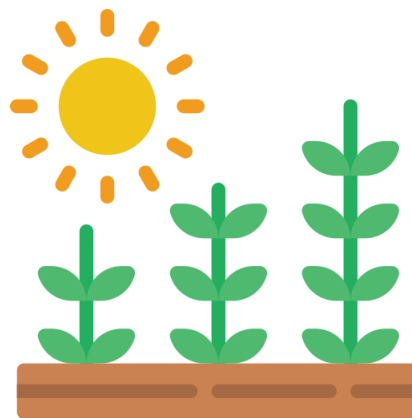


HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY



GROUP 9 – FINAL REPORT

SMART GARDEN APP



Lecturer:
Assoc. Prof. Quan Thanh Tho

Course:
**PRACTICE ON SOFTWARE
ENGINEERING**

July 29th, 2020

GROUP MEMBER

GROUP 9 -- MEMBER LIST			
No.	Name	ID	Email
1	Truong Minh Duy	1652113	1652113@hcmut.edu.vn
2	Ngo Nguyen Duy An	1752063	an.ngo1999@hcmut.edu.vn
3	Pham Tuan Anh	1651006	1651006@hcmut.edu.vn
4	Vo Dong Ho	1752219	ho.vo.2801@hcmut.edu.vn
5	Tran Minh Hung	1652271	1652271@hcmut.edu.vn

Contents

1	Introduction	3
1.1	About	3
1.2	Device lists	3
2	Requirements	4
2.1	Functional requirements	4
2.2	Non-functional requirements.....	4
3	Task assign	5
3.1	Device tasks	5
3.2	Use-case diagram/ scenario	5
4	Architecture design	60
4.1	Architectural pattern	60
4.2	Deployment diagram	60
5	Future improvements	61
6	Document history	62

1 Introduction

1.1 About

The goal of this project is make an app that allow real-time, interactive capturing, monitoring and interacting with the many important elements crucial to the growth of many agricultural products such as temperature, humidity, and light and enable user the ability to control IOT output device in real time. Moreover, based on the collected data, this app also recommends a threshold that is suitable for the environment or warns the user if the condition is over their pre-determined threshold, helps them have a better experience with gardening.

1.2 Device lists

After a discussion, our group choose using some devices below :

1. Temperature - air humidity
2. Light intensity sensor
3. Display light

2 Requirements

2.1 Functional requirements

These features will be in the final product :

1. **Log in/log out** : Users can log in/log out to the system
2. **Manually turn on/off devices** : This app allow users to turn on/off their devices
3. **Send notification to user and setting device properties** : The app also sent notification to user when light, air humidity or temperature hit a certain point which is determined by user setting.
4. **Register new device and plant** : Users can register new devices/new plants and view their status in the app
5. **Manage device and plant information** : User can change device or and plant information. This app also allow user to remove plant.
6. **Automatically control devices** : The app will automatically turn on or off the lighting system depend on the light settings of user
7. **View report (with 5 mode)**: The app generates the report to allow users to view information about the temperature, air humidity and light. The report also has 5 mode for choosing : overview (view nearest values), hourly (view measurements by hour in today), daily (view measurements by day in this month), monthly (view measurements by month in this year), custom (view measurements by hour in user's chosen day)
8. **Threshold recommendation system**: This app using AI to recommend temperature, humidity and light density threshold. The threshold will be recommended based on historical data. The specific AI requirements will covers in later part.
9. **Refresh view report and AI data** : This app has a refresh button inside view report to allow users to update the newest data for both views and recommend function.

2.2 Non-functional requirements

Our app have some non-functional requirements as we list below :

1. The app should be provided on Android device
2. App respond's time for any function should be less then 5 secs.
3. The app size is maximum 300MB

3 Task assign

3.1 Device tasks

After a discussion, we classify requirements as group and divide the tasks :

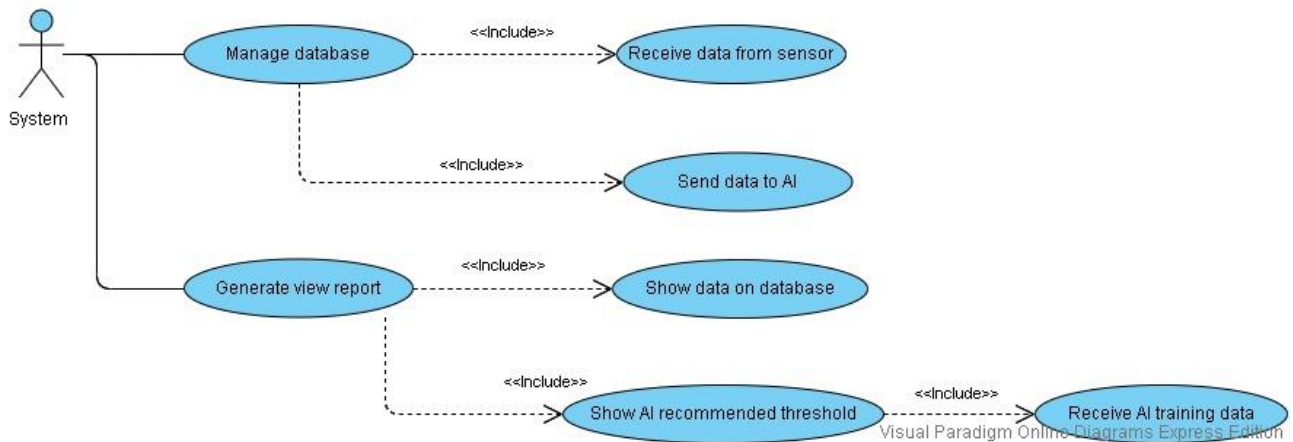
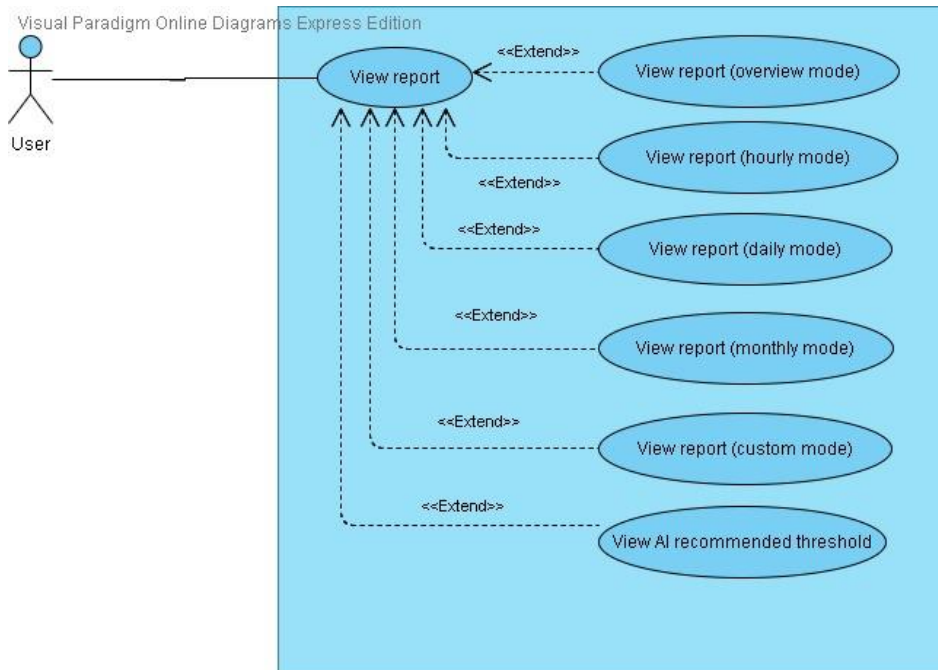
- **Truong Minh Duy** : View report, threshold recommendation system and refresh view report and AI data.
- **Vo Dong Ho** : Manually turn on/off devices.
- **Ngo Nguyen Duy An** : Register new device and plant and manage plant information.
- **Pham Tuan Anh** : Send notification to user.
- **Tran Minh Hung** : Manage device informations and automatically turn on or off devices

3.2 Use-case diagram/scenario

This section is about use-case detail for use-case the student is in-charged. We also notice that we do not show the detail of non-interactive requirements.

• Truong Minh Duy :

1. Use case diagrams



2. User-story

- As the user's view, the system should generate a report about the temperature, air humidity and light to allow user view the measurements.
- Moreover, the system should have various mode to satisfy many different requirements of users.
- This app requires user determine the threshold. However, as the user's view, normally I don't know how to determine this number. Moreover, all garden parameters are not fixed. For example, temperatures rarely change for a short time but often change for long periods, such as rising in the morning and falling in the evening. Therefore, using the fixed threshold is not an effective solution and our group decide to make recommended threshold system to help user solve this problem.
- This app needs data updating function to help users can update the latest data without having to reload the app. However, as we have an AI algorithm with some randomness, update data without changing anything make an AI result slightly differ which can lead to confusion. In addition, AI algorithm base on historical data and this properties require view report data and AI results update synchronous updates. As a result, our group provides the refresh button and let the user decide the update moment by yourself.

3. Use case scenarios

- Although we have three requirements, AI-related requirement can be considered as non-interactive requirements and the view report and refresh requirement can be merged. As a result, this part contains only one use-case scenarios.

Use Case ID:	1		
Use Case Name:	View reports about the temperature, air humidity and light		
Created By:		Last Updated By:	
Date Created:		Date Last Updated:	
Actors:	User		
Description:	Reports about the temperature, air humidity and light automatically generate after the user click a View report button		
Trigger:	None		
Preconditions:	User must logged into their account to view report		
Postconditions:	None		
Normal Flow:	<ol style="list-style-type: none">1. User click to the button “View Report”2. Report automatically generated after the user click a view report button .3. The user choose the device ID in spinner lists.4. The user choose the suitable mode5. The system access the database and get the measurements6. The system send data to AI server7. The system receive result from AI server and show in report8. The system allow the user view report on screen9. After view the report, user click ‘exit’ button to end.		
Alternative Flows:	Alternative 1 at step 9 9a. The user can change the device using spinner or change the mode using button. The flow will continue at step 5-9 Alternative 2 at step 9 9b. The user can click on “refresh” button to update the newest data. The flow will continue at step 5-9		
Exceptions:	None		
Notes and Issues:			

4. AI requirements

These features must be required in the final product :

1. **Handle various type of data :** AI can handle various type of data which contains both evenly/unevenly spaced time-series data
2. **Stable result:** If AI cannot provide the fixed result in multiple time run, the result should be stable (change in a little scale) to avoid making the user confuse.
3. **Make a prediction based on historical data:** AI only training based on historical data or it handles all past measurements of specific devices and independent with view mode.
4. **Remove outliers in very short time :** If we have some measurement in very short time, AI will choose the latest measurement. The short time is determined by the measurement time.
5. **Decreasing weight of past observation:** The measurements in the past observations should not be weighted equally, the AI uses the algorithm to decrease weights over time. These properties ensure the newest value make the largest effect on the final result.
6. **Detect trend and seasonality on data:** AI can handle well if exist the trend or seasonality on the data
7. **Reasonable training time:** Although the number of measurements can be changed, AI training time should have the reasonable time (rarely larger than 10s) to ensure the good user experience.

5. AI algorithm

5.1 Input/Output


Input: The time-series data, in other words, the pair value consist of both measurements and the date system get this measurement.

Output: The recommended threshold based on time-series forecasting.

5.2 Algorithms

5.2.1 Data preparation

At the beginning, data is stored in the database as the picture below:

<input type="checkbox"/>	 Edit	 Copy	 Delete	Light92	2020-10-08 17:05:30	L	300
<input type="checkbox"/>	 Edit	 Copy	 Delete	Light92	2020-10-08 17:21:30	L	300
<input type="checkbox"/>	 Edit	 Copy	 Delete	TempHumi1	2020-01-13 17:05:30	TH	20:82
<input type="checkbox"/>	 Edit	 Copy	 Delete	TempHumi1	2020-07-09 17:05:33	TH	55:20

measurement is 20:82, the application will send 20 as the temperature measurement and 82 as the humidity measurement.

After receiving the date and measurement data from the application, the AI server needs to convert it into a time-series format. At the final stage of this step, the data have the format at the picture below:

```
<TimeSeries>
{datetime.datetime(2019, 7, 11, 11, 45): 30,
 datetime.datetime(2019, 7, 11, 11, 45, 1): 32,
 datetime.datetime(2019, 7, 13, 3, 0): 34,
 datetime.datetime(2019, 7, 14, 0, 0, 20): 36,
 datetime.datetime(2019, 7, 15, 0, 0): 38,
 datetime.datetime(2020, 7, 16, 0, 0): 40}
</TimeSeries>
```

5.2.2 Check seasonality of original data

In time-series data, seasonality is the presence of variations that occur at specific regular intervals less than a year, such as weekly, monthly, or quarterly. Seasonality may be caused by various factors, such as weather, vacation, and holidays and consists of periodic, repetitive, and generally regular and predictable patterns in the levels of a time series.

The seasonality of the data contains two-part: seasonal time and seasonal value. Whereas seasonal time means that the data must be measure at a specific interval, seasonal value means the value has a repeat property. The constraint of seasonal time must strict, however, the constraint of seasonal value can be slightly flexible (the number can slightly differ)

Date	Measurement
2020-07-27 17:00:00	130
2020-07-27 17:01:00	150
2020-07-27 17:02:00	170
2020-07-27 17:03:00	130
2020-07-27 17:04:00	150

2020-07-27 17:05:00	170
2020-07-27 17:06:00	130

The above table is the example of seasonal data. Consider the final measurement row, if the value slightly change to 130.05, this data still seasonal. However, if we change the final data row to 2020-07-27 17:06:01, this data become unseasonal.

Check seasonal time: When the application sends data to the AI server, data is already sorted. Now, we create a new array that contains all the different times of two consecutive observations. Finally, we consider the standard deviation of the new array, if the standard deviation is very small (compare with 1), our group concludes data is seasonal.

Check seasonal value : Our group use two methods to check seasonal value. The data is considered seasonal value if both method return **True**:

Method 1 : Count mean crossing

- This method involves count the number of times where the graph of the function goes through mean value. If the result greater than 1, this method returns true. We also can use this method to find an approximate frequency of the signal
- Advantages: Fast, accurate (increasing with data length). Works well for long low-noise sines, square, triangle, ...
- Disadvantages: If the result of this method is true, we cannot sure that the data is seasonal. However, if the result of this method is false, we can conclude the data is unseasonal and does not need to use method 2 to confirm it.

Method 2 : Autocorrelation analysis

- **Autocorrelation**, also known as **serial correlation**, is the correlation of a signal with a delayed copy of itself as a function of delay. Informally, it is the similarity between observations as a function of the time lag between them.
- Advantages: This method is the one of the efficient methods for finding the true

fundamental of any repeating wave, even with strong harmonics or completely missing fundamental. The result is more reliable compare with method 1.

- Disadvantages: This method needs more time to compute compare with method 1. Moreover, using autocorrelation require to determine some parameters to achieve the results.

At the end of this step, if the original data is seasonal, AI should return the peak of the data. If not, AI will run into the next step.

5.2.3 Handle unevenly spaced data

Without loss of generalization, AI only consider unevenly spaced data as it is more general than evenly spaced data. There are two main approach :

Directly handle unevenly spaced data: In this approach, we can use sequence model to directly handle unevenly spaced data. However, the number of algorithms for this case is limit and hard to evaluate, so our group does not choose this approach.

Convert into evenly spaced data and handle missing value : A more common approach to analyzing unevenly spaced time series is to transform the data into equally spaced observations using some form of **interpolation** - most often linear - and then to apply existing methods for equally spaced data.

This group uses the second approach for this project. After data preparation, our group will convert data to a regularly-spaced time series using **a moving average** to avoid aliasing. Another problem is that we must determine the sampling period. A good sampling period can help detect outliers and reduce training time.

When the application sends data to the AI server, data is already sorted. Our group determines **measurement time** is the difference (in second) between the first value and final value. The larger the measurement time, the larger the sampling period.

Using “trial and error” method, the table below is the final decision of our group (all value in second)

Measurement time	Sampling period
$\leq 3,600$ s	1 s

> 3,600 s but ≤ 86,400 s	10 s
> 86,400 s but ≤ 800,000 s	100 s
> 800,000 s but ≤ 2,419,200 s	300 s
> 2,419,200 s but ≤ 10,000,000 s	1000 s
> 10,000,000 s	3000 s

At the end of this step, data successfully convert into evenly spaced data.

5.2.4 Handle missing value

After AI transforms unevenly spaced data to evenly spaced data, in practice, the regular table always missing the final value. The reason is that in the above step, our algorithm can make some cases in which measurement time is not divisible by the sampling period. The final value which belongs to the remaining time will not cover in this algorithm.

Our group consider some methods to solve this problem :

Propose new method to determine sampling period : In theory, if new algorithm can find the sampling period which is a divisor of measurement time, all value will not lost. However, if the measurement time is prime, there is no solution for this problem. Moreover, the sophisticated algorithm will require more time to compute and also increase training time.

Moving average from newest to latest : Another method that can be considered is to apply the moving average from newest to lowest. This method does not solve this problem, it is just to remove the latest value instead of the newest value as it has the lowest weight. However, if the application only has a few measurements in a short time, remove any value will make the result less reliable.

Adding the final value with some randomness: The final method is adding the final value into the table. Our group will add some new measurements with final value, but adding some value will make the prediction algorithm consider value unchanged and it affect the final result. Therefore, our group adding the final value with add/minus little random value. It makes the

final result more precise, but also make the AI results not fix. In the testing phase, our group proves that the result rarely changes in range 1, so it stable enough to avoid making user confusing.

