Smart Shoebox
(Shoes care solution utilizing IoT concept)

Ko Byunghee, Kwon Gyuhyeok, Kim Junghyun, Shin Minki
Information System Department
College of Engineering
Hanyang University
Seoul, South Korea

***Abstract*—This document is about the realization of automatic remote control for shoesthrough IoT. We will make smart shoes cabinet that provides this kind of features with other different kind of functions. Smart function to manage the shoes to be neat and pleasant to wear, recommendation function to recommend proper shoes for the user, analyzing the shoes information to classify are the main three things we want to realize.**

***Index Terms*—shoebox; shoes care; shoes rack; IoT;**

Role	Name	Task and description etc
User	Kwon Gyuhyeok	Suggest the actual features for the Smart Shoebox users can feel comfortable and interesting to use and let the developer know what they want and what they need.
Customer	Shin Minki	Suggest the actual features for the Smart Shoebox customers can feel comfortable and interesting to use. The feedback from both user and customer will be considered developing the Smart Shoebox.
Software developer	Kim Junghyun	Focusing on the Technical aspects of the Smart Shoebox while developing to satisfiy the user and customer's need.
Development manager	Ko Byunghee	Consider the service side of the Smart Shoebox while developing the system to satisfiy the user and customer's need.

TABLE I
ROLE ASSIGNMENT

I. INTRODUCTION

Many people experience difficulty managing their own shoes in a decent and pleasant form. Especially for the people living alone, keeping shoes clean and sweet smelling becomes a tough task to manage. When it rains, shoes get wet and dirty. Can you imagine the smell and feel of the shoe? Even worse the smell starts from the entrance to the place where you will go to sleep. This is when the actual management features are required.

What if someone or something could take care of my shoes periodically and automatically. If the shoes could be managed regularly with the aspects of humidity, temperature, and sterilization, it will save money and also provide a pleasant day with a cozy footwear. To realize the concepts of taking care of our shoes, we will develop a shoebox which manages shoes condition by controlling humidity and temperature automatically and periodically.

We are going to use Arduino to support with humidity and temperature recognition by receiving inputs through switches or sensors. Internet of Things (IoT) is also on the base of the idea. The ability to control things (especially shoes in this case) through internet is the main concept we are trying to realize. We are looking forward to create an integrated service tool such as situation awareness, automatic computing, self-growing.

The website will be designed and developed through html, ruby on rails and bootstrap. As the importance of user interface is becoming a big issue, we will try to make the interface more consumerized so that the user can easily use the Smart Shoebox.

II. REQUIREMENT

A. Optimization function

When we wear shoes, they easily become in a state of high temperature and humidity which causes the disgusting smell, which is also the best environment for bacteria to grow. As a result, there is a need to control the condition of the cabinet keeping the shoes. To provide an optimized environment automatically and also on user?s demand is the goal. (There is a need for defining optimized temperature and humidity)

1) Temperature/Humidity control through electric fan (automatic): The sensor receives temperature and humidity as inputs and provides an optimized environment as an output.

2) Temperature/Humidity control through ultraviolet lamp (automatic): The sensor receives temperature and humidity as inputs and provides an optimized temperature and humidity as output.

3) Drying feature (on demand): In case the user's shoes get wet by rain or other liquids the user can request for drying will operate (1), (2).

4) Sterilization function (on demand): In case the user feels the need for sterilization, user can request for this function, which operates (1), (2). This function (4) differs from (3) in degrees of intensity.

5) Deodorization function (on demand): In case the user feels the need for deodorization, user can request for this function, which triggers a deodorant to shoot out.

6) Deodorization function (automatic): The user can set regular intervals to trigger the deodorant to shoot out.

7) Intensity control feature: The user can choose the intensity level of (1), (2). Intensity is calculated as number between 1 to 5.

B. Management function

Different type of shoes requires different type of proper cares. The shoe rack needs to understand and recognize the shoes type and provide a proper management for the shoes. (Modeling : changing ambiguous information into actual concept.)

1) Shoe categorization function (bar-code scanning): Shoe categorization through capturing the barcode for the shoes.

2) Shoe categorization function (user input based): Shoe categorization through selected category of the user.

3) Shoe categorization function (captured image): Shoe categorization through captured images of the shoes.

4) Shoe categorization function (3D scanning): Shoe categorization through 3D scanning of the shoes.

5) Setting the proper management tool: After Shoe categorization, based on the shoes category, the shoe rack provides the proper setting. (There is a need for defining proper setting for each category) The proper setting is different in the aspect of the intensity from Optimization environment function.

C. Analysis function

To keep the user's shoes in high quality we can provide an analysis for the shoes the user own.

1) Absence of shoes analysis (Base information): We have decided to analyze the absence of shoes by sensor and use it as a base information for other analysis functions.

2) Durability analysis: Durability is set to decrease by the time the shoe has been put on increases.

3) Life prediction analysis: Based on the information of (1), we provide the expected life of the shoes.

4) Preference analysis (personal): Based on the information of (1) for one user, we provide the preference information of the shoes. More the user put on, more the preference increases.

5) Preference analysis (general): Based on the information of (1) for a number of users, we provide the preference information of the shoes for general aspect. Using this Big data, the user can know which shoes are popular nowadays.

6) Frequency analysis: Based on the information of (1), we provide the frequency information for the shoes.

7) Walking habit analysis (health care): Based on the information flatness of the shoes , we provide the information about walking habit of the users.

D. Recommendation function

Smart Shoes cabinet will provide recommendation information with percentages based on different kind of aspects. Of course the final choice is up to the user.

1) Recommendation based on weather forecast : With weather API, the proper type of shoes is recommended.

2) Recommendation based on the use of shoes: Recommending the shoes type which matches with the user?s activity.

3) Recommendation based on the color of shoes: Recommending the shoes color which balances with the users clothing color.

4) Notice of recommendation rate by color: Showing the recommendation rate by different colors. For example, if the shoes are recommended, a specific color will appear on the shoe rack or on the screen the user is looking at.

5) Notice of recommendation rate by percentage: Showing the recommendation rate by percentage. If the shoes are recommended strongly, the percentage will appear on the shoe rack or on the screen the user is looking at.

E. Notification function

Shoes easily get dirty, since when people do activities, shoes are the first thing that touches the ground. The shoes cabinet will provide notification for contamination of dirt or rainwater by checking on the weight difference.

1) Recognition of contamination by sensor: With the increased weight, notification is given for contamination.

2) Notification for contamination by message: After the recognition of contamination, the information is notified to the user through messages.

F. Networking / Remote control function (UI)

Without the function for internet control, it becomes nothing more than a drying machine. With this networking function on the base, the user is able to take care of the users shoes any time, anywhere. This is the most important feature we will concentrate on. Providing the IoT environment is the main goal.

1) Control function through web programming (main): With web based program, the user can interact with the smart shoe care software and other provided information.

2) Control function through mobile (sub): With mobile application, the user can interact with the smart shoe care software and other provided information.

3) Control function through embedded system (sub): With embedded system, the user can interact with the smart shoe care software and other provided information.

III. Development Environment

A. Choice of software development platform

1) Platform used for developing: package We will use both Windows and MAC OS . Since Windows is the most popular OS used worldwide and MAC OS is the second most popular OS leaving out all the other versions of Windows. We thought MAC OS X will become more popular. We also thought using other OS besides windows will mean a lot for us to use another environment to develop a software.

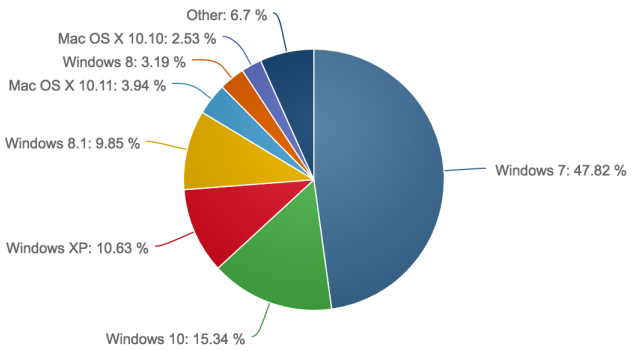

Fig. 1. Market share reports (January, 2016 to March, 2016)

2) Programming language used for developing: We are using Arduino, SQLite, Ruby and HTML. We are trying to provide a web service with arduino acting inside the Smart Shoebox. The frontend will be using html and css, while the backend will be using ruby and ruby on rails as a application framework. We thought of using amazon EC2 but if we think of the server as a localhost, we might be using only ruby and ruby on rails for the server with SQLite.

Programming language	Reason
Arduino(hardware)	The main hardware part of our project is based on Arduino. The Smart Shoebox has functions to work provide behavioral motions such as recognizing the temperature and humidity of the shoebox, turning on the fan or infrared lamp as a result of it and so on.
MySQL(server side)	We need to have a database to save information about the shoes, users. To easily get and set and manage the information, we have decided to use a database management tool.
Ruby	We first thought of php for the work between the server and web side environment, since we have all learned php in another course. Though we thought it would be much better to learn a new language for this project. Ruby on rails was the interesting programming language in the aspect that it shortens and simplifies the code much more than the php.
HTML5 and CSS3(client side)	We have decided the user interface environment as a web-based structure. The functions of Smart Shoebox will be triggered and managed in the web.

TABLE II
Programming language used for developing

3) Cost estimation (Software / Hardware): TABLE III and TABLE IV

Device	Price (won)
Arduino uno R3	7,500
Bread board	2,400
Wifi module(ESP8266)	9,000
Temperature Humidity sensor	3,000
Pressure Sensor	14,000
Fan(actuator)	4,000
Board	2,000
USB cable	500
jump wire	2,500
M-F wire	2,000
Resistance	200 (5 per unit)
Small LED lamp	1,000
AA battery	1,200
transistor	500
TOTAL	49,800

TABLE III
COST ESTIMATION(HARDWARE)

Software	Task Description
Source Tree(v1.8.3)	Version control
Git(v2.8.1)	Project control
Github	Remote repository
Sublime Text3(3103)	Text editor
mockflow	Wireframe creation
Mac OS X El Capitan	Operating System
Windows 8 / 10	Operating System
Arduino(v1.6.8)	Text editor for Arduino
TOTAL	0

TABLE IV
COST ESTIMATION(SOFTWARE)

B. Software in use

We have researched to find out if there is any existing software or algorithm in use doing a similar task we are trying to provide. We were really surprised to find so much information related to our project. There was a lot of algorithms and systems during the research. The most interesting and related ones were the three below.

