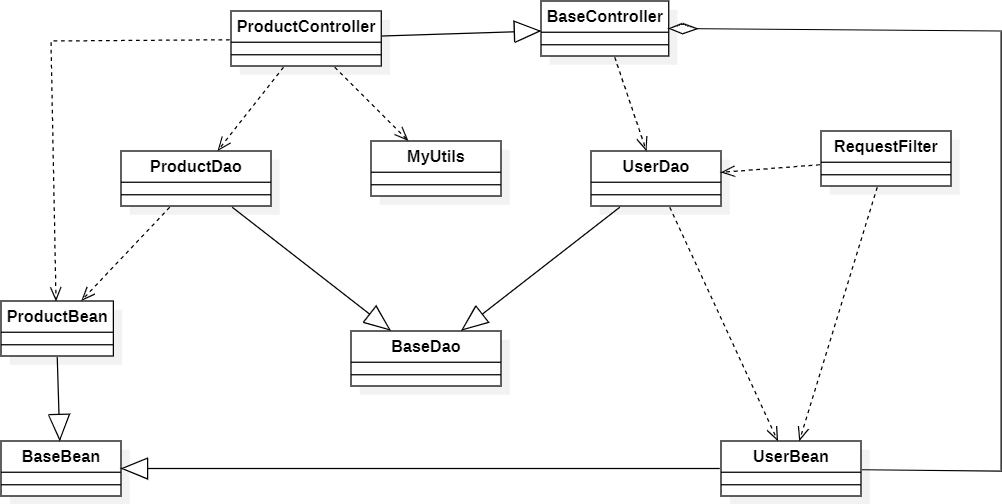
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Class Name** | **Method** | **Description** |
| 1 | BaseEntity |  | The default class of entity,  which was inherited by many other entity class |
| 2 | Store |  | Store class, which is related to view Store process |
| 3 | StoreRepository | FindByName(name): store | This class define method interact with database |
| 4 | StoreRequest | StoreController | Request data from client |
| 5 | StoreController | updateStore(id,StoreRequest): Response | to navigate when the API has called |
| 6 | StoreServiceIpml | updateStore(StoreID,  StoreRequest): void | Function to update Store |
| requestToEntity(StoreRequest): Store | Convert Request data to entity |
| 7 | StoreResponse |  | Response data from server |
| 8 | <<interface>> StoreService |  | Interface to define function |
| 9 | NotFoundException |  | Exception when store not found |
| 10 | BadRequestException |  | Exception when Request is invalid |

1. Sequence Diagram(s)
   1. Delete Store

*[Provide the detailed design for the feature <Feature Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

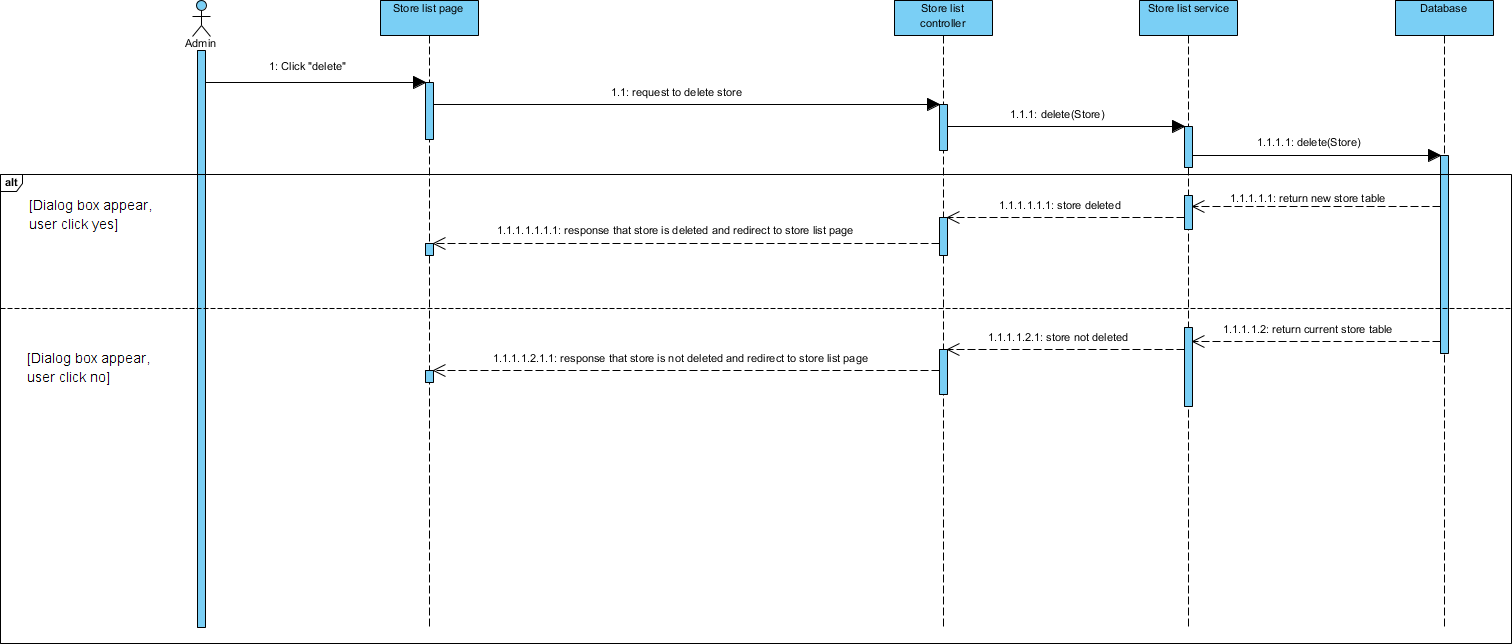
1. Class Diagram

*[This part presents the class diagram for the relevant feature]*



1. Class Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Name** | **Type** | **Require** | **Max-Length** | **Description** |
| 1 | Họ tên | Text | Yes | N/A | Display full name user |
| 2 | Địa chỉ | Text | Yes | N/A | Display address user. |
| 3 | Số điện thoại | Number | Yes | 10 | Display phone number of user. |
| 4 | Email | Input-text | Yes | N/A | Display address of user |
| 5 | Thao tác | Button | N/A | N/A | Button click all function of manage customer. |

