C. Report No.3 Software Requirement Specification

1. User Requirement Specification

Nowadays, there are many reports about the unhealthy side-effects found in the foods we buy and eat. Large food which is introduced by chemical to grow food quicker and use pesticides to prevent loss from being destroyed by worms. In this country, it’s cheaper for a family to eat fast food or meats than it is to eat organically grown fruits and vegetables. In addition to the foods, we are developing industry and modern too fast with a lot of harmful effects for environment, air quality as poor as less space to relax after work at home. A small garden in house which is not only makes your house look good but they also keeps you healthy with fresh air and safe fruit or vegetables; this is also the best place to reduce stress and lighten mind which cannot be done being inside four walls. But we have to face too much pressure and differences works in day, this make us have less time to take care a garden day by day. Come from real needs, users want a system which help them to take care a small garden in house less time, smarter and flexibility. The system should meet the below needs:

1.1. Monitoring the status of garden

Users can tracking indexes of garden through sensors.

Users can monitor the status of garden from anywhere in anytime.

**1.2. Response with expected problems**

User can be notified when unexpected problems occurred of weather with their garden (rain, high temperature ...).

System can automatically react with problems which is harmful for garden.

**1.3. Manage the system**

Users can control one or multi devices to solve some specific tasks such as limit volume water for some kind of plant in blooming lately.

Users can manage all devices, tracking reliability and durability of devices; add, remove or configure a specific devices.

Users can start or stop any automatically actions in their garden

**1.4. Schedule**

Users can create a plan which using time of each device in the system such duration time of sensor measurements by manually or automatically.

Users will be received auto schedule after processing indexes on server.

Users need to be notified some reminder with situations which meet plant’s requirement or auto actions deadline nearly

**1.5. Consumption and durability**

All sensors which used in the system, have to a long battery life.

All sensors have a great durability when usually contact with water day by day.

**1.6. High security**

Users have to be authenticated before using the system

Data and commands which transmit in system via RF and Wi-Fi should be protected.

**1.7. Interoperability**

New end devices can be added to the system and interact with other ones regardless of manufacturing origin.

2. System Requirement Specification

2.1 External Interface Requirement

2.1.1. User interface

The interface must be designed to be satisfied the following requirement:

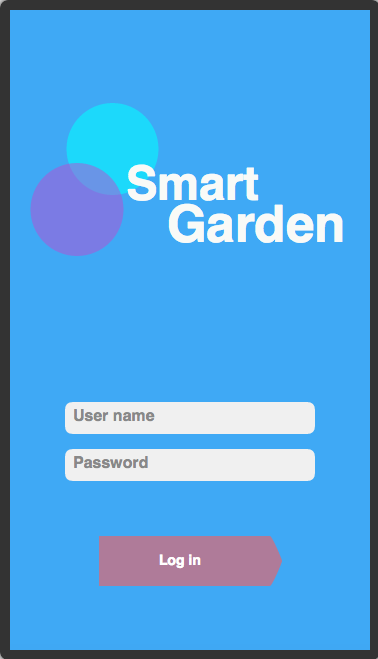
The interface is divided by tabs, which will allow users to easily switch between different parts of the program.

Be simple and user-friendly.

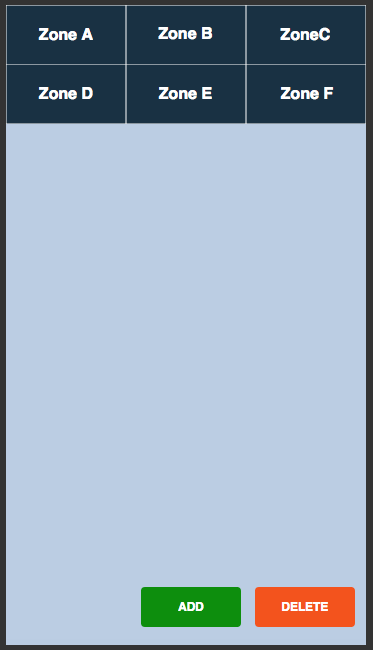
Meet all the main functions and easily to identify each of functions.

Use obvious icons to avoiding misunderstanding.

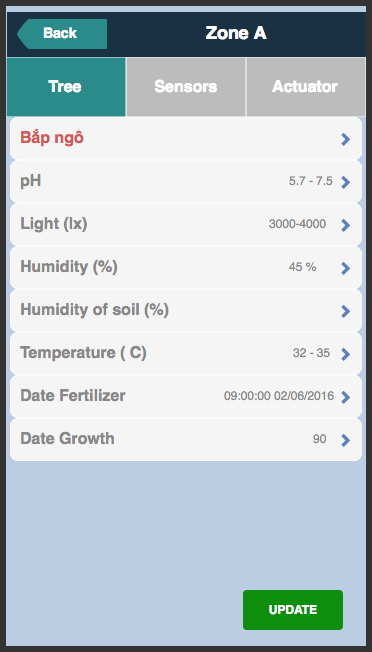
User interface use Graphical User Interface must be simple, clear, and easy to use. Each screen will have the instruction to guide user how to use it.

**2.1.1.1. Login**

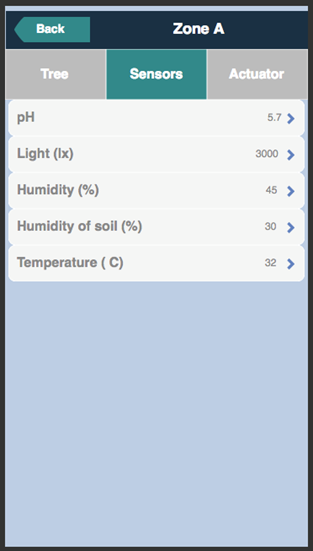
**Figure 3. “Login” screen**

**2.1.1.2. Zone management**

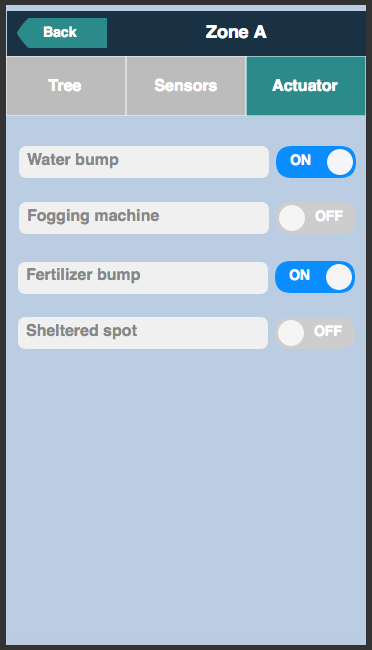
**Figure 4. “Zone management” screen**

**2.1.1.3. Tree information**

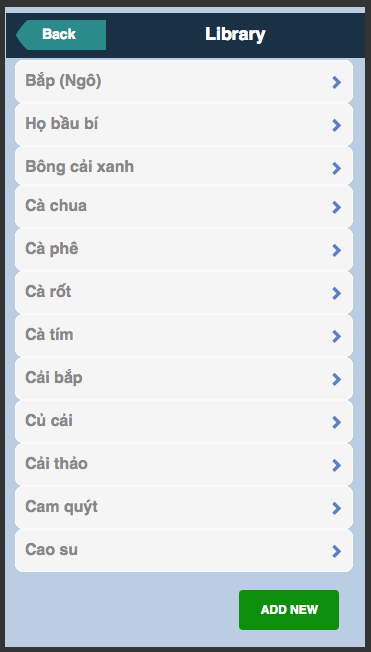
**Figure 5. “Tree information” screen**

**2.1.1.4 Sensors management**

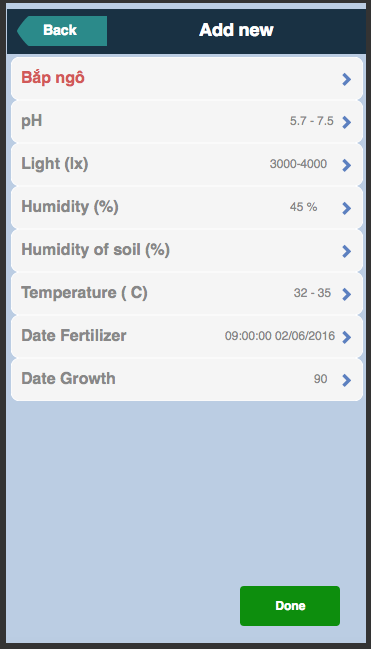
**Figure 6. “Sensors information” screen**