In the final product, our group choosing third method for AI algorithm. At the end of this step, our data is ready for forecasting

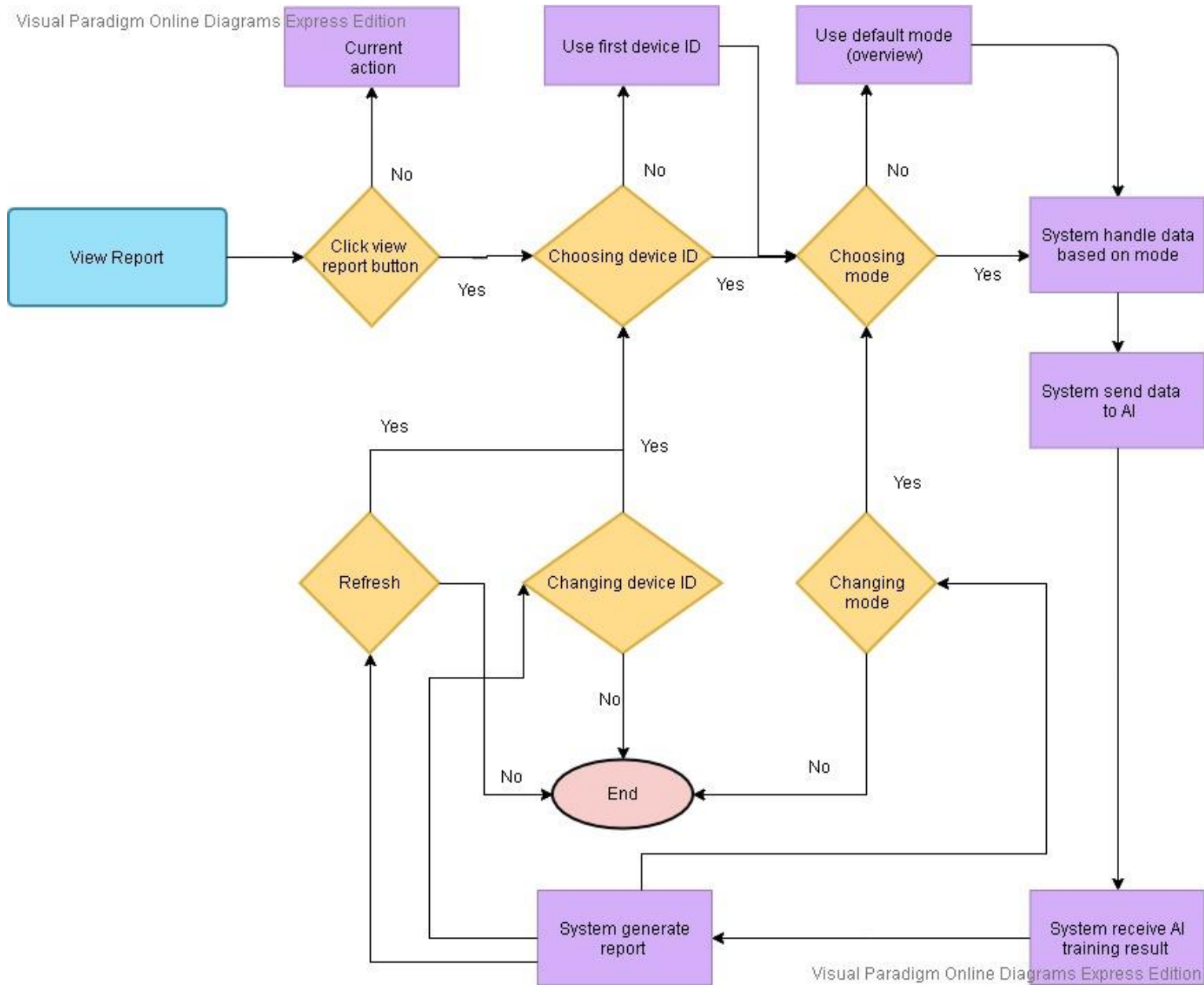
5.2.5 Recommend threshold

In the final step, our group test many algorithms for time-series forecasting such as ARIMA, Vector Autoregression (VAR),.. and decide to choose **Holt Winter's Exponential Smoothing** for the final product as it satisfies all requirements of our group.

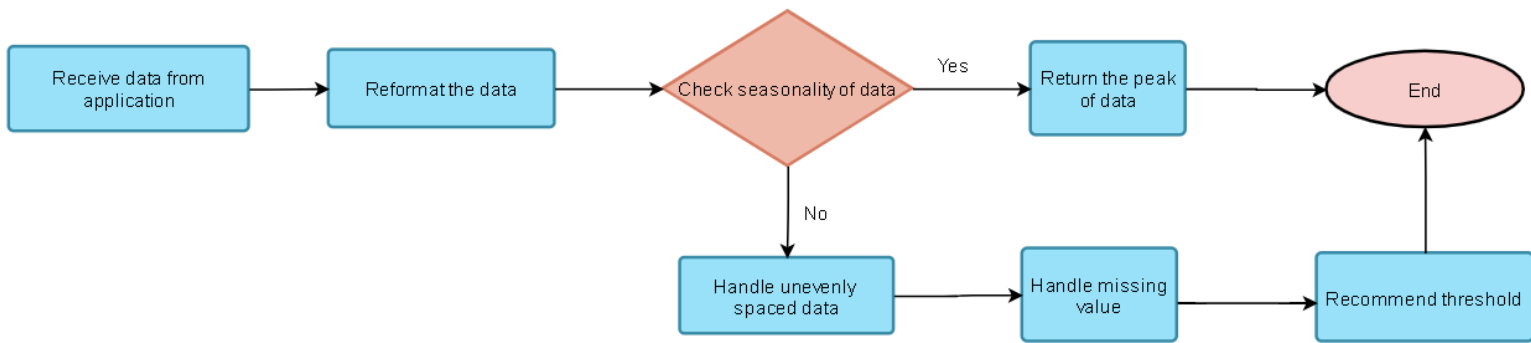
Holt Winter's Exponential Smoothing already handle trend in data, so our group does not need to use any additional algorithm for this task. After having the model using Holt Winter's Exponential Smoothing, our group using this model to predict the next 90 sampling period. Finally, as our threshold is the maximum threshold, the maximum value of 90 predicted value is the AI result.

6. Flow chart

For view report and refresh function :



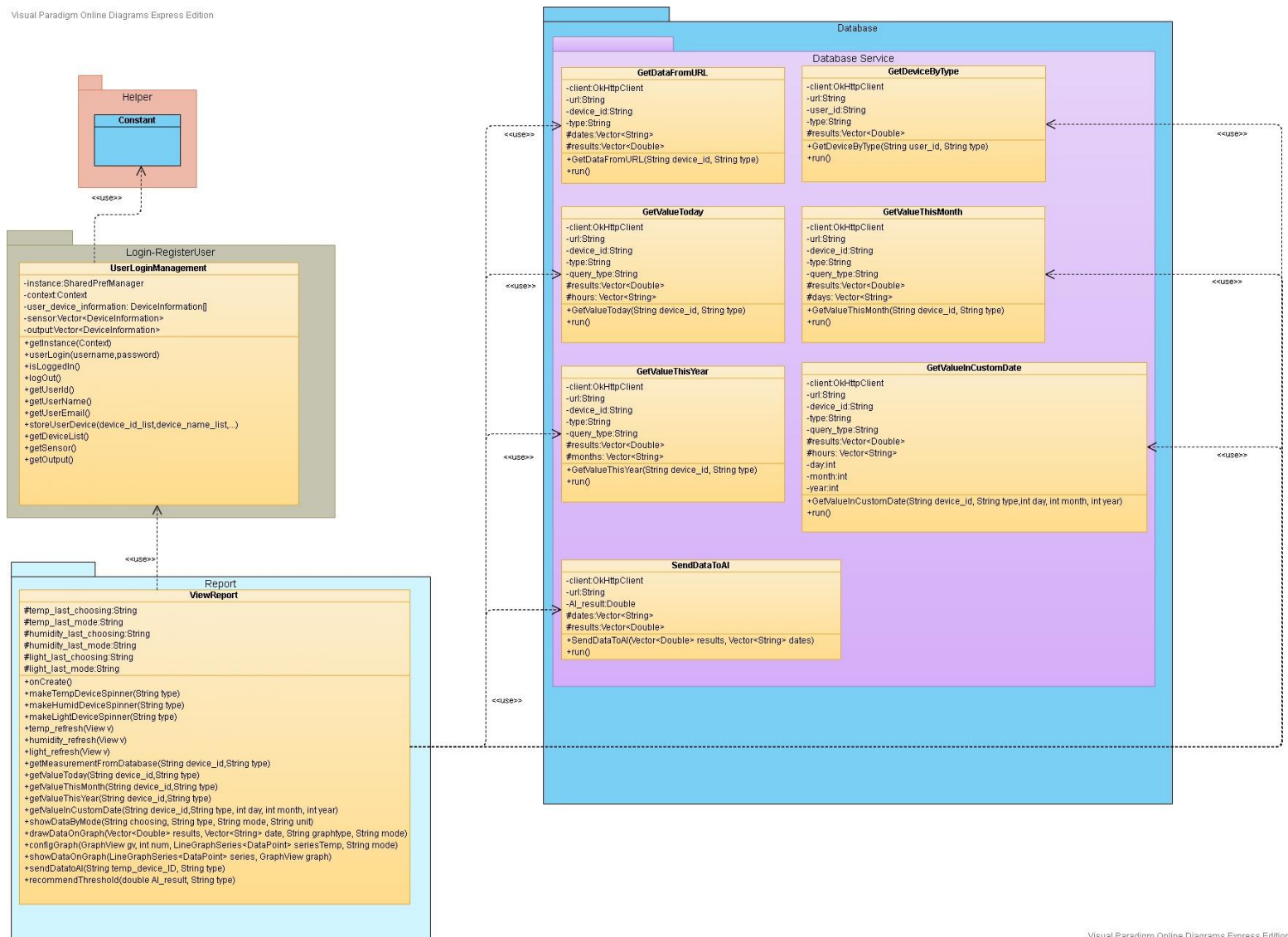
For training AI data :



7. Class diagram

The class diagram below is the class diagram for view report and refresh function. About the AI function, it is implemented as the Python file in server. Therefore, our group decide to implement it in deployment diagram

Visual Paradigm Online Diagrams Express Edition



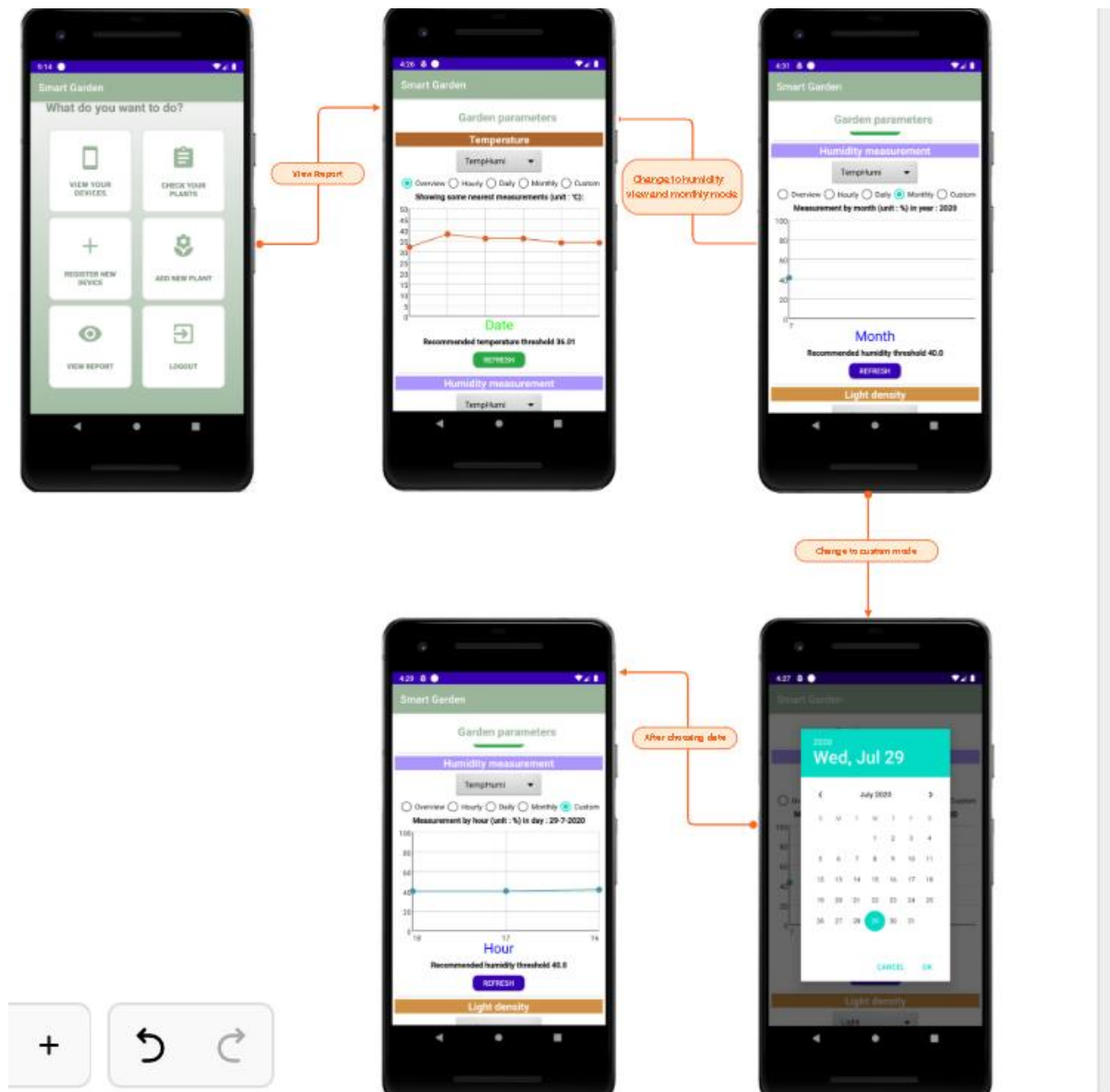
- **Class UserLoginManagement:** control user login profile
 - **getInstance(context):** Return the static instance of UserLoginManagement for the context to use
 - **userLogin(user_id, user_name):** put the user_id and user_name onto the SharedPreferences editor.
 - **isLoggedIn():** Check if a user is logged in or not

- **logOut():** clear stored parameter of the SharedPreferences editor when login
- **getUserID(..):** return user's id
- **Class Constant in package Helper:** get all constant value use for whole project from the configuration file. This class is implemented by other teammates.
- **Class ViewReport:** handle view report activity
 - **onCreate():** default function for activity
 - **makeTemp/Humidity/LightDeviceSpinner(type):** make a spinner list for temperature/humidity/light tab to allow user choosing device ID.
 - **getMeasurementFromDatabase(device_id,type):** Return measurement for overview mode
 - **getValueToday(device_id,type):** Return value for hourly mode
 - **getValueThisMonth(device_id,type):** Return value for daily mode
 - **getValueThisYear(device_id,type):** Return value for monthly mode
 - **getValueInCustomDate(device_id,type,day,month,year):** Return value for custom mode
 - **showDataByMode(choosing,type,mode,unit):** Base on the chosen device ID and mode of user, this function will call suitable function in five function above to return the desired value.
 - **drawDataOnGraph(results,date,graphtype,mode):** Make the dates into x-axis of the graph and call two below function to produce the graph.
 - **configGraph(graphview,num,seriesTemp,mode):** Handle the title, the font size and color of the graph.
 - **showDataOnGraph(series,graph):** show the point into the graph view
 - **sendDatatoAI(device_ID,type):** send data to AI server and return the result
 - **recommendThreshold(AI_result,type):** show the AI result to screen for user
- **All the class in Database Service:** have two function contain the constructor and function **run()** to perform function corresponding with the class name.

8. Mock up

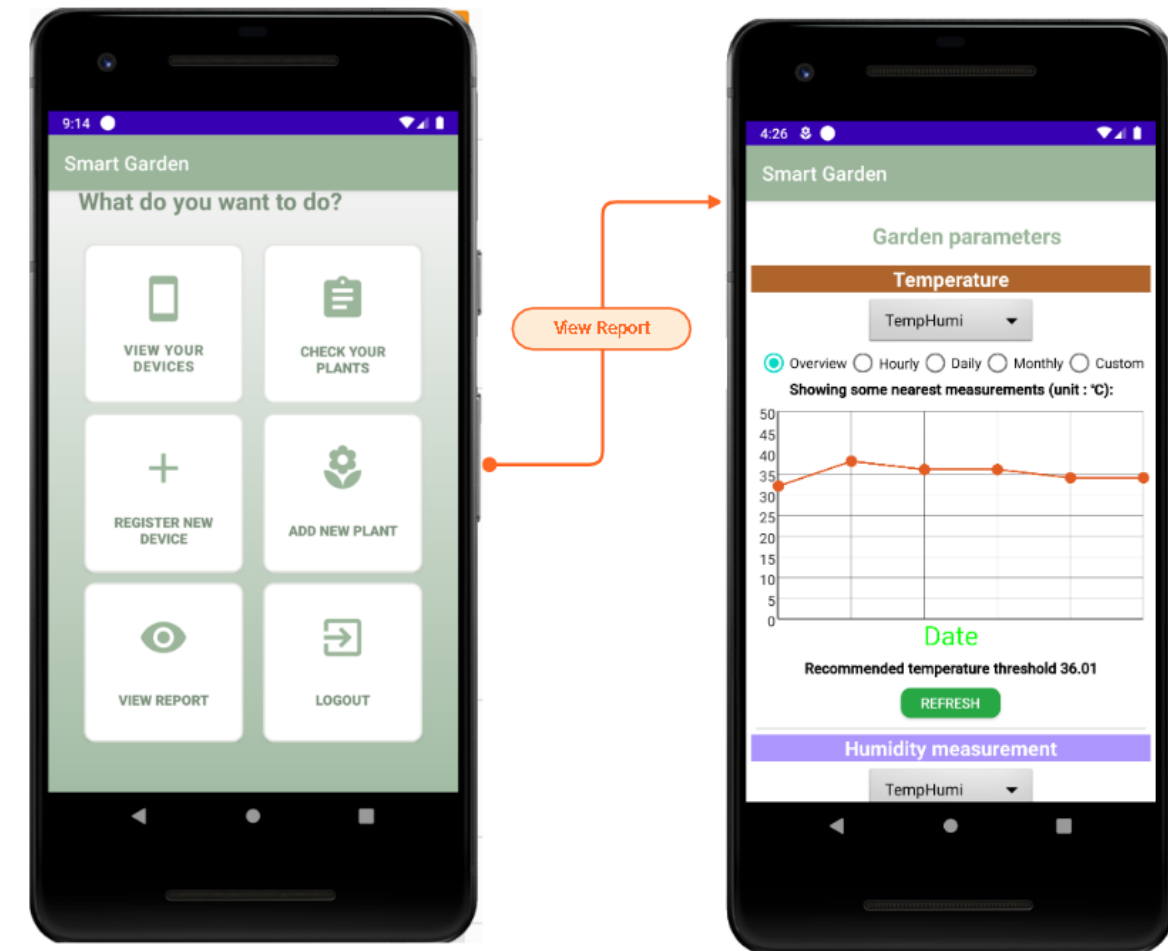
- For view report:

Include 4 step : View report - Change to humidity view and monthly mode - Change to custom mode - After choosing date