1) Temperature Humidity Control system: As anyone can think of the air conditioner or greenhouse there were already a lot of systems and devices doing the actual part of our project to control the temperature and humidity for the given environment. (For our home, or for growing plants in the optimized temperature and humidity, and so on.) Even there were a lot of information about making the Arduino actually work as we planned to.

Fig. 2. Advance Temperature Control (http://infusionva.com/)

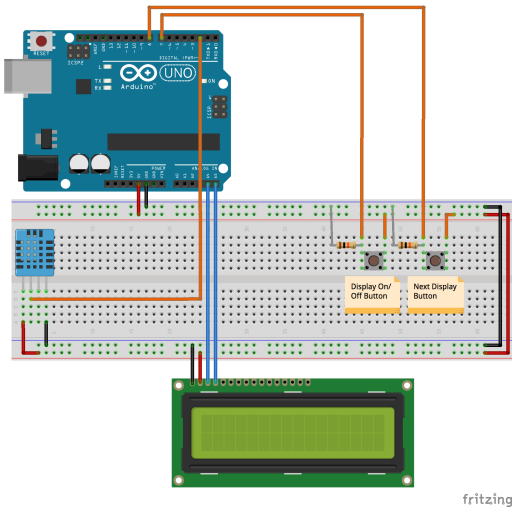

Fig. 3. Airconditioner automatic control through Arduino

2) Recommendation System : There is an extensive class of Web applications that involve predicting user responses to options. Such a facility is called a recommendation system. In Figure above we see an example utility matrix, representing users? ratings of movies on a 1?5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for Harry Potter I, II, and III, TW for Twilight, and SW1, SW2, and SW3 for Star Wars episodes 1, 2, and 3. The users are represented by capital letters A through D. The goal of a recommendation system is to predict the blanks in the utility matrix. This recommendation system was in common with our

project in the point that we will provide a recommendation information for the shoes with the weather APT and color of the shoes matching with the user?s clothes.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Fig. 4. utilitymatrix

3) Classification Algorithm in datamining: Basic Principle (Inductive Learning Hypothesis): Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

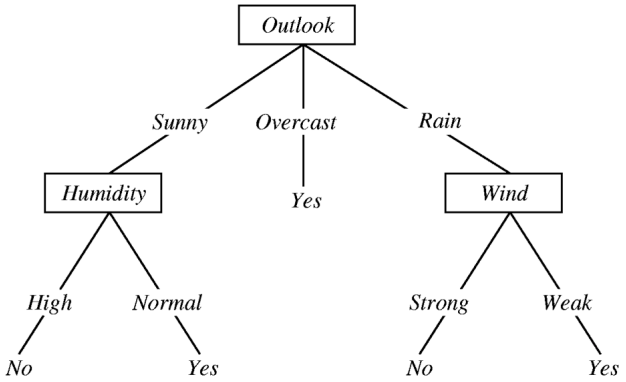

Fig. 5. decisiontree learning

This algorithm was in common with our project in the point that we will classify the shoes.

4) Wifi technology: The IOT typically employ some kind of embedded technology that allows them to sense conditions such as pressure, humidity, temperature, motion, number of people in an area, etc. And then a technology allows them to connect to other things or the cloud so that they can send the information as well as be programmed. The choice of technology is usually dictated by the physical characteristics of the environment, such as the presence of wood, concrete, metal etc., the density of sensors, desired range, and data rates. Among these technologies, we choice the Wi-Fi as the most successful tool.

Even though there are many ways to communicate wireless, we have decided to use wifi as the communication tool because we also use the website in the base.

Also as we continued the research there were many advantages using the wifi.The most important part was the low cost of wifi. In the semiconductor world, most chips have roughly the same production cost, given some variability in die size and process technology, and with costs falling with increasing volume. While Wi-Fi chips have historically been priced higher than those based on the technologies more often associated with M2M and IoT, manufacturing costs are not necessarily greater. We expect, then, that Wi-Fi chips used in IoT applications and produced in very large numbers will in fact be very cost effective and comparable in price to any other suitable wireless components. And the greater functionality of Wi-Fi (security, power management, robust and mature firmware and drivers, etc.) adds significant value to that cost calculation.

Also the low power gave us the reason to use wifi. IoT applications will seldom require 802.11n and 802.11ac levels of throughput, but lower-cost and more power-friendly solutions based on these (single-stream, for example) will typically be applied and implemented using modern low-power semiconductor process technologies. Of course, not all IoT applications require battery power, but those that do will see no disadvantage from going the Wi-Fi route as Wi-Fi?s long history of power-saving capabilities applies quite nicely to IoT.

Fig. 6. IoT wifi technology

5) IOT example : Temperature Monitoring: Importance of monitoring temperature in industries is quite important as it affects the environment and the process of business e.g. food industry, process industry or any high tech industry such as data centers. Temperature is one of the important parameters which is monitored and controlled to ensure quality output. Most of the high tech industries will have high end climate control systems to monitor and manage temperature, however all organizations cannot afford the luxury of high end systems. Apart from small scale industry, temperature monitoring is important to people with remote cabins, warehouses, or any other location where access to location is difficult. With advancement in software technology and electronics it has become technically feasible to make pocket size high speed and high memory devices. The system is based on the Raspberry Pi which is cost effective and core OS is soft real time OS based on Linux. With Linux as OS, variety of features can be offered such as security, license free usage and community support. The frequency with which data is scanned is customizable and currently programmed to scan every 1 Hr (can be scaled). The data is stored in local MySQL database with updates

sent to cloud. The Raspberry Pi also acts as a Server which can be accessed to see logged data. The data in cloud also represents data graphically and can be downloaded in CSV format. One of the biggest benefits of using Raspberry Pi is the sensor network expansion. The sensors can be added can be configured and further extended with legacy RJ45 cable to and from Sensor Network. Capability to store data from other sensors such as light sensor, humidity, wind sensor, etc. gives flexibility to the system to customize and accommodate quite a lot of wants.

In this IOT example(Temperature Monitoring) we decided to use the ThingSpeak for getting information from Arduino sensor and compatible programming with Raspberry Pi.

6) Selfstarter and Refinery CMS : Ruby on rails web application: Among funding platforms galore, Self-starter helps you host your own crowdfunding site for your project. Donations are easily made via Amazon Payments, or you can choose your preferred provider. Simple and easy, driven by Ruby on Rails.

Creating a website and managing content is much easier with Refinery CMS. It's a simple modular and extendable content management system. Released in 2009, it's currently undergoing improvements by the community in 10 different languages

From these two open sources, we studied and try to follow the ruby on rails framework for our website. We were really surprised to find a lot of open sources that we can easily find and learn.

7) Ruby on rails: Ruby on Rails, or simply Rails, is a web application framework written in Ruby under the MIT License. Rails is a model?view?controller (MVC) framework, providing default structures for a database, a web service, and web pages. It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing. In addition to MVC, Rails emphasizes the use of other well-known software engineering patterns and paradigms, including convention over configuration (CoC), don't repeat yourself (DRY), and the active record pattern.

8) Database Management System: A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA, and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one DBMS. Database management systems are often classified according to the database model that they support; the most popular database systems since the 1980s have all supported the relational model as represented by the SQL language.

We have decided to use the database to store information about the users and the shoes inside the Smart Shoebox. With the stored information we can realize the feature of managing all the shoes the user possess.

```ruby
class ExtractIssueTracker < Mongoid::Migration
  TRACKER_MAPPING = {
    'ErrbitTracPlugin::IssueTracker'        => 'trac',
    'IssueTrackers::BitbucketIssuesTracker' => 'bitbucket',
    'IssueTrackers::FogbugzTracker'         => 'fogbugz',
    'IssueTrackers::GithubIssuesTracker'    => 'github',
    'IssueTrackers::GitlabTracker'          => 'gitlab',
    'IssueTrackers::JiraTracker'            => 'jira',
    'IssueTrackers::LighthouseTracker'      => 'lighthouse',
    'IssueTrackers::PivotalLabsTracker'     => 'pivotal',
    'IssueTrackers::RedmineTracker'         => 'redmine',
    'IssueTrackers::UnfuddleTracker'        => 'unfuddle'
  }

  def self.up
    App.all.each do |app|
      next unless app.attributes['issue_tracker'].present?
      next unless app.attributes['issue_tracker']['_type'].present?

      options = app['issue_tracker'].dup
      options.delete('_type')
      options.delete('_id')

      type = app.attributes['issue_tracker']['_type']
      updated_at = options.delete('updated_at')
      created_at = options.delete('created_at')

      next unless TRACKER_MAPPING.include?(type)

      tracker = {
        'type_tracker' => TRACKER_MAPPING[type],
        'options'      => options,
        'updated_at'   => updated_at,
        'created_at'   => created_at
      }

```

Fig. 7. Example for database in ruby on rails

This is the open source we have researched to study the database inside ruby on rails.

C. Task Distribution

We have decided to distribute the project in big parts to make each participants to be responsible for the assigned parts. Still every person has to know how the project is going on in a big picture while being responsible for the assigned part.

Name	Responsible Part
Kwon GyuHyeok	Arduino programming
Shin MinKi	SQLite, database schema
Kim JungHyun	Ruby on rails, Wireless fidelity control
Ko ByungHee	Ruby on rails, Bootstrap

TABLE V
TASK DISTRIBUTION

IV. SPECIFICATION

The specification is mostly in pseudocode and additionally graphs and charts will be used to specify the requirements. Particularly, mockflow will be used to wireframe the user interface. The pseudocode is a little bit close to the programming languages we have already learned during other courses while disregarding the details of the grammar.