**2.1.1.5. Actuator management**

**Figure 7. “Actuator management” screen**

**2.1.1.6. Library**

**Figure 8. “Library” screen**

**2.1.1.7. Add new**

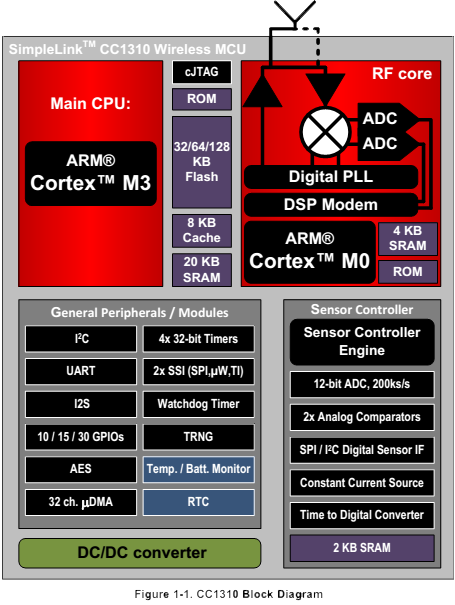
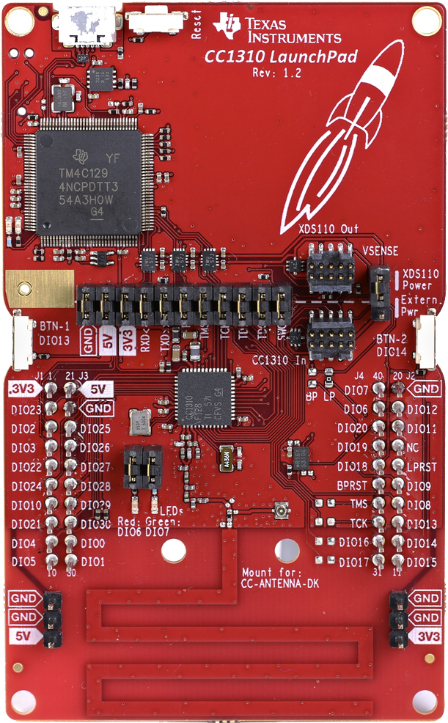
**Figure 9. “Add new” screen**

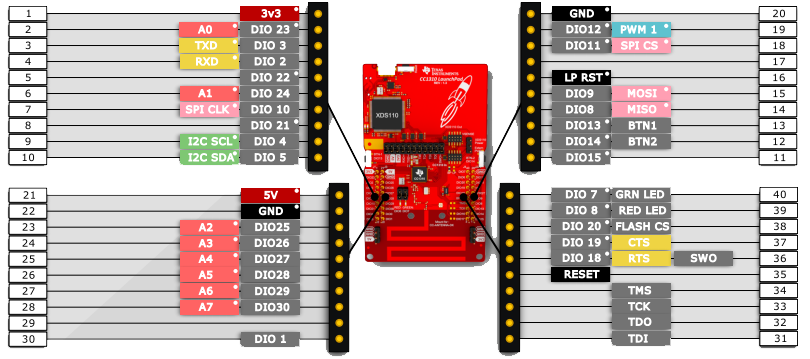
2.1.2. Hardware interface

The system use variety sensors which must transmit data via RF or Bluetooth to Controller to process and update via Wi-Fi to Internet where users can tracking system with mobile. So the hardware interface that the system using must be designed to be satisfied the following requirement:

* Low cost hardware sensors module which have reliability data and long endurance.
* Strong MCU have multi communication with sensors like I2C, SPI, UART; multi connection in system via RF, Bluetooth and Wi-Fi.
* Easy replace a node or a parts that not affect other parts in system

2.1.2.1. Device MCU – CC1310 Launchpad (Texas Instruments)





* ***Microcontroller***

- ARM Cortex M3

- EEMBC CoreMark Score: 142\* - EEMBC ULPBench Score: 158\*

- Up to 48MHz Clock Speed

- 128KB of In-System Programmable Flash

- 8KB of SDRAM for Cache

- 20KB of Ultralow Leakage SRAM

- 2-Pin cJTAG and JTAG Debugging

- Supports Over-the-Air Upgrade (OTA)

* ***Ultralow Power Sensor Controller***

- Can Run Autonomous From the Rest of the System

- 16-bit Architecture - 2KB of Ultralow Leakage SRAM for Code and Data

* ***Peripherals***

- All Digital Peripheral Pins Can Be Routed to Any GPIO

- Four General-Purpose Timer Modules (Eight 16-Bit or Four 32-Bit Timers, PWM Each)

- 12-Bit ADC, 200 ksamples/s, 8-Channel Analog MUX

- Continuous Time Comparator

- Ultralow Power Clocked Comparator

- Programmable Current Source

- UART – 2× SSI (SPI, MICROWIRE, TI) – I2C – I2S

- Real-Time Clock (RTC)

- AES-128 Security Module

- True Random Number Generator (TRNG)

- Support for Eight Capacitive Sensing Buttons

- Integrated Temperature Sensor for ARM

* ***Low Power***

- Wide Supply Voltage Range: 1.8 to 3.8 V

- Active-Mode RX: 5.5 mA

- Active-Mode TX at +10 dBm: 12.9 mA

- Active-Mode MCU 48 MHz Running Coremark: 2.5 mA (51 µA/MHz)

- Active-Mode MCU: 48.5 CoreMark/mA

- Active-Mode Sensor Controller at 24 MHz: 0.4 mA + 8.2 µA/MHz

- Sensor Controller, One Wake Up Every Second Performing One 12-Bit ADC Sampling: 0.85 µA

- Standby: 0.6 µA (RTC Running and RAM and CPU Retention)

- Shutdown: 185 nA (Wakeup on External Events)

* ***RF Section***

- Excellent Receiver Sensitivity –124 dBm using Long-Range Mode, –110 dBm at 50 kbps

- Excellent Selectivity: 52 dB

- Excellent Blocking Performance: 90 dB

- Programmable Output Power up to +14 dBm

- Single-Ended or Differential RF Interface

- Suitable for Systems Targeting Compliance with Worldwide Radio Frequency Regulations

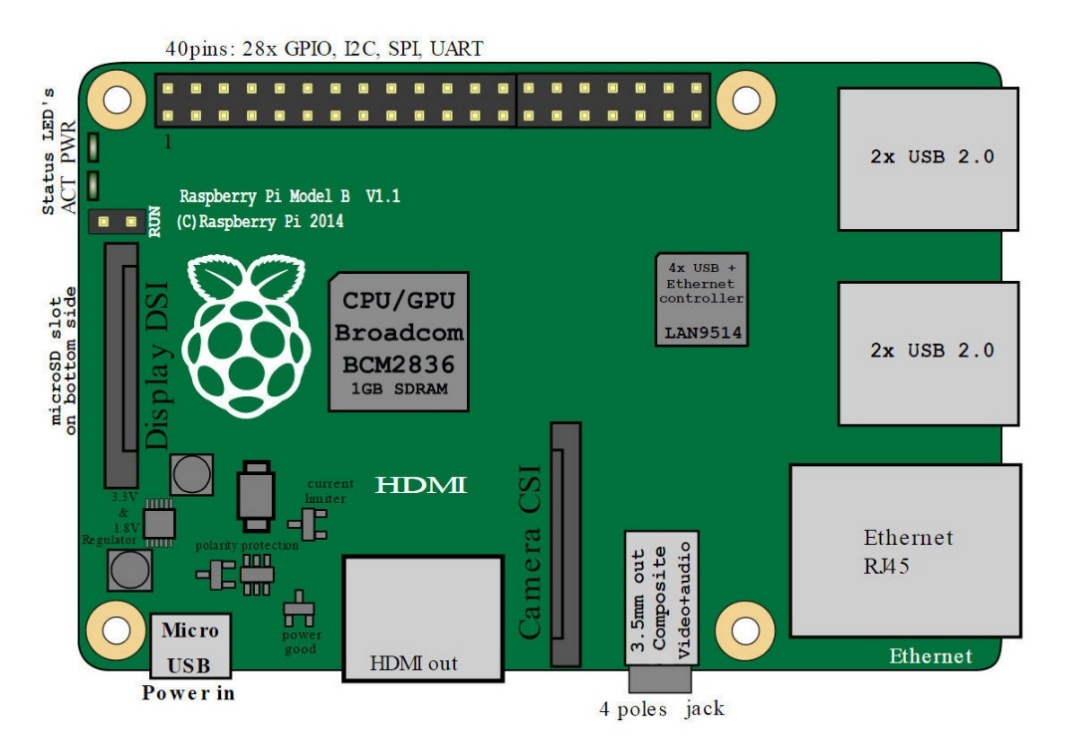
• ETSI EN 300 220, EN 303 131, EN 303 204 (Europe)

