North South University

Project Report

Course: CSE427

Section: 1

Semester: Spring 2019

Instructor: Shaikh Shawon Arefin Shimon

Submission Date: 30 April 2019

Name	Id
Md. Amdadul Bari	1510813042

Abstract

Krishok is an IoT based farming solution for Bangladeshi farmers to grow better amount of crops with better quality. Another outcome of this project is to help the farmers to face any kind of unknown condition which they did not face in their farming life. The whole system is automated where the device will notify the the suggestions to the farmers by analysing the data which it will get from it's sensors, cameras, learning, and analytical capability. The system will also help the farmers by suggesting and predicting which types of crop will be helpful for attaining better profit under the range of options. It will also guide a farmer towards a good solution of that crops. The future scope of the project is, it will open job opportunity with a minimum budget but maximum profit in reply. The opened job opportunity of the project will be for those people who have a little technical knowledge.

System Architecture

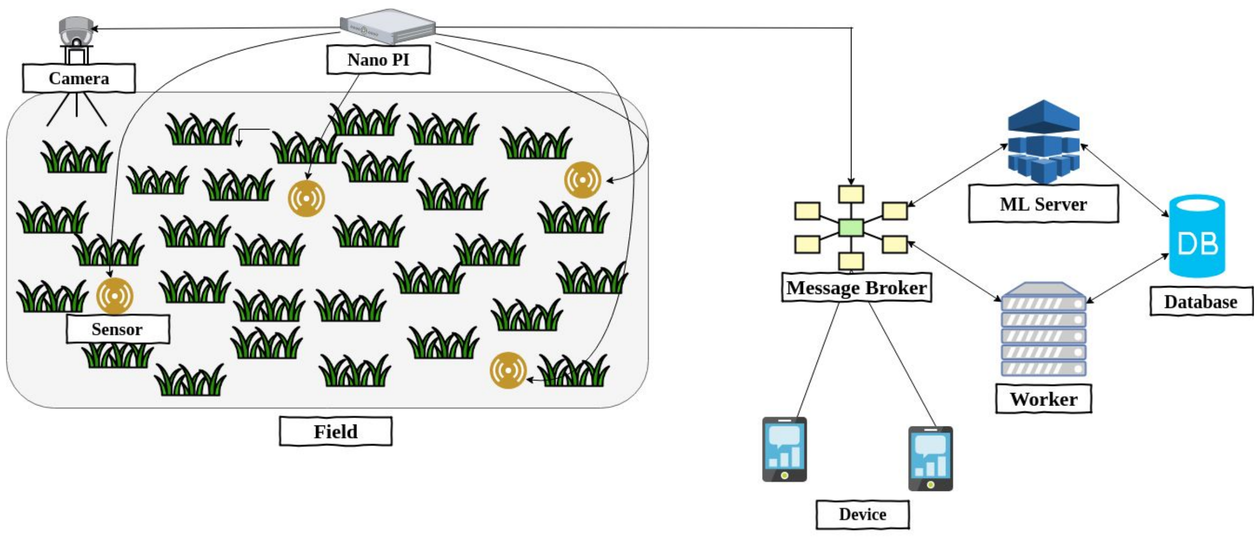


Fig 1 : System architecture of KrishokIoT

The system is controlled by a raspberry pi. Some sensors are connected with the raspberry pi, and the pi collects data from the sensors and sends the data to the worker. The worker will process those data, insert them in the database if needed and send some of the data to ML servers. Then the ml servers send the response back to the worker and the worker sends back the data to the pi and other connected android devices of the farmer as a command, alert or report. For keeping the communication stable and lightweight MQTT protocols has been used and to stream the video to the cloud , we have used WebRTC based Kurento Media Server.

Input Space Partitioning:

Mainly these four type of data type is used in the backend of the system. For the types input spaces are partitioned as the following table.

