HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

──────── \* ───────

**THESIS**

SUBMITTED FOR PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

**ENGINEER**

IN

**INFORMATION TECHNOLOGY**

**HYDROPONIC SYSTEM FOR INDOOR PLANT: SERVER APPLICATION MODULE**

Author: **Trần Việt Hải**

Class ICT-59

Supervisor: **PhD.** **Ngô Quỳnh Thu**

HANOI 5-2019

# REQUIREMENTS FOR THE THESIS

1. Student information

Student name: Trần Việt Hải

Tel: 039-840-9548 Email: haitranviet96@gmail.com

Class: ICT-59 Program: ICT

This thesis is performed at: School of Information and Communication Technology–

Hanoi University of Science and Technology

From 21/01/2019 to 25/05/2019

2. Goal of the thesis

* To help the farmer to have an easier automated way for plant cultivation.
* To create and design a front to back system for manage all IoT parts for the system.

3. Main tasks

* Learn the general knowledge in Internet of Things field.
* Study the lightweight messaging protocol (Message Queuing Telemetry Transport Protocol).
* Analyze real system requirements suitable for farming need and habits.
* Base on system requirements, design and implement the communication and control module for the system.
* Learn Spring Boot framework to build the server side and Reactjs to build the client side for the system.

4. Declaration of student:

I – *Trần Việt Hải* - hereby warrants that the Work and Presentation in this thesis are performed by myself under the supervision of *PhD. Ngô Quỳnh Thu*.

All results presented in this thesis are truthful and are not copied from any other work.

|  |  |
| --- | --- |
|  | *Hanoi, 25 May 2019*  Author  *Trần Việt Hải* |

5. Attestation of the supervisor on the fulfillment of the requirements of the thesis:

|  |  |
| --- | --- |
|  | *Hanoi, 25 May 2019*  Supervisor  *PhD. Ngô Quỳnh Thu* |

# ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to all the people who have taught, helped and supported me during the training and learning time at Hanoi University of Science and Technology.

Foremost, I would like to express my sincere thanks my supervisor, PhD. Ngo Quynh Thu, who has been constantly giving careful guidance and invaluable advice to help me overcome many challenges during my work. Undoubtedly, I truly appreciate her willingness to spend her little free time for various business in deep discussions between us. I am thankful to have her as my supervisor.

My sincere thanks also go to MSc. Pham Ngoc Hung for helping me do this thesis. Without him leading me working on the designing phase, I would not complete the project for this thesis.

Second, I am appreciative the staffs at School of Information and Communication Technology for their benevolence of giving such a good education in which we are taught with comprehensive knowledge and experiences, which become strong foundation for our future career.

Third, I want to mention my grateful for to my teammates, who have shown their hardworking and responsibility in every project we went through the last five years at HUST. Also, I want to thank my collaborates in Network Communication Technology laboratory to help me finish this huge project for this thesis.

Last but not least, I would like to express my grateful to my family - my father, my mother and my sister for their unconditional and endless love and always beside me to support every step of my life.

# TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Ngày nay, khi sự nóng lên toàn cầu đang ngày một ảnh hưởng tiêu cực tới mọi lĩnh vực trong đời sống, thì nông nghiệp là một trong những mảng bị ảnh hưởng lớn nhất. Trong nông nghiệp, có rất ít các loài cây có thể sinh trưởng và phát triển tốt trên những môi trường khắc nghiệt, và chúng cũng không thường được sử dụng để làm nguồn lượng thực cho con người. Tuy nhiên, với sự phát triển của khoa học công nghệ như hiện nay, ngành nông nghiệp thông mình chính là chìa khóa để khắc phục tình trạng này, đặc biệt là phương pháp thủy canh. Cụ thể hơn, thủy canh là kĩ thuật trồng cây trong dung dịch và không sử dụng đất mà trồng trực tiếp vào môi trường nước chứa dinh dưỡng hoặc giá thể mà không phải là đất. Những người nông dân đã sử dụng phương pháp thủy canh đều nhận định rằng họ thu hoạch với năng suất cao hơn nhiều lần so với các phương pháp thủ công truyền thống.

Hiện nay, việc áp dụng các giải pháp công nghệ thông tin trong các hệ thống thủy canh này cũng có thể mang lại nhiều giá trị hữu hình hơn cho nông dân như tăng sản lượng và giảm chi phí. Do đó, trong đồ án này, em đã chọn cách nuôi trồng những loại cây rau ăn lá hay ăn quả thông thường trên hệ thống thủy canh. Hệ thống của em được vận hành một cách tự động bởi những thiết bị được gắn cảm biến trên một hệ thống IoT. Người nông dân khi sử dụng hệ thống này chỉ cần gieo hạt và kết nối vào hệ thống. Sau đó, hệ thống sẽ tự động điều chỉnh các thiết bị giúp cho cây có một môi trường phát triển lý tưởng để sinh trưởng và phát triển.

Nhờ vậy, hệ thống giảm tối đa công việc của người nông dân. Đồng thời, nó sẽ thu thập dữ liệu để phân tích hiển thị và điều khiển ngược lại để có thể tối ưu hóa nặng suất cây trồng cũng như là tiết kiệm tối đa nguồn dinh dưỡng và nguồn nước được sử dụng.

# ABSTRACT OF THESIS

Nowadays, when global warming is adversely affecting all areas of life, agriculture is one of the most affected areas. In agriculture field, there are only a few plants that can grow and develop well in harsh environments, and they are not used as a source of food for humans. However, with the development of science and technology today, smart agriculture is the key to remedy this situation, especially, hydroponics. More specifically, hydroponics is a technique of growing plants in solution and not using soil but growing directly into nutrient-containing water or substrate but not soil. Hydroponic growers have found they get yields many times greater when they switch from conventional methods.

Currently, the application of information communication technology solutions in these hydroponic systems can also bring a lot of tangible value than that to farmers such as increasing production and lowering costs. Therefore, in this thesis, I will focus on designing and developing an information communication technology hydroponics system applied Internet of Things solutions to cultivate leafy vegetables and short-term fruit giving crops. My system is operated by collecting data by devices mounted with sensors. Thereby, analyzing to bring added value for customer. Farmers who use this system just need to sow the seeds and connect hydroponic frame to the system. After that, the system will automatically control the device to adjust parameters to help the plant have an ideal development environment for growth and development.

Thus, this system minimizes the work of farmers. At the same time, it will collect data for display, analysis and control devices to be able to optimize crop productivity as well as maximize the efficient of nutrients and save water consumption.

# TABLE OF CONTENTS

[**REQUIREMENTS FOR THE THESIS ii**](#_Toc9644513)

[**ACKNOWLEDGEMENTS iii**](#_Toc9644514)

[**TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP iv**](#_Toc9644515)

[**ABSTRACT OF THESIS v**](#_Toc9644516)

[**TABLE OF CONTENTS vi**](#_Toc9644517)

[**LIST OF TABLES viii**](#_Toc9644518)

[**LIST OF FIGURES ix**](#_Toc9644519)

[**ACRONYMS x**](#_Toc9644520)

[**Chapter 1: Introduction 1**](#_Toc9644521)

[**1.1 Motivation 1**](#_Toc9644522)

[**1.2 Thesis objective and scope 2**](#_Toc9644523)

[**1.3 Solution orientation 2**](#_Toc9644524)

[**1.4 Thesis layout 3**](#_Toc9644525)

[**Chapter 2: Requirements Analysis 4**](#_Toc9644526)

[**2.1 Present condition 4**](#_Toc9644529)

[**2.2 Feature overview 6**](#_Toc9644530)

[2.2.1 General use case diagram 6](#_Toc9644535)

[2.2.2 Use case “Authentication” decomposition 8](#_Toc9644536)

[2.2.3 Use case “CRUD crop” decomposition 8](#_Toc9644537)

[2.2.4 Use case “Register device” decomposition 9](#_Toc9644538)

[2.2.5 Use case “Control device” decomposition 9](#_Toc9644539)

[2.2.6 Business processes 10](#_Toc9644540)

[**2.3 Use case specification 13**](#_Toc9644541)

[2.3.1 Use case “Sign up” specification 13](#_Toc9644547)

[2.3.2 Use case “Login” specification 14](#_Toc9644548)

[2.3.3 Use case “Create crop” specification 15](#_Toc9644549)

[2.3.4 Use case “Register device” specification 16](#_Toc9644550)

[2.3.5 Use case “Manual Control device” specification 17](#_Toc9644551)

[**2.4 Supplementary specification 18**](#_Toc9644552)

[2.4.1 Functionality 18](#_Toc9644559)

[2.4.2 Usability 18](#_Toc9644560)

[2.4.3 Performance 18](#_Toc9644561)

[2.4.4 Reliability 18](#_Toc9644562)

[**Chapter 3: Technology 19**](#_Toc9644563)

[**3.1 OVERVIEW OF THE INTERNET OF THINGS (IoT) 19**](#_Toc9644567)

[**3.2 MQTT PROTOCOL 19**](#_Toc9644568)

[3.2.1 MQTT Overview 19](#_Toc9644569)

[3.2.2 Architecture and operating mechanism of MQTT protocol 20](#_Toc9644570)

[3.2.3 Quality of service – QoS 20](#_Toc9644571)

[3.2.4 Security in MQTT 21](#_Toc9644572)

[**3.3 Eclipse Paho 21**](#_Toc9644573)

[**3.4 Java Spring Boot framework 22**](#_Toc9644574)

[**3.5 ReactJS 23**](#_Toc9644575)

[**Chapter 4: Application development and deployment 25**](#_Toc9644576)

[**4.1 Architecture design 25**](#_Toc9644579)

[4.1.1 Choose software architecture 25](#_Toc9644585)

[4.1.2 Overview design 27](#_Toc9644586)

[4.1.3 Package design specification 28](#_Toc9644587)

[4.1.4 APIs design specification 29](#_Toc9644588)

[**4.2 Design specification 30**](#_Toc9644589)

[4.2.1 Interface design 30](#_Toc9644596)

[4.2.2 Class design 30](#_Toc9644597)

[4.2.3 Database design 32](#_Toc9644598)

[**4.3 Application implementation 34**](#_Toc9644599)

[4.3.1 Library and framework 34](#_Toc9644607)

[4.3.2 Achievement results 35](#_Toc9644608)

[4.3.3 Main screens and features 36](#_Toc9644609)

[**4.4 Deployment 41**](#_Toc9644610)

[4.6.1 System setup 41](#_Toc9644614)

[4.6.2 System evaluation 42](#_Toc9644615)

[**Chapter 5: Feature solutions and contributions 43**](#_Toc9644616)

[**5.1 Authentication service 43**](#_Toc9644618)

[**5.2 Keep Alive service 45**](#_Toc9644619)

[**5.3 Collect service 47**](#_Toc9644620)

[**5.4 Control service 48**](#_Toc9644621)

[**Chapter 6: Conclusions and orientations 49**](#_Toc9644622)

[**6.1 Conclusions 49**](#_Toc9644629)

[**6.2 Orientations 49**](#_Toc9644630)

[**REFERENCES 51**](#_Toc9644631)

# LIST OF TABLES

[**Table 1** Use case **“**Sign up” specification table 13](#_Toc9644867)

[**Table 2** Input data for “Sign up” Use case 14](#_Toc9644868)

[**Table 3** Use case **“**Login” specification table 14](#_Toc9644869)

[**Table 4** Input data for “Login” Use case 15](#_Toc9644870)

[**Table 5** Use case **“**Create crop” specification table 15](#_Toc9644871)

[**Table 6** Input data for “Create crop” Use case 16](#_Toc9644872)

[**Table 7** Use case **“**Register device” specification table 16](#_Toc9644873)

[**Table 8** Input data for “Register device” Use case 17](#_Toc9644874)

[**Table 9** Use case **“**Manual Control device” specification table 17](#_Toc9644875)

[**Table 10** Input data for “Manual Control device” Use case 18](#_Toc9644876)

[**Table 11** APIs list for back-end and front-end 29](#_Toc9644877)

[**Table 12** Libraries and tools 35](#_Toc9644878)

[**Table 13** Project statistics table 36](#_Toc9644879)

# LIST OF FIGURES

[**Figure 1** System Design Block Diagram 3](#_Toc9644719)

[**Figure 2** Hachi’s hydroponic system 5](#_Toc9644720)

[**Figure 3** Data Block Diagram 6](#_Toc9644721)

[**Figure 4** General use-case Diagram 7](#_Toc9644722)

[**Figure 5** “Authentication” Use case Diagram 8](#_Toc9644723)

[**Figure 6** “CRUD Crop” Use case Diagram 8](#_Toc9644724)

[**Figure 7** “Register device” Use case Diagram 9](#_Toc9644725)

[**Figure 8** “Control device” Use case Diagram 9](#_Toc9644726)

[**Figure 9** Create Crop process Activity Diagram 11](#_Toc9644727)

[**Figure 10** Register device process Activity Diagram 12](#_Toc9644728)

[**Figure 11** Architecture and mechanism of MQTT 20](#_Toc9644729)

[**Figure 12** Request and Response handling mechanisms of Spring Framework 22](#_Toc9644730)

[**Figure 13** A React component 24](#_Toc9644731)

[**Figure 14** MVC Model 25](#_Toc9644732)

[**Figure 15** Service-Oriented Architecture 26](#_Toc9644733)

[**Figure 16** UML Package diagram 27](#_Toc9644734)

[**Figure 17** Backend Server Package Design Specification 28](#_Toc9644735)

[**Figure 18** Main Page Mockup 30](#_Toc9644736)

[**Figure 19** Authentication Communication Diagram 31](#_Toc9644737)

[**Figure 20** Keep Alive Communication Diagram 31](#_Toc9644738)

[**Figure 21** Control Communication Diagram 32](#_Toc9644739)

[**Figure 22** Database of a Hydroponic IoT system 33](#_Toc9644740)

[**Figure 23** Sign up User Interface 36](#_Toc9644741)

[**Figure 24** Login User Interface 37](#_Toc9644742)

[**Figure 25** User Profile Interface 37](#_Toc9644743)

[**Figure 26** List crop User Interface 38](#_Toc9644744)

[**Figure 27** Create new Crop User Interface 38](#_Toc9644745)

[**Figure 28** Crop Details User Interface 39](#_Toc9644746)

[**Figure 29** Device Details User Interface 39](#_Toc9644747)

[**Figure 31** List of Devices User Interface 40](#_Toc9644748)

[**Figure 32** Register Device User Interface 40](#_Toc9644749)

[**Figure 33** The device authentication sequence in the system 44](#_Toc9644750)

[**Figure 34** The keep alive sequence in the system 46](#_Toc9644751)

[**Figure 35** The collect sequence in the system 47](#_Toc9644752)

# ACRONYMS

|  |  |
| --- | --- |
| **Acronym** | **Whole word** |
| **IoT** | Internet of Things |
| **MQTT** | Message Queuing Telemetry Transport |
| **MVC** | Model – View – Controller |
| **HTTP** | Hyper Text Transfer Protocol |
| **EC** | Electro Conductivity |
| **QoS** | Quality of Service |
| **SSID** | Service Set Identifier |
| **AP** | Access Point |
| **API** | Application programing Interface |
| **UI** | User Interface |
| **CRUD** | Create Read Update Delete |
| **JDK** | Java Development Kit |
| **JRE** | Java Runtime Environment |
| **IDE** | Integrated Development Environment |
| **UML** | Unified Modeling Language |
| **JWT** | JSON Web Token |

# Chapter 1: Introduction

## Motivation

Over last few years, Hydroponics is an advanced method of growing trees that many investors care about and build not only because day by day it becomes more popular and profitable but also be one of the best solutions against global warming. The hydroponic system possesses many advantages compared to the traditional farming models. Since hydroponically developed plants plunge their roots straightforwardly into nutrient-rich solutions, they get what they require much more effectively than plants growing in soil, so they need much smaller root systems and can divert more energy into leaf and stem growth. With smaller roots, farmer can grow more plants within the same area and get more yield from the same amount of space (which is particularly great news in case you're growing in a limited area like a greenhouse or on a balcony or window-ledge inside). Hydroponic plants also grow faster. Many pests are carried in soil, so doing without it generally gives you a more hygienic growing system with less problems of disease. Therefore, hydroponics is perfect for indoor growing, you will be able to grow plants all year round and thereby increase significantly production of the plant.