1. Sequence Diagram(s)
2. Design:
   1. Search Store

*[Provide the detailed design for the feature <Feature Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

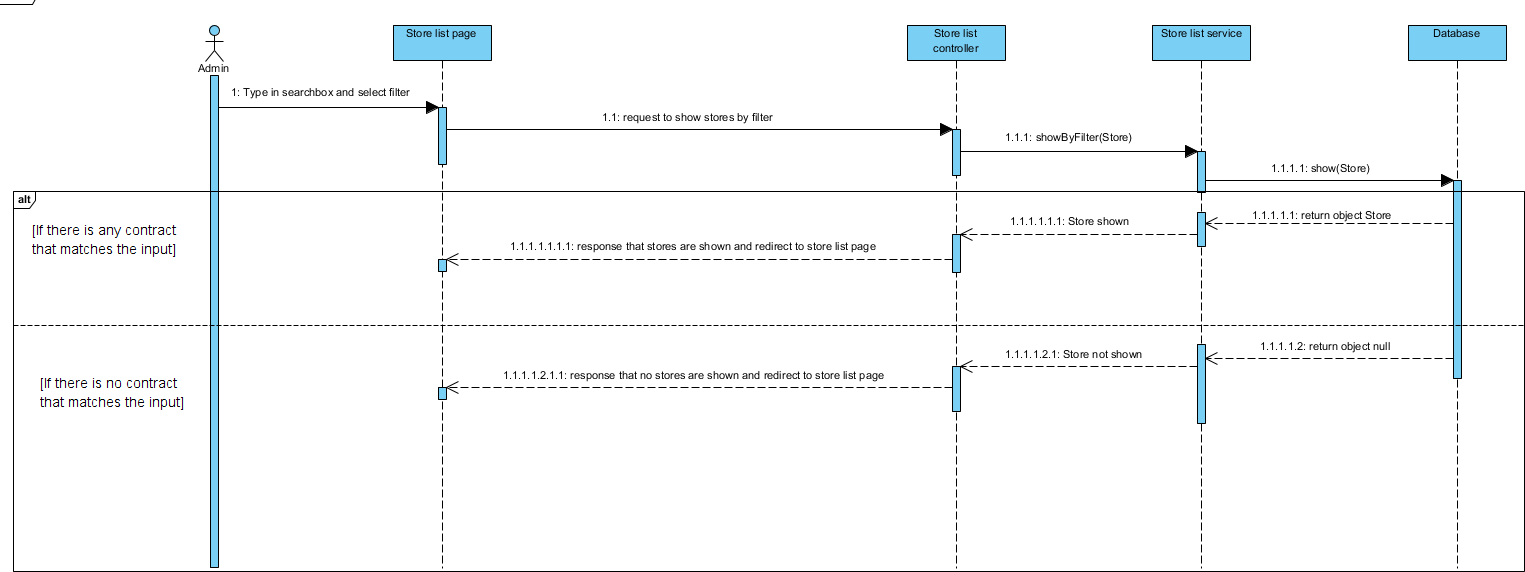
1. Class Diagram

*[This part presents the class diagram for the relevant feature]*

1. Class Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Name** | **Type** | **Require** | **Max-Length** | **Description** |
| 1 | Họ tên | Text | Yes | N/A | Display full name user |
| 2 | Địa chỉ | Text | Yes | N/A | Display address user. |
| 3 | Số điện thoại | Number | Yes | 10 | Display phone number of user. |
| 4 | Email | Input-text | Yes | N/A | Display address of user |
| 5 | Thao tác | Button | N/A | N/A | Button click all function of manage customer. |

1. Sequence Diagram(s)

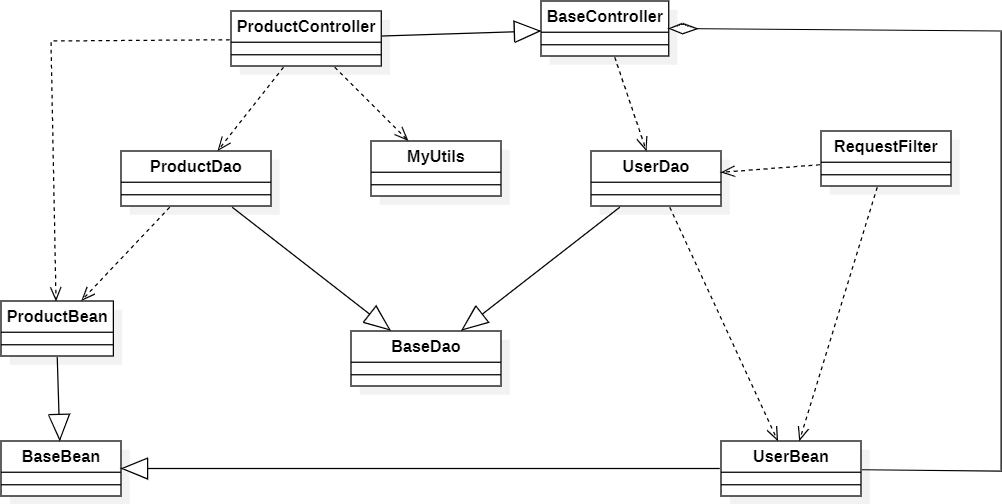


1. Design:
   1. View Map

*[Provide the detailed design for the feature <Feature Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

1. Class Diagram

*[This part presents the class diagram for the relevant feature]*



1. Class Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Name** | **Type** | **Require** | **Max-Length** | **Description** |
| 1 | Họ tên | Text | Yes | N/A | Display full name user |
| 2 | Địa chỉ | Text | Yes | N/A | Display address user. |
| 3 | Số điện thoại | Number | Yes | 10 | Display phone number of user. |
| 4 | Email | Input-text | Yes | N/A | Display address of user |
| 5 | Thao tác | Button | N/A | N/A | Button click all function of manage customer. |