Table of Input Space Partition:

Data Type	Block
Integer	Positive Number
	Negative Number
	Zero
Double	Positive Number
	Negative Number
	Zero
Float	Positive Number
	Negative Number
	Zero
String	Empty and only space
	Small Text
	LargeText
	Null

Example of Input Space Partition:

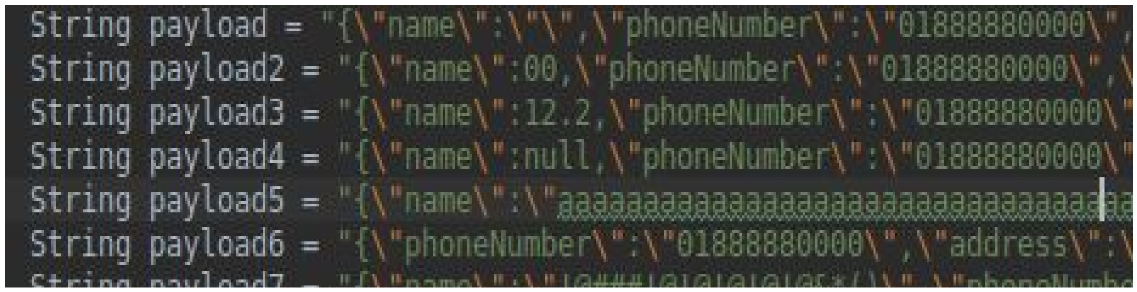
Data Type	Range Type	
Integer	Positive Number	1,100,150
	Negative Number	-1500,5233
	Zero	0
Double	Positive Number	11.22,25.22
	Negative Number	-58.22,85.55
	Zero	0
Float	Positive Number	55.5f,5589.245f
	Negative Number	-98.69f,-69.55f
	Zero	0.00f
String	Empty and only space	“” , “ “
	Small Text	“NSUCSE427”
	LargeText	“aaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaa”
	Null	null

Annotation used for Input Space Partition:

1. Parametrized
2. Theory

3. Data Point
4. Test
5. Before - After
6. Setup

Example of Input Space Partition from Code:



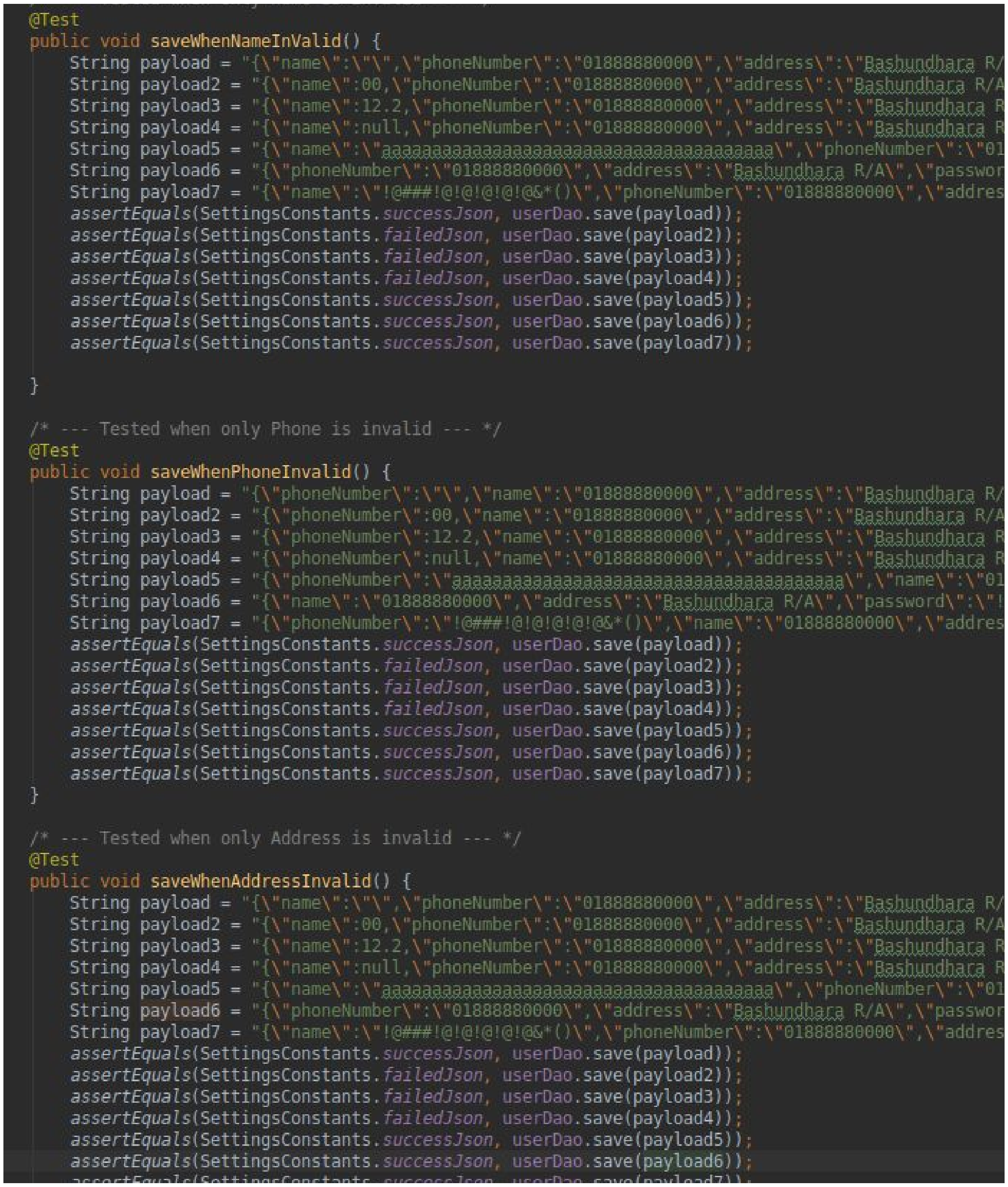
```
String payload = "{\"name\":\"\",\"phoneNumber\":\"01888880000\",
String payload2 = "{\"name\":00,\"phoneNumber\":\"01888880000\",\
String payload3 = "{\"name\":12.2,\"phoneNumber\":\"01888880000\"
String payload4 = "{\"name\":null,\"phoneNumber\":\"01888880000\"
String payload5 = "{\"name\":\"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
String payload6 = "{\"phoneNumber\":\"01888880000\",\"address\":\
```