• FCC CFR47 Part 15 (US)

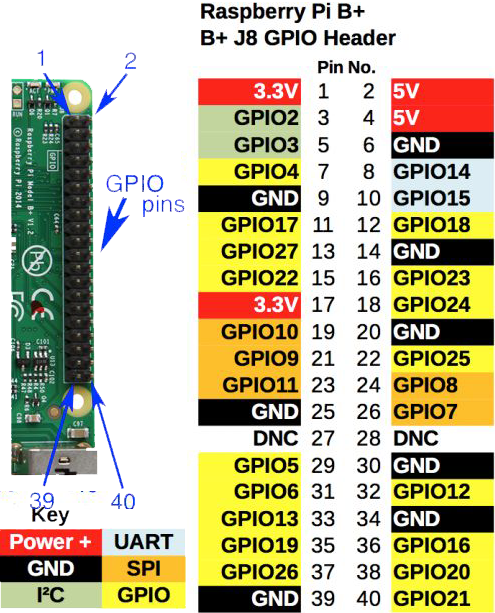
• ARIB STD-T108 (Japan)

- Wireless M-Bus and IEEE 802.15.4g PHY

2.1.2.2. Device MCU – Raspberry Pi 3 Mode B

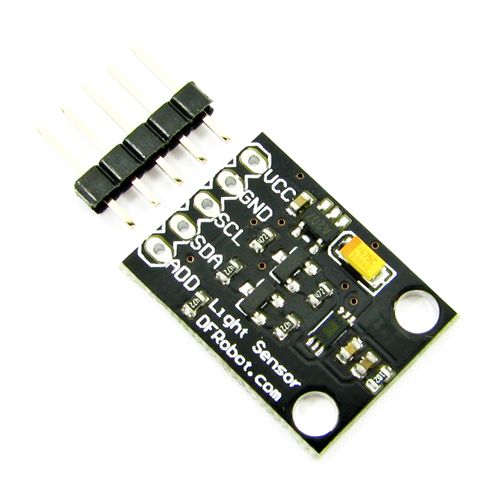


* ***A 1.2GHz 64-bit quad-core ARMv8 CPU***
* ***802.11n Wireless LAN***
* ***1GB RAM***
* ***4 USB ports***
* ***40 GPIO pins***
* ***Ethernet port***
* ***Micro SD card slot (now push-pull rather than push-push)***

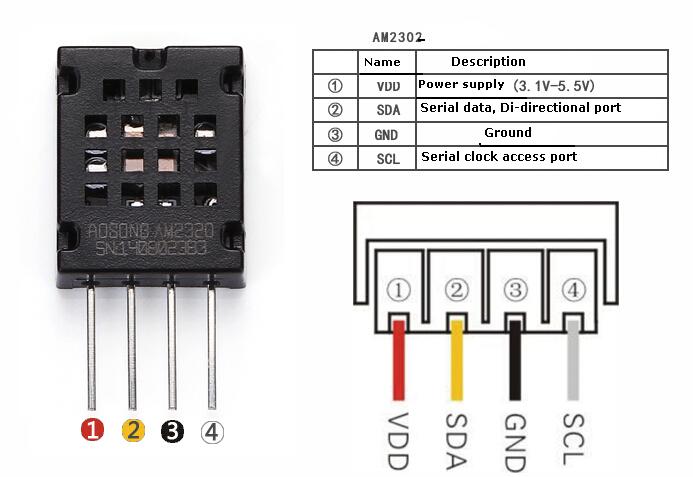


2.1.2.3. Sensors

*+ Light sensor:* which can get intensity of light around in a zone. Recommend: module BH1750 FVI

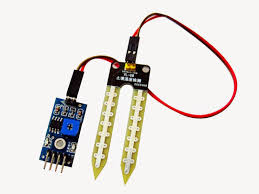
* Power input:
* Protocol: I2C

+ *Temperature and Humidity sensor:* get indexes of temperature and humidity in air. Recommend: AM2320



* Power input:
* Protocol: I2C

+ *Soil moisture sensor:* get index about humidity, water in soil

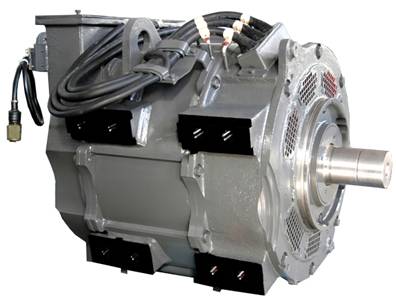


* Power input:
* Output: Analog, Digital

+ *PH sensor:* get index of pH degree in water

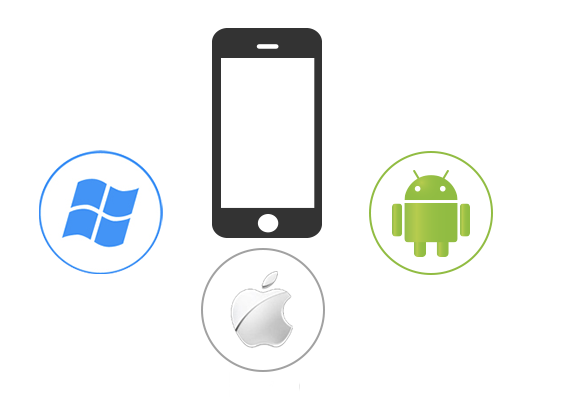
2.1.2.4. Actuators

The system must control actuators such as water pump, traction motor, and nebulizer which are used with 220V AC with relay.



2.1.2.5. Device users

The system is managed by user via mobile, so hardware interface for users is most popular in mobile market such as mobile using android, iOS or windows phone. We are recommend android mobile.



2.1.3. Software Interface

- Mobile Application: Android OS (v4.0 or above)

- Java Web server: Springs & Hibernate

- Database: MySQL

2.1.4. Communication Protocol

The system will applied to variety kinds of garden, small or large garden; it must satisfied distance communication in a garden. Users need a simple system which not affect too much garden’ space so electric wired between devices, so a wireless system is recommend with Wi-Fi and RF waves.

The device MCU in system have many kinds of protocol such as I2C, SPI, and UART to communicate with sensors. We recommend use I2C protocol to control easily and add more sensors to the systems.

