

Capstone project document

GREEN - HOA LAC



Version 1.0 approved

Prepared by Green Hoa Lac group

Date created: 05-08-2017

Last updated: 08-15-2017

|  |  |  |
| --- | --- | --- |
| Green – Hoa Lac | | |
| Group members | Pham Quang Khang | SE03624 |
| Le Anh Tuyen | SE03863 |
| Nguyen Thi Ly Linh | SE04216 |
| Bui Manh Tri | SE03590 |
| Lam Duc Thang | SE03529 |
| Ho Quang Hao | SE03856 |
| Supervisor | Phan Duy Hung |  |
| Extra supervisor | N/A |  |
| Capstone project code | GHL |  |
| Capstone project name | Green – Hoa Lac |  |

Table of Contents

[I. Introduction 7](#_Toc490591022)

[1. Introduction and motivation 7](#_Toc490591023)

[2. Existing products 7](#_Toc490591024)

[2.1. Schneider Sprinkler Timer and Controller 8](#_Toc490591025)

[2.2. HT-01 Mobile phone Controller 9](#_Toc490591026)

[3. Scope 9](#_Toc490591027)

[4. Definitions, Acronyms and Abbreviations 11](#_Toc490591028)

[II. Project management Plan 12](#_Toc490591029)

[1. Project organization 12](#_Toc490591030)

[1.1. Software process model 12](#_Toc490591031)

[1.2. Roles and responsibilities 14](#_Toc490591032)

[2. Tools and infrastructures 16](#_Toc490591033)

[2.1. Hardware 16](#_Toc490591034)

[2.2. Tools and software 19](#_Toc490591035)

[3. Risk management plan 22](#_Toc490591036)

[3.1. Risks register 22](#_Toc490591037)

[3.2. Risk probability and impact 22](#_Toc490591038)

[4. Communication plan 23](#_Toc490591039)

[4.1. Project structure 24](#_Toc490591040)

[4.2. Project communication 24](#_Toc490591041)

[4.2.1. Format, content, and level of detail of key project information 24](#_Toc490591042)

[4.2.2. Project report 25](#_Toc490591043)

[5. Project schedule 26](#_Toc490591044)

[III. System Requirement Specification 28](#_Toc490591045)

[1. Green Hoa Lac Android Application and PCM 28](#_Toc490591046)

[1.1. Functional Requirement Specification 28](#_Toc490591047)

[1.1.1. UC01 – Login 28](#_Toc490591048)

[1.1.2. UC02 – View weather’s information 32](#_Toc490591049)

[1.1.3. UC03 – View application’s information 34](#_Toc490591050)

[1.1.4. UC04 – Turn on node 36](#_Toc490591051)

[1.1.5. UC05 – Turn off node 40](#_Toc490591052)

[1.1.6. UC06 – Set schedule for node 44](#_Toc490591053)

[1.1.7. UC07 – View schedules 48](#_Toc490591054)

[1.1.8. UC08 – Cancel schedules of node 51](#_Toc490591055)

[1.1.9. UC09 – Recharge 54](#_Toc490591056)

[1.1.10. UC10 – View node’s history 56](#_Toc490591057)

[1.1.11. UC11 – Search node’s history 58](#_Toc490591058)

[1.1.12. UC12 – Synchronize nodes 61](#_Toc490591059)

[1.1.13. UC13 – Synchronize node’s history 64](#_Toc490591060)

[1.1.14. UC14 – Logout 67](#_Toc490591061)

[1.2. Non- Functional Requirement Specification 70](#_Toc490591062)

[2. Green Hoa Lac Web Management System 70](#_Toc490591063)

[2.1. Functional Requirement Specification 70](#_Toc490591064)

[2.1.1. UC01 – Login 70](#_Toc490591065)

[2.1.2. UC02 – View user’s information 74](#_Toc490591066)

[2.1.3. UC03 – Add user 76](#_Toc490591067)

[2.1.4. UC04 – Search user’s information 79](#_Toc490591068)

[2.1.5. UC05 – Update user’s role 82](#_Toc490591069)

[2.1.6. UC06 – View user’s history 85](#_Toc490591070)

[2.1.7. UC07 – Lock user 87](#_Toc490591071)

[2.1.8. UC08 – Unlock user 89](#_Toc490591072)

[2.1.9. UC09 – View activity history 92](#_Toc490591073)

[2.1.10. UC10 – Search activity history 95](#_Toc490591074)

[2.1.11. UC11 – Add new node 97](#_Toc490591075)

[2.1.12. UC12 – Update node’s position 100](#_Toc490591076)

[2.1.13. UC13 – View node’s information 102](#_Toc490591077)

[2.1.14. UC14 – View node’s history 104](#_Toc490591078)

[2.1.15. UC15 – Update node’s information 107](#_Toc490591079)

[2.1.16. UC16 – Delete node 109](#_Toc490591080)

[2.1.17. UC17 – Sort activity history 112](#_Toc490591081)

[2.1.18. UC18 – Logout 115](#_Toc490591082)

[2.2. Non-Functional Requirement Specification 117](#_Toc490591083)

[IV. System design 117](#_Toc490591084)

[1. Architecture design 117](#_Toc490591085)

[1.1. System architecture design 118](#_Toc490591086)

[1.2. Serial port (UART) communication protocol 119](#_Toc490591087)

[1.3. GSM 120](#_Toc490591088)

[2. Mechanical design 121](#_Toc490591089)

[3. Hardware design 121](#_Toc490591090)

[4. Firmware design 122](#_Toc490591091)

[4.1. Turn on node 123](#_Toc490591092)

[4.2. Turn off node 124](#_Toc490591093)

[4.3. Set schedule 125](#_Toc490591094)

[4.4. Cancel schedule 126](#_Toc490591095)

[4.5. Get schedule 127](#_Toc490591096)

[5. Software design 127](#_Toc490591097)

[5.1. Login 128](#_Toc490591098)

[5.2. Turn on node 129](#_Toc490591099)

[5.3. Turn off node 130](#_Toc490591100)

[5.4. Set schedule 131](#_Toc490591101)

[5.5. Cancel Schedule 132](#_Toc490591102)

[5.6. Synchronize node 133](#_Toc490591103)

[6. Database design 133](#_Toc490591104)

[6.1. Server database 133](#_Toc490591105)

[6.2. Android application database 135](#_Toc490591106)

[V. Implement and testing 136](#_Toc490591107)

[1. Testing life cycle 136](#_Toc490591108)

[2. Testing model and tools 137](#_Toc490591109)

[2.1. Testing model 137](#_Toc490591110)

[2.2. Testing tools 138](#_Toc490591111)

[3. Test case and log bugs 138](#_Toc490591112)

[4. Implement 140](#_Toc490591113)

[4.1. Version 1.0 140](#_Toc490591114)

[4.2. Version 1.1 142](#_Toc490591115)

[5. Evidences 142](#_Toc490591116)

[5.1. Hardware device broken 143](#_Toc490591117)

[5.2. Turn off node unsuccessfully as Admin role 144](#_Toc490591118)

[5.3. Test set schedule unsuccessfully 145](#_Toc490591119)

[VI. Conclusion and perspective 145](#_Toc490591120)

[VII. Reference 146](#_Toc490591121)

## I. Introduction

### 1. Introduction and motivation

In the world nowadays, modern electric- electronic technologies are gaining more and more popular; and many developed countries like Japan, Israel, or America have applied them to many industries or agriculture relating areas. These are also the first countries using irrigation systems, watering remote, or aircrafts spraying pesticides, especially the most popular and useful system is watering remote which helps reduce worker’s efforts, save water source and rise productivity. This system have been applied from small gardens to large farms.

Despite the above benefits, Vietnamese people actually do not have potential opportunities to approach watering remote system. There are a few of watering remote systems applied large scope in Vietnam and they are also not stable or do not content most people’s requirements like: low budget, set schedule. Some difficulties such as high cost, import devices from foreign countries or manage or extend depended on others parties (producer or distributers) are barriers for people to use the system. In addition, although there are systems or devices like HT-01 or Schneider, they are not satisfied by user because their limitation.

As a result, the group decide to research and develop new watering remote system which overcomes most of the above disadvantages and is appropriate to Vietnamese people. The system is expected to use modern devices and technologies like android application, web management to remote, set schedule and especially manage in large and discrete area.

### 2. Existing products

An excellent watering remote system requires the productivity and save cost. It cannot stop with only actions turn on or off bumps through text message or phone call from long distance. User’s requirements are increasing rapidly. With the technology development especially in embedded systems, people expect their watering remote system as automatically as possible, which can reduce need in human resources. For example: Set schedule for turning on or off the bumps for a long time like daily or weekly; automatically check account. In addition, to use the system, customers tend to pay a huge amount of money than a basic system with only the two actions.

Nevertheless, in Vietnam they focus mainly on functions like turn on or off nodes. Setting schedule and warning account in nodes are rare or have limitation such as duration time (lower than 24 hour), interact manually to check account regularly. Moreover, there is also a restriction in user management because these systems allow a small number of user interacting.

#### 2.1. Schneider Sprinkler Timer and Controller



Figure 1: Schneider Sprinkler Timer and Controller

Mechanism: Controlling by telephone’s text messages

* Turn on or off device with long distance by telephone
* Send text message to telephone to announce or warn status of devices such as: be on, be off, power loss, be cut off, SIM banned
* Activate on or off devices by text messages or called miss
* Set up time for turning on or off devices by one message (30 minutes – 23.5 hours)

Advantages:

* Thanks to external antenna, these devices can receive high signal, high interference resistance
* High touch and stable

Disadvantages:

* High cost
* A limitation in set time for devices: cannot set time longer 24 hours

#### 2.2. HT-01 Mobile phone Controller

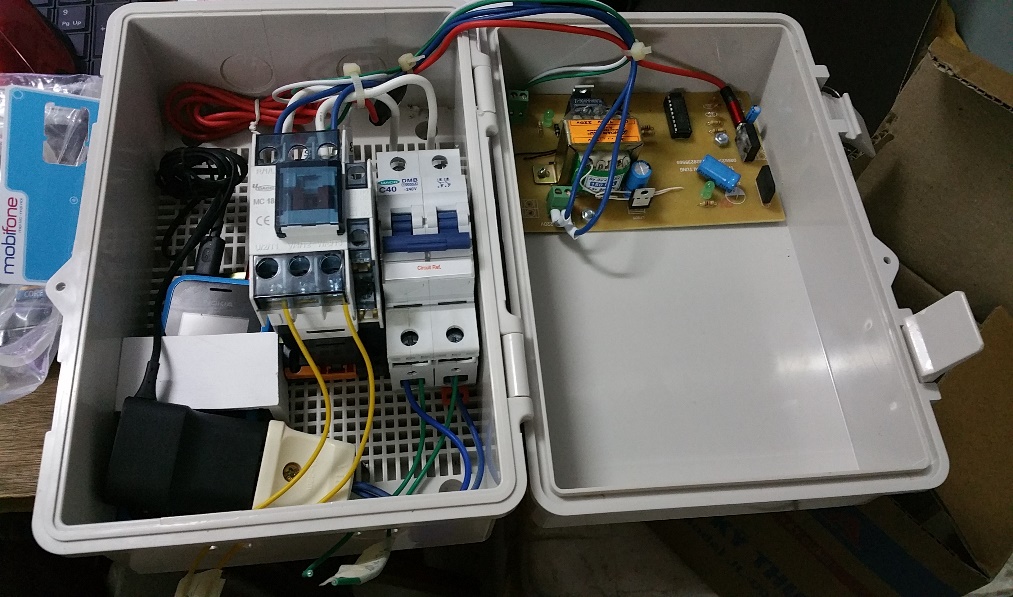


Figure 2: HT-01 Mobile phone Controller

Mechanism: Controlling by telephone’s calls

* Turn on or off device with long distance by telephone
* Call to a telephone number installed in HT -01 and valid code to turn on devices. The same for turn off devices but using other valid code.

Advantages:

* Low price for farmers and gardeners

Disadvantages:

* There are only two function: turn on and turn off devices. Cannot set schedule for devices

### 3. Scope

The group develops a new watering remote system called Green – Hoa Lac system with two main parts: android application to control and web management to set up, update users and areas applied the system.

On the one hand with the web management (called Green - Hoa Lac web management system), we manage nodes and users of whole system. First of all, Google map is applied to manage node’s locations because it provides accurate and portable locations, which is usually updated in case of changing. This way help us reduce time and efforts to design and build a new atlas served for node’s position and other information management. In addition, one area can divide into many subareas and that can be controlled by many other people, which lead to a need in management user and activity history of each node. As a result, we can easily shrink or extend scope of areas expected.

On the other hand with an android application (called Green - Hoa Lac android application), we use it remote pumps in each node, set scheduler for both short and long time duration for each node, and also can view weather forecast, activity history. The application use the same database and Google map with the above web. We control and set schedule for pumps through Subscriber Identity Module and the need for internet connection is limited.

Table 1: Scope and boundaries of whole system

|  |  |  |
| --- | --- | --- |
| Feature | Sub Feature | Description |
| Green- Hoa Lac web management system | Node’s management | * Create, update, delete node in the map * Manage information, description, history and position of node |
| User’s management | * Create, update, delete, lock or unlock users * View activity history |
| Green – Hoa Lac android application | Turn on/ off node | Control turn on or off node with long distance |
| Set/ get/ cancel schedule | Set or cancel schedule for node with detailed time, duration with frequency like one day, daily, weekly. Get all list of schedules of nodes |
| View weather forecast | With support of the Internet connection, user can view weather forecast directly in the application to control and set scheduler. |
| Synchronize and view node history | The application enable users control the system according to two ways which are online and offline. Therefore, synchronize and view node’s history are very important to control users and nodes. |

### 4. Definitions, Acronyms and Abbreviations

Table 2: Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| # | Acronyms | Definitions |
| 1 | GHLS | Green - Hoa Lac system |
| 2 | GHLWMS | Green - Hoa Lac web management system |
| 3 | GHLA | Green - Hoa Lac android application |
| 4 | PCM | Pump control module |

## II. Project management Plan

### 1. Project organization

#### 1.1. Software process model

Our project applies both plan-driven and agile processes. In plan-driven process, process activities are planned in advance and progress is measured against this plan while in agile process, planning is incremental and it is easier to change the process to reflect changing requirements.

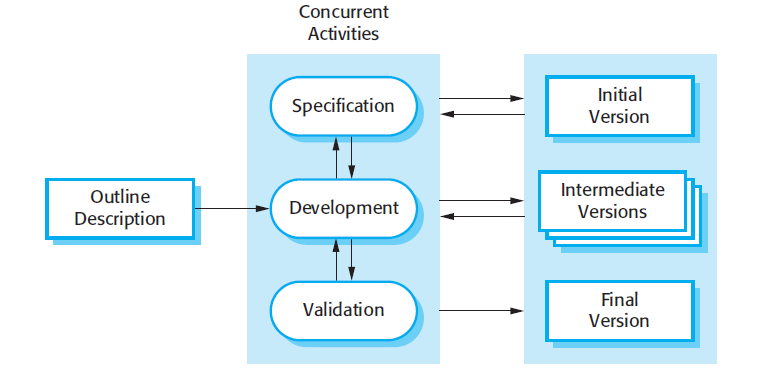


Figure 3: The Incremental Development Model

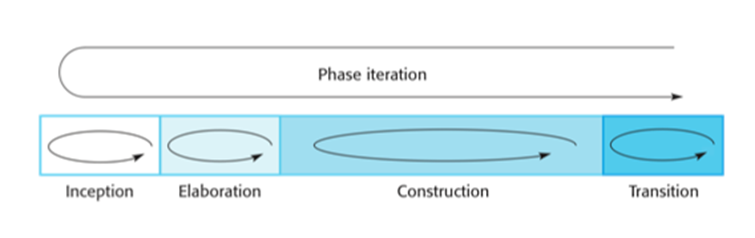


Figure 4: The phases of iterative development

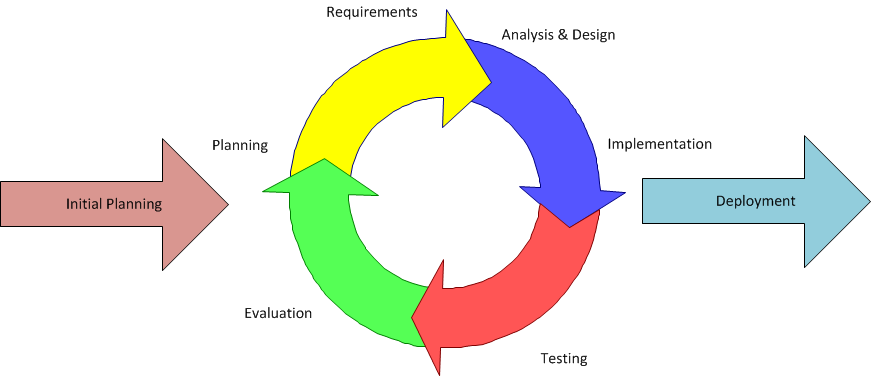


Figure 5: System Process Model

On the other hand, there is a combination between the Iterative and Incremental Software Process Model in the project. Iterative and incremental software development begins by the way plan and continue through iterative development cycles involving continuous user’s feedback and the incremental addition of features concluding with the deployment of completed software at the end of each cycle, which is appropriate for a gradual increase in feature additions and a cyclical release and upgrade pattern. These models are usually chosen in some situation. Firstly, the cost of accommodating changing requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model. Secondly, it is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. In addition, more rapid delivery and deployment of useful software to the customer is possible. Finally, these model do not require knowledge, skill of team’s members in advance.