Fig 2: Input Space Partitions

Above code is a part of test method [Class: UserDaoTest - Line:55] of Valid/Invalid username checker. We can see in the string different types of value for the name key is present. Almost all possible type inputs are given to check whether the method execute successfully every time or not.

The test methods are implemented in seperate functions to keep the simplicity of the code.

This type of test cases are used in all test methods.



```java
@Test
public void saveWhenNameInValid() {
    String payload = "{\"name\":\"\",\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/
    String payload2 = "{\"name\":00,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/A
    String payload3 = "{\"name\":12.2,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R
    String payload4 = "{\"name\":null,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R
    String payload5 = "{\"name\":\"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\",\"phoneNumber\":\"01
    String payload6 = "{\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/A\",\"passwor
    String payload7 = "{\"name\":\"!@###!@!@!@!@!@&*()\",\"phoneNumber\":\"01888880000\",\"addres
    assertEquals(SettingsConstants.successJson, userDao.save(payload));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload2));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload3));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload4));
    assertEquals(SettingsConstants.successJson, userDao.save(payload5));
    assertEquals(SettingsConstants.successJson, userDao.save(payload6));
    assertEquals(SettingsConstants.successJson, userDao.save(payload7));

}

/* --- Tested when only Phone is invalid --- */
@Test
public void saveWhenPhoneInvalid() {
    String payload = "{\"phoneNumber\":\"\",\"name\":\"01888880000\",\"address\":\"Bashundhara R/
    String payload2 = "{\"phoneNumber\":00,\"name\":\"01888880000\",\"address\":\"Bashundhara R/A
    String payload3 = "{\"phoneNumber\":12.2,\"name\":\"01888880000\",\"address\":\"Bashundhara R
    String payload4 = "{\"phoneNumber\":null,\"name\":\"01888880000\",\"address\":\"Bashundhara R
    String payload5 = "{\"phoneNumber\":\"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\",\"name\":\"01
    String payload6 = "{\"name\":\"01888880000\",\"address\":\"Bashundhara R/A\",\"password\":\"!
    String payload7 = "{\"phoneNumber\":\"!@###!@!@!@!@!@&*()\",\"name\":\"01888880000\",\"addres
    assertEquals(SettingsConstants.successJson, userDao.save(payload));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload2));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload3));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload4));
    assertEquals(SettingsConstants.successJson, userDao.save(payload5));
    assertEquals(SettingsConstants.successJson, userDao.save(payload6));
    assertEquals(SettingsConstants.successJson, userDao.save(payload7));
}

/* --- Tested when only Address is invalid --- */
@Test
public void saveWhenAddressInvalid() {
    String payload = "{\"name\":\"\",\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/
    String payload2 = "{\"name\":00,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/A
    String payload3 = "{\"name\":12.2,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R
    String payload4 = "{\"name\":null,\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R
    String payload5 = "{\"name\":\"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\",\"phoneNumber\":\"01
    String payload6 = "{\"phoneNumber\":\"01888880000\",\"address\":\"Bashundhara R/A\",\"passwor
    String payload7 = "{\"name\":\"!@###!@!@!@!@!@&*()\",\"phoneNumber\":\"01888880000\",\"addres
    assertEquals(SettingsConstants.successJson, userDao.save(payload));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload2));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload3));
    assertEquals(SettingsConstants.failedJson, userDao.save(payload4));
    assertEquals(SettingsConstants.successJson, userDao.save(payload5));
    assertEquals(SettingsConstants.successJson, userDao.save(payload6));
    assertEquals(SettingsConstants.successJson, userDao.save(payload7));
```

