

**IoT Enabled Smart Farming System**

Software Design Document

Names : Kamitha Bodhinayake

Chamith Mawela

Jinadi Rashmika

Workstation : Institute of Java and Software Engineering

Date : 01/27/2017

**TABLE OF CONTENTS**

[1. INTRODUCTION 2](#_Toc473297703)

[1.1 Purpose 2](#_Toc473297704)

[1.2 Scope 2](#_Toc473297705)

[1.3 Overview 2](#_Toc473297706)

[1.4 Definitions and Acronyms 3](#_Toc473297707)

[2. SYSTEM OVERVIEW 4](#_Toc473297708)

[3. SYSTEM ARCHITECTURE 5](#_Toc473297709)

[3.1 Architectural Design 5](#_Toc473297710)

[4. DECOMPOSITION DESCRIPTION 7](#_Toc473297711)

[5. DATA DESIGN 9](#_Toc473297712)

[5.1 Data Description 9](#_Toc473297713)

[5.2 Data Dictionary 10](#_Toc473297714)

[5.3 ER Diagram 11](#_Toc473297715)

[6. HUMAN INTERFACE DESIGN 12](#_Toc473297716)

[6.1 Overview of User Interface 12](#_Toc473297717)

[6.2 Screen Images 12](#_Toc473297718)

[6.3 Screen Objects and Actions 13](#_Toc473297719)

[7. REQUIREMENTS MATRIX 15](#_Toc473297720)

[8. APPENDICES 15](#_Toc473297721)

# INTRODUCTION

## Purpose

The purpose of this document is to adduce a detailed description of the designs of the *IoT Enabled Smart Farming System* which is going to be created for remote controlling agricultural lands. This document is intended for our team to use the designs as guidelines to implement the project and also for the team’s instructor, Mr. Namal Yasanga and clients of this project Mr. Nuwan Gomas and Mr. Gihan as it fulfils one of the requirements of the project. Lastly, this document could be used for designers who try to upgrade or modify the present design of the smart farming system.

## Scope

Our project aims at developing an *IoT Enabled Smart Farming System* to control moisture of the soil remotely. We have a limited time period to design, implement and test our system and it is actually a very big challenge. Therefore, we have decided to design the system only for controlling the moisture condition of the agriculture land via a web application.

We are planning to invent a smart device which can be established in the agricultural land to collect the details soil moisture, temperature and humidity of the land. The device consists of sensors and other required components to measure the moisture, temperature and humidity of the land and inform the user. Then the owner of the farming land can review the condition of the land and supply enough water to the land.

It is very easy as the farmer does not need to visit his or her farming land frequently. But the owner is capable of controlling the moisture level remotely just through connecting to the web. It just need to have a Wi-Fi connection to get required current condition of the farming land and control it remotely.

This document gives a detailed description of the software architecture of the smart farming system. It specifies the structure and design of some of the modules discussed in the previous documents. It also displays some of the use cases that had transformed into sequential and activity diagrams.

## Overview

This document is written according to the standards for Software Design Documentation explained in “IEEE Recommended Practice for Software Design Documentation”.

Sections 3 – 5 contain discussions of the designs for the project with diagrams, section 6 shows samples of UI from the system, and section 7 contains the UML diagrams. The appendices contain the setup and configuration needed for the UUIS, a list of functions that are implemented in this version, and that are to be implemented in future version, and a list of tools and environment used in the entire project, along with the time contribution of team members. The appendices also include the test report and test cases.