1. Sequence Diagram(s)

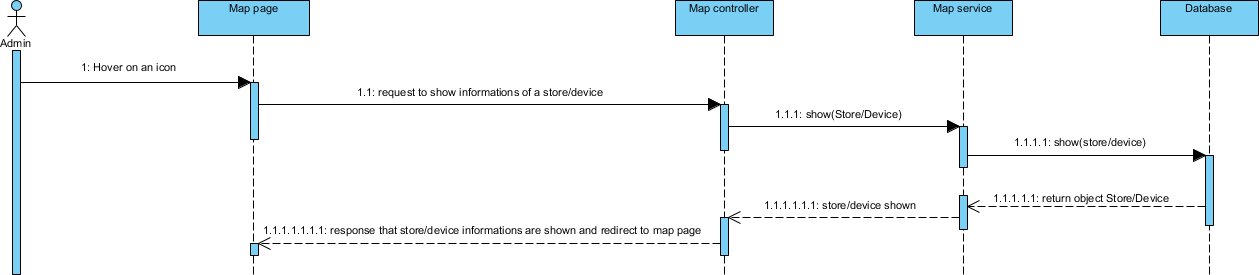


Figure: Admin sequence diagram

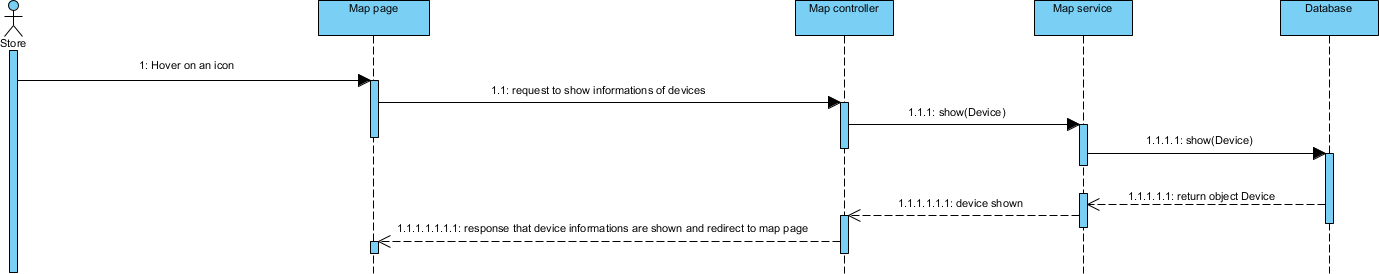


Figure: Store sequence diagram

1. Design:

## Data & Database Design

# Software Testing Documentation

## Overall Description

* + 1. Purpose

The primary purpose of this chapter is to detect and prevent defects which may be created by developers while developing the software and this may lead to software failures. On the other hand, another objective of this chapter is to provide information about the level of quality and to make sure that the end result meets the business and user requirements. It contains the following sections:

* Test model
* Test Plan
* Testing levels
* Testing types
* Test plan
* Test Stages
* Resources
* Test milestone
* Test Deliverables
* Test case
* Test report
  1. Test Model

This project follows the V-Model process to implement testing.

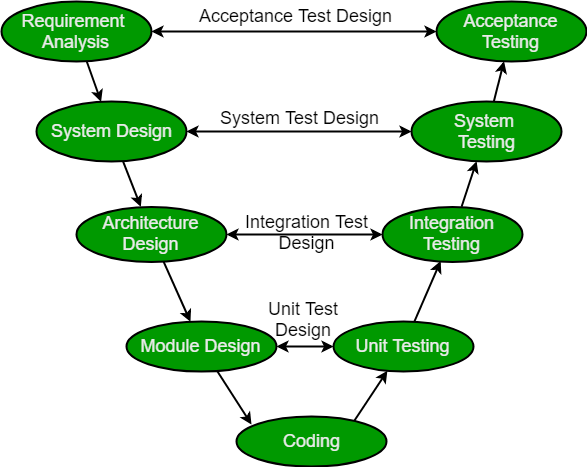


Figure 2.1: V-model process

|  |  |
| --- | --- |
| **Verification Phases** | **Validation Phases** |
| **User Requirement:** This is the first phase in the development cycle where the product requirements are understood from the. customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business  requirements can be used as an input for acceptance testing | **Acceptance Testing:** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment. |
| **Software Requirement Specification:** Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an  earlier stage leaves more time for the actual test execution later. | **System Testing**: System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution. |

|  |  |
| --- | --- |
| **Architectural Design:** Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken.  The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD). The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage. | **Integration Testing:** Integration testing is associated with the architectural Designphase. Integration tests are performed to test the coexistence and communication of the internal modules within the system. |
| **Detailed Design:** In this phase, the detailed internal Designfor all the system modules is specified, referred to as Low Level Design(LLD). It is important that the Designis compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be Designed at this stage based on the internal module Designs. | **Unit Testing:** Unit tests Designed in the module Designphase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing. |

You can reference at: https://viblo.asia/p/v-model-trong-kiem-thu-phan-mem-la-gi-tim-hieu-voi-vi-du- sdlc-stlc-Qbq5QMEL5D8

* 1. Testing Levels

About testing stage, we use four main software testing stages.

Unit Testing



During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as Designed. In this phase, a unit can refer to a function, individual program or even a procedure, and a [White-box Testing](http://stackoverflow.com/questions/402161/black-box-vs-white-box-testing) method is usually used to get the job done. One of the biggest benefits of this testing phase is that it can be run every time a

piece of code is changed, allowing issues to be resolved as quickly as possible. It’s quite common for software developers to perform unit tests before delivering software to testers for formal testing.

You can reference at: https://topdev.vn/blog/unit-test-la-gi/

Integration Testing



Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is Designed to find interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they aren’t properly integrated, it will affect the functionality of the software program. In order to run these types of tests, individuals can make use of various testing methods, but the specific method that will be used to get the job done will depend greatly on the way in which the units are defined.

