**Chapter 1**

**Introduction**

**1.1 Introduction**

The biggest problem faced during production of crops, leading to wastage or below par production is non-timely watering in the field or inaccurate amount of water being poured in the field. At times, due to the human tendency, either greater or lesser amount of water is allowed to enter the field thereby destroying the crop. This marks the first major problem. Also water-level in the source tank sometimes goes low or sometimes get over-drained. Thus information regarding scarcity or abundance of water in the reservoir is the second major problem. Many times the farmer is far away from the field and is therefore unable to get the current status of the field. Hence his periodic visit is must on the field to take care of the water requirement, chemical requirement, and other production related issues. Thus for timely observation, automatic control over such parameters would ease the burden of any individual.

With this project, the current problems related to farming are solved and practically implemented solutions are provided. Using Internet of Things (IoT) a whole new concept of farming using networks is introduced reducing labor; updating farmer about the live conditions of farm on the mobile devices and providing records of their farm on c9 cloud. It makes the process handy with the click a button reformation.

**1.2 Problem Statement**

To design and develop a system which is smart, providing automatic control and monitoring of field activities through an android application.

**1.3Objective**

To provide complete solution to field activities, irrigation problems and storage problems using remote controlled robot, smart agriculture system respectively.

This system avoids over irrigation, under irrigation, top soil erosion and reduce the wastage of water. The main advantage is that the system’s action can be changed according to the situation (crops, weather conditions, soil etc.).

IoT sensors are capable of providing farmers with information will help to increase the yield and will offer precise data which can be used to improve farming techniques over time.

**1.4 Methodology**

The project basically has three modules:

**Module 1: Data Collection and Cloud Storage**

The data from soil moisture sensor, temperature sensor and IR sensor will be sensed and monitored by arduino. Arduino UNO has been provided with GSM(Global System for Mobile Communication) modem, providing internet facility. The data collected from arduino will be stored on cloud.

**Module 2: Automatic Control System**

When the moisture and the temperature data is less than the threshold value arduino will send a signal to the motor. The motor will pump the water into the field.

**Module 3: Android Application**

An android application is developed to check the on-going field activities. When the user sets the system, he has to register himself and has to enter the field information like name, address, and city and phone number. When data from the sensors are stored on the cloud, the person can access the data through the android application.

**1.5 Limitations**

1. The entire system is built on arduino uno hence the proper working of arduino is must.
2. The system is completely dependent on the IoT sensors. Hence the sensors should always be in ready to work condition.
3. The user can have access to the system only when provided with proper network facility.
4. The system is available all the time except when the server is down.

**Chapter 2**

**Literature survey**

**2.1 Need for Research**

India, who’s 17% of GDP (Gross Domestic Profit) depends on agriculture, is not a developed nation in terms of modernization in agriculture. The high cost of labor, uncertainty in the production of crops, lack of knowledge about new methods, continuing with the same orthodox and traditional means to go about agriculture, the inefficient use of proper irrigational facilities results in low productivity.  Due to this uncertainty in the irrigation process the crops may also dry up. By conducting proper research, the current problems related to farming are solved and practically implemented solutions are provided.

**2.2 Existing Approach**

Irrigationis an artificial way of supplying water to the land or soil. It is used to assist in the growing of agricultural crops, maintenance of landscapes, and re-vegetation of disturbed soils in dry areas and during periods of inadequate rainfall. Additionally, irrigation also has a few other uses in crop production, which include protecting plants against frost, suppressing weed growth in grain fields and preventing soil consolidation.

Irrigation is mostly done using canal systems in which water is pumped into fields after regular interval of time without any feedback of water level in field. This type of irrigation affects crop health and produces a poor yield because some crops are too sensitive to water content in soil.

**Traditional Methods of Irrigation**

**1. Rahat (Persian wheel)**

Rahat is a traditional irrigation system, also known as the Water Wheel or the Persian wheel; it is made of a chain of buckets attached to a wheel which is then moved by another wheel using human or animal power.

**2. Chain Pump**

The chain pump is type of a water pump in which several circular discs are positioned on an endless chain. One part of the chain dips into the water, and the chain runs through a tube, slightly bigger than the diameter of the discs.

**Modern Methods of Irrigation**

**1. Sprinkler System**

Sprinkler Irrigation is a method of supplying irrigation water which is similar to rainfall. Water is distributed through a system of pipes usually by pumping. It is then sprayed into the air and irrigated entire soil surface through spray heads so that it breaks up into small water drops which fall to the ground.

**2. Drip System**

Drip irrigation is a type of micro-irrigation, where the soil has the potential to save water and nutrients by allowing water to drip slowly to the roots of plants, either from above the soil surface or buried below the surface. Drip irrigation systems distribute water through a network of valves, pipes, tubing, and emitters.

**2.3 Proposed Approach**

This project presents a smart system that uses a low cost soil moisture sensor and temperature sensor to control water supply in water deficient areas. Data acquired from all the sensor nodes is sent through wireless communication modules to a centralized server that controls water supply. The farmer will get the data from both the systems and can have access to field activities using an android application.

**Chapter 3**

**Software Requirement Specification**

The aim of this document is to provide complete, consistent, unambiguous, verifiable, modifiable SRS which forms the basis of Software Development.

**3.1 Hardware Interface**

**1. Moisture Sensor**

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

**2. Temperature Sensor**

DTH11 sensor is used to measure the temperature and humidity around the field. It measures temperature in degree Celsius (°C) and humidity in percentage (%) in the environment and sends the data to the arduino.

**3. IR Sensor**

An infrared sensor (IR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in IR-based motion detectors and thus help in detecting the intrusion occurring in the field land.

**4. Arduino UNO**

**Arduino**is open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world.

The arduino uno board is an important component of this project as it controls and co-ordinates the entire smart agriculture system.