A. Optimization function

1) Temperature/Humidity control through electric fan (automatic):

2) Temperature/Humidity control through ultraviolet lamp (automatic): The arduino sensor receives temperature and humidity as inputs and truns on fan and ultraviolet lamp when temperature drops below 15 Celsius degree or humidity is higher than

```
// Check temperature and humidity and optimize automatically
function check_temperature(){
    if (temperature < 15) {
        return false
    else
        return true
}
//When humidity is less than 40, it is better to have higher temperature

function check_humidity(){
    if (humidity > 40 )
        return false
    else
        return true
}

//실내 적정 습도가 50~60%이고 신발장은 40%이하가 적절하다고 합니다

function auto_control_temperature_humidity(){
    if(check_teperature() == true and check_humidity() == true){
        fan_off()
        lamp_off()
    }
    else{
        fan_on(MID)
        lamp_on(MID)
    }
}
```

3) Drying feature (on demand): In case the user?s shoes get wet by rain or other liquids the user can request for drying function. The function will turn on fan on maximum rate and intensity of lamp will be middle.

```
boolean check_demand_drying

function demand_control_temperatureHumidity(){
    if(check_demand_drying == false){
        fan_off()
        lamp_off()
    }
    else{
        fan_on(MAX)
        lamp_on(MID)
    }
}
```

4) Sterilization function (on demand): In case the user feels the need for sterilization, user can request for this function. The function will turn on fan on middle rate and intensity of lamp will be maximum.

```
boolean check_demand_sterilization

function demand_control_temperature_humidity(){
    if(check_demand_sterilization == false){
        fan_off()
        lamp_off()
    }
    else{
        fan_on(MID)
        lamp_on(MAX)
    }
}
```

5) Deodorization function (on demand): In case the user feels the need for deodorization, user can request for this function, which triggers a deodorant to shoot out. The Arduino will trigger the deodorant.

```
boolean check_demand_deodorization

function demand_control_deodorization(){
    if(check_demand_deodorization == true){
        deodorization_on()
        check_demand_deodorization = false
    }
}
```

6) Deodorization function (automatic): This function will provide the Smart Shoebox to trigger the deodorant to shoot out in one hour interval.

```
function auto_control_deodorization(){
    deotoranzation_on()
    delay(one hour) // one hour interval
}
```

7) Intensity control feature: The user can choose the intensity level of fan and lamp. Intensity is calculated as number between 1 to 3, which means maximum, middle, minimum.

```
fucntion fan_on(int a){
    if a==3 // turn fan MAX
    else if a==2 //  turn fan MID
    else // turn fan MIN
}

fucntion lamp_on(int a){
    if a==3 // turn lamp MAX
    else if a==2 // turn lamp MID
    else // turn lamp MIN
}
```

B. Management function

This management function is mostly about modeling and categorizing the shoes. Through the serial number scanning, captured image, 3D scanning and user input, the information about the user and shoes are managed in the database.

1) Shoe categorization function (bar-code scanning):

2) Shoe categorization function (user input based):

3) Shoe categorization function (captured image):

4) Shoe categorization function (3D scanning):

5) Setting the proper management tool: Modeling

The big arrow pointing to the database will be one of the ways of receiving the information of the shoes by the user. Like we mentioned at above part the ways will be the serial number scanning, analyzing the captured image, 3D scanning and analyzing user input.

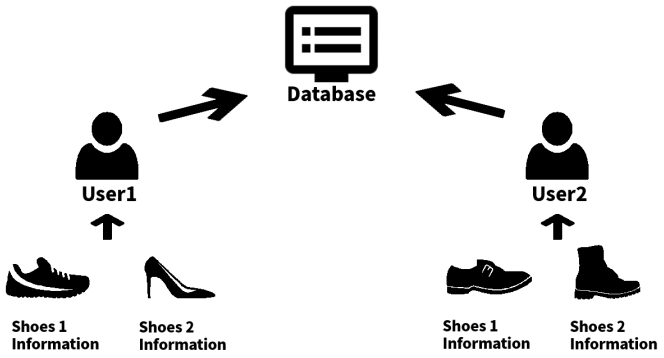

Fig. 8. Shoe information into the database

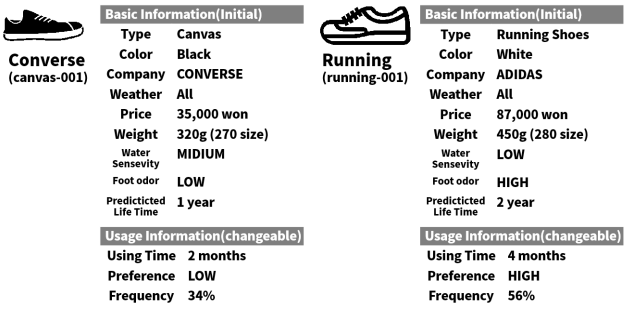

Fig. 9. Example of the shoes modeling

As we can see in Fig.7 the saved information of the shoes will be assigned to each model we will provide before. For each model the proper care solution will be provided. The modeling of each shoe into a number of category and setting the proper care solution will be the most important part to realize this function. Also at the same time it is the most ambiguous part before the design phase.

C. Analysis function

1) Absence of shoes analysis (Base information): We have decided to analyze the absence of shoes by weight sensor and use it as a base information for other analysis functions. The absence timer will be on and count the time of the shoes absence.

```
int absence_time

function check_absense(){
    if (weight_sensor == 0)
        absence_timer_on()
    else
        absence_timer_off()
}
```

2) Durability analysis: Durability is set to decrease by the time the shoe has been put on increases. We have decided the average shoes usage time as 12 hours a day and the durability changes while the absence time increases. Less than 2 weeks is considered to be a new one. Between 2 weeks and 4 weeks is considered to be not bad. Between 4 weeks and 8 weeks is considered to be washed. Between 8 weeks and 16 weeks is considered to be careful. More than 16 weeks the shoes is considered to be replaced.

```
// average time wearing shoes : 12 hours a day
string durability

function check_durability(){
    if absence_time < 2 weeks
        durability = new one
    else if 2 weeks < absence_time < 4 weeks
        durability = not bad
    else if 4 weeks < absence_time < 8 weeks
        durability = need an washing at least once
    else if 8 weeks < absence_time < 16 weeks
        durability = be careful
    else
        durability = recommend to buy new one
}
```

3) Life prediction analysis: Based on the information of absence time, we provide the expected life of the shoes. We subtract the absence time from the original life time.

```
int shoes_number // assigned number for each shoes
int life_time

function check_life_time(int shoes_number, int absence_time) {
    life_time = original_life_time(shoes_number) - absence_time
    return life_time
}
```

4) Preference analysis (personal):

5) Preference analysis (general): Based on the information of absence time for a number of users, we provide the preference information of the shoes in general aspect. Since we all store the information about the shoes of the user in the database this is possible in both personal and general aspect. Using this Big data, the user can know which shoes are popular nowadays.

```
int count_using_time = absence_time
int preference // server variable

function check_preference(){
    if count_using_time > 2 weeks {
        count_using_time = count_using_time - 2 weeks
        // one week use : preference + 1
        preference = preference + 1
    }
}
```

6) Frequency analysis: Based on the information of absence time, we provide the frequency information for the shoes in percentage. We devide the used days by the whole day since the user started to use the Smart Shoebox.

```
(6)  Frequency analysis

int use_date

function check_frequency(){
        return use_date / whole_date * 100
}
```

D. Recommendation function

Smart Shoes cabinet will provide recommendation information with percentages based on different kind of aspects. Of course the final choice is up to the user. The main recommendation standard is the weather. So we have decided to place a screen for the weather.

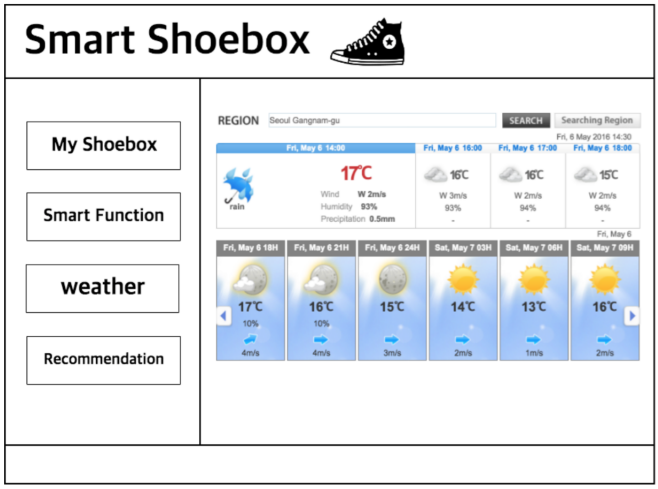

Fig. 10. Weather on the web screen

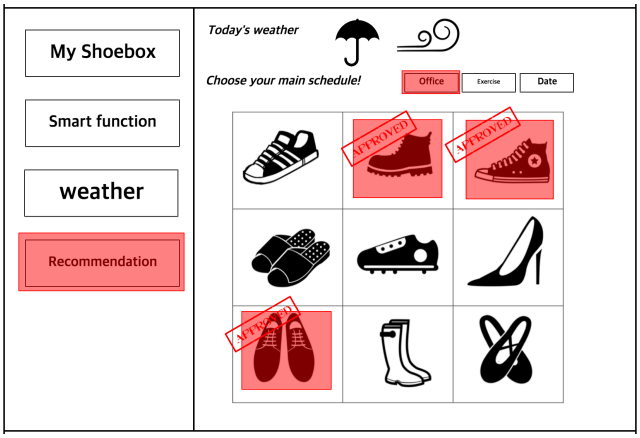

Fig. 11. Recommendation on the web screen

1) Recommendation based on weather forecast : With weather API, the proper type of shoes is recommended. We decided to classify weather in three cases. ('Sunny' and 'Cloudy' and 'Rainy or Snowy') We will restrict the recommendation choices by the weather becomes bad. The word proper (the proper shoes for the weather) will be represented in the shoes modeled information in the database.

```
string weather

fuction weather_forcast(String Weather){ // from weather API
    if(Weather='sunny'){
    return 1;
}
else if(Weather='cloudy'){
    return 2;
}
else if(Weather='rainy' or Weather='snowy'){
    return 3;
}
}
function recommend_weatherforcast(){
    switch(Weather_forcast()){
    case 1 : return shoes1
    case 2 : return shoes2
    case 3 : return shoes3  // sunny -> shoes1,2,3  cloudy -> shoes2,3  rainy -> shoes3
            break
    }
}
```