Fig 3: Input Space Partitions

Graph Partitioning

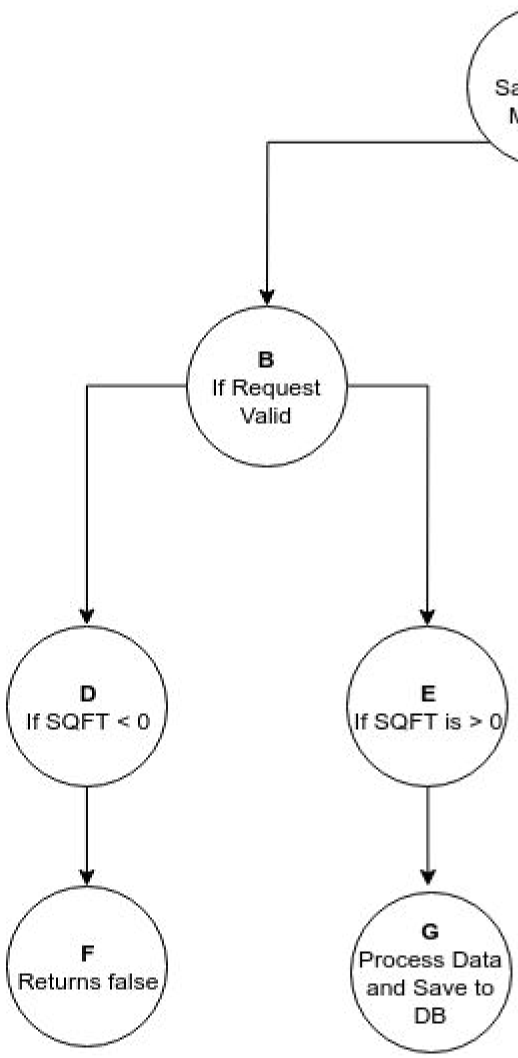


Fig 4 : Graph of Land Save method

It is a graph of a land saving api of the backend of this project. If the API gets any valid hit then it goes deeper and check the body of that request. After that if it found that the the square feet parameter is less than zero then it immediately returns false ,