#### 1.2. Roles and responsibilities

Table 3: Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Full name | Role | Responsibilities |
| 1 | Ph.D. Phan Duy Hung | Supervisor | * Supporting in raise idea and giving advices for design the system. * Approving and supporting process to run project. * Suggesting solutions when the project has issues. |
| 2 | Pham Quang Khang | Project manager | * Managing member’s tasks, set scheduler and risk * Set common rules for all members in project * Organizing meeting, communication plan to keep track project’s process. * Approve solutions to resolve issues |
| Developer | * Follow process of project and common rules. * Develop Green Hoa Lac web management system |
| 3 | Le Anh Tuyen | Technical leader | * Design system architecture |
| Developer | * Follow process of project and common rules. * Develop interaction with hardware |
| 4 | Nguyen Thi Ly Linh | Developer | * Follow process of project and common rules. * Develop Green Hoa Lac web management system |
| QA | * Keeping all member on process and follow common rule * Controlling quality of the projects: time, function, risk. |
| 5 | Bui Manh Tri | Developer | * Follow process of project and common rules. * Develop Green Hoa Lac android application |
| 6 | Lam Duc Thang | Developer | * Follow process of project and common rules. * Develop Green Hoa Lac android application |
| 7 | Ho Quang Hao | Tester | * Follow process of project and common rules. * Test all cases following requirement. |
| QA | * Controlling quality of the projects: time, function, risk. |

### 2. Tools and infrastructures

#### 2.1. Hardware

Table 4: Hardware

|  |  |  |
| --- | --- | --- |
| Name | Image | Information |
| Module SIM800A MH |  | Designed for global market, SIM800 is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz and PCS 1900MHz. SIM800 features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4  Digital information:   * Voltage: 9-12V DC\_2A * SIM800A * Size: 75x75mm * Module developed from PCB of Module Sim 900A V1 * Be like Module Sim 900A V1, Module use sets of commands AT message, call phone * Because Sim800A does not have a mechanism selecting manually or automatically, Jum ( SW2 ) on PCB module Sim 900A V1 does not have impact when changing Jum |
| Hall sensor A04E |  | The Allegro® A1101-A1104 and A1106 Hall-effect switches are next generation replacements for the popular Allegro 312x and 314x lines of unipolar switches. The A110x family, produced with Bi CMOS technology, consists of devices that feature fast power-on time and low-noise operation. Device programming is performed after packaging, to ensure increased switch point accuracy by eliminating offsets that can be induced by package stress. Unique Hall element geometries and low offset amplifiers help to minimize noise and to reduce the residual offset voltage normally caused by device over molding, temperature excursions, and thermal stress.  Digital information:   * Voltage: 3.8- 24V * Temperature: -40- 85oC * Amperage: 25mA |
| Module DS1307+AT24C32 |  | Supporting to communicate with DS18B20  Digital information:   * Using DS1307 * Using EEPROM AT24C32 * Standard I2C |
| Module Relay 1 channel 5V- 220V/10A |  | Control devices through Relay  Digital information:   * Signal into control: DC5V * Default control:   + Turn off - 0, turn on – 1   * Changing J1, J0 to change control level * Output:   + Contact point relay 220V 10A  + NC : close  + NO : open   * Symbol power:   + VCC, GND are common power  + VSS+ , VSS- are power of Relay |

#### 2.2. Tools and software

Table 5: Tools and software

|  |  |  |
| --- | --- | --- |
| Tools | Image | Information |
| Window 10 |  | * Computer operating system developed and released by Microsoft * Used for: programming, hosting * Version: Windows 10 |
| GitHub |  | * Used for software development and other version control tasks, control version, source code |
| Google drive |  | * Storing documents like: software requirement specification, communication plan or risk management |
| Balsamiq Mockups |  | * Draw screen mockup for web and android application. * Version: 3.5.14 |
| Astah |  | * Draw diagrams for the whole system * Version: 7.1.0 |
| Robo 3T |  | * Manage system’s database server * Version: 1.1.1 |
| Fritzing |  | Fritzing is an open-source hardware initiative that makes electronics accessible as a creative material for anyone. We offer a software tool, a community website and services in the spirit of Processing and Arduino, fostering a creative ecosystem that allows users to document their prototypes, share them with others, teach electronics in a classroom, and layout and manufacture professional pcbs.   * Version: 0.9.3b |
| IDE Arduino |  | * Write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open source software * Version: |
| Android studio |  | * Build android application intelligent code * Version: 2.2.3 |
| Web storm |  | * Code web * Version: 2017.2 |
| Chrome |  | * Browser support to browsing internet |
| Office |  | * Project 2013 * Excel 2013 * Word 2013 * Power point 2013 |

### 3. Risk management plan

#### 3.1. Risks register

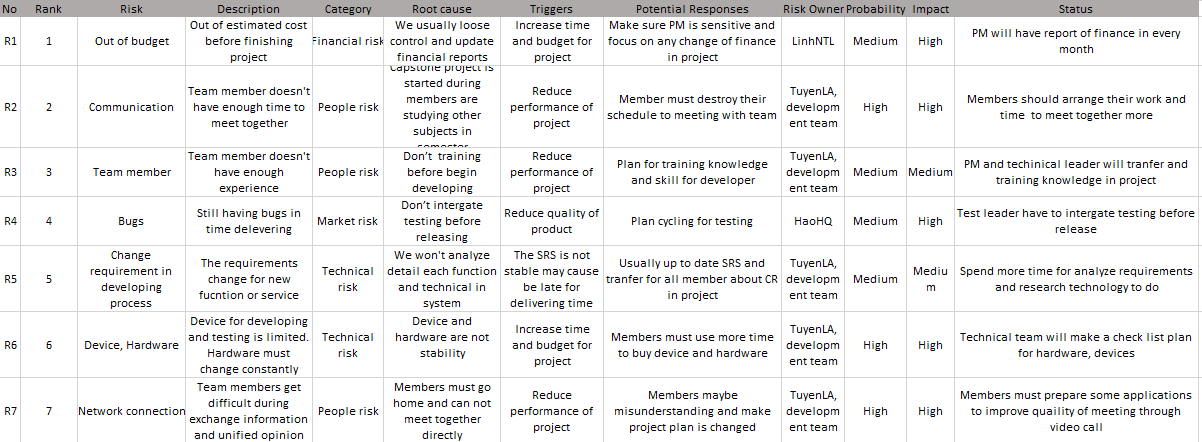


Figure 6: Risks registers

#### 3.2. Risk probability and impact

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | HIGH |  |  | R2  R6  R7 |
| **PROBABILITY** | MEDIUM |  | R3  R5 | R1  R4 |
|  | LOW |  |  |  |
|  |  | LOW | MEDIUM | HIGH |
|  |  |  | **IMPACT** |  |

Figure 7: Risk Probability and impact

### 4. Communication plan

#### 4.1. Project structure

Project Leader

Pham Quang Khang

**Supervisor**

Dev Team

Test leader

Ho Quang Hao

Bui Manh Tri

Lam Duc Thang

TechnicalLeader

Le Anh Tuyen

Main Supervisor

Project team

Ph.D. Phan Duy Hung

Le Anh Tuyen

QA

 Team

Pham Quang Khang

Nguyen Thi Ly Linh

**Green – Hoa Lac group**

Nguyen Thi Ly Linh

Ho Quang Hao

#### 4.2. Project communication

##### 4.2.1. Format, content, and level of detail of key project information

Table 6: Format, content, and level of detail

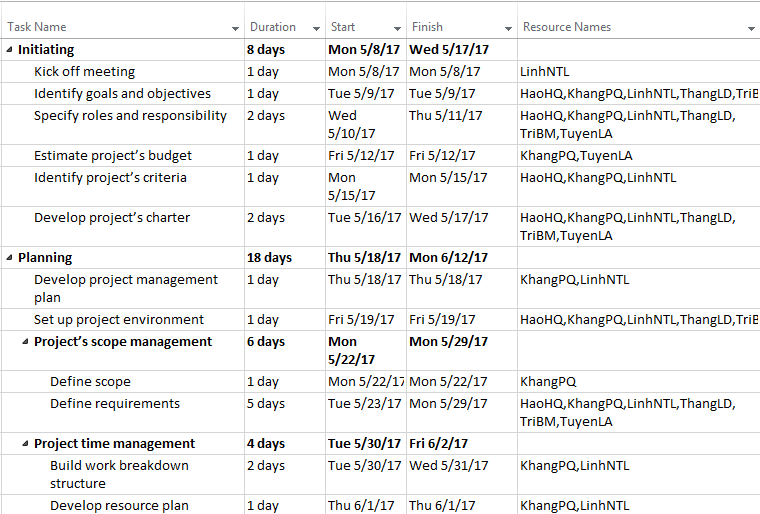
|  |  |  |  |
| --- | --- | --- | --- |
| **Information** | **Author** | **Receiver** | **Method/Technology** |
| Schedule updates | Project manager | Project team  Supervisor | Email  Group meeting |
| Project status | Project manager | Project team  Supervisor | Email  Group meeting |
| Agenda/Meeting Minutes | Project manager | Project team | Email |
| Issues | Test Leader | Project manager | Email  Group meeting |
| Status report | Project team | Project manager  Supervisor | Email |
| Project announcement | Project manager | Project team  Supervisor | Email  Instant message |
| User requirement | Project manager  Supervisor | Project team | Email  Hard copy |

##### 4.2.2. Project report

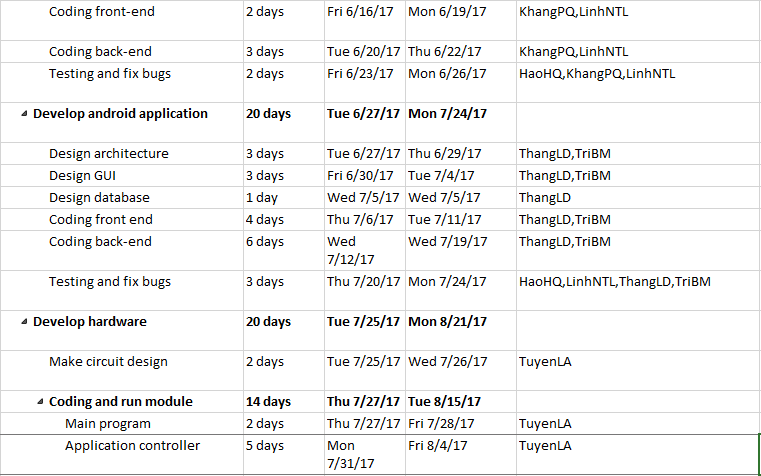
Table 7: Project report

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | Participants | Frequency/When | Method |
| Daily Meeting  Report | Author: Team leader (developer and tester)  Participants: all members in develop and test team | 17h30 each day (Monday- Friday) | Short Meeting |
| Weekly Status Report | Author: Pham Quang Khang  Distribution list: Nguyen Thi Ly Linh, Le Anh Tuyen, Ho Quang Hao | Friday | E-mail |
| Ad-hoc Discussion | Initiator: Ph.D. Phan Duy Hung or Le Anh Tuyen or Pham Quang Khang  Participants: all members | Event-based | Email  Skype  Meeting |

### 5. Project schedule







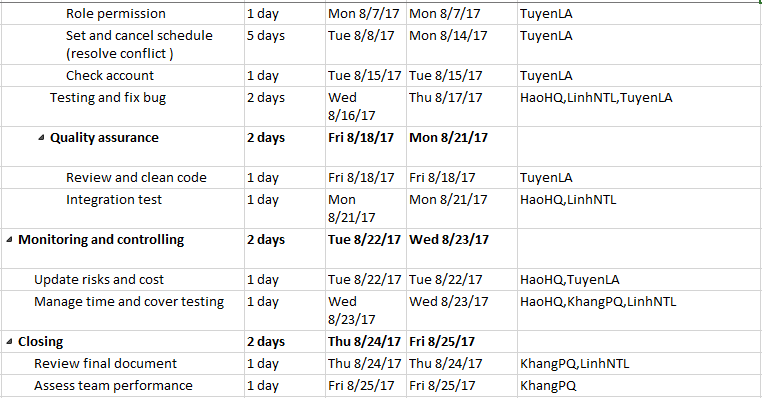


Figure 8: Project schedule

## III. System Requirement Specification

### 1. Green Hoa Lac Android Application and PCM

#### 1.1. Functional Requirement Specification

##### UC01 – Login

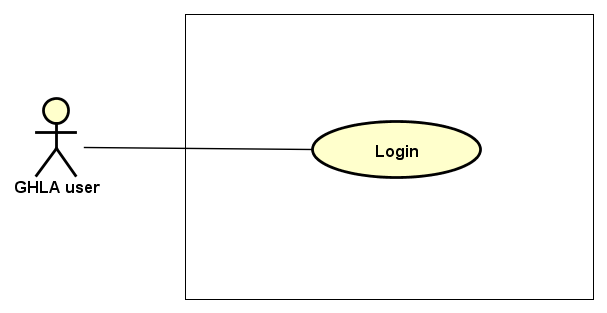
###### Screen Design

|  |  |
| --- | --- |
| Figure 47: Login application screen | Figure 48: Google sign in screen |
| Figure 49: Home screen | Figure 50: Navigation drawer screen |

Table 27: Login application screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Login | Button |  |  | Navigate to Google Sign In Page. |

###### Use case specification



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC01** | **Use Case Name** | | **Login** | |
| **Author** | | **ThangLD** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to login in the android application when he/she have had an account and his/her account is still active (or not blocked) | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Open GHLA | | | | |
| *2* | GHLA | Display Login screen with the following field:   * Login button | | | | |
| *3* | User | Click on Login button. | | | | |
| *4* | GHLA | Navigate to Google Sign In Page | | | | |
| *5* | User | Choose an account in list Google accounts or Enter other email and password for other account. | | | | |
| *6* | GHLA | Validate the account and then display Home screen | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 4 in the main flows, if users had logged for the first time and there is no internet connection at the present time, | |
| **Sub step** | **Actor** | **Action** |
| 4.1 | GHLA | Display Home Screen (Map) with data getting from the last login |

|  |  |  |
| --- | --- | --- |
| **AT2** | At step 5 in the main flows**,** if users enter wrong email account or email’s password, | |
| **Sub step** | **Actor** | **Action** |
| 5.1 | GHLA | Display an error with message "Couldn’t find your Google Account” or “Wrong password. Try again”, try again step 5. |

|  |  |  |
| --- | --- | --- |
| **AT3** | At step 5 in the main flows**,** although email account exists, it is blocked or in role “member” or does not exist in database. | |
| **Sub step** | **Actor** | **Action** |
| 5.1 | GHLA | Display a popup with message “Check your account or Internet connection!” |
| 5.2 | User | Click “Ok” |
| 5.3 | GHLA | Return step 2 |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR01 | GHLA user includes 3 objects: Admin, Manager, Member |
| BR02 | * For the first time logging the GHLA, user must have Internet connection for their android device * For the next time logging the GHLA,   + If having Internet connection, GHLA automatically load data (map, node, activity history) from the server’s database and send log to the server.  + If not having Internet connection, GHLA will use data in local device (realism) from the previous time logged. |

##### UC02 – View weather’s information

###### Screen Design

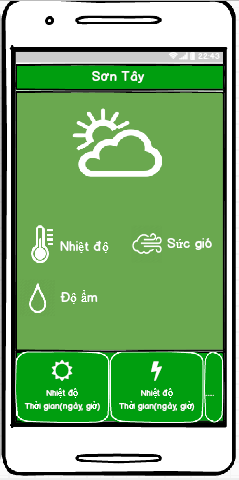


Figure 51: View weather's information screen

Table 28: View weather's information Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Location | Text |  |  | Name of province |
| 2 | Temperature | Text |  |  | Temperature at the present time. unit : oC |
| 3 | Speed of wind | Text |  |  | Speed of wind at the present time. unit: m/s |
| 4 | Amount of rain | Text |  |  | Amount of rain. unit: % |
| 5 | Weather forecast | List view |  |  | Weather forecast for the present day and the next 4 day |

###### Use case specification

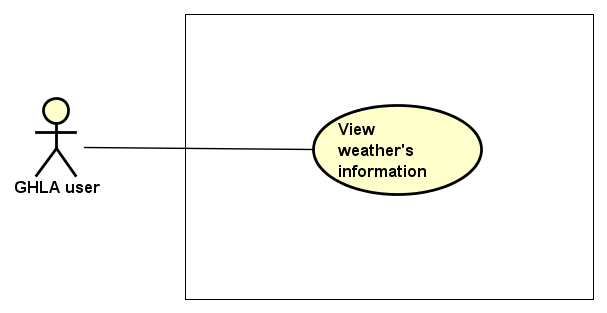


Figure 52: View weather's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC02** | **Use Case Name** | | **View weather’s information** | |
| **Author** | | **TriBM** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to view information of weather forecast | | | | |
| **Precondition** | | There is Internet connection | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Touch button on top left of the Home Screen | | | | |
| *2* | GHLA | Display a navigation drawer on the left of the screen | | | | |
| *3* | User | Select and touch “Weather’s information” item on the drawer | | | | |
| *4* | GHLA | Display “View weather’s information” screen | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 4 in the main flows, if there is no internet connection at the present time, | |
| **Sub step** | **Actor** | **Action** |
| 4.1 | GHLA | Display message “Check your account or internet connection” on popup at center |
| 4.2 | User | Touch “Ok” button on the popup |
| 4.3 | GHLA | Return Home Screen |

##### UC03 – View application’s information

###### Screen Design

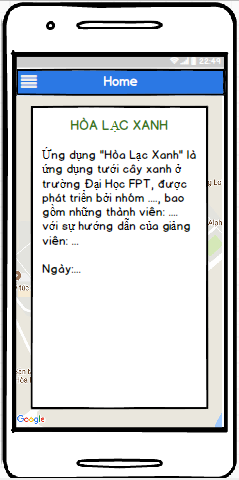


Figure 53: View application's information screen

Table 29: View application's information Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Information of project and instructors | Text |  |  |  |
| 2 | Team’s members | Text |  |  |  |

###### Use case specification

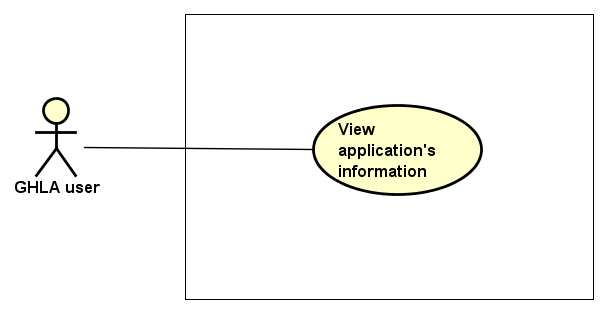


Figure 54: View application's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC03** | **Use Case Name** | | **View application’s information** | |
| **Author** | | **TriBM** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to view application information | | | | |
| **Precondition** | | User logged GHLA | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Touch button on top left of the Home Screen | | | | |
| *2* | GHLA | Display a navigation drawer on the left of the screen | | | | |
| *3* | User | Select and touch “Application information” item on the drawer | | | | |
| *4* | GHLA | Display “View application information” screen | | | | |

##### UC04 – Turn on node

###### Screen Design



Figure 55: Turn on node screen

Table 30: Turn on node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Mode | Radio group |  |  | There are 3 types of mode:   * Remote control * Schedule * Check and cancel schedule |
| 2 | Status | Drop down list |  |  | Status depends on type of mode. If mode is in Remote control, status contains:   * Turn on node * Turn off node |
| 3 | Duration | Drop down list |  |  | There are 4 items to select:   * Unlimited * 15 minutes * 20 minutes * 30 minutes |
| 4 | Time | Spinner |  |  | Time is disable in this case |
| 5 | Days in week | Check box |  |  | Days in week is disable in this case |
| 6 | Day | Button |  |  | Day is disable in this case |
| 7 | Send | Button |  |  | Send a request to PCM |

###### Use case specification

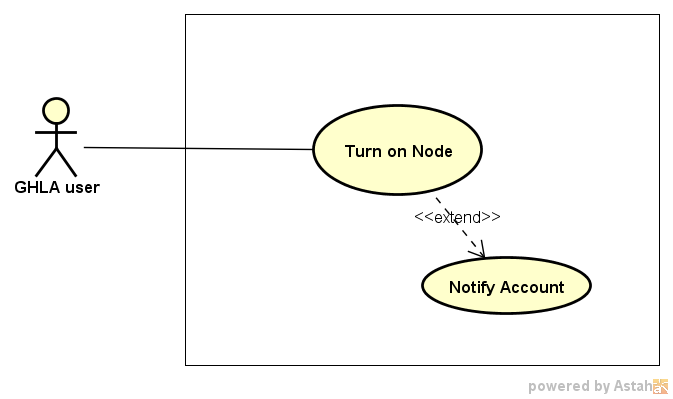


Figure 56: Turn on node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC04** | **Use Case Name** | | **Turn on node** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to turn on node | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control node” screen as default | | | | |
| *3* | User | * Select “Remote control” mode, * Select “Turn on” status * Touch “Send” button | | | | |
| *4* | GHLA | Display a confirm dialog with message “Are you sure perform this action!” | | | | |
| *5* | User | Touch “Ok” or “Cancel” button on the dialog | | | | |
| *6* | GHLA | If user select “Ok”, GHLA validate the phone number and send a bit string with the content user select in step 3 to PCM.  Else, return step 2. | | | | |
| *7* | Module GHL | * Turn on node immediately during the duration time and send * Check account and send message to user if account is lower than 10.000 VND | | | | |