**5. LCD Display**

A 16x2 LCD (Liquid Crystal Display) with arduino uno is provided to notify the user about the status of the motor.

**3.2 Software Interface**

**1. Arduino Software (IDE)**

It is used to dump the code of the given sensor onto the arduino kit so as to check the moisture level in the soil and to find the humidity in the temperature.

**2. Android SDK**

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in Java programming language using the Android software development kit (SDK), but other development environments are also available.

**3. c9 Cloud Platform**

The IoT service will work across a range of protocols. The c9 cloud platform will help the user of the system safeguard their field data on timely based manner.

* 1. **Memory Constraints**
* A 64 GB memory android phone.
* c9 Cloud platform is further used to store the sensor data.

**Chapter 4**

**Specific requirements**

**4.1 External Interface**

An android application is used to monitor the on-going field activities.

**4.2 Functions**

* The sensing elements like temperature sensor, soil moisture sensor are connected to the arduino uno board, reads the atmospheric temperature and soil moisture content respectively.
* IR sensors are also used to detect the animal intrusion in the field.
* Arduino is then connected to the internet using a GSM module.
* The arduino, further stores the sensor data on the c9 Cloud.
* The agriculture system is then monitoredusing an android application.
* If the user is new to the system, he has to first register himself with his name, address, and city and phone number. By doing this the registered user can enter into smart agriculture system.
* The user has to then login into the system.
* Depending on the data retrieved from the cloud, the motor is switched on and off.

**4.3 Performance Requirements**

* The proper working of the arduino uno board, has it controls the entire agriculture system.
* It is necessary that all the sensors are in good condition.
* An android phone with proper internet connectivity to retrieve the sensor data from the cloud.

**4.4 Software System Attributes**

**4.4.1 Reliability**

The system is completely reliable when provided with proper internet .

**4.4.2 Availability**

The system is available all the time except when the server is down. So 80% of the time system will be available for use.

**4.4.3 Security**

The system can be used and viewed only by the registered users. A person with no user account cannot log into the system.

**4.4.4 Maintainability**

This project can be modified by adding different sensors required and can be extended to the required landscape.

**4.4.5 Portability**

The usage of arduino uno board makes the system portable to use.

**4.5 Organizing the Specific Requirements**

**4.5.1 System Mode**

The system works in user mode.

**4.5.2 User Class**

The user class needs planning permission to change from one use class to another, although there are exceptions where the legislation does allow some changes between uses.

**4.5.3 Object**

Hardware Used

* Arduino UNO
* Temperature & Humidity Sensor
* Soil Moisture Sensor
* IR Sensor
* GSM Modem

Software Used

* Android Studio

**4.5.4 Features**

* Crop water management
* Pest management and control work
* Precision Agriculture
* Cost effective method
* High quality crop production

**4.5.5 Stimulus**

Cloud stores the data given by sensors used in field.

**4.5.6 Response**

Participants have shared orientations in implementing public initiatives on biodiversity protection and smart agriculture as well as experience in mobilising resources for developing relevant projects in localities.

**4.5.7 Functional Hierarchy**

**Step 1**: Storing the sensor data on the c9 Cloud.

**Step 2**: Retrieving the sensor data using an android application.

**Step 3**: Automatic supply of water based on sensor data.

**Chapter 5**

**Design phase**

**5.1 Schema Diagram**

Registration\_Info

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Address | City | Phone\_Number |

Sensor\_Info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Time | Temp | Moisture | Object\_Detection | Motor\_Status |

Fig 5.1 Schema Diagram

**5.2ER Diagram**

An entity-relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities.

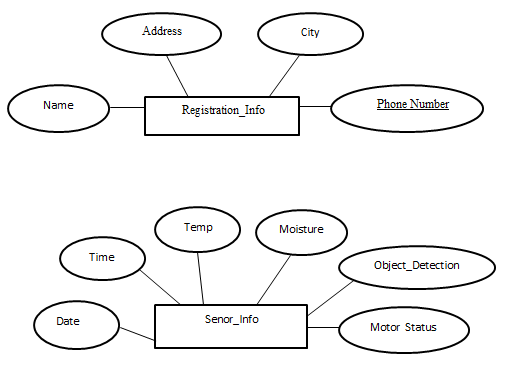
****

Fig 5.2 ER Diagram

**5.3 Architectural Design**

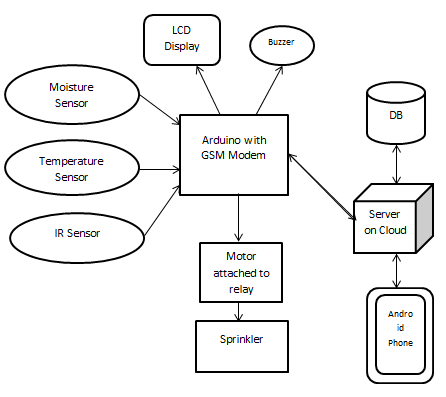
****

Fig 5.3 Architectural Design

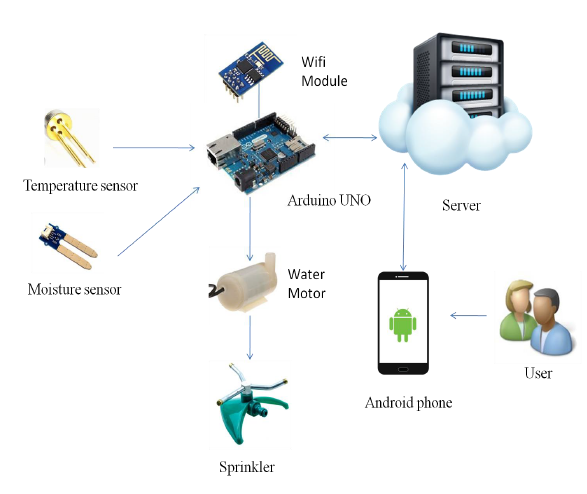
Sensing elements like soil moisture sensor, temperature sensor and IR sensor is connected to arduino, which is further connected to a GSM modem. The data collected from arduino will be stored in a remote server. The user can then view the on-going field activities using an android application.