2) Recommendation based on the use of shoes: Recommending the proper shoes type which matches with the user?s activity. The word proper (the proper shoes for the activity of the user) will be represented in the shoes modeled information in the database.

```
string user_purpose

function recommend_user_purpose(){
    if user_purpose == exercise
        return shoe1
    else if user_purpose == office(formal)
        return shoe2
    else if user_purpose == (informal)
        return shoe3
}
```

3) Recommendation based on the color of shoes: Recommending the proper shoes color which balances with the users clothing color. The word proper (the proper shoes for the clothes of the user) will be represented in the shoes modeled information in the database.

```
fuction recommned_color(class shoe){
    if shoe.color matches clothes.color
        return recommend
    else
        return not recommended
}
```

4) Notice of recommendation rate by color:

5) Notice of recommendation rate by percentage: Showing the recommendation rate by different colors. For example, if the shoes are recommended, a specific color will appear on the shoe rack or on the screen the user is looking at. Showing the recommendation rate by percentage. If the shoes are recommended strongly, the percentage will appear on the shoe rack or on the screen the user is looking at.

```
function notice_recommedation(class shoe) {
    if recomedation_rate > 80
        recommedation_color = green
        display recommedation_rate, recommedation_color
    else if 60 < recomedation_rate < 80
        recommedation_color = lightgreen
        display recommedation_rate, recommedation_color
    else if 40 < recomedation_rate < 60
        recommedation_color = orange
        display recommedation_rate, recommedation_color
    else if recomedation_rate < 40
        recommedation_color = red
        display recommedation_rate, recommedation_color
}
```

E. Notification function

1) Recognition of contamination by sensor:

2) Notification for contamination by message: With the increased weight, notification is given for contamination. The weight to be sensed is defined to be 20g. After the recognition of contamination, the information is notified to the user through messages on the screen.

```
function check_contamination(){
    if weight_sensor_now > weight_sensor_average + 20
        notification_message()
}
```

F. Networking / Remote control function (UI)

1) Control function through web programming (main):

2) Control function through mobile (sub):

3) Control function through embedded system (sub): To realize this function we will provide a user interface through mainly web and additionally mobile and a screen for the Smart Shoebox. The already mentioned functions before will be on the UI so that the user can interact with the system. The temperature and humidity will be provided on the screen the user will be seeing. Optimaization function, management function, analysis function, recommendation function, notification function will be on the main screen for users to be able to use.

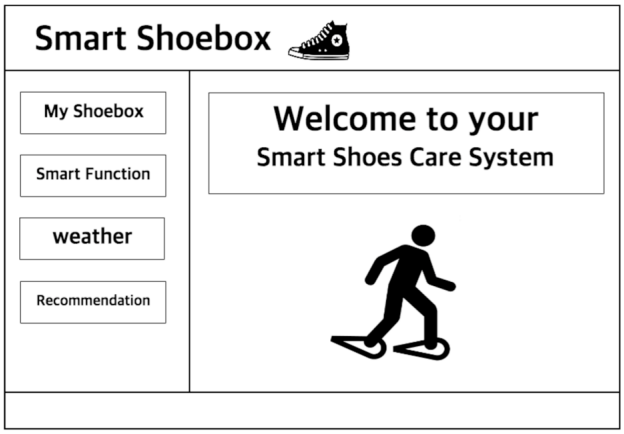

Fig. 12. Main page for Smart Shoebox

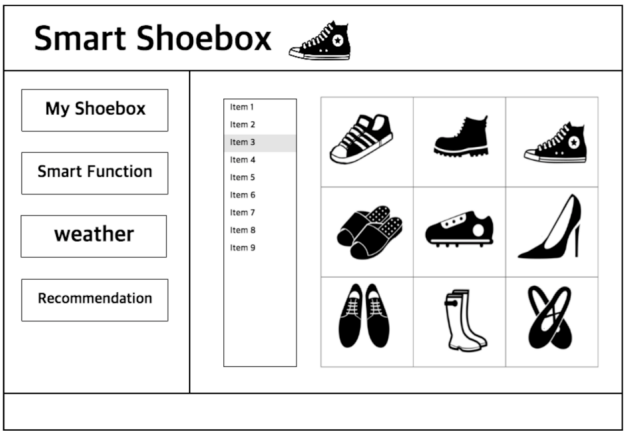

Fig. 13. Shoes status for Smart Shoebox

V. Architecture Design and Implementation

A. Overall architecture

We have three modules consisting of the Smart shoebox. First part is the Arduino module, which receives the request of the user by jQuery and reacts. It also provides temperature, humidity, pressure information to the web server, which is provided by the sensor of the Arduino. The second and third part is a set of web service. One is the front end module(HTML/CSS) and the other is the back end module(SQLite). The front end is mainly reacting with the client(user) and back end is mainly reacting with the server. The user can add a new item and sign up to the Smart Shoebox service. The server will provide the detail information of the registered item and recommend. As you can see in Fig. 12 all modules interact with each other.

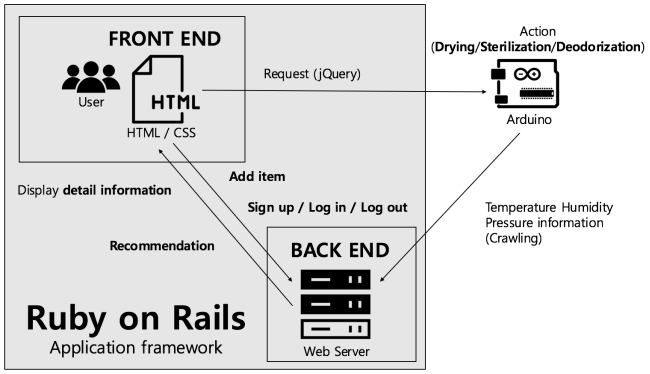

Fig. 14. Overall system architecture

B. Directory organization

We have three big folders, which follows the overall system architecture. One is the arduino and another is website folder for the ruby on rails and the last one is a folder for the LaTex documentaion.

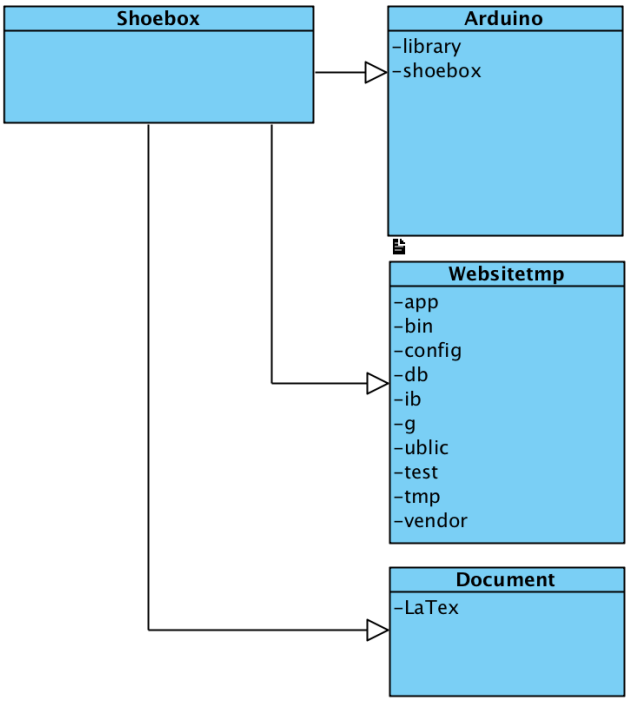

Fig. 15. Directory organization

This is the directory organization using UML. There are three big parts including Arduino, Website and Document.

Directory	File names	Module name
/project/arduino/library	dht11.cpp, dht11.h	arduino
/project/arduino/shoebox	shoebox.ino	arduino
website/app/controller	application-controller.rb, home-controller.rb, mybox-controller.rb, smart-controller.rb	rails model
/project/website/app/helper	application-helper.rb, home-helper.rb, mybox-helper.rb, smart-helper.rb	rails helper
/project/website/app/models	shoe.rb, user.rb	rails model
/project/website/app/views	...	rails view
/views/home	index.html.erb	rails view
/views/layouts	application.html.erb	rails view
/views/mybox	information.html.erb, detail.html.erb, new.html.erb	rails view
/views/smart	index.html.erb	rails view
/project/website/bin	bundle, rails, rake, setup	rails bin
/project/website/config	routes.rb	rails routes
/project/website/db	schema.rb,seeds.rb	rails db
/project/website/db/migrate	20160525053629-devise-create-users.rb, 20160525053708-create-shoes.rb	rails db migrate
/project/website/lib		rails lib
/project/website/log	development.log	rails log
/project/website/public	404.html, 422.html, 500.html	rails public
/project/document/LaTex	document.tex, document.aux, document.log, document.pdf	documentation

TABLE VI
DIRECTORY ORGANIZATION

C. Module1:Arduino

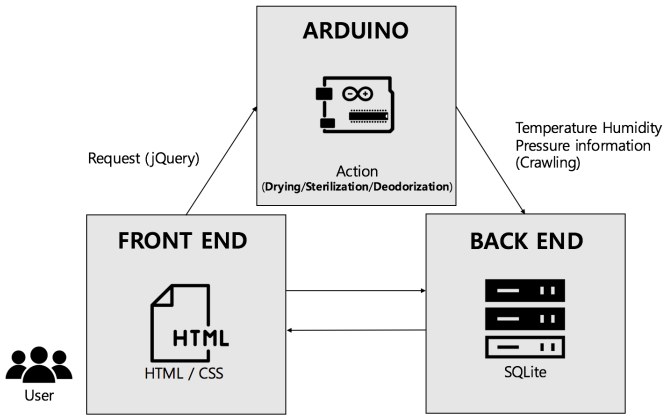

Fig. 16. Module1 : Arduino

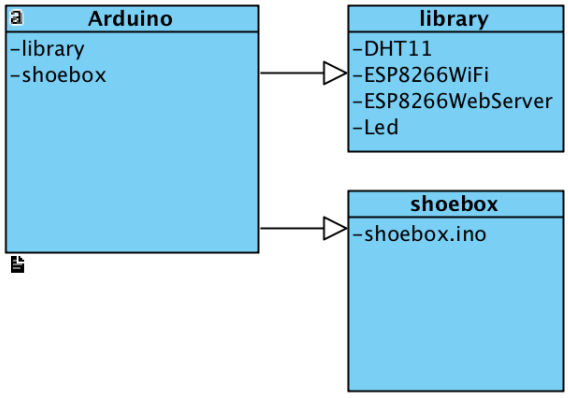

Fig. 17. Arduino : Directory organization

1) Purpose: First, we are trying to receive information about the temperature, humidity, and pressure by each related sensors. The pressure sensor is for the recognition of the shoes. Second, we are trying to perform the smart functions the user will be requesting. Third, we are trying to interact with the Smart Shoebox with wifi module of Arduino.