You can reference at: https://viblo.asia/p/tim-hieu-ve-kiem-thu-tich-hop-integration-testing- yMnKM94AK7P

System Testing



System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven’t played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

You can reference at: https://viblo.asia/p/system-testing-kiem-thu-he-thong-aWj53pOPK6m

Acceptance Testing



The final level, Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business’ needs. Once this process has been completed and the software has passed, the program will then be delivered to production.

You can reference at:

* 1. Testing Types

The different types of testing that will be carry out this project are:

* Function Test
  + The target-on-test should focus on any requirements for test that can be traced directly to uses cases or business rules. The goals of these test are to verify proper data acceptance, processing and retrieval and the appropriate implementation of the business rules. This type of testing is based upon black box techniques, that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or result.
  + The implementation of functional test will be passed if all functional cases in Test case document are tested and passed.
* GUI Test
  + Graphical User Interface (GUI) Testing verifies a user’s interaction with the software. The goal of GUI testing is to ensure that the GUI provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, GUI testing ensures that the objects within the GUI function as expected and conform to requirements.
  + GUI test will be performed fully on all screens
  + This test is targeted to cover the verification of the overall look and feel of the Gas-Meter system including initial position, color, focus, initial button, text view, edit text, screen sizes and sentences width.
* Acceptance test
  + The acceptance testing is a test conducted to determine if the requirements of a specification or contract are met.
  + It involves alpha testing and beta testing. Alpha testing takes place at developers’ sites and involves testing of the operational system by internal member, before it is released to external users. Beta testing takes place at user’s sites and involves testing by a group of users who use the system at their own locations and provide feedback, before the system is release to all users.

## 2 Test Plan

* 1. Features to be tested

|  |  |  |
| --- | --- | --- |
| Group of function | Functions | Actor |
| Login | Register with email | User |
| Login with email |
| Logout |
| Personal information | View personal information |
| Update personal information |
| View gas-meter report | View gas concentration |
| View temperature |
| View weight of gas cylinder |
| View Notification | View notification |
| Manage user | View list user | Admin, Store |
|  | Add user |
| Update user |
| Search user |
| Show location of user |
| Manage contract | View list contract |
| Create contract |
| Update contract |
| Search |
| View list broken device | View list broken device |
| View Map | View map |
| Manage store | View list store | Admin |
| Create store |
| Update store |
| Search store |

* 1. Features to be tested
* Out of scope features will not be tested.
* Non-functional requirements will not be tested.
  1. Testing tool
     + Google Sheet: Used to track bugs.



* + - Microsoft Excel: Used to manage test cases and manage bugs.



* + - Android Phone: Used to test features and displays.



* + - Android Studio: To view logs, inspect elements.
    - Google Chrome: Use to view the web page, bug logging page, etc..



* + - Postman: To manage the list of all APIs and manually test API’s result.



* + - Visual studio code: Use to perform unit tests.
    - Mockito: Mockito is a framework that supports the creation of unit tests using mock objects (Mock or TestDouble) in an easy-to-use manner without creating "noise" from unrelated interactions.



* 1. Test Stages

|  |  |  |  |
| --- | --- | --- | --- |
| Type of tests | Stage of test | | |
| Unit testing | Integration testing | System testing |
| Function test | X | X | X |
| GUI Test |  | X | X |
| Acceptance test |  |  | X |
| System test |  |  | X |

* 1. Resources

1. Human Resources

|  |  |  |
| --- | --- | --- |
| ID | Resource | Responsibilities |

|  |  |  |
| --- | --- | --- |
| 1 | Project Manager | * Responsible for Project Schedules and overall success of the project. * Review test cases and test report. |
| 2 | Test Leader | * Review test cases and test report. * Review overall quality of the project. |
| 3 | Testers | * Preforming the actual system testing * Create test plan * Create test cases * Perform tests * Write test reports * Logbugs |
| 4 | Developers | * Create and perform Unit Tests. * Fix bugs |

1. Environment

In this project, the test stage is very important, and must have a certain environment. However, in this project, the effort of the team is limited, so we will use our devices to execute tests. Besides that, we focus on the function, so we will ignore the line’s speech configuration speech, configuration of the server.

The following table describes the testing environment:

|  |  |  |
| --- | --- | --- |
| Type of testing | Software | Hardware |
| Unit test | JUnit4, Mockito |  |
| Integration test |  | * Android Studio Virtual Device * Samsung Galaxy A30 OS Android 10, Ram 4GB, Rom 64GB * Samsung A6 Plus OS Android 10, Ram 3GB, Rom 32GB * Sony XZS OS Android 8.0, Ram 4GB, Rom 64GB |
| System test |  | * Android Studio Virtual Device * Samsung Galaxy A30 OS Android 10, Ram 4GB, Rom 64GB * Samsung A6 Plus OS Android 10, Ram 3GB, Rom 32GB * Sony XZS * OS Android 8.0, Ram 4GB, Rom 64GB |

* 1. Test Milestones

|  |  |  |
| --- | --- | --- |
| **No** | **Milestones** | **Delivery Date** |
| 1 | Project Registration | 14/09/2020 |
| 2 | Submit: Report1\_Project Introduction | 28/09/2020 |

|  |  |  |
| --- | --- | --- |
| 3 | Submit: Report2\_Project Management Plan | 11/10/2020 |
| 4 | Submit: Report3\_System Requirement Specification | 25/10/2020 |
| 5 | Submit: Report4\_Software Design Document | 20/11/2020 |
| 6 | Submit: Report5\_Test Documentation | 01/12/2020 |
| 7 | Submit: Report6\_Software User Guides | 08/12/2020 |
| 8 | Submit Final Report | 23/12/2020 |
| 9 | Submit all project resources | 23/12/2020 |
| 10 | Project defense | 31/12/2020 |

* 1. Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverables** | **Responsibilities** | **Completion Date** |
| Test plan | Testers | 29/09/2020 |
| Test cases | Testers | 03/11/2020 |
| Test case Review | Testers + Project Manager | 01/12/2020 |
| Defect Log | All members | 13/12/2020 |
| Final Test Summary Report | All members | 21/12/2020 |