The sensors continuously sense the field. When the threshold value of the soil drops the moisture sensor will signal the arduino, which will further trigger the relay to turn the motor on. Upon reaching the specified threshold, the motor is turned off.

IR sensor is used to detect the animal intrusion happening on the field. Buzzer is used to notify the animal intrusion.

The LCD display with arduino uno is used to display the state of the motor i.e. motor on/ motor off.

* 1. **Client Server Model**



GSM

Fig 5.4 Client Server Model

The Client Server Architecture is shown in the figure. The major IoT sensors are been displayed in this architecture. The sensor data storage is also been showed in the figure.**5.5UML Diagram**

The Unified Modelling Language (UML) is a general purpose modelling in the field of software engineering which is designed to provide a standard way to visualize the design of a system.

**5.5.1 Advance Class Design**

The class modelling captures the static structure of a system by characterizing the objects in the system, the relationships between the objects and the attributes and operations for each class of objects.

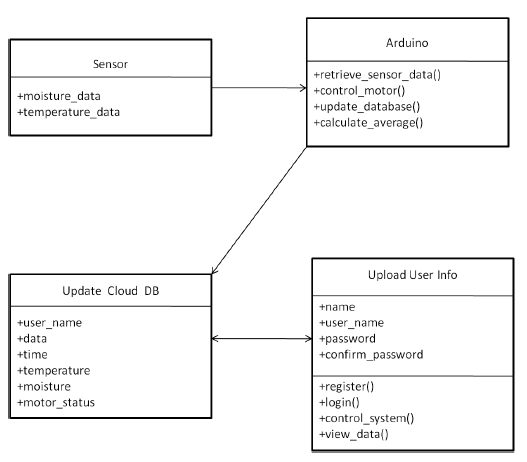


Fig 5.5.1 Advance Class Design

**5.5.2 Advanced State Design**

A **state diagram** is a type of **diagram** used in computer science and related fields to describe the behaviour of systems.

**State diagram for condition of the field**

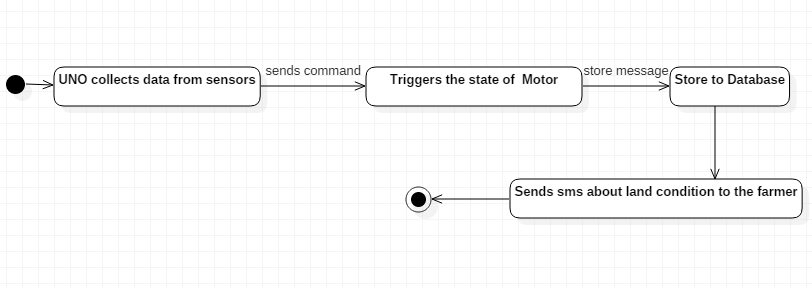


Fig 5.5.2.1 State diagram for field condition

**State diagram for detection of an object**

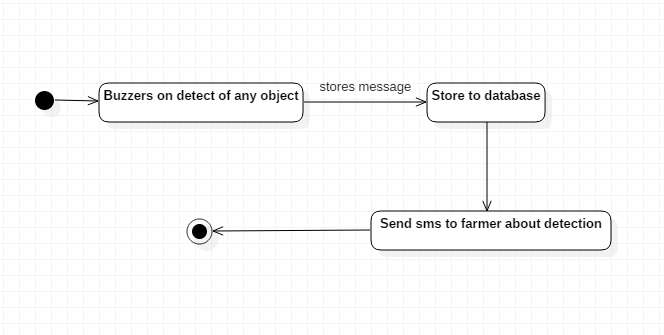


Fig 5.5.2.2 State diagram for object detection

**5.5.3 Advance Interaction Model**

**Sequence Model**

A Sequence model is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence model shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

1. **Sequence Model for Control System**

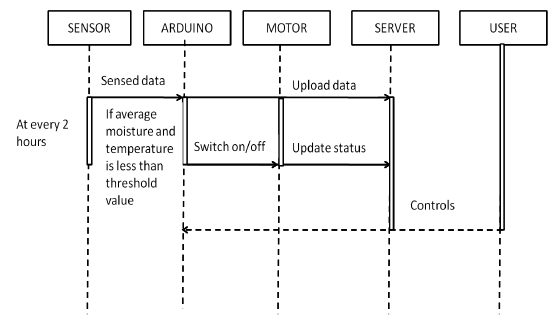


Fig 5.5.3.1 Sequence Model for Control System

The above figure shows the interaction between the components that controls the water supply automatically. Sensor senses the data and sends it to arduino. Arduino will switch on/off the motor that controls the sprinkler. At the same time arduino updates the database on server.

1. **Sequence Model for User Interaction**

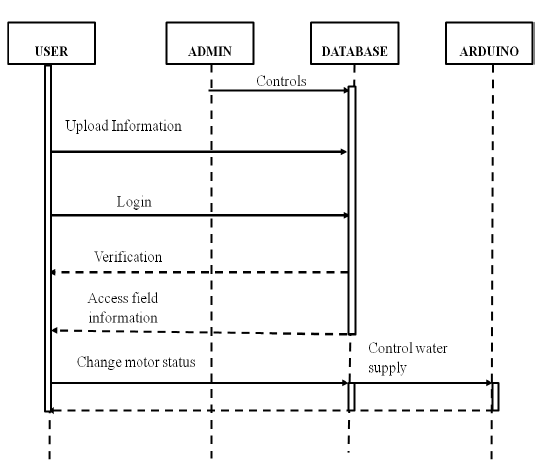


Fig 5.5.3.2 Sequence Model for User Interaction

The above figure shows the interactions between the user, admin, database and the arduino. First the user registers to an application by uploading the user information and then logs into an application. Admin verifies the user ID and password. User can access the field information stored in the server on cloud and also he can have control over the water supply by changing the status of the motor according to his need.