2) Functionality: The functionality fits to the purpose of the module. It provides wirless communication between the Smart Shoebox and the user interacting with the website. Also the additional sensors can measure the temperature, humidity and pressure. Consequently with the informations of the sensor the fan, infrared lamp and deodorant will carry out the smart functions we designed.

3) Location of Source Code: /project/shoebox/arduino

4) Class component: There are three components in the Arduino class.

- Setup( ) : This is the work done only at the first time when the Arduino starts to activate. (For example, setting up the rate or the wifi module.)

```
void setup() {
  Serial.begin(9600);
  esp8266.begin(9600);

  pinMode(11, OUTPUT);
  digitalWrite(11, LOW);

  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);

  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  sendData("AT+RST\r\n",2000,DEBUG);
  sendData("AT+CIOBAUD?\r\n",2000,DEBUG);
  sendData("AT+CWMODE=3\r\n",1000,DEBUG);
  sendData("AT+CWLAP\r\n",3000,DEBUG);

  sendData("AT+CWJAP=\"AndroidHotspot5051\",\"rbgur123!@#\"\r\n",5000,DEBUG);
  sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address - 192.168.43.194
  sendData("AT+CIPMUX=1\r\n",1000,DEBUG);
  sendData("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n",1000,DEBUG);
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80

  dht.begin();

}
```

- Loop( ) : This is the ongoing work of Arduino, while the Smart Shoebox is on. It is repeatedli reading the temperature and humidity from the sensor.

```
void loop() {

 float humi = dht.readHumidity(); // 습도값 읽기

 float temp = dht.readTemperature(); // 온도값 읽기

  Serial.print("Humidity : ");
  Serial.print(humi);
  Serial.println();
  Serial.print("Temperature : ");
  Serial.print(temp);
  Serial.println();

 FSR_value = analogRead(sensorPin);
 Serial.print("Sensor : ");
 Serial.println(FSR_value);

 if(FSR_value > 10){
 digitalWrite(10, HIGH);
 }
 else{
  digitalWrite(10,LOW);
  using_time = using_time + 1;
 }
 Serial.println(using_time);

  char buf[16];
  String strTemp = dtostrf(temp, 4, 1, buf);
  String strTemp2 = dtostrf(using_time, 4, 1, buf);
  String strTemp3 = dtostrf(humi, 4, 1, buf);

    String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
```

- Library : The open source provided to help developing Arduino.

5) Where it is taken from: It is taken from the library list of Arduino official homepage. Arduino Playground is a wiki where all the users of Arduino can contribute and benefit from their collective research.

(http://playground.arduino.cc/Main/LibraryList)

6) How and Why we use it: We connect all the modules, sensors, LED light bulbs, fan and infrared lamp to the Arduino Uno, the main board. Additionally, with the libraries provided, we write down the code for the the functions and compile. When the codes are uploaded to the main board, the required functions are performed.

D. Module2:Ruby on Rails

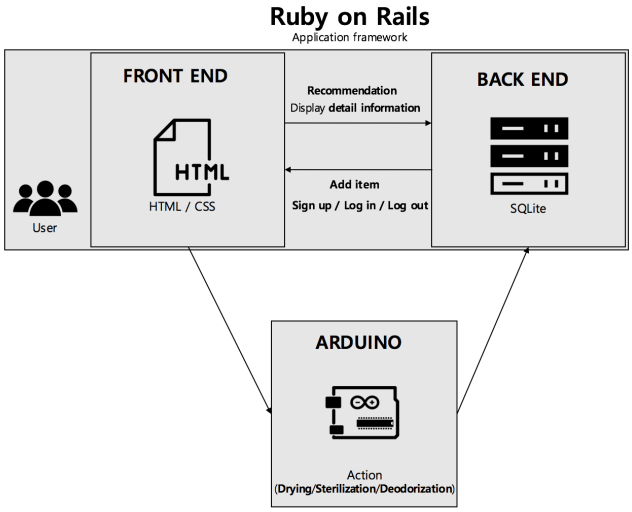

Fig. 18. Module2 : Ruby on Rails

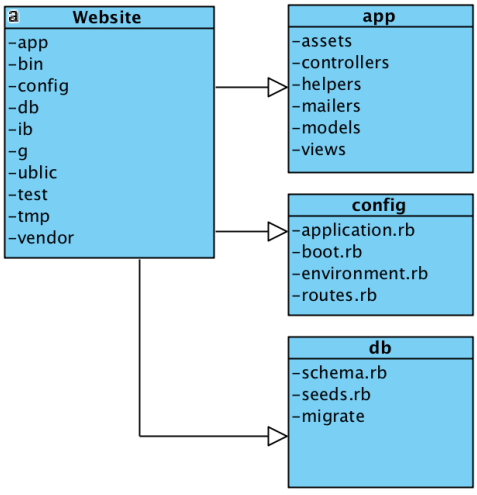

Fig. 19. Ruby on Rails : Directory organization

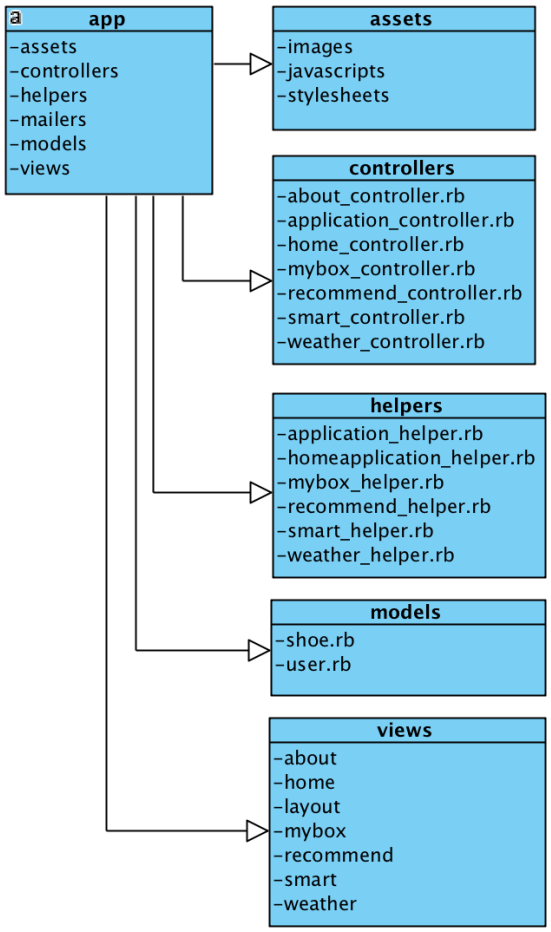

Fig. 20. Ruby on Rails : app : Directory organization

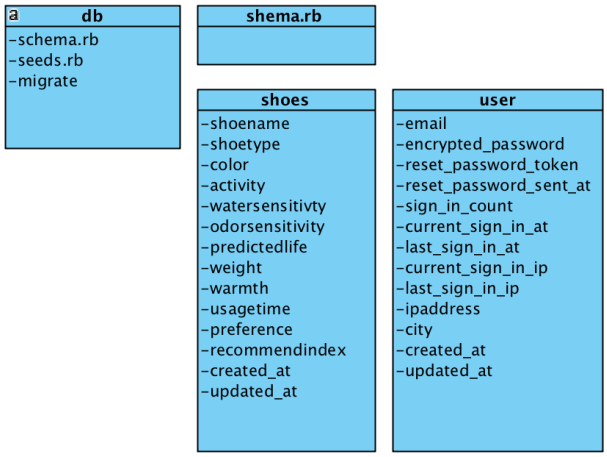

Fig. 21. Ruby on Rails : db : Directory organization

```
ActiveRecord::Schema.define(version: 20160525053708) do

  create_table "shoes", force: :cascade do |t|
    t.string   "shoename"
    t.string   "shoetype"
    t.string   "color"
    t.integer  "activity"
    t.integer  "watersensitivity"
    t.integer  "odorsensitivity"
    t.integer  "predictedlife"
    t.integer  "weight"
    t.integer  "warmth"
    t.integer  "usagetime",        default: 0, null: false
    t.integer  "preference"
    t.integer  "recommendindex",   default: 0, null: false
    t.boolean  "jogging"
    t.boolean  "casual"
    t.boolean  "sports"
    t.boolean  "trip"
    t.boolean  "storll"
    t.boolean  "official"
    t.boolean  "date"
    t.boolean  "business"
    t.boolean  "soccer"
    t.boolean  "basketball"
    t.boolean  "house"
    t.integer  "user_id"
    t.datetime "created_at",                    null: false
    t.datetime "updated_at",                    null: false
  end
```

Fig. 22. Modeling Shoes in database

```
create_table "users", force: :cascade do |t|
  t.string   "email",                  default: "", null: false
  t.string   "encrypted_password",     default: "", null: false
  t.string   "reset_password_token"
  t.datetime "reset_password_sent_at"
  t.datetime "remember_created_at"
  t.integer  "sign_in_count",          default: 0,  null: false
  t.datetime "current_sign_in_at"
  t.datetime "last_sign_in_at"
  t.string   "current_sign_in_ip"
  t.string   "last_sign_in_ip"
  t.string   "ipaddress"
  t.string   "city"
  t.datetime "created_at",                          null: false
  t.datetime "updated_at",                          null: false
end
```

Fig. 23. Modeling Users in database

1) Purpose: The purpose of using the Ruby on Rails is to provide a easily accessed user interaction environment through web service. IoT will be realized through the web service, since it will let the user to take care of their shoes anytime, anywhere. So we have decided to use the Ruby on Rails application framework.