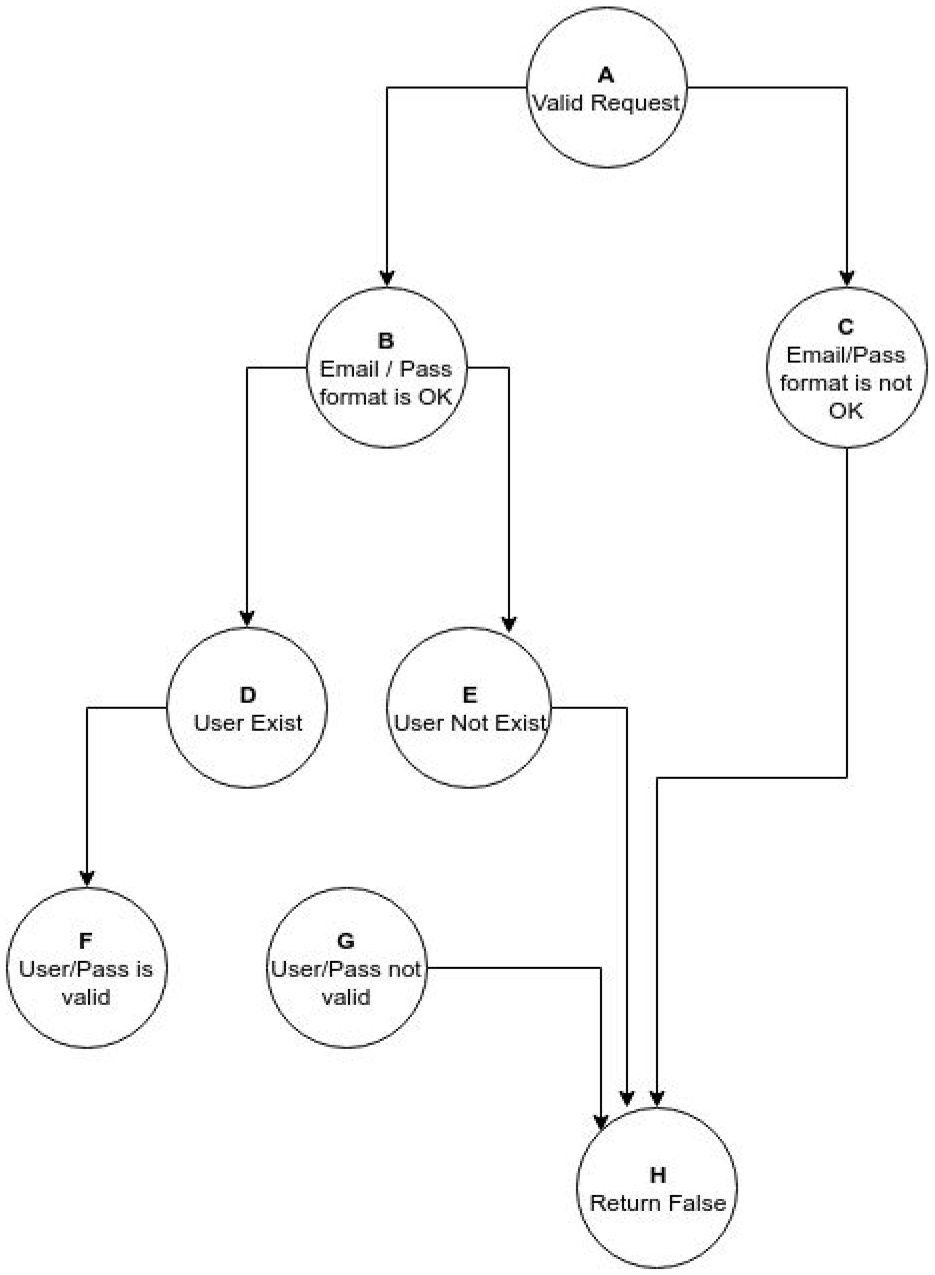


Fig 5: Graph of Login Feature

Fig 5 is the graph of login module of the API. If the API gets any valid hit in the login endpoint then it checks whether the email and password format is right or not. If it is right then it checks whether the user exists or not as shown in the graph.
If the user exists then it checks if the user/pass combination is valid. If valid it logged in the user, otherwise declines the request.

UI Testing

UI TESTING FOR ANDROID

espresso

Espresso is a testing framework for Android to make it easy to write reliable user interface tests. Espresso test has been used to test action with the user interface. It also allows the test wait until all background activities have finished. It comes with the android studio .It is used in this project because it is very stable and easy to use.