2.2 System Overview Use Case

**Figure 16. System overview use case**

2.3. List of Use Case

2.3.1. <User> Login

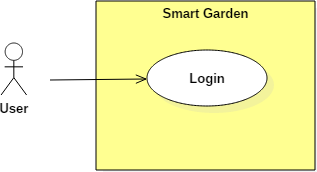


Figure 18. “Login” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC01** | | | |
| **Use Case No.** | **UUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Login | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - To manage and control the system, user must login into the system  **Goal:**  - Guest will become user to use function in the system after they login.  **Triggers:**  - User must input username and password which was provided and press button “Login”  **Preconditions:**  - User must have provided account from system.  **Post Conditions:**  - Success: User will moves to main page after login  - Fail: Display a popup message for errors  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access mobile application | Display Login Screen | | 2 | Fill 2 fields with username and password which was provided | Check validation of username and password | | 3 | Press “Login” button | Check login data in database  Log user into the system  Move to main page |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access mobile application | Display Login Screen | | 2 | Not fill enough fields or wrong format of textbox | Display "Username or password is wrong" message. | | No | Actor Action | System Response | | 1 | Access mobile application | Display Login Screen | | 2 | Fill invalid provided account’ information | Check validation of username and password | | 3 | Press “Login” button | Display "Invalid username or password" message. |   **Relationships:**  - NA | | | |

Table 9. ”Login” specification.

2.3.2. <User> Library Overview Use Case

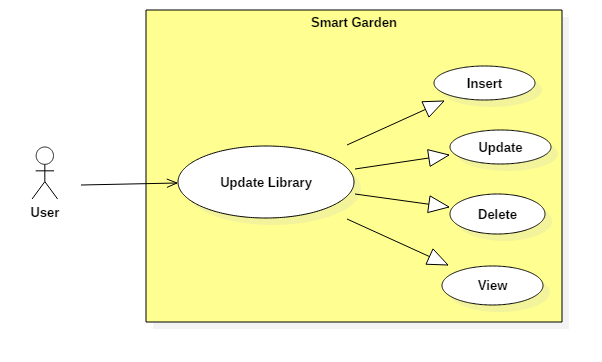


Figure 20. <User> Library Overview use case

2.3.2.1 <User> Library View

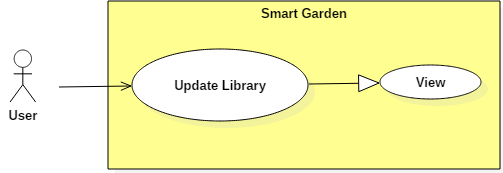


Figure 18. “Library View” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC02** | | | |
| **Use Case No.** | **UUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Library View | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User need to understand standard information of plants to manage system based on indexes of sensors. Library give user recommend information of tree.  **Goal:**  - User can understand easily about information of plants which they don’t need to find in other places outside system.  **Triggers:**  - NA  **Preconditions:**  - System must provide information on core database which join with a personal database for each user.  **Post Conditions:**  - Success: User can view list of plants which is on core database and personal database.  - Fail: Display a popup message for errors. Log errors to system in case conflict between personal database and core database.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | User choose one plant which they want | Navigate to Information of Plants  Show standard information which is load from core database |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | User touch “Add” button | Navigate to Add new page |  |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | User touch and hold on one or many kind of tree | Choose plants which user pressed | | 3 | User touch “Delete” button | Delete plants which user pressed  Update to database |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Errors from join database or system  Log errors. | | 2 | Waiting for information from database | Display errors message |  |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | User choose one plant which they want | Can’t get information of that plant | | 3 | Waiting for information | Display errors message |   **Relationships:**  - NA | | | |

Table 9. ”Library View” specification.

2.3.2.1 <User> Library Update

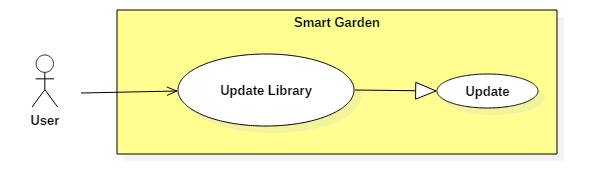


Figure 18. “Library Update” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC03** | | | |
| **Use Case No.** | **UUC03** | **Use Case Version** | 2.0 |
| **Use Case Name** | Library Update | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User want to change standard information of specific plant. This use case allow user to change some information which is different with core database  **Goal:**  - System can work based on indexes which user changed. User can change threshold for sensors in system with their wish.  - User can change some indexes which they want. That help people have a real control with system to suitable with different conditions of garden.  **Triggers:**  - NA  **Preconditions:**  - Information which will change by user is exist on core database.  - The changed of user is not affect to data on core database, it will be added to personal database  **Post Conditions:**  - Success: User change data successful. Changed data is not affect to core database.  - Fail: Display a popup message for errors. Log errors to system in case conflict between personal database and core database.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch on one plant which they want | Navigate to Information of Plants  Show standard information which is load from core database | | 3 | Touch on each field of indexes and edit information | Validation indexes in fields. | | 4 | Touch “Done” button | Update to database  Display a message “Update Successful” |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | User touch and hold on an object | Turn on Delete button and allow to choose more than one object | | 3 | Touch on Delete button | Delete objects which user chose |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | User choose one plant which they want | Navigate to Information of Plants  Show standard information which is load from core database | | 3 | Touch on each field of indexes and edit information.  Input wrong data types of field | Validation data which user input  Display an error message. |  |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | User choose one plant which they want | Navigate to Information of Plants  Show standard information which is load from core database | | 3 | Touch on each field of indexes and edit information | Validation indexes in fields. | | 4 | Touch “Done” button | Update to database fail  Display a message “Update Fail” |   **Relationships:**  - NA | | | |

Table 9. ”Library Update” specification.

2.3.2.3 <User> Library Insert

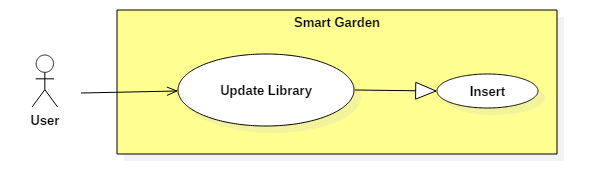


Figure 18. “Library Insert” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC04** | | | |
| **Use Case No.** | **UC04** | **Use Case Version** | 2.0 |
| **Use Case Name** | Library Insert | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - Allow user can add more to their database (personal) with some kind of plants which they want  **Goal:**  - User can control the system with any kind of plant, there are not depend too much on core database of system  **Triggers:**  - NA  **Preconditions:**  - System must provide a personal database for each user to avoid conflict between difference users on core database.  **Post Conditions:**  - Success: User can add any plants which they want to personal database which system can work on. Users set threshold of sensors to control actuators by themself.  - Fail: Display a popup message for errors. Log errors to system in case conflict between personal database and core database.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch “Add” button | Navigate to Add new screen with fields which ask to input full data on | | 3 | User fill all fields in screen | Validation data which user input | | 4 | User touch “Add” button | Insert new data to database.  Display message “Add successful” |   **Alternative Scenario:**  - NA  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch “Add” button | Navigate to Add new screen with fields which ask to input full data on | | 3 | Users don’t fill enough fields in screen or input wrong types of data | Validation data which user input  Display message errors to user’s screen |  |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library page | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch “Add” button | Navigate to Add new screen with fields which ask to input full data on | | 3 | Waiting for information | Display errors message | | 4 | User touch “Add” button | Insert to database fail  Log errors  Display message error to user |   **Relationships:**  - NA | | | |

Table 9. ”Library Insert” specification.

