**Green Core**

**The Smart Gardener**

System Requirement Specification

**Submitted To:**

**Version: 0.2**

Project Team (Group G27)

B A Medawatta

M M H Tharushika

H U K M Pabasara

J A N C Niroshana

K C Gamage

K S A Ahamed

Supervisor

Dr Dinuni K Fernando

Co Supervisor

Ms. Sithara Fernando

**Table of Contents**

[List of Figures 5](#_Toc44548591)

[List of Tables 6](#_Toc44548592)

[Group Members 7](#_Toc44548593)

[Document Change History 8](#_Toc44548594)

[Functional Items Summary 9](#_Toc44548595)

[Abbreviations 10](#_Toc44548596)

[1. Introduction 11](#_Toc44548597)

[1.1. Problem Domain 11](#_Toc44548598)

[1.2. Current System & limitations 12](#_Toc44548599)

[1.3. Vision 12](#_Toc44548600)

[1.4. Goals and Objectives 12](#_Toc44548601)

[1.5. Proposed Solution 12](#_Toc44548602)

[1.6. Scope Overview 13](#_Toc44548603)

[1.6.1. In Scope 13](#_Toc44548604)

[1.6.2. Out of Scope 14](#_Toc44548605)

[1.7. Objective (document) 14](#_Toc44548606)

[2. Assumptions and Dependencies 14](#_Toc44548607)

[2.1. Assumptions 14](#_Toc44548608)

[2.2. Dependencies 14](#_Toc44548609)

[3. Limitations and Constraints 14](#_Toc44548610)

[4. Feasibility Study 14](#_Toc44548611)

[4.1 Technical Feasibility 14](#_Toc44548612)

[4.2 Schedule Feasibility 15](#_Toc44548613)

[4.3 Economic Feasibility 15](#_Toc44548614)

[4.4 Legal and Ethical Feasibility 15](#_Toc44548615)

[4.5 Operational Feasibility 16](#_Toc44548616)

[5. Functional Requirements – Mobile App 16](#_Toc44548617)

[5.1. Manage User 16](#_Toc44548618)

[5.1.1 Register User (FR1) 16](#_Toc44548619)

[5.1.2 User Login (FR2) 19](#_Toc44548620)

[5.1.3 Forgot Password (FR6) 23](#_Toc44548621)

[5.1.4 View User Details (FR3) 26](#_Toc44548622)

[5.1.5 Update User Details (FR4) 27](#_Toc44548623)

[5.1.6 Change Password (FR5) 29](#_Toc44548624)

[5.1.7 Reset Password (FR6) 32](#_Toc44548625)

[5.2. Manage Units 32](#_Toc44548626)

[5.2.1 Link Units (FR8) 32](#_Toc44548627)

[5.2.2 View All Units (FR9) 35](#_Toc44548628)

[5.2.3 View Single Unit (FR10) 37](#_Toc44548629)

[5.2.4 Control Actuators 39](#_Toc44548630)

[5.3 Chat Module 40](#_Toc44548631)

[5.3.1 Send New Message (FR12) 40](#_Toc44548632)

[5.3.2 View All Messages (FR13) 41](#_Toc44548633)

[5.3.3 View Single Message (FR14) 42](#_Toc44548634)

[5.3.4 Reply Message (FR15) 44](#_Toc44548635)

[5.4 Alerts 45](#_Toc44548636)

[6 Functional Requirements – Web App 45](#_Toc44548637)

[6.1 Manage Users 45](#_Toc44548638)

[6.1.1 Register User (FR17) 45](#_Toc44548639)

[6.1.2 User (Admin) Login (FR18) 49](#_Toc44548640)

[6.1.3 Forgot Password (FR22) 51](#_Toc44548641)

[6.1.4 Rest Password (FR21) 51](#_Toc44548642)

[6.1.5 View User (Admin) Details (FR19) 53](#_Toc44548643)

[6.1.6 Update User (Admin) Details (FR20) 55](#_Toc44548644)

[6.1.7 View Users (FR24) 57](#_Toc44548645)

[6.1.8 Search Users (FR25) 59](#_Toc44548646)

[6.1.9 View Single User (FR26) 61](#_Toc44548647)

[6.2 Manage Units 61](#_Toc44548648)

[6.2.1 Search Units (FR28) 61](#_Toc44548649)

[6.2.2 View All Units (FR27) 64](#_Toc44548650)

[6.2.3 View Single Unit (FR29) 66](#_Toc44548651)

[6.2.4 View Vulnerable Units (FR30) 67](#_Toc44548652)

[6.3 Chat Module 69](#_Toc44548653)

[6.3.1 Send Message (FR33) 69](#_Toc44548654)