##### UC05 – Turn off node

###### Screen Design



Figure 57: Turn off node screen

Table 31: Turn off node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Mode | Radio group |  |  | There are 2 types of mode:   * Remote control * Schedule * Check and cancel schedule |
| 2 | Status | Drop down list |  |  | Status depends on type of mode. If mode is in Remote control, status contains:   * Turn on node * Turn off node |
| 3 | Duration | Drop down list |  |  | Duration is disable in this case |
| 4 | Time | Spinner |  |  | Time is disable in this case |
| 5 | Days in week | Check box |  |  | Days in week is disable in this case |
| 6 | Day | Button |  |  | Day is disable in this case |
| 7 | Send | Button |  |  | Send a request to PCM |

###### Use case specification

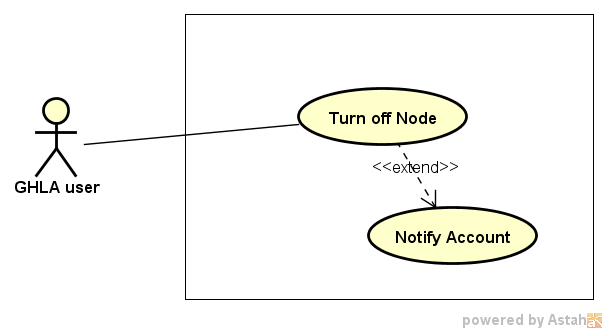


Figure 58: Turn off node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC05** | **Use Case Name** | | **Turn off node** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to turn off node | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control node” screen as default | | | | |
| *3* | User | * Select “Remote control” mode, * Select “Turn off” status * Touch “Send” button | | | | |
| *4* | GHLA | Display a confirm dialog with message “Are you sure perform this action!” | | | | |
| *5* | User | Touch “Ok” or “Cancel” button on the dialog | | | | |
| *6* | GHLA | If user select “Ok”, GHLA validate the phone number and send a bit string with the content user select in step 3 to PCM.  Else, return step 2. | | | | |
| *7* | PCM | * Check user and turn off node immediately * Check account and send message to user if account is lower than 10.000 VND | | | | |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR03 | GHLA users can turn off pump in a node immediately in accurate cases:   * They are people who turn on the pump * They are managers or admin |

##### UC06 – Set schedule for node

###### Screen Design

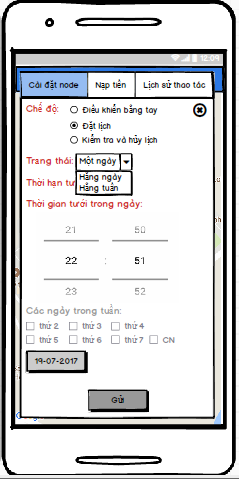


Figure 59: Set schedule for node

Table 32: Set schedule for node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Mode | Radio group |  |  | There are 3 types of mode:   * Remote control * Schedule * Check and cancel schedule |
| 2 | Status | Drop down list |  |  | Status depends on type of mode. If mode is in Schedule, status contains:   * One day * Daily * Weekly |
| 3 | Duration time | Drop down list |  |  | Duration time contains:   * Unlimited * 15 minutes * 20 minutes * 30 minutes |
| 4 | Time | Spinner |  |  | Time depends on type of mode.  If mode is in schedule, time get default at present time  Else, time is disable |
| 5 | Days in week | Check box |  |  | Day in week depends on type of mode and status.  If mode is in schedule and status is one day or daily, days in week is disable  Else, if mode is in schedule and status is weekly, days in week contains a group of check box with content from Monday to Sunday |
| 6 | Day | Button |  |  | If mode is in schedule and status is one day, Day is enable. |
| 7 | Send | Button |  |  | Send a request to PCM |

###### Use case specification

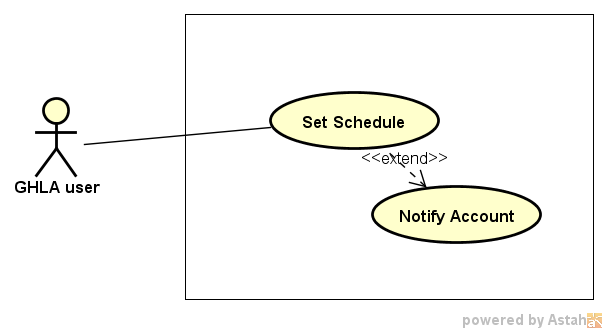


Figure 60: Set schedule for node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC06** | **Use Case Name** | | **Set schedule for node** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to set schedule for node | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control and Schedule node” screen as default | | | | |
| *3* | User | * Select “Schedule” mode, status, duration time, time, days in week, select a day. * Touch “Send” button | | | | |
| *4* | GHLA | Display a confirm dialog with message “Are you sure perform this action!” | | | | |
| *5* | User | Touch “Ok” or “Cancel” button on the dialog | | | | |
| *6* | GHLA | If user select “Ok”, GHLA validate the phone number and send a bit string with the content user select in step 3 to PCM.  Else, return step 2. | | | | |
| *7* | PCM | * Set up schedule according received message, validate it and then send a message with content “Set up schedule successfully” to user * Check account and send message if account is lower than 10.000 VND | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT** | At step 7 in the main flows, if user set up a schedule for the one day and time for this day passed, | |
| **Sub step** | **Actor** | **Action** |
| 7.1 | PCM | Send a message with content “The day or time passed. This schedule is not performed !” to user |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT2** | At step 7 in the main flows, if user set up a schedule that conflicts with other existed schedules | |
| **Sub step** | **Actor** | **Action** |
| 7.2 | PCM | Send a message with content “There are conflicts between existed schedules and your schedule! Try other set schedule!” to user |

##### UC07 – View schedules

###### Screen Design



Figure 61: View Schedules

Table 33: View schedules Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | List Schedules | Dialog |  |  | Show all schedules in a list view with 2 buttons:   * Delete all button * Cancel button |

###### Use case specification

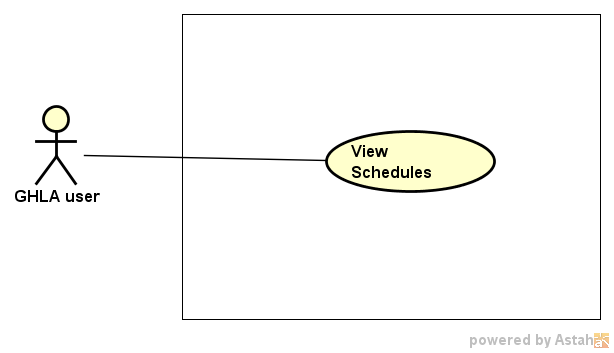


Figure 62: View schedules of node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC07** | **Use Case Name** | | **View schedules of node** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to view all schedules of nodes | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control and Schedule node” screen as default | | | | |
| *3* | User | Select “Check and cancel schedule” mode | | | | |
| *4* | GHLA | Send request that is view schedules of node to PCM | | | | |
| *5* | PCM | Send all schedules according to request of GHLA | | | | |
| *6* | GHLA | Display a list view with all schedules. | | | | |

##### UC08 – Cancel schedules of node

###### Screen Design

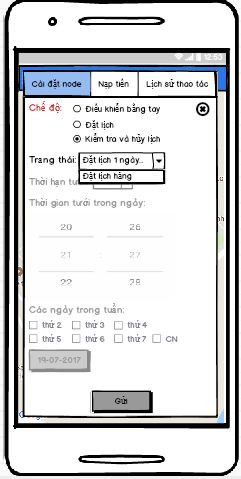


Figure 63: Cancel schedule for node screen

Table 34: Cancel schedule for node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Mode | Radio group |  |  | There are 3 types of mode:   * Remote control * Schedule * Check and cancel schedule |
| 2 | Status | Drop down list |  |  | Status depends on type of mode. If mode is in Check and cancel schedule, Status contain list of all schedules that users want to cancel. |
| 3 | Send | Button |  |  | Send a request to PCM |

###### Use case specification

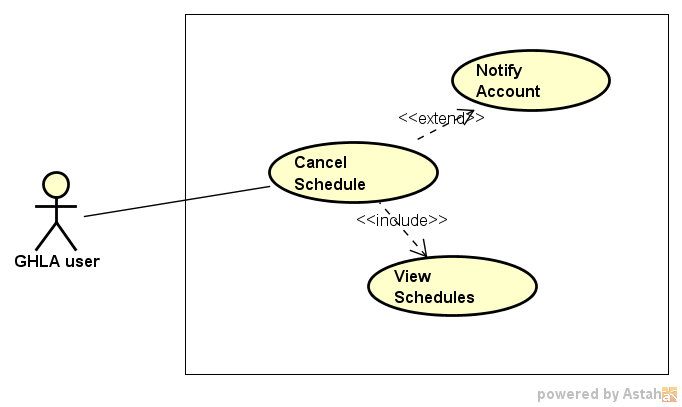


Figure 64: Cancel schedule for node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC08** | **Use Case Name** | | **Cancel schedule for node** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to cancel one or all schedules of node | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control and Schedule node” screen as default | | | | |
| *3* | User | Select “Check and cancel schedule” mode | | | | |
| *4* | GHLA | Send request that is view schedules of node to PCM | | | | |
| *5* | PCM | Send all schedules according to request of GHLA | | | | |
| *6* | GHLA | Display a list view with all schedules. | | | | |
| *7* | User | Select one schedule on List view or touch Delete button on the screen | | | | |
| *8* | GHLA | Display “Check and cancel schedule” screen with status includes all schedules you have selected to cancel | | | | |
| *9* | User | Touch “Send” button on the screen | | | | |
| *10* | GHLA | Display a confirm dialog with message contain all schedules user want to cancel | | | | |
| *11* | User | Select “Ok” Button | | | | |
| *12* | GHLA | Send a request cancel schedule to PCM | | | | |
| *13* | PCM | * Cancel all schedule according to the request and send a message to android device to notify for user * Check account and send message to user if account is lower than 10.000 VND | | | | |

##### UC09 – Recharge

###### Screen Design



Figure 65: Recharge screen

Table 35: Recharge Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Card number | Text box |  |  | It is a place used to fill card number |
| 2 | Recharge | Button |  |  |  |

###### Use case specification

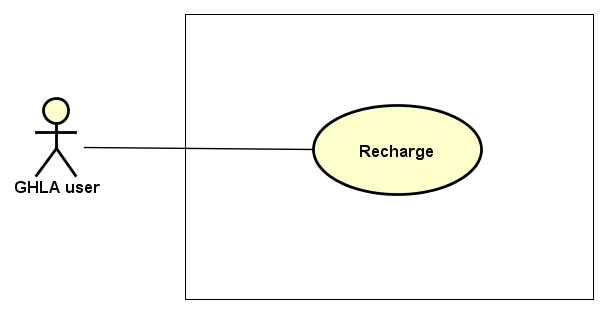


Figure 66: Recharge Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC08** | **Use Case Name** | | **Recharge** | |
| **Author** | | **TriBM, TuyenLA** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to recharge for subscriber identity module of node | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control and Schedule node” screen as default | | | | |
| *3* | User | Select “Recharge and Check account” tab | | | | |
| *4* | GHLA | Display “Recharge and Check account” screen | | | | |
| *5* | User | * Fill card number on text box on the screen * Click “Recharge” button | | | | |
| *6* | GHLA | Recharge money for the above node through Switchboard | | | | |

##### UC10 – View node’s history

###### Screen Design



Figure 67: View node's history screen

Table 36: View note's history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Node’s history | List view |  |  | The list view contains items including actor, description and time. |
| 2 | View History | Button |  |  |  |
| 3 | Search | Text field |  |  |  |
| 4 | Synchronize node’s history | Button |  |  |  |

###### Use case specification

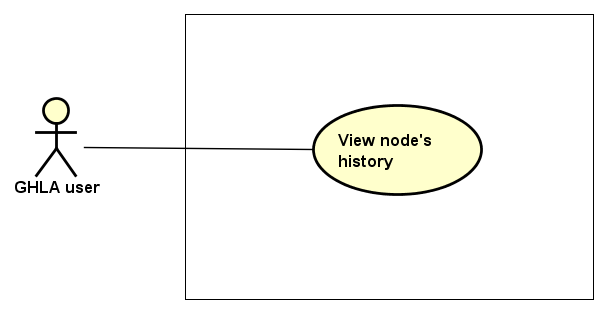


Figure 68: View node's history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC10** | **Use Case Name** | | **View node’s history** | |
| **Author** | | **ThangLD** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to view history of node they select | | | | |
| **Precondition** | | There is an Internet connection | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control node” screen as default | | | | |
| *3* | User | Select “History” tab | | | | |
| *4* | GHLA | Display “History” screen | | | | |
| *5* | User | Select and touch “View Log” button on the screen | | | | |
| *6* | GHLA | Load activity history from the database server and display on the list view | | | | |

##### UC11 – Search node’s history

###### Screen Design



Figure 69: Search node's history

Table 37: Search node's history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Node’s history | List view |  |  | The list view contains items including actor, description and time. |
| 2 | View History | Button |  |  |  |
| 3 | Search | Text filed |  |  |  |
| 4 | Synchronize node’s history | Button |  |  |  |

###### Use case specification

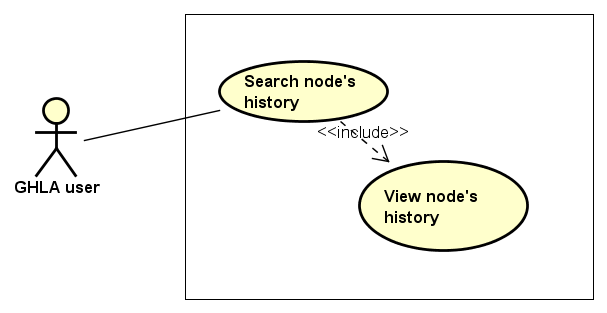


Figure 70: Search node's history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC11** | **Use Case Name** | | **Search node’s history** | |
| **Author** | | **TriBM** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to search node’s history | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control node” screen as default | | | | |
| *3* | User | Select “History” tab | | | | |
| *4* | GHLA | Display “History” screen | | | | |
| *5* | User | Select and touch “View Log” button on the screen | | | | |
| *6* | GHLA | Load activity history from the database server and display on the list view | | | | |
| *7* | User | Enter content in “Search” text field | | | | |
| *8* | GHLA | Return a list view that satisfies the above content | | | | |

##### UC12 – Synchronize nodes

###### Screen Design



Figure 71: Synchronize nodes Screen

Table 38: Synchronize nodes Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Synchronize | Popup |  |  | Confirm dialog to synchronize nodes on the map |

###### Use case specification

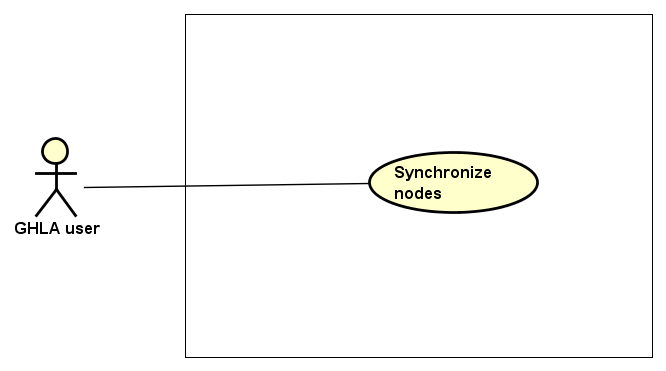


Figure 72: Synchronize nodes Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC12** | **Use Case Name** | | **Synchronize nodes** | |
| **Author** | | **ThangLD** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows a user to be able to synchronize nodes with the database server. | | | | |
| **Precondition** | | There is an internet connection | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Touch button on top left of the Home Screen | | | | |
| *2* | GHLA | Display a navigation drawer on the left of the screen | | | | |
| *3* | User | Select and touch “Synchronize nodes” item on the drawer | | | | |
| *4* | GHLA | Display a confirm dialog on the screen with content “Are you sure to synchronize nodes?” following by two button “Agree” and “Disagree” | | | | |
| *5* | User | Select and touch one of the two above buttons | | | | |
| *6* | GHLA | If user select “Agree”, GHLA will synchronize nodes with database in server and reload Home page with map.  Else, return step 2. | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 3 in the main flows, if there is no internet connection at the present time, | |
| **Sub step** | **Actor** | **Action** |
| 4.1 | GHLA | Display Home Screen (Map) with a dialog message “You need have Internet connection to perform this action!” following a button “Ok” |
| 4.2 | User | Select and touch “Ok” button |
| 4.3 | GHLA | Display Home Screen (Map) |

##### UC13 – Synchronize node’s history

###### Screen Design



Figure 73: Synchronize node's history Screen

Table 39: Synchronize node's history Screen Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Node’s history | List view |  |  | The list view contains items including actor, description and time. |
| 2 | View history | Button |  |  |  |
| 3 | Search | Text field |  |  |  |
| 3 | Synchronize node’s history | Button |  |  |  |

###### Use case specification

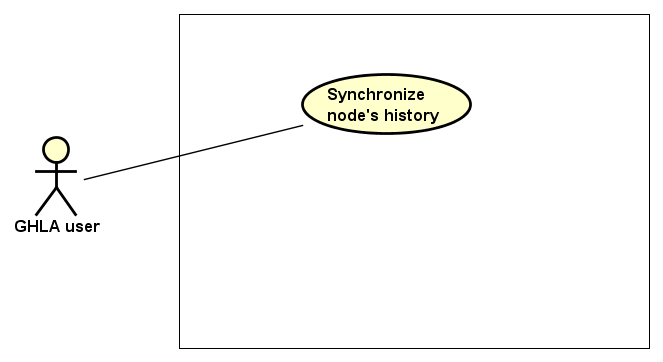


Figure 74: Synchronize node's history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC13** | **Use Case Name** | | **Synchronize node’s history** | |
| **Author** | | **ThangLD** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to synchronize node’s history | | | | |
| **Precondition** | | There is an internet connection | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Select and touch a node on Home Screen | | | | |
| *2* | GHLA | Display “Control node” screen as default | | | | |
| *3* | User | Select “History” tab | | | | |
| *4* | GHLA | Display “History” screen | | | | |
| *5* | User | Select and touch “Synchronize node’s history” button on the screen | | | | |
| *6* | GHLA | Synchronize all node’s histories that are dependent on the local device and are not dependent on the database server to the database server.  Then, display a confirm dialog with message “Synchronize successful!” following a button “Ok” | | | | |
| *7* | User | Select and touch “Ok” button | | | | |
| *8* | GHLA | Display “History” screen | | | | |

##### UC14 – Logout

###### Screen Design

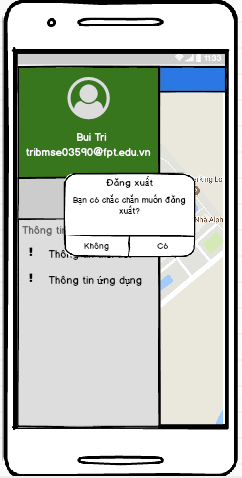


Figure 75: Logout Screen

Table 40: Logout Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Confirm | Dialog |  |  | Confirm dialog |