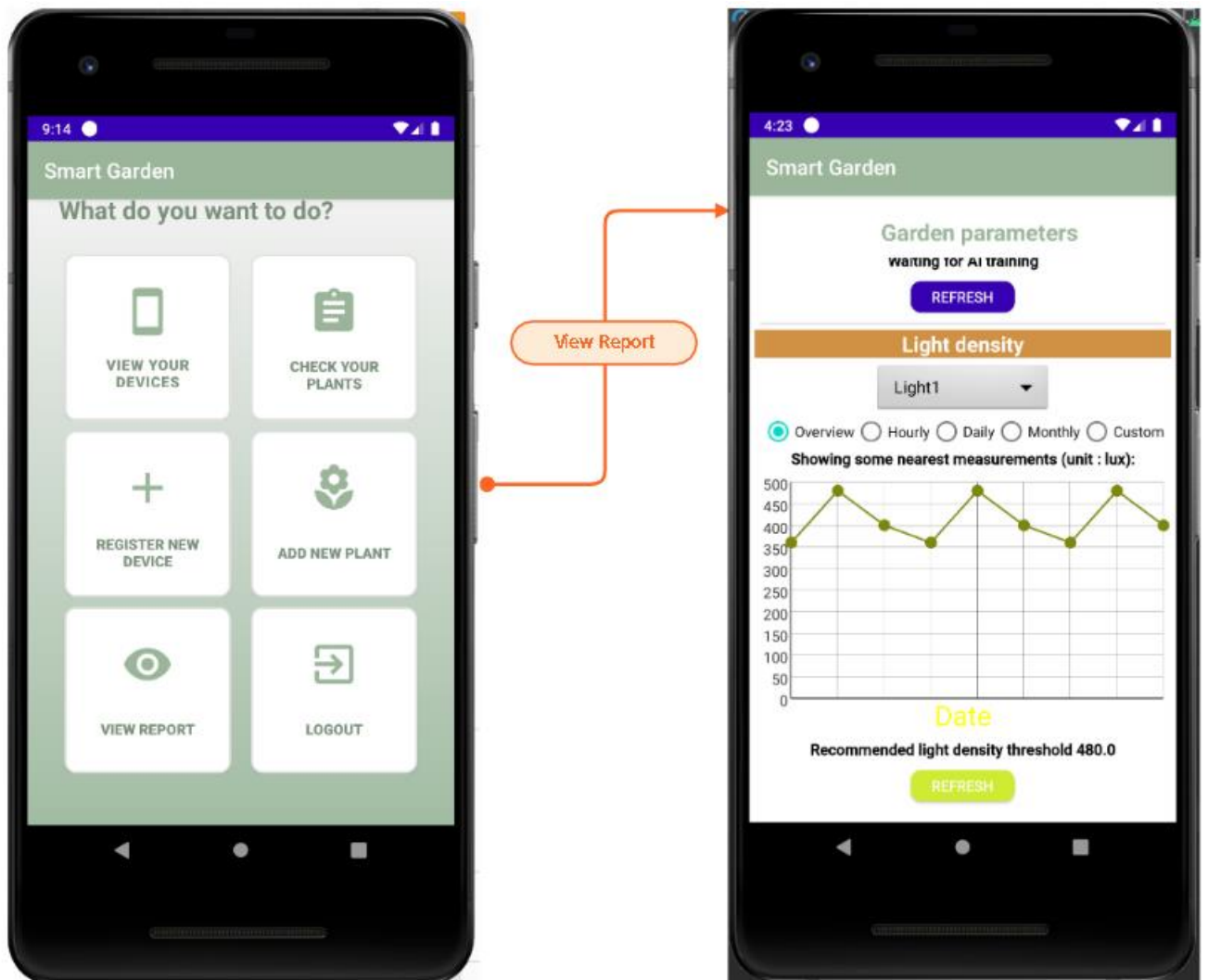
- For AI function:

AI result for normal data : The result near the nearest value



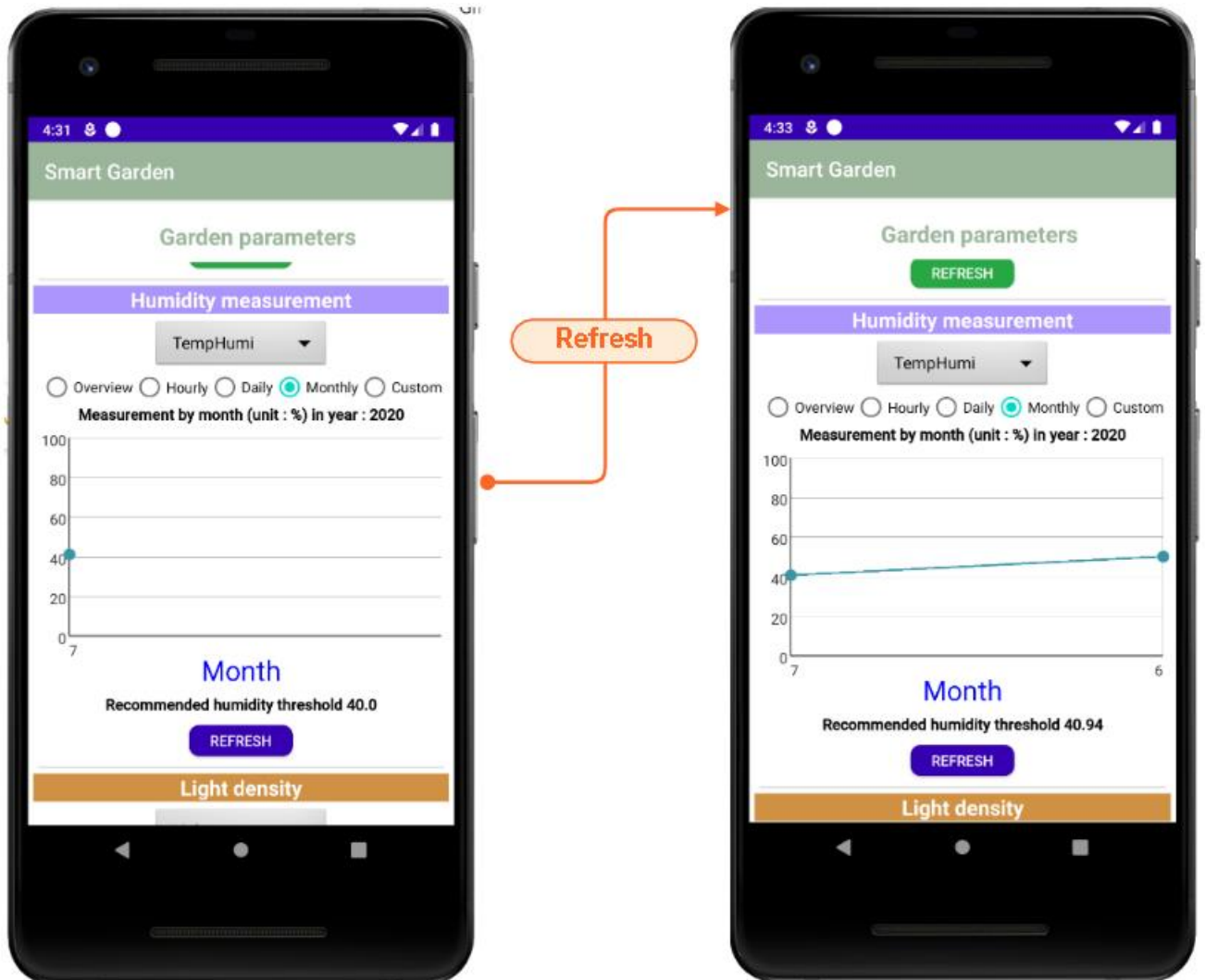
- For AI function:

AI result for seasonal data : The result is the peak of periodic data



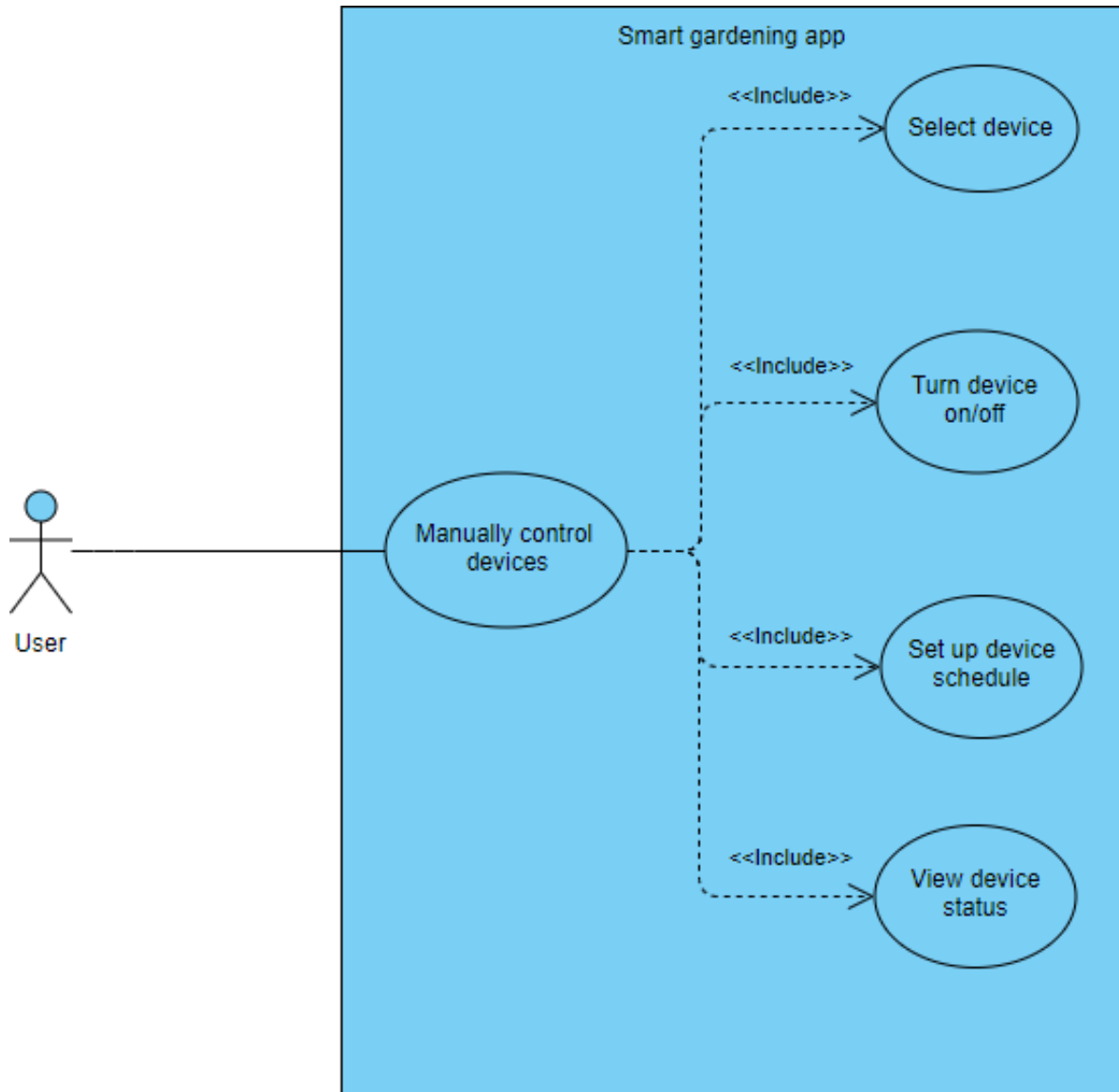
- For refresh function:

The view data and AI result update without requiring reload the application



• Vo Dong Ho :

1. Use case diagrams :



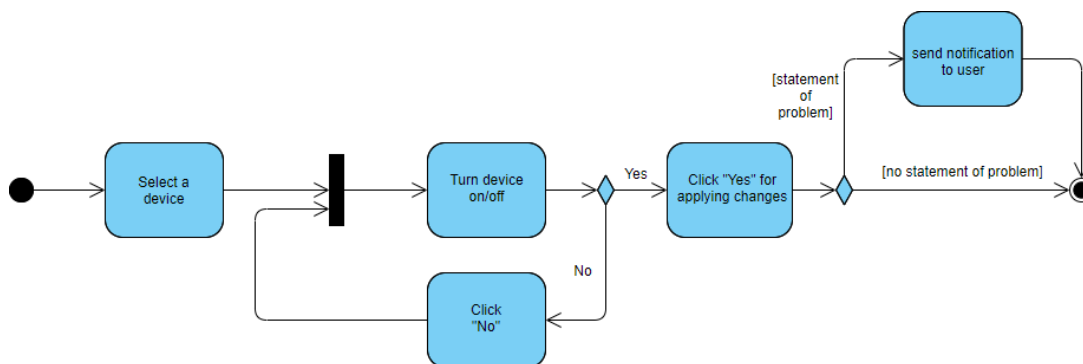
2. User-story

- As I finish my gardening recently, I decide to turn off all of the devices. But I want to have a button in an app that allow me to conveniently turn all the devices off.

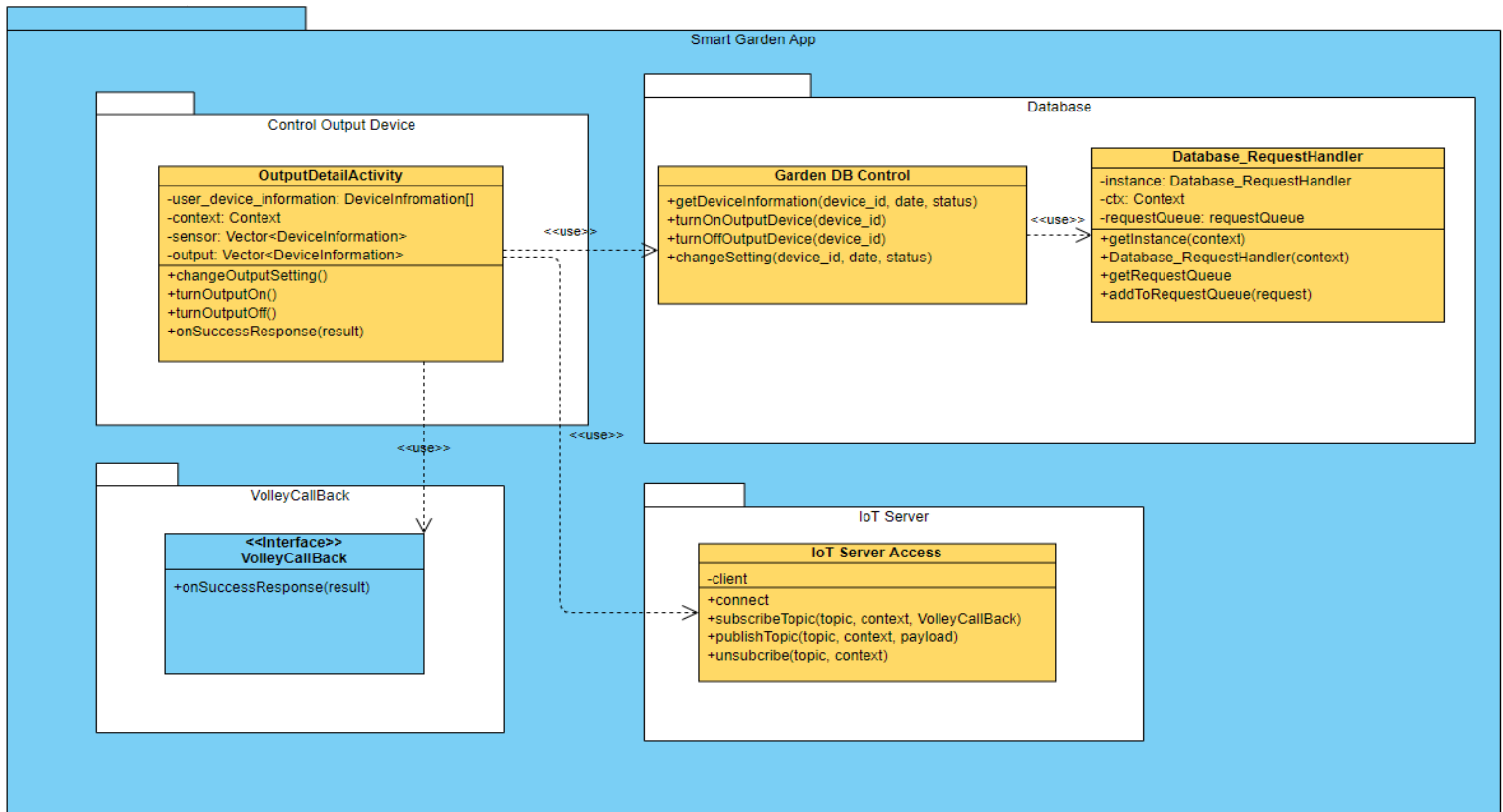
3. Use case scenarios :

Use-case ID	2
Use-case name	Manually turn on/off device
Actor	User
Description	User turns on/off any device in the app.
Preconditions	The mobile app is installed. At the present, user is in the devices tab.
Normal Flow	<ol style="list-style-type: none"> 1. User clicks on a specific device. 2. System shows that device tab, containing list of options. 3. User chooses option "Turn on/off". 4. System shows a message for confirmation. 5. User clicks on option "Yes". 6. System turns the device off if it's currently on and turn it on if it's currently off.
Exceptions	<p>Exception 1 in step 5:</p> <ol style="list-style-type: none"> a. User clicks on option "No". b. System returns to that device tab. <p>Exception 2 in step 6:</p> <ol style="list-style-type: none"> a. Cannot turn device on because it cannot work due to technical or connection error. b. System sends a message for device error.
Alternative Flows	<p>Alternative 1 in step 3:</p> <ol style="list-style-type: none"> a. User chooses another option in that device tab.