**5.6Swimlane Activity Model**

Activity diagrams are graphical representations of workflow of stepwise activities and actions with support for choice, iteration and concurrency. In the UML, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

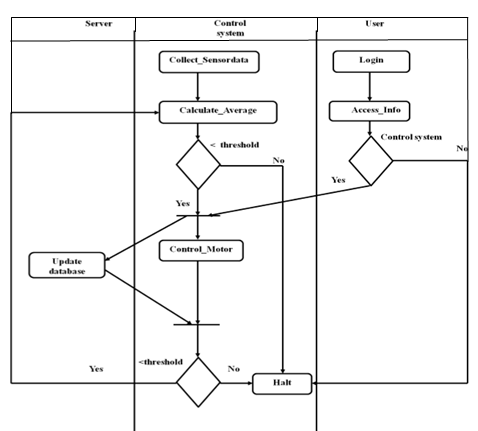


Fig 5.6 Swimlane Activity Model

**Chapter 6**

**Implementation**

**6.1 Implementation Details**

**Android code for login**

**package** com.example.admin.demo;  
**import** android.content.Intent;  
**public class** Loginpage **extends** AppCompatActivity {  
 EditText **t1**, **t2**;  
 Button **b1**;  
 String **s1**, **s2**, **ReturnValue**;  
**public void** login(View v) {  
**s1** = **t1**.getText().toString();  
**s2** = **t2**.getText().toString();  
**if** (**t1**.getText().toString().matches(**""**)) {  
 Toast.*makeText*(getApplicationContext(), **"User name is empty"**, Toast.***LENGTH\_LONG***).show();  
 } **else if** (**t2**.getText().toString().matches(**""**)) {  
 Toast.*makeText*(getApplicationContext(), **"Password empty"**, Toast.***LENGTH\_LONG***).show();  
 } **else** {  
 String url = **"https://smartagriculture-gurusirsi.c9users.io/alogin.php"**;  
**new** userlogin().execute(url);  
 } }  
**public void** userreg(View v) {  
 Intent i3 = **new** Intent(getApplicationContext(), Registration.**class**);  
 startActivity(i3); }  
**public void** forgot(View v) {  
 Intent i3 = **new** Intent(getApplicationContext(), forgotpassword.**class**);  
 startActivity(i3); }  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
**super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_loginpage***);  
**t1** = (EditText) findViewById(R.id.***editText***);  
**t2** = (EditText) findViewById(R.id.***editText2***);  
**b1** = (Button) findViewById(R.id.***button***); }  
**private class** userlogin **extends** AsyncTask<String, Void, Void> {  
@Override  
**protected void** onPreExecute() {  
**super**.onPreExecute();  
 Toast.*makeText*(getApplicationContext(), **"reading values"**, Toast.***LENGTH\_LONG***).show();  
 }  
@Override  
**protected** Void doInBackground(String... params) {  
 List<NameValuePair> nameValuePairs = **new** ArrayList<NameValuePair>();  
 nameValuePairs.add(**new** BasicNameValuePair(**"username"**, **s1**));  
 nameValuePairs.add(**new** BasicNameValuePair(**"password"**, **s2**));  
 DbHttpResponse myHttpResponse = **new** DbHttpResponse();  
 String rspTxt = myHttpResponse.getResponseString(params[0], nameValuePairs);  
**ReturnValue** = rspTxt;  
**return null**; }  
@Override  
**protected void** onPostExecute(Void aVoid) {  
**super**.onPostExecute(aVoid);  
**int** startval = 0;  
 startval = **ReturnValue**.indexOf(**"<!DOCTYPE"**);  
**if** (startval >0) {  
**ReturnValue** = **ReturnValue**.substring(0, startval); }  
**ReturnValue** = **ReturnValue**.replace(**"\r\n\r\n"**, **""**);  
 String type = **ReturnValue**.trim();  
**if** (type.equals(**"farmer"**)) {  
 Intent i1 = **new** Intent(getApplicationContext(), Farmreoptions.**class**);  
 Bundle bb=**new** Bundle();  
 bb.putString(**"custid"**,**s1**);  
 i1.putExtras(bb);  
 startActivity(i1);  
 } **else** {  
 Toast.*makeText*(getApplicationContext(), **"Invalid Password"**, Toast.***LENGTH\_LONG***).show();  
 }  
 Toast.*makeText*(getApplicationContext(), **ReturnValue**, Toast.***LENGTH\_LONG***).show();  
 }}

}

**Android code for View Message**

**package** com.example.admin.demo;

**import** android.support.v7.app.AppCompatActivity;

**public class** viewmessage **extends** AppCompatActivity {

String **data** = **""**;

TableLayout **tl**;

TableRow **tr**;

TextView **label**,**qty**;

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_viewmessage***);

StrictMode.*enableDefaults*();

**tl** = (TableLayout) findViewById(R.id.***table***);

String result = **null**;

InputStream is = **null**;

**try** {

HttpClient httpclient = **new** DefaultHttpClient();

HttpPost httppost = **new** HttpPost(**"https://smartagriculture-gurusirsi.c9users.io/viewmessage.php"**);

HttpResponse response = httpclient.execute(httppost);

HttpEntity entity = response.getEntity();

is = entity.getContent();}

**catch** (Exception e) {

}

**try** {

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is,**"iso-8859-1"**), 8);

StringBuilder sb = **new** StringBuilder();

String line = **null**;

**while** ((line = reader.readLine()) != **null**) {

sb.append(line + **"\n"**);