## Test Process Model

Figure: Test process model

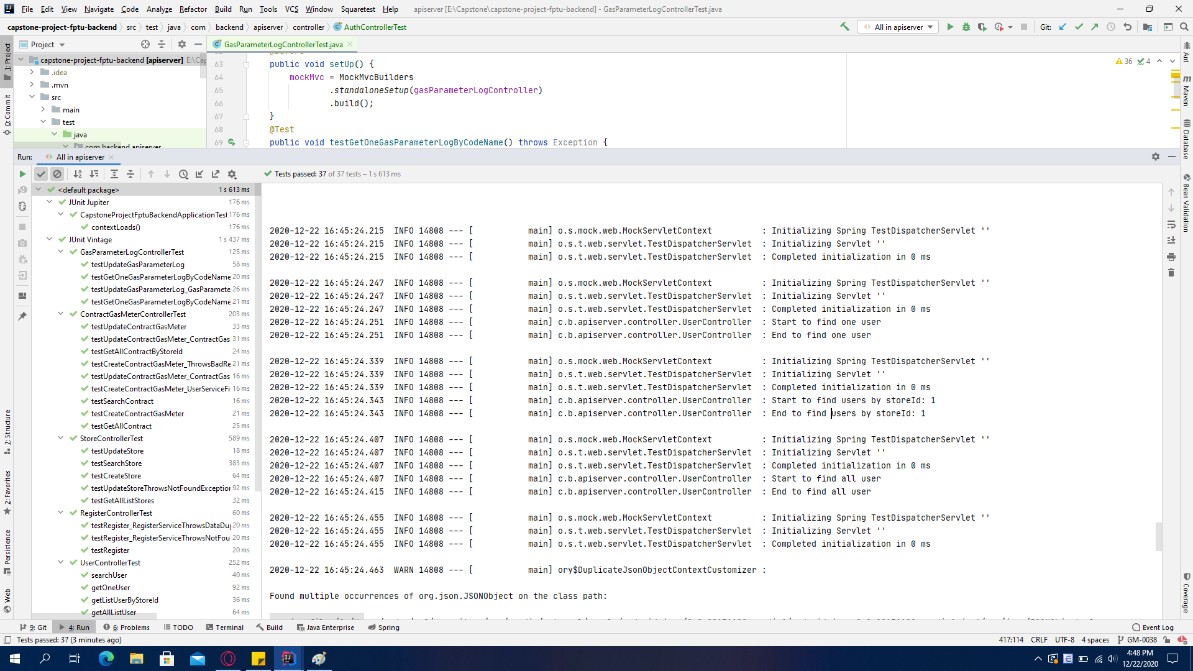
## Test Cases

See the reference for detail of test cases

* + - *Unit Test Cases: Gas\_Metter\_ Report5\_Unit Test Case.xls*
    - *Other Test Cases: Gas\_Metter \_Report5\_ Test Case.xls]*

## Test Result

* 1. Unit test

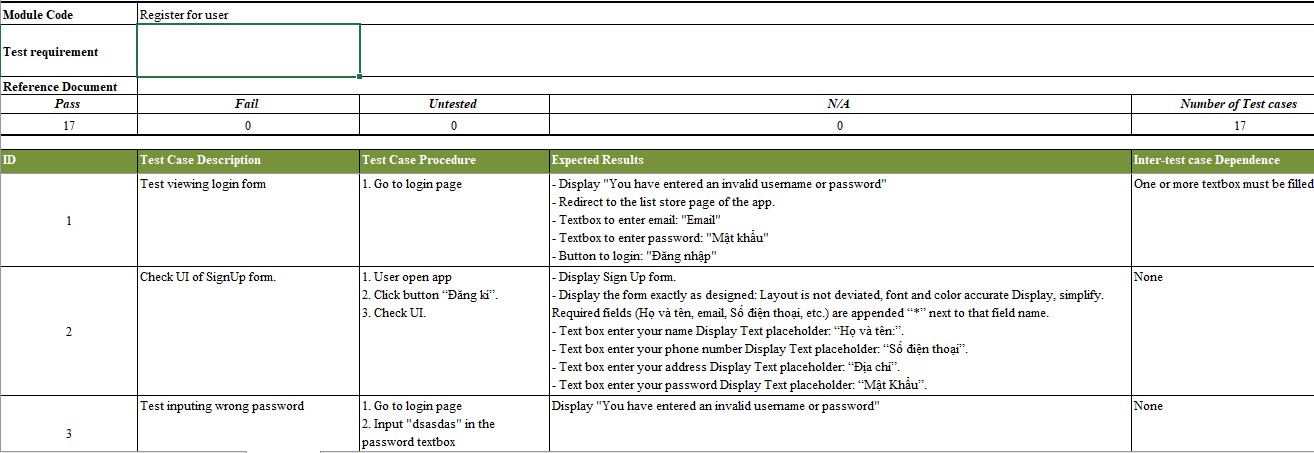


* 1. Integration Test and System Test

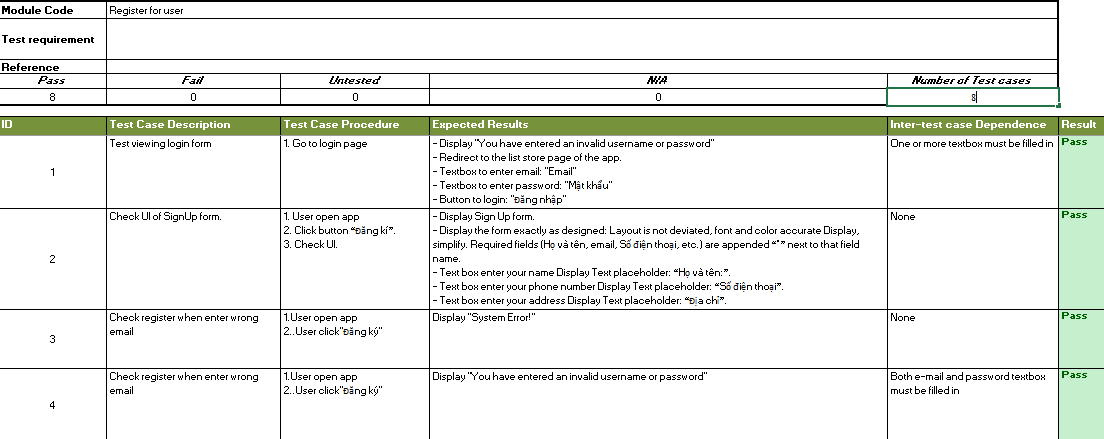
Integration test and System test are done by testers to ensure that combined units work correctly and that the system functions as intended.