## Definitions and Acronyms

IoT: Internet of Things

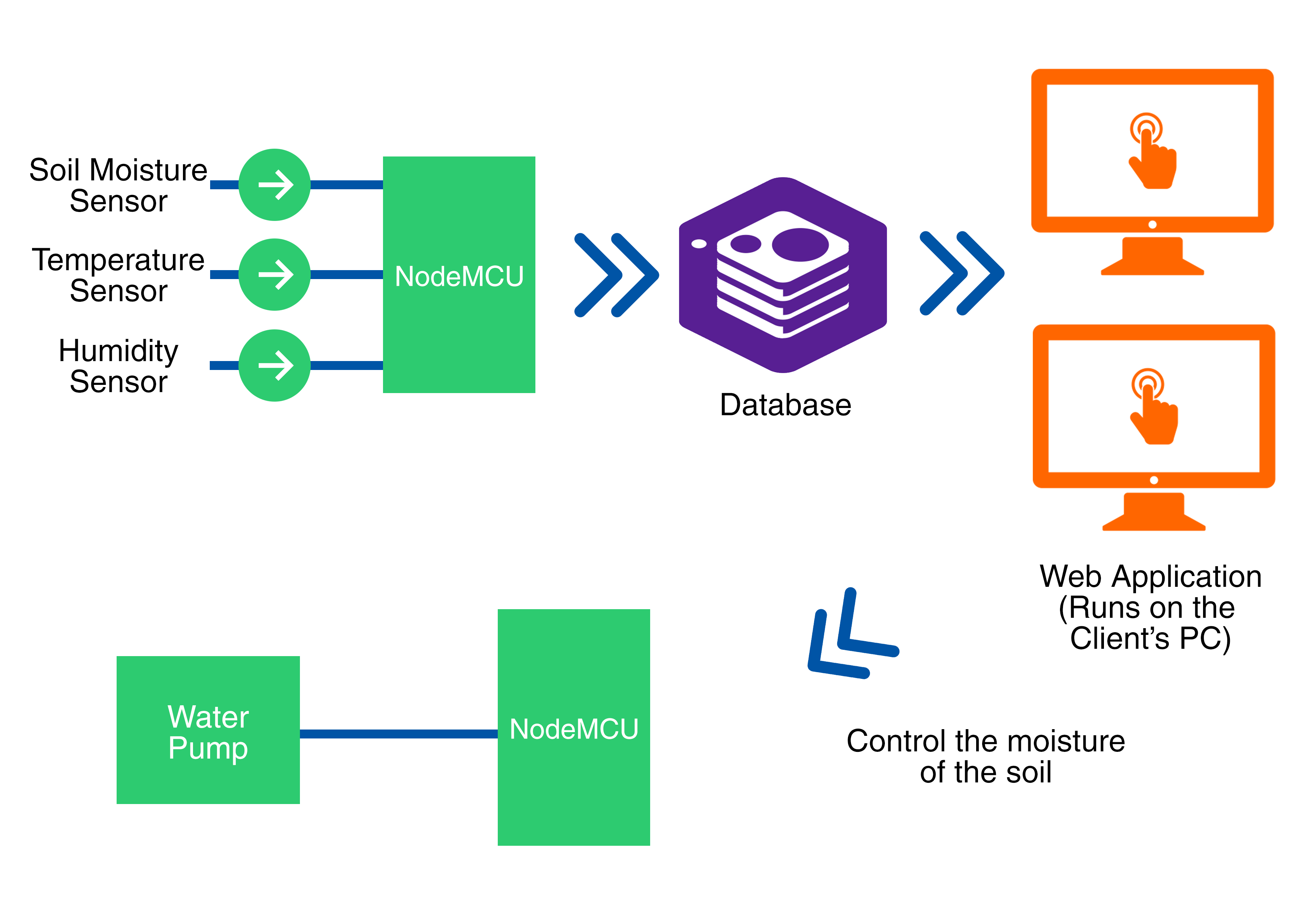
UML: Unified Modelling Language

UI: User Interface

IEEE: Institute of Electrical and Electronic Engineers

# SYSTEM OVERVIEW

Figure 1 - System overview



In this project we basically use three types of sensors as presented in the above figure;

* Soil Moisture Sensor
* Temperature Sensor
* Humidity Sensor
* Water Controlling Sensor

The data acquired from these sensors are detected by NodeMCU and those data is transferred to the database through Wi-Fi.

Then by login to the relevant website, the user can review all the current details and data history of his/her farm.

As well the user can control the moisture of the soil by providing required water quantity to the agricultural land. This is done by special sensor (Water Controlling Sensor) which is connected to the water supply system in the farm. The command that user gives is detected by NodeMCU via Wi-Fi and gives the command to the water supply system.