4. Flow chart



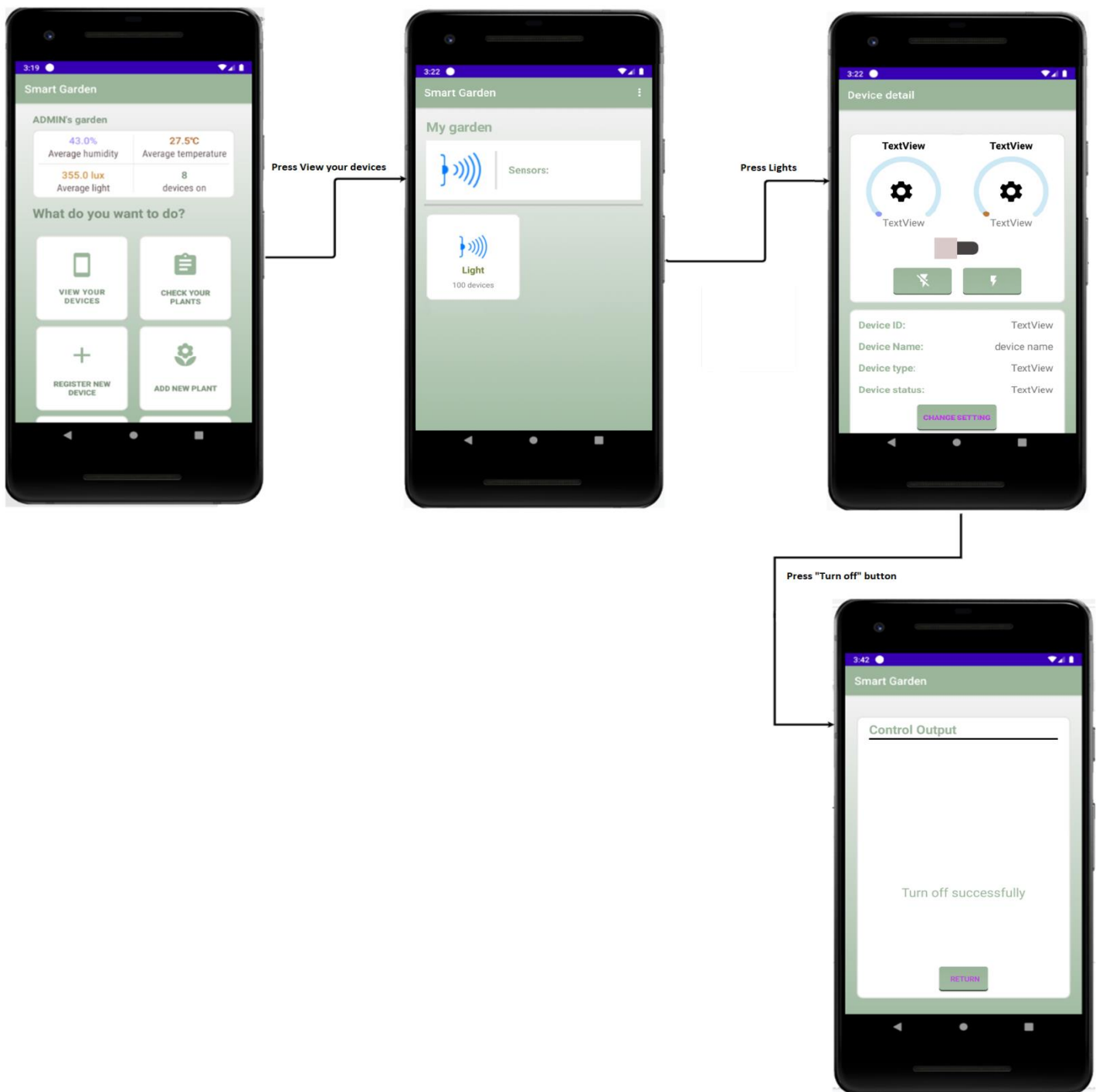
5. Class diagram



- **Class Garden DB control:** contains functions that related to DB server
 - + **getDeviceInformation(device_id, date, status):** request database to get information about a specific device.
 - + **turnOnOutputDevice(device_id):** sends request to database to turn on output devices.
 - + **turnOffOutputDevice(device_id):** sends request to database to turn off output devices.
 - + **changeSetting(device_id, date, status):** request database to change device setting.
- **Class Database_RequestHandler:** a class that handle sending request to the database server.
 - + **getInstance(context):** get the static Database_RequestHandler instance for the context.
 - + **getRequestQueue():** if requestQueue is null, then initial it, else return it.
 - + **addToRequestQueue(req):** add req onto the req queue for handling request.
- **Class IoT Server Access:** contains function that allow user to work with topic in server:
 - + **Connect(context):** connect to the MQTT server.
 - + **Subscribe(topic, context):** subscribe to a topic on MQTT server. When message arrive, each class using this method will have different implementations.
 - + **Publish(topic, payload, context):** publish payload to the topic.
 - + **Unsubscribe(topic, context):** unsubscribe from a topic.

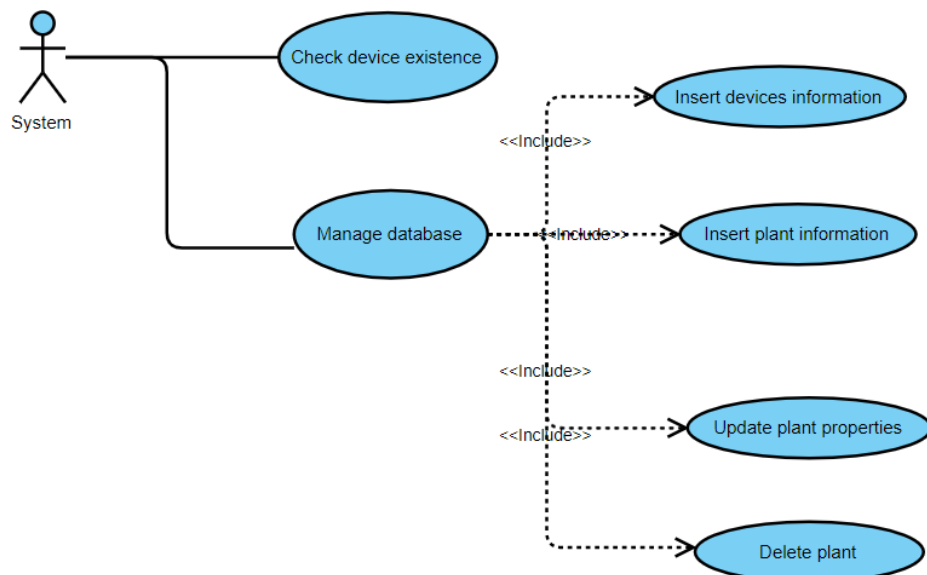
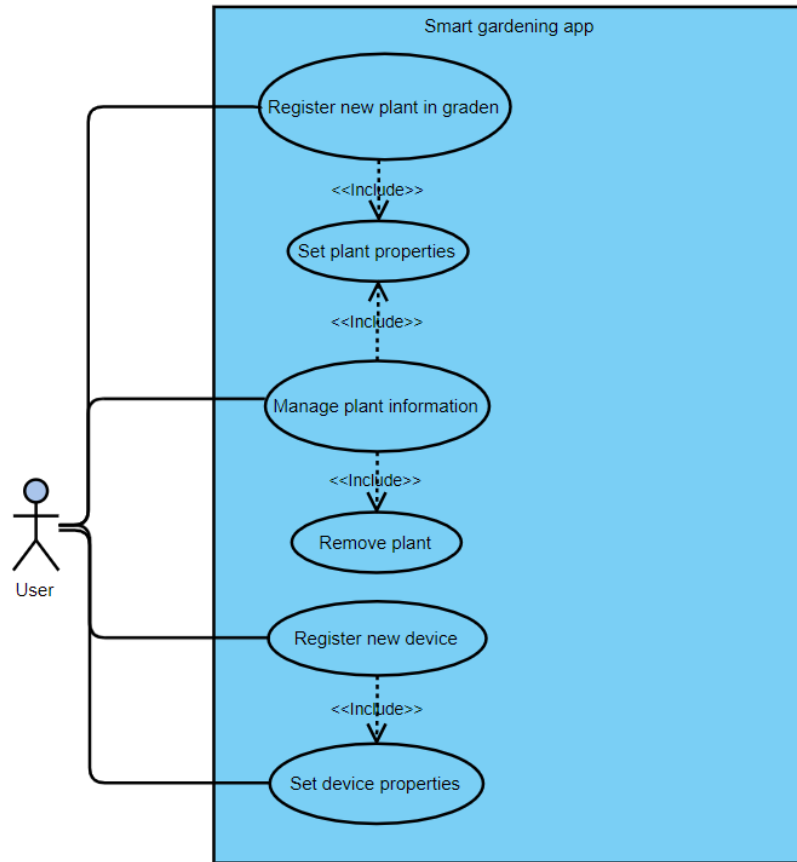
- **Interface VolleyCallBack:** contain function when a response from database action arrive successfully.
- **Class OutputDetailActivity:** views output devices's status and contains functions like turn on/off or changing setting, ...:
 - + **changeOutputSetting():** change output attributes then sends to database.
 - + **turnOutputOn():** checks output status and turns it on if it currently off, then subscribe status to MQTT server.
 - + **turnOutputOff():** checks output status and turns it off if it currently on, then subscribe status to MQTT server.
 - + **onSuccessResponse(result):** inheritance method. For this class the method will get all the device in the database to update the system management if device is updated else it will do nothing .

6. Mock up



• Ngo Nguyen Duy An :

1. Use case diagrams :



2. User story:

Function	User's story	Accept criteria
Register device	As a garden owner, I want to use my newly bought pair of devices through the app to check condition in my garden	<ul style="list-style-type: none">+ See device type after search correct sensor+ Set own settings for sensor and output+ Able to see new devices in device list+ Able to see device information in device detail+ Able to turn on/off new output device+ Able to control device manually depend on user settings
Register plant	I want to register a new plant that I just bought and see the information of that plant	<ul style="list-style-type: none">+ Set properties for plant+ Able to see and view plant detail
Change plant properties	I realize that some information of the plant that I set is wrong and I want to change it	<ul style="list-style-type: none">+ Able to see new setting of the plant
Remove plant	A plant is no longer in my garden and I do not want to see it in the app because of confusion	<ul style="list-style-type: none">+ Able to delete the plant+ Not able to see the plant in list and plant detail

3. Use case scenario

Use case ID	3.
Use case name	Register new device
Actor	User
Description	Allow user to register a new pair of devices on the system and see their status
Trigger	When user have a new pair of devices that they want to register to the system
Preconditions	User are currently on home tab
Normal flow	<ol style="list-style-type: none"> 1. User press on "Register new device" button in menu 2. User specifies sensor id and sensor name and press "Search" button. 3. System check if device is registered or not and exist on IOT server or not 4. System generate setting form for user 5. User specify own threshold(s), linked output id, linked output name 6. The system check if the output is register or not. 7. System update new device pair in database 8. System show register result to user
Exceptions	<p>Exception at step 3:</p> <ol style="list-style-type: none"> 3.1. If user input wrong format 3.2. The app will show warning wrong format <p>Exception at step 3:</p> <ol style="list-style-type: none"> 3.3. If sensor with same ID already exist on system 3.4. The system will display message "Device is already registered" <p>Exception at step 3:</p> <ol style="list-style-type: none"> 3.5. If sensor does not exist on the server 3.6. System will display message "No device with id found" <p>Exception at step 6:</p> <ol style="list-style-type: none"> 6.1. If output device already exist on system 6.2. System will display message "Device is already registered" <p>Exception at step 7:</p> <ol style="list-style-type: none"> 7.1. If failed to update database 7.2. The registration message will show error
Alternative flow	<p>Alternative flow at 5:</p> <ol style="list-style-type: none"> 5.3. If user click on "Use default" 5.4. The app will automatically set up default value(s)

Use case ID	4.1
Use case name	Add new plant
Actor	User
Description	Allow user to register new plant on the system and see their status
Trigger	When user have new plant that they want to register to the system
Preconditions	User are currently on home tab
Normal flow	<ol style="list-style-type: none"> 1. User press on “Add new plant” button in menu 2. User specifies plant name, plant date, buy location, amount and press “Search” button. 3. System check if plant with the similar information is registered or not 4. System update new plant data on database
Exceptions	<p>Exception at step 3:</p> <ol style="list-style-type: none"> 3.1. If user input wrong format or leave field empty 3.2. The app will show warning wrong format <p>Exception at 4:</p> <ol style="list-style-type: none"> 4.1. If failed to update database 4.2. The registration message will show error
Alternative flow	<p>Alternative at step 4:</p> <ol style="list-style-type: none"> 4.1. If user choose none on linked device id 4.2. The registration message will show successfully but can not view device reading

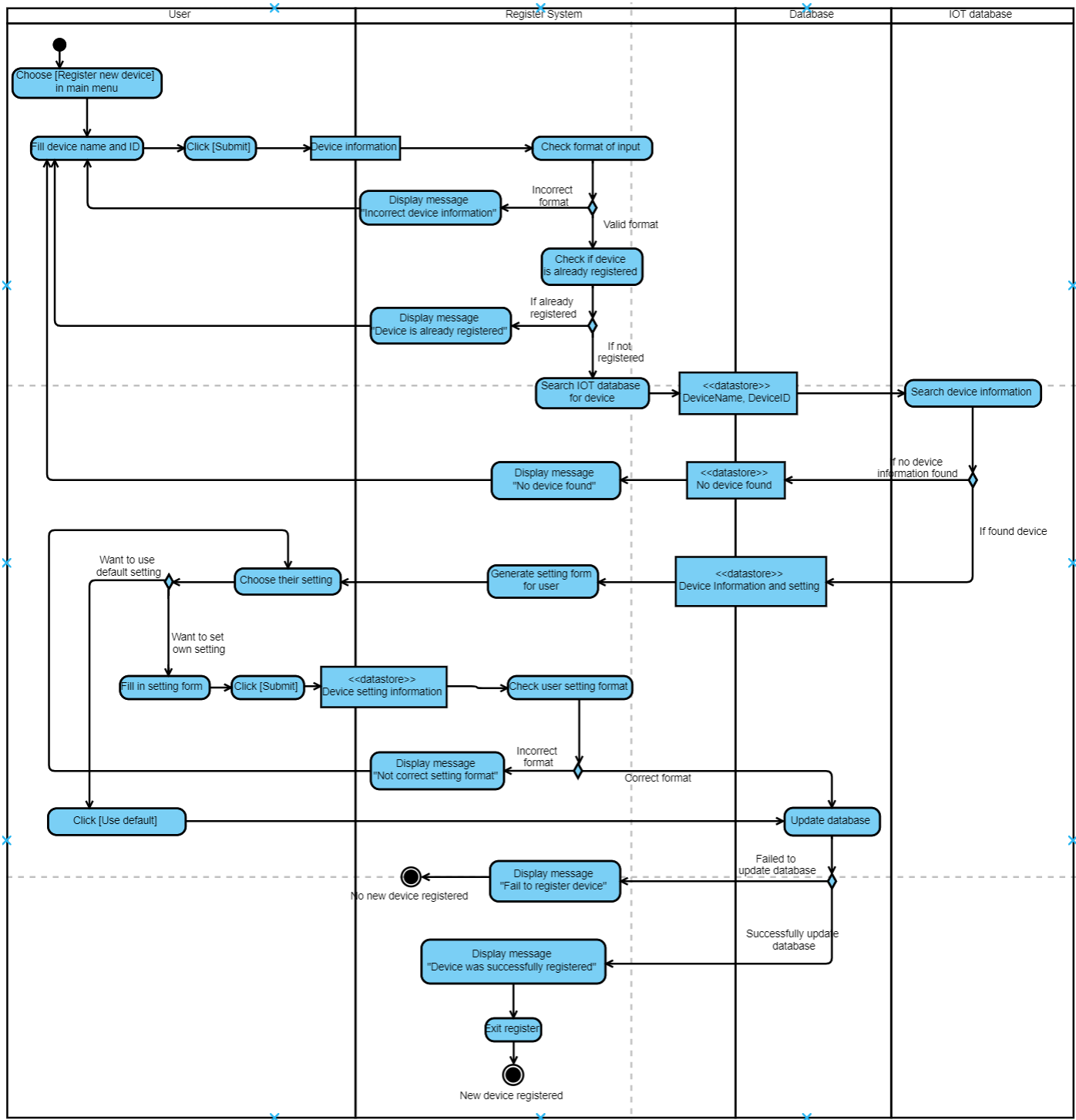
Use case ID	4.2
Use case name	Change plant properties
Actor	User
Description	Allow user to change plant properties on the system
Trigger	When user want to change a plant’s properties that they have registered on the system
Preconditions	User are currently on home tab
Normal flow	<ol style="list-style-type: none"> 1. User press on “View plant” button in menu 2. User choose the plant that user want to change properties 3. In plant detail, user press on “Change properties” button 4. User specify new plant properties and information. 5. User click Submit 6. System check if the new information is correct or not. 7. System update new plant data on database
Exceptions	<p>Exception at step 4:</p> <ol style="list-style-type: none"> 4.1. If user leave field empty 4.2. The app will show warning “Required field missing”

	<p>Exception at step 4</p> <p>4.3 If user input is not correct format</p> <p>4.4 System will send warning to alert user.</p> <p>Exception at step 4:</p> <p>4.5. If failed to update database</p> <p>4.6. The registration message will show error</p>
Alternative flow	<p>Alternative at step 4:</p> <p>4.1. If user press "Return button"</p> <p>4.2. The change process will end and return back to view detail activity.</p>

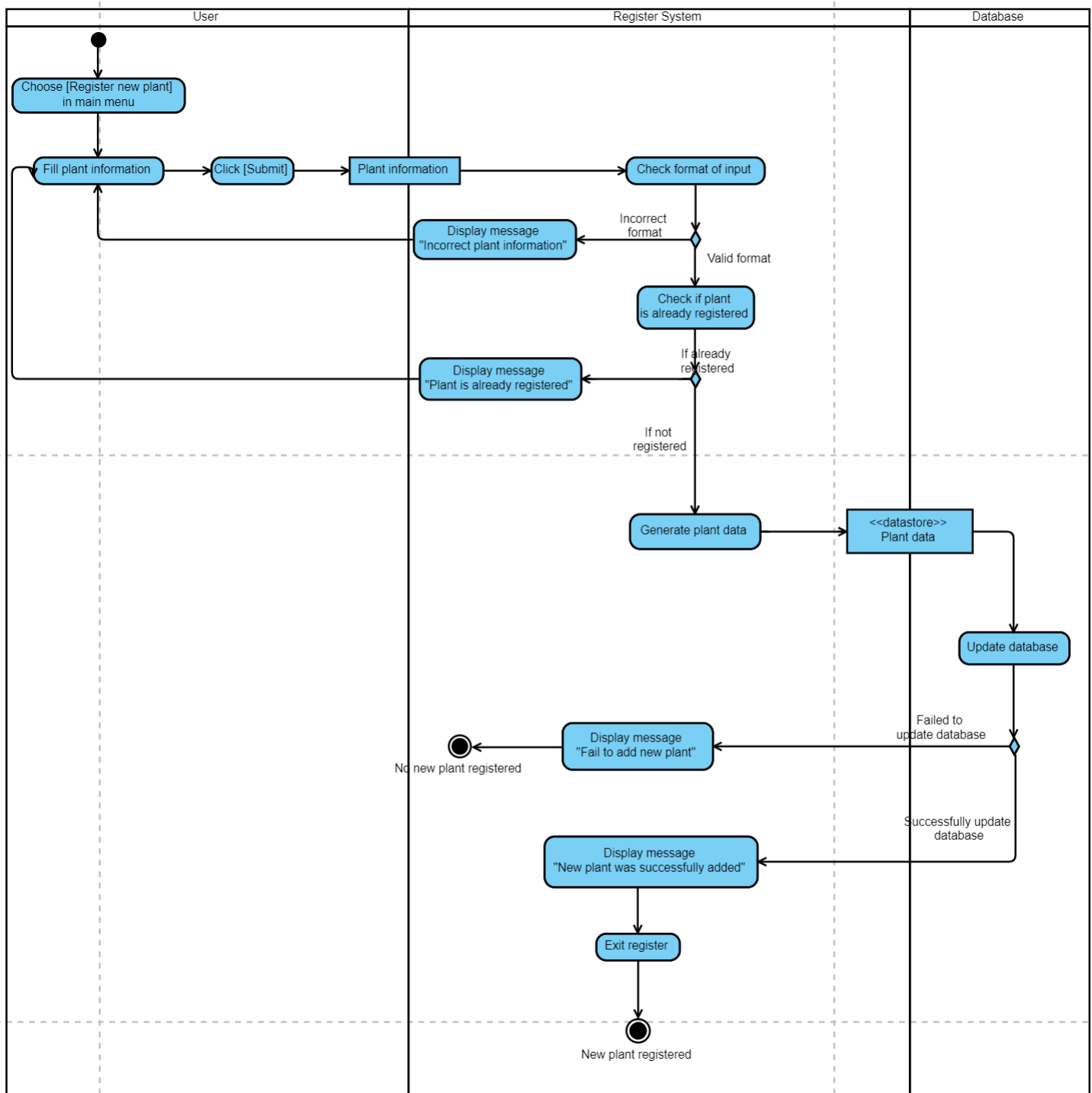
Use case ID	4.3
Use case name	Remove registered plant
Actor	User
Description	Allow user remove a registered plant on the system
Trigger	When user have new plant that they want to register to the system
Preconditions	User are currently on home tab
Normal flow	<ol style="list-style-type: none"> 1. User press on "View plant" button in menu 2. User choose the plant that user want to change properties 3. In plant detail, user press on "Remove plant" button 4. System ask if user want to continue 5. User choose "Yes" 6. System ask for account password 7. User specified own password and click "Ok" 8. System delete plant data on the system
Exceptions	<p>Exception at 8:</p> <p>8.1. If failed to update database</p> <p>8.2. The message will show "Something went wrong"</p>
Alternative flow	<p>Alternative at step 4:</p> <p>4.1. If user choose "No"</p> <p>4.2. The app will return to plant detail tab</p> <p>Alternative at step 6:</p> <p>6.1. If user choose "Cancel"</p> <p>6.2. The app will return to plant detail tab</p> <p>Alternative at step 7:</p> <p>7.1. If user insert wrong password</p> <p>7.2. The app will return to plant detail tab</p>