2) Functionality: There are several functionalities for the Ruby on Rails module. First user is able to check the status of the Smart Shoebox. The status contains the meaning of temperature and humidity. Second, the users are able to check the weather outside so that he or she could choose the proper shoes. Third with the weather information and the shoes information saved in the database recommendation function will be realized. Fourth, using the wifi, it will control the Smart Shoebox remotely. Last but not least, with Ruby on rails, the user information will be saved and managed in the database.

3) Location of Source Code: /project/shoebox/website

4) Class component: Ruby on Rails use the so called MVC model to specify the events and actions of the web service.

M satnds for model, where we save and use the data of the user and shoes. (You can easily think of a database table when we develop web service.)V stands for view, which takes care of the visualization of the html. It is the factors forming the front end of the web service. (HTML, CSS, JAVASCRIPT) C stands for controller and it takes care of the actions, receiving requests from the users and look up for the data from the model and visualizae through the view.

The user accesses the Smart Shoebox's URL and send request. Then the 'Controller' receives the request and look up the 'Model' and bring the data from the database. At last with the data brought from the model, 'View' visualizes so that the user can see.

5) Where it is taken from: It is taken from the rails installer site. RailsInstaller is the quickest way to go from zero to developing Ruby on Rails applications.

(http://railsinstaller.org/en)

6) How and Why we use it: Based on the Ruby programming language and MVC model we make a web service for the users of the Smart Shoebox. It is possible to developing a web service from scratch, but it is not a good idea to do so, when it is so important and emphasized to use an open sources. Application framework for the Ruby language is already provided and we have decided to build on top of it.

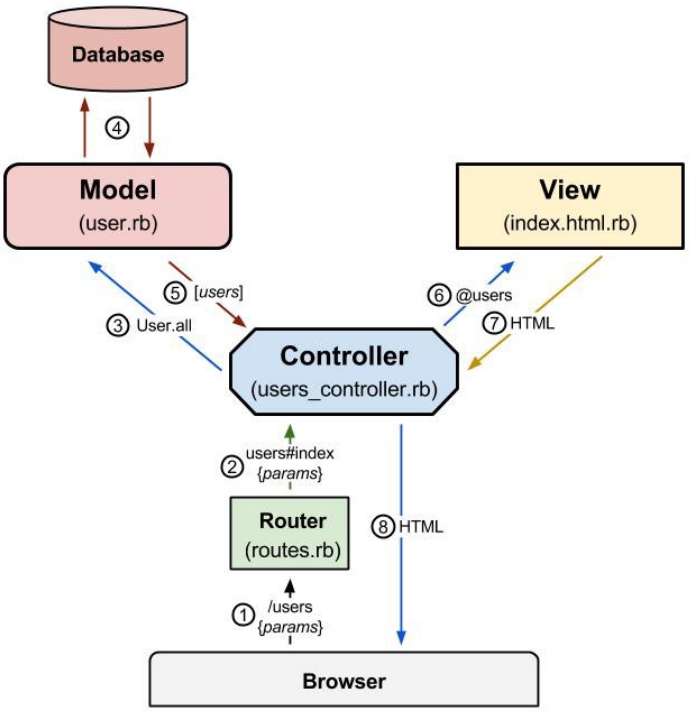

Fig. 24. MVC

VI. Use Cases

A. Use case 1 : Setup

This case is for the users using the Smart Shoebox for the first time. If it is the first time for the user to use Smart Shoebox, you have to sign up for the website. If you are already registered, you can login with your id and password.

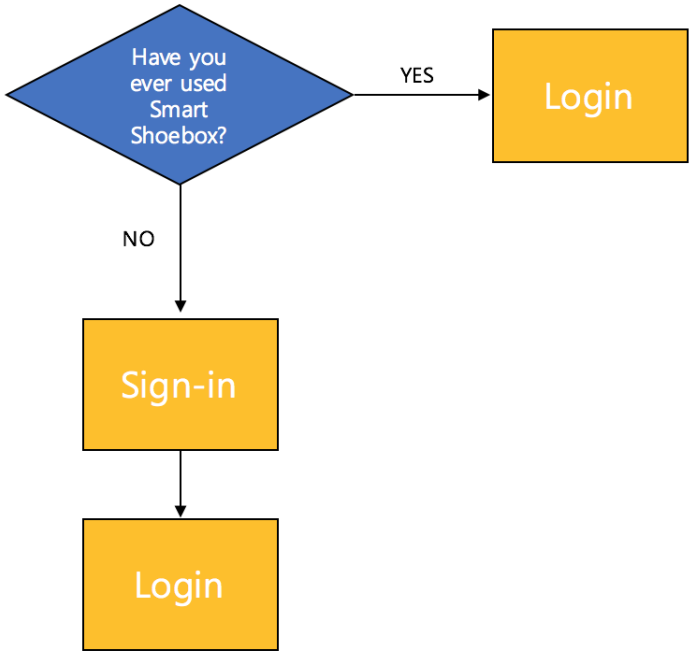

Fig. 25. Use case 1 : Setup

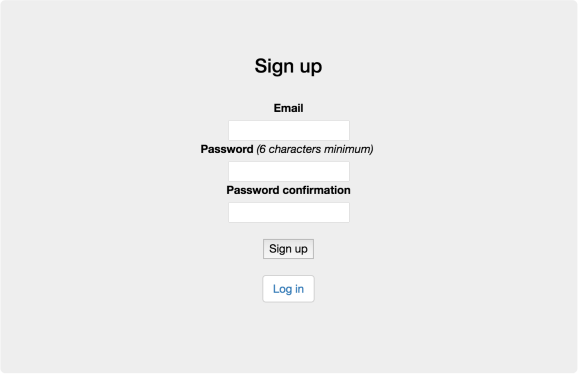

Fig. 26. Sign up page

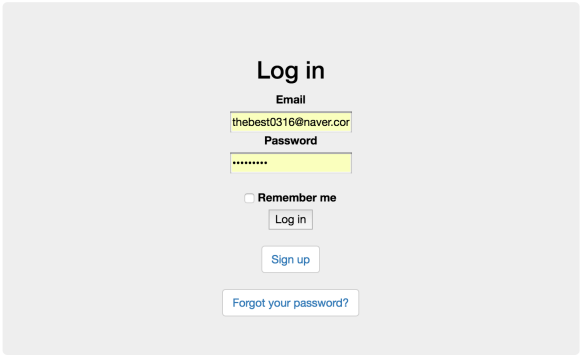

Fig. 27. Log in page

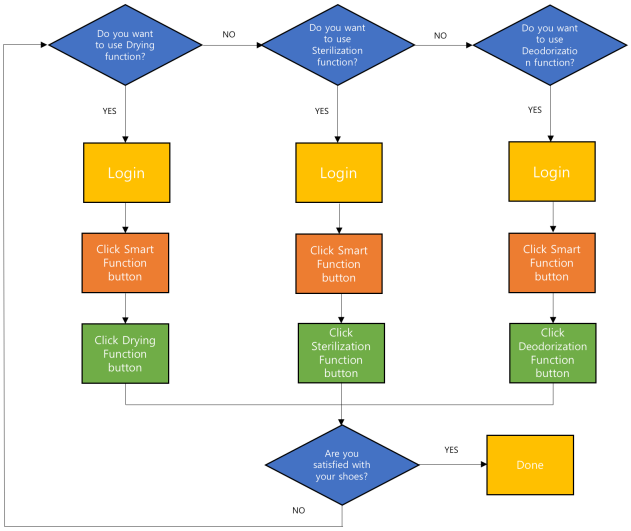

Fig. 29. Use case 2 : Users request for a Smart function

Fig. 28. After logging in

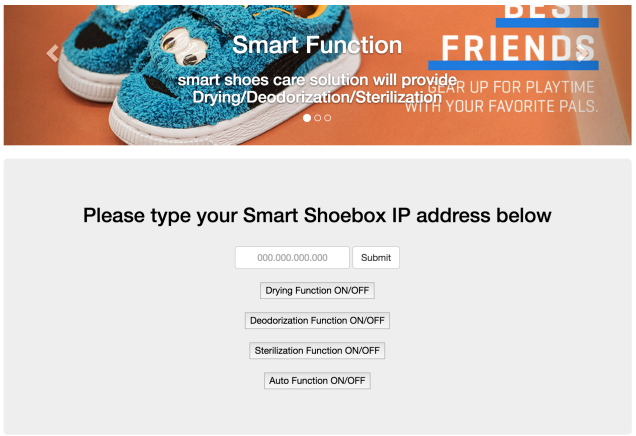

Fig. 30. Smart function page

B. Use case 2 : Users request for a Smart function

This case is taking care of the needs of the user to use the smart function of the Smart Shoebox. (Optimization, Drying, Sterilization, Deodorization) After the use case one, the user is in the website. You can click on the Smart function tab on the top. In the page of Smart function, you have to type in the IP address of the Smart Shoebox. After typing in the IP addresses, all the user has to do is click the function which the user needs. You can use the functions as long as you want and click the button again to turn it off.

C. Use case 3 : Users request for the weather forecast

This case is taking care of the needs of the user to see the weather. After the use case one, the user is in the website. You can click on the Weather tab on the top. In the page of Weather, the user can choose the city, which he or she is interested in. There are few cities of South Korea provided.