GUI testing is also done during this process to ensure that elements and functions load correctly, text is readable, and the website interface looks good in various browser sizes.

This is sample tests of Integration Test and System Test in Gas-Meter system:



*Figure:* Integration Test



*Figure:* System Test

## Test Report

* 1. Unit test

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit Test Report** | | | |
| Project\_name | Gas\_Meter | Creater | PhuongNH,HungVD |
| Project code | Gas\_Meter | Reviewer/Approver |  |
| Document Code | Gas\_Meter\_Report | Issue Date |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Function | Passed | Failed | Untested | Total Test Case |
| 1 | performAuthentication | 1 | 0 | 0 | 1 |
| 2 | getAllListUser | 1 | 0 | 0 | 1 |
| 3 | findUserById | 1 | 0 | 0 | 1 |
| 4 | findUserByEmail | 1 | 0 | 0 | 1 |
| 5 | Save(User) | 1 | 0 | 0 | 1 |
| 6 | changePassword | 1 | 0 | 0 | 1 |
| 7 | findAllUserByStoreId | 1 | 0 | 0 | 1 |
| 8 | searchAllUser | 1 | 0 | 0 | 1 |
| 9 | listEntityToResponse | 1 | 0 | 0 | 1 |
| 10 | getAllListStores | 1 | 0 | 0 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | createStore | 1 | 0 | 0 | 1 |
| 12 | updateStore | 1 | 0 | 0 | 1 |
| 13 | searchStore | 1 | 0 | 0 | 1 |
| 14 | requestToEntity | 1 | 0 | 0 | 1 |
| 15 | getAllContractGasMeter | 1 | 0 | 0 | 1 |
| 16 | getAllContractGasMeterByStoreId | 1 | 0 | 0 | 1 |
| 17 | getOneContractGasMeter | 1 | 0 | 0 | 1 |
| 18 | getOneContractGasMeterByUserId | 1 | 0 | 0 | 1 |
| 19 | createContractGasMeter | 1 | 0 | 0 | 1 |
| 20 | updateContractGasMeter | 1 | 0 | 0 | 1 |
| 21 | findByGasMeterCode | 1 | 0 | 0 | 1 |
| 22 | getParameterLogById | 1 | 0 | 0 | 1 |
| 23 | createParameterLog | 1 | 0 | 0 | 1 |
| 24 | updateParameterLog | 1 | 0 | 0 | 1 |
| 25 | getAllNotify | 1 | 0 | 0 | 1 |
| 26 | createHistoryNotification | 1 | 0 | 0 | 1 |
| 27 | getAllNotifyByUserId | 1 | 0 | 0 | 1 |
| 28 | findOneNotifyById | 1 | 0 | 0 | 1 |
| 29 | updateNotificaiton | 1 | 0 | 0 | 1 |
| 30 | getAllFcmTokenByUserIdAndStatus | 1 | 0 | 0 | 1 |
| 31 | saveToken | 1 | 0 | 0 | 1 |
| 32 | findByToken | 1 | 0 | 0 | 1 |
| 33 | getAllListGasParameterFireBase | 1 | 0 | 0 | 1 |
| 34 | createNewSnapshotFirebase | 1 | 0 | 0 | 1 |
| 35 | updateGasParameterFireBase | 1 | 0 | 0 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 36 | deleteParameterFireBase | 1 | 0 | 0 | 1 |
|  | Total | 36 | 0 | 0 | 36 |

* 1. Integration Test

We have deployed all integration test cases that the system can have. In our systems, we have a total 482 test cases and all test cases are passed.

The number of integration test in phase is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Integration Test Report** | | | |
| Project\_name | Gas\_Meter | Creater | PhuongNH,ThanhTN |
| Project code | Gas\_Meter | Reviewer/Approver |  |
| Document Code | Gas\_Meter\_Report | Issue Date |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Role | N  o | Function | Passed | Failed | Untested | Total Test Case |
| User | 1 | Test Login User | 3 | 0 | 0 | 3 |
| 2 | Test Logout User | 1 | 0 | 0 | 1 |
| 3 | Test Register user | 17 | 0 | 0 | 17 |
| 4 | Test Update Personal Information | 29 | 0 | 0 | 29 |
| 5 | Test View Gas\_Meter Report | 16 | 0 | 0 | 16 |
| 6 | Test View Notification | 30 | 0 | 0 | 30 |
| Store, Admin | 7 | Test store Login | 3 | 0 | 0 | 3 |
| 8 | Test store Logout | 1 | 0 | 0 | 0 |
| 9 | Test View list user | 10 | 0 | 0 | 10 |
| 10 | Test create user | 48 | 0 | 0 | 48 |
| 11 | Test update user | 36 | 0 | 0 | 36 |
| 12 | Test search user | 25 | 0 | 0 | 25 |
| 13 | Test view location of user store | 28 | 0 | 0 | 28 |
| 14 | Test view list contract | 10 | 0 | 0 | 10 |
| 15 | Test create contract | 38 | 0 | 0 | 38 |
| 16 | Test update contract | 26 | 0 | 0 | 26 |
| 17 | Test search contract | 17 | 0 | 0 | 17 |
| 18 | Test view list broken | 4 | 0 | 0 | 4 |
| 19 | Test view map | 48 | 0 | 0 | 48 |
| Admin | 20 | Test Admin Login | 3 | 0 | 0 | 3 |
|  | 21 | Test Admin Logout | 1 | 0 | 0 | 1 |
|  | 22 | Test View list store of Admin | 10 | 0 | 0 | 10 |
|  | 23 | Test create store of Admin | 38 | 0 | 0 | 38 |
|  | 24 | Test update store of Admin | 26 | 0 | 0 | 26 |
|  | 25 | Test search store of Admin | 14 | 0 | 0 | 14 |
|  |  | Sub Total | 482 |  |  | 482 |