4. Activity diagrams

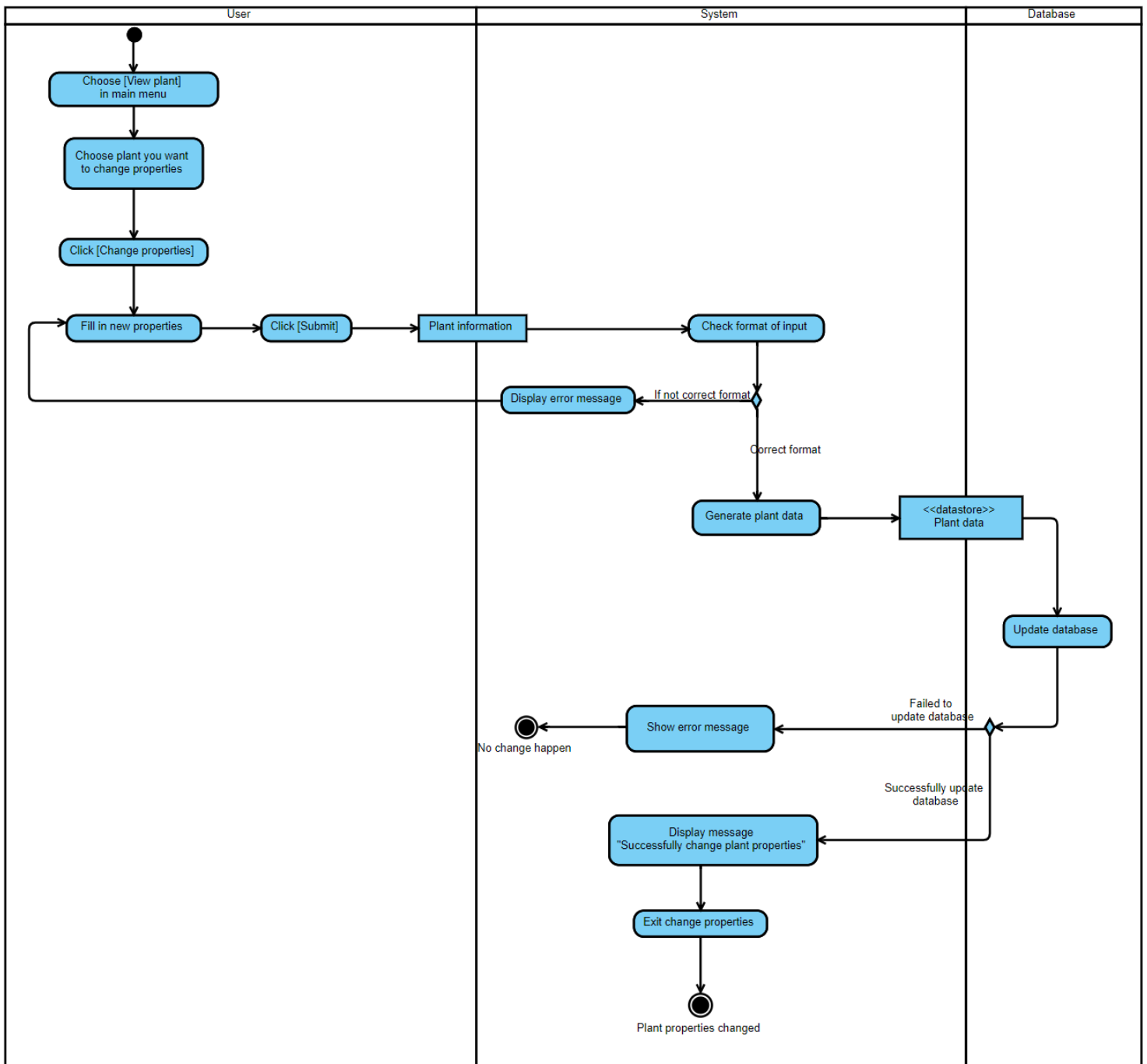
- For register new device:



- For register new plant:

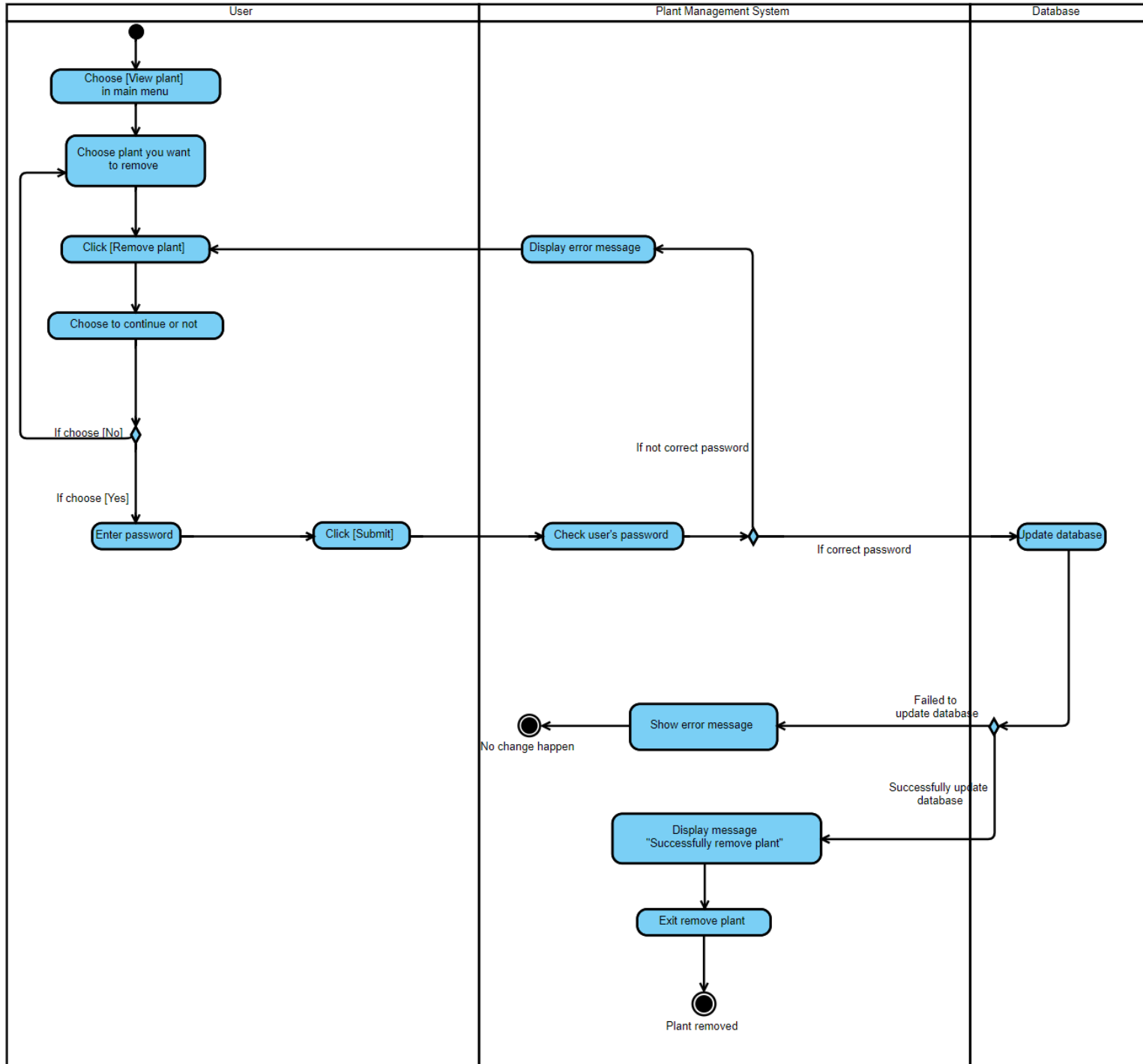


- For change plant properties:



Activity diagram for change plant properties

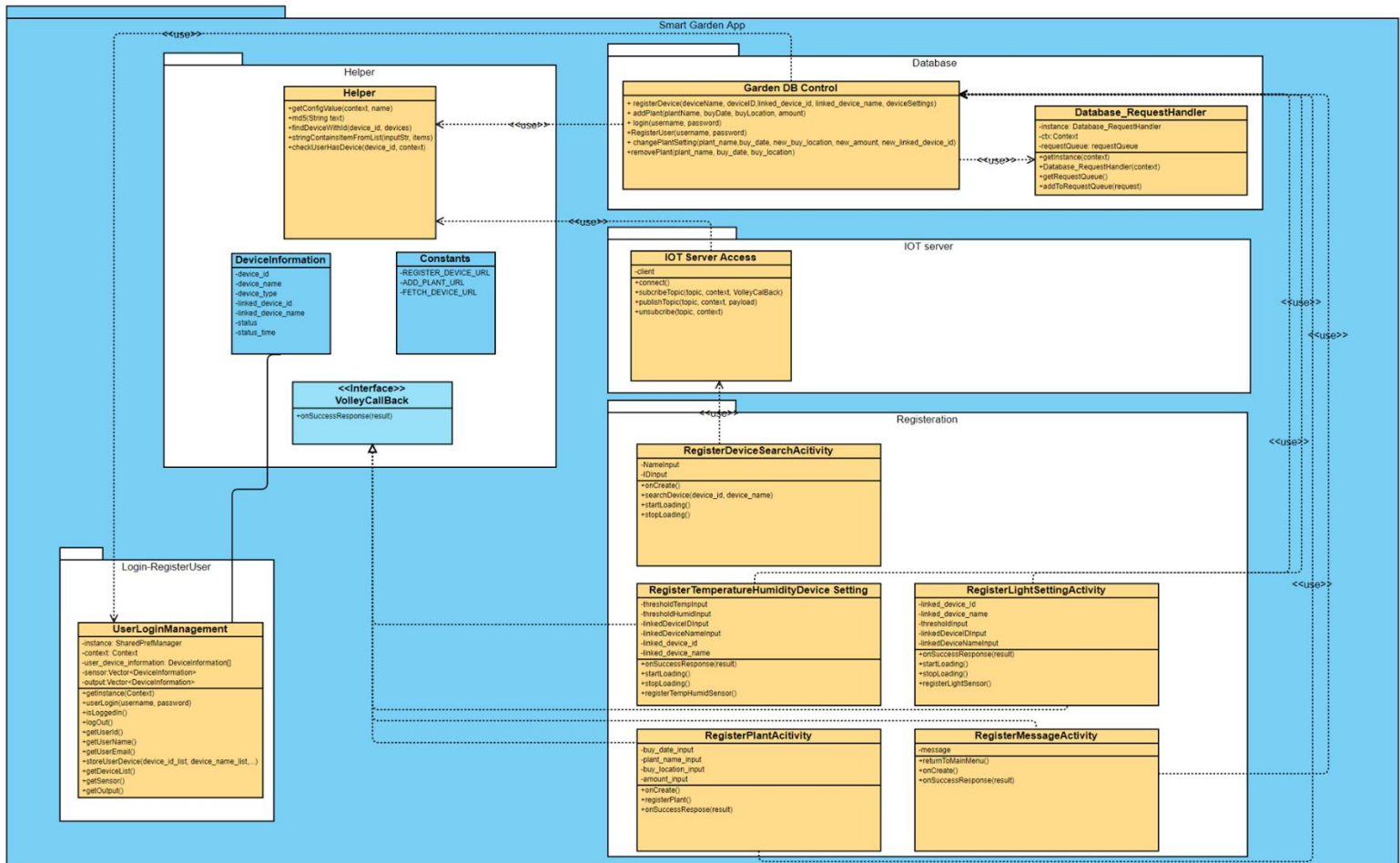
- For remove plant:



Activity diagram for remove plant

5. Class diagram

- For register device and register plant

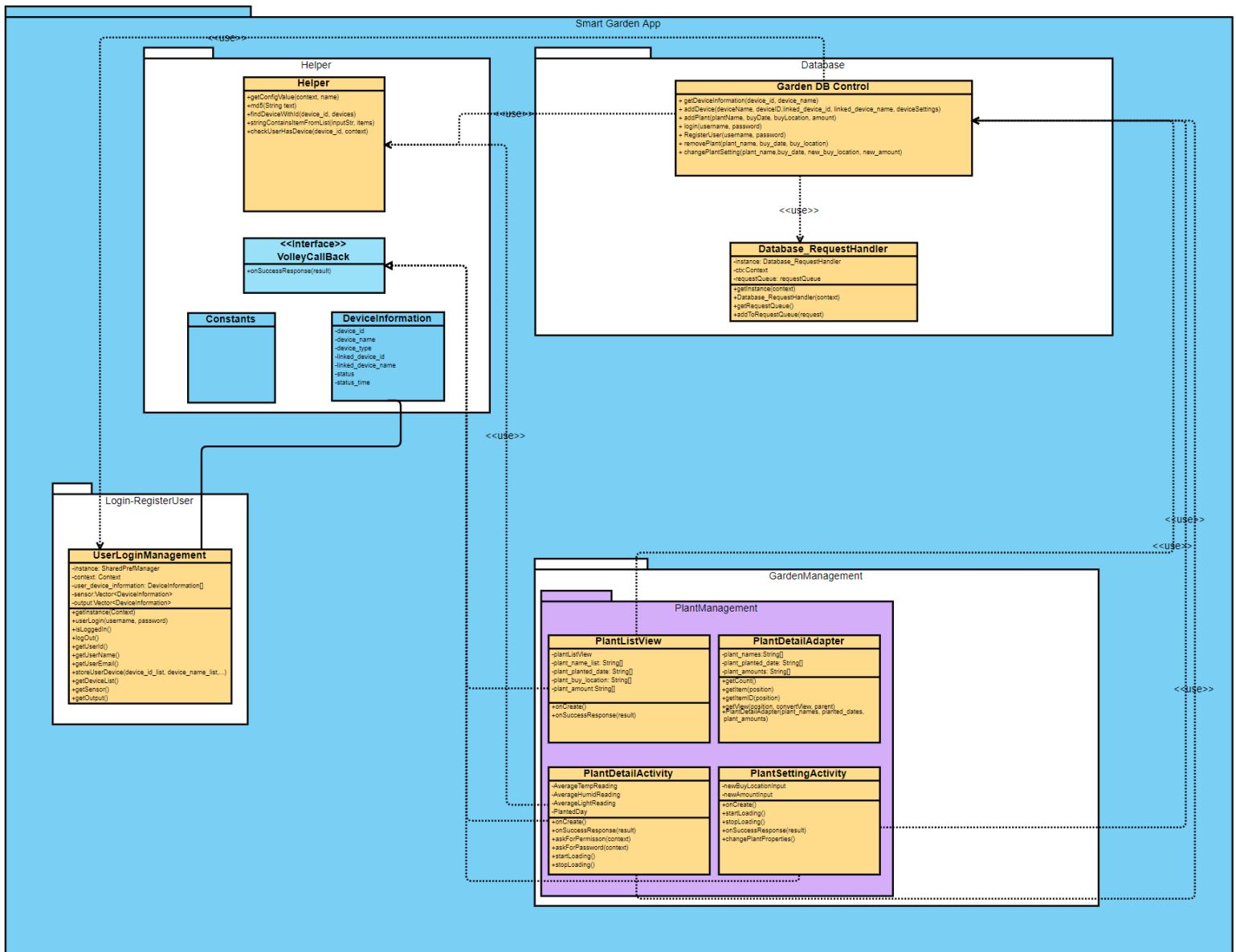


- **Class Garden DB control:** contains functions that related to DB server like insert device, register user,...:
 - **RegisterDevice(device_id, device_name, linked_device_id, linked_device_name, device_settings):** send request to database server to add new device with provided information
 - **addNewPlant(plant_name, buy_date, buy_location, amount):** send request to database to add new plant.
 - **changePlantSetting(plant_name, buy_date, new_buy_location, new_amount, new_linked_device_id):** change plant properties with given input
 - **removePlant(plant_name, buy_date, buy_location):** remove user plant based on plant_name, buy_date and buy location
- **Class Database_RequestHandler:** a class that handle sending request to the database server
 - **getInstance(context):** get the static Database_RequestHandler instance for the context
 - **getRequestQueue():** if requestQueue is null, then initial it, else return it
 - **addToRequestQueue(req):** add req onto the req queue for handling request
- **Class IoT Server Access:** contains function that allow user to connect to IoT server, subscribe, publish,... for a certain topic:
 - **Connect(context):** connect to the MQTT server