[6.3.2 View All Messages (FR31) 71](#_Toc44548655)

[6.3.3 View Single Message (FR32) 72](#_Toc44548656)

[6.3.4 Reply Message (FR34) 74](#_Toc44548657)

[6.3.5 View New Messages (FR34) 75](#_Toc44548658)

[6 Non-Functional Requirements 76](#_Toc44548659)

[6.1 Availability 76](#_Toc44548660)

[6.2 Usability 76](#_Toc44548661)

[6.3 Dependability 77](#_Toc44548662)

[6.4 Security 77](#_Toc44548663)

[6.5 Maintainability 77](#_Toc44548664)

[6.6 Performance 77](#_Toc44548665)

[6.7 Security 77](#_Toc44548666)

[6.8 Capacity 77](#_Toc44548667)

[7 System Requirements 77](#_Toc44548668)

[8 Stakeholders & Use Case Diagrams 78](#_Toc44548669)

[8.1 Stakeholders 78](#_Toc44548670)

[8.1.1 Admin 78](#_Toc44548671)

[8.1.2 Unregistered User 78](#_Toc44548672)

[8.1.3 Registered User 78](#_Toc44548673)

[8.2 Use Case Diagrams 79](#_Toc44548674)

[8.2.1 Users 79](#_Toc44548675)

[8.2.2 Admin 80](#_Toc44548676)

[9 System Architecture 80](#_Toc44548677)

[9.1 Architecture Diagram 80](#_Toc44548678)

[9.2 Component Diagram 81](#_Toc44548679)

[10 System Design 82](#_Toc44548680)

[10.1 Class Diagram 82](#_Toc44548681)

[10.2 Sequence Diagrams 82](#_Toc44548682)

[10.3 Activity Diagrams 83](#_Toc44548683)

[10.3.1 Login Activity Diagram-Mobile App 83](#_Toc44548684)

[10.3.2 Login Activity Diagram-Web App 84](#_Toc44548685)

[10.3.3 Chat module Activity Diagram 85](#_Toc44548686)

[10.3.4 Automated Watering of Plants Activity Diagram 86](#_Toc44548687)

[10.3.5 Manual Watering of Plants Activity Diagram 87](#_Toc44548688)

[10.3.6 Controlling light intensity automated Activity Diagram 88](#_Toc44548689)

[10.3.7 Controlling light intensity manual Activity Diagram 89](#_Toc44548690)

[10.3.8 Adding nutrients automated Activity Diagram 90](#_Toc44548691)

[10.3.9 Adding nutrition manual Activity Diagram 91](#_Toc44548692)

[11 User Interface Flow Diagram 92](#_Toc44548693)

[12 Approval 92](#_Toc44548694)

# List of Figures

# List of Tables

# Group Members

|  |  |  |
| --- | --- | --- |
| **Name** | **Index Number** | **Signature** |
| B A Medawatta | 17001072 |  |
| M M H Tharushika | 17001765 |  |
| H U K M Pabasara | 17020611 |  |
| J A N C Niroshana | 17001196 |  |
| K C Gamage | 17000475 |  |
| K S A Ahamed | 17020034 |  |

|  |  |
| --- | --- |
| **DOCUMENT REVISION HISTORY** | |
| Document Version Number | Date |
| V0.1 | 2020/05/30 |
| V0.2 | 2020/06/20 |
|  |  |
|  |  |
|  |  |

# Document Change History

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Requested By | Document Version |
| 01 | Initial SRS with functional requirements | Client | V0.1 |
| 03 | Screens for functional requirements | - | V0.2 |
| 04 |  |  |  |
| 05 |  |  |  |
| 06 |  |  |  |

# Functional Items Summary

|  |  |  |  |
| --- | --- | --- | --- |
| BR No. | BR Description | Functional Item No. | Functional Item Description |
| 1 | Green Core – Mobile App | FR1 | Register User |
| FR2 | User Login |
| FR3 | View User Details |
| FR4 | Update User Details |
| FR5 | Change Password |
| FR6 | Forgot Password |
| FR7 | Navigation |
| FR8 | Link Units |
| FR9 | View All Units |
| FR10 | View Single Unit Details |
| FR11 | Control Actuators |
| FR12 | Send New Message |
| FR13 | View All Messages |
| FR14 | View Single Message |
| FR15 | Reply Message |
| FR16 | Alerts |
| Green Core – Web App | FR17 | Register User (Admin) |
| FR18 | User (Admin) Login |
| FR19 | Forgot Password |
| FR20 | View User (Admin) Details |
| FR21 | Update User (Admin) Details |
| FR22 | Change Password |
| FR23 | Navigation |
| FR24 | View Users |
| FR25 | Search Users |
| FR26 | View Single User |
| FR27 | Search Units |
| FR28 | View All Units |
| FR29 | View Single Unit |
| FR30 | View Vulnerable Units |
| FR31 | Send Message |
| FR32 | View All Messages |
| FR32 | View Single Message |
| FR33 | Reply Message |
| FR34 | View New Messages |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Explanation |
| FR | Functional Requirement |