###### Use case specification

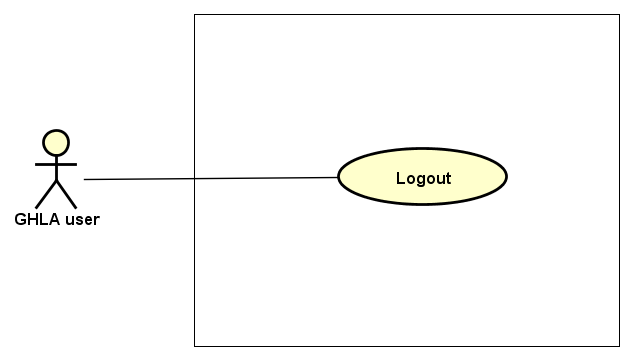


Figure 76: Logout Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC14** | **Use Case Name** | | **Logout** | |
| **Author** | | **ThangLD** | **Version** | **1.0** | **Date** | **22/06/2017** |
| **Actor** | | GHLA User | | | | |
| **Description** | | The function allows an user to be able to logout the application | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Touch button on top left of the Home Screen | | | | |
| *2* | GHLA | Display a navigation drawer on the left of the screen | | | | |
| *3* | User | Select and touch “Logout” item on the drawer | | | | |
| *4* | GHLA | Display confirm dialog in the screen with 2 button “Agree” and “Disagree” | | | | |
| *5* | User | Select “Agree” or “Disagree” button and touch it | | | | |
| *4* | GHLA | If user select “Agree” button, logout user from the application  Else, return step 2 | | | | |

#### 1.2. Non- Functional Requirement Specification

Table 41: Nonfunctional requirements for GHLA

|  |  |  |
| --- | --- | --- |
| # | Nonfunctional requirements | Description |
| 1 | Security | Users have to sign in with Google’s accounts |
| 2 | Availability | The user session timed out is dependent on Google |
| 3 | Usability | Language: Vietnamese |

### 2. Green Hoa Lac Web Management System

#### 2.1. Functional Requirement Specification

##### 2.1.1. UC01 – Login

###### 2.1.1.1. Screen Design

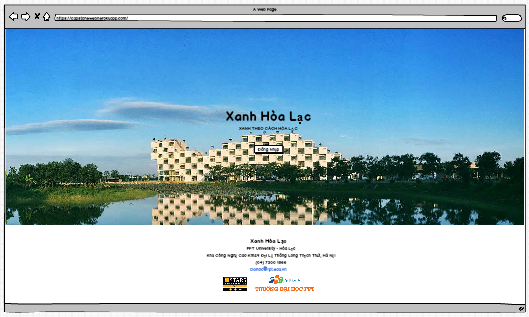


Figure 9: Login Screen

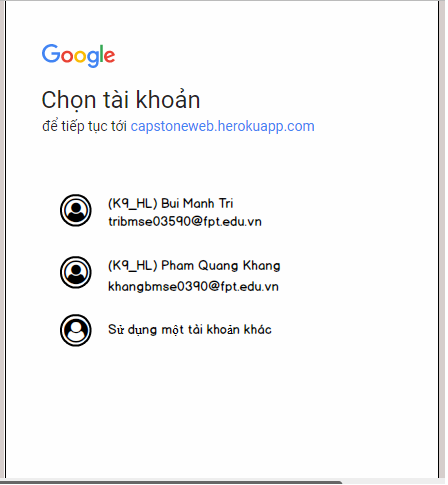


Figure 10: Google Sign in Screen

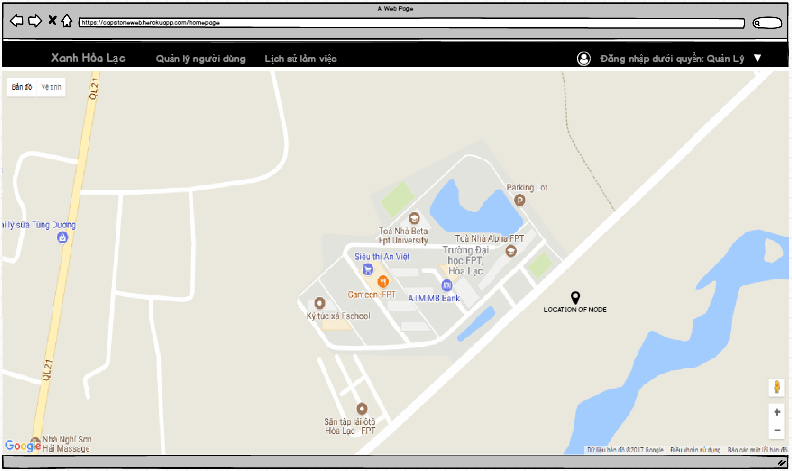


Figure 11: Home Page

Table 8: Login Screen Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Login | Button |  |  | Navigate to Google Sign In Page. |

###### Use case specification

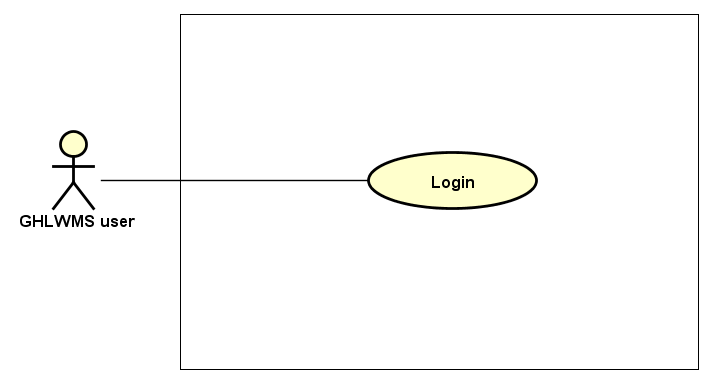


Figure 12: Login Use Case Diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC01** | **Use Case Name** | | **Login** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLMS User | | | | |
| **Description** | | The function allows users to be able to login in the website when he/she have had an account and his/her account is still active (or not blocked). | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Type URL: <https://capstoneweb.herokuapp.com>  into location field of internet browser and then press enter | | | | |
| *2* | GHLWMS | Display Login screen with the following field:   * Login button | | | | |
| *3* | User | Click on Login button. | | | | |
| *4* | GHLWMS | Navigate to Google Sign In Page | | | | |
| *5* | User | Choose an account in list Google accounts or Enter other email and password for other account. | | | | |
| *6* | GHLWMS | Validate the account and then display Home screen | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 5 in the main flows**,** if users enter wrong email account or email’s password, | |
| **Sub step** | **Actor** | **Action** |
| 5.1 | GHLWMS | Display an error with message "Couldn’t find your Google Account” or “Wrong password. Try again”, try again step 5. |

|  |  |  |
| --- | --- | --- |
| **AT2** | At step 5 in the main flows**,** although email account exists, it is blocked or in role “member” or does not exist in database. | |
| **Sub step** | **Actor** | **Action** |
| 5.1 | GHLWMS | Return step 2. |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR04 | GHLWMS user includes 2 types of role: Admin and Manager. |
| BR05 | There are 3 types of users which are managed in GHLWMS including Admin, Manager and Member |

##### UC02 – View user’s information

###### Screen Design

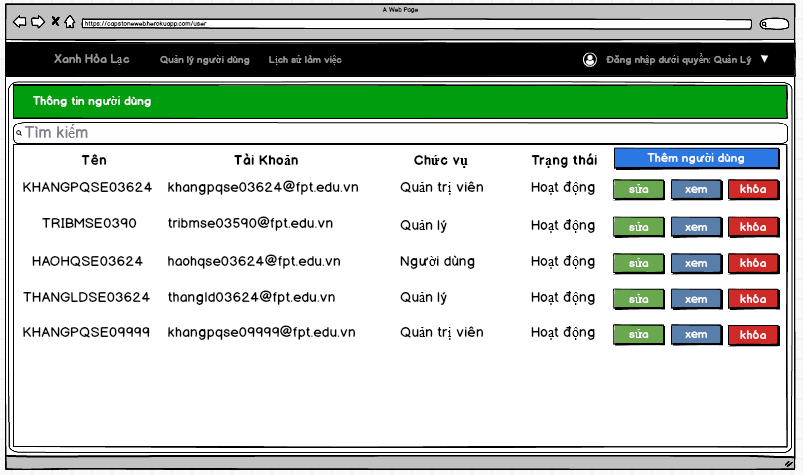


Figure 13: View user's information Screen

Table 9: View user information screen information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | List all users | Table |  |  | The table contains 5 columns: username, account, role, status and action (update role, view log, block (button)) |

###### Use case specification

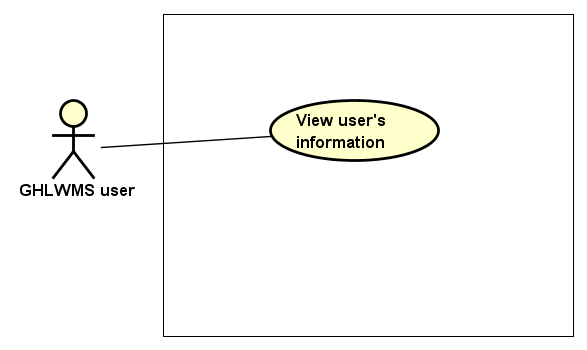


Figure 14: View user's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC02** | **Use Case Name** | | **View user’s information** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to view user’s information in GHLWMS with the following fields: username, account, role or status | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display list all accounts in “User management” page in a table with the following columns groups:   * (Username, account, role, status) * (Update role, view log, block) | | | | |

##### UC03 – Add user

###### Screen Design

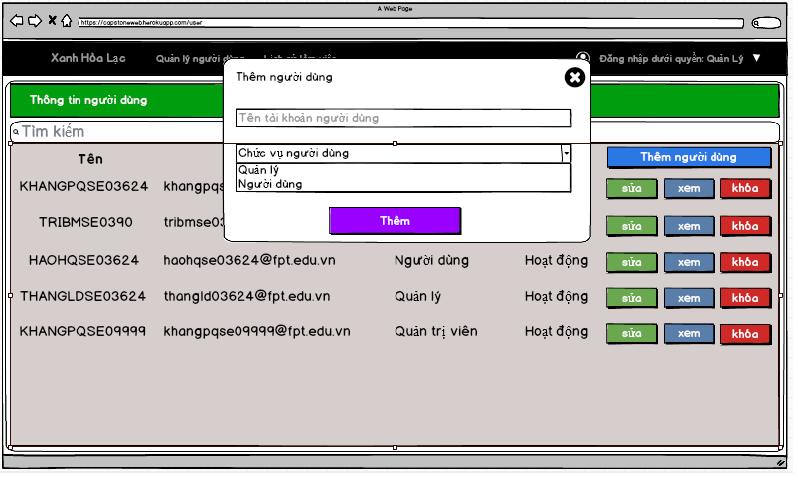


Figure 15: Add user screen

Table 10: Add user Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | User name | Text box | Yes |  | mail |
| 2 | Role of user | Drop down list | Yes |  | There 2 types of role:   * Member * Manager |
| 3 | Add | Button |  |  |  |

###### Use case specification

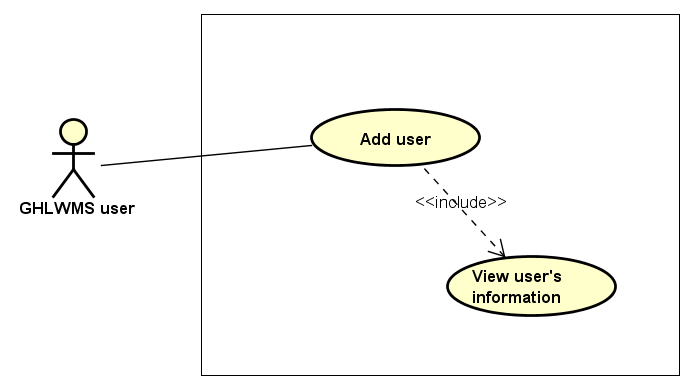


Figure 16: Add user Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC03** | **Use Case Name** | | **Add user** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows users to be able to add accounts to GHLWMS. | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Click button “Add” on right top of the screen. | | | | |
| *4* | GHLWMS | Display Add user popup with the following fields:   * User name (text box) * Type of role (drop down list) * Add (button) | | | | |
| *5* | User | * Enter User name * Select type of role * Click on Add button | | | | |
| *6* | GHLWMS | * Validate User name, role and then add to the database * Display message “Add new user successfully” * Hide the popup and reload list users in User management page | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 4 in the main flows**,** if users do not enter email account or select type of role | |
| **Sub step** | **Actor** | **Action** |
| 4.1 | GHLWMS | Display an error with message "User name or role must be filled”, try again step 4 |

|  |  |  |
| --- | --- | --- |
| **AT2** | At step 6 in the main flows**,** if users enter wrong syntax of email account | |
| **Sub step** | **Actor** | **Action** |
| 6.1 | GHLWMS | Display an error with message "Wrong username”, try again step 4. |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR06 | Domain of mail is one of the following:   * Gmail.com * Fpt.edu.vn * Fe.edu.vn * Gmail.com.vn |

##### UC04 – Search user’s information

###### Screen Design

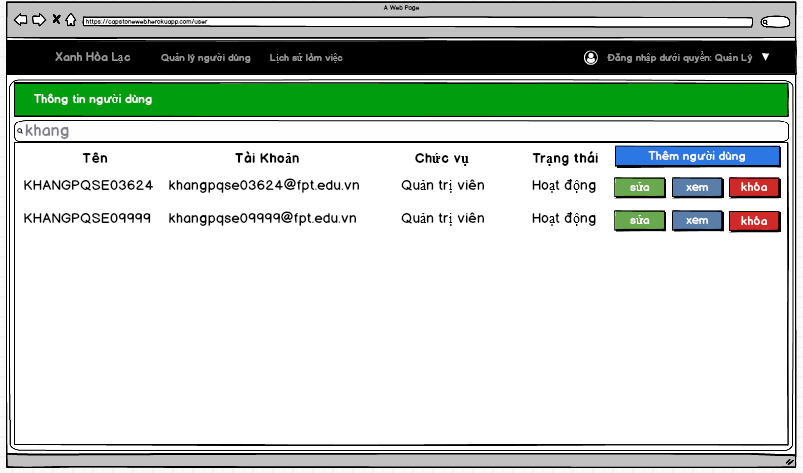


Figure 17: Search user's information screen

Table 11: Search user's information Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Filter | Text box |  |  | Filter by username account, role or status. |
| 2 | List user’s information | Table |  |  | The table contains 5 columns: username, account, role, status and action (update role, view log, block) |

###### Use case specification

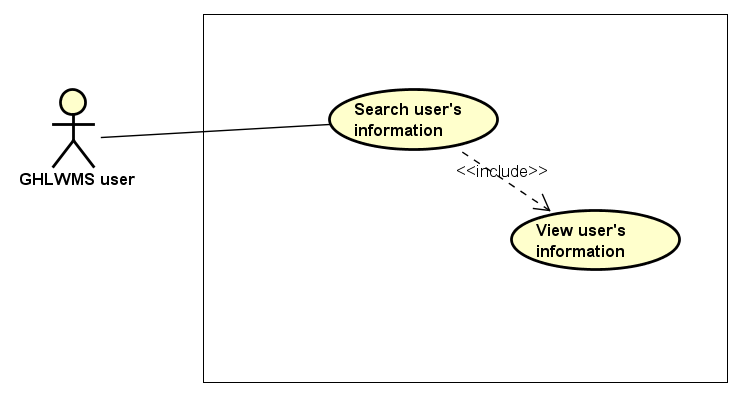


Figure 18: Search user's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC04** | **Use Case Name** | | **Search user’s information** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to search user ’s information in GHLWMS by username, account, role or status | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Enter information in field “Filter”. | | | | |
| *4* | GHLWMS | Display list accounts that satisfy the above information in the table with the following field groups:   * (Username, account, role, status) * (Update role, view log, block) | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 3 in the main flows**,** if users enter characters that do not match any information | |
| **Sub step** | **Actor** | **Action** |
| 3.1 | GHLWMS | Display “No result is found”. |

##### UC05 – Update user’s role

###### Screen Design

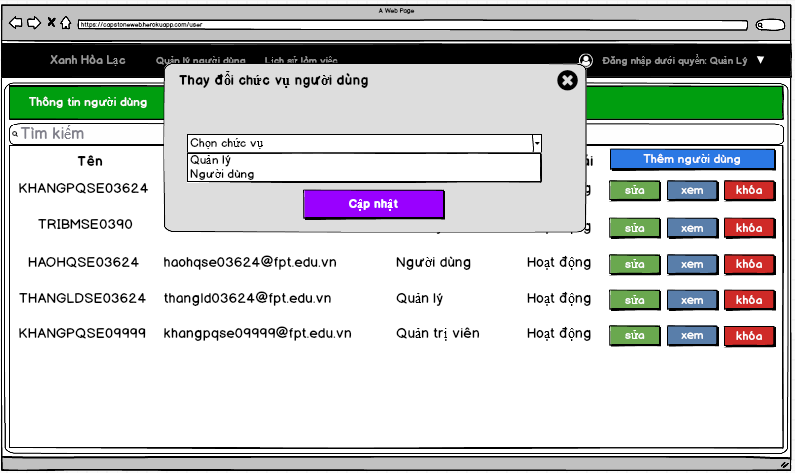


Figure 19: Update user's role screen

Table 12: Update user's role Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Type of role | Drop down list |  |  | There are 2 type of roles:   * Manager * Member |
| 2 | Update | Button |  |  |  |

###### Use case specification

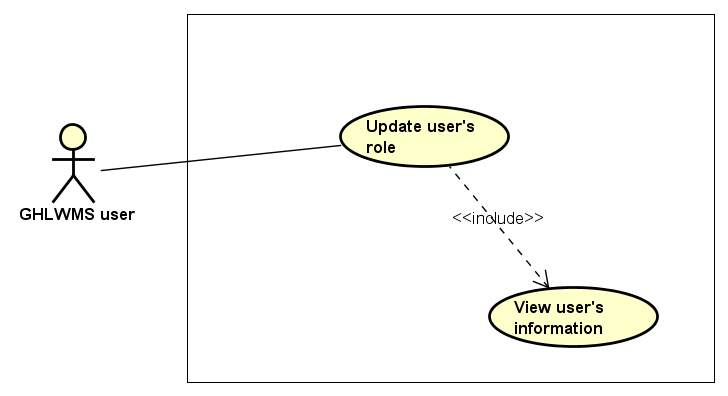


Figure 20: Update user's role Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC05** | **Use Case Name** | | **Update user’s role** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to update other user’s role | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Select an account and click “Update role” button | | | | |
| *4* | GHLWMS | Display “Update role” popup | | | | |
| *5* | User | * Select a role from the drop down list role * Click “Update” button | | | | |
| *6* | GHLWMS | * Validate and update role in the database * Display message “Update successfully” in popup * Hide popup, reload list of users in User management page | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 6 in the main flows**,** if users selected account of their own or account of others who are in the same role or Admin | |
| **Sub step** | **Actor** | **Action** |
| 6.1 | GHLWMS | Display error with message “Access denied” |

|  |  |  |
| --- | --- | --- |
| **AT2** | At step 6 in the main flows**,** if users did not select a role in the drop down list and then click “Update” button | |
| **Sub step** | **Actor** | **Action** |
| 6.1 | GHLWMS | Display error with message “Role must be chosen” |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR07 | GHLWMS users cannot update user’s role in some cases:   * Update their own role * Update other user’s role who has the same role with user working now. * Update role of Admin |