This method is very familiar in developed countries such as Japan, Korea, Russia, ... In Vietnam, Hydroponic farms, which scales up to thousands of m² have been built about 10 years ago. These farms can potentially increase productivity by as much as ten times compared to traditional agriculture, all without using land and harmful pesticides. In Da Lat and some southern provinces, the model of hydroponic farms not only bring economic value but also provide a large quantity of safe vegetables to the market every year. However, these farms have not implemented hydroponics systems that use IoT solutions to collect crop data, analyze it, and make decisions about the plant's habitat. Consequently, the effectiveness of these methods is quite low, mainly for monitoring, not yet combined with data analysis and also depends too much on farmer's efforts to control the system.

Therefore, if we can solve these problems of hydroponics system in Vietnam, it can bring huge benefit not only for farmer but also home country agriculture. By applying IoT solutions for collecting, analyzing and controlling for indoor hydroponic system, they will: (i) decrease system cost: includes both initial and operating costs, (ii) become scalable system and can be suitable for models from small to big field, (iii) increase plant production by control how the best amount of nutrition, and water are mixed appropriately for each plant, (iv) automatic watering, mixing the solutions and keep the greenhouses' environmental conditions at best for plant and saving more water and nutrition consumption. In short, they will increase crop production and beneficial to farmers.

## Thesis objective and scope

From the above problem, there are fundamental parameters that got to be checked and balanced for any sort of plants. Consequently, indeed high-tech hydroponic systems will have a few comparable capacities. Hence, to capture the essential capacities of a hydroponic framework, we are going look at the frameworks as of now within the advertise. On the market today, there are some products that combine IoT and hydroponic plant production such as Hachi, Greenbot or Lisado.

Their system can monitor the variables that influence the plant development such as temperature, humidity, light intensity, pH level, give warning and collect date for user. Moreover, farmers can control plant’s device both manual and semi-automatically. Though, Hachi's frame are not able of analyzing environment data to provide control strategies to save planting resources.

## Solution orientation

To meet these requirements, we will present a indoor hydroponic system model that includes the following modules:

First, IoT infrastructure module includes IoT devices, sensors, devices that can affect the plant's habitat. The module tasked with collecting data from the sensor, receiving control commands and executing control mechanisms.

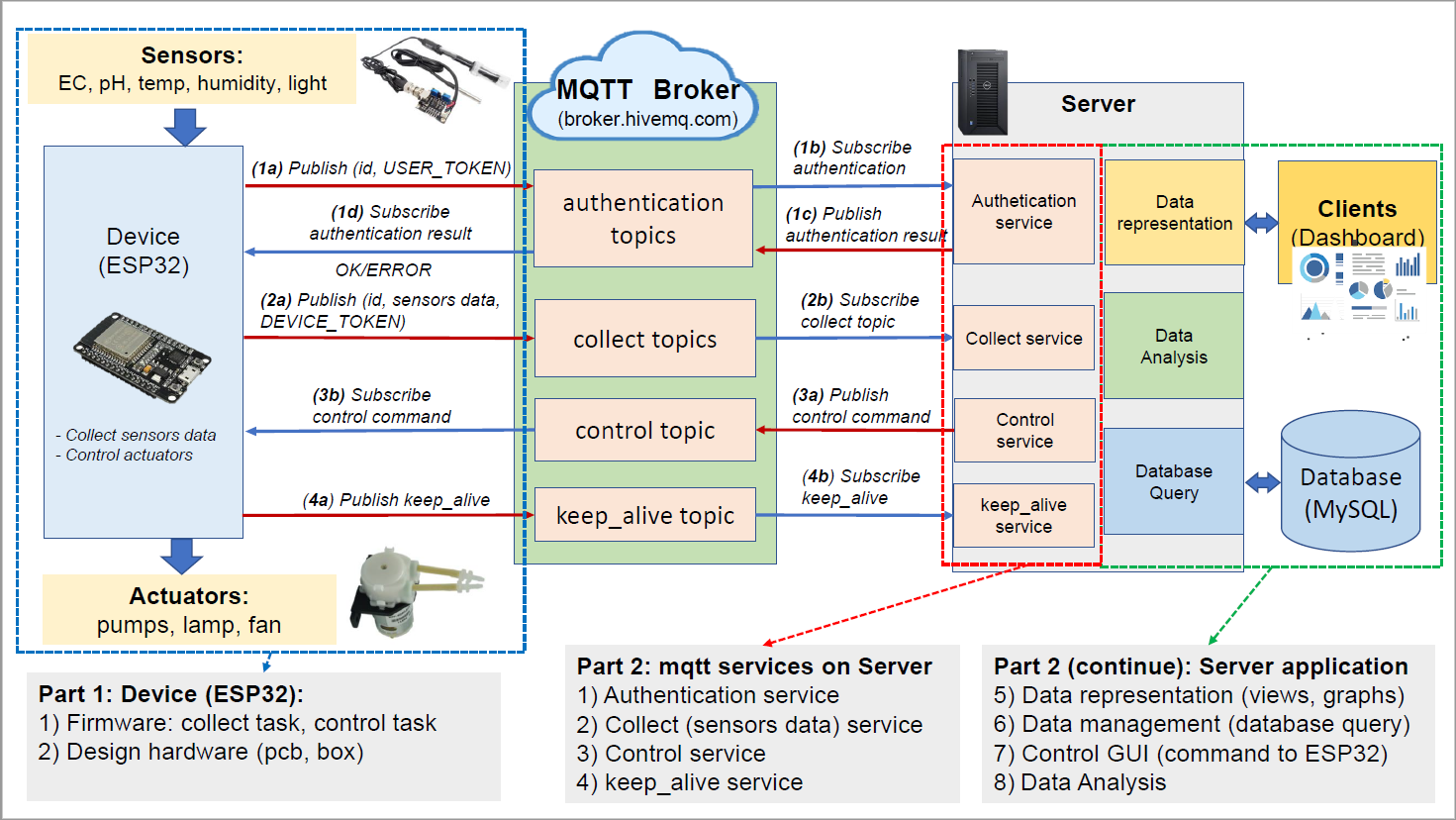
Secondly, Server application module is the medium between the device and the data processing layer which includes a mechanism for managing the delivery of messages from the device side.

Last, data processing module will process and manage data. Received data will be analyzed and evaluated to make control decisions for the equipment. Management applications help users easily manage objects, equipment.

As the objective is build a system can meet farmer desired which is low cost and high productivity. We will use Spring boot which is a Java framework as server side of the server application. This server will receive all the collected data by devices. Then, it manages them and put them in the rational database. In this case, we use a relational database management system because we want to keep data integrity and Logic-related discrete data requirements are identified. The database management we use is MySQL. Above that, for client interface, Reactjs definitely is the best option not only cause JavaScript nowadays becoming one the most popular languages but also the language is so quickly to adapting and creating application server. Among those servers and devices, we use MQTT protocol for server and device communication. In contrast, HTTP protocol connects between server side and client side which are using the same API.

Our solution is building two servers coming along. One server will take care of all the data analysis and data management. The other one provides user interface, which also called front-end server. The purpose of front-end server is providing user the interface to interact with the system. Users can control their devices. Whilst, the backend will be in charge of automatically control devices based on data collected. Then, we have a front to back synchronizing system.

With such solutions, I have built a server application module that has full fill all the functions that satisfied users requirements. The system help user monitor all the data being through and control device if they want wirelessly and do not need their appearance there.



**Figure 1** System Design Block Diagram

## Thesis layout

The rest of this graduation thesis is organized as follows.

Chapter 2 presents all the step from analyzing user requirement to the process of producing system requirement.

In chapter 3, I introduce all the applied technology, which is Spring Boot for back end, ReactJS for front-end and some other libraries and protocols.

After that, chapter 4 will demonstrate how the system can be deploy and how the system organizes. Also, the chapter will have result of all the project and the pictures of the user interface of application.

Chapter 5 show all my features solutions and contribution belong to thesis. This chapter presents the main contribution of the project, which is a set of services that help the system has the ability to communicate with the device.

Finally, Chapter 6 is the last chapter, which promote how system can be in the future. In reality, the orientation to boost it to production are mentioned in this section. The content in this chapter also talks about what has been achieved and has not been achieved. If it is possible to complete the issues raised here, the system will be able to future proof and ready to go to production.

# Chapter 2: Requirements Analysis



## Present condition

As mentioned in chapter 1, there are some products that combine IoT and hydroponic plant production such as Hachi, Greenbot or Lisado. Among those, Hachi is most prominent one.

Hachi’s product aim two main customers - urban residents and farm owners. Urban-home version includes features like: (i) Monitor humidity, soil moisture, (ii) Monitor light intensity, (iii) Warnings when environmental conditions exceed the critical level. (iv) users can easily set the appropriate factor and alert conditions most suitable to the plants, (iv) control devices such as pumps, LEDs or cameras via smartphones and Internet. Whilst, farmer version includes features like: (i) monitor humidity, soil moisture, light intensity, pH in water, (ii) warnings when environmental conditions exceed the critical level (iii) control mode option so that users can easily set the appropriate factor and alert conditions most suitable to the plants, (iv) control devices such as pumps, LEDs or cameras through smartphones and Internet, and some additional feature: (v) support Barcode for easy traceability of agricultural products, (vi) use the Zigbee wireless connection protocol to connect slave nodes together and use Wi-Fi to connect to the master node. In short, the Hachi’s hydroponic system have the model as follows:



**Figure 2** Hachi’s hydroponic system

Therefore, a fully automatic vegetable cultivation indoor system, which capable of self-analyzing and supporting farmers, will save farmers a lot of farming effort and not only decrease invested money but also increase productivity.

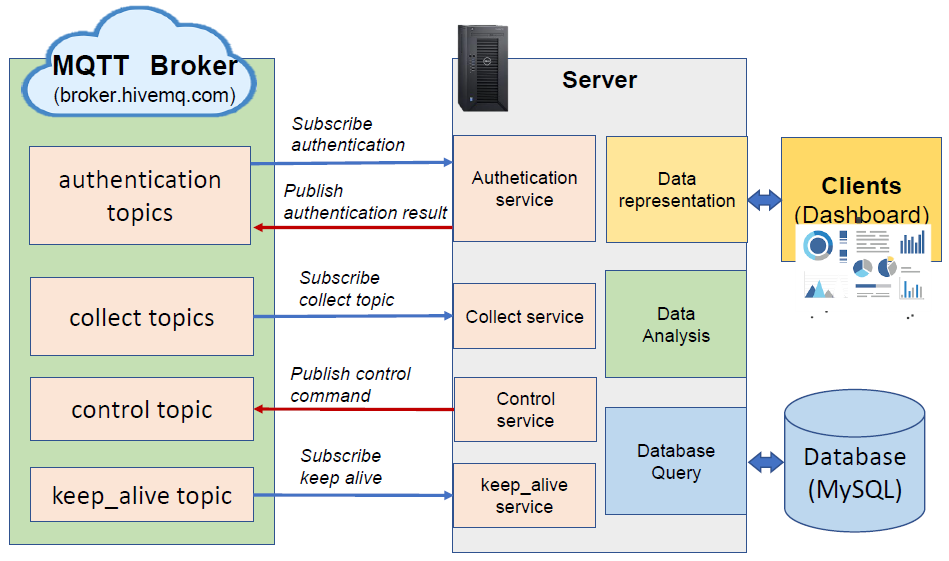
After going through what nowadays hydroponics system on the market can do, we can cover all the requirements of an automated hydroponics system.

The system must have an authentication service to keep security for the system and device, prevent all threats that can ham both the server either user’s device. A communication system which easily to catch but also need to be safe and stable enough to make sure both sides receive data on time and lossless.

The devices must connect to the system wirelessly. Also, the connection must be stable and have capable to reconnect automatically even when disconnected to a power source or lost wireless communication.

Portable device has all user needed sensor that measure the environmental parameters that may affect the development of the plant and send data to the processor server. These sensors can easily remove if not needed. During device production, devices can be divided to multiple model which have differences in number and kinds of sensors to targeting different users subject.

The system is scalable with larger models and is suitable for a wide variety of control circuits. The system ensures real-time accuracy. The system is fully automatic, but users can manually control it if desired. The system can analyze sent data to make resource-saving decisions. The system retains a control log. The system can display diagrams of real time measured data.



**Figure 3** Data Block Diagram

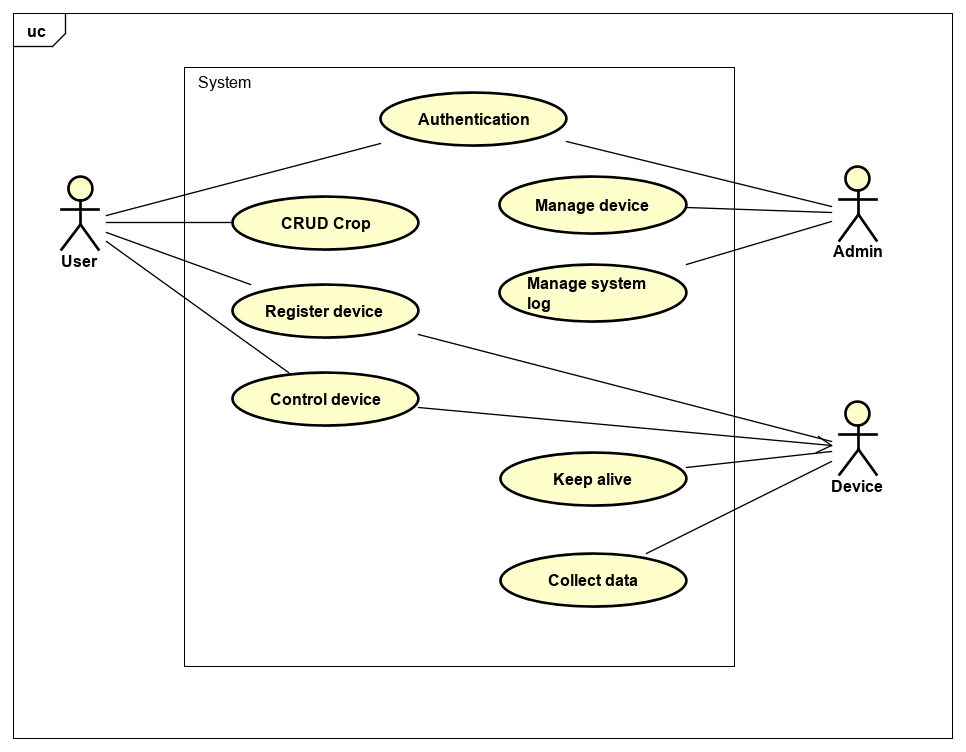
These are the most common requirements of an IoT application system that fully meets the criteria outlined in the question set. No current system has fully met these requirements.

## Feature overview



### General use case diagram

With the declared requirements for the system, the general use case diagram design is as follows:



**Figure 4** General use-case Diagram

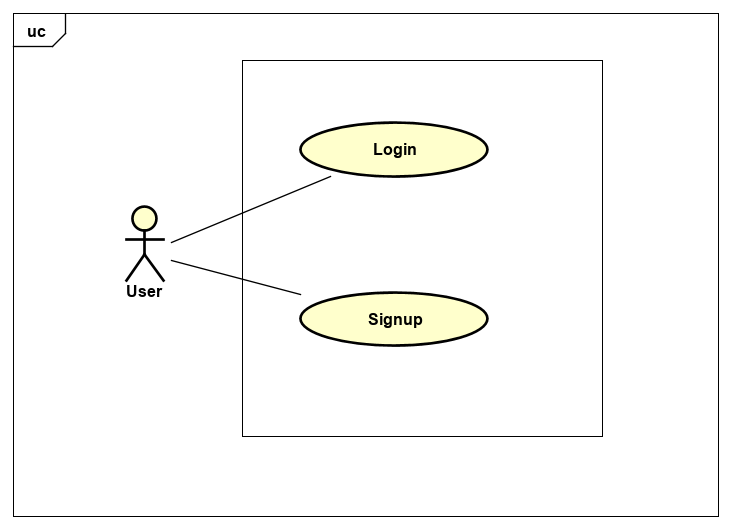
In this diagram, three main actors are User, Device and Admin.

Users, as farmers and normal user, and admin can make authentication action such as Signup, Login.

After login, Users use the system to create, update, read, delete their own Crop. Also, they can register new device and ready to link it to the crop.

When users start planting and the system create crop for them successfully. They can control their device to work. Or, they just leave it to the server automatic collect and plant for them.