}

is.close();

result = sb.toString();

} **catch** (Exception e) {

}

**try** {

JSONArray jArray = **new** JSONArray(result);

TableLayout tv = (TableLayout) findViewById(R.id.***table***);

tv.removeAllViewsInLayout();

**int** flag = 1;

**int** k=0;

**for** (**int** i =0; i < jArray.length(); i++) {

TableRow tr = **new** TableRow(viewmessage.**this**);

tr.setLayoutParams(**new** LayoutParams(

LayoutParams.***FILL\_PARENT***,

LayoutParams.***WRAP\_CONTENT***));

**if** (flag == 1) {

TextView b6 = **new** TextView(viewmessage.**this**);

b6.setText(**"Message"**);

b6.setTextColor(Color.***BLUE***);

b6.setTextSize(15);

tr.addView(b6);

TextView b19 = **new** TextView(viewmessage.**this**);

b19.setPadding(10, 0, 0, 0);

b19.setTextSize(15);

b19.setText(**"Date"**);

b19.setTextColor(Color.***BLUE***);

tr.addView(b19);

tv.addView(tr);

**final** View vline1 = **new** View(viewmessage.**this**);

vline1.setLayoutParams(**new** TableRow.LayoutParams(TableRow.LayoutParams.***FILL\_PARENT***, 1));

vline1.setBackgroundColor(Color.***WHITE***);

tv.addView(vline1);

flag = 0;

} **else** {

JSONObject json\_data = jArray.getJSONObject(k);

TextView b = **new** TextView(viewmessage.**this**);

String stime = String.*valueOf*(json\_data.getString(**"message"**));

b.setText(stime);

b.setTextColor(Color.***RED***);

b.setTextSize(15);

tr.addView(b);

TextView b1 = **new** TextView(viewmessage.**this**);

b1.setPadding(10, 0, 0, 0);

b1.setTextSize(15);

String stime1 = json\_data.getString(**"rdate"**);

b1.setText(stime1);

b1.setTextColor(Color.***BLACK***);

tr.addView(b1);

tv.addView(tr);

**final** View vline1 = **new** View(viewmessage.**this**);

vline1.setLayoutParams(**new** TableRow.LayoutParams(TableRow.LayoutParams.***FILL\_PARENT***, 1));

vline1.setBackgroundColor(Color.***WHITE***);

tv.addView(vline1);

k=k+1;

}

}

} **catch** (JSONException e) {}  
}

}

**Chapter 7**

**Testing**

**7.1System Setup**

Smart agriculture using IoT is an IoT based system. Hence testing has been carried out separately that is proper working of LED, sensors like moisture sensor, temperature sensor and IR sensor and also authentication of the user has been done in android which resulted in required output. Then all these modules are integrated and tested for the completing function.

**7.2Types of test carried out**

**7.2.1Test Case Specification**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Testing Arduino Board** | | | | **Use Case ID: UC1** | | |
| **1. Module Overview** | | | | | | |
| *Test the arduino board with LED* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| Digital signal. | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *Blinking of LED at regular interval.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected Results | Actual Results | | Pass/Fail | Remarks |
| UC1-1 | Digital signal | Toggle LED at regular interval | LED got switched on and off at regular interval | | Pass | Hardware correctness |

Table7.2.1 TestingArduinoBoard

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Testing Temperature Sensor** | | | | **Use Case ID: UC2** | | |
| **1. Module Overview** | | | | | | |
| *Displays atmospheric temperature* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Temperature values.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *Display temperature value in the serial monitor* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected  Results | Actual  Results | | Pass/Fail | Remarks |
| UC2-1 | Temperature data | Display the accurate temperature on the LCD. | The accurate temperature is displayed on the LCD. | | Pass | Hardware correctness |

Table 7.2.2 Testing Temperature Sensor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Testing Moisture Sensor** | | | | **Use Case ID: UC3** | | |
| **1. Module Overview** | | | | | | |
| *Displays soil moisture* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Moisturevalues.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *Display Moisture value in the serial monitor* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected  Results | Actual  Results | | Pass/Fail | Remarks |
| UC3-1 | Moisture data | Display the accurate moisture on the LCD. | The accurate moisture is displayed on the LCD. | | Pass | Hardware correctness |

Table 7.2.3 Testing Moisture Sensor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name:Data Collection** | | | | **Use Case ID: UC4** | | |
| **1. Module Overview** | | | | | | |
| *Collects the sensor data and stores it on cloud.* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Temperature and Moisture sensor values.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *The collected sensor data is stored on the cloud.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected Results | Actual Results | | Pass/Fail | Remarks |
| UC4-1 | Sensor data | Store the data on the cloud | Stores the data on the cloud | | Pass | Logic coverage |

Table 7.2.4 Data Collection

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Automatic Control System** | | | | **Use Case ID: UC5** | | |
| **1. Module Overview** | | | | | | |
| *Automatically controls the motor based on the sensor values.* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Moisture and Temperature sensor data.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *Controls the water supply.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected  Results | Actual  Results | | Pass/Fail | Remarks |
| UC5-1 | Field is dry, Temperature  is high | Motor should switch on and motor status should get updated on thecloud. | Motor gets  Switched on and motor status got updated on thecloud. | | Pass | Logic coverage |
| UC5-2 | Field is dry, Temperature is low | Motor should switch on and motor status should get updated on thecloud. | Motor gets  switched on and motor status got updated on thecloud. | | Pass | Logic coverage |
| UC5-3 | Intruder found on Field | Buzzer beeps | Buzzer beeps | | Pass | Logic coverage |