##### UC06 – View user’s history

###### Screen Design

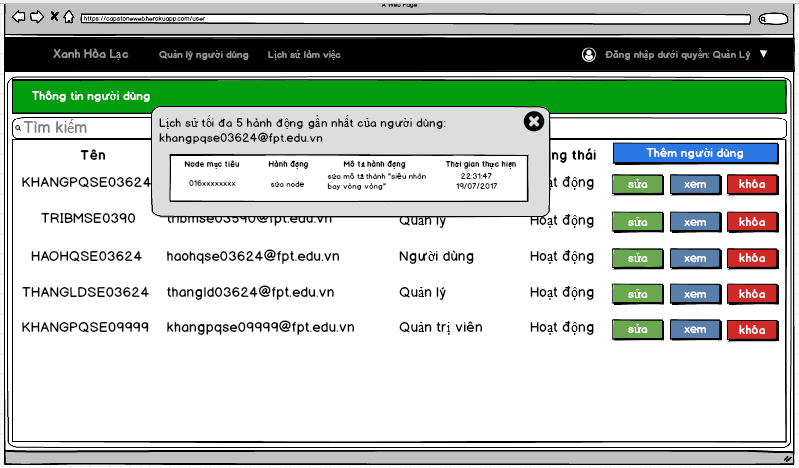


Figure 21: View user's history screen

Table 13: View user's history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | List 5 previous activities of the account | Popup |  |  | Content of list is divided into 4 main columns: target node, activity, description, time |

###### Use case specification

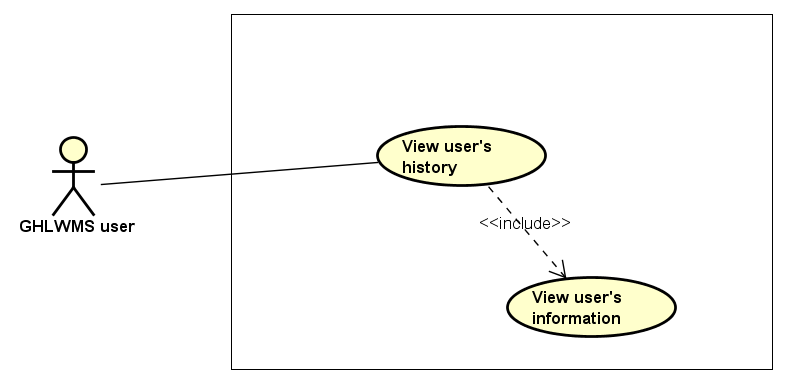


Figure 22: View user's history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC06** | **Use Case Name** | | **View user’s history** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows a user to be able to view activities history of all users. | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Select an account and click “View history” button | | | | |
| *2* | GHLWMS | Display list 5 previous activities with information about: (target node, action, description, time) of the account in a popup | | | | |

##### UC07 – Lock user

###### Screen Design

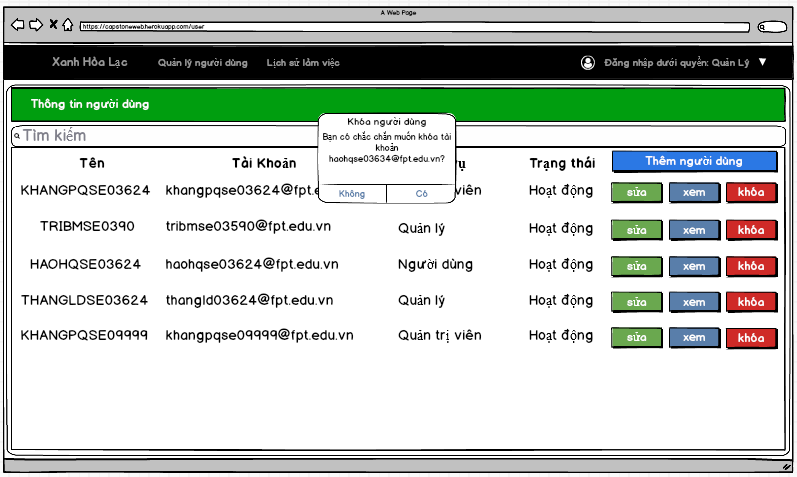


Figure 23: Lock user screen

Table 14: Log user Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Lock user | Dialog |  |  | Confirm dialog |

###### Use case specification

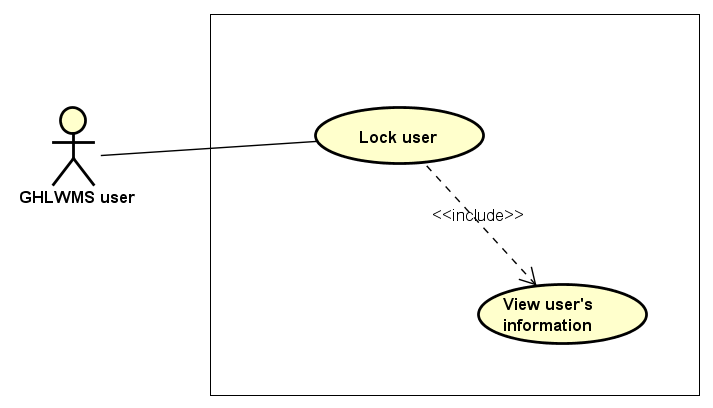


Figure 24: Lock user Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC07** | **Use Case Name** | | **Lock user** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to lock other user | | | | |
| **Precondition** | | Login GHLWMS successfully | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Select an account and click “Lock” button | | | | |
| *4* | GHLWMS | Display “Confirm to lock user” popup | | | | |
| *5* | User | Click button Agree/ Disagree on the popup | | | | |
| *6* | GHLWMS | If click “Disagree”, stay at “User management” page  Else:   * Validate and update status in the database * Display message “Lock the user successfully” in popup * Hide popup, reload list of users in User management page | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 6 in the main flows**,** if users selected account of their own or account of others who are in the same role or Admin | |
| **Sub step** | **Actor** | **Action** |
| 6.1 | GHLWMS | Display error with message “Access denied” |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR08 | GHLWMS users cannot lock an account in some cases:   * Lock their own accounts * Lock other user’s role who has the same role with user working now. * Lock role of Admin |

##### UC08 – Unlock user

###### Screen Design

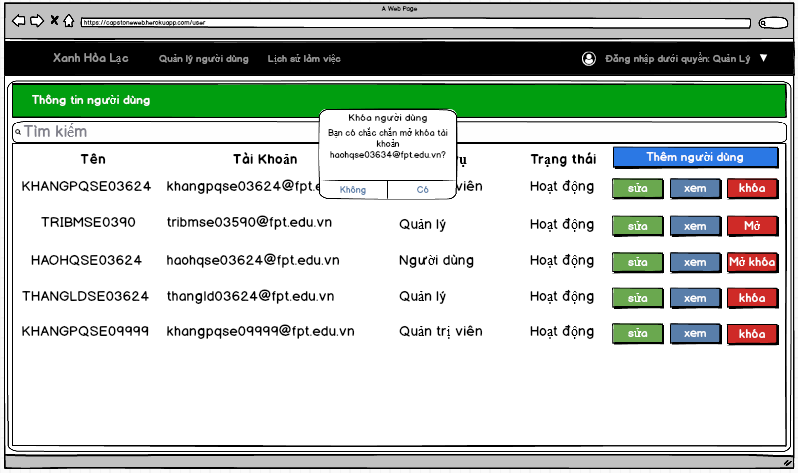


Figure 25: Unlock user screen

Table 15: Unlock user Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Unlock user | Dialog |  |  | Confirm dialog |

###### Use case specification

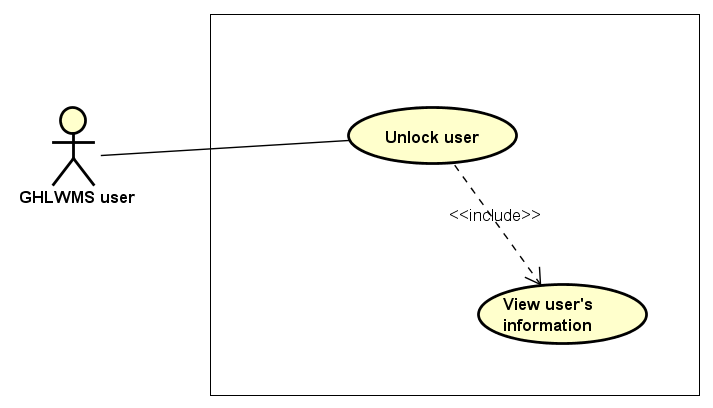


Figure 26: Unlock user Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC08** | **Use Case Name** | | **Unlock user** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to unlock other user | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “User management” on the navigation bar | | | | |
| *2* | GHLWMS | Display “User management” page | | | | |
| *3* | User | Select an account and click “Unlock” button | | | | |
| *4* | GHLWMS | Display “Confirm to unlock user” popup | | | | |
| *5* | User | Click button Agree/ Disagree on the popup | | | | |
| *6* | GHLWMS | If click “Disagree”, stay at “User management” page  Else:   * Validate and update status in the database * Display message “Unlock the user successfully” in popup * Hide popup, reload list of accounts in User management page | | | | |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR09 | GHLWMS users can unlock an account in some cases:   * Admin can unlock all type of account. * Manager can unlock other manager’s accounts and other members |

##### UC09 – View activity history

###### Screen Design

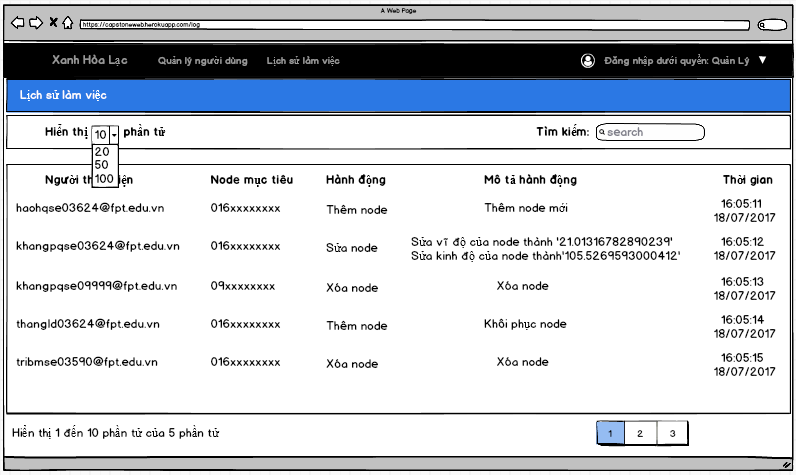


Figure 27: View activity history screen

Table 16: View activity history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | List activity history | Table |  |  | The table contains 5 columns: user, target node, activity, description, time |

###### Use case specification

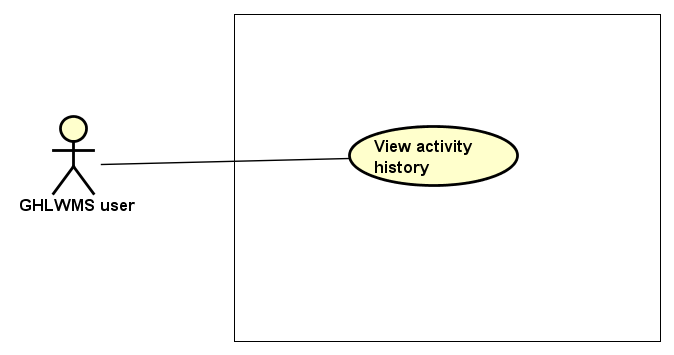


Figure 28: View activity history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC09** | **Use Case Name** | | **View activity history** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to view activity history in GHLWMS with the following fields: user, target node, action, description, time | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Activity history” on the navigation bar | | | | |
| *2* | GHLWMS | Display “Activity history” page with a table which has default top 10 previous actions. | | | | |

##### UC10 – Search activity history

###### Screen Design

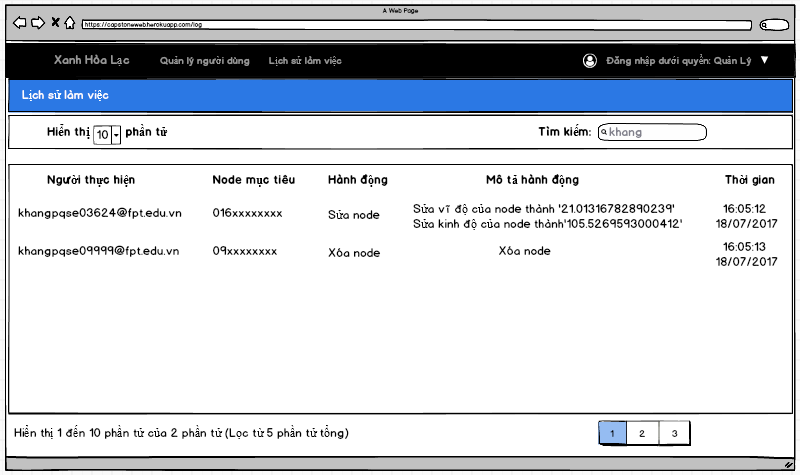


Figure 29: Search activity history screen

Table 17: Search activity history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Search | Text field |  |  | Filter by user, target node, action, description, time. |
| 2 | List activity history | Table |  |  | The table contains 5 columns: user, target node, activity, description, time |

###### Use case specification

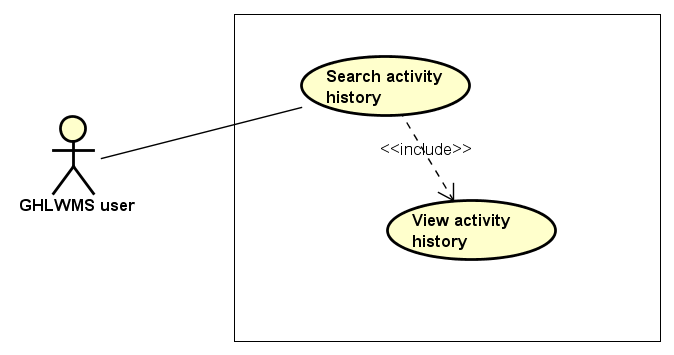


Figure 30: Search activity history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC10** | **Use Case Name** | | **Search activity history** | |
| **Author** | | **LinhNTL** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to search activity history in GHLWMS by user, target node, action, description, time | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Activity history” on the navigation bar | | | | |
| *2* | GHLWMS | Display “Activity history” page | | | | |
| *3* | User | Enter information in “Search” field | | | | |
| *4* | GHLWMS | Display list histories that satisfy the above information in the table with the following fields: user, target node, action, description, time | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 3 in the main flows**,** if users enter characters that do not match any information | |
| **Sub step** | **Actor** | **Action** |
| 3.1 | GHLWMS | Display “No result is found”. |

##### UC11 – Add new node

###### Screen Design

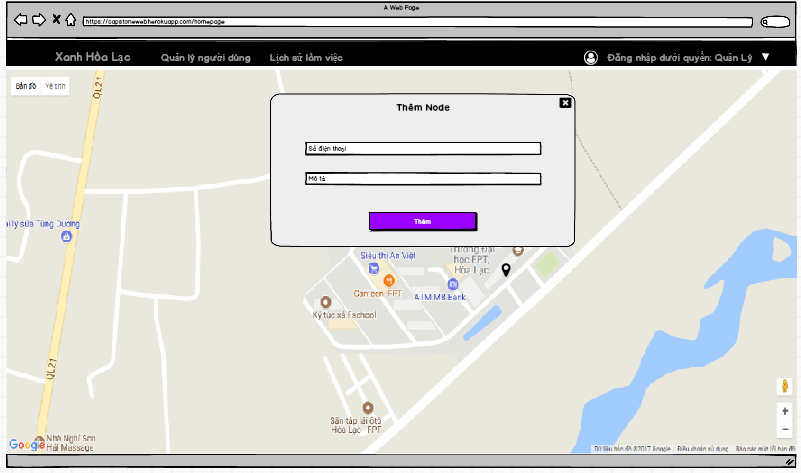


Figure 31: Add new node screen

Table 18: Add new node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Telephone number | Text field | Yes | 15 | Must be a string of number |
| 2 | Description | Text field | No |  |  |
| 3 | Add | Button |  |  |  |

###### Use case specification

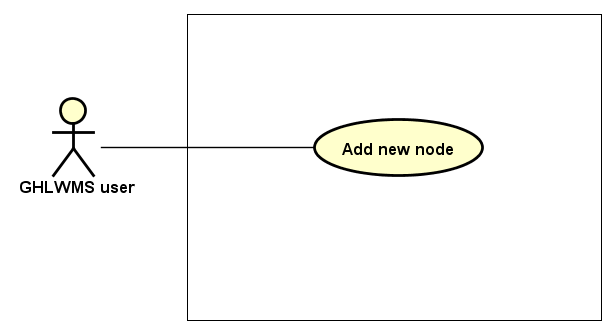


Figure 32: Add new node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC11** | **Use Case Name** | | **Add new node** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows a user to be able to add new node to GHLWMS. | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a position for new node and click there | | | | |
| *4* | GHLWMS | Display Add node popup with the following fields:   * Telephone number (text field) * Description (text field) * Add (button) | | | | |
| *5* | User | * Enter telephone number * Enter description * Click on Add button | | | | |
| *6* | GHLWMS | * Validate telephone number and then add new node to the database * Display message “Add new node successfully” * Hide the popup and reload “Green Ho La” page (or home page) | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 6 in the main flows**,** if telephone number is invalid syntax, | |
| **Sub step** | **Actor** | **Action** |
| 6.1 | GHLWMS | Display an error with message "wrong phone number”, try again step 5 |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR10 | Telephone number must be in the following syntax:   * First number is 0 or the two first numbers is 84 * If first number is 0: there must have minimum 10 numbers after the first * If the two first numbers is 84: there must have minimum 11 numbers after the two first |
| BR11 | Does not exist 2 nodes that have the same location. |

##### UC12 – Update node’s position

###### Screen Design

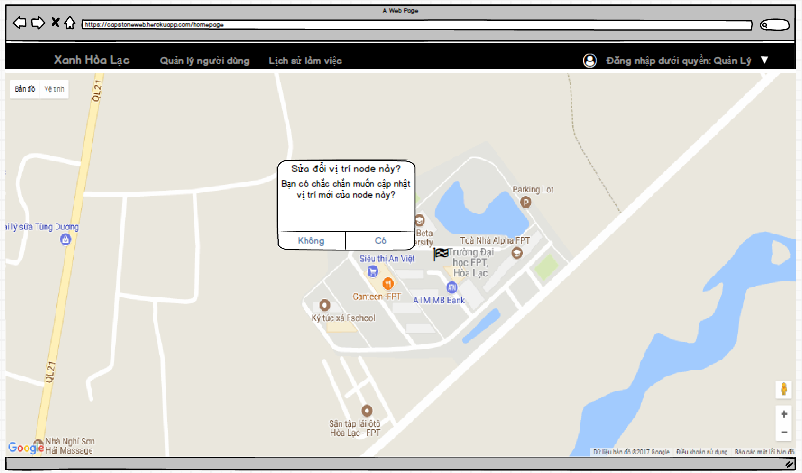


Figure 33: Update node's position screen

Table 19: Update node’s position Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Update node’s position | Dialog |  |  | Confirm dialog |

###### Use case specification

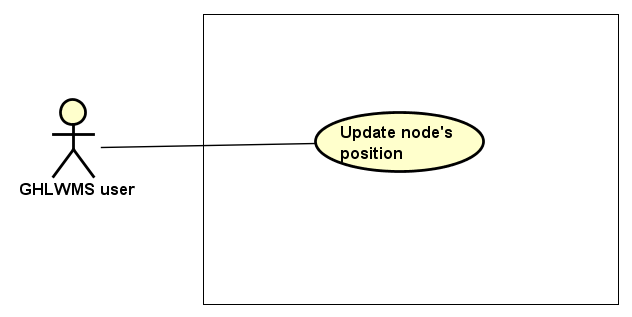


Figure 34: Update node's position Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC12** | **Use Case Name** | | **Update node’s position** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows a user to be able to update node’s position in GHLWMS. | | | | |
| **Precondition** | | Exist at least one node in GHLWMS | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a node, move the node to new position in the map. | | | | |
| *4* | GHLWMS | Display “Confirm to update position of node” with the following fields:   * Agree button * Disagree button | | | | |
| *5* | User | Click button Agree/ Disagree on the popup | | | | |
| *6* | GHLWMS | * If users enter “Disagree” button, hide the popup * Else :   + Update new position of the node in the database  + Display “Update node’s position successfully”  + Stay at “Green Ho La” page (or home page) | | | | |