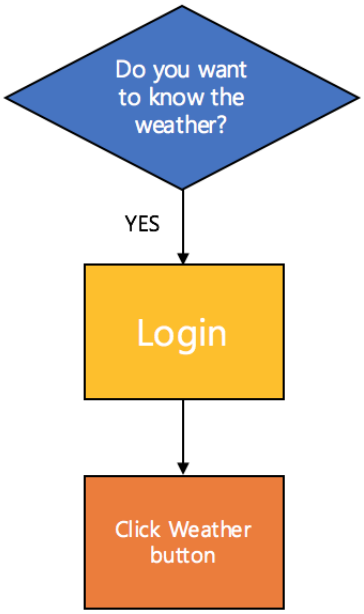

Fig. 31. Use case 3 : Users request for the weather forecast

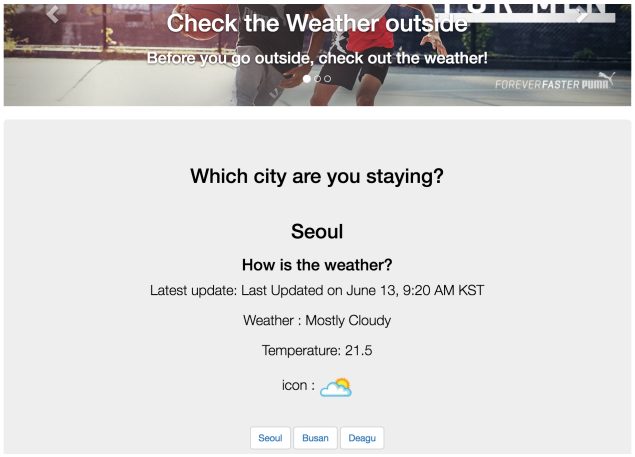

Fig. 32. Weather page

D. *Use case 4 : Users request for Recommendation*

This case covers the needs of the user to receive a recommendation of the shoes. After the use case one, the user is in the website. You can click on the Recommendation tab on the top. In the page of Recommendation, the user can choose the main activity and the city, which he or she is staying or panning to go. After selecting these two, the user submits. Based on the two information, the website provides recommendation points for each shoes.

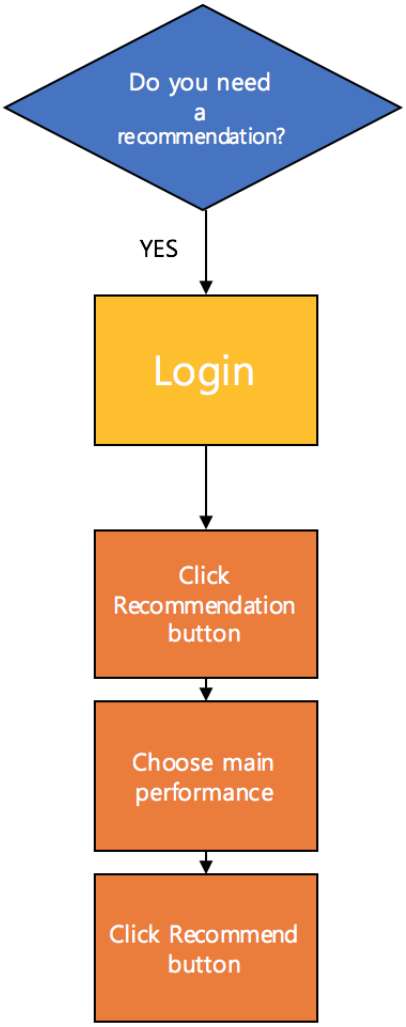

Fig. 33. Use case 4 : Users request for Recommendation

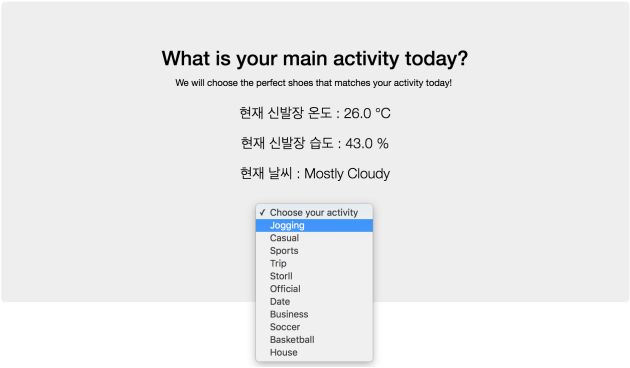

Fig. 34. Recommendation page

E. Use case 5 : Adding the shoes to the Smart Shoebox

This case is to explain the process for the user adding the shoes to the Smart Shoebox. After the use case one, the user is in the website. You can click on the My shoebox tab on the top. In the page of My shoebox, the user can add the items the user possess.

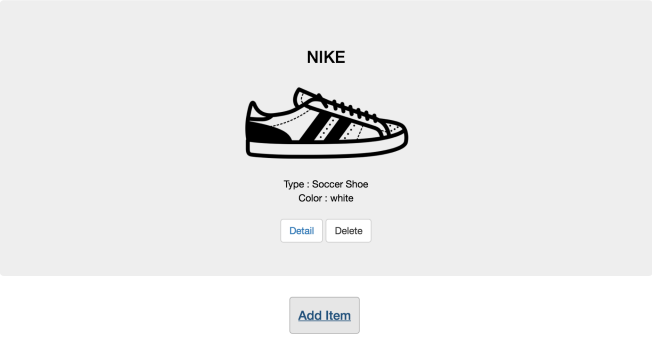

Fig. 35. My Shoebox page

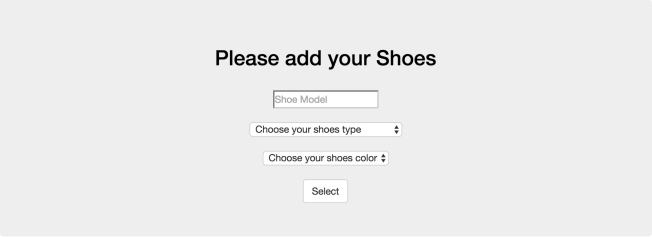

Fig. 36. Adding a new item to Smart Shoebox

F. Use case 6 : Users request for detail information about the shoes

This case covers the needs of the user to see the detail information of the shoes contained in the Smart Shoebox. After the use case one, the user is in the website. After the use case one, the user is in the website. You can click on the My shoebox tab on the top. In the page of My shoebox, the user can choose the shoes he or she is interested in. If the user click the detail button each attached to the added item, the user can see all the detailed additional information.

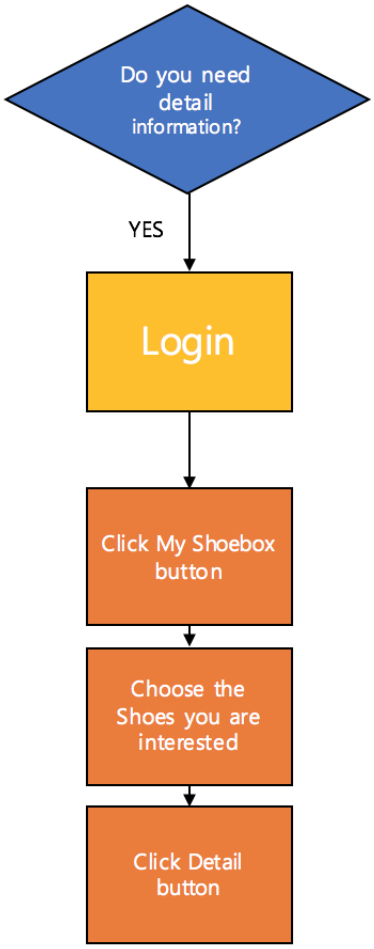

Fig. 37. Use case 6 : Users request for detail information about the shoes

VII. Installation Guide

A. Introduction

This is the Step-by-Step Guide to software installation and maintenance. Actually, Smart Shoebox is activated through the website. Because of this feature of IoT, it seems to be simple and easy for the installation. But still there are several steps to follow to install the Smart Shoebox software.

B. Installation procedure

1) Step1: Arduino Setting and Smart Shoebox Setting: First we have to set the data for the used wifi and password inside the Arduino code. As you can See in Fig. 28. (Android-Hotspot5051 and rbgur123) this part is setting the environment for the Smart Shoeboxes wifi.

```
void setup() {
  Serial.begin(9600);
  esp8266.begin(9600);

  pinMode(11, OUTPUT);
  digitalWrite(11, LOW);

  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);

  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  sendData("AT+RST\r\n",2000,DEBUG);
  sendData("AT+CIOBAUD?\r\n",2000,DEBUG);
  sendData("AT+CWMODE=3\r\n",1000,DEBUG);
  sendData("AT+CWLAP\r\n",3000,DEBUG);

  sendData("AT+CWJAP=\"AndroidHotspot5051\",\"rbgur123!@#\"\r\n",5000,DEBUG);
  sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address - 192.168.43.194
  sendData("AT+CIPMUX=1\r\n",1000,DEBUG);
  sendData("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n",1000,DEBUG);
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80

  dht.begin();
```

Fig. 38. Step1:Arduino Setting and Smart Shoebox Setting

2) Step2: Website Sign-up: After the Arduino part is handled we have the Smart Shoebox in hand. Now we can communicate through website. We have to go into the website and sign up. As you can see in Fig. 29. you can easily sign up by typing email address and password.

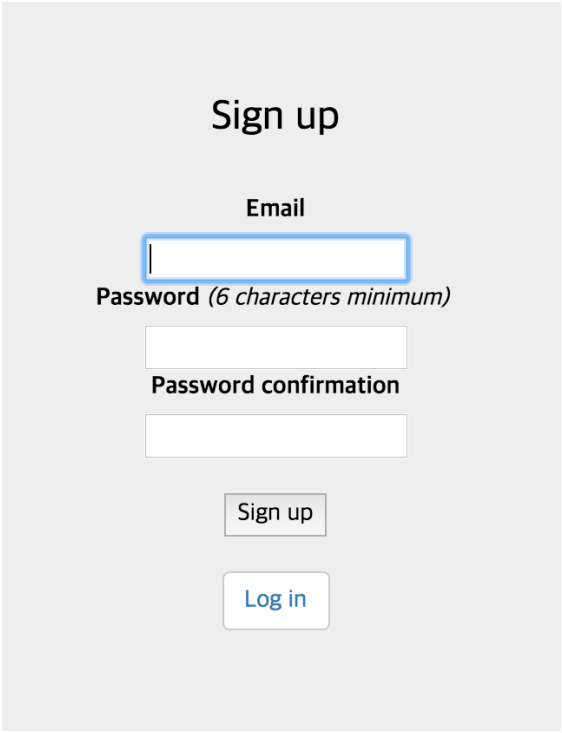

Fig. 39. Step2: Website Sign-up

3) Step3: Login: After we sign up for the Smart Shoebox website, we have to login to the website. Type in the email address and password. We will be in the main page of the website.