Table 7.2.5 Automatic Control System

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Registration** | | | | **Use Case ID: UC 6** | | |
| **1. Module Overview** | | | | | | |
| *Takes user information name, Address, city and phone number stores them in the database.* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Name, Address, city, phone number confirm password* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *Stores the user information in the database.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected  Results | Actual  Results | | Pass/Fail | Remarks |
| UC6-1 | *Name :*  Rosy  *Address:*  PB Road  *City:*  Banglore  *Phone number:*  9876543210 | Store the data in the database  successfully. | Data is  successfully stored | | pass | Logic coverage |
| UC6-2 | *Name :*  Rosy  *Address:*  PB Road  *City:*  Banglore  *Phone number:*  9876543210 | show a message as  “password does not match”. | show a  message as “password does not match”. | | pass | Logic coverage |
| UC6-3 | *Name :*  Rosy  *Address:*  PB Road  *City:*  Banglore  *Phone number:*  9876543210 | show a  message as “fields cannot be  Empty”. | shows a  message as “fields cannot be  empty”. | | pass | Logic coverage |

Table 7.2.6 Registration

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: Login** | | | | **Use Case ID: UC7** | | |
| **1. Module Overview** | | | | | | |
| *Checks for the user’s authentication.* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *User name, password.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *If the user is authenticated, he gets logged in successfully else it will show a message as “Incorrect User name or Password”.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use Case ID | Input Data | Expected Results | Actual Results | | Pass/Fail | Remarks |
| UC7-1 | *User name:* 9876543210*Password:*  Rosy | user should login successfully | User is logged in successfully | | Pass | Authentication coverage |
| UC7-2 | *User name:* 9876543210*Password:*  Rosy | show a message as “IncorrectUser  name or Password”. | Shows a message as “Incorrect User name or Password”. | | Pass | Authentication coverage |
| UC7-3 | *User name: Password*: | show a message as “fields cannot be  empty”. | shows a message  as “fields cannot be empty”. | | Pass | Authentication coverage |
| UC7-4 | *User name:* 9876543210*Password:*  Rosy | show a message as “fields cannot be  empty”. | shows a message  as “fields cannot be empty”. | | Pass | Authentication coverage |
| UC7-5 | *User name: Password:* | show a message as “fields cannot be  empty”. | shows a message as “fields cannot be empty”. | | Pass | Authentication coverage |

Table 7.2.7 Login

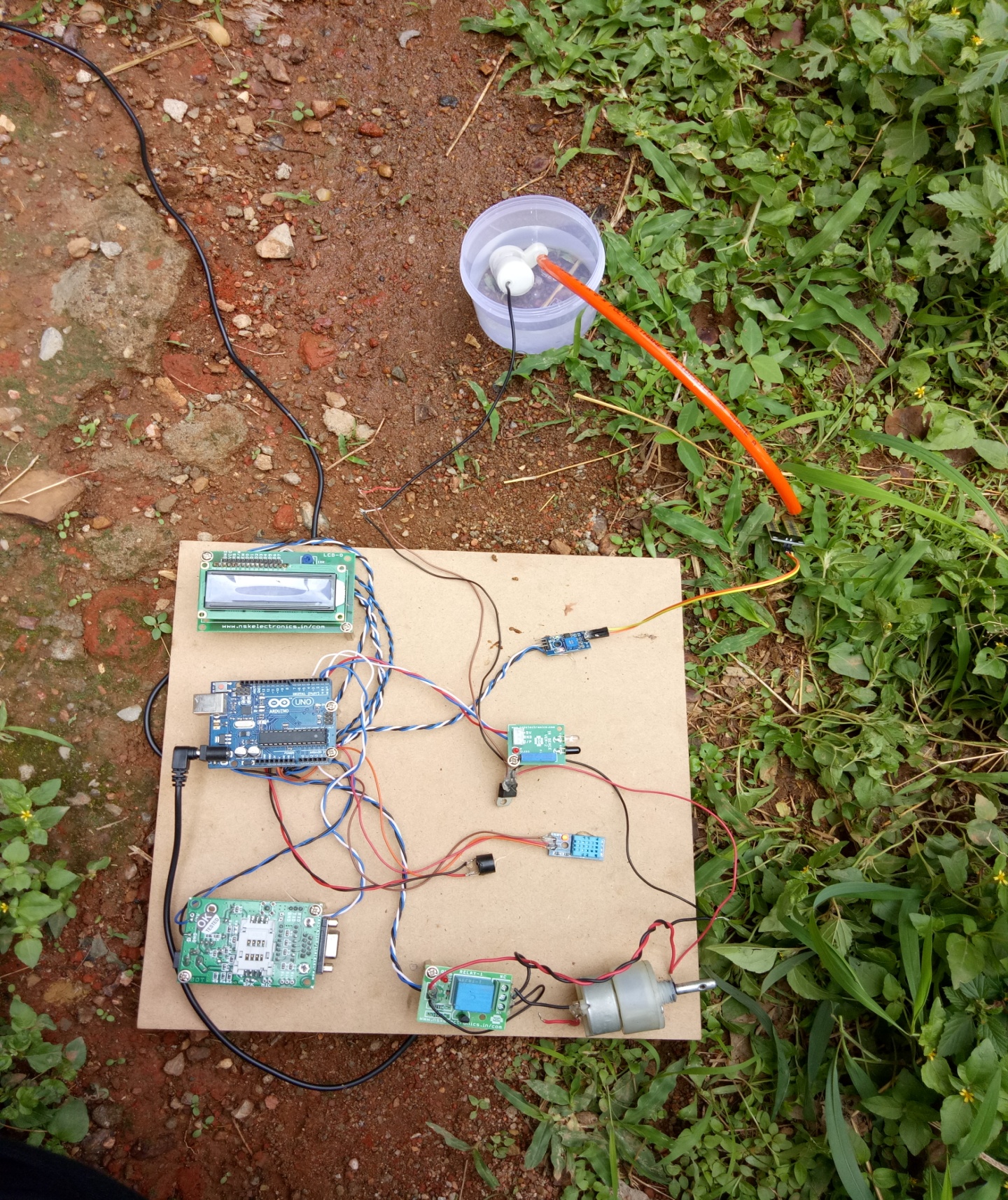
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module Name: View Cloud Data** | | | | **Use Case ID: UC8** | | |
| **1. Module Overview** | | | | | | |
| *To view the sensor data stored on the c9 cloud.* | | | | | | |
|  | | | | | | |
| **1.1 Inputs to Module** | | | | | | |
| *Click on the “ view data” button.* | | | | | | |
|  | | | | | | |
| **1.2 Outputs from Module** | | | | | | |
| *View the temperature, moisture sensor data and motor status.* | | | | | | |
|  | | | | | | |
| **2. Test Data** | | | | | | |
| Use  Case ID | Input Data | Expected Results | Actual Results | | Pass/Fail | Remarks |
| UC8-1 | Click on the “view data” button. | View the temperature, moisture sensor data and motor status. | View the temperature, moisture sensor data and motor status. | | Pass | Logic coverage |