##### UC13 – View node’s information

###### Screen Design

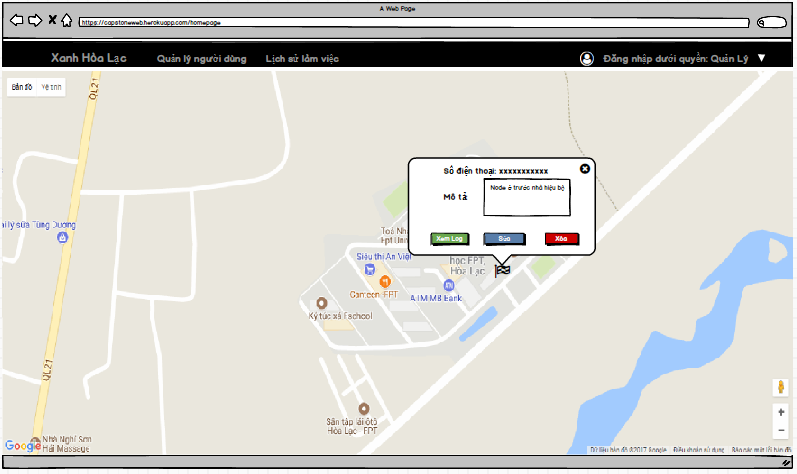


Figure 35: View node's information screen

Table 20: View node's information Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Telephone number | Text |  |  |  |
| 2 | Description | Text box |  |  |  |
| 3 | View history | Button |  |  |  |
| 4 | Update | Button |  |  |  |
| 5 | Delete | Button |  |  |  |

###### Use case specification

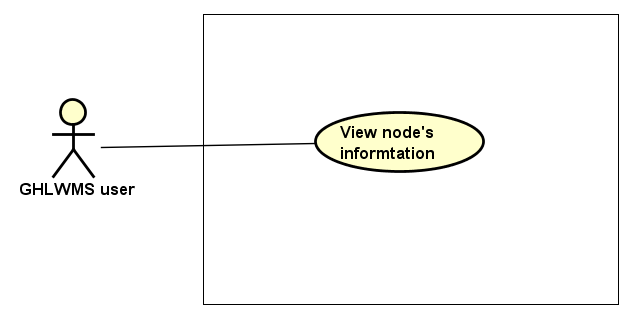


Figure 36: View node's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC13** | **Use Case Name** | | **View node’s information** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows a user to be able to view information of a node including telephone number and description. | | | | |
| **Precondition** | | Exist at least one node in GHLWMS | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a node and click the node. | | | | |
| *4* | GHLWMS | Display “Node’s information” window | | | | |

##### UC14 – View node’s history

###### Screen Design

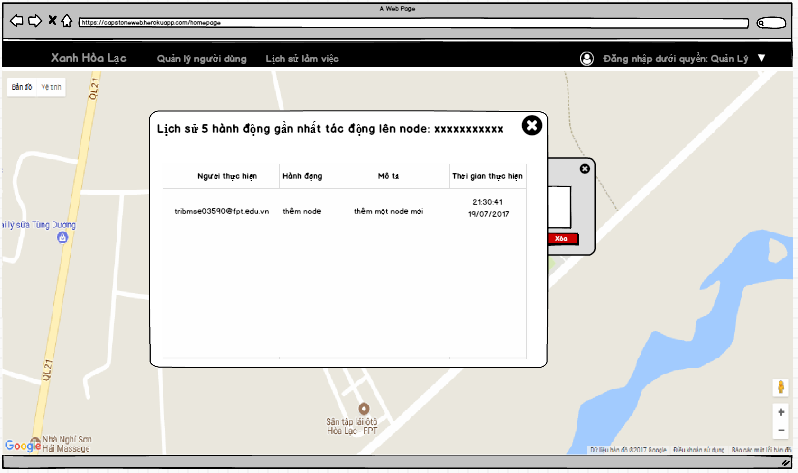


Figure 37: View node's history screen

Table 21: View node's history screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Top 5 previous action in the node | Popup |  |  | It contains 4 columns: users, action, description, and time |

###### Use case specification

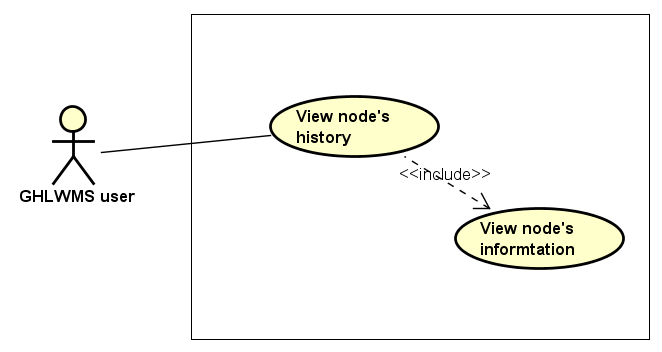


Figure 38: View node's history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC14** | **Use Case Name** | | **View node’s history** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows a user to be able to view list 5 previous the previous node’s activities in GHLWMS. | | | | |
| **Precondition** | | Exist at least one node in GHLWMS | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a node and click the node. | | | | |
| *4* | GHLWMS | Display “Node’s information” window | | | | |
| *5* | User | Click “View log” button in the window | | | | |
| *6* | GHLWMS | Display “List node’s activities” popup with a table which contains 4 columns: user, action, description, time | | | | |

##### UC15 – Update node’s information

###### Screen Design

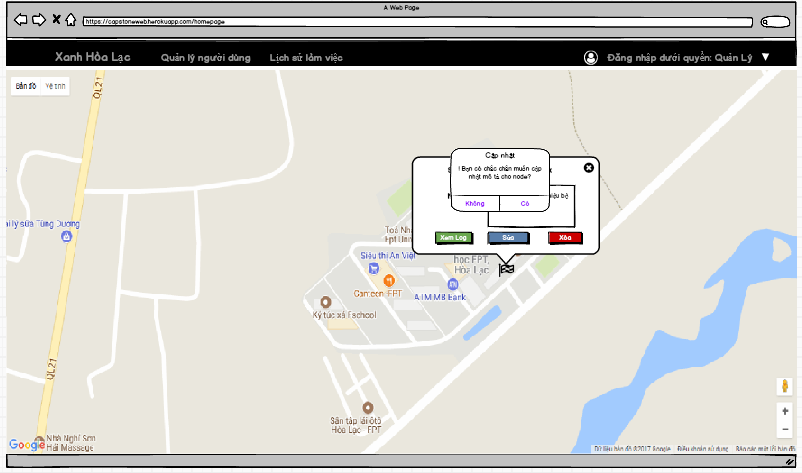


Figure 39: Update node's information screen

Table 22: Update node's information Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Update node’s information | Dialog |  |  | Confirm dialog |

###### Use case specification

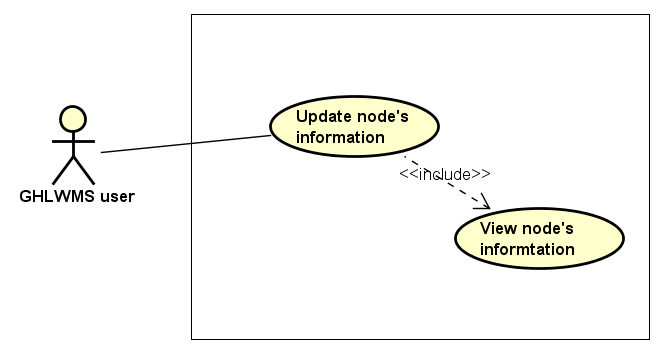


Figure 40: Update node's information Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC15** | **Use Case Name** | | **Update node’s information** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to update node’s information | | | | |
| **Precondition** | | Exist at least one node in GHLWMS | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a node and click the node. | | | | |
| *4* | GHLWMS | Display “Node’s information” window | | | | |
| *5* | User | * Enter new description in “Description” text box in the window * Click “Update” button in the window | | | | |
| *6* | GHLWMS | Display “Confirm to update node’s information” popup | | | | |
| *7* | User | Click “Agree” or “Disagree” button the confirm popup | | | | |
| *8* | GHLWMS | * If click “Agree” button, the new node’s information is updated in the database * Else, return step 4 | | | | |

**Alternative flows**

|  |  |  |
| --- | --- | --- |
| **AT1** | At step 5 in the main flows**,** if click “Update” button without enter new description | |
| **Sub step** | **Actor** | **Action** |
| 5.1 | GHLWMS | Return step 4 |

##### UC16 – Delete node

###### Screen Design



Figure 41: Delete node screen

Table 23: Delete node Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Delete node | Dialog |  |  | Confirm dialog |

###### Use case specification

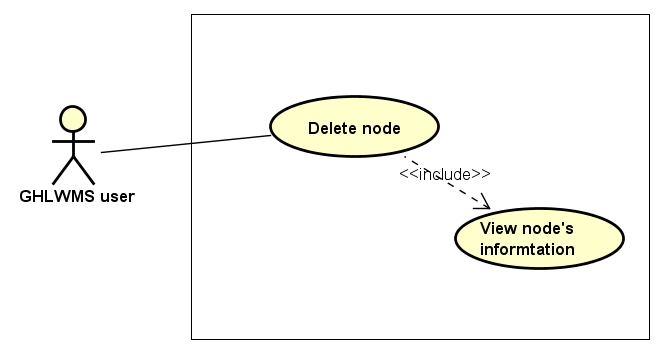


Figure 42: Delete node Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC16** | **Use Case Name** | | **Delete node** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to delete node | | | | |
| **Precondition** | | Exist at least one node in GHLWMS | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Green Ho La” in the navigation bar | | | | |
| *2* | GHLWMS | Display “Green Ho La” page by a map with nodes (also is home page by default when login) | | | | |
| *3* | User | Select a node and click the node. | | | | |
| *4* | GHLWMS | Display “Node’s information” window | | | | |
| *5* | User | * Enter new description in “Description” text box in the window * Click “Delete” button in the window | | | | |
| *6* | GHLWMS | Display “Confirm to delete the node” popup | | | | |
| *7* | User | Click “Agree” or “Disagree” button the confirm popup | | | | |
| *8* | GHLWMS | * If click “Agree” button, update status of the node (IsDeleted = true) in the Database * Hide the node in the map * Else, return step 4 | | | | |

**Business Rules**

|  |  |
| --- | --- |
| ***#*** | ***Rule Description*** |
| BR12 | In case, users delete a node and then add a new node with the same telephone number, GHLWMS consider they are the same node and save an activity history for the node as “Restore a node which was deleted before” |

##### UC17 – Sort activity history

###### Screen Design

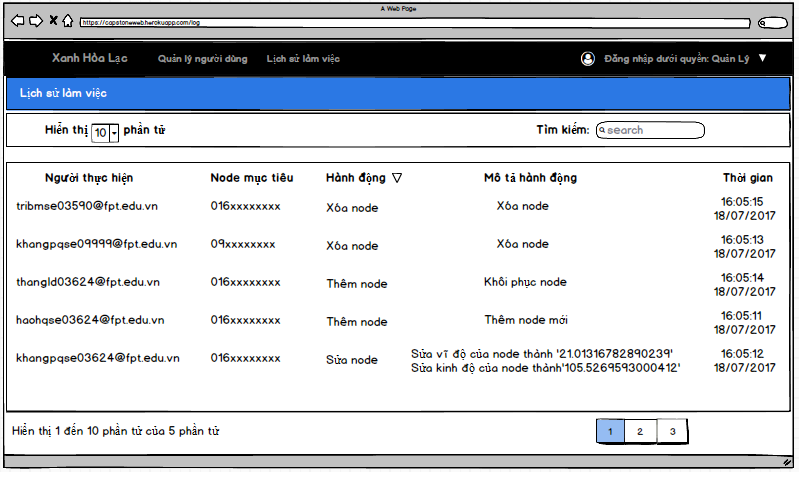


Figure 43: Sort activity history screen

Table 24: Sort activity history Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | List activity history | Table |  |  | The table contains 5 columns: user, target node, activity, description, time |

###### Use case specification

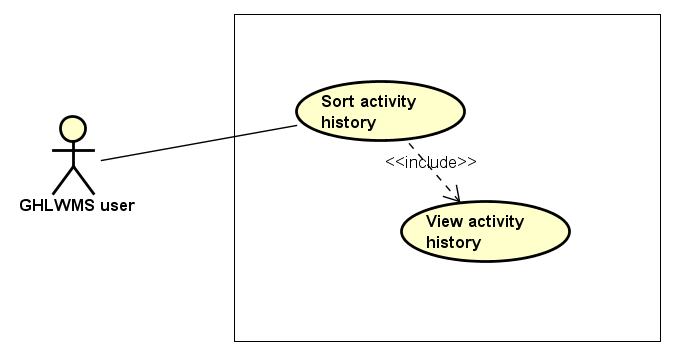


Figure 44: Sort activity history Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC17** | **Use Case Name** | | **Sort activity history** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to sort activity history by user, target , node, activity, description or time | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click “Activity history” on the navigation bar | | | | |
| *2* | GHLWMS | Display “Activity history” page with a table which has default top 10 previous actions. | | | | |
| *3* | User | Click on one of column’s title in the table | | | | |
| *4* | GHLWMS | Sort list activity history by the title selected by users. | | | | |

##### UC18 – Logout

###### Screen Design

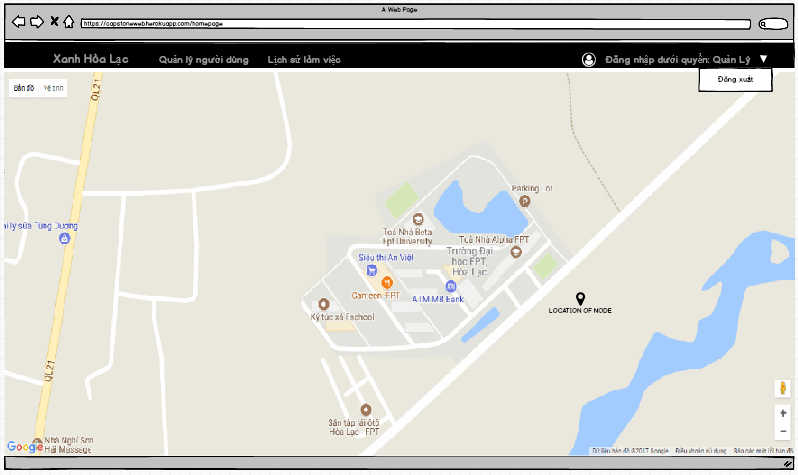


Figure 45: Logout screen

Table 25: Logout Screen definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Field Name** | **Type** | **Mandatory** | **Max Length** | **Description** |
| 1 | Logout | An Item of drop down list |  |  |  |

###### Use case specification

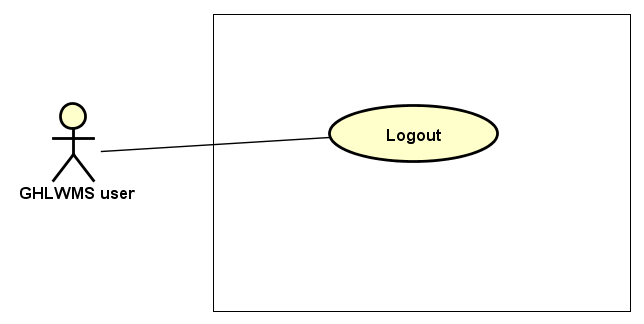


Figure 46: Logout Use case diagram

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | | **UC18** | **Use Case Name** | | **Log out** | |
| **Author** | | **KhangPQ** | **Version** | **1.0** | **Date** | **16/06/2017** |
| **Actor** | | GHLWMS User | | | | |
| **Description** | | The function allows an user to be able to log out off GHLWMS | | | | |
| **Precondition** | |  | | | | |
| **Trigger** | |  | | | | |
| **Post-Condition** | |  | | | | |
| **Main flows** | | | | | | |
| ***Step*** | ***Actor*** | ***Action*** | | | | |
| *1* | User | Click a drop down list on the right top of the navigation bar | | | | |
| *2* | GHLWMS | Display list item in the drop down list | | | | |
| *3* | User | Select “Logout” | | | | |
| *4* | GHLWMS | Logout user and navigate to “Login” page | | | | |

#### 2.2. Non-Functional Requirement Specification

Table 26: Nonfunctional requirements for GHLWMS

|  |  |  |
| --- | --- | --- |
| # | Nonfunctional requirements | Description |
| 1 | Security | Users have to sign in with Google’s accounts |
| 2 | Availability | The user session must time out after 24 hours |
| 3 | Usability | Language: Vietnamese |

## IV. System design

### Architecture design

#### System architecture design

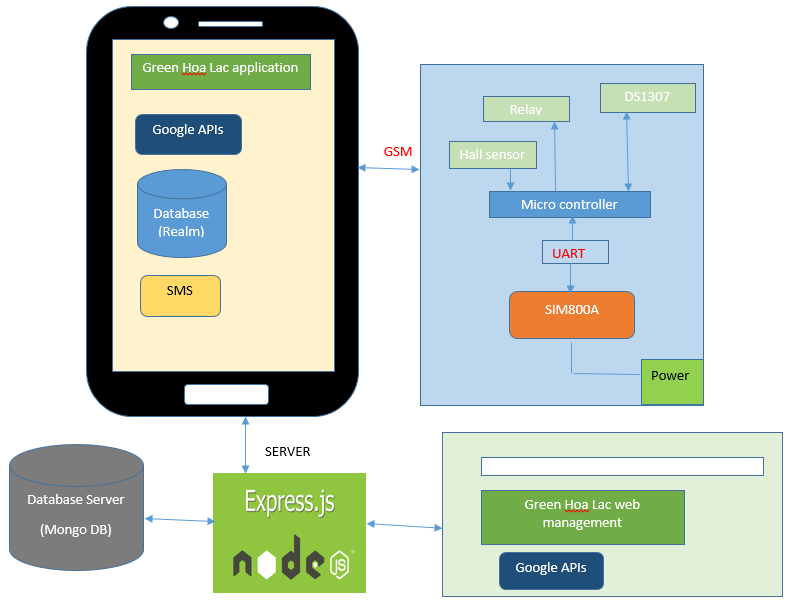


Figure 77: Architecture design

Green Hoa Lac system has 5 parts:

* Green Hoa Lac mobile application:

Android platform is chosen to develop Green Hoa Lac mobile application. Users can control this application through Android containers.

The application uses local database (Realm) to store map information in case of losing Internet connection.

The application sends SMS messages to control parts of Micro controller, and receive notification and messages by SMS messages.

We use GSM for the reason that is the application and Module Node interact each other through SMS.

* Module Node

Arduino Mega is chosen as the micro controller process unit that connected five modules: Module SIM800A MH, Sensor Hall, Relay, Module DS1307+AT24C32

Module SIM800A MH has features of sending and receiving SMS from mobile device, then move the SMS string data to micro controller. In this time, Arduino is receiving string data and optimizing this string, sending signal to each part of GHLS.

Controller and Module SIM800A MH interact each other using UART protocol

* Server database

We use mongo DB as a server database of GHLS.

All information about user, node, session and history are store in this database

* Green Hoa Lac management web

We design a web to manage whole GHLS. Any actions related users are performed and shown on this web. In addition, to extend or shrink real scope of GHLS, we also do through the web.