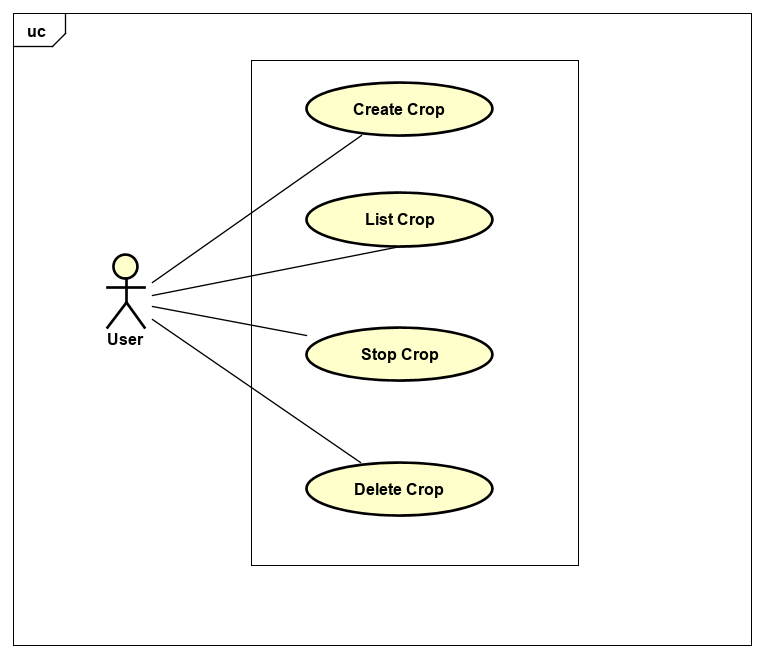
### Use case “Authentication” decomposition



**Figure 5** “Authentication” Use case Diagram

In the figure, we can see that use case “Authentication” includes two use case which is Login and Signup. These use cases are first case when the user start the web application. Every user needs to sign up a user account to login to the system and to use every other function of the web application.

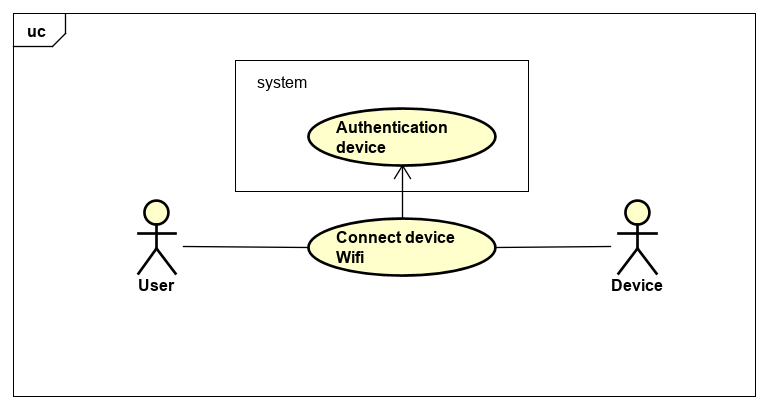
### Use case “CRUD crop” decomposition



**Figure 6** “CRUD Crop” Use case Diagram

The “CRUD crop” use case contains of four other use case which is user create crop, user read list of their own crop, stop collecting crop and delete the unused or harvested crop.

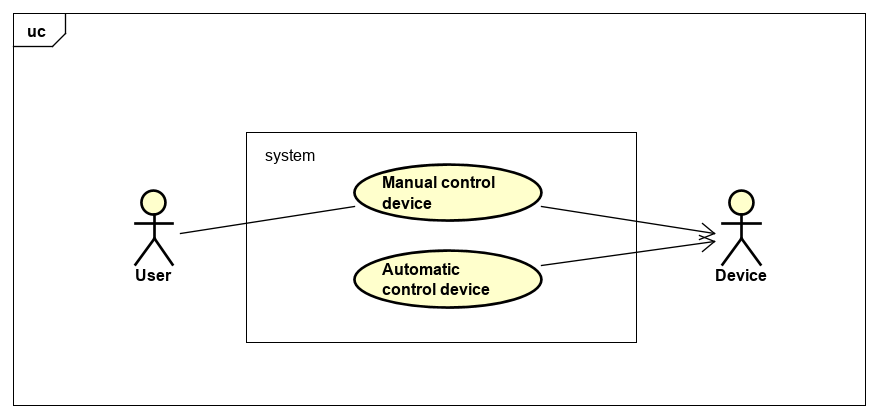
### Use case “Register device” decomposition



**Figure 7** “Register device” Use case Diagram

The “Register device” use case is divided into two use cases. The “Authentication device” use case works on the server itself. In the other hand, the “Connect device Wi-Fi” use case is the physically use case that happen between only the device and user.

### Use case “Control device” decomposition



**Figure 8** “Control device” Use case Diagram

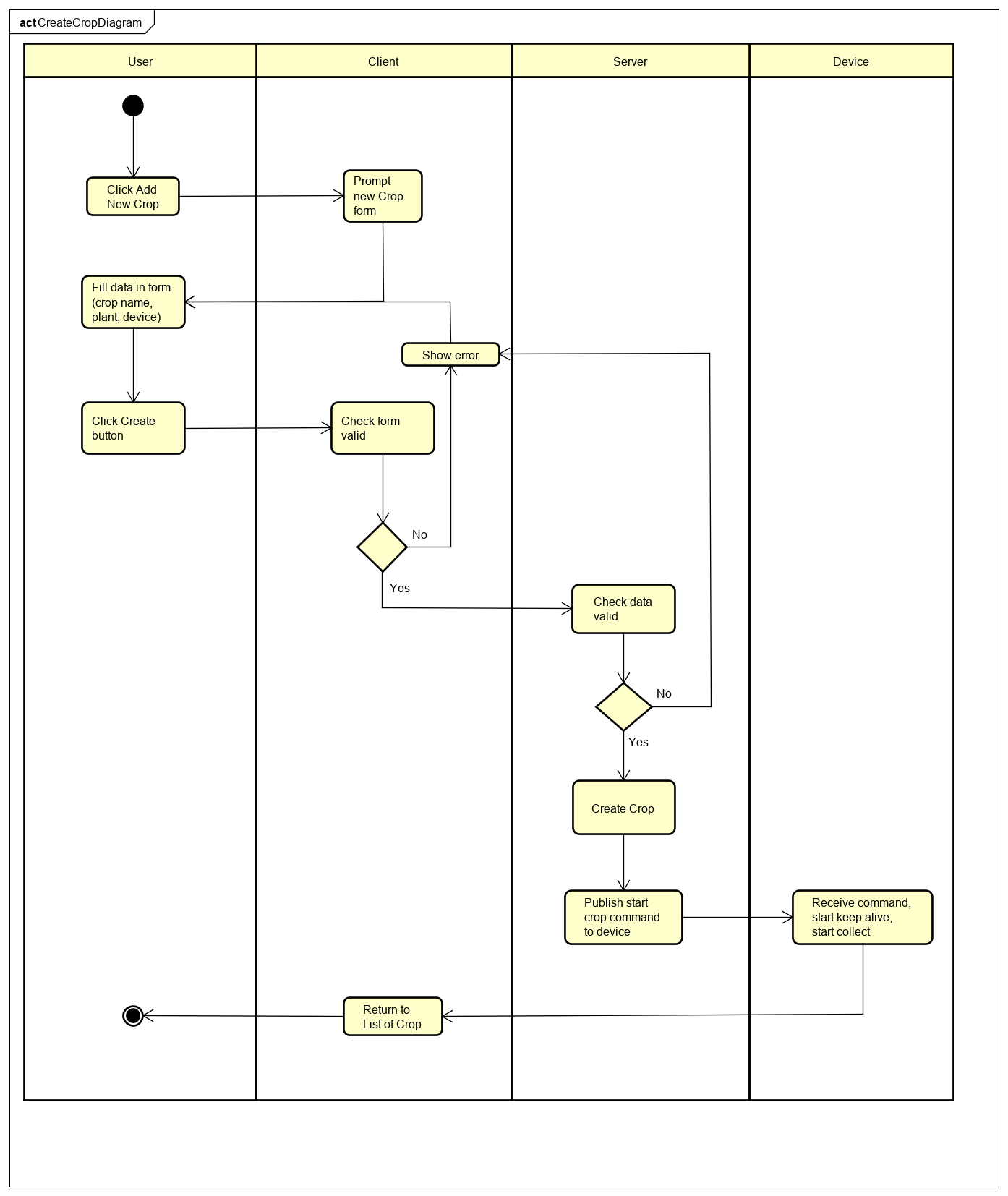
As the figure shown, the “Control device” use case is split into a manual control command use case or automatic control command use case. Specifically, the device can be control by two action which is automatic by the system or manual that produced by the user.

### Business processes

In this system, there two main business processes: Create Crop process and Register Device process.

The details of these processes are modeled in the sub-section of each process.

#### Create Crop process

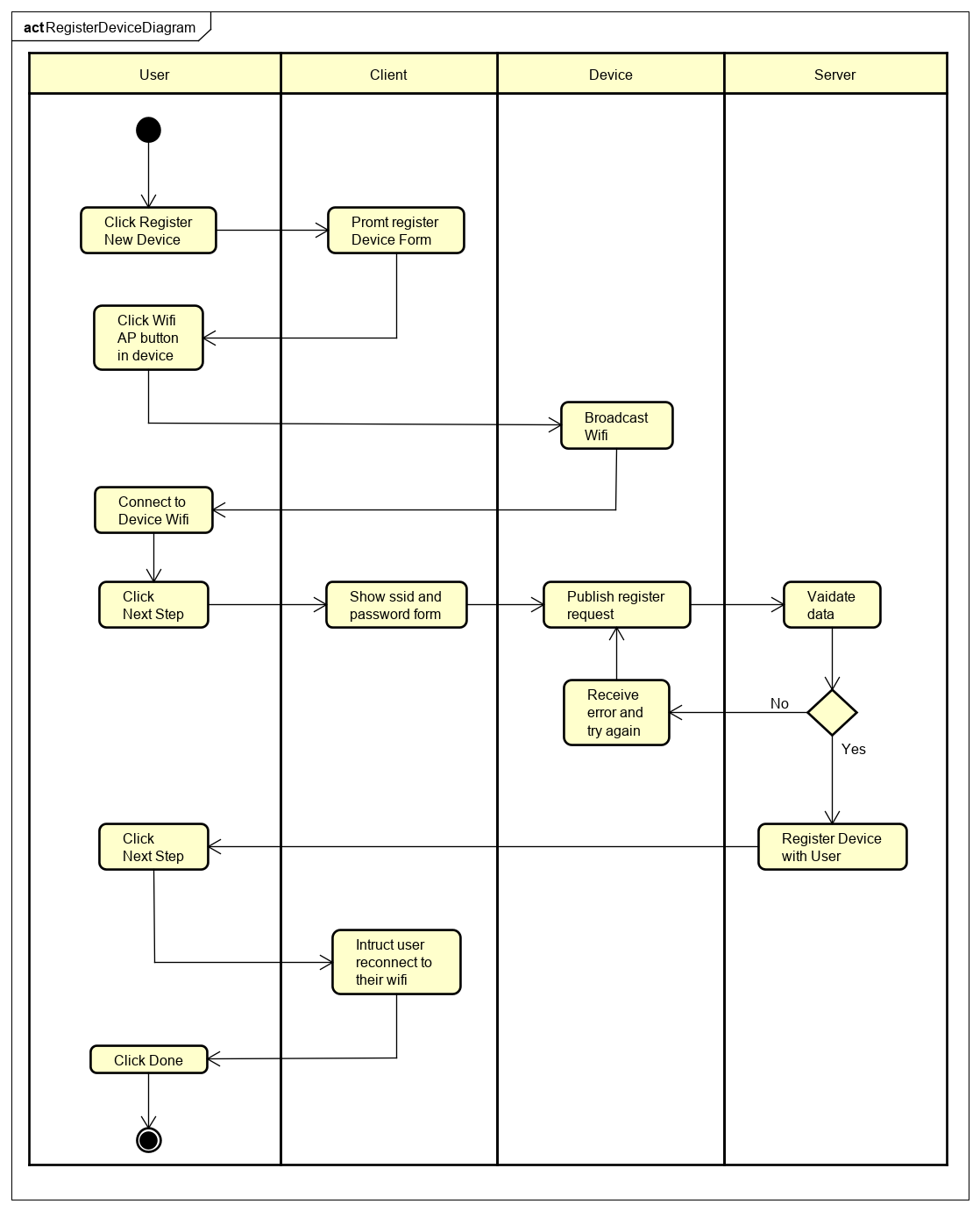


**Figure 9** Create Crop process Activity Diagram

This process takes place when a user who already has an account, has a registered device is preparing to start a new season. After successful login, user will go to the crop page and click the button “Add new Crop” to navigate to new crop page. In this page, user need to fill in the name of new crop and choose from the available device list. If only one field blanks, the form can not be submitted and will raise the error notification for user.

Then, the server receives client request to create new crop and validates it. At this point, if the validation false or stop by any problems, the server will response to client the error. After creating the crop successfully, the server will publish start command to the corresponding device so that the device can begin to collect data. Last, the server sends a successful response to client to display.

#### Register Device process



**Figure 10** Register device process Activity Diagram

Same as the create crop process, the register device process will go to new device page which will show a step by step instruction for user to follow. The first step is turning on the ESP32 AP and connecting to it. Next, user will see an input form to connect to their own Wi-Fi. After that, the device connects to user’s Wi-Fi and publish the register request, which carries the user token, to authentication topic. The server will validate the received data. In both case success or false, the system always publishes a message back to a corresponding authentication result topic. If validation successfully, the server registers the device for that user. Finally, user is redirected to device list page to check the new device on the list.

## Use case specification



### Use case “Sign up” specification

**Table 1** Use case **“**Sign up” specification table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC001 | **Use case name** | Sign up |
| **Actor** | User | | |
| **Prerequisites** | All user who have not had account yet | | |
| **Success/Main flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 1 | System | Prompt a Sign-up Form | | 2 | User | Fill data in the form (description below) | | 3 | System | Validate username, email | | 4 | User | Click “Sign up” button | | 5 | System | Add new user to database if data is valid | | 6 | System | Notify user success message | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | User | Can cancel returning or restart any time | |  | System | Notify user if any condition not satisfy | | 6a | System | Display errors if the data is not valid, back to step 2 | | | |
| **Post-conditon** | Don’t have any | | |

\* Input data of personal information include these data fields:

**Table 2** Input data for “Sign up” Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Data field** | **Description** | **Mandatory** | **Condition** | **Example** |
| 1. | Full name | Name of user | Yes | 4-40 | Trần Việt Hải |
| 2. | Username | Username of user | Yes | 3-15 | haitran |
| 3. | Email | Email of user | Yes | Must be email | haitran@gmail.com |
| 4. | Password | Password of the account | Yes | 6-20 | 123456 |

### Use case “Login” specification

**Table 3** Use case **“**Login” specification table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC002 | **Use case name** | Login |
| **Actor** | User | | |
| **Prerequisites** | All user who had account | | |
| **Success/Main flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 1 | System | Prompt a Sign in Form | | 2 | User | Fill data in the form (description below) | | 3 | User | Click “Sign in” button | | 4 | System | Create new JWT if data is valid | | 5 | System | Notify user success message | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | User | Can cancel returning or restart any time | |  | System | Notify user if any condition not satisfy | | 6a | System | Display errors if the data is not valid, back to step 2 | | | |
| **Post-conditon** | Logged in user | | |

\* Input data of personal information include these data fields:

**Table 4** Input data for “Login” Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Data field** | **Description** | **Mandatory** | **Condition** | **Example** |
| 1. | Username/Email | Username or email of user | Yes | 4-40 | haitran |
| 2. | Password | Password of the account | Yes | 6-20 | 123456 |

### Use case “Create crop” specification

**Table 5** Use case **“**Create crop” specification table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC003 | **Use case name** | Create crop |
| **Actor** | User | | |
| **Prerequisites** | User login successfully | | |
| **Success/Main flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 1 | System | Prompt a Create crop Form | | 2 | User | Fill data in the form (description below) | | 3 | User | Click “Create crop” button | | 4 | System | Add new crop to database if data is valid | | 5 | System | Publish to topic “nct\_control” to start crop | | 6 | System | Notify user success message | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | User | Can cancel returning or restart any time | |  | System | Notify user if any condition not satisfy | | 6a | System | Display errors if the data is not valid, back to step 2 | | | |
| **Post-conditon** | Don’t have any | | |