2.3.2.4 <User> Library Delete

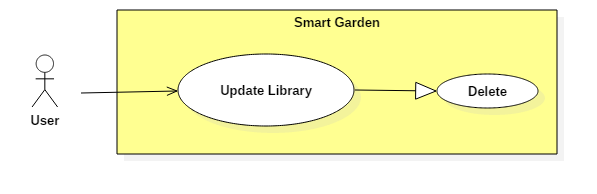


Figure 18. “Library Delete” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC05** | | | |
| **Use Case No.** | **UC05** | **Use Case Version** | 2.0 |
| **Use Case Name** | Library Delete | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - Allow user to delete one or many plants’ information in personal database  **Goal:**  - User can delete any data of plants which they want. That is not affect to core database of all system.  **Triggers:**  - NA  **Preconditions:**  - Information which will delete by user is exist.  **Post Conditions:**  - Success: User can delete one or many plants which they want.  - Fail: Display a popup message for errors. Log errors to system in case conflict between personal database and core database.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch and hold on one object plant in list | Turn on Delete button and allow to choose more than one object | | 3 | User touch on Delete button | Update to database to delete one or many object which user chose.  Update list of plants in Library Screen |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | User touch on one object of plant | Navigate to Library View Screen |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch and hold on one object plant in list | Turn on Delete button and allow to choose more than one object | | 3 | User touch on Delete button | Fail when update to database to delete one or many object which user chose.  Display a message error to user “Delete unsuccessful” |  |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Access Library screen | Get Joined database between core database and personal database.  Show list of plants | | 2 | Touch and hold on one object plant in list | Turn on Delete button and allow to choose more than one object | | 3 | User touch on Delete button | Update to database to delete one or many object which user chose.  Fail when update list of plants in Library Screen  Display a message error to user |   **Relationships:**  - NA | | | |

Table 9. ”Library Delete” specification.

2.3.3. <User> View Status Use Case

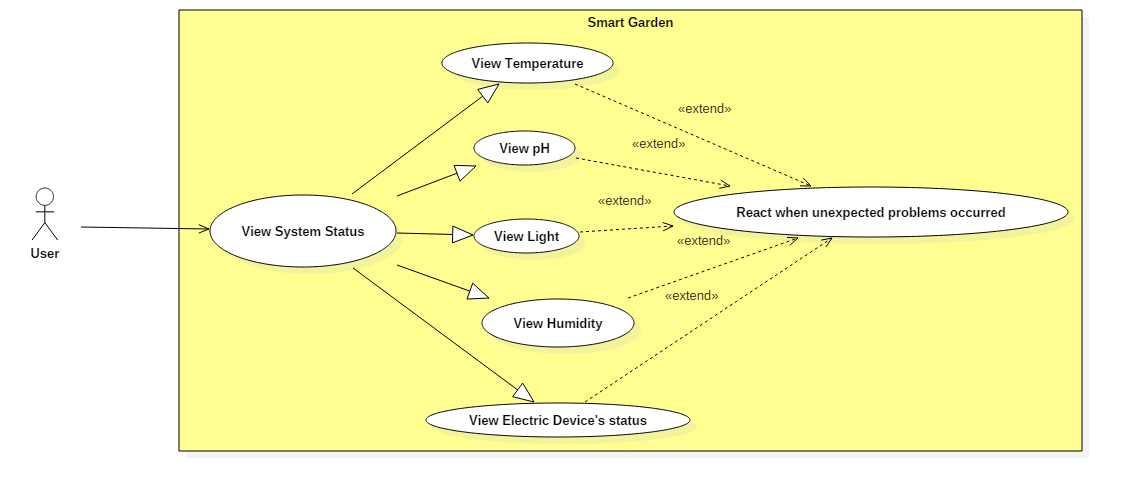


Figure 20. <User> View Status use case

2.3.3.1. View Temperature

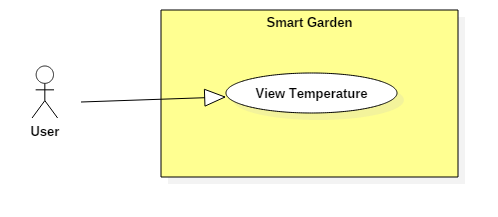


Figure 21. “View Temperature” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC006** | | | |
| **Use Case No.** | **UC0006** | **Use Case Version** | 2.0 |
| **Use Case Name** | View Temperature | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User can monitor temperature via web server and mobile application.  **Goal:**  - Users can view temperature every time when they access to system  **Triggers:**  - After user logged to the system and touch on “Sensor” tab.  **Preconditions:**  - User must logged into the system.  - Temperature sensors must be connected to the system.  **Post Conditions:**  - Success: Information must be displayed in real time  - Fail: Display a message error to user, resent command to check status and ask user want to get indexes of temperature sensor again.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Temperature” line | Send command to get new index from temperature sensor  Display a message with new index |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on line of other sensors | Send command to get new index from other sensors  Display a message with new index |  |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Send command to get new index from all sensors  Refresh Sensor screen with new indexes |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Temperature” line | Fail when send command to get new index from temperature sensor  Display a message error about connect between sensor and controller. |   **Relationships:**  - N/A | | | |

Table 11. “View Temperature” specification.

2.3.3.2. View Humidity

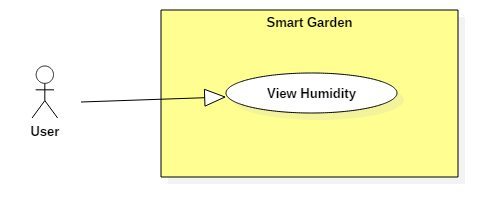


Figure 21. “View Humidity” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC007** | | | |
| **Use Case No.** | **UC0007** | **Use Case Version** | 2.0 |
| **Use Case Name** | View Humidity | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User can monitor humidity in statistic via web server and mobile application.  **Goal:**  - Users can view humidity every time they access the system  **Triggers:**  - After user logged to the system and touch on “Sensor” tab.  **Preconditions:**  - User must logged into the system.  - Humidity sensors must be connected to the system.  **Post Conditions:**  - Success: Information must be displayed in real time  - Fail: Display a message error to user, resent command to check status and ask user want to get indexes of humidity sensor again.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Humidity” line | Send command to get new index from humidity sensor  Display a message with new index |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on line of other sensors | Send command to get new index from other sensors  Display a message with new index |  |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Send command to get new index from all sensors  Refresh Sensor screen with new indexes |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Humidity” line | Fail when send command to get new index from humidity sensor  Display a message error about connect between sensor and controller. |   **Relationships:**  - N/A | | | |

Table 11. “View Humidity” specification.

2.3.3.3. View Soil Moisture

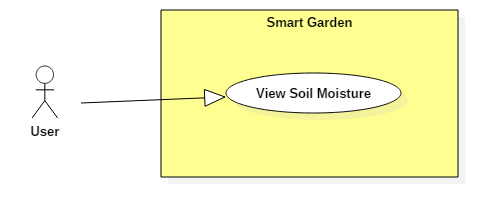


Figure 21. “View Soil Moisture” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC008** | | | |
| **Use Case No.** | **UC0008** | **Use Case Version** | 2.0 |
| **Use Case Name** | View Soil Moisture | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User can monitor soil moisture via web server and mobile application.  **Goal:**  - Users can view soil moisture every time they access to system.  **Triggers:**  - After user logged to the system and touch on “Sensor” tab.  **Preconditions:**  - User must logged into the system.  - Soil moisture sensors must be connected to the system.  **Post Conditions:**  - Success: Information must be displayed in real time  - Fail: Display a message error to user, resent command to check status and ask user want to get indexes of soil moisture sensor again.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Temperature” line | Send command to get new index from temperature sensor  Display a message with new index |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on line of other sensors | Send command to get new index from other sensors  Display a message with new index |  |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Send command to get new index from all sensors  Refresh Sensor screen with new indexes |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Soil Moisture” line | Fail when send command to get new index from temperature sensor  Display a message error about connect between sensor and controller. |   **Relationships:**  - N/A | | | |

Table 11. “View Soil Moisture” specification.

2.3.3.4. View Light

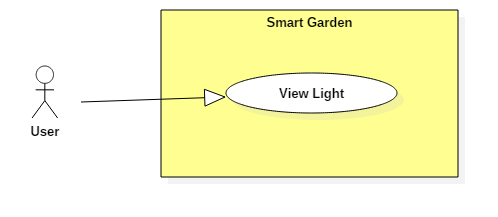


Figure 21. “View Temperature” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC009** | | | |
| **Use Case No.** | **UC0009** | **Use Case Version** | 2.0 |
| **Use Case Name** | View Light | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User can monitor light via web server and mobile application.  **Goal:**  - Users can view light every time after they access to the system  **Triggers:**  - After user logged to the system and touch on “Sensor” tab.  **Preconditions:**  - User must logged into the system.  - Light sensor must be connected to the system.  **Post Conditions:**  - Success: Information must be displayed in real time  - Fail: Display a message error to user, resent command to check status and ask user want to get indexes of light sensor again.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Light” line | Send command to get new index from temperature sensor  Display a message with new index |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on line of other sensors | Send command to get new index from other sensors  Display a message with new index |  |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Send command to get new index from all sensors  Refresh Sensor screen with new indexes |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Light” line | Fail when send command to get new index from temperature sensor  Display a message error about connect between sensor and controller. |   **Relationships:**  - N/A | | | |

Table 11. “View Light” specification.