# Introduction

## Problem Domain

With busy lifestyles people tend to stay at home leisurely for limited time. However, with the current sustainable trend, people tend to grow their own produce. With this growing trend, people with limited time should have a feasible way of gardening with a minimum effort. So, there should be a convenient way of getting organic produce to the household with a minimum effort. Also, an agricultural revolution is going on these days because of the pandemic which occurred in the recent past. But after this is gone people may not have time to take care of their plants. For the above problem, we introduce a smart garden concept which is an automated way of gardening by using a mobile phone.

## Current System & limitations

Though there are systems developed to automate small scale indoor gardens, there are no automated systems to control outdoor gardens. Even in the indoor automated gardens we have to use their smart garden environment. We can’t plug in the system to our garden directly.

**Limitations of Current system**

* Not applicable for outdoor gardens
* Must use their pots to plant
* Only small plants are possible

## Vision

Achieve sustainable crops even you have a lack of knowledge or time

## Goals and Objectives

The main goal is to develop a smart garden environment that allows the user to provide a real-time gardening experience via a mobile phone.

Objectives

* People can get fresh vegetables and food from their garden with the aid of a mobile application.
* Track the progress/ status of each individual plant's status.
* Adjust the growing environment remotely and in an automated way.
* Get periodic notifications of individual plat status via mobile App
* Administrators can collect, view data of all gardens via web application

## Proposed Solution

A smart Garden which allows the users to control their garden while they are away. The automatic gardening feature will check the environmental conditions and maintain the conditions required for the plants to grow, Manual method allows the users to control the plant conditions which will give full remote control of the garden to the user. This includes an IoT device, a mobile app, and an admin portal.

The system mainly consists of two main deliverables,

Mobile Application and IoT Device

* System for the user consists of one or more IoT device units and a mobile app. The users can register to the system using the app and then connect the units for their account. A unit contains multiple sensors and actuators which are used to control the garden.
* Using the mobile app, the users can control actions such as adding nutrients, watering the plants when the soil moisture level is low, changing lighting conditions, etc. With the help of our mobile app users can get fresh vegetables and food from their garden. They can track the progress of their harvest, adjust input levels and growth settings remotely.
* With Bug repellent system to repel bugs, a buzzer will automatically play a noise when bugs get nearby. Also, users will be notified about bug attacks via mobile application. Plant watering system will automatically and manually water the plants when the soil moisture level is low, and the surrounding temperature is high. If the user does not have sufficient knowledge of the plant, once the user starts cultivating the seeds, the app monitors the status of the plant and provides suggestions to improve the status of the plant.

Web Application

* Web application helps administrators to manage user accounts, devices and get statistics about them. This also helps to collect data from our users' gardens. This data collection will help us to improve our system in terms of better predictions. Users' privacy will be retained safely and only sensor readings will be utilized for statistics and future predictions.

## Scope Overview

Users (possible actors) of the system:

* Admins - Web Application
* Users - Mobile Application

### In Scope

* Mobile Application
  + Control and get feedback of the IoT units
  + Manages IoT units
  + Manage the settings related to user accounts
  + Show notifications related to IoT devices
  + View the status of the garden (water levels, moisture levels)
  + Give planting tips to users
* IoT Device
  + Plant watering system
  + Lighting system
  + Add nutrients when the soil nutrients (such as Ammonia) when it is needed.
  + Bug repellant system
* Web Application
  + Analyze the data from all the gardens with web Application
  + Manage Plant Data (Admin)
  + Monitor each IoT device’s health and notify if something is wrong
  + Manage the data related to the tips provided by the mobile app.

### Out of Scope

* Train a machine learning model using the data gathered by the sensors to predict optimal conditions to grow specific plant types
* Monthly report generation for mobile app
* Live view of the garden via a mobile app

## Objective (document)

This document explains the functionality of the process of “Green Core” that is built to control gardens remotely. Focus of this document is to explain each functionality in detail, for the purpose of system development and provide an understanding of the proposed “Smart Garden” to all the stakeholders of the project.