- **Subscribe(topic, context):** subscribe to a topic on MQTT server. When message arrive, each class using this method will have different implementations.
- **Publish(topic, payload, context):** publish payload to the topic.
- **Unsubscribe(topic, context):** unsubscribe from a topic
- **Class UserLoginManagement:** control user login profile
 - **getInstance(context):** Return the static instance of UserLoginManagement for the context to use
 - **userLogin(user_id, user_name):** put the user_id and user_name onto the SharedPreferences editor.
 - **isLoggedIn():** Check if a user is logged in or not
 - **logout():** clear stored parameter of the SharedPreferences editor when login
 - **storeUserDevice(..):** store user's device information.
- **Interface VolleyCallBack:** contain function when a response from database action arrive successfully
- **Class Helper:** get properties value from the configuration file
 - **getConfigValue(Context context, String name):** get the value of name field in the config file.
 - **checkUserHasDevice(String device_id, Context context):** check if user has already registered device or not.
 - **Md5(text):** return md5 hasing value of text, used to compare password
 - **findDeviceWithId(device_id, devices):** return the DeviceInformation of device with device id
 - **stringContainsItems(inoutStr, items):** check if input string contains any item in the items array
- **Class RegisterDeviceSearchActivity:** control UI for activity register device
 - **onCreate():** default function for activity
 - **startLoading():** display loading screen and disable buttons
 - **stopLoading():** stop display loading screen
 - **searchDevice(device_id, device_name):** search for sensor on MQTT server by sybcribe and wait for response
- **Class DeviceLightSettingActivity:** control UI for setting light sensor device properties when register new device
 - **onCreate():** default function for activity
 - **startLoading():** display loading screen and disable buttons
 - **stopLoading():** stop display loading screen
 - **registerLightSensor():** take input from input field, check and register new light sensor if input is valid
- **Class DeviceTemperatureHumiditySettingActivity:** control UI for setting light sensor device properties when register new device
 - **onCreate():** default function for activity and set functionality for buttons inside activity
 - **startLoading():** display loading screen and disable buttons
 - **stopLoading():** stop display loading screen
 - **registerTempHumidSensor():** take input from input field, check and register new temperature humidity sensor if input is valid
- **Class RegisterPlantActivity:** control UI for activity register plant
 - **onCreate():** default function for activity and set functionality of submit button, date picker and device spinner
 - **onSuccessResponse(result):** inheritance method. For this class the method will get the message from the result and check that if successfully register plant then go to message screen, else display fail messages
 - **registerPlant():** take user input, check for format and then register plant is no exception happen
- **Class RegisterMessageActivity:** control UI for display register activity

- **onCreate()**: default function for activity, initialize function of buttons
- **onSuccessResponse(result)**: inheritance method. For this class the method will get all the device in the database to update the system management if register device else it will do nothing
- **isMyServiceRunning(Class service)**: check if system service is running or not. In this case, used to check if record measurement service has activated or not

- For change plant properties and remove plant:

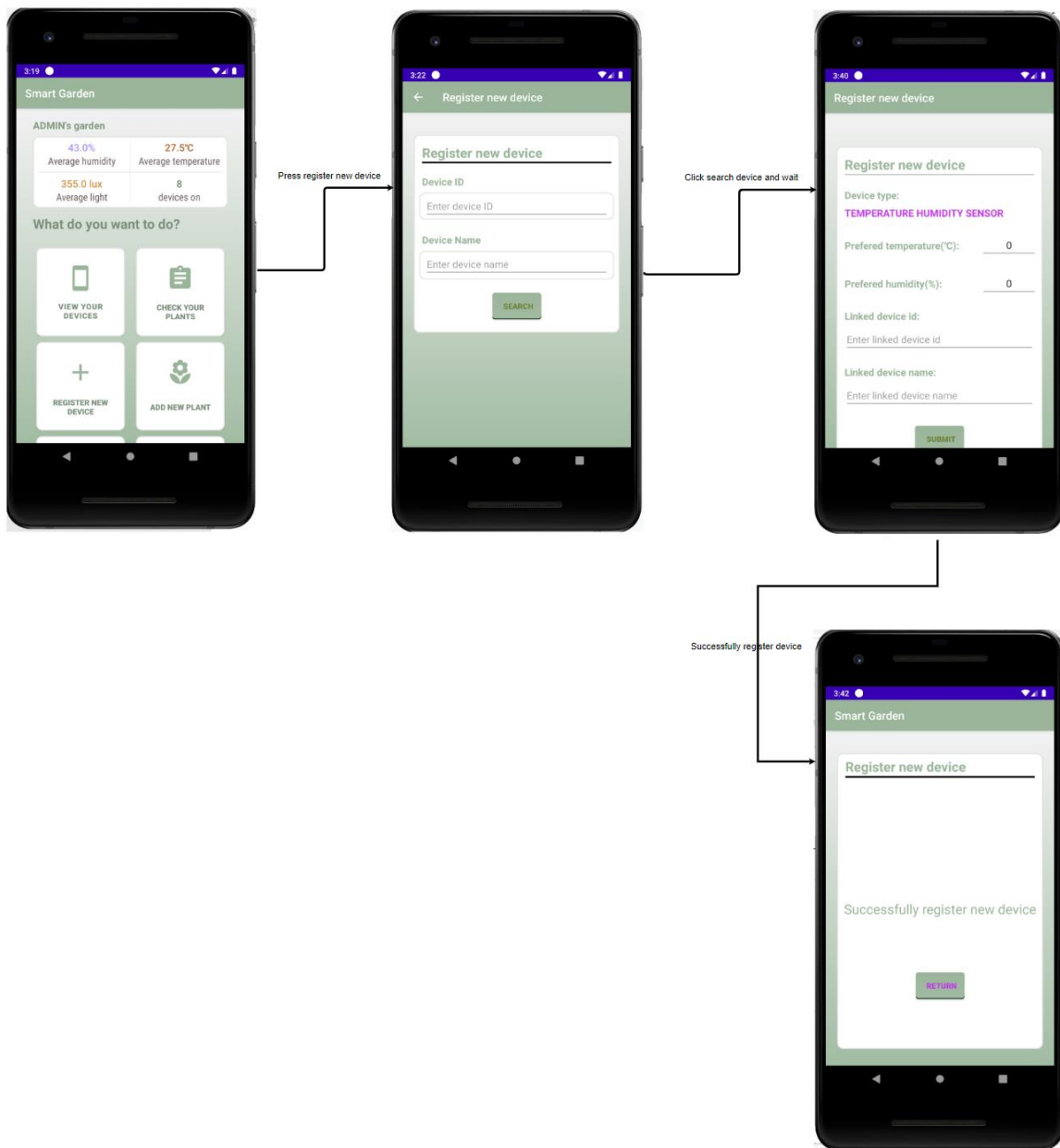


- **Class PlantListView**: control UI for activity register device
- **onCreate()**: default function for activity. In this class this function will fetch all plant information from database and display to user

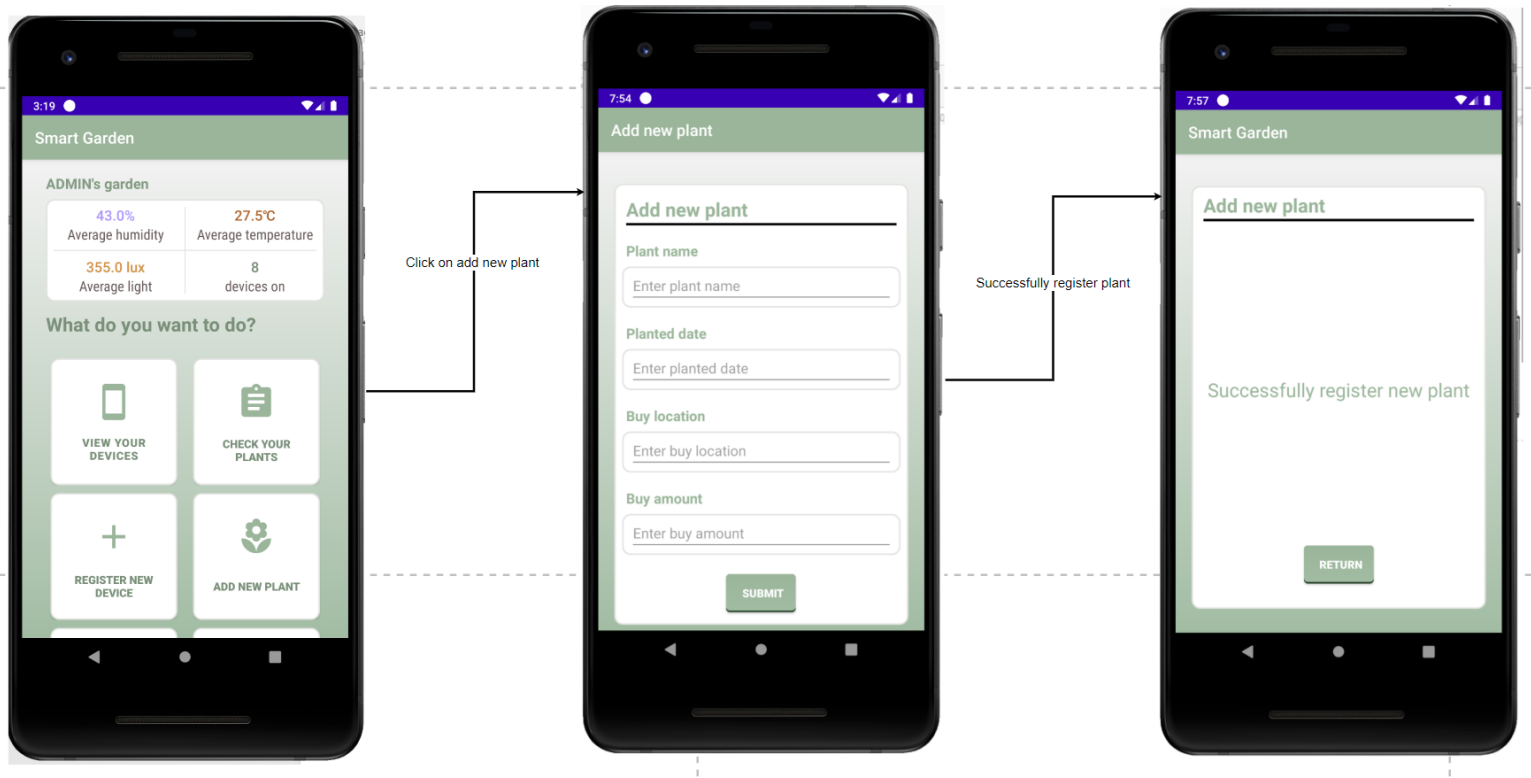
- **onSuccessResponse(result):** this function is the result of fetching all plant information, then we will display them after handle the data
- **Class PlantDetailAdapter:** get plants detail and generate view for each of them
 - **PlantDetailAdapter(plant_names, planted_dates, plant_amount):** constructor of the class, initialize the plant data to generate view
 - **getCount():** return number of item
 - **getItem(position):** return item at position
 - **getItemId(position):** return item id at position
 - **getView(position, convertView, parent):** return a view for the plant data at position
- **Class PlantDetailActivity:** control UI for showing plant detail and removing plant.
 - **onCreate():** default function for activity and set functionality for buttons inside activity.
 - **startLoading():** display loading scene and disable buttons.
 - **stopLoading():** stop display loading scene.
 - **askForPermission(context):** ask for user permission to remove plant.
 - **askForPassword(context):** ask for user password when removing plant.
 - **onSuccessResponse(result):** get all sensor devices last reading and handle data to display to user.
- **Class PlantSettingActivity:** control UI for activity change plant properties
 - **onCreate():** default function for activity and set functionality of submit button, return button.
 - **startLoading():** display loading scene and disable buttons.
 - **stopLoading():** stop display loading scene.
 - **onSuccessResponse(result):** inheritance method. For this class the method will get the message from the result of change plant properties, display it then return back to plant list view
 - **changePlantProperties():** get all user input, check format then change plant properties if correct format, else show warning

6. Mock up

- For register new device:



- For register new plant:



- For change plant properties:

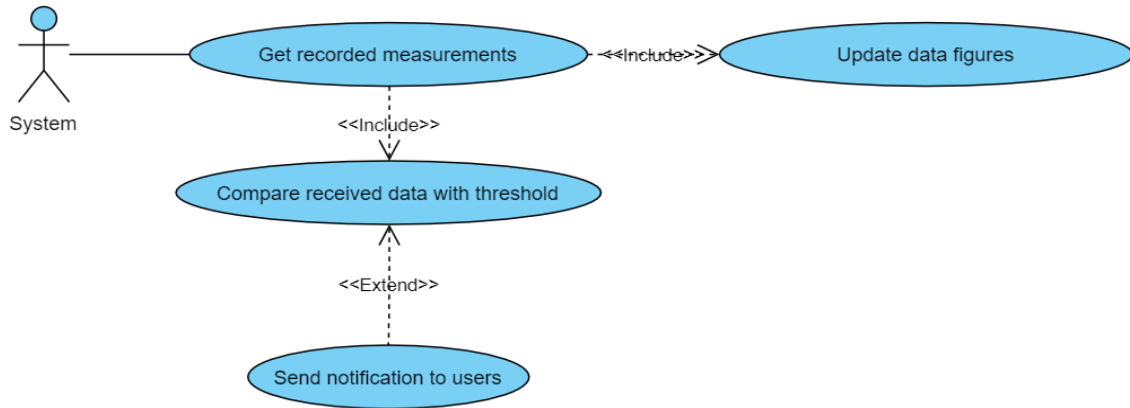


- For remove plant



• Pham Tuan Anh :

1. Use case diagrams :



2. User-story

As a user, I want to know the environment condition of my garden like temperature, air humidity, light intensity whether they are good or not.

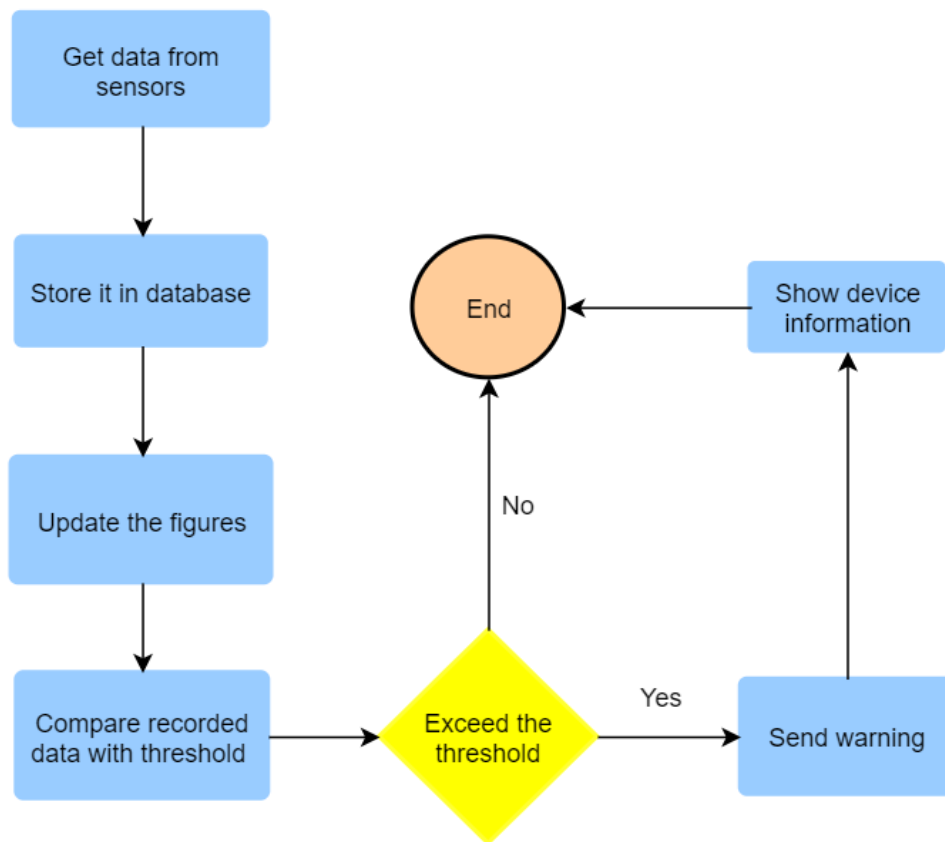
Therefore, the system get measurements from sensor devices in the garden and then it compares the received figures with the threshold (check if the figures are greater or less than the threshold number), they will send the notification to users if it exceeds the limits, it also updates the last reading data automatically.

On the other hand, if the actual data has no problems with the limits, the system will not send notification.

3. Use case scenarios

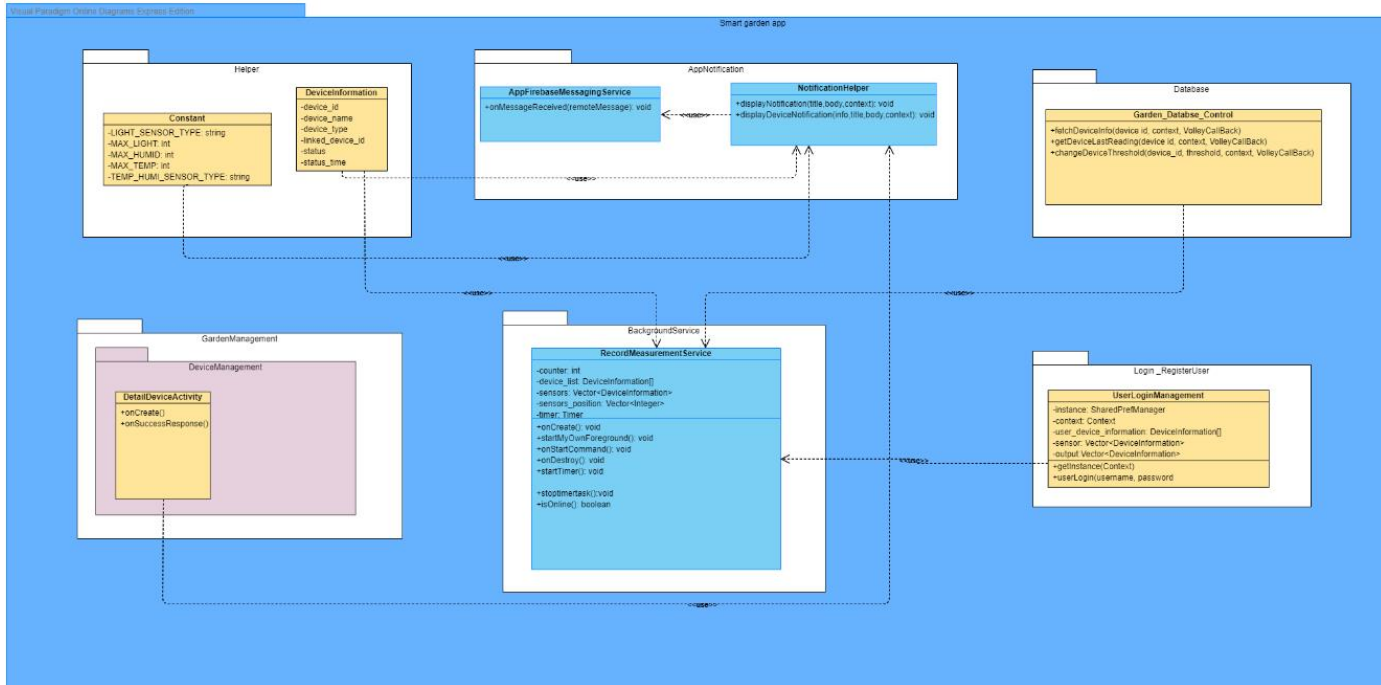
Use case ID	5
Use case name	Send notification to users
Actor	System
Description	Send notifications to user when temperature, humidity measurements and light intensity hit a certain point.
Precondition	When users are using the app
Normal flows	<ol style="list-style-type: none">1. System gets recorded measurements.2. System stores received data in database.3. System updates data automatically.4. System compares the recorded measurements with threshold.5. System shows information about sensors on UI of the app.6. System sends notification to users.7. Warning bar is appeared on the screen.
Alternative flows	Alternative flow at 6: 6.1 If the recorded measurement is exceeded the threshold. 6.2 The system will not send the notification
Exceptions	None

4. Flow chart



5. Class diagram.

For sending notification function



Class Garden_Database_Control: contains functions that related to DB server

+**FetchDeviceInfo()**: get information from database server.