2.3.3.4. Refresh View Sensor

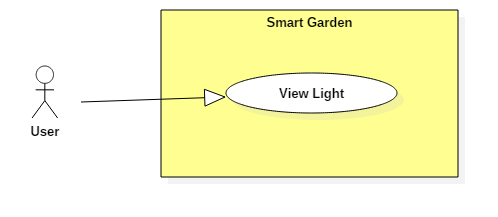
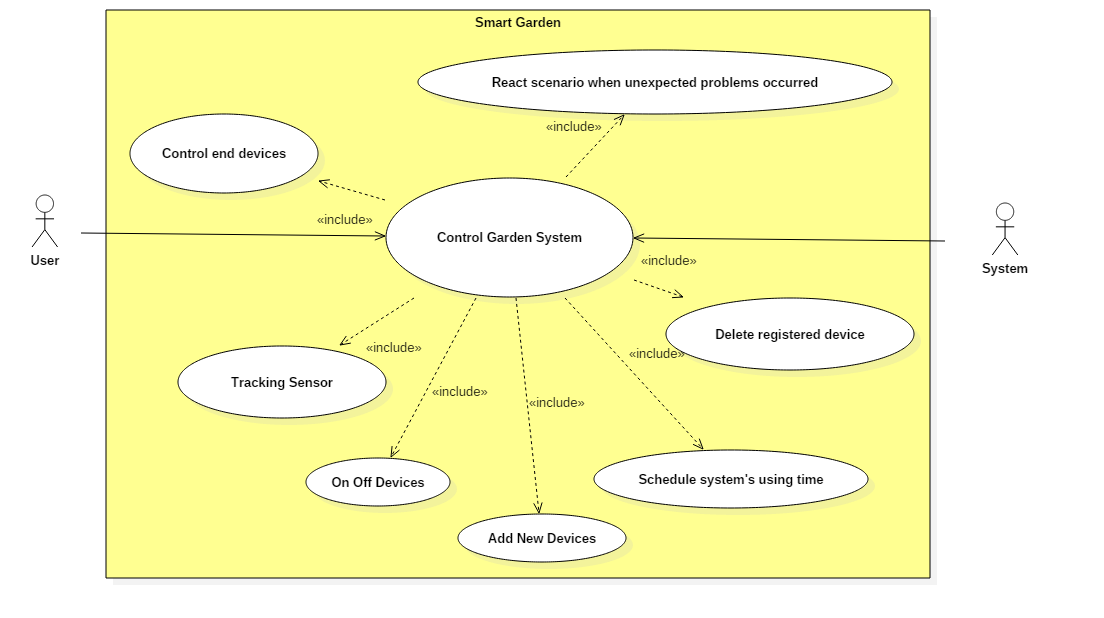


Figure 21. “Refresh View Sensor” use case

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC010** | | | |
| **Use Case No.** | **UC0010** | **Use Case Version** | 2.0 |
| **Use Case Name** | Refresh View Sensor | | |
| **Author** | Pham Hoang Chinh | | |
| **Date** | 30/1/2015 | **Priority** | Normal |
| **Actor:**  - User  **Summary:**  - User can monitor all indexes of sensors.  **Goal:**  - Users can view all index at any time when they want  **Triggers:**  - After user logged to the system and touch on “Sensor” tab.  **Preconditions:**  - User must logged into the system.  - Light sensor must be connected to the system.  **Post Conditions:**  - Success: Information must be displayed in real time  - Fail: Display a message error to user, resent command to check status and ask user want to get indexes of light sensor again.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Send command to get new indexes from sensors  Display a message with new indexes |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on other lines in Sensor Screen | Send command to get new index from other sensors  Display a message with new index |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Touch on “Sensor” tab | Navigate to Sensor screen  Display current indexes of sensors in nearly time. | | 2 | Touch on “Refresh” button | Fail when send command to get new index from sensors.  Show any index which they have.  Display a message error about connect between sensor and controller. |   **Relationships:**  - N/A | | | |

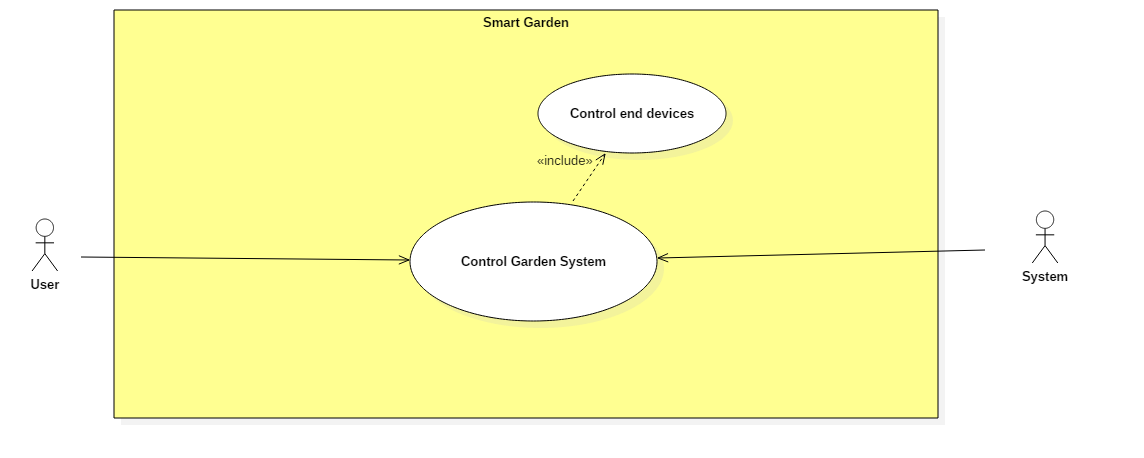
Table 11. “Refresh View Sensor” specification

**2.3.4. <User>,<System> Control Garden System**



**Figure 17 . . <User>,<System> Control Garden System**

**2.3.1.1. Control end devices**

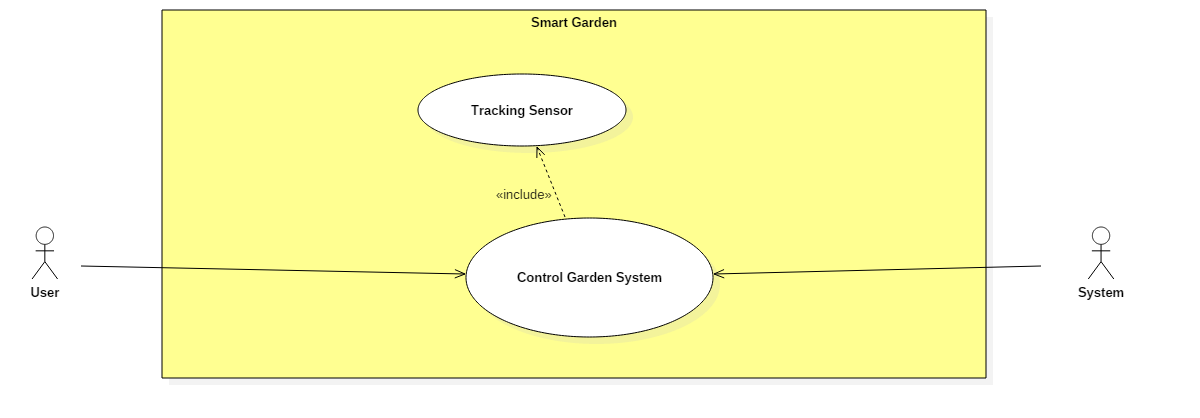


**Figure 18. “Control end devices” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC01** | | | |
| **Use Case No.** | **UUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | Control end devices | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - End devices can be controlled by user (manual) or system (automatically).  **Goal:**  - User will control end devices in manual.  - System will control end devices in automatically.  **Triggers:**  - Sensors send information to system.  - User turn on in manual mode.  **Preconditions:**  - System is connected with all end devices and sensors.  **Post Conditions:**  - Success: System operating successfully.  - Fail: Show error message.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - User goes manager a Zone. | - System is connecting with all devices. | | 2 | - Default system is controlling end devices automatically. | - System control end devices based on information taking from sensors. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - User goes manager a Zone. | - System is connecting with all devices. | | 2 | - User turn on manual mode (on/off on end devices) | - System control end devices based on what user choose. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | - System can’t connect to sensors | - Message will be shown “System can’t connect to sensors” message. | | 2 | - System can’t connect to end devices | - Message will be shown “System can’t connect to end devices”. |   **Relationships:**  - Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  - All end devices and sensors must be connected with system and work well. | | | |

Table 9. “Control end devices” specification.