* 1. System Test

|  |  |  |  |
| --- | --- | --- | --- |
| **System Test Report** | | | |
| Project\_name | Gas\_Meter | Creater | PhuongNH,ThanhTN |
| Project code | Gas\_Meter | Reviewer/Approver |  |
| Document Code | Gas\_Meter\_Report | Issue Date |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Role | N  o | Function | Passed | Failed | Untested | Total Test Case |
| User | 1 | Test Login User | 3 | 0 | 0 | 3 |
| 2 | Test Logout User | 1 | 0 | 0 | 1 |
| 3 | Test Register user | 8 | 0 | 0 | 8 |
| 4 | Test Update Personal Information | 14 | 0 | 0 | 14 |
| 5 | Test View Gas\_Meter Report | 8 | 0 | 0 | 8 |
| 6 | Test View Notification | 16 | 0 | 0 | 16 |
| Store, Admin | 7 | Test store Login | 3 | 0 | 0 | 3 |
| 8 | Test store Logout | 1 | 0 | 0 | 0 |
| 9 | Test View list user | 5 | 0 | 0 | 5 |
| 10 | Test create user | 24 | 0 | 0 | 24 |
| 11 | Test update user | 18 | 0 | 0 | 18 |
| 12 | Test search user | 13 | 0 | 0 | 13 |
| 13 | Test view location of user store | 14 | 0 | 0 | 14 |
| 14 | Test view list contract | 5 | 0 | 0 | 5 |
| 15 | Test create contract | 20 | 0 | 0 | 20 |
| 16 | Test update contract | 13 | 0 | 0 | 13 |
| 17 | Test search contract | 9 | 0 | 0 | 9 |
| 18 | Test view list broken | 2 | 0 | 0 | 2 |
| 19 | Test view map | 23 | 0 | 0 | 23 |
| Admin | 20 | Test Admin Login | 3 | 0 | 0 | 3 |
|  | 21 | Test Admin Logout | 1 | 0 | 0 | 1 |
|  | 22 | Test View list store of Admin | 5 | 0 | 0 | 5 |
|  | 23 | Test create store of Admin | 20 | 0 | 0 | 20 |
|  | 24 | Test update store of Admin | 13 | 0 | 0 | 13 |
|  | 25 | Test search store of Admin | 7 | 0 | 0 | 7 |
|  |  | Sub Total | 249 |  |  | 249 |

* 1. User Acceptance Test

|  |  |  |  |
| --- | --- | --- | --- |
| **User Acceptance Test** | | | |
| Project\_name | Gas\_Meter | Creater | PhuongNH,ThanhTN |
| Project code | Gas\_Meter | Reviewer/Approver |  |
| Document Code | Gas\_Meter\_Report | Issue Date |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Check lists | | | | |
|  | ID |  | Yes | No |
| General | GM-001 | Every buttons have activity and working normally | ۷ |  |
| GM-002 | Spelling and grammatical is correct | ۷ |  |
| GM-003 | Update functionality of any record on page has confirmation | ۷ |  |
| GM-004 | All mandatory fields are validated | ۷ |  |
| GM-005 | All required field cannot be blank/space | ۷ |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | GM-006 | All numeric values are properly formatted | ۷ |  |
| GM-007 | Error diaLogmessages are displayed | ۷ |  |
|  | | | | |
| GUI and Usability | GM-008 | Screens are designed following project requirement | ۷ |  |
| GM-009 | User-friendly and easy-to-use screen interface | ۷ |  |
| GM-010 | All fields on page are all reasonably aligned | ۷ |  |
| GM-011 | The information is arranged symmetrically with the right distance between the  components. | ۷ |  |
| GM-012 | The important fields and buttons are located where they are easy to see. | ۷ |  |
| GM-013 | Information is displayed in the right order | ۷ |  |
| GM-014 | The text easy to read. Do not use slang, acronyms, and abbreviations | ۷ |  |
| GM-015 | Font size, text style and color as specified in SRS. | ۷ |  |
| GM-016 | Has Copyrighted icons and images | ۷ |  |
| GM-017 | The text is clear, concise, and meaningful | ۷ |  |
| GM-018 | The notification in mobile displays all necessary and timely information | ۷ |  |
| GM-019 | Full and detailed display of necessary parameters | ۷ |  |
| GM-020 | Displays exactly position on the map | ۷ |  |
|  | | | | |
| Database | GM-021 | Correct data is saved in database. | ۷ |  |
| GM-022 | Not null column has values | ۷ |  |
| GM-023 | Fields are designed with correct data type and data length. | ۷ |  |
|  | | | | |
| Performance | GM-024 | Quick response display | ۷ |  |
| GM-025 | Location in the map appeared almost immediately | ۷ |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | GM-026 | Receive uninterrupted notifications | ۷ |  |
|  | | | | |
| Security | GM-027 | Check password security and password policy enforcement. | ۷ |  |
| GM-029 | Verify old password When change password | ۷ |  |
| GM-030 | Check application logout functionality. | ۷ |  |
|  | | | | |

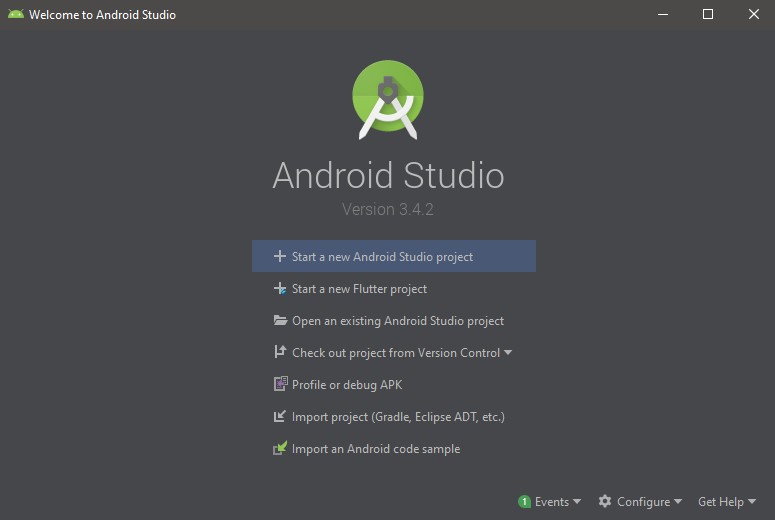
# Release Package & User Guides

## Installation Guides

\*Note : Users need connection network to use the features in the app.