# SYSTEM ARCHITECTURE

## Architectural Design

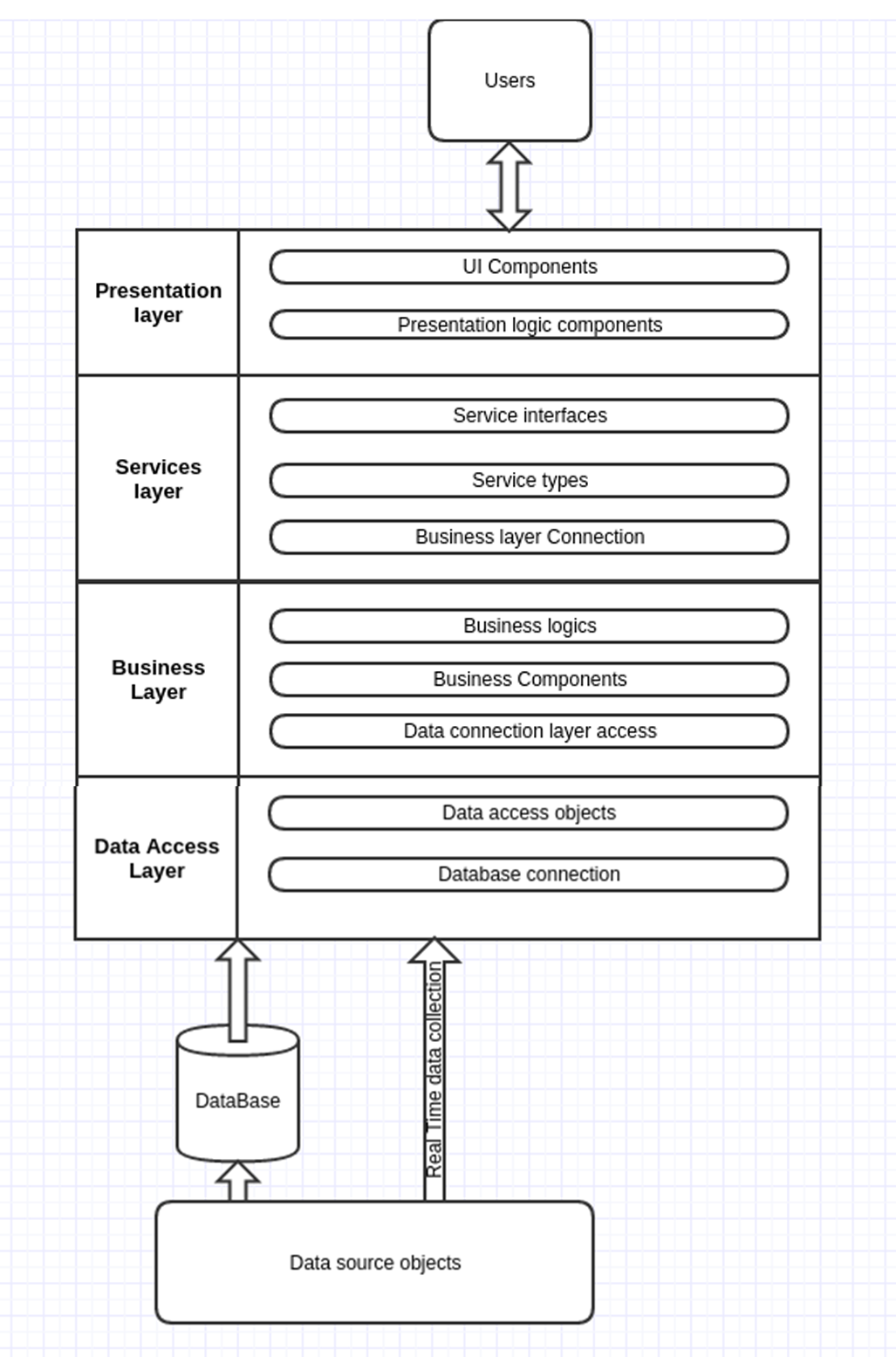


Figure 2 - Three tier architecture

Here we have used the three tier architecture to implement the IOT project. We have presentation layer, which we are going to implement using html and CSS technologies. It is the interface which the user is going to see. Then comes the service layer, which we are going to implement using servlet, jsp and ajax technologies.

The duty of the service layer is to access services for the presentation layer and the and connect with the business layer.

Then we have the business layer, where business logics are written as expected by the client and to the necessity of analyzing parts. Also business layer is responsible for the accessing of data layer and process those data with the logics provided and sending those data to the services layer.

Next, the data layer is responsible for the extracting data from the databases where data is stored from the data devices like sensors and other electronic equipment. But also we have to produce data to the data layer itself through Wi-Fi technologies, in order to get real time data for the representation in the presentation layer.