**2.3.1.2. Tracking sensor**

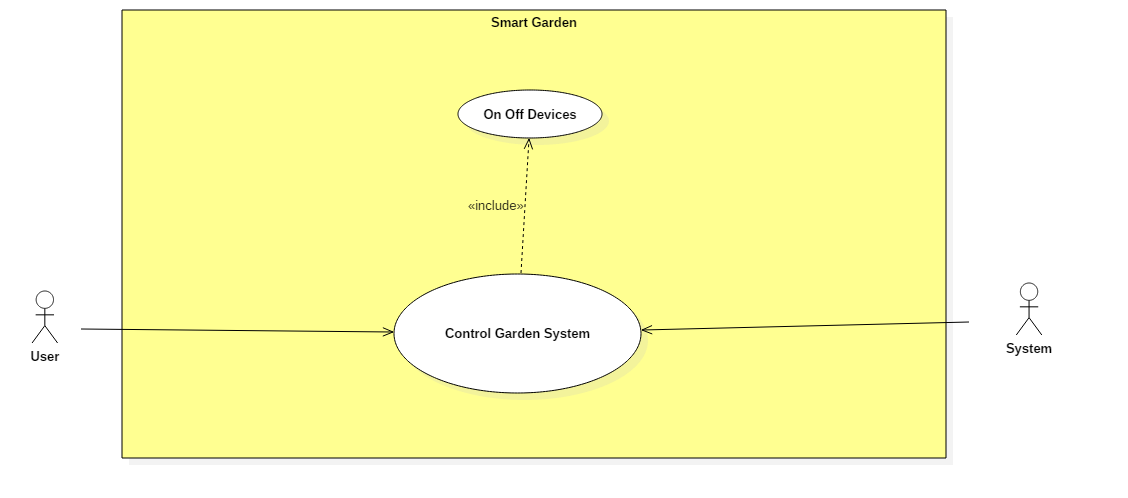


**Figure 19. “Tracking sensor” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UUC02** | | | |
| **Use Case No.** | **UUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Tracking sensor | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - User/System get information from sensor.  **Goal:**  - Show information from sensor.  **Triggers:**  - System is connected with sensor.  **Preconditions:**  - All sensors must be connected with system and work well.  **Post Conditions:**  - Success: Loading information to system.  - Fail: Can’t show information to system or user. Dialog will show message base on exception.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Choose zone to manage. | - Loading information from choosing zone’s sensors. | | 2 |  | - Control end device based on information loading and default index. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Choose zone to manage. | - Loading information from choosing zone’s sensors. | | 2 | - Control end devices in manual mode. | - Control end device based on user. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | - System can’t connected with sensors. | - Message will be shown “can’t connected with sensors”. |   **Relationships:**  - Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  - All sensors must be connected with system and work well. | | | |

Table 10. “Tracking sensor” specification.

**2.3.1.3. On off devices**

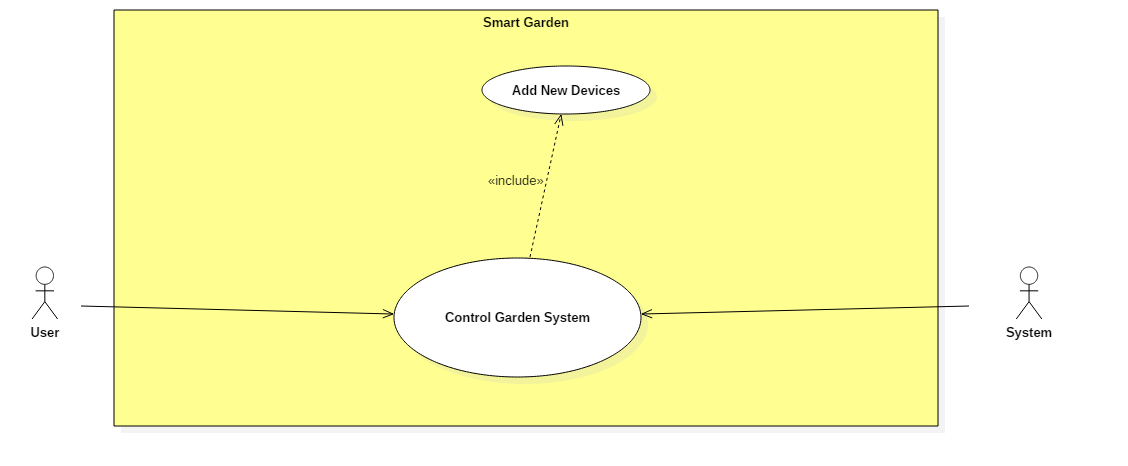


**Figure 21. “On off devices” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC01** | | | |
| **Use Case No.** | **HUC01** | **Use Case Version** | 2.0 |
| **Use Case Name** | On off devices | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - This function will be control devices (turn on or off) automatically or manual.  **Goal:**  - Devices turn on/off automatically by using sensors.  - Devices turn on/off in manual mode.  **Triggers:**  - Devices is connect to the system.  - Sensors is connect to the system.  **Preconditions:**  - System can read information from sensors.  **Post Conditions:**  - Success: Device is turn on/off.  - Fail: Can’t connected with device. Warning message will be shown “Can’t connected with device”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Choose Zone. | - Connect with all device in this Zone. | | 2 | - Turn on/off device. | - Device turn on/off. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | - Turn on/off device. | - Message will be shown “Can’t connect with device”. |   **Relationships:**  - Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  **-** Device turn on/off automatically based on information taking from sensors. Or user can turn it on/off manual. | | | |

Table 11. “On off devices” specification.