+**getDeviceLastReading()**: get the latest measurement(s) from sensors based on device ID.

Class UserLoginManagement: control user login profile

+**getInstance(context)**: Return the static instance of UserLoginManagement for the context to use

+**userLogin(user_id, user_name)**: put the user_id and user_name onto the SharedPreferences editor.

Class DeviceDetailActivity: Display all information corresponding to the sensor that user selected in DeviceListView.

+ **onCreate()**: default function for activity, here used to fetch all information of the sensor that we collected.

+ **onSuccessResponse(result)**: inheritance method. For this class the method will get the message from the result of getting measurements of the sensor and display them to user.

Class Constants: contains constants for the program.

Class DeviceInformation: contains functions that return device's attributes (id, name, type, threshold, status, linked_device_name, linked_device_id).

Class NotificationHelper: The main function of this class is to display warnings and display device information.

+**displayNotification()**: send to the notification show on the screen.

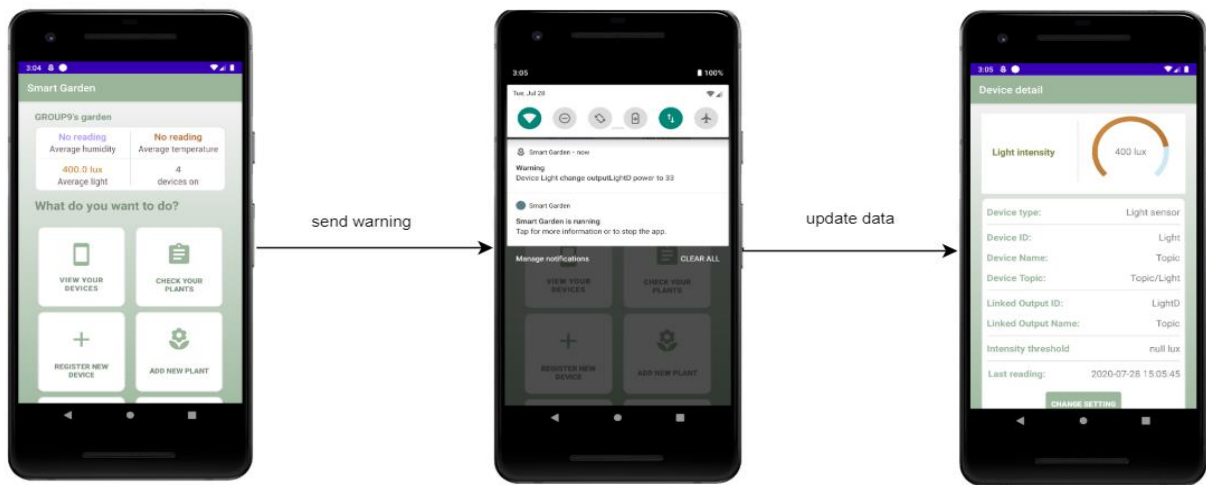
+**displayDeviceNotification()**: show the information of the device such as device_id, device_name, device_type...

Class AppFirebaseMessagingService: make the warnings show on the tab.

Class RecordMeasurementService: contain some functions run in the background of the app so as to record

the measurements continuously.

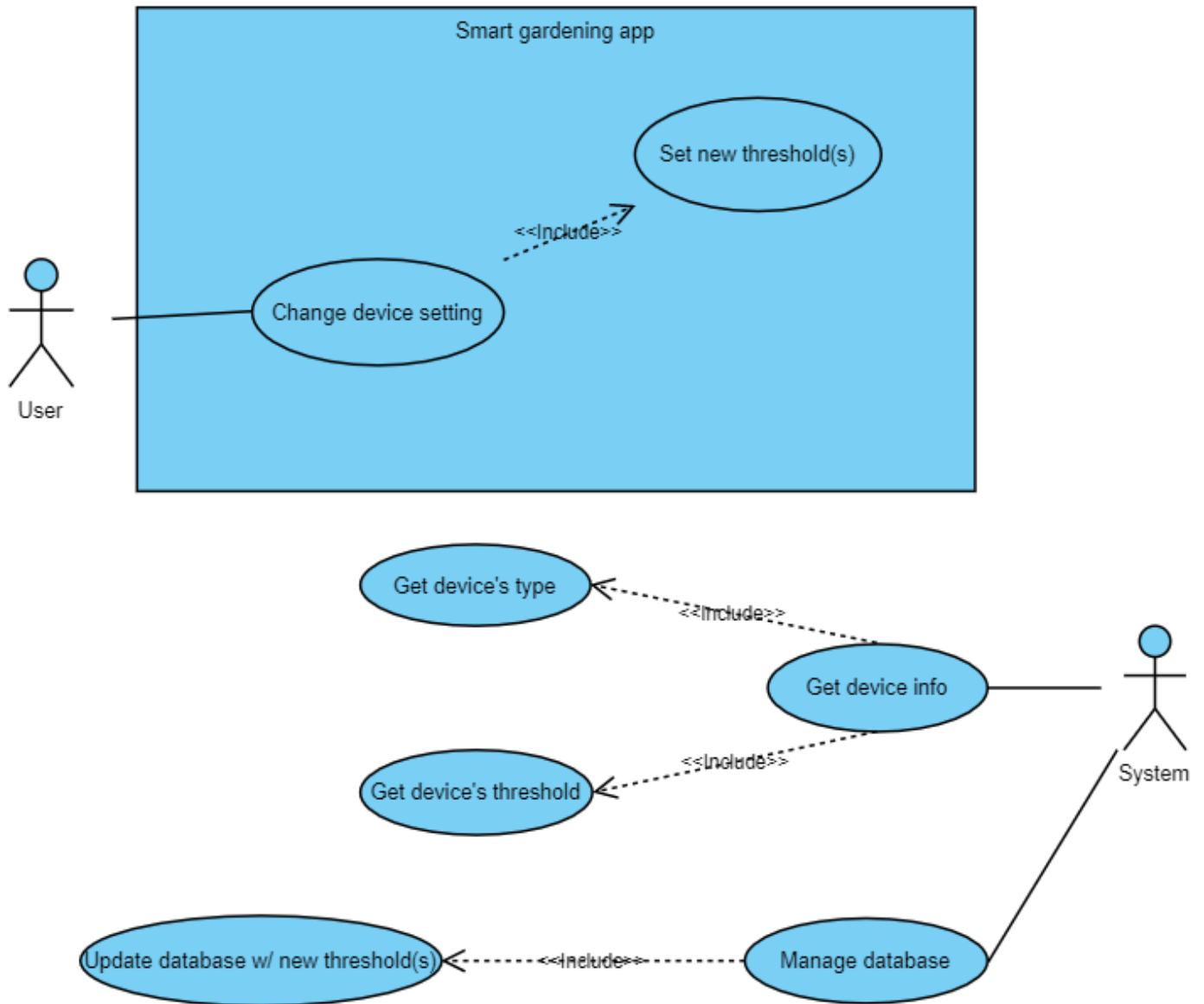
6. Mock-up



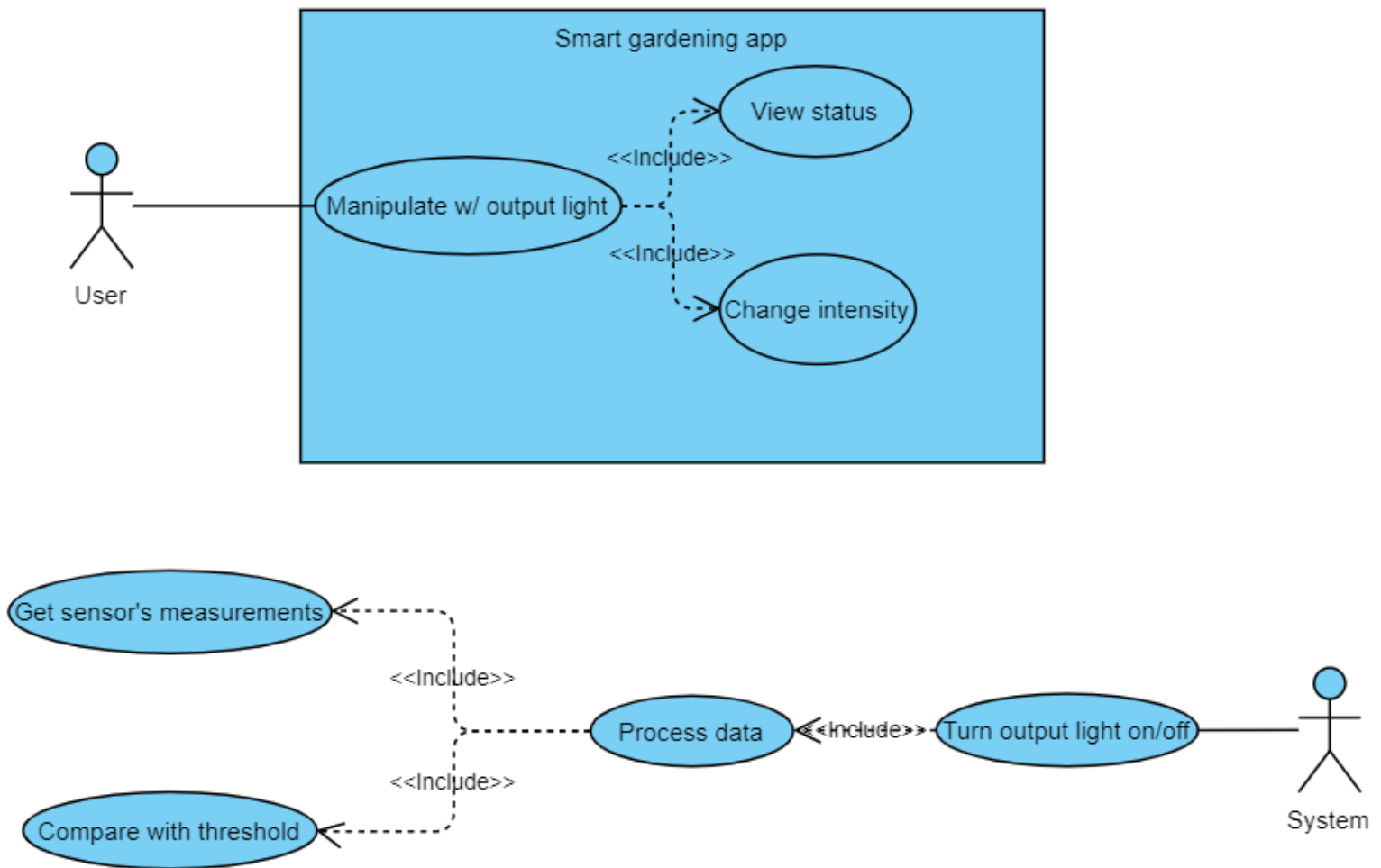
• Tran Minh Hung :

1. Use case diagrams

a. Change Device Setting



b. Automatically control device



2. Use case story

a. Change device setting

As a user, I want to review the thresholds that I set for my sensors and make changes if necessary.

b. Automatically control device

As a user, I want the signal light attached to my sensors to automatically turn on whenever the measurements exceed limits so that I can take necessary actions. And also, that signal light should be off if every returns normal.

3. Use case scenarios

a. Change device setting

Process description: The process Setting devices setting allows user to review the threshold of their sensors (light, hump/temp) and change it if they want to.

The process is triggered upon user choosing the option "Change setting" in device detail tab and is finished upon user clicking on "Submit".

Use case name	Change device setting
Actor	User
Description	User can set up new threshold for the sensor(s) that they registered
Precondition	User already logged in and register at least 1 sensor
Normal flow	1. User clicks on "Change setting" button in device detail tab 2. Device is a Temp/Humi sensor, user proceeds to fill in 2 boxes, "New preferred temperature" and "New preferred humidity (%)" 3. User clicks on "Submit" button 4. The new thresholds are acceptable, system print out successfully updated message.
Exceptions	Exception 1 in step 2 2.1. If user has not filled in any boxes, system will print "empty required filed" message Exception 2 in step 2 2.2. If user filled in the boxes with invalid values, system will print "Invalid threshold message" Exception 3 in step 2 2.3. If user filled in the boxes with values that are too high (over 500 lux, 50°C or 100%), system will print "Threshold is too high" message
Alternative flow	Alternative 1 in step 2 2.1 Devices is a Light sensor, user proceeds to fill in 1 box, "New preferred light intensity"

b. Automatically control device

Process description: The process Automatically control device allows the user to view the status of output light in the application and also change the intensity of output light.

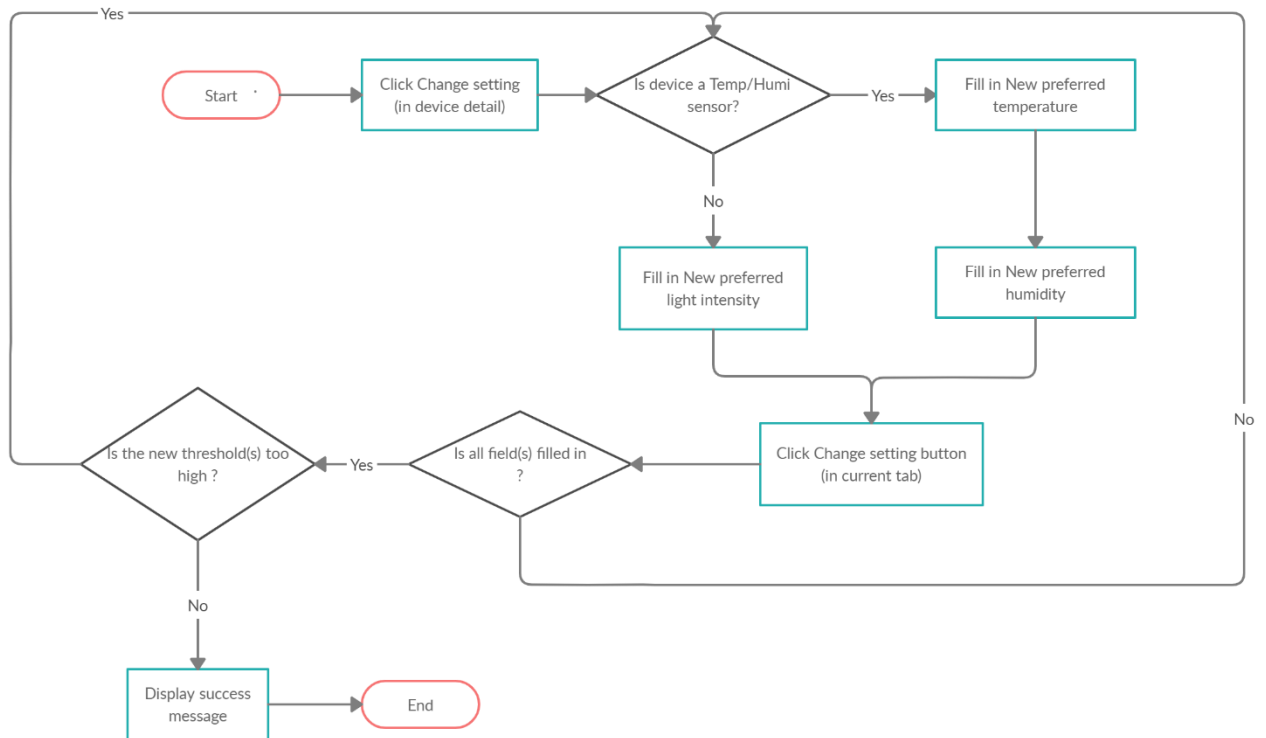
The precondition is user already register at least a pair of sensor (input) + output light (output)

The process is triggered upon user choosing the option "Change setting" in device detail tab and is finished upon user clicking on "Submit".

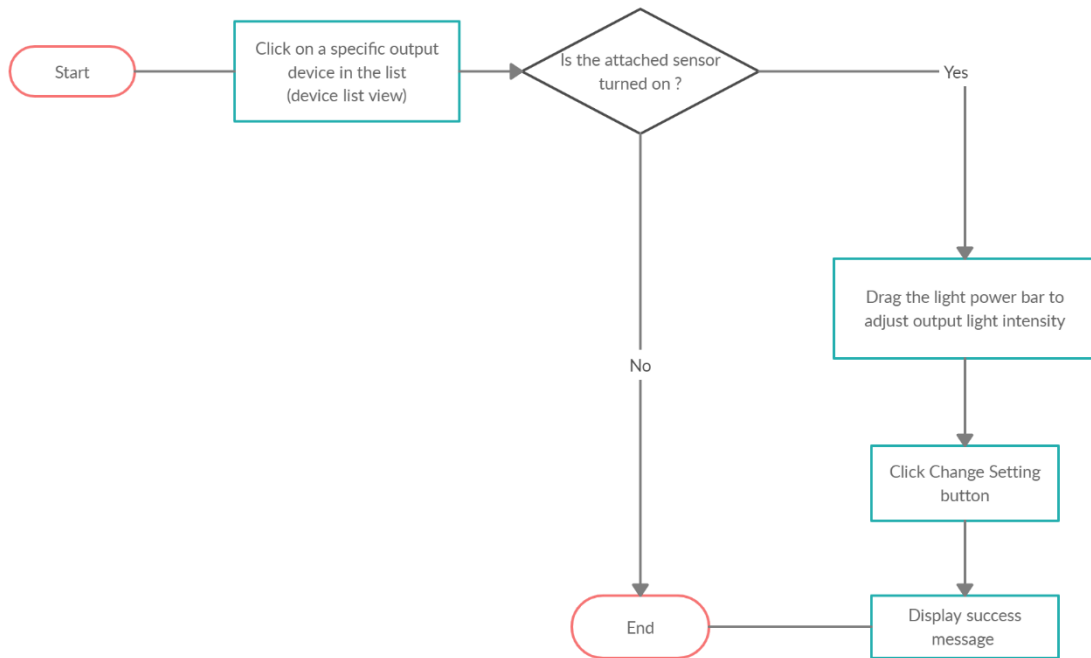
Use case ID	7
Use case name	Automatically control device
Actor	User
Description	System controls and turns on/off output automatically following the setting of thresholds and current measurements.
Precondition	User already set up at least one pair of input (sensor) + output (output light)
Normal flow	1. User clicks on a specific output device in device list view tab 2. The sensor attached with the output light is currently on, user can drag the light power bar to adjust the intensity of the output light. 3. User clicks on "Change setting" button 4. System displays setting saved message.
Exceptions	None
Alternative flow	Alternative 1 at step 2 2.1 The sensor attached with the output light is currently off, user cannot drag the light power bar.