# DECOMPOSITION DESCRIPTION

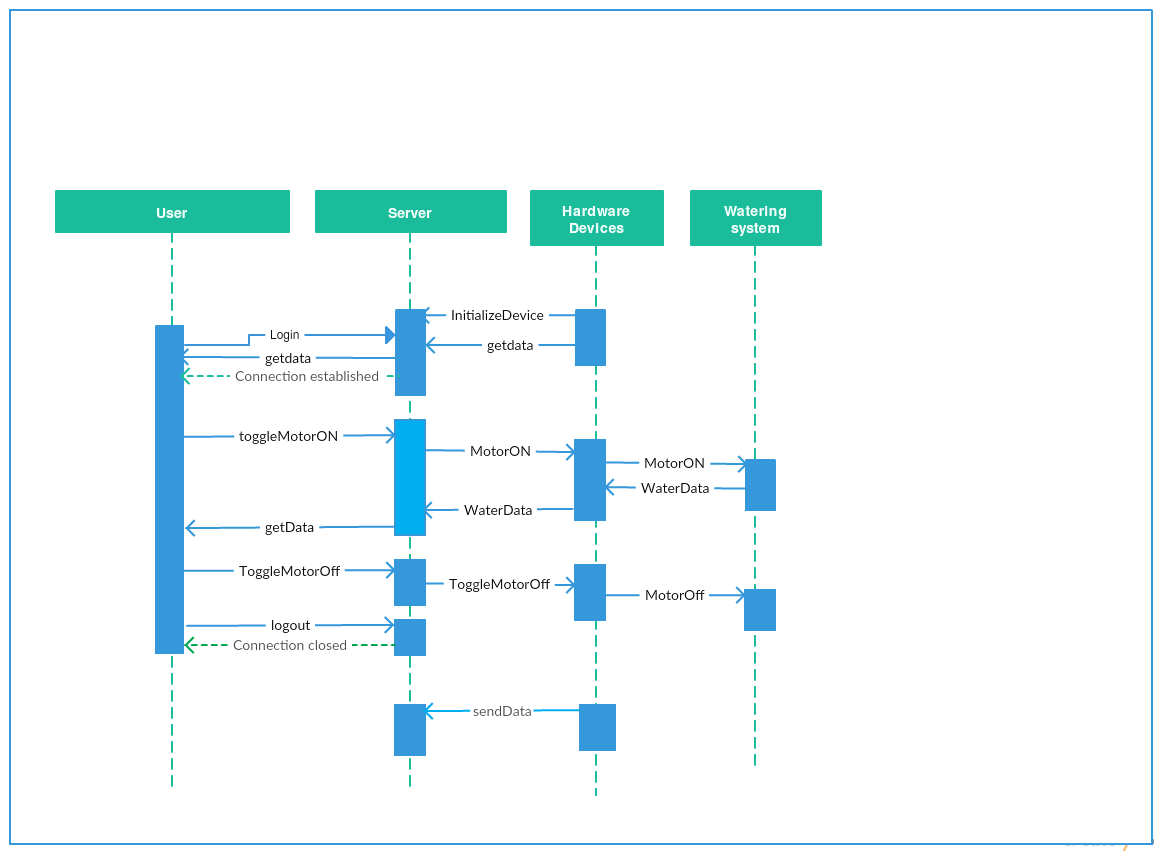


Figure 3 - Sequence diagram

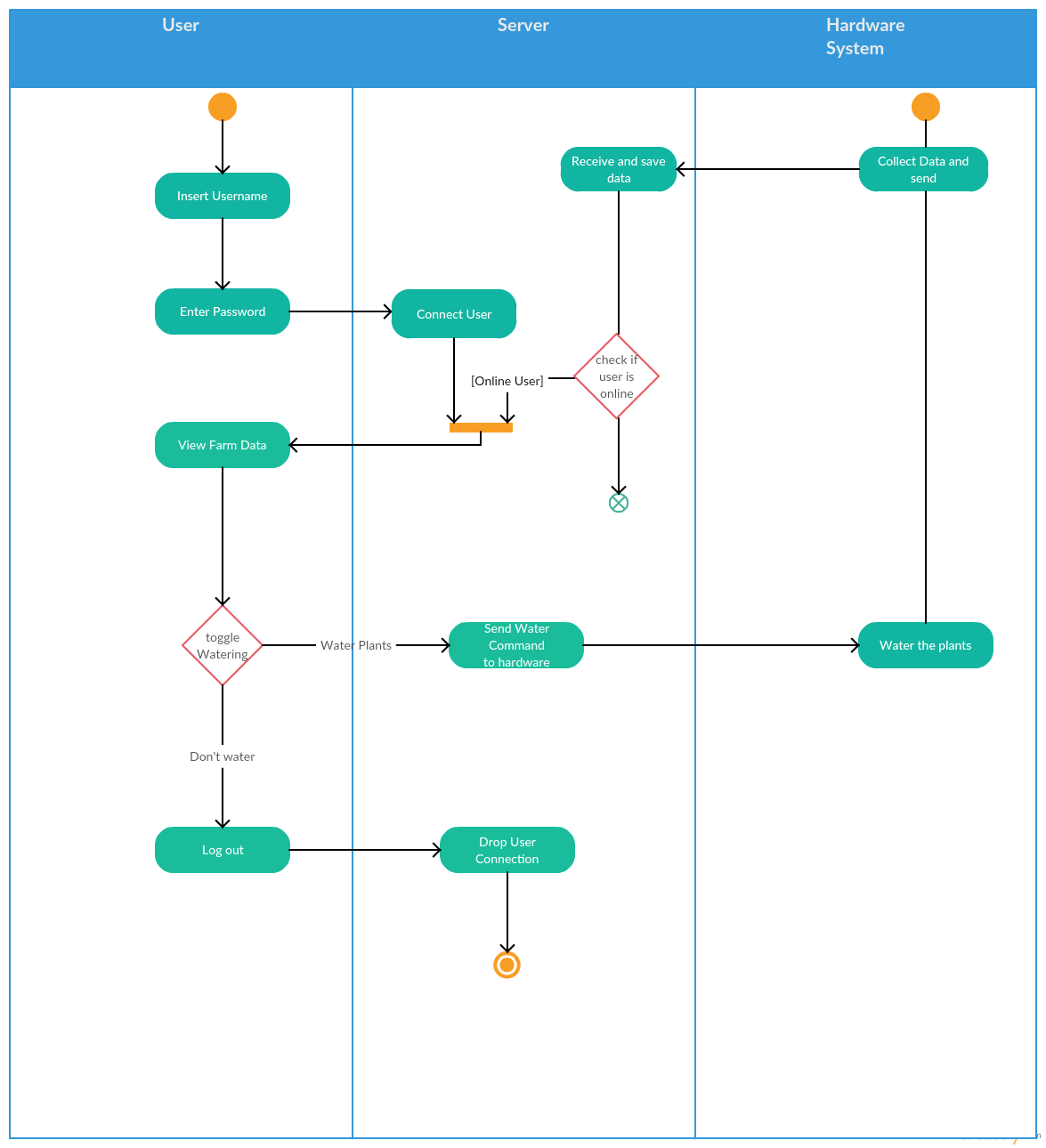


Figure 4 - Activity diagram

# DATA DESIGN

## Data Description

MySQL database and JDBC to communicate with the database that is installed locally on the server.

**UserDTO:**

public class userDTO {

private String UserID;

private String Name;

private String Password;

public userDTO() {

}

public userDTO(String UserID, String Name, String Password) {

this.UserID = UserID;

this.Name = Name;

this.Password = Password;

}

}

**averagedailydetailDTO:**

public class averagedailydetailDTO {

private Date DateSaved;

private int AvgTemp;

private int AvgHum;

private double WaterUsage;

public averagedailydetailDTO() {

}

public averagedailydetailDTO(Date DateSaved, int AvgTemp, int AvgHum, double WaterUsage) {

this.DateSaved = DateSaved;

this.AvgTemp = AvgTemp;

this.AvgHum = AvgHum;