Table 7.2.8 View Cloud Data

**Chapter 8**

**Result**

**8 Hardware kit**

****

**8.1 Screenshots**



Fig 8.1 The first page of the android application.

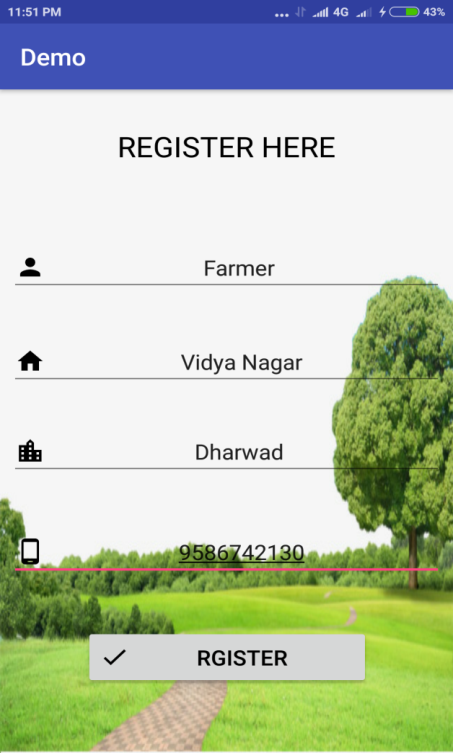


Fig 8.2 The registration page for the new user

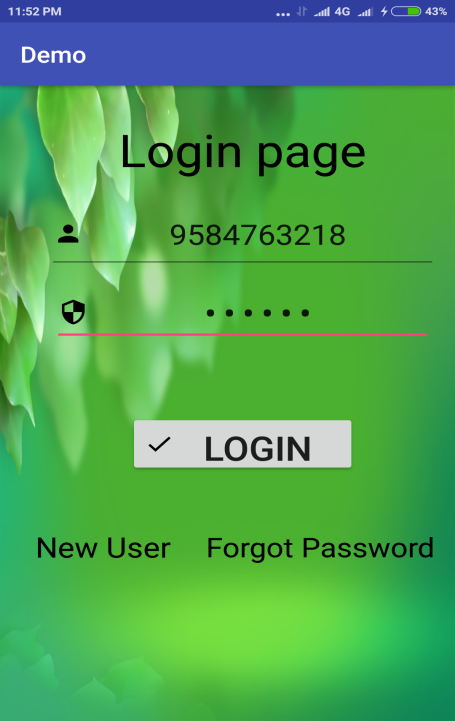


Fig 8.3 The login page of a registered user

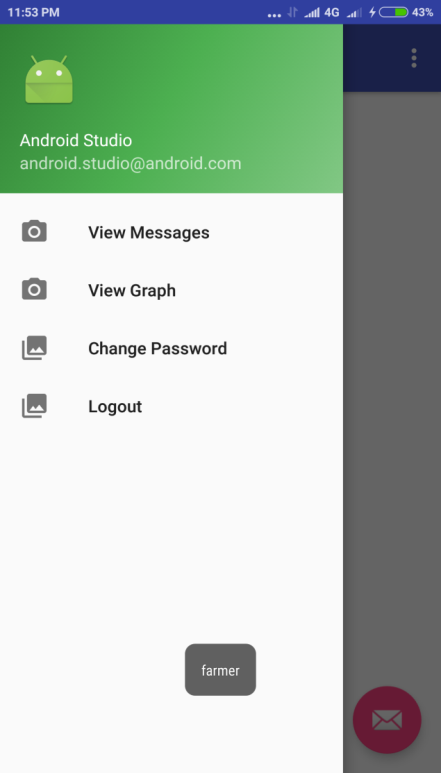


Fig 8.4 User Options

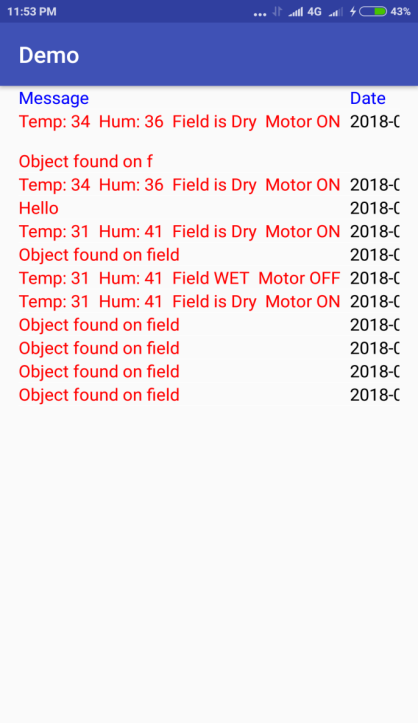


Fig 8.5 This page displays the output of IoT sensors

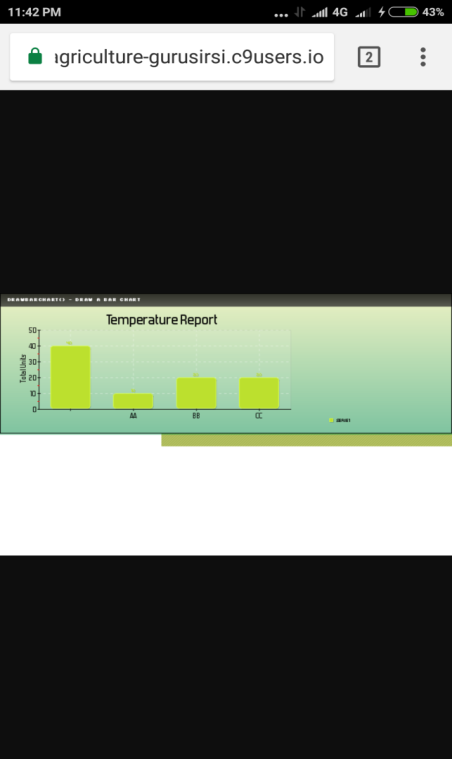


Fig 8.6 This page displays the sensor details in a graphical view

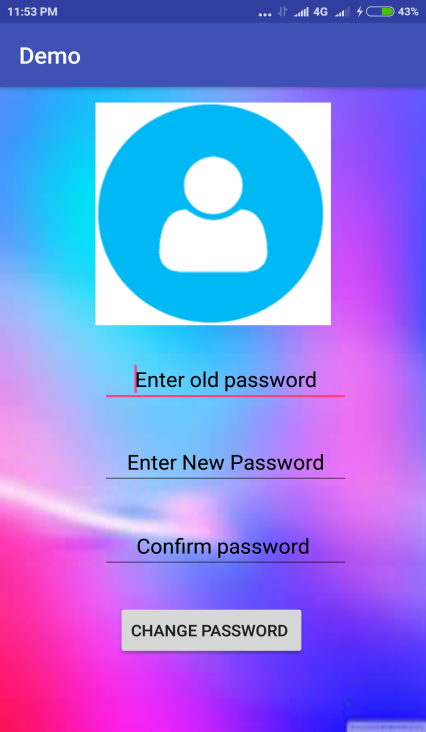


Fig 8.7 User option to change the current password

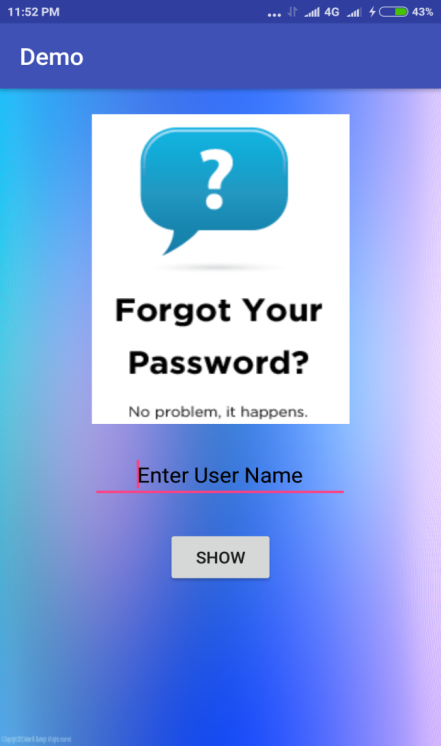


Fig 8.8 User option to change password when the user forgets their password

* The Smart agriculture system can be used specifically in agricultural field.

**Chapter 9**

**Applications**

* This system can also be used in all other fields like home garden, schools, colleges, plantations etc.
* Applicable in fresh water deficiency area, as it reduces the wastage of fresh water.
* Smart Irrigation also has a few other uses in crop production, which include protecting plants against frost, suppressing weed growth in grain fields and preventing soil consolidation.

**Chapter 10**

**Conclusion**

The system for agriculture filed offers a potential solution to support specific irrigation management that allows the producers to maximize their productivity while saving the water. The system is designed using an Arduino Uno. The temperature and the soil moisture sensor to detect the field temperature and moisture content of the soil and the sensor values are sent to the cloud via GSM module which connects the system to the internet.