# Assumptions and Dependencies

## Assumptions

* Necessary AWS environments will be provided on time.
* Necessary access for the email Gateway integrations will be provided on time.
* The users will be using standard and latest browsers to access the system (The system may give some issues with older browser versions such as IE 8 and earlier versions)

## Dependencies

* Availability of hosting environment and access for third-party integrations

# Limitations and Constraints

* Multi Language support will not be available

# Feasibility Study

### 4.1 Technical Feasibility

* The main product of Green Core is an IoT device built using Nodemcu and an Arduino and a mobile application developed by using React Native, JSX, and CSS and for the backend, we will be using Node.js and for the database, we will be using MongoDB. Most of these technologies are freely available or could be bought for a low cost.
* For the administration, we are using another web app. It is hosted using a dedicated AWS server which is also available for free for a year. After a year we will have to pay and upgrade or look for another server to host our application. The dedicated hardware of 1GB RAM and 1 core CPU will be sufficient at the start. But as the user base grows, we will have to opt for a more powerful server. But for development purposes, these technologies are certainly sufficient.

### 4.2 Schedule Feasibility

* Green Core should finish development by the end of August. And since we are following agile methodology for development, requirement gathering also happens continuously parallel to development.
* It means more and more features may get added from time to time. But in the current context, considering the number of features that we must implement as at now it will be possible. Also, it is worth noting that since we are using MERN stack, Arduino, node MCU, and AWS it makes development much easier and hassle-free which accelerates the development time to make it possible to be done by the deadlines.

### 4.3 Economic Feasibility

* As we are developing this project with profitable motive the production cost should be much lower
  + The web application is hosted in the AWS free-tier server, which is provided by AWS for free for one year and this bears zero cost for the first year of operation.
  + As we use MongoDB for our database which is free and there are 25Million database operations provided for the free tier
  + The web application should have a domain and we will have to pay a small annual subscription for the domain. ($12 per year)
  + The development tools like visual studio code Arduino IDE can be used free of charge for the project.
  + As we use NodeMcu and Arduino for the IoT units the cost will be around 15$ per unit
* Publishing an Android app requires a developer account which we will have to make a one-time payment for the account. ($25 one-off payment)
* Since it is developed only by a team of university students, development cost also can be factored as none
* Therefore, in the short run, the development takes a very small cost ($37 + costs for units) (if we did not factor the developers cost). But in the long run, as the system grows, we may have to bear a moderate cost for hosting fee in AWS as well as to allow more capacity in the MongoDB

### 4.4 Legal and Ethical Feasibility

* Sensitive information will not be published to the outside world by the system
* Users are visible only for the admins

### 4.5 Operational Feasibility

* We have surveyed over 100 people from different areas of the island who have started farming in their home gardens after the pandemic started. We let them know about the idea of this system and asked whether they think it would be useful for them in their field. Over 85% of the participants of this survey gave positive feedback saying they are interested in such an idea and it would solve most of their problems.
* As most of the fruits and vegetables we buy from the markets are heavily treated with pesticides and various types of chemicals, people are much willing to grow their own food. But the real issue occurs when it comes to time. As this system solves the time issue more people are willing to use this kind of solution.

# Functional Requirements – Mobile App

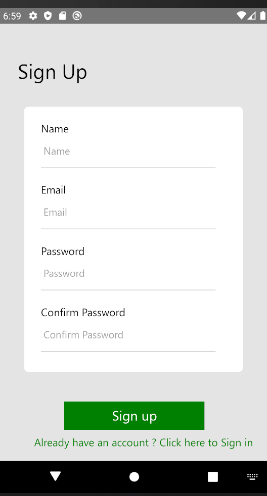
## Manage User

* Through this function Users can be viewed (Existing users in the system), created, updated, and deleted from the system.

### 5.1.1 Register User (FR1)

* When a new user installs the Green-Core app, the user needs to register for an account creation.
* User details will be saved in the database upon the click on “Sign Up” button, only if verification is successful. If not, the user will get an error message.

Mockup of the ‘Registration’ function is given below.



|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Name | To enter User’s name | Field type - Text Box  Field Length - Maximum 255  Data type - String  Mandatory |  |
| Email | To enter user’s email address | Field type - Text Box  Field Length -  Maximum 254  Data type - email  Mandatory  Valid email address | When email is already exists in the database,  “Email Already exists”  inline error message will be prompt. |
| Password | To enter user’s  password | Field type - Text Box  Field Length -  Maximum 16  Minimum 6  Data type - String  Mandatory |  |
| Confirm Password | To reenter user’s password | Field type - Text Box  Field Length -  Maximum 16  Minimum 6  Data type - String  Mandatory | Check whether the two passwords entered are the same. If not display error message “Confirm password should be a match” |
| Buttons | | | |
| Sign Up | To save the user details | Field Type - Button | Upon click, the user should be successfully saved in the system |

*Table 5.1.1:1: Input Field Validations for Sign Up*

#### 5.1.1.1 Basic flow of events

* The use case starts when the user clicks on the ‘Don’t have account click here to Sign Up’ text available in the ‘Login' screen
* User views the ‘Sign Up’ screen
* User Enter the Name, email, password, confirm password
* User clicks on the ‘Sign Up’ button
* User directs to ‘login’ screen
* The use case gets the end

#### 5.1.1.2 Alternative Paths

* When User clicks on “Already have an account? Click here to Sign In” text, user will navigate to Login Screen.