this.WaterUsage = WaterUsage;

}

}

**todayweatherdetailDTO:**

public class todayweatherdetailDTO {

private Date TimeSaved;

private Date DateSaved;

private String FarmID;

private int Temperature;

private int Humidity;

public todayweatherdetailDTO() {

}

public todayweatherdetailDTO(Date TimeSaved, Date DateSaved, String FarmID, int Temperature, int Humidity) {

this.TimeSaved = TimeSaved;

this.DateSaved = DateSaved;

this.FarmID = FarmID;

this.Temperature = Temperature;

this.Humidity = Humidity;

}

}

**farmDTO:**

public class farmDTO {

private String FarmID;

private String UserID;

private String Description;

public farmDTO() {

}

public farmDTO(String FarmID, String UserID, String Description) {

this.FarmID = FarmID;

this.UserID = UserID;

this.Description = Description;

}

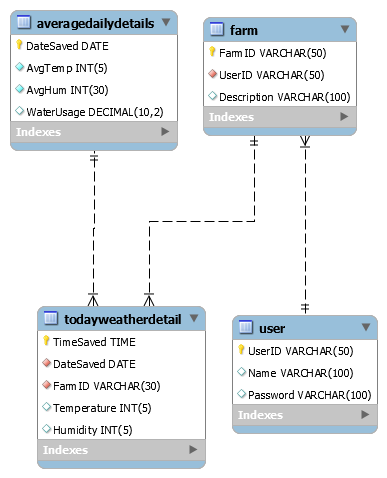
}

## Data Dictionary

Data Dictionary Excel file attached to this document.