* Server

NodeJS Express Application is chosen to develop APIs such as: Weather, Node, Log, User and Authentication.

#### Serial port (UART) communication protocol

The SIM800 GPRS/GSM+GPS Shield is used UART protocol to communicate with an Arduino/Arduino clone.

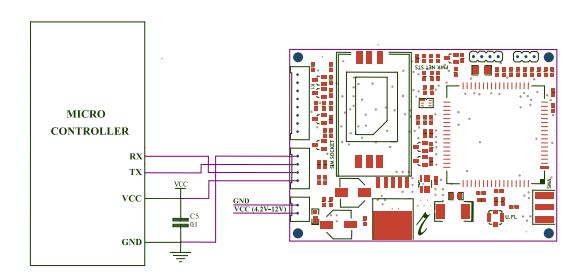


Figure 78: Serial port (UART) communication protocol

#### GSM

GSM (Global System for Mobile Communications) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones, first deployed in Finland in December 1991. As of 2014, it has become the de facto global standard for mobile communications – with over 90% market share, operating in over 219 countries and territories.

GSM networks operate in a number of different carrier frequency ranges (separated into GSM frequency ranges for 2G and UMTS frequency bands for 3G), with most 2G GSM networks operating in the 900 MHz or 1800 MHz bands. Where these bands were already allocated, the 850 MHz and 1900 MHz bands were used instead.

GSM was intended to be a secure wireless system. It has considered the user authentication using a pre-shared key and challenge-response, and over-the-air encryption.

Short Message Service (SMS) Message Format:

Short Message Service or SMS messages, as defined in RFC 5724, are short two way alphanumeric paging messages that can be send to and from SMS clients. SMS clients, including text messaging service component of phone, Web, or other mobile communication systems, are an integral part of the GSM (Global System for Mobile Communications) network technology. SMS uses standardized communications protocols to allow fixed line or mobile phone devices to exchange short text messages. SMS messages can be used to transport almost any kind of data (within the character limit).

### Mechanical design

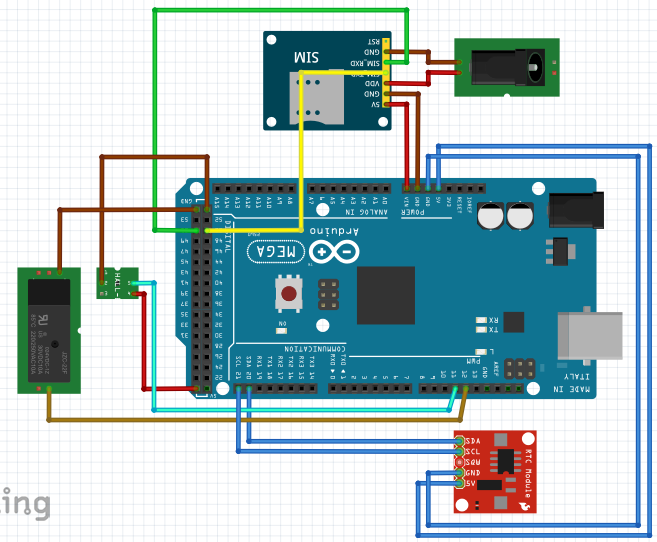


Figure 79: Breadboard

### Hardware design

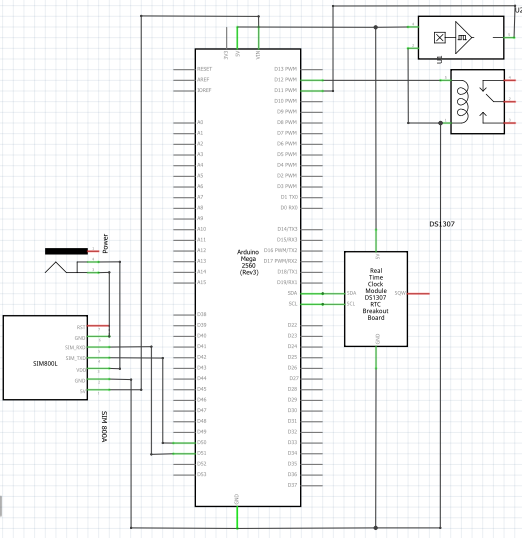


Figure 80: Schematic

### Firmware design

#### 4.1. Turn on node



Figure 81: Turn on node flow chart

#### 4.2. Turn off node

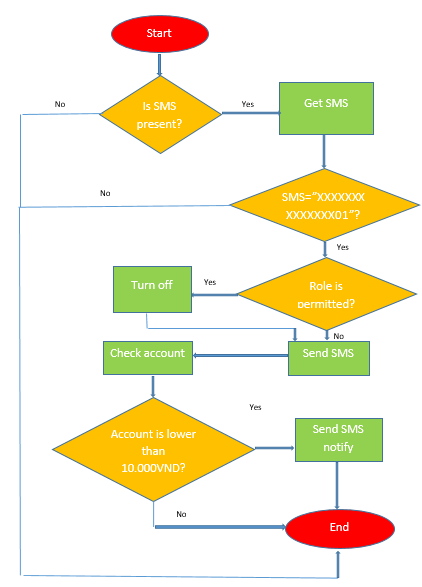


Figure 82: Turn off node flow chart

#### 4.3. Set schedule

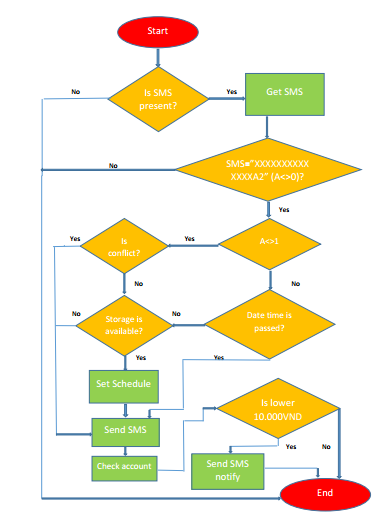


Figure 83: Set Schedule flow chart

#### 4.4. Cancel schedule

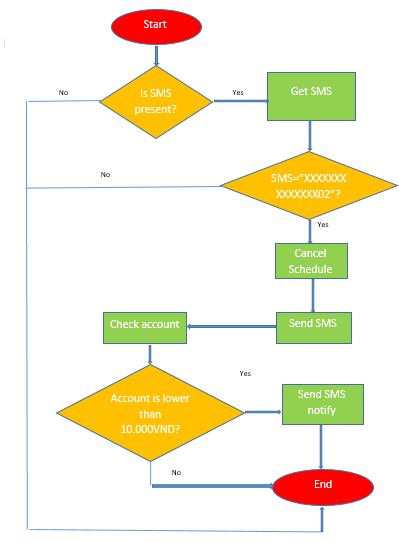


Figure 84: Cancel Schedule

#### 4.5. Get schedule

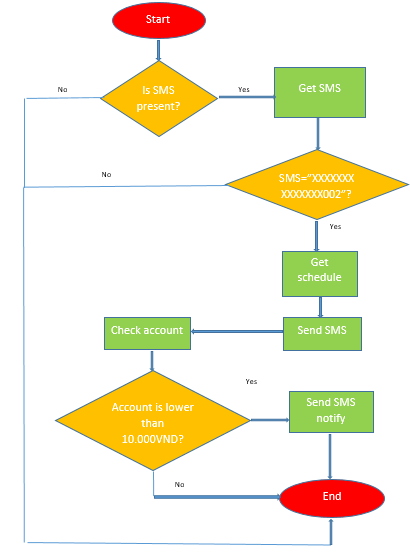


Figure 85: Get schedule

### Software design

For the application, we have developed many function. However, there are six main functions related to authentication, node control, synchronize.

#### 5.1. Login

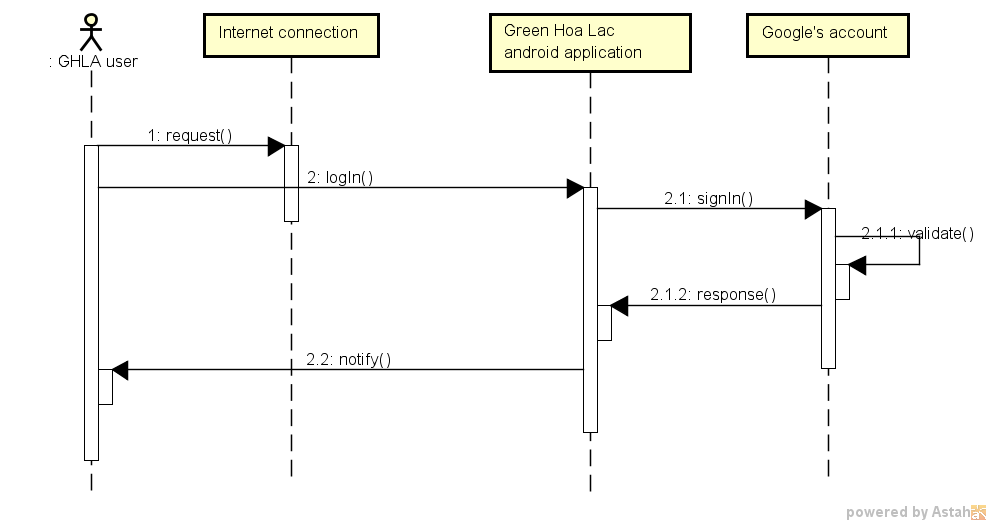


Figure 86: Login sequence diagram

#### 5.2. Turn on node

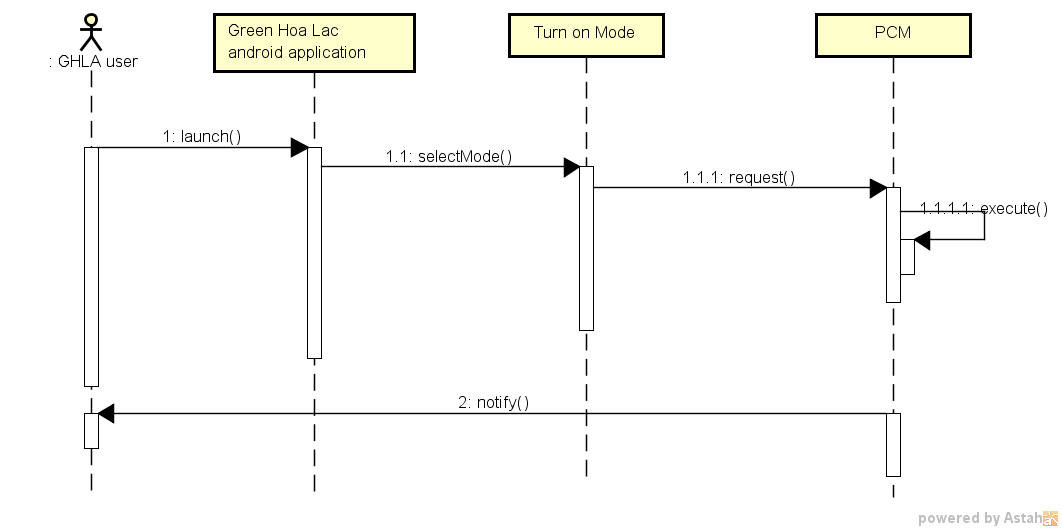


Figure 87: Turn on node sequence diagram

#### 5.3. Turn off node

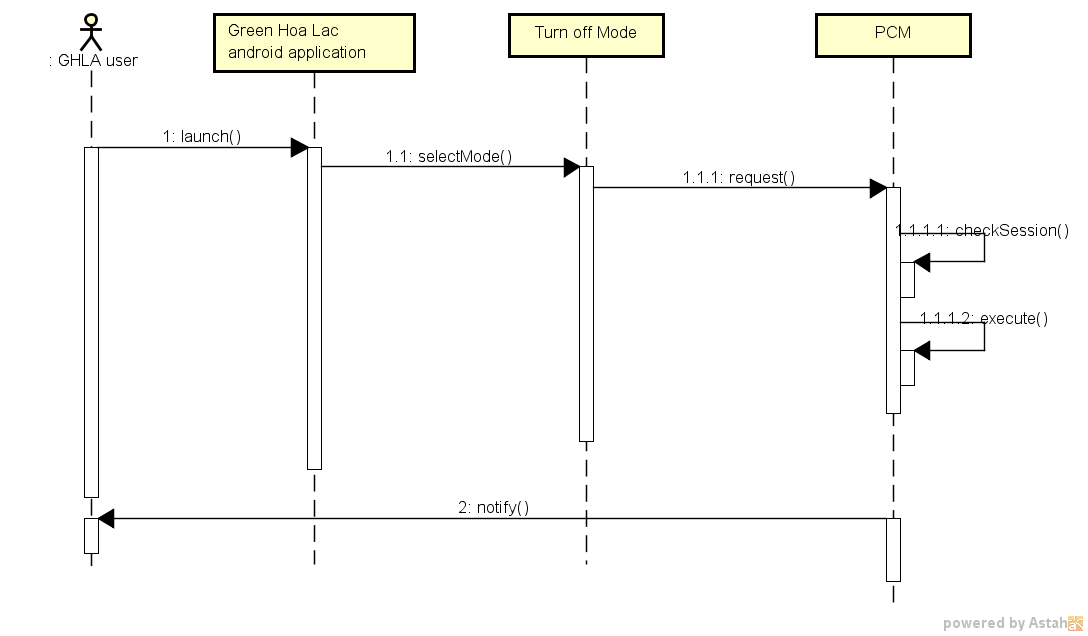


Figure 88: Turn off node sequence diagram

#### 5.4. Set schedule

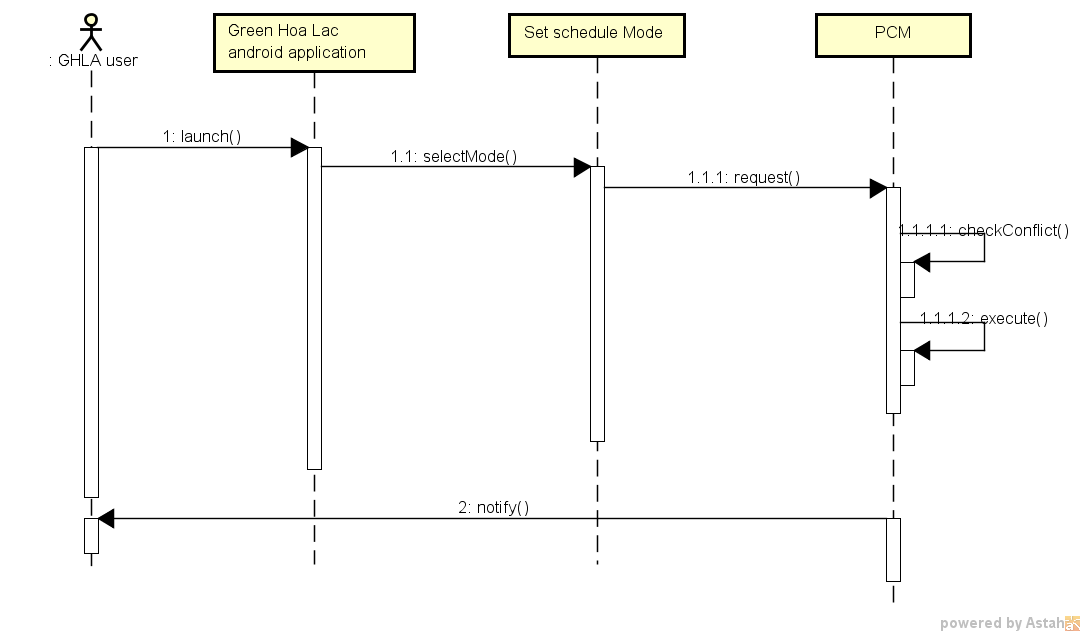


Figure 89: Set schedule sequence diagram

#### 5.5. Cancel Schedule

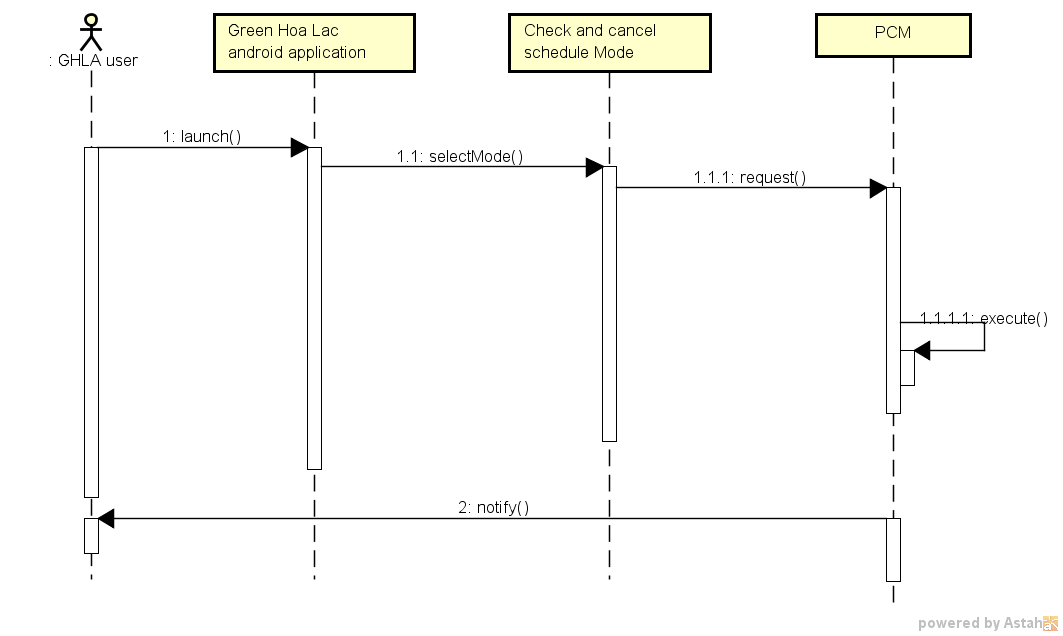


Figure 90: Cancel Schedule

#### 5.6. Synchronize node

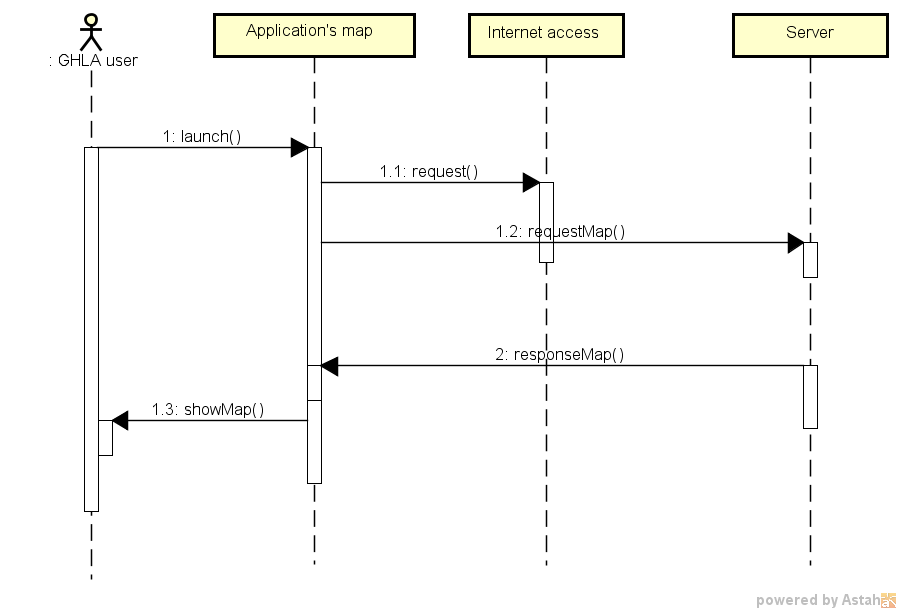


Figure 91: Synchronize node sequence diagram

### Database design

#### 6.1. Server database

We choose MongoDB to develop system’s database, which is NoSQL databse.