\* Input data of personal information include these data fields:

**Table 6** Input data for “Create crop” Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Data field** | **Description** | **Mandatory** | **Condition** | **Example** |
| 1. | Crop name | Name of crop | Yes | 4-140 | Happy Farming |
| 2. | Plant | Type of plant | Yes | Selection | Lettuce |
| 3. | Device | Device used | Yes | Selection | ESP32\_1 |

### Use case “Register device” specification

**Table 7** Use case **“**Register device” specification table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC004 | **Use case name** | Register device |
| **Actor** | User, Device | | |
| **Prerequisites** | User login successfully, plugged in AP-enabled device | | |
| **Success/Main flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 1 | System | Prompt a Register device Form | | 2 | User | Fill data in the form (description below) | | 3 | Device | After connected to the internet, publish authentication request to topic “nct\_authentication” | | 4 | System | Register device to user with authentic token if data receive is valid | | 5 | User | Click “Done” button | | 6 | System | Notify user success message | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | User | Can cancel returning or restart any time | |  | System | Notify user if any condition not satisfy | | 6a | System | Display errors if the data is not valid, back to step 2 | | | |
| **Post-conditon** | User’s Linked device | | |

\* Input data of personal information include these data fields:

**Table 8** Input data for “Register device” Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Data field** | **Description** | **Mandatory** | **Condition** | **Example** |
| 1. | User token | Token for authentication | Yes | Hidden | ‘A 512 Hash Map’ |
| 2. | SSID | Name of user’s WIFI | Yes |  | ABCDEFGHI |
| 3. | WIFI Password | Password of user’s WIFI | Yes |  | 12345678 |

### Use case “Manual Control device” specification

**Table 9** Use case **“**Manual Control device” specification table

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case code** | UC005 | **Use case name** | Manual Control device |
| **Actor** | User | | |
| **Prerequisites** | User login successfully, collecting device | | |
| **Success/Main flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | | 1 | System | Prompt a Actuator control Form | | 2 | User | Fill data in the form (description below) | | 3 | User | Click “Send command” button | | 4 | System | Publish a command to device if data valid | | 5 | System | Notify user success message | | | |
| **Alternative flow of event** | |  |  |  | | --- | --- | --- | | **#** | **Doer** | **Action** | |  | User | Can cancel returning or restart any time | |  | System | Notify user if any condition not satisfy | | 5a | System | Display errors if the data is not valid, back to step 2 | | | |
| **Post-conditon** | User’s Linked device | | |

\* Input data of personal information include these data fields:

**Table 10** Input data for “Manual Control device” Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Data field** | **Description** | **Mandatory** | **Condition** | **Example** |
| 1. | Actuator type | Actuator user want to control | Yes | Hidden | 1 |
| 2. | Params | Time for that actuator work | Yes | Number in seconds | 60 seconds |
| 3. | Action | Action to execute | Yes | ON/OFF | ON |

## Supplementary specification



### Functionality

On every event, the system needs to provide an error handler to inform user through user interface so that user is aware of what error is happening.

### Usability

The system is user friendly and online help makes using the system easy.

### Performance

The information is refreshed at regular intervals depending upon whether some updates have occurred or not.

The software shall respond to the user in not less than two seconds from the time of the request submittal. The software shall be allowed to take more time when doing large processing jobs.

Responses to view information shall take no longer than 3 seconds to appear on the screen.

### Reliability

The system is available 100% for user due to the importance of data; the damages can happen by incorrect or incomplete data.

The system shall be operational 24 hours a day and 7 days a week.

Even if the system fails, the system will be recovered back up within an hour or less.

# Chapter 3: Technology



## OVERVIEW OF THE INTERNET OF THINGS (IoT)

Internet of Things (IoT) is a term used to describe objects existed in a connected architecture. This phrase was coined by Kevin Ashton in 1999. He was a scientist who founded the Auto ID Center at MIT, the university that helped establish global standards for RFID, as well as some other types of sensors.

IoT is widely used in publications from companies and analysts. IoT are devices that can connect through Wi-Fi, 3G, 4G, Bluetooth, ZigBee. Intelligent devices can include wearable smart devices or smart home appliances.

By 2020, there will be about 50 billion objects connected to the Internet. An IoT network can hold up to 50 to 100 trillion connected objects and the network can track the movement of each object.

IoT is being applied to all aspects of social life, such as: smart home, emergency response, smart agriculture, medical, traffic, emergency response.

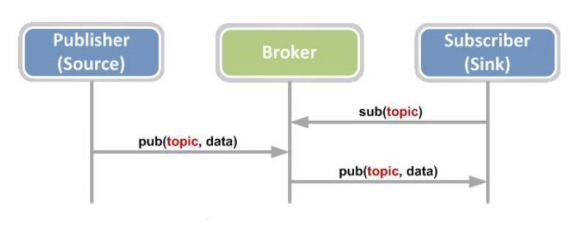
## MQTT PROTOCOL

### MQTT Overview

MQTT (Message Queuing Telemetry Transport) was designed by two engineers, Andy Stanford-Clark and Arlen Nipper, in 1999. It is a simple, easy-to-use, publish / subscribe message transfer protocol. It has low bandwidth, high reliability, and the ability to operate in unstable transmission conditions. Therefore, it is suitable for M2M and IoT applications.

MQTT operates on a client / server basis (each sensor is a client; a server is a broker middleware) via TCP / IP.

### Architecture and operating mechanism of MQTT protocol



**Figure 11** Architecture and mechanism of MQTT

In the system using MQTT, the client connects to a MQTT server (broker). Each client will subscribe a few topics to listen to the message.

Message: The data is transported through the MQTT protocol over the network to the application. Each message will have topic name and QoS.

Client: A program or device that uses the MQTT protocol, connects to the server and performs the action. Publish: Send messages to other clients. Subscribe: Subscribe to a topic which client wants to receive message. Unsubscribe: Unsubscribe to a topic. Disconnect: Disconnect from server.

Server: An intermediary between clients to receive subscriptions from the client and to publish messages to clients. The actions of the Broker include: (i) Accept incoming connection from Client, (ii) Accept the messages that Client publishes, (iii) Publish messages to subscribed Client, (iv) Manage subscribe and unsubscribe requests from the Client.

Topic: The labels are attached to a message to match the subscription to the server. The server sends the message to the clients through the registered subscription.

### Quality of service – QoS

Quality of Service is an agreement between the sender and receiver in ensuring message delivery. There are 3 levels of QoS in MQTT:

First is at most once (QoS 0): Messages reach the server at most once. The transmission depends on the reliability of TCP / IP.

Seconds is At least once (QoS 1): The server receives the message identified by a PUBACK message. If there is an error, or the confirmation message is not received after a certain period, the message will be sent again, with a DUP bit in the header of the message. So, the message is sent to the server at least once. If the client does not receive a PUBACK message for a defined period or an error occurs, the client sends the PUBLISH message again, with the DUP set up. When receiving a repeat message from the client side, the server will publish the messages to the subscriber and send another PUBACK message.

Last is Exactly once (QoS 2): Repeated messages do not reach the application. This is the highest level when delivering messages as repeated messages are not accepted. When QoS = 2, a message has a message ID in its header. If there is an error, or after a period, the protocol will redo the result of the final confirmation message or PUBLISH or PUBREL. Therefore, the message will reach the subscriber exactly once.

### Security in MQTT

MQTT is a lightweight protocol that does not have many security mechanisms, in practice it uses SSL / TLS encryption.

At the network layer, using a physical security network or VPN as the basis for the connection between the client and the broker is a way to provide secure and reliable connectivity.

At the application level, MQTT provides a user / password authentication when a client connects to the broker. Another way is to use load-level encryption on the application layer so that the information is transmitted securely without the full transport coding.

To apply the MQTT protocol in this problem we will use a very well-known library for MQTT support, namely, Paho. We will use the Eclipse Paho library for java.

## Eclipse Paho

Java Paho Client is a Java based MQTT library for developing applications running on the JVM or compatible platforms such as Android. Java Paho Client provides two APIs:

MqttAsyncClient is an entirely unsynchronized API, with the incoming data being processed in the pre-registered Callback Interface.

MqttClient is a synchronized wrapper around MqttAsyncClient, where functions appear synchronously with the application.

## Java Spring Boot framework

Spring is the most popular open source framework for building enterprise applications. The core features of Spring can be used to develop Java desktop, mobile and web application. In this project we will use spring to create a web server that allows users to interact with the hydroponic frames they have. The request and response handling mechanisms of the Spring Framework are shown in diagrams:



**Figure 12** Request and Response handling mechanisms of Spring Framework

As displayed in the figure, when there is an incoming URL request: all of them is received by The Spring Font Controller (DispatcherServlet). The DispatcherServlet finds an entry of handler mapping from the XML file and forwards the request to the controller. After executing the process from the client request, it implements the logic defined in the controller and returns the ModelAndView object. Based on the values ​​in the ModelAndView, the Font Controller sends requests to the View Resolver, the View Resolver looks up the config files returned by the Controller. Finally, the Font Controller will find the view file that was previously defined by the View Resolver and send the result to the user.

Spring Boot is basically an extension of the Spring framework which eliminated the boilerplate configurations required for setting up a Spring application. It takes an opinionated view of the Spring platform which paved the way for a faster and more efficient development eco-system.

Here are just a few of the features in Spring Boot: (i) Opinionated ‘starter’ dependencies to simplify build and application configuration, (ii) Embedded server to avoid complexity in application deployment, (iii) Metrics, Health check, and externalized configuration, (iv) Automatic config for Spring functionality – whenever possible.

In a few words, we can say that the Spring Boot is simply an extension of Spring itself to make the development, testing, and deployment more convenient.

## ReactJS

React (as known as React.js or ReactJS) is a JavaScript library for building client interfacing. It is developed by Facebook and a huge community of personal engineers and companies. It lets you compose complex UIs from small and isolated pieces of code called “components”.

React can be utilized as a base within the single-web page development or mobile phone applications. Because it is ideal for its expecting utilize of being the fastest strategy to fetch quickly changing information that has to be recorded. In any case, fetching information is just the starting of what happens on a web page. Consequences, complex React applications as a rule require the utilize of extra libraries for state management, routing, and interaction with an API.

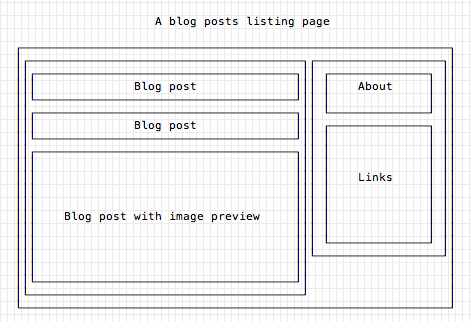
Then, there a lot of framework similar to ReactJS like Ember.js or Angular. But I choose ReactJS because:

React is less complex than other alternatives. At the time when React was announced, Ember.js and Angular 1.x were the predominant choices as a framework. Both these imposed too many conventions on the code that porting an existing app was not convenient at all. React made a choice to be very easy to integrate into an existing project, because that’s how they had to do it at Facebook in order to introduce it to the existing codebase. Also, those 2 frameworks brought too much to the table, while React only chose to implement the View layer instead of the full MVC stack.

ReactJS come out really on time. At the time, Angular 2.x was announced by Google, along with the backwards incompatibility and major changes it was going to bring. Moving from Angular 1 to 2 was like moving to a different framework, so this, along with execution speed improvements that React promised, made it something developers were eager to try.

ReactJS is maintained by Facebook. Being backed by Facebook obviously is going to benefit a project if it turns to be successful, but it’s not a guarantee, as you can see from many failed open source projects by both Facebook and Google as an example.

ReactJS base on components. A component is one isolated piece of interface. For example, in a typical blog homepage you might find the Sidebar component, and the Blog Posts List component. They are in turn composed by components themselves, so you could have a list of Blog post components, each for every blog post, and each with its own peculiar properties.



**Figure 13** A React component

# Chapter 4: Application development and deployment



## Architecture design



### Choose software architecture

#### MVC Model

Data processing is a complex matter, as data needs to be processed and stored in the most basic way possible so that it can be converted to a variety of formats later, depends on the needs of the user interface.

Therefore, processing and saving of data must be broken down into sections, each has its own role, and in general all systems will consist of three main components: model which is object and data, view as user interface, controller in charge of data processor and controller.

This model can clearly distinguish the role of each component in the system, so we will apply the MVC for the current system.

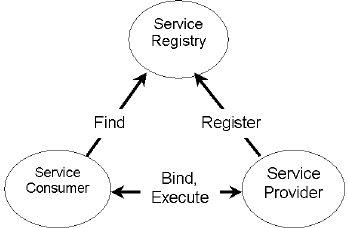


**Figure 14** MVC Model

We will divide the system into 3 parts: (i) Model: includes Entities, data and classes that interact with the data; (ii) View: We will assume that the client side is the view, and within that, there are smaller MVC models. (iii) Controller: includes logical processing and mapping elements of requests from users.

#### Service-oriented architecture (SOA)

To minimize the dependency between layers in the MVC model, we will use the Service-Oriented Architecture



**Figure 15** Service-Oriented Architecture

With this architecture the system will be divided into different services, each will be registered with a "service registry". When any layer needs a service, it just needs to look in the "service registry" from there and map to the service it needs. Services will be initialized in the system and called beans. The classes will use the Dependency Injection mechanism to invoke the beans that they need.

This architecture guarantees the ability to divide the workflow in the MVC model. On top of that, it can reduce the dependency between layers in the MVC model through Dependency Injection. Also, a request to change or expand can easily be fulfilled by adding services to the system without affecting the other layers. There is a "service registry" that separates layers, not letting them interfere with each other.

For this problem, we will use an open source framework that represents the Service-Oriented Architecture: Spring Boot of java language.

#### Combine two architecture

In this project, we intent to use both of these architectures. The Service-Oriented Architecture has already applied to back-end server using Spring Boot framework. And, we applied MVC Model for the whole project.

As mentioned earlier, there are two servers assigned to do two differences purposes which are providing user interface for user and handling all the data management tasks in 1.3 section.

Therefore, the component View in MVC model is the front-end server. The component Controller and Model are implemented in the back-end server. These two servers connect by the same pre-defined API.

The front-end server is coded in ReactJS and using npm as a package manager. Users interact only with this server. All user interface and pre-validate data logic are implemented here. Any responses and requests user made belong to this server.

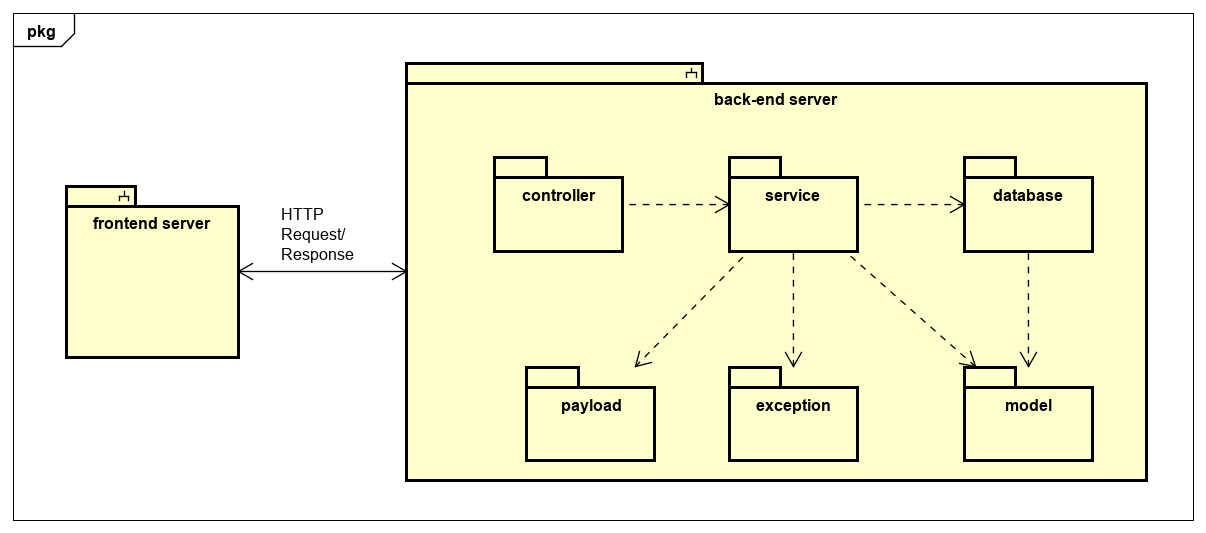
If one server ever is coming into a maintenance or off-line by subjective or objective causes, it will never affect the second server. By this design, both servers are safe when another gets problems.