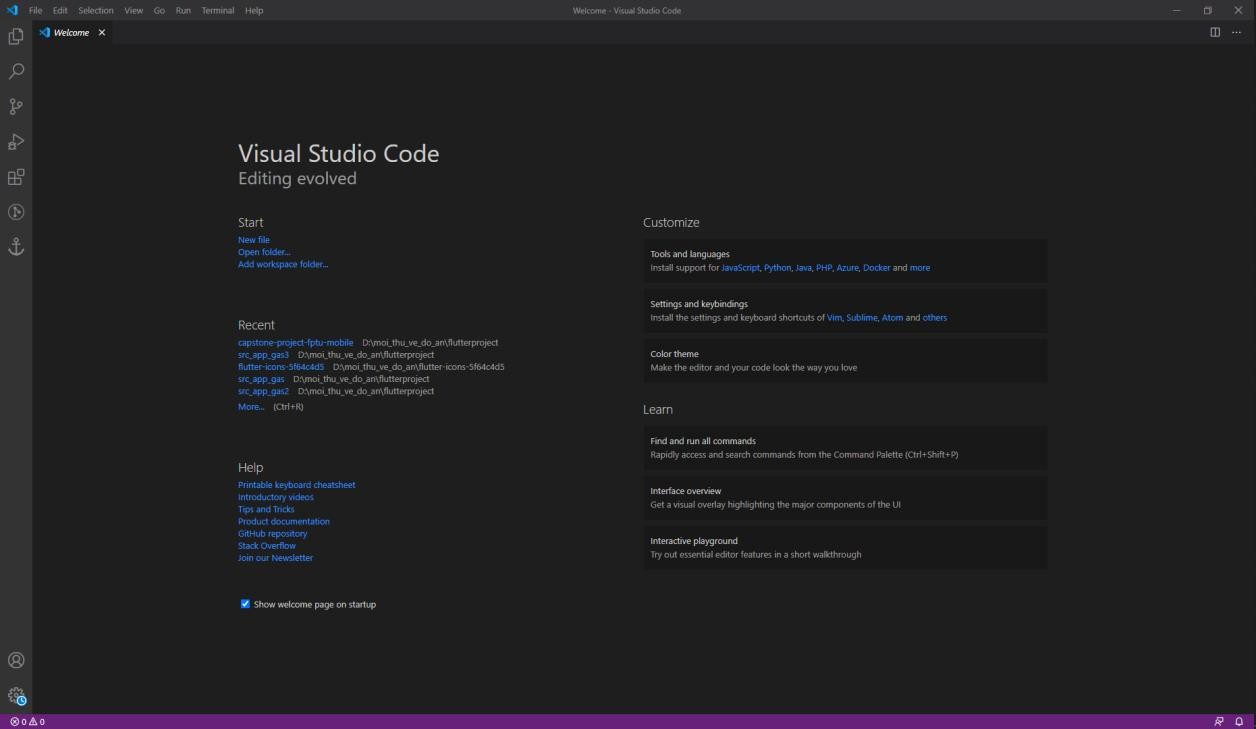
* 1. Setup Files
     + Android Studio

Dowload Android Studio from [https://developer.android.com/studio,](https://developer.android.com/studio) then follow the instruction to install.



* + - Visual Studio Code

Download VSCode from <https://code.visualstudio.com/download>, then follow the instruction to install.



* + - IntelliJ IDEA

Download IntelliJ IDE from <https://www.jetbrains.com/idea/download/>, the follow the instruction to install.



* + - Flutter



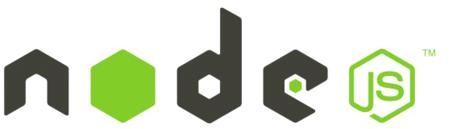
We use Flutter for developing and running GasMeter mobile application. You can go: <https://flutter.dev/docs/get-started/install>, then follow the instruction to install.

* + - JDK 8



Download and install JDK 8 following this link <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>.

* + - NodeJS



Install NodeJS by following this link <https://nodejs.org/en/download/>. The npm will automatically be install along with NodeJS.

* + - Angular 8



Install Angular 8 by following this link <https://angular.io/guide/setup-local>.

* + - MySQL



Download and Install MySQL from this link: https://dev.mysql.com/downloads/workbench/

* 1. Installation Instruction
     1. Mobile Application

Step 1: Clone project.

$ git clone <https://dangnhtp@bitbucket.org/capstone-fptu-2020/capstone-project-fptu-mobile.git>

Step 2: Go to project folder.

$ cd flutterproject\capstone-project-fptu-mobile Step 3: Update dependencies.

$ flutter pub get

Step 4: Connect with Android Device. Step 5: Run project in Device.

$ flutter run lib/main.dart

* + 1. Gas\_meter front-end

Step 1: Clone project.

$ git clone https://bitbucket.org/capstone-fptu-2020/capstone-project-fptu-frontend/src/master/-fptu- frontend/src/master/

Step 2: Go to project folder.

$ cd capstone-project-fptu-frontend Step 3: Update dependencies.

$ npm install

Step 4: Connect with Android Device.

$ npm audit fix

Step 5: Run project in Device.

$ ng serveA

* + 1. Gas\_meter back-end

Step 1: Clone the project git clone

<https://bitbucket.org/capstone-fptu-2020/capstone-project-fptu-backend/src/master/> Step 2: Go to the directory containing the project

cd /capstone-project-fptu-backend/src/master/ Step 3 : Create database gasmeterdb in MySQL Step 4: Run backend

mvn spring-boot:run