NoSQL is a database generation with main characteristics that are nonrelational, distributed, open source, horizontal scalable. Storage, processing from very small to very large data, up to petabytes of data in the system need to bear high load and fault tolerance with the requirements of low hardware resources.

Mongo's main goal is to retain the familiarity of SQL. Therefore, the query is quite similar to SQL, MongoDB is quite suitable for programmers who are familiar with SQL query language. MongoDB has a large volume of features and high performance. With rich data types, multiple queries and reduced development time in modeling objects.

MongoDB is best suited to dynamic query needs, requiring fast speed for a large database. MongoDB is fast, fast, and fast. MongoDB supports field searching, search results, and syntax searches. Queries can return the specified fields in the text, and may also include user-defined javascript functions. Just like relational databases, any field in MongoDB is indexed.

MongoDB also supports master-slave, fragment, and sharding.

MongoDB uses a process to process data requests, manages the data format, performs the management operations below, and is the main processor. MongoDB provides a service for processing queries from the application layer, locating data in a cluster of fragmented nodes called mongos.

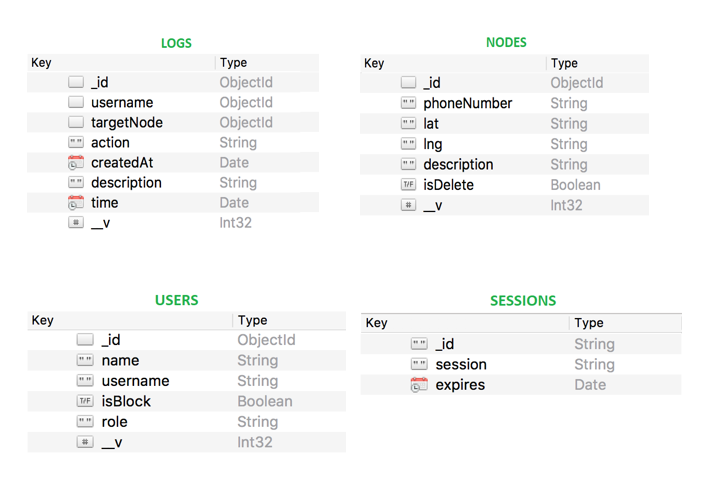


Figure 92: Server database (MongoDB)

In above design, we have:

logs.username = users.\_id  
logs.targetNode = nodes.\_id

#### 6.2. Android application database

Realm mobile database is an open source, embedded database library optimized for mobile use. Realm uses a “data container” model. Our data objects are stored in a Realm *as objects. We decided to choose Realm for some reasons:*

* Realms store **native objects:** The Realm Mobile Database has bindings for many popular languages for mobile app development, including Swift, Java, Objective-C, C#, and JavaScript (using React Native). The objects you store in a Realm are the objects you work with in the rest of your code.
* Realms are **zero-copy:** data is not copied in and out of the database to be accessed; you’re working with the objects directly.
* Realms implement the **live objects pattern:** if you have an instance of an object stored in a Realm and something else in your application updates that object, your instance will reflect those changes.
* Realms are **cross-platform:** as long as you don’t store platform-specific objects in a Realm, the data can be synced across operating systems. (In fact, the actual Realm data files can be copied between platforms.)
* Realms are **offline-first:** Your application works the same way with a Realm whether or not it’s synchronized via the Realm Object Server, and whether or not the device is online at any given moment. When connectivity is available, changes will be synchronized transparently.
* Lastly, Realms are **ACID-compliant.**

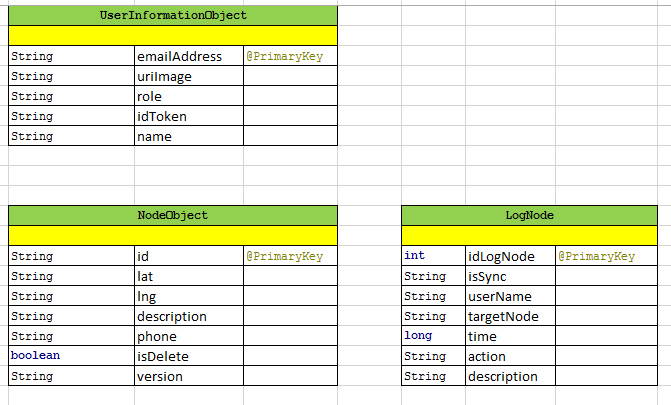


Figure 93: Realm database design

## V. Implement and testing

### 1. Testing life cycle

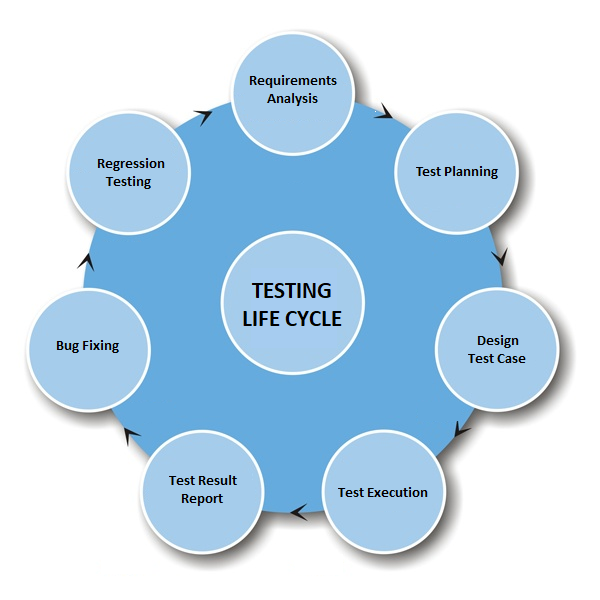


Figure 94: Testing life cycle

### 2. Testing model and tools

#### 2.1. Testing model

We choose V-Model to perform testing the system. V-Model is one of the many software development models. Testing of the product is planned in parallel with a corresponding phase of development in V-model. It provide a simple process to perform project and test.

There are several reasons why using V-model:

* Simple and easy to use.
* Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.

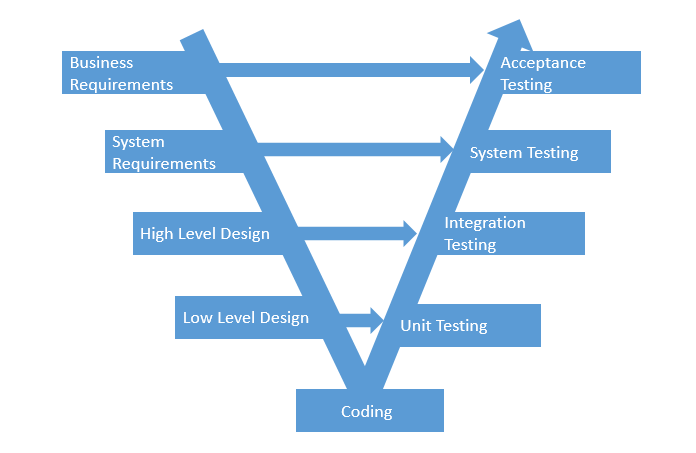


Figure 95: V testing model

#### ***Testing tools***

* Mobile device: SamSung Galaxy A5, Note4.
* Emulators: Genymotion, default Android emulators.
* Hardware: Arduino Mega, HallA04E, SIM800, DS1307, Relay.
* Record tools: Microsoft Excel, Word, Atlassian (\*).

Atlassian is a website used to log bugs.

### 3. Test case and log bugs

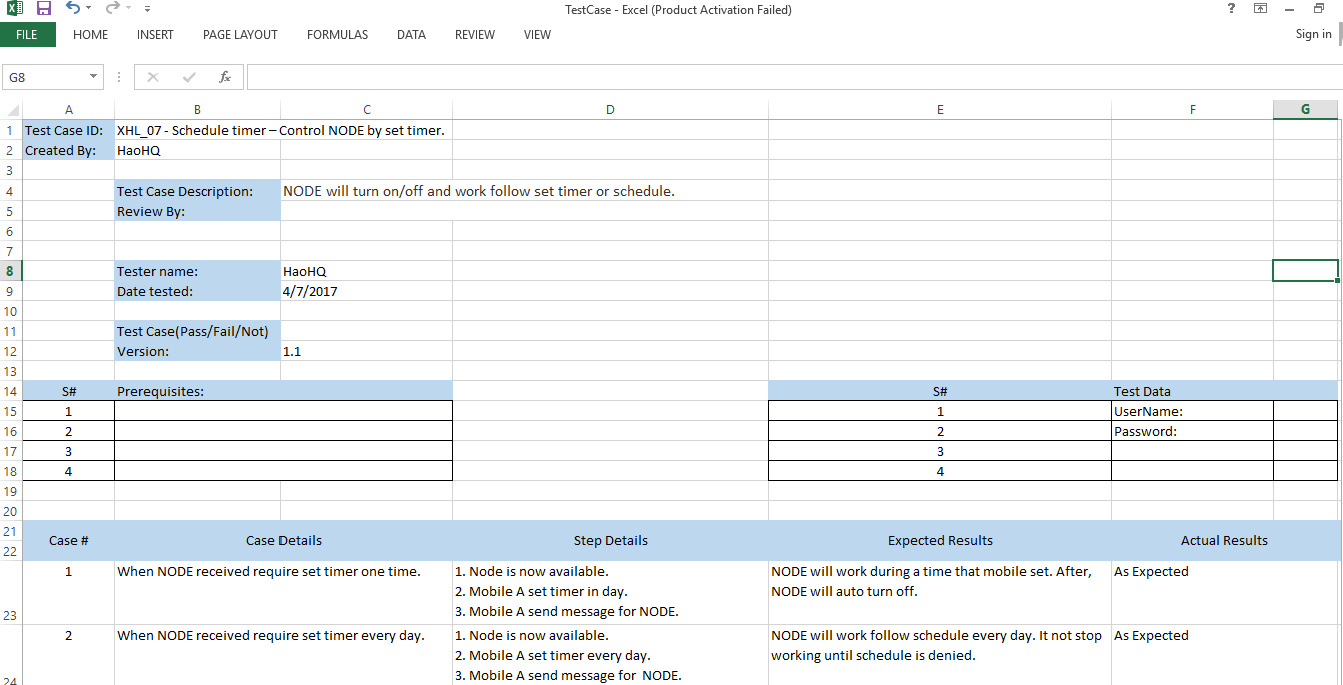


Figure 96: Test case

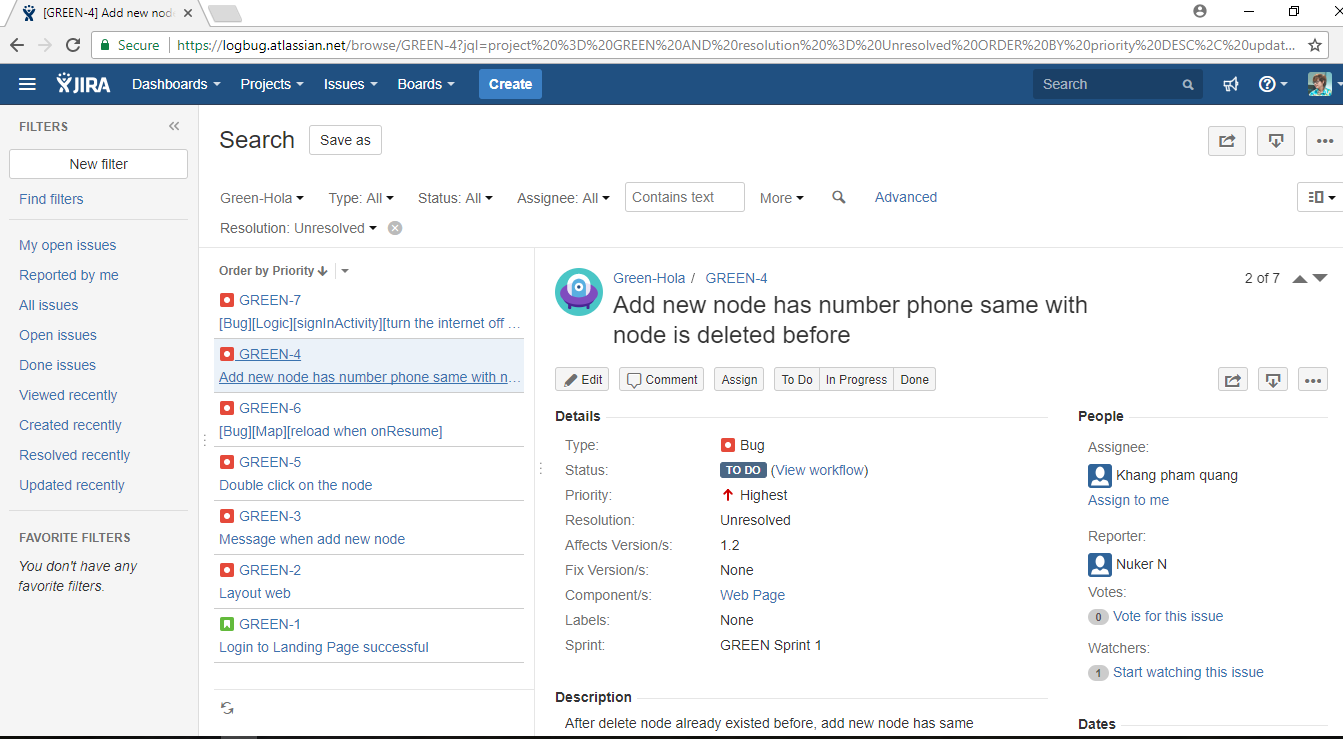


Figure 97: Log bug

### 4. Implement

#### 4.1. Version 1.0

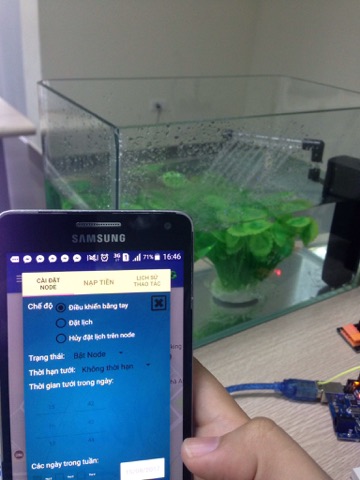
We have the first version of GHLS with features like:

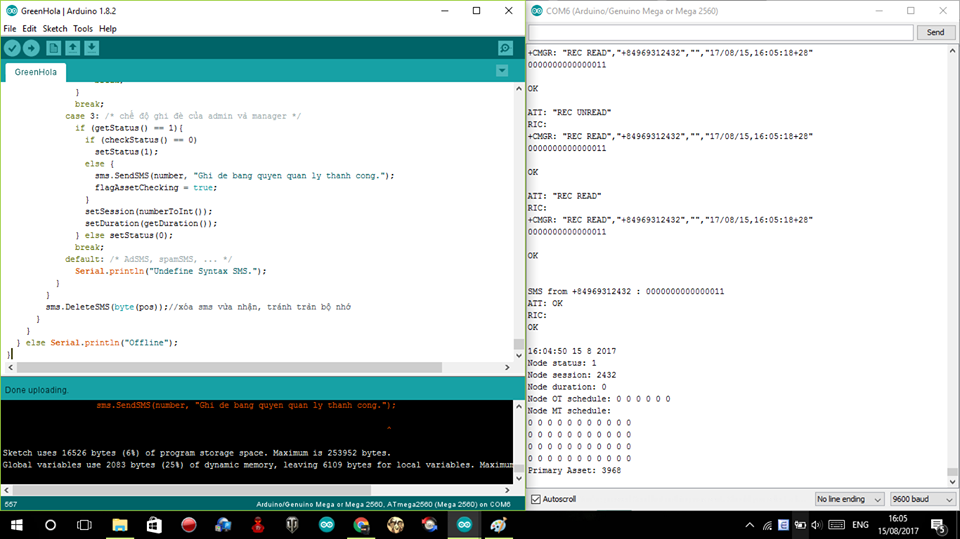
* Turn On/Of Node
* Set schedules
* Cancel schedules
* Notify account

With this prototype, we expected it can cover all main functions of GHLS, and be used Q&A technical in LAB room environment.

Besides the hardware module, we gave a software application in mobile device, which is helpful to control GHLS.

* Send On/Off request mode
* Send set schedule request mode
* Get map, update nodes
* Cancel schedule





The version is kind of stable and most of responses from module SIM800 are corrected as expected.

#### 4.2. Version 1.1

This version is based on version 1.0, which run on real pump with new update feature like:

* Turn On/Of Node
* Set schedules
* Cancel schedules
* Notify account

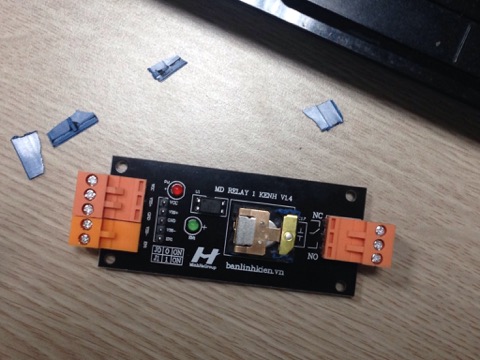
In addition, the software had been updated more functions for User as below:

* Send On/Off request mode
* Send set schedule request mode
* Get map, update nodes
* Schedule information
* Cancel schedule
* Weather, user information

The version is stable, easy to set up. Especially, we have customized graphic user interface to be useful for users.

### 5. Evidences

#### 5.1. Hardware device broken



It is so hard to protect the Hardware. This device does not have a protecting layer so it is easier to break the signal line and Module will get error.

#### 5.2. Turn off node unsuccessfully as Admin role

|  |  |
| --- | --- |
|  |  |

In this case, we controlled to turn off pump as Admin role. However, we could not turn off and receive a message “Ghi de bang quyen quan ly thanh cong”.

After debug, we recognized this error related with android application, the application send wrong syntax to module SIM800A.

We fixed this error change syntax of Admin role. After that, we can turn off the pump successful.

#### 5.3. Test set schedule unsuccessfully

|  |  |
| --- | --- |
|  |  |

In this case, we scheduled for a present day (15/08/2017) with future time (17:14) (the present time is 16:12). However, we could not set this schedule and receive a message “Khong the hen gio vao thoi gian trong qua khu”.

After debug, we recognized this error related with android application, the application send wrong syntax time to module SIM800A with time is 15/07/2017 because date picker got value 1 month early, which is the reason why we could not set schedule for node.

We fixed this error by the way plus one month to value of date picker. After that, we can set schedule successful.

## VI. Conclusion and perspective

During a period of four months, all team members have practiced and learned a huge amount of knowledge in many aspects such as: embedded hardware, software, development process. In addition, we also have improved other skills:

* Communication and teamwork
* Time management
* Studying new technologies
* Researching and applying software development process
* Leadership and Q&A skill

Green – Hoa Lac system has divided into sub modules successfully:

* A hardware module containing: Module SIM800A MH, Hall sensor A04E, Module DS1307+AT24C32 and Module Relay 1 channel 5V- 220V/10A
* An android application to interact with hardware module
* A website to manage the whole system
* A database to manage information of the whole system

In addition, we will refine the GHLS better for business purposes in the future. We want to develop this system that satisfies not only cost of product but also quality to customers. We believe that this system can be successful soon by our efforts.

## VII. Reference

[GSM] https://en.wikipedia.org/wiki/GSM