In back end server, Model components are the database package and model package. The database package is in charge of database connection and model linking. Whilst, model package contains all the class define data models.

In additional, Control components are controller and services packages. Classes in controller package define end point for the server. With each URL endpoint, controller return a service function for specific meaning end point provide. After the function run, it will return a suitable response in order to send to View display.

So, after this section, we understand the way we apply MVC into our system.

### Overview design

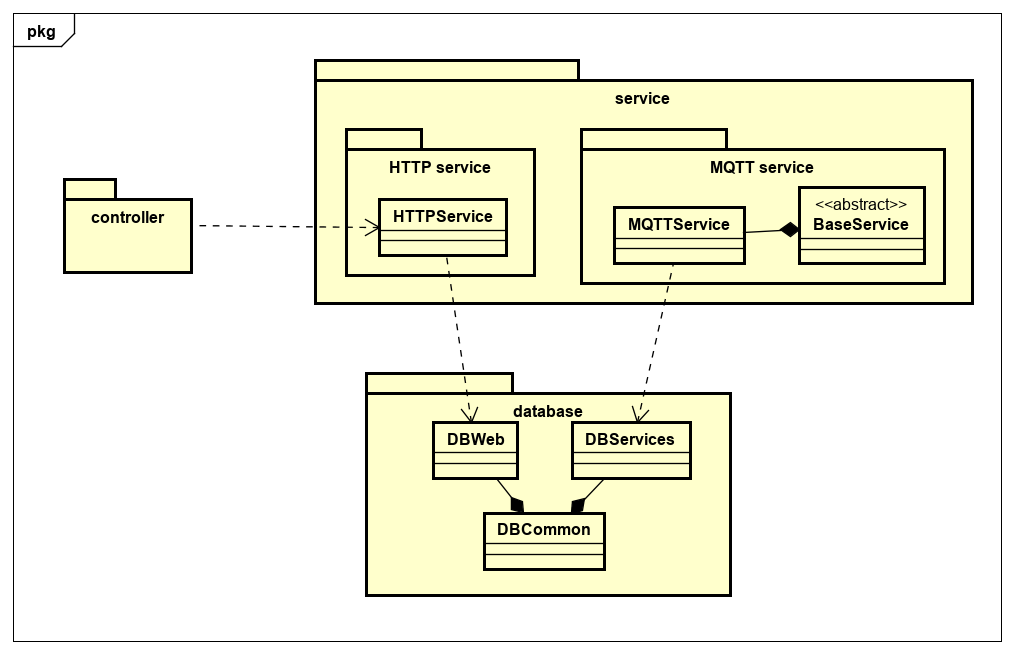


**Figure 16** UML Package diagram

As you see in the figure above, in the back-end server, there are six main packages: controller, service, database, payload, exception, model. Each package contains classes with difference purpose. The controller, as mentioned in 4.1.1.3, is in charge of defining the endpoint for backend server and calling corresponding service function for each endpoint. Whilst, the service package includes the classes providing logic and request handling function. The database package has the classes which connect to the database and act like an intermediate between the server and database. The model package holds the class model of object that provide an interface with the table in the database. Payloads are classes define request and response model which can be understand as the interface between frontend and backend server. In the other hand, the exception package was born to contain all the throwable exceptions the requests made.

### Package design specification

The main group of packages in the backend server control all the request and logic function:



**Figure 17** Backend Server Package Design Specification

In this Figure, service package is divided into two packages. They serve difference flow of the server. HTTP services call by controller in order to response HTTP request called to server. MQTT services have a BaseSevice, which is abstract class, extended by other classes. The purpose of MQTTSevice class is subscribe to MQTT defined topic and handle all the published message to that topic. Furthermore, the services interact with corresponding database class which extend DBCommon. For each database class only is in charge of one kind of return to serve MQTT service and web API.

### APIs design specification

Here, I will list all the APIs the system needs and uses:

**Table 11** APIs list for back-end and front-end

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Purpose** |
| GET | /api/user/checkUsernameAvailability | Check Username Availability |
| GET | /api/user/checkEmailAvailability | Check Email Availability |
| POST | /api/auth/signup | Sign up |
| POST | /api/auth/signin | Sign in |
| GET | /api/user/me | Get user details |
| GET | /api/crop | Get all crops |
| GET | /api/crop/{cropId} | Get crop details |
| POST | /api/crop | Create crop |
| DELETE | /api/crop | Delete crop |
| POST | /api/crop/stop | Strop crop |
| GET | /api/device | Get all devices |
| GET | /api/device/{deviceId} | Get device details |
| POST | /api/device/control | Control device |
| GET | /api/plant | Get all plant type |
| GET | /api/sensor | Get all sensor data |

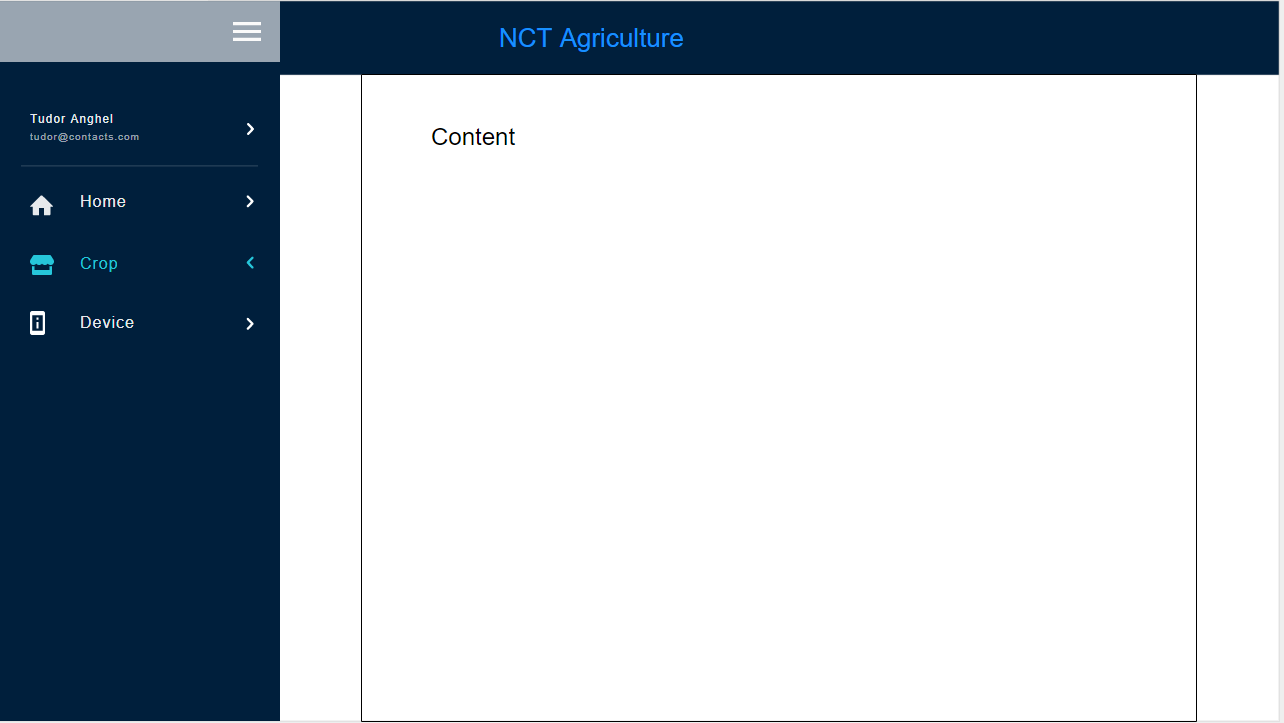
As you can see in the table, there are some noticeable APIs. The “/api/crop” API was born to serve all the requests relate to crop. The function to create a crop is done by calling a POST API which have the body is a JSON object with parameters: (i) name, (ii) deviceId, (iii) plantId. Name of the crop, device id and plant id are mandatory. To control a device, the client call POST API: “/api/device/control”. This API also needs four mandatory properties for the device to know what command to make in the body: (i) deviceId, (ii) actuatorName, (iii) action, (iv) param.

## Design specification



### Interface design

The User Interface of the application is developed on ReactJS and some other libraries to help building various components. We aim to a responsive application and can be show on both smartphone and web. Therefore, the screen resolution is not fix and be flexible. The background should be white and have the header and sidebar as follows.

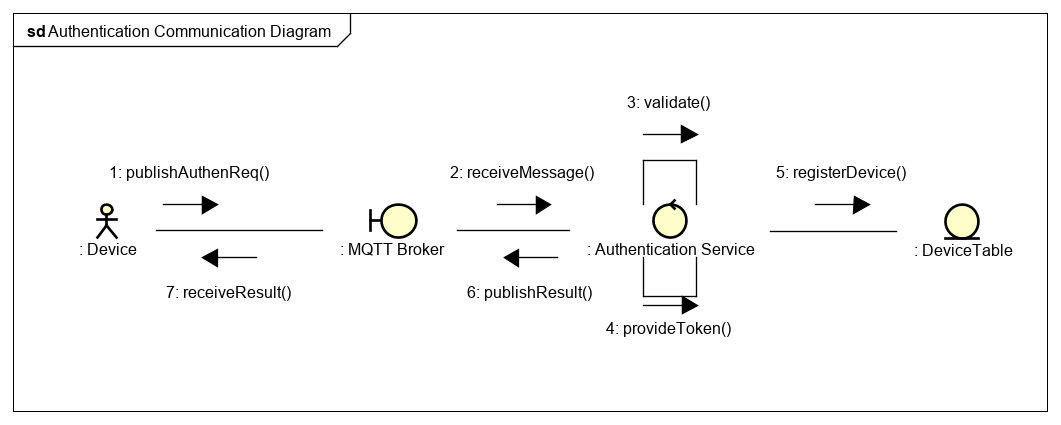


**Figure 18** Main Page Mockup

### Class design

In the backend system, processing services play as the most important part in the system. Therefore, our system has four service class in MQTT package which is most influence in the workflow: (i) Authentication Service (ii) Collect service (iii) Keep Alive Service (iv) Control Service. This is one of the feature solutions for this project. Now, I will introduce the class design of these class. Later, I would like to put the details design of these classes to be presented later in chapter 5 feature solutions and contributions.

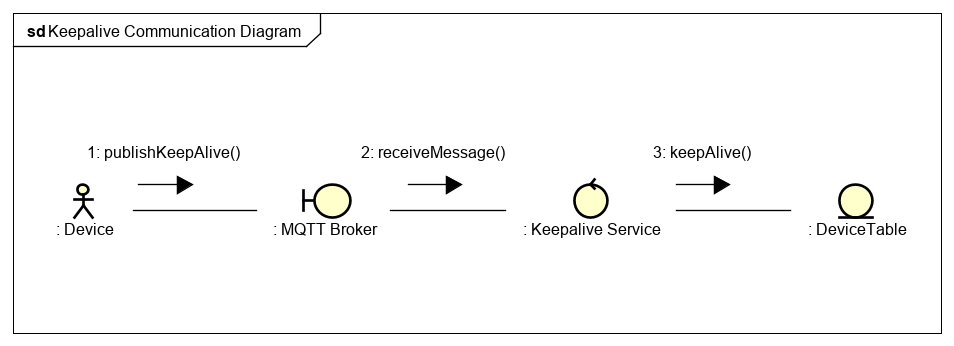
#### Authentication Service Class



**Figure 19** Authentication Communication Diagram

This class was mean to be an authentication method for device. After manufacturing, the device at first was a blank device which does not belong to any users. Therefore, the class was born to register device to specify user for the system.

#### Keep Alive Service Class



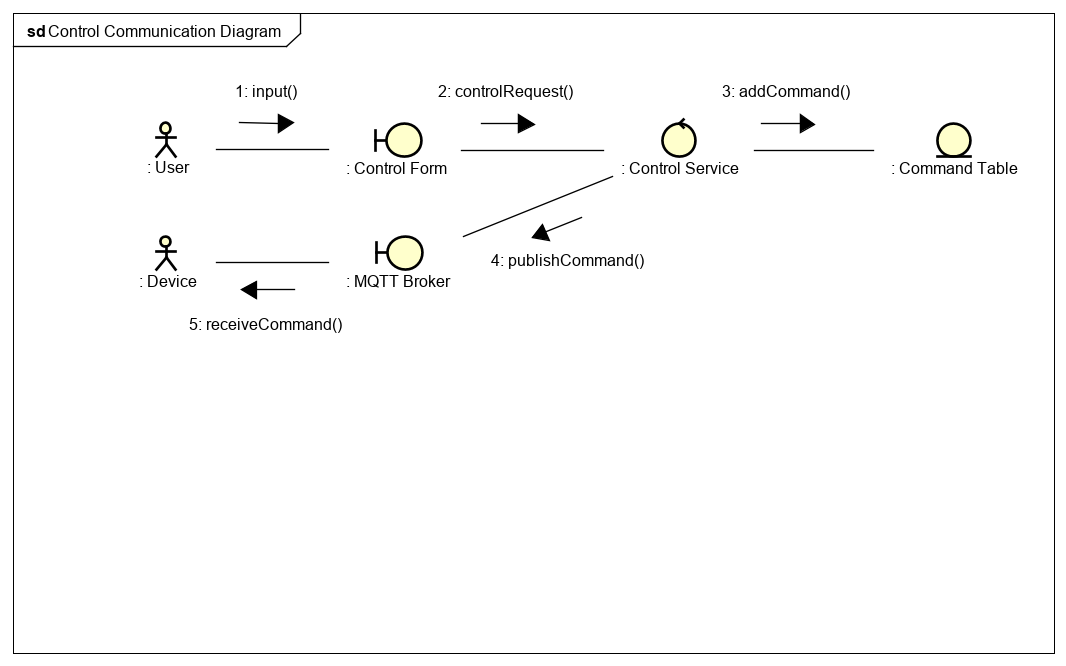
**Figure 20** Keep Alive Communication Diagram

After starting crop, the device begins to collect. Though, the connection between server and device are sometime not stable because of poor connection or out of power. As the results, we create the keep alive service in order to tracking of state of device.

#### Collect Service Class

Similar to keep alive, the collect task of device starts after starting crop. the device collects the data from its sensor, wrap into a message and publish to MQTT topic. Then, the server receives the message from MQTT, read the data and add them to database.

#### Control Service Class



**Figure 21** Control Communication Diagram

Last but not least, in the figure above is how control service class work. The service makes sure all the control from user as well as automate system to arrive to the device correctly.

### Database design

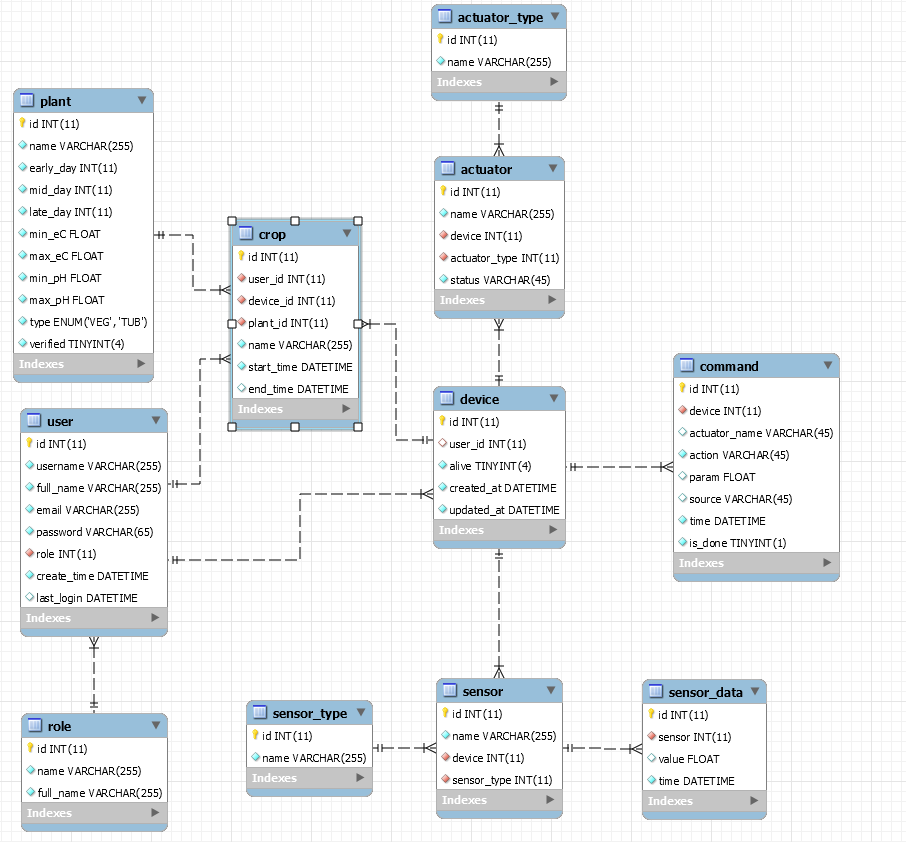
Based on the requirements, we can see that the main object of this problem is the hydroponic crops.