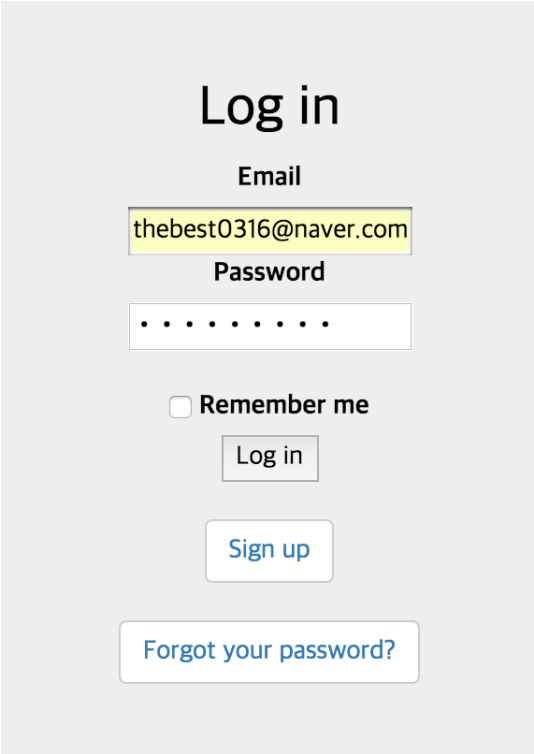

Fig. 40. Step3: Login

4) Step4: Setting the IP address in website: After we login for the Smart Shoebox website, we have to set the IP address in the website. Click the Smart function tab and there is a blank where we can type in the IP address of the Smart Shoebox.

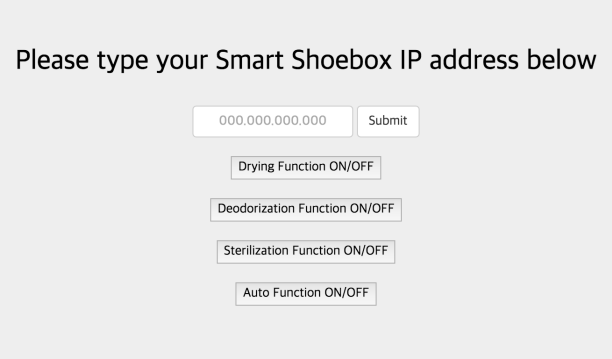

Fig. 41. Step4: Setting the IP address in website

5) Step5: Thing Speak sign up: Thing Speak is the website, help us receive the temperature and humidity of the Smart Shoebox in realtime. We need to sign up for this website also.

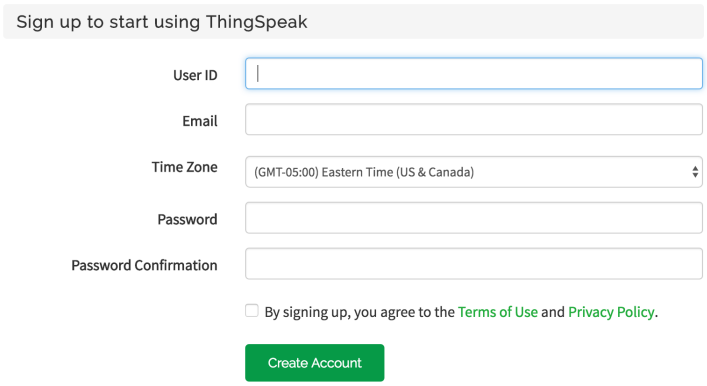

Fig. 42. Step5: Thing Speak sign up

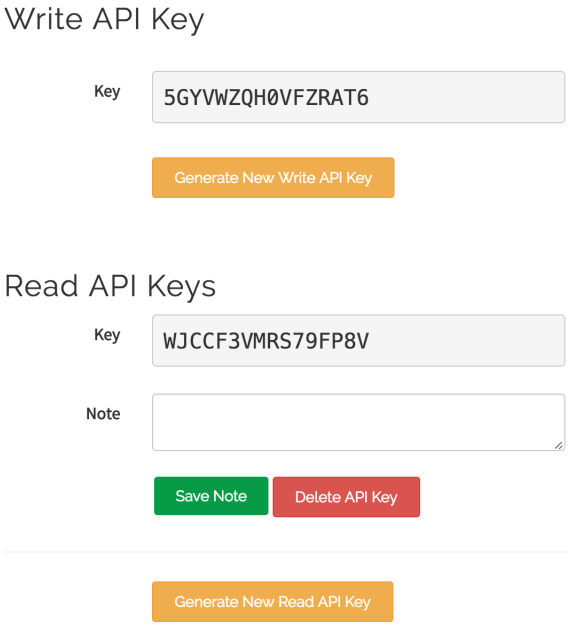

Fig. 44. Step6: Extracting the API key

6) Step6: Extracting the API key: After signing up for the Thing Speak, we have to extract the API key to let the website and Smart Shoebox to communicate. This step will allow the Arduino sensor information to reach Thing Speak, and Thing Speak will send the information to the Smart Shoebox website.

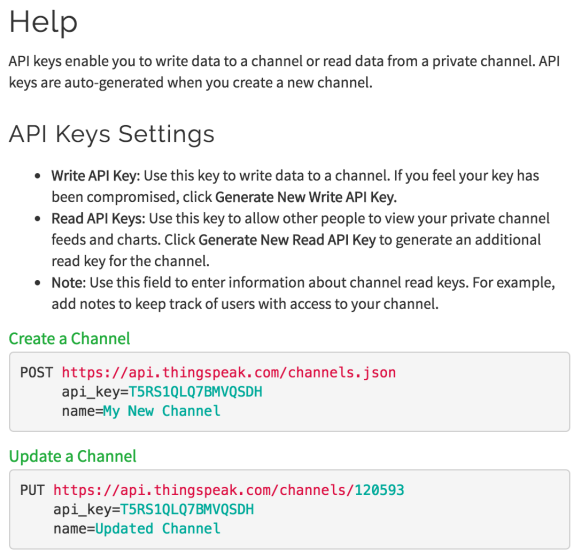

Fig. 43. Step6: Extracting the API key

7) Step7: Inserting the API key into the Website: After letting the communication between Smart Shoebox and Thing Speak and website all in one, we have use the API and channel to show the real time information of the Smart Shoebox. The user can easily see the temperature and humidity changing in the Smart function page.

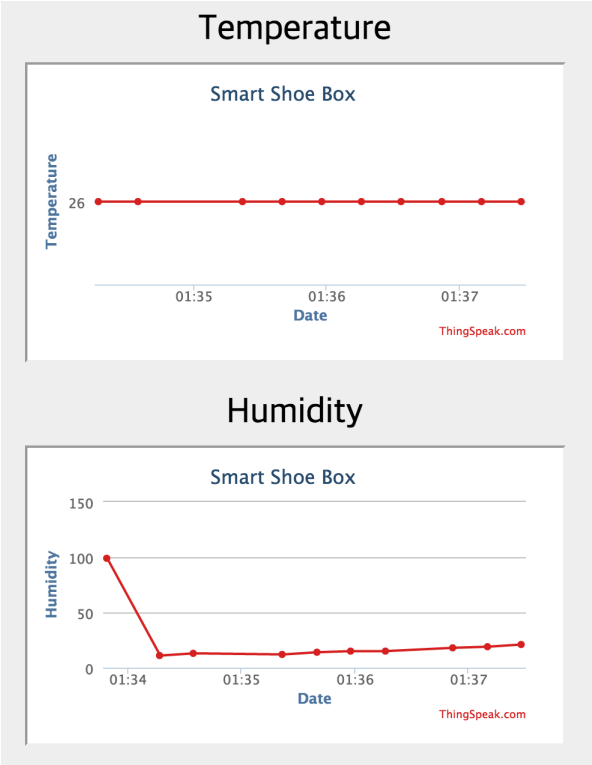

Fig. 45. Step7: Extracting the API key and showing the realtime graph

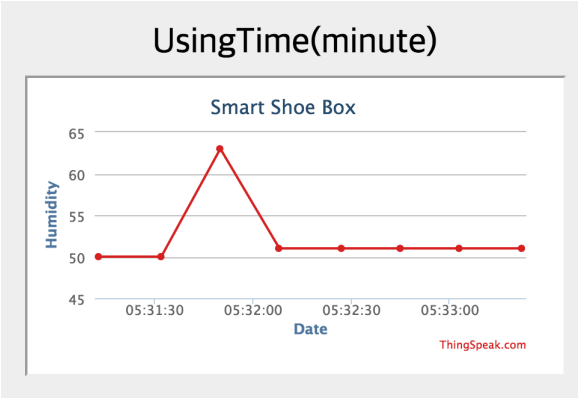

Fig. 46. Step7: Extracting the API key and showing the realtime graph

VIII. Discussion

First of all, experiencing a team project was a biggest and helpful thing for all of the team mates. Before this Software Engineering course, we were all not used to communicating and cooperating for one goal, to make a working software. 3 months of process, thinking up of an interesting theme, thinking up for requirements, specifications and so on, was really helpful for all the team mates to become a better software engineer. The most difficult part was communicating with each other. There were a lot of cases when we were thinking of different features, while we thought we were thinking the exact same thing. We had to keep talk and show our thoughts to each other to manage and cooperate.

Also learning a new programming language like Ruby, Bootstrap, Arduino was so hard and we spent a lot of time to understand the basics on the background. We all thought that making a software was something we can easily do with only the concepts we learn in other courses, but it was not such a thing. We need to search and look up for more specific and up to date information all the time. Now as we know how to struggle to look for what we need, we might be able to learn and do something progressed than what we have done right now for the project.