#### 5.1.1.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.5:1)

#### 5.1.1.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.5:1)

#### 5.1.1.5 System integrations

* N/A

#### 5.1.1.6 Negative Paths

* If the user clicks on the ‘Sign Up’ button by leaving the Mandatory fields empty. System should display the error messages.

#### 5.1.1.7 Messages

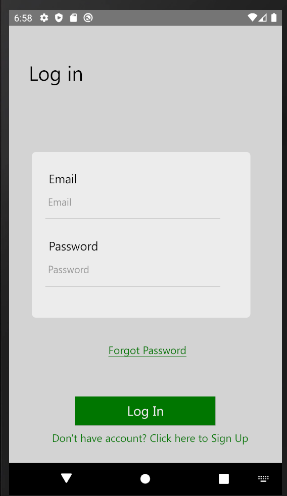
|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR5-MSG1 | Empty Username | In-line message | “Name is required” |
| FR5-MSG2 | Empty Email | In-line message | “Email should be Valid” |
| FR5-MSG3 | Empty password | In-line message | “Password must be at least 6 character long” |
| FR5-MSG4 | Empty confirm password | In-line message | “Password confirmation is required” |

*Table 5.1.1:2 Messages for Sign Up*

### 5.1.2 User Login (FR2)

* Users can log in to the system by entering valid email and password.
* System will generate a token on login and user will keep logged in until the token expire.

Mockup of the ‘Login’ function is given below.



|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Email | To enter user’s email address | Field type - Text Box  Field Length -  Maximum 254  Data type - email  Mandatory  Valid email address |  |
| Password | To enter user’s  password | Field type - Text Box  Field Length -  Maximum 16  Minimum 6  Data type - String  Mandatory |  |
| Buttons | | | |
| Sign In | To log in the to the system | Field Type - Button | Upon click, the user should be successfully logged in to the system |
| Forgot Password | To allow the user to reset password | Field Type – Text | Upon click user will be directed to request password change  screen |
| Signup | To allow a new user to register | Field Type – Text | Upon click the user will be directed to the signup screen |

*Table 5.1.2:1: Input Field Validations for Login*

#### 5.1.2.1 Basic flow of events

* The use case starts when the user opens the app or expiration of a token.
* User views the ‘Login' screen.
* User Enter the email, password.
* User clicks on the ‘Sign In’ button
* User directs to ‘Home’ screen
* The use case gets the end

#### 5.1.2.2 Alternative Paths

* Users can click on “Don’t have an account click here to Sign Up’ text available in the ‘Sign Up screen, the user will navigate to Sign Up Screen.
* User can click on “forgot password” user will direct to forgot password screen

#### 5.1.2.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.2:1)

#### 5.1.2.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.2:1)

#### 5.1.2.5 System integrations

* N/A

#### 5.1.2.6 Negative Paths

* If the user clicks on the ‘Log In’ button by leaving the Mandatory fields empty. System should display the error messages.
* If a user enters an incorrect email or password, System should display the error messages.

#### 5.1.2.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR5-MSG1 | Incorrect Email and password | Alert message | “Incorrect email or password” |

Table 5.1.2:2: Messages for Sign Up

### 5.1.3 Forgot Password (FR6)

* When the user forgets his/her password, the user can click on “forgot password” in the Login screen and request for a password reset.
* Then the system will send an email including a verify token.
* Users can enter the verify token in “verify token” screen and navigate to “reset password” screen.