**2.3.1.4. Add new devices**

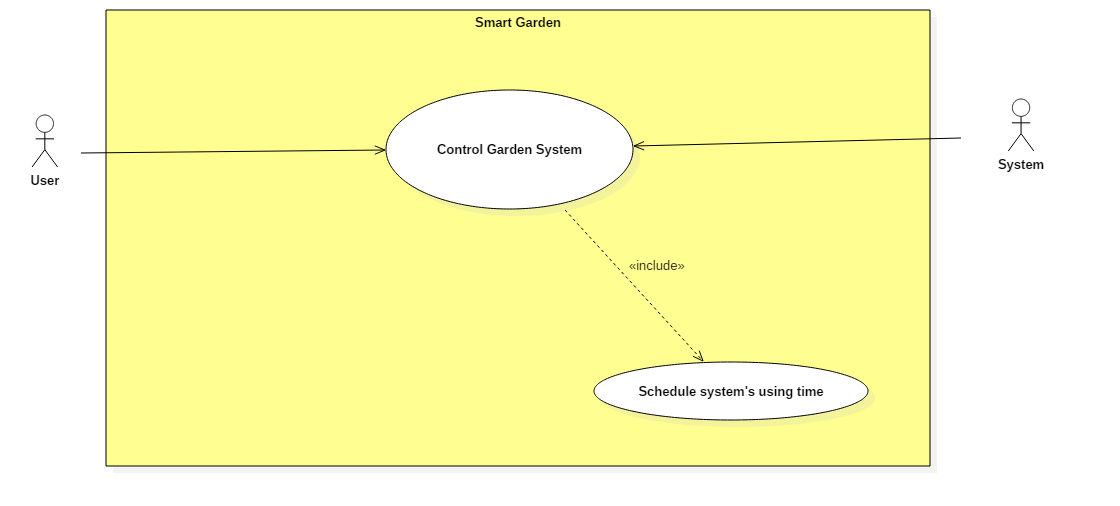


**Figure 22. “Add new devices” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC02** | | | |
| **Use Case No.** | **HUC02** | **Use Case Version** | 2.0 |
| **Use Case Name** | Add new devices | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - Add new devices to control garden system.  **Goal:**  - Add new devices to control garden system. - Control this device by using control garden system.  **Triggers:**  - N/A  **Preconditions:**  - N/A  **Post Conditions:**  - Success: Add new devices to control garden system successfully. Message will be shown “Add new devices to control garden system successfully”.  - Fail: Add new devices to control garden system fail. Message will be shown “Add new devices to control garden system fail”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Connect new device with system. |  | | 2 | - Add new device. | - Message will be shown “New device is added”. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Connect new device with system. |  | | 2 | - Add new device. | - Message will be shown “New device is not added”. |   **Exceptions:**  **-** N/A  **Relationships:**  **-** Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  **-** New devices must be connected with the system. | | | |

Table 12. “Add new devices” specification.

**2.3.1.5. Schedule system’s using time**

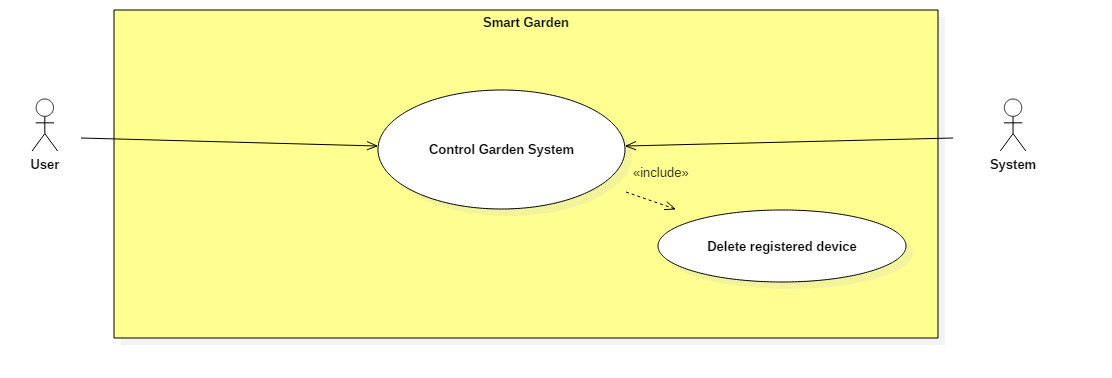


**Figure 23. “Schedule system’s using time” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC03** | | | |
| **Use Case No.** | **HUC03** | **Use Case Version** | 2.0 |
| **Use Case Name** | Schedule system’s using time | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - Schedule to manure.  **Goal:**  - Using time schedule to manure.  **Triggers:**  - N/A.  **Preconditions:**  - Schedule coincides with the time.  **Post Conditions:**  - Success: Manure successfully.  - Fail: Can’t manure. Warning message will be shown “Can’t manure, please check your devices.”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Choose Zone. | - Connect with all device in this Zone. | | 2 | -Choose fertilizer. | - Device turn on. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | - Choose Zone. | - Connect with all device in this Zone. | | 2 | -Choose fertilizer. | - Message will be shown “Can’t manure, please check your devices.”. |   **Relationships:**  **-** Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  **-** Schedule coincides with the time or user can turn it on/off manually. | | | |

Table 13. “Schedule system’s using time” specification.

**2.3.1.6. Delete registered device**

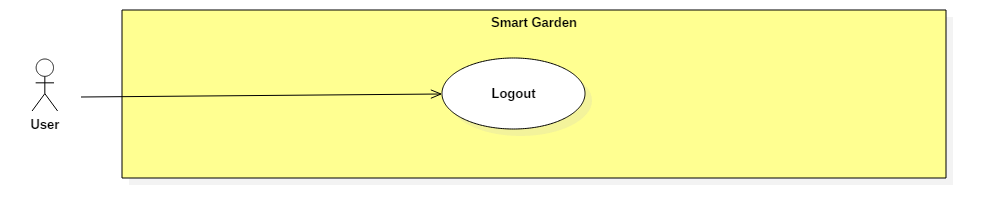


**Figure 24. “Delete registered device” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC04** | | | |
| **Use Case No.** | **HUC04** | **Use Case Version** | 2.0 |
| **Use Case Name** | Delete registered device | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  - System  **Summary:**  - Registered device can be deleted from system.  **Goal:**  - Delete registered device from system.  **Triggers:**  - N/A  **Preconditions:**  - N/A  **Post Conditions:**  - Success: Registered device deleted successfully. Message will be shown “Device deleted successfully”.  - Fail: Registered device deleted fail. Message will be shown “Device deleted fail”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Manager devices. | - Connect with all devices. | | 2 | - Delete device. | - Delete device from system. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No. | Actor Action | System Response | | 1 | - Manager devices. | - Connect with all devices. | | 2 | - Delete device. | - Message will be shown “Device deleted fail”. |   **Relationships:**  **-** Have <<include>> relationship with “Control Garden System” use case.  **Business Rules:**  **-** N/A | | | |

Table 14. “Delete registered device” specification.

**2.3.2. User logout**



**Figure 25. “User logout” use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – HUC05** | | | |
| **Use Case No.** | **HUC05** | **Use Case Version** | 2.0 |
| **Use Case Name** | User logout | | |
| **Author** | NghiHH | | |
| **Date** | 30/1/2015 | **Priority** | High |
| **Actor:**  - User  **Summary:**  - Logout from system  **Goal:**  - User can logout.  **Triggers:**  - N/A.  **Preconditions:**  - User login.  **Post Conditions:**  - Success: Logout successfully. Message will be shown “Logout successfully”.  - Fail: Can’t logout. Warning message will be shown “Can’t logout. Please try again”.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Logout | - Logout from system  - Message will be shown “Logout successfully”. |   **Alternative Scenario:**  **-** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Logout | - Logout from system  - Warning message will be shown “Can’t logout. Please try again”. |   **Relationship:**  **-** N/A  **Business Rules:**  **-** Logout from system. | | | |

Table 15. “Logout” specification.

3. Software System Attribute

3.1 Usability

The system is expected to apply for garden of users who has a garden in home but they don’t have enough time to take care or tracking usually. There are many plant’ indexes which gardener are tracking to get higher productivity in agriculture but the system has some basic sensors which can help users keep plants live and limit harmful weather with their garden.

System provide friendly GUI for users on their mobile which is closely with people in nowadays.

3.2 Reliability

System is expected to run continuously for years without errors (or in some cases recover by themselves if an error occurs).

Ensure that when system starts a "mission", it has a high probability of completing that mission without experiencing a failure.

System can safely be shut down for repair, or another way to repair without stopping system.

3.3 Availability

The server shall be working 24/7. When the system goes in under-maintenance, the page or application will display message "System is maintaining at the moment. Please check again later". Current system will work directly with index from sensors.

3.4 Security

Guarantee the data and application protection from being stolen and modified by encoding and decoding data.

3.5 Maintainability

All code shall be fully documented. All program files shall include comments concerning authorship and date of last change.

The code shall be modular to permit future modifications.

3.6 Portability

The system shall be designed to control and run on Web platform, Android platform. Make sure that control device must be portable device as smartphone, tablet…

Provided hardware can be plug-and-play

3.7 Performance

This is the system's performance characteristics:

Capacity: 1 end user a time.

Response time for a transaction:

Average: second

Maximum: seconds