## ER Diagram

Figure 5 - ER Diagram



# HUMAN INTERFACE DESIGN

## Overview of User Interface

UI is designed according to UI design principles.

The structure principle: UI is organized in such a way that related things are combined together and unrelated things are separated.

The simplicity principle: It is easy to follow the provided interface. In the case of mistake, system displays error message.

The visibility principle: All system functions are available through UI. It does not overwhelm users with too many alternatives.

The feedback principle: Through the system of messages, the design keeps users informed of actions, errors, or exceptions.

## loginScreen Images

Figure 6 - Login Screen

Figure 7 - Main page of the webpage



## Screen Objects and Actions

Current Temperature and Humidity: The current temperature and humidity of the farm were displayed in here. It’s synced with the sensors.

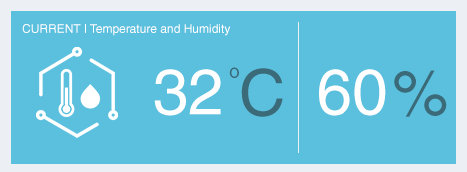


Figure 8 - Current Temperature and Humidity

Weather details: Weather forecast display in here taken from google weather.

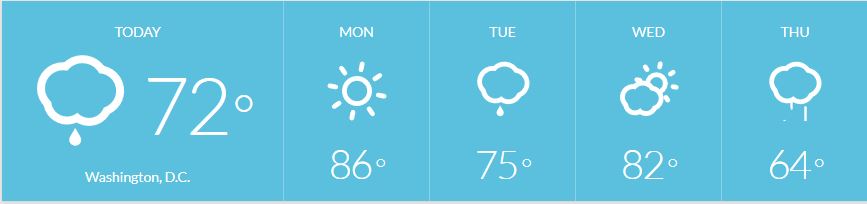


Figure 9 - Weather forecast

Water Level: Current state of the soil moisture displays here. And a reminder given for the user if there is any issue in water inside the farm. And user can control the water supply from here.

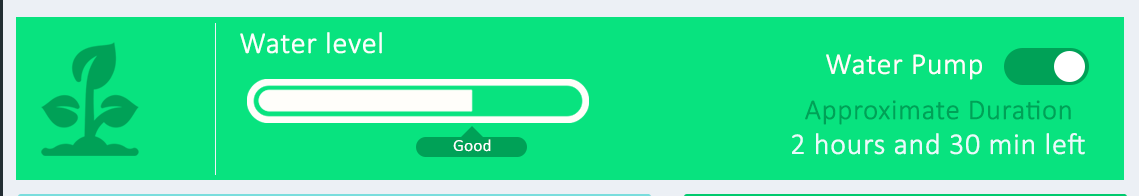


Figure 10 - Water level

Water Usage: Previous week water usage display in here as a graph. Therefore, user can get a rough idea about today water usage.

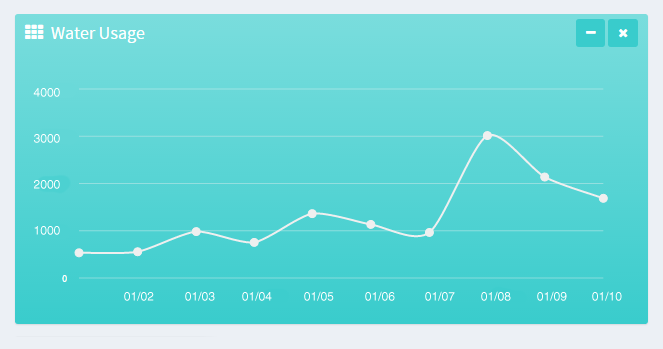


Figure 11 - Water usage

# REQUIREMENTS MATRIX

|  |  |
| --- | --- |
| **Functional Requirement** | **System Components which satisfies the relevant functional requirements** |
| Take the weather conditions of the farm | Using sensors take the relevant reading and visualize to the user by using Web Application. |
| Detect soil moisture of the farm | Using soil moisture sensor take the readings and visualize them on the Web Application with the status. |
| Control the moisture level of the soil | By analyzing the moisture level user can control the water supply to the farm remotely. |

# APPENDICES

http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee­SDD.pdf