Mockup of the ‘Forgot Password’ function, “Verify Token” screen is given below.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Verify Token | To enter verify token | Field type - Text Box  Field Length - Maximum 4  Data type - String/Number  Mandatory |  |
| Buttons | | | |
| Submit | To send the token for password reset request | Field Type - Button | Upon click, the user should navigate to reset password screen. |

Table 5.1.3:1: Input Field Validations for Forgot Password

Mockup of the ‘Forgot Password’ function, “Reset Password” screen is given below.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| New Password | To enter new password | Field type - Text Box  Field Length - Maximum 16  Data type - String  Mandatory |  |
| Re Enter New Password | To reenter the new password | Field type - Text Box  Field Length - Maximum 1  Data type - String  Mandatory | Check whether the two passwords entered are the same. If not display error message |
| Buttons | | | |
| Save Changes | To save the updated password | Field Type - Button | Upon click, the password should be successfully updated in the system |

*Table 5.1.3:1: Input Field Validations for Change Password*

#### 5.1.3.1 Basic flow of events

* The use case starts when the user clicks on the ‘Forgot Password’ button available in the ‘Login’ screen
* User views the ‘Verify Token’ screen
* User enter verify Token
* User click on the “Submit” button
* User navigates to the “Reset Password” screen
* User enters the new password
* User reenters the new password
* User clicks on the ‘Save Changes’ button
* User navigate to the login screen
* The use case gets the end

#### 5.1.3.2 Alternative Paths

* User can click on “Resend Token” button in the “Verify Token” screen

#### 5.1.3.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.3:1)

#### 5.1.3.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.3:1)

#### 5.1.6.5 System integrations

* N/A

#### 5.1.3.6 Negative Paths

* If the user clicks on the ‘Save Changes’ button in the “Reset password” screen by leaving the Mandatory fields empty. System should display the error messages
* If user clicks on the “Submit” button in the “Verify token” screen by leaving it empty, system should display the error messages

#### 5.1.3.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR5-MSG1 | Empty New Password | In-line message | ‘New Password is required’ |
| FR5-MSG2 | Empty Re Enter New Password | In-line message | ‘Reenter password’ |
| FR5-MSG3 | Empty verify Token | In-line message | ‘Verify Token is required’ |

Table 5.1.3:2: Messages for Sign Up

### 5.1.4 View User Details (FR3)

* Upon a click on the ‘View profile’ function, the list of existing users in the system should be displayed along with the search facility.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Card | | | |
| Card | This table should display a list of details related to the user account | Non-editable | Details such as username, email telephone number and location |
| Fields | | | |
| Email | Display the email associated with the current account | Non-editable  Display only |  |
| Username | Display the username associated with the current account | Non-editable  Display only |  |
| Telephone number | Display the telephone number associated with the current account | Non-editable  Display only |  |

*Table 5.1.4:1: Input field validations for View User Details screen*

#### 5.1.4.1 Basic Flow of events

* When the user clicks view profile in the drawer menu user will navigate to this screen