The arduino checks the conditions for irrigation and performs automatic irrigation. The system can also be handled via android application it can be accessed from anywhere. Thus the smart agriculture system saves up to 25% more water than their traditional cousins. The smart agriculture system using IoT has been designed and implemented in this paper. The system developed is beneficial and works in cost effective manner. It reduces the water consumption to a great extent. It reduces the maintenance; the power consumption could be reduced by using solar power. The system can be used in green houses. The System is very useful in areas where water availability is a major problem. The productivity of the crop increases and the wastage of crops is very much reduced using this agriculture system. The developed system is more helpful and gives more feasible results.

**Chapter 11**

**Future scope**

The project can be upgraded using wireless sensors instead of wired sensors for covering larger field area. The android application can be added with different features like storing section wise field information to control different parts of the field.

In the near future, a modern day agriculture farm could be managed based on data from each field, such as rotation of the crop, yield of the crop, soil condition, amount of infestation spread, and conditions of climate where crops are grown, decision models would determine the site specific soil tillage, pre-treatment of the seedbed, and sowing density of the farm. In the growth season, the modern farmer decides about site-specific application of fertilizers, supported by crop growth models and field measurements, the most important of which is soil coverage of crops in the early growth stage. A spraying machine aided with optical sensors for detection of diseases and weeds is used for the treatment of the crops. During harvesting, sensors monitor the bulk mass flow of harvested raw produce in the harvesting machine and also product properties that are valued commercially such as level of protein and moisture content of cereals and sugar content of sugar beets. These data, related to the field where it is captured and position of the machine are mapped in the records to support site-specific crop management in subsequent growing season.