According to the general design of the system, each hydroponic Crop will contain one control devices with two tasks: collect and control. In addition, only one plant can be planted on each crop, because each crop only uses a certain nutrient solution that may not be suitable for multiple type of plants. However, each crop is monitored by a person and one person can monitor multiple crops so that would not be the back step.

Each tree has different growing conditions at different stages. What the current system does cannot do is automatically processing each tree differently from time to time the growth of that plant. So, we need a way to store plant data so that the system can understand the limits of the tree from time to time.

When we harvest a crop, the crop is completed, the end time recorded and stored in the database for logging purposes. Meanwhile, other crops will still exist and if it already reaches the time to harvest, there should be a way to notify user about that.

From the above analysis, we will design the database of this problem as follow:



**Figure 22** Database of a Hydroponic IoT system

The system will include users. User account information will be stored in the “user” table. A user will belong to a certain ROLE. The information about the ROLE will be stored in the “role” table.

Each “crop” will be made up of one “plant”, one "device" and managed by one “users”. Old and new frames can be distinguished through the start time and end time (harvested / un-harvested). The crop harvestable is defined by the minus of the time now to the start time.

The “device” table will store information about existing equipment and alive status. Moreover, each device can be register to one user at a time and can be re-register to another.

The “actuator” table is defined to be actuators for the device. Each actuator here is one of “actuator\_type” and belong to one device. With this design, the device can have the flexible amount of device and type of its actuators. Also, whenever we want to add a new actuator, we just need to add a row in the “actuator” table.

The “sensor” table has the same design as the “actuator” table which have the table “actuator\_type” now become “sensor\_type”. Each sensor in this table belong to one device.

The “sensor\_data” table will be used to collect received data through time. Every row in this table contain the “value” for the value of sensor measured and a sensor which the data belong to.

The “command” table will store control commands for the system.

To ensure that the system can accurately meet the needs of various plants in different phases, the table “plant” table that stores detailed information about the plant, including development time, plant parameters... through each growing period of the plant.

The days field will store the development time needed for every plant type. Its information will be used by the automation services to control the crop ‘s device to active device’s actuators in order to change the environment which best suit for the crop in each development state. Other fields such as EC and pH, etc. will be stored as float: min value, max value also serve the same purpose.

There are a “verified” value in the “plant” table. It is defined to help user can have add and product new kind of plant here. However, in the scope of this thesis we do not implement it yet. But in the future, when we can develop the machine learning to verify the plant data. Then we can apply it so that user can add their own plant.

With the database developed as such, we can meet the previously mentioned requirements.

## Application implementation



### Library and framework

List of libraries and tools:

**Table 12** Libraries and tools

|  |  |  |
| --- | --- | --- |
| **Purpose** | **Tool** | **URL link** |
| Programming IDE | IntelliJ IDEA 2019.1.2 | https://www.jetbrains.com/idea |
| Web Framework | Spring Boot 2.1.5 | https://spring.io/projects/spring-boot |
| Java Development Kit | Java SE Development Kit 10 | https://www.oracle.com/java/ |
| Database Management | MySQL Community Server Win64 | https://dev.mysql.com/ |
| Visual database design tool | MySQL Workbench Community 8.0 | https://www.mysql.com/products/workbench/ |
| JavaScript runtime built | NodeJS 10.15.3 (includes npm 6.4.1) | https://nodejs.org/ |
| React UI library | Ant Design v3.18.2 | https://ant.design/ | |
| React charting library | Recharts 1.6.0 | http://recharts.org/ |
| API development environment | Postman | https://www.getpostman.com/ |
| MQTT testing tool | MQTTBox | http://workswithweb.com/mqttbox.html |

### Achievement results

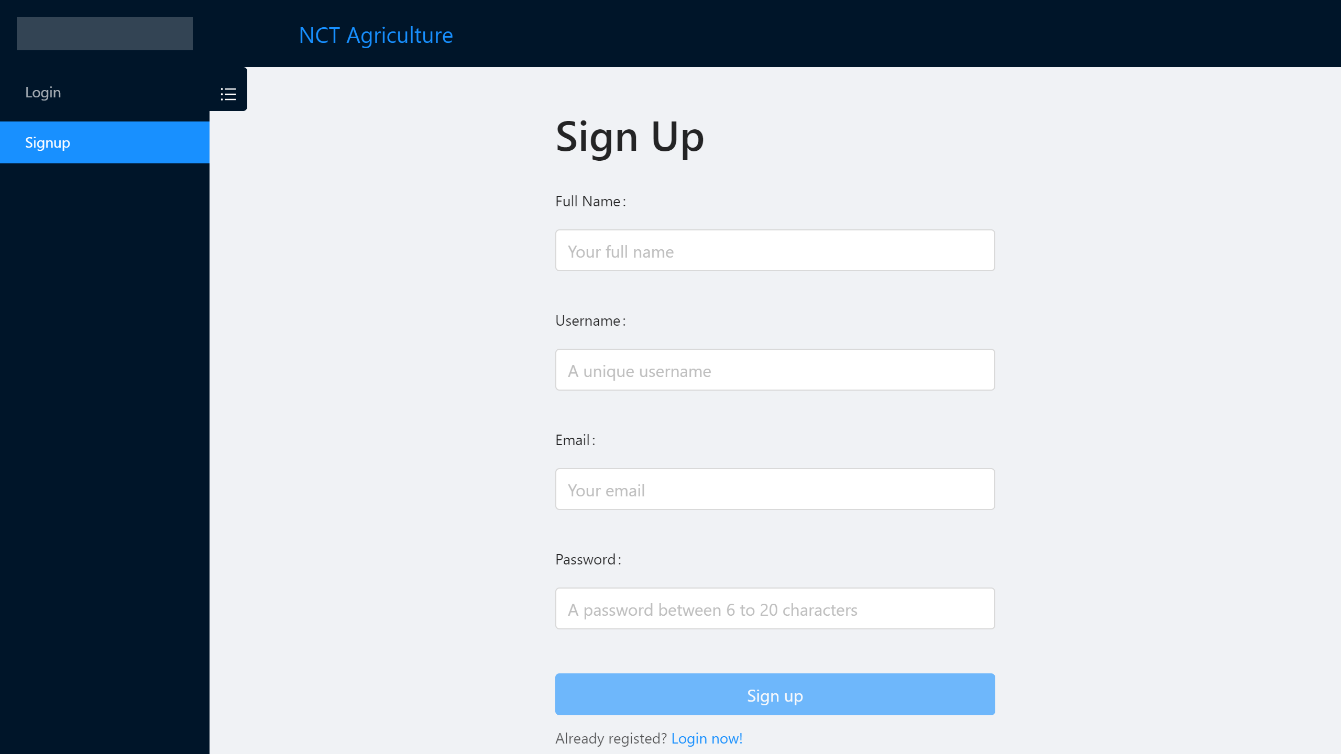
After implementation the achievement result of this thesis is two servers working well together that help the purpose of creating the backbone system for hydroponics indoor cultivation system. The system has reached the giving requirement from the start. The automated system base on predefine data to automatic control indoor hydroponics crop, especially oriented for greenhouse, also household farmer.

**Table 13** Project statistics table

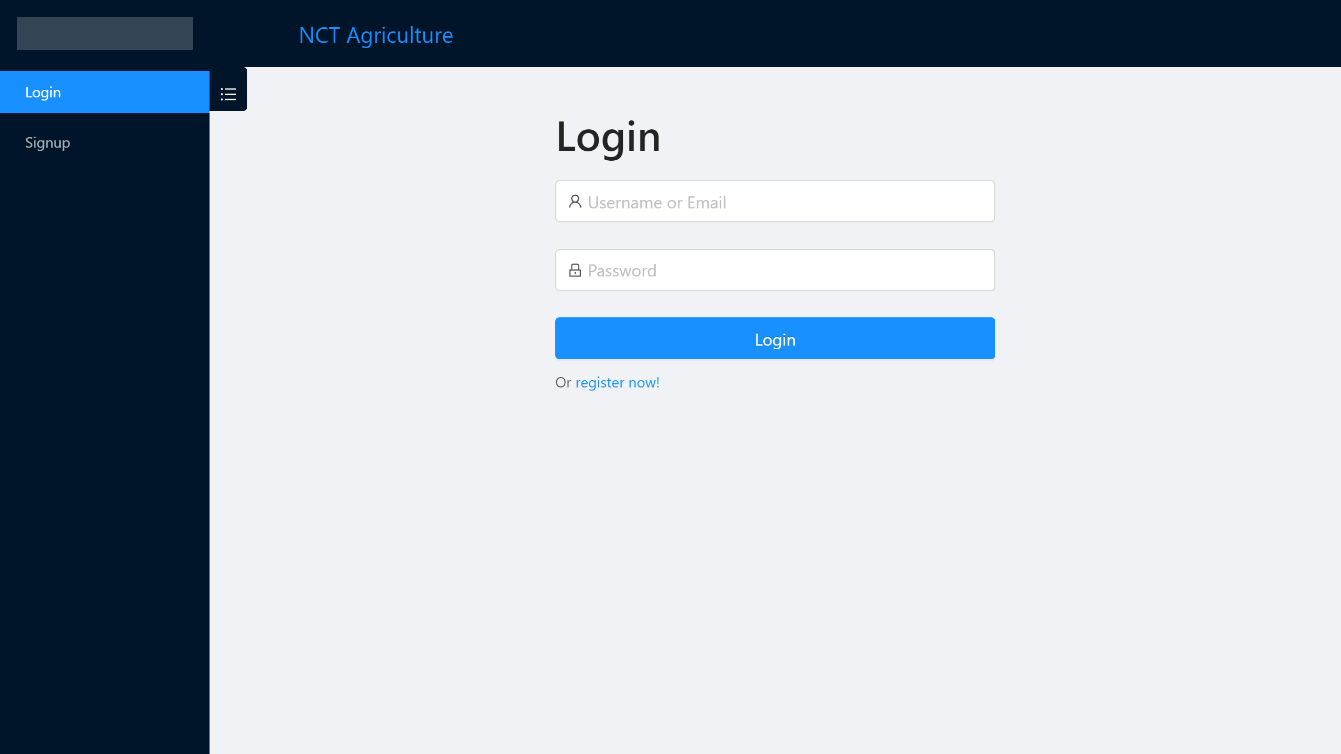
|  |  |  |
| --- | --- | --- |
| **Statistics** | **Backend server** | **Frontend server** |
| Language | Java | Reactjs |
| Lines of code | 3469 | 3215 |
| Numbers of class | 60 | 24 |
| Size of code | 748 KB Files | 7.2 MB Files |

### Main screens and features

Authentication function User Interface looks like follows:

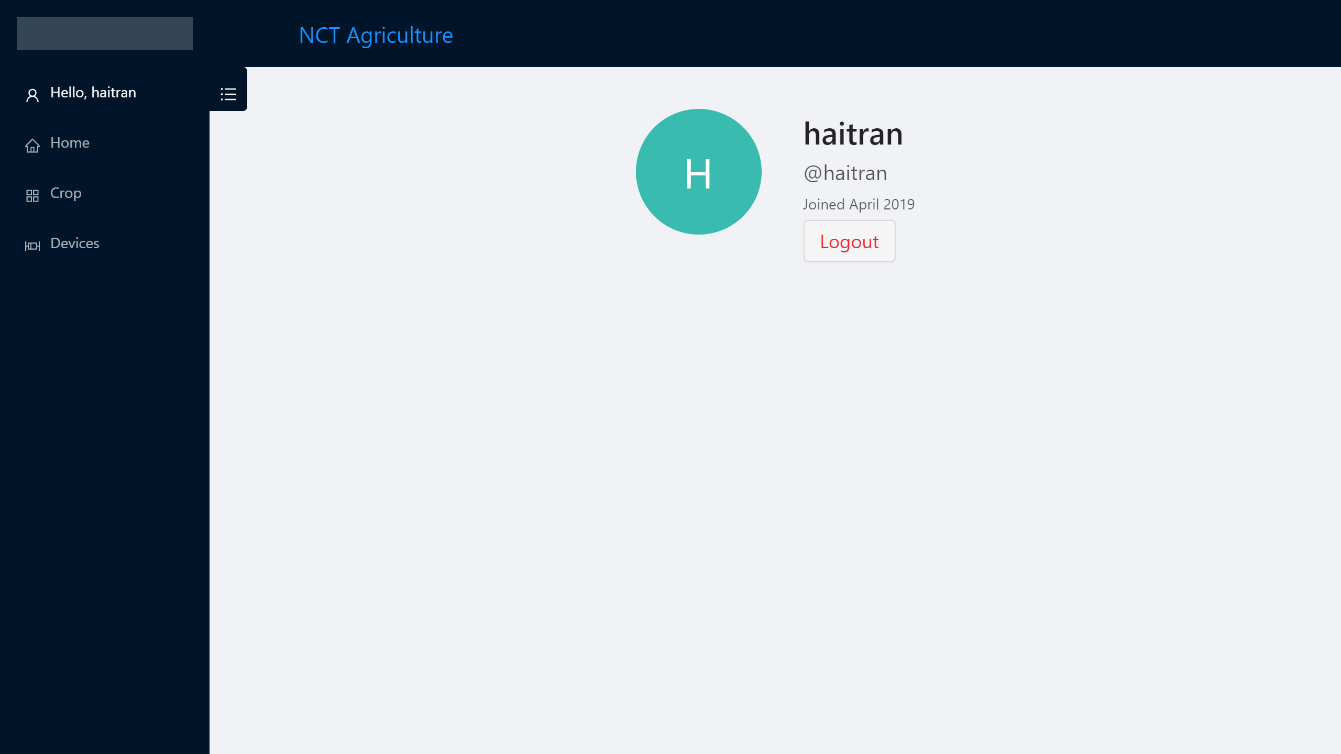


**Figure 23** Sign up User Interface



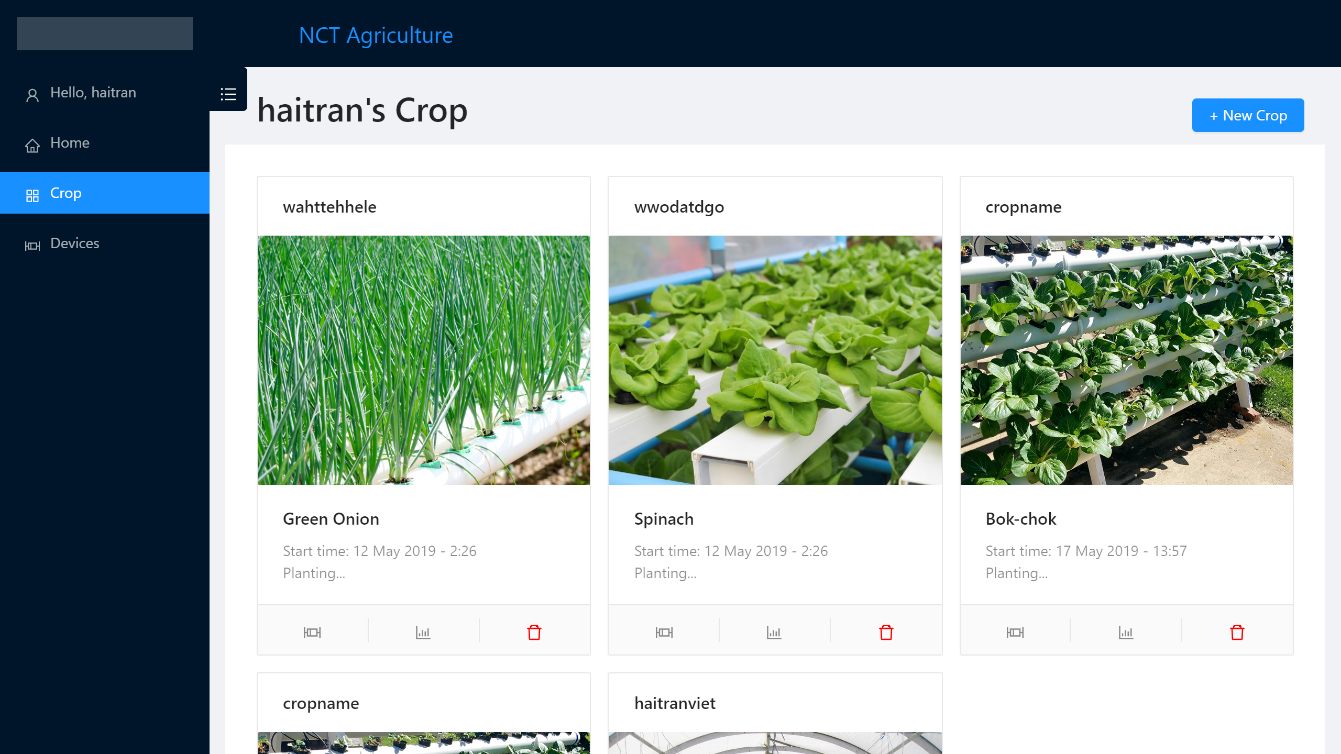
**Figure 24** Login User Interface

After signed in, users can see their profile page like this:



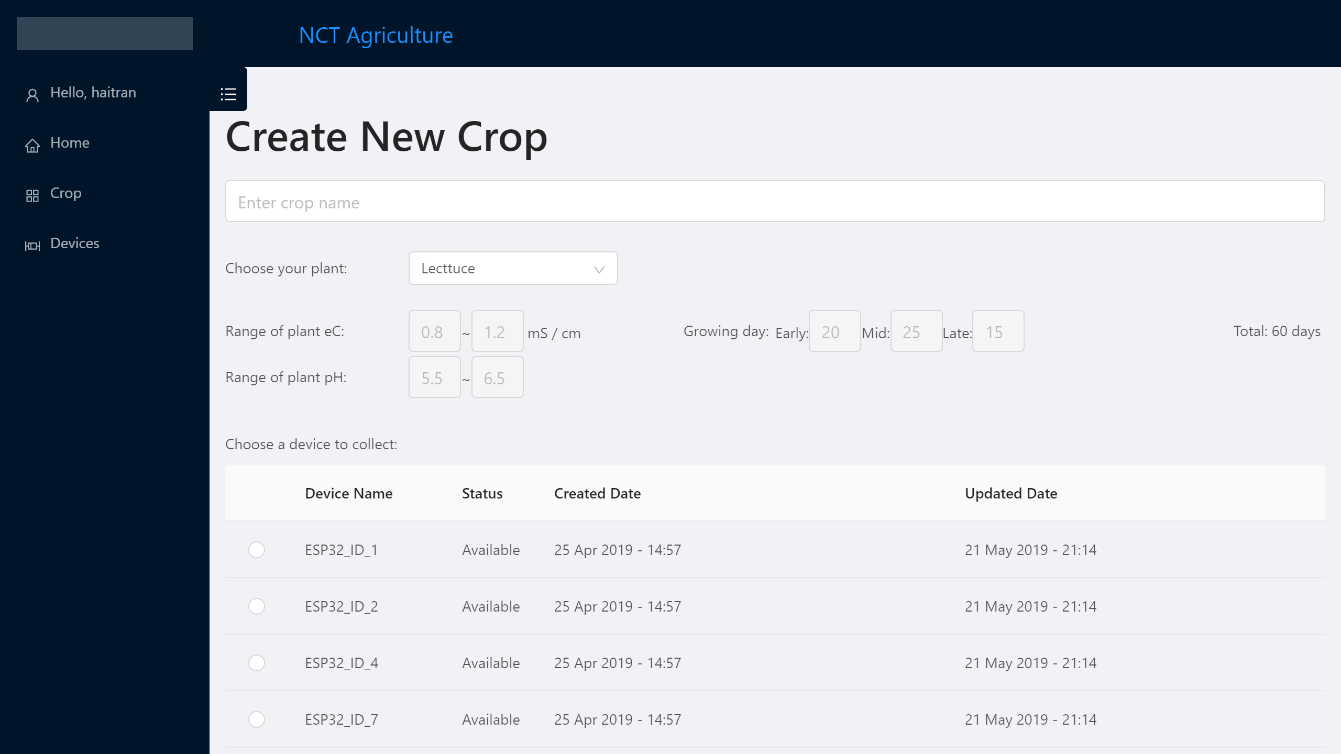
**Figure 25** User Profile Interface

User can find their crop list in the Crop tab:



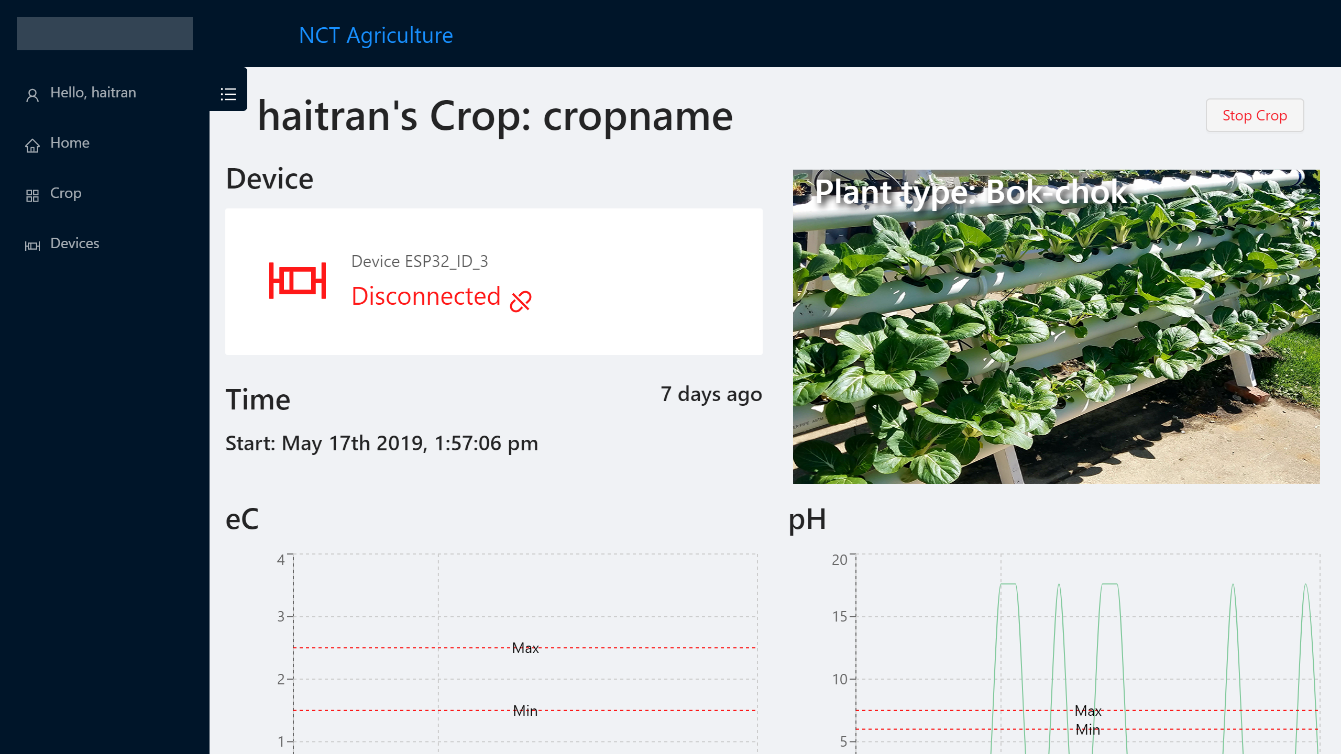
**Figure 26** List crop User Interface

There are three buttons under each crop card corresponding to the following function: show device details, show crop statistics and delete crop. User always can add more crop to their farm by the New Crop button on top of the page.



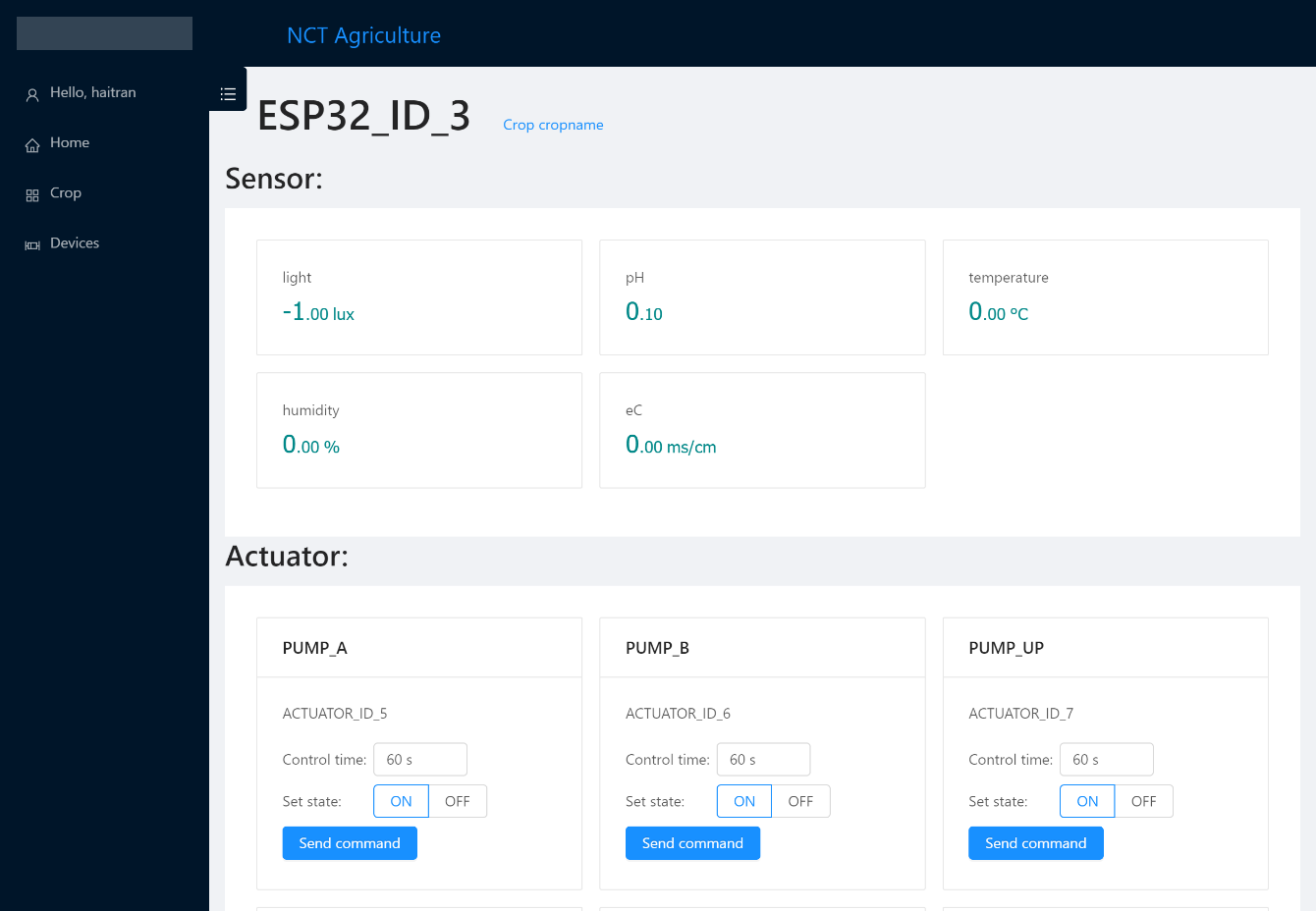
**Figure 27** Create new Crop User Interface

This is where user can add new crop. The form contains of plant preset data. The data change when choosing another plant. All the data are being disabled because customize function have not implemented.



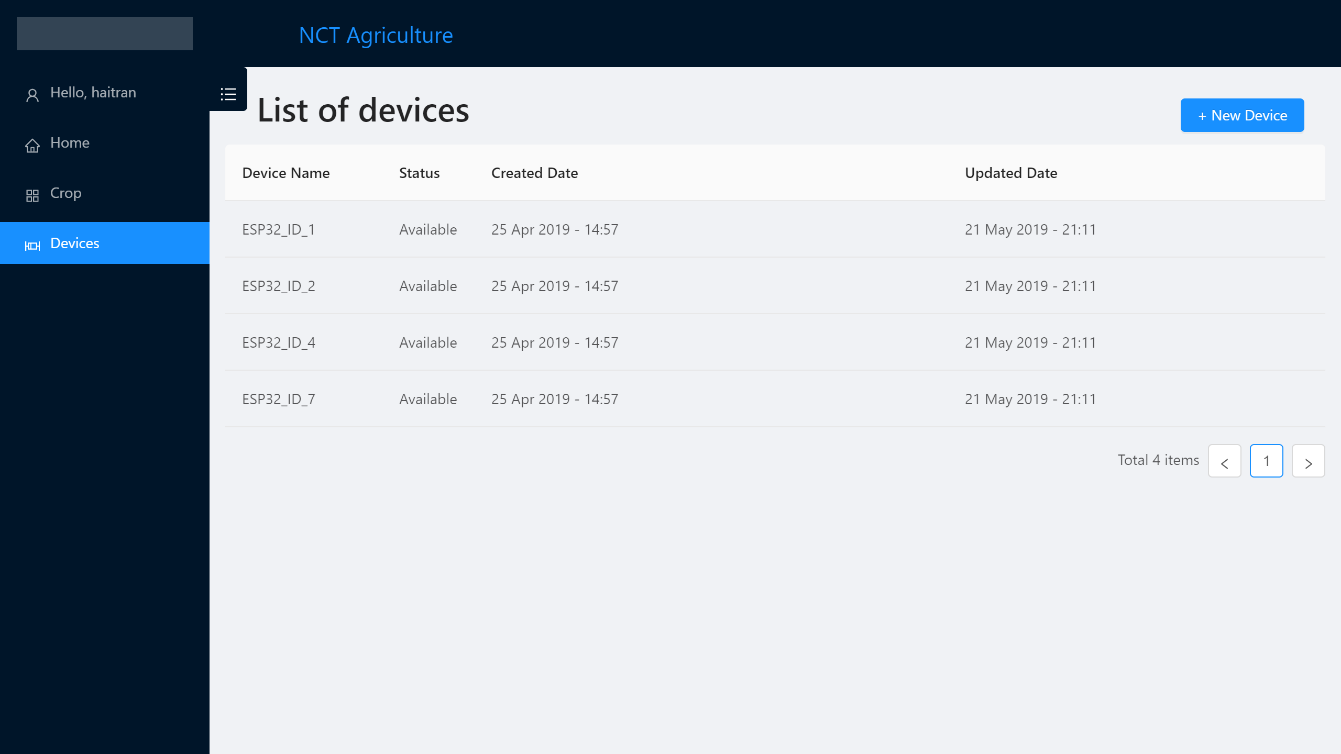
**Figure 28** Crop Details User Interface

The crop detail screen is the screen show data collected from the crop.