#### 5.1.4.2 Alternative Paths

* The user clicks on the back button

#### 5.1.4.3 Validations

* N/A

#### 5.1.4.4 Actions

* N/A

#### 5.1.4.4 System integrations

* N/A

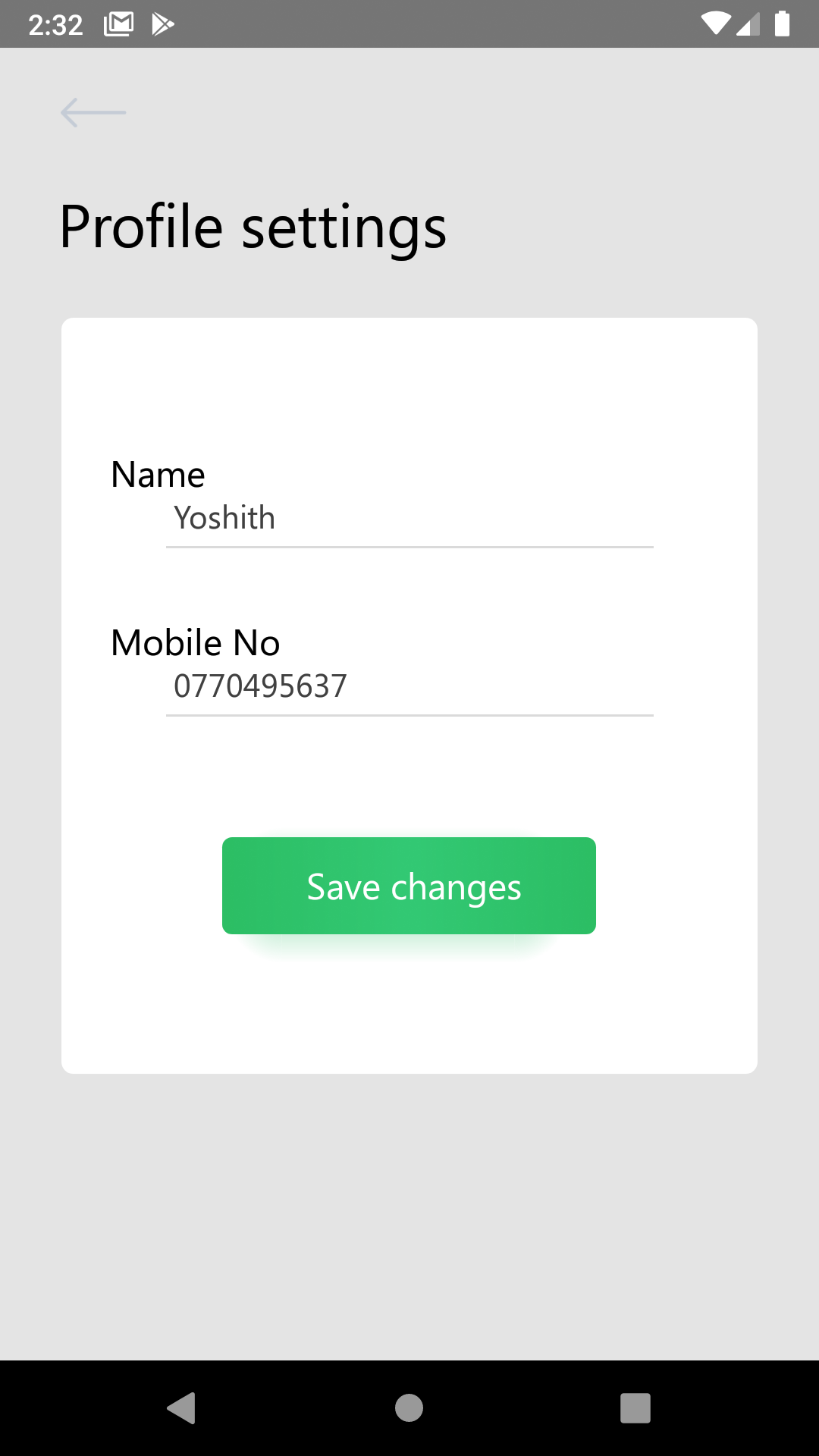
#### 5.1.4.2 Negative Paths

* N/A

### 5.1.5 Update User Details (FR4)

* User details must be able to be updated upon click on the ‘Edit Details’ button on the ‘View User Details’ screen.
* Through the edit user details function, the User’s name and the mobile number should be able to update.

Mockup of the ‘Update User Details’ function is given below



|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Name | Display User’s name | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value – Selected Username  Mandatory |  |
| Mobile Number | Display user’s mobile number | Field type - Text Box  Field Length - Maximum 20  Data type - String  Default Value – Selected User’s Mobile number |  |
| Buttons | | | |
| Save Changes | To save the updated user details | Field Type - Button | Upon click, the user should be successfully updated in the system |

*Table 5.1.5:1: Input Field Validations for Update User Details*

#### 5.1.5.1 Basic flow of events

* The use case starts when the user clicks on the ‘Edit Details’ button available in the ‘View User Details’ screen
* User views the ‘Update User’ screen
* User updates the Name
* User updates the Mobile number
* User clicks on the ‘Save Changes’ button
* The use case gets the end

#### 5.1.5.2 Alternative Paths

* N/A

#### 5.1.5.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.5:1)

#### 5.1.5.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.5:1)

#### 5.1.5.5 System integrations

* N/A

#### 5.1.5.6 Negative Paths

* If the user clicks on the ‘Save Changes’ button by leaving the Mandatory fields empty. System should display the error messages