4. Flow charts

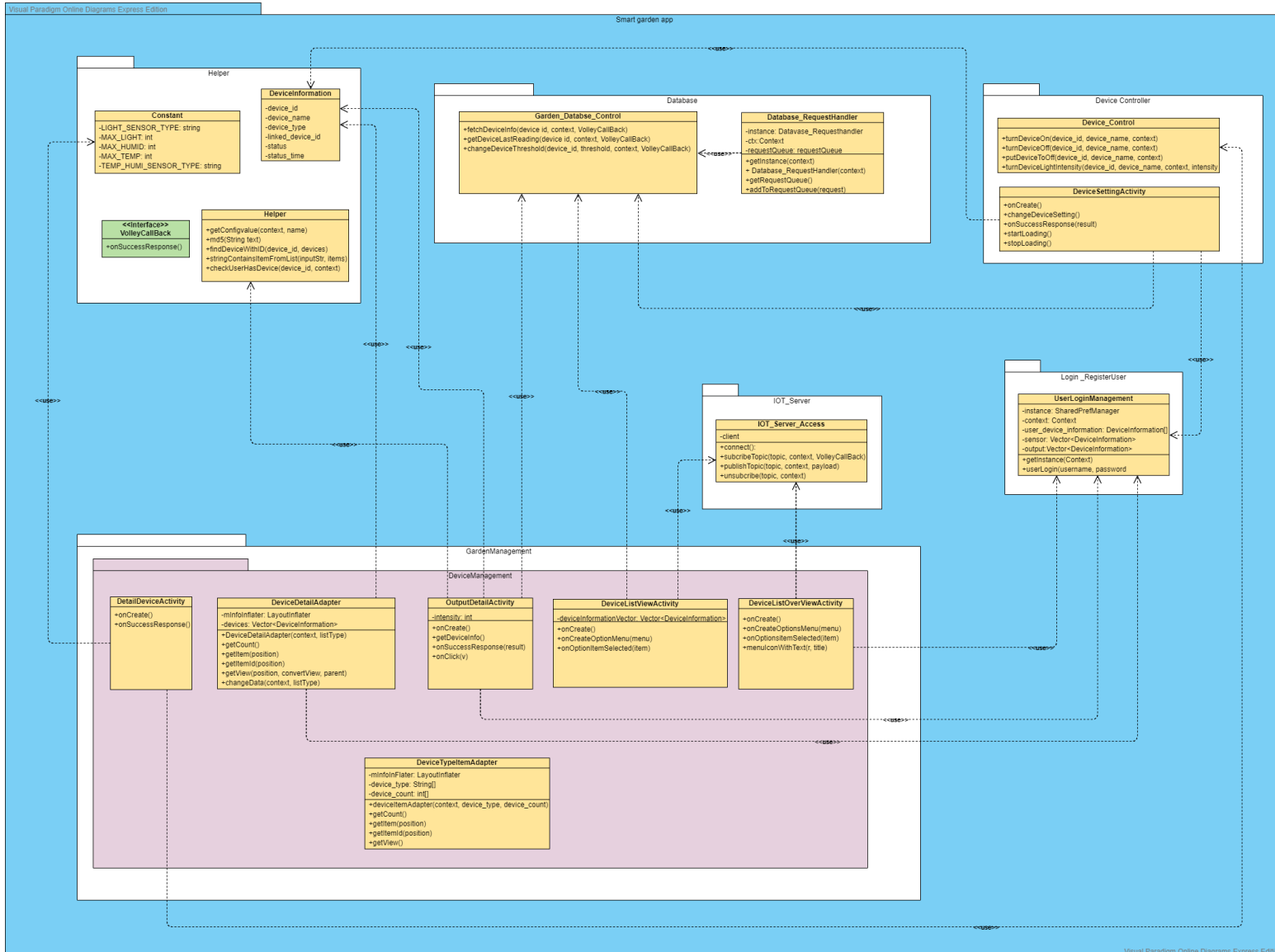
- For change device settings:



- For automatically turn on/off devices :



5. Class diagram and description



+ Package Device Controller

Class DeviceSettingActivity: Display sensor's threshold(s) based on its type (Light or Temp/Humi). Also let user changes threshold(s).

- **onCreate():** default function for activity, here used to get the sensors threshold(s), set up suitable view for different types of sensors and set functionality for buttons.
- **ChangeDeviceSetting():** filter function for data that user input.
- **onSuccessResponse():** inheritance method. For this class the method will get the message from the result of sending user data of changing threshold(s) and then return to DeviceListView.

+ Package IOT_Server

Class IoT_Server_Access: contains function that allow user to work with topic in server:

- **Connect(context):** connect to the MQTT server.
- **Subscribe(topic, context):** subscribe to a topic on MQTT server. When message arrive, each class using this method will have different implementations.
- **Publish(topic, payload, context):** publish payload to the topic.
- **Unsubscribe(topic, context):** unsubscribe from a topic.

Interface VolleyCallBack: contain function when a response from database action arrive successfully.

+Package Login_RegisterUser

Class UserLoginManagement: control user login profile

- **getInstance(context):** Return the static instance of UserLoginManagement for the context to use

+Package Database

Class Database_RequestHandler: a class that handle sending request to the database server.

- **getInstance(context):** get the static Database_RequestHandler instance for the context.
- **getRequestQueue():** if requestQueue is null, then initial it, else return it.
- **addToRequestQueue(req):** add req onto the req queue for handling request.

Class Garden_Database_Control: contains functions to access database to get information or to update database.

- **fetchDeviceInfo(device_id, context, volleyCallBack):** get the information of device from database based on device ID.
- **getDeviceLastReading(device_id, context, volleyCallback):** get the latest measurement(s) from sensors based on device ID.
- **changeDeviceThreshold(device_id, threshold, context, VolleyCallBack):** update database with new threshold(s).

+Package Helper

Interface VolleyCallBack: contain function when a response from database action arrive successfully

Class Helper: get properties value from the configuration file

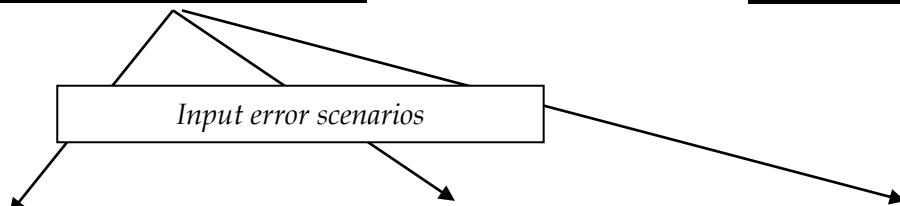
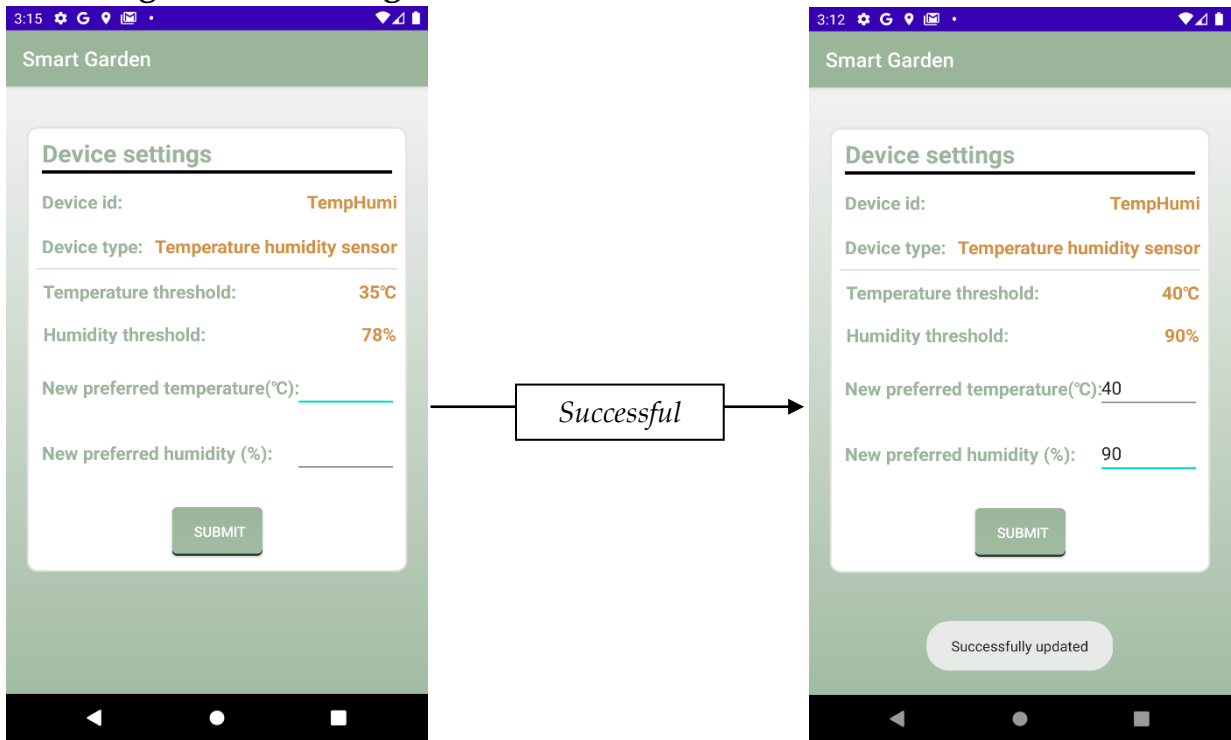
- **getConfigValue(Context context, String name):** get the value of name field in the config file.
- **findDeviceWithId(device_id, devices):** return the DeviceInformation of device with device id

Class Constants: contains constants for the program

Class DeviceInformation: contains functions that return device's attributes (id, name, type, threshold, status, linked_device_name, linked_device_id)

6. Mock up

a. Change device setting



Smart Garden

Device settings

Device id: TempHumi

Device type: Temperature humidity sensor

Temperature threshold: 35°C

Humidity threshold: 78%

New preferred temperature(°C): 27

New preferred humidity (%):

SUBMIT

Required field empty

Smart Garden

Device settings

Device id: TempHumi

Device type: Temperature humidity sensor

Temperature threshold: 35°C

Humidity threshold: 78%

New preferred temperature(°C): 27

New preferred humidity (%): 0000

SUBMIT

Invalid threshold

Smart Garden

Device settings

Device id: TempHumi

Device type: Temperature humidity sensor

Temperature threshold: 35°C

Humidity threshold: 78%

New preferred temperature(°C): 99

New preferred humidity (%): 70

SUBMIT

Temperature threshold is too high

b. Automatically control device

Smart Garden

Device settings

Device id: TempHumi

Device type: Temperature humidity sensor

Temperature threshold: 35°C

Humidity threshold: 78%

New preferred temperature(°C):

New preferred humidity (%):

SUBMIT

*humidity is over limit
=> LightD is On*

Smart Garden

Humidity

80%

Temperature

32°C

Linked Sensor Reading

Device status: On

TextView

Device ID: Light_D

Device Name: Light_D_01

Device Topic: Topic/LightD

Smart Garden

Device settings

Device id: TempHumi

Device type: Temperature humidity sensor

Temperature threshold: 40°C

Humidity threshold: 90%

New preferred temperature(°C): 40

New preferred humidity (%): 90

SUBMIT

Successfully updated

*humidity is now under limit
=> LightD is turned Off*

Smart Garden

Humidity

80%

Temperature

32°C

Linked Sensor Reading

Device status: Off

TextView

Device ID: Light_D

Device Name: Light_D_01

Device Topic: Topic/LightD

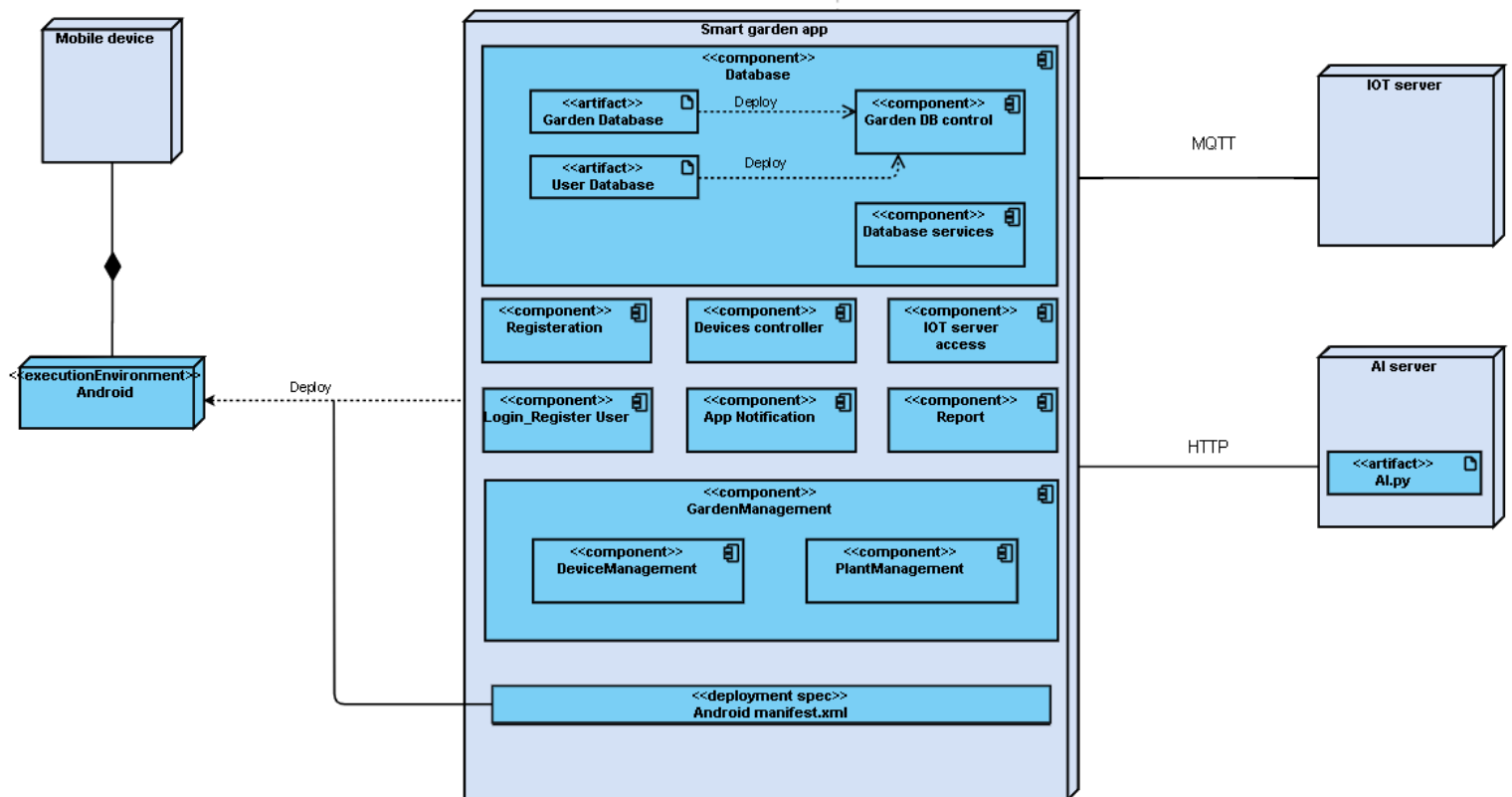
4 Architecture design

4.1 Architectural pattern

Our group reaches a unanimous decision for using the Model-View - Control pattern (MVC pattern) to implement our app.

4.2 Deployment diagram

The picture below is our deployment diagram of our group



5 Future improvements

In this part, our group will show some future improvement ideas which can be used for adding new functions for our app in the future.

5.1 Adding area parameter

In the final product, the user can view the measurements by choosing a user ID. This function is very useful if the user wants to measure effective gardening at the position. In other words, by viewing the measurement of each device, the user can keep track of information on specific locations and evaluate how this position is good for gardening. This feature is good when we have a small number of plants and in a small area such as in the home garden.

However, if the user is a company, this app should provide a function that allows user group devices into the area. This function allows the user to have a general insight into the area and help the user determines what position is the most effective solution for their plants.

Adding area parameter not only adding features for view report but also adding some features such as : add new area, change the area of the device, delete the area,... to satisfy many requirements of users.

5.2 Adding AI features

In the final product, our AI only detects outliers in a very short time which is already determined by the sampling period. The better app should have the more general AI algorithm which can detect general outlier measurements and remove it from the database.

Moreover, at the moment, the plant information of our app just only store passively in our database. In other words, the user can only register and modify their plant information. As each plant has its appropriate conditions for growth and development, by adding area function, our group should develop the corresponding AI to help the user decides which area is the best position for the specific plant.

5.3 About multi-platform app

In this course, our group provide this app on Android devices. Naturally, our group provide this app in more platform such as : PC, web... in order to enhance user experiences and convinience.

6 Document history

DOCUMENT HISTORY			
Version	Date	Changed by	Changes
1.0	20/4/2020	Duy	Summary a draft version of the report
1.1	21/4/2020	Duy	Changes after receive TA 's feedbacks
1.2	21/4/2020	An	Fix grammar errors
1.3	21/4/2020	Hung	Changes use-case details
2.0	27/4/2020	Hung	Fix a use-case diagram after receive TA 's feedbacks
2.1	2/5/2020	An	Add deployment diagram
2.2	3/5/2020	Ho	Fix notation mistakes in deployment diagram
2.3	3/5/2020	Duy	Fix connection mistakes in deployment diagram
3.0	3/5/2020	Hung	Fix use-case diagrams 6 & 7
3.1	3/5/2020	Every member	Add mock up for their own features
3.2	10/5/2020	Every member	Remove soil moisture sensor in our device list
F.0	25/7/2020	Duy	Modify view report feature
F.1	25/7/2020	Duy	Add AI and refresh feature
F.2	25/7/2020	An	Add manage plant informations
F.3	25/7/2020	Hung	Add manage device informations
F.4	28/7/2020	Every member	Add final version for their own features
F.5	29/7/2020	Duy	Add future improvements
F.6	30/7/2020	An	Change deployment diagram
F.7	30/7/2020	An	Modify overview
F.8	30/7/2020	Anh	Change class diagram for his feature