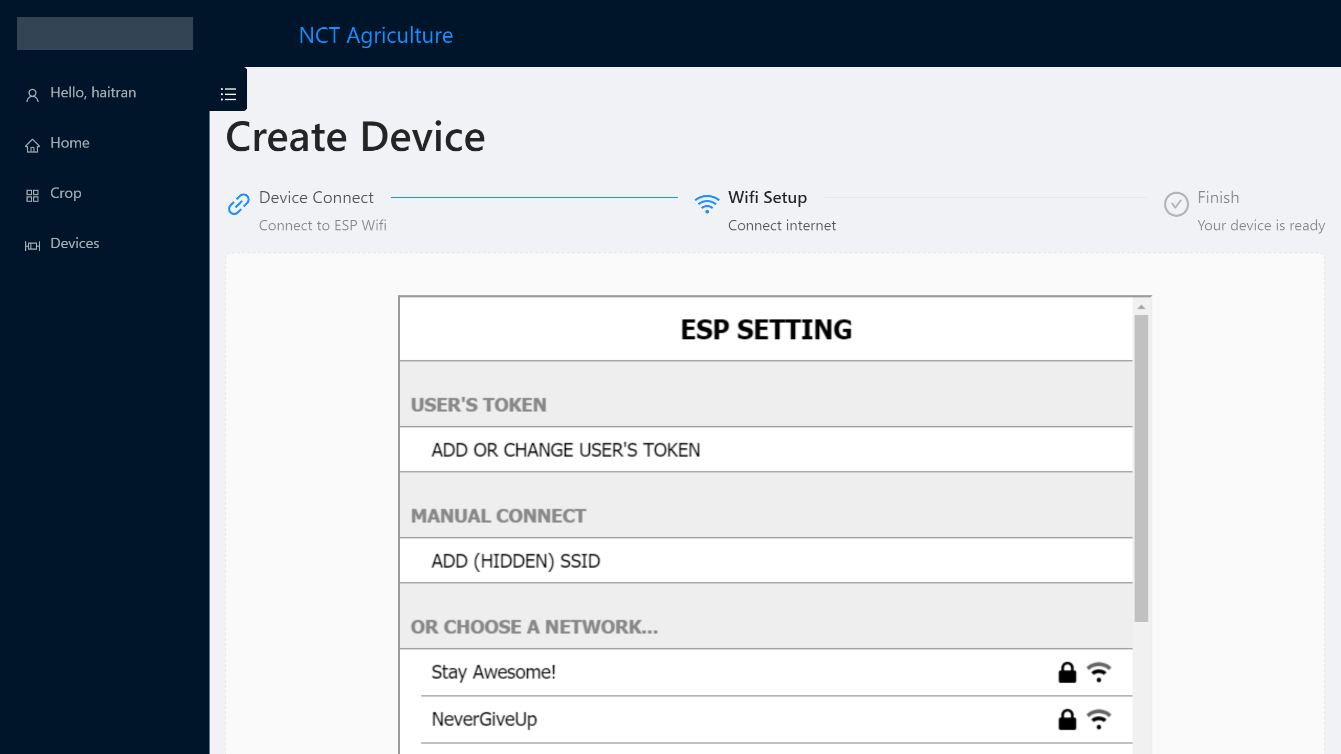
**Figure 29** Device Details User Interface

Device detail page will list the sensors and actuators currently plugin to device. User can control those actuators here.



**Figure 31** List of Devices User Interface

The device list belongs to user.



**Figure 32** Register Device User Interface

Create Device screen includes an instruction for user step by step to register their new device.

## Deployment



### System setup

#### Backend server

Prerequisite for this project is installed IntelliJ IDEA, MySQL Server, JDK 10, and have a basic knowledge of these technologies.

First of all, create MySQL database:

1. create database ivofarm

Then, import the database from *src/main/resources/database.sql* file.

Next, change MySQL username and password as per your MySQL installation by

open *src/main/com/agriculture/nct/util/AppConstants.java* file. Change these properties as per your MySQL installation.

1. public static final String MYSQL\_USERNAME = "ivofarm";
2. public static final String MYSQL\_PASSWORD = "ivofarm";

Finally, run the spring boot application by using the IDE to setup your own environment or typing the following command. However, using the command, you need Maven installed first.

1. mvn spring-boot:run

After that, the server will start on port 8080.

#### Frontend server

In the project directory, you can run:

1. npm start

This command runs the app in the development mode. Open http://localhost:3000 to view it in the browser.

For production, run the command:

1. npm run build

This command will build the app for production to the build folder. It correctly bundles React in production mode and optimizes the build for the best performance. The build is minified, and the filenames include the hashes. Then, the app is ready to be deployed!

### System evaluation

Compared with the general criteria of a hydroponic IoT application system described in the requirements, the system has and has not met the criteria.

First, the one of the noticeable met requirements is low cost. The average cost for a manual hydroponic system is about 4 million VND (including truss, basic tools and solutions). Whilst, the average cost of a high-tech hydroponic cultivation truss available in Vietnam will be around 20 million VND, not including operating costs (including truss, basic tools, solution, sensor circuit and phone application). For this system, the total cost of equipment is about 6 million VND. Thus, compared to current high-tech hydroponic systems, it is possible to reduce investment fund by a half.

Secondly, the system is scalable for multiple devices. We can use a lot of different types of sensor circuits on the IoT equipment such as ESP32, ESP8286, Arduino, etc. The only condition for any device that wants to run well in this system is to send the correct order of the messages and content of the messages. Easy add or delete device in the system.

Thirdly, the application of high technology for farmers, who do not have much experience in hydroponic, can still operate well. How the system work make farmer does not need to continuously take care of the plant. Automated control based on existing data to help users do not need to have the knowledge about the plant.

However, there are some requirements not met. High yield comes with high quality. Due to the lack of facilities, the system cannot be put into actual operation to evaluate the specific quality of the product.

In general, the system has not met all the criteria, but the system has met most of the requirements of an IoT aquaculture system. Accordingly, we can clearly see the advantages and disadvantages of the system.

The system is designed quite completely from the gateway to the client. Rapid data acquisition and response system with changing conditions based on the ability to control the device through the MQTT protocol. Also, it has a high ability to apply to real life because of the good scalability.

Nevertheless, the system still has many errors and many cases have not calculated all possible exceptions when applied to real life. There is no ability to track individual parameters with different monitoring cycles, need to be upgraded to this feature.

# Chapter 5: Feature solutions and contributions

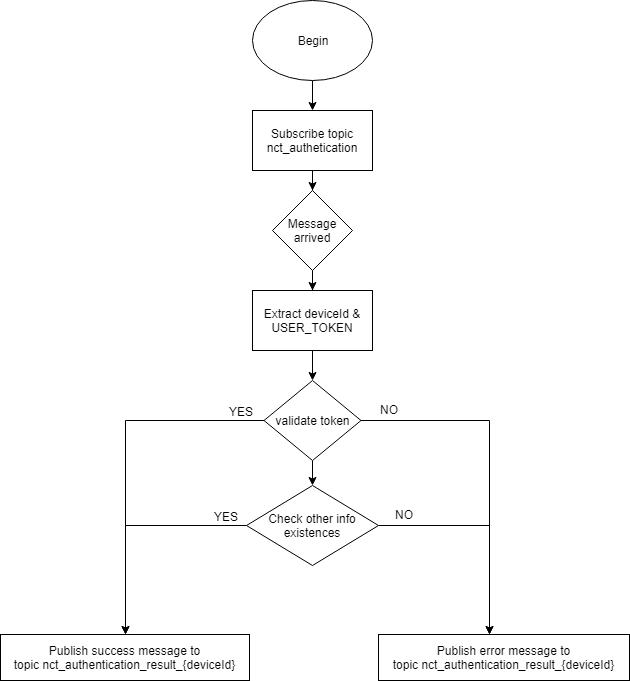


## Authentication service

As state in 4.3.2.1 section, the authentication service will register the new device to a specific user accessing the system.

There are two actors accessing the system, which are the user and the device. User authentication is a popular feature, we do not go too deep into it in the scope of this project. However, I will brief overview how the user authentication work. The system uses JWT method to provide each user login to the system a token which have an expired date (by default I set it to 7 days). We call this USER\_TOKEN which is a requirement to call other APIs in the system.

The authentication sequence for a device will be as follows:



**Figure 33** The device authentication sequence in the system

The implementation sequence is described as follows:

Here, the server and the devices communicate with each other through the MQTT protocol. To receive the authentication information of the devices, the server subscribed a predefined topic for authentication called “nct\_authentication”.

We predefine authentication messages of devices in the json form:

1. {
2. "user\_token": "abcxyz123456",
3. "id": 1
4. }

Where:

id: identification of the device was previously granted and has been stored in the database

user\_token: The token that taken when user inputting to connect ESP32 to users’ Wi-Fi. This token contains user authorization so the system can understand who send the request and make security better.

Once the authentication message is published, we check that if authentication fails and the id of that device exists. We will publish the error authentication message of that device to the authentication result topic.

The error message on the corresponding topic of the device is nct\_authentication\_result\_{id}.

If the authentication is successful, we also publish a successful authentication message to the topic nct\_authentication\_result\_{id}.

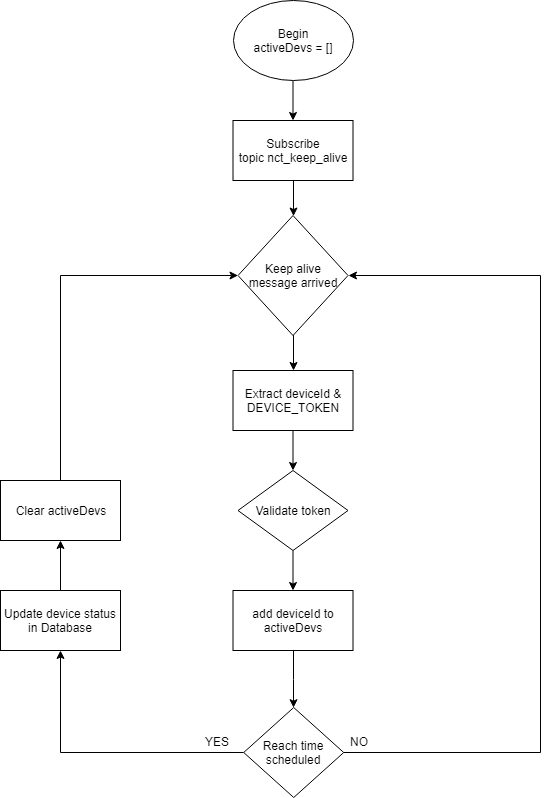
After receiving the successful result, the device changes its state to available and ready to start crop at any time.

## Keep Alive service

As mentioned in section 4.3.2.2, the keep\_alive service is set up to determine if a device is still connected to the server or not. When a device lost connection to the server for some reason, its state will be changed by the server and considered as no longer accessible and associable.

As you know so far, the system uses JWT method to provide each user login to the system a token. However, the system also generates another token called DEVICE\_TOKEN. After the user send a create crop request to the server, the server will process this request and publish a start crop command with this token generated with crop ID, user ID and device ID which is a requirement for device to publish keep alive or collect package to the system.

The sequence of keep\_alive is shown in the following figure:



**Figure 34** The keep alive sequence in the system

First, we create an empty list (LIST), which will contain the ID of the devices sending the keep\_alive message to the server for a period.

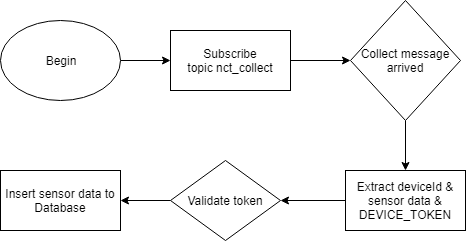
Here we set the default test time is one-minute. Devices outside the farm are programmed to send every one-minute a keep\_alive message containing their ID to the nct\_keep\_alive topic.

Each device sending the keep\_alive message will be parsed out and added to the list above. Of course, the logic is not adding new device ID if the ID has already existed in the list.

After one-minute, the first thing to do is replicate activeDevices list to other one, and clear activeDevices. Then, we use a SQL statement specify for this job to change the alive status in the database. This SQL statement only changes the device that have changed their status lately and does not touch other devices. Therefore, the optimal of the statement is very good and effective. After the updating, we will continue from the beginning.

## Collect service

The collect service may not be complicate as keep alive and authentication service. However, as we stated previously in section 4.3.2.3, collect service is one of the most important service in the system.



**Figure 35** The collect sequence in the system

The device sends data package through MQTT protocol. To receive the collect package of the devices, the server subscribed to a predefined topic for collecting data called “nct\_collect”.

A package contains collected data look like this:

1. {
2. "id":1,
3. "device\_token":"abcxyz123456",
4. "timestamp": "timestamp",
5. "packet\_no": 123,
6. "temperature":30.1,
7. "humidity":52.4,
8. "EC":0.2,
9. "pH":1.5,
10. "water\_level":100
11. }

Every time the server receives a package from the topic “nct\_collect”. It will start to extract the data from it. Then, the server will validate the token the package provides. If successfully validated, the data collected will be added to database.

## Control service

Control service is the simplest service. The purpose of this service is mentioned in section 4.3.2.4. Every command that sent to devices must go through this service. The service will keep the command in the right model so that the device can understand and execute correctly.

# Chapter 6: Conclusions and orientations



## Conclusions

The above is the project that I made. Due to limited time as well as lack of experience in exploring this field, the shortcomings cannot be avoided. Here are the points we have and have not done when implementing this project.

After completing the project, the project explored the systems of Internet of Things and hydroponics system works. From that, we built a managing system for hydroponics in door and apply Internet of Things for controlling and monitoring. The project also provides models and tools to solve the problem. After the implementation process, we have built a system to solve some of the initial problem.

During project implementation, I learned the knowledge of building systems with Model-View-Controller architecture; use of MySQL databases; as well as working with services, get to know HTTP protocol and successfully apply lightweight Internet of Things protocol like MQTT protocol. Moreover, I learn and understand how a system work, design from back to front whole system communication and workflow. Especially, I built a server for receiving and processing data; Control the equipment on the frame, built a web app for users to monitor plant parameters and controls them.

However, due to many subjective and objective factors such as the limited time and human errors, there are still many points to be improved. For example, plant production testing is not paid attention to; The design interface is still not easy to follow, the user experience is poor.

ReactJS is a new JavaScript Framework and stick to the basic is inevitable. However, there is many better new ways of implementation are not aware as Redux for state management. Our implementation does not have that yet.

Also, because of time limitation, the testing process have not done properly. Therefore, implemented code may secretly contain some uneasy to trace bugs. That make the quality of the system has not been evaluated yet.

## Orientations

Over the implementation process, I have proposed a number of solutions to improve the system so that it can be put into practical use.

In the future, we may put the system into trial production version.

All the service, which I have already prepared by implemented by the most untangling design in development, can be decompose to multiple microservices. Applying the microservices model on services will be the future of this project. If so, the design will be clearer and easier to adapt to future scaling.

One of other options is Build-in photo capture and analysis to track development of the plant and pests overseeing. The system collect image from user and notify if the plant starting to have any signal of plant diseases.

Also, we can take the machine learning to upgrade the plant modules. This allow user can add their own plant. After that, the system can apply machine learning to verify effective of new plant. Consequently, user can choose new plant with the best solution have been approved.

Furthermore, the system has not had implement MQTT system separately. If so, the system will be more independent. Besides, we can control how the traffic flow into the system. So, we can optimize system capacity and workload.

As I mention before about the ReactJS library, we can implement Redux for managing application state.

The functionality of the current system is simple, lacking many necessary tools. After adding some of the above functions, I believe the system can completely deploy and work in the production schema.

# REFERENCES

1. Moises Macero, Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber, Apress, December 2017
2. L. C. Antonio, V. Alvaro, B. Atonie, Z. Marco and Ermanno, IoT in five Days, E-book, 2016
3. Peterson L. L. and Davie B. S., Computer Networks: A Systems Approach, 2nd ed., Mogran-Kaufmann, 1999.
4. V. Palande, A. Zaheer, and K. George, “Fully Automated Hydroponic System for Indoor Plant Growth,” Procedia Computer Science, vol. 129, pp. 482–488, 2018.
5. E. Bakhtarov, “Handling of iframes in React,” Emir Bakhtarov, 15-Dec-2016.
6. “Spring Boot + Spring Security + JWT + MySQL + React Full Stack Polling App - Part 1,” CalliCoder, 05-Feb-2018. [Online]. Available: https://www.callicoder.com/spring-boot-spring-security-jwt-mysql-react-app-part-1/. [Accessed: 24-May-2019].
7. P. Dayaram, “Using Chart.js with React,” Pradeep Dayaram, 15-Aug-2017.
8. Berners-Lee T., Hypertext Transfer Protocol (HTTP), CERN, <URL:ftp:/info.cern.ch/pub/www/doc/http-spec.txt.Z>, last visited May 2019.