#### 5.1.5.7 Messages

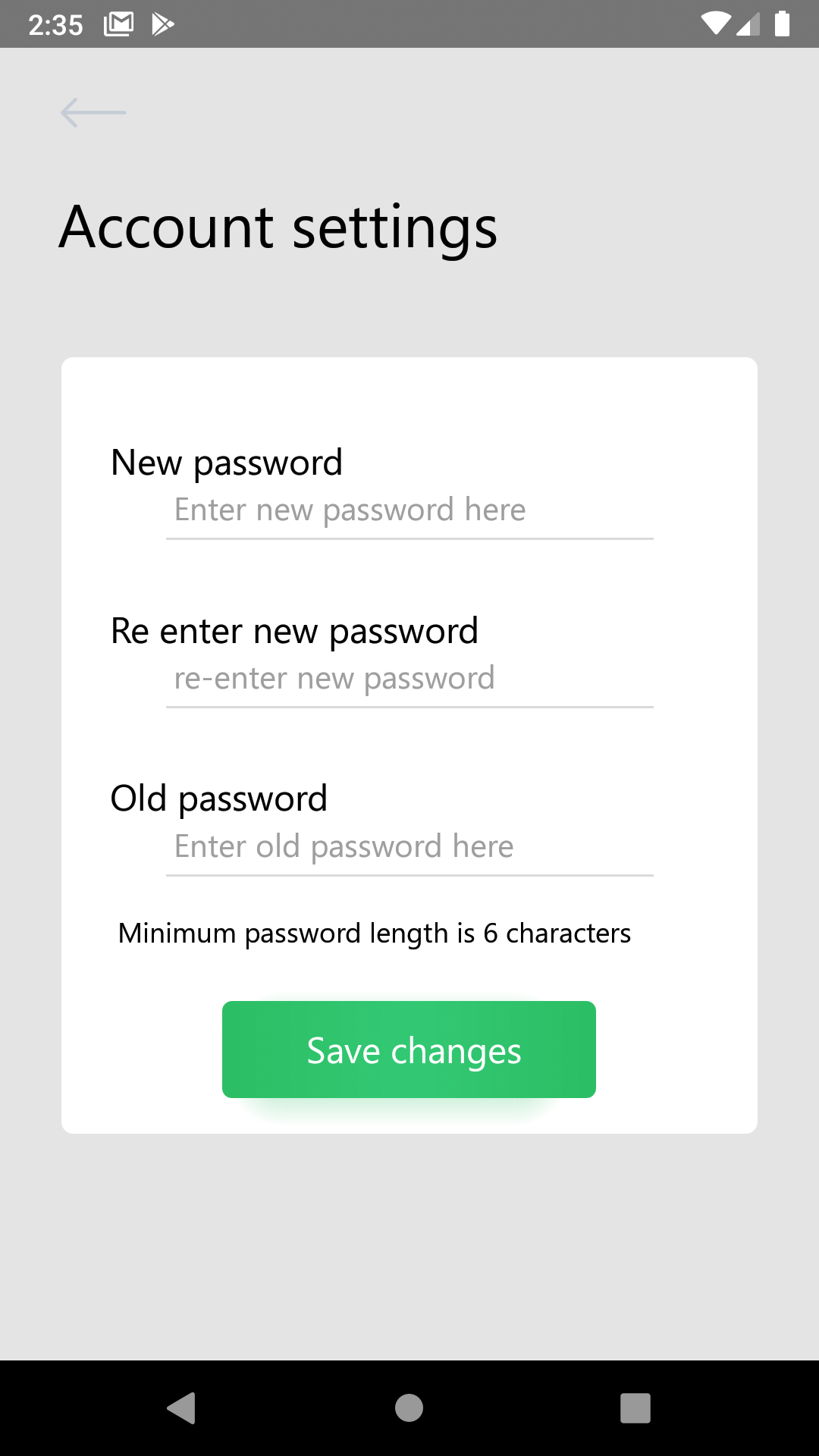
|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR8-MSG1 | Empty User Name | In-line message | ‘User’s name is required’ |

Table 5.1.5:2 Messages for Update User Details

### 5.1.6 Change Password (FR5)

* Users should be able to update their Password upon clicking on the ‘Change Password’ button on the ‘View User Details’ screen.
* Through the ‘Change Password’ function, the password should be able to update.

Mockup of the ‘Change Password’ function is given below



|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| New Password | To enter new password | Field type - Text Box  Field Length - Maximum 255  Data type - String  Mandatory |  |
| Re Enter New Password | To re enter the new password | Field type - Text Box  Field Length - Maximum 255  Data type - String  Mandatory | Check weather the two passwords entered are the same. If not display error message |
| Old Password | To re enter the old password | Field type - Text Box  Field Length - Maximum 255  Data type - String  Mandatory | Check whether the entered old password is valid. If not display error message |
| Buttons | | | |
| Save Changes | To save the updated password | Field Type - Button | Upon click, the password should be successfully updated in the system |

*Table 5.1.6:1: Input Field Validations for Change Password*

#### 5.1.6.1 Basic flow of events

* The use case starts when the user clicks on the ‘Change Password’ button available in the ‘View User Details’ screen
* User views the ‘Change Password’ screen
* User enters the new password
* User reenters the new password
* User enters the old password
* User clicks on the ‘Save Changes’ button
* The use case gets the end

#### 5.1.6.2 Alternative Paths

* N/A

#### 5.1.6.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.6:1)

#### 5.1.6.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.6:1)

#### 5.1.6.5 System integrations

* N/A

#### 5.1.6.6 Negative Paths

* If the user clicks on the ‘Save Changes’ button by leaving the Mandatory fields empty. System should display the error messages

#### 5.1.6.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR5-MSG1 | Empty New Password | In-line message | ‘New Password is required’ |
| FR5-MSG2 | Empty Re Enter New Password | In-line message | ‘Re enter password’ |
| FR5-MSG3 | Empty Old Password | In-line message | ‘Old password is required’ |

Table 5.1.6:2 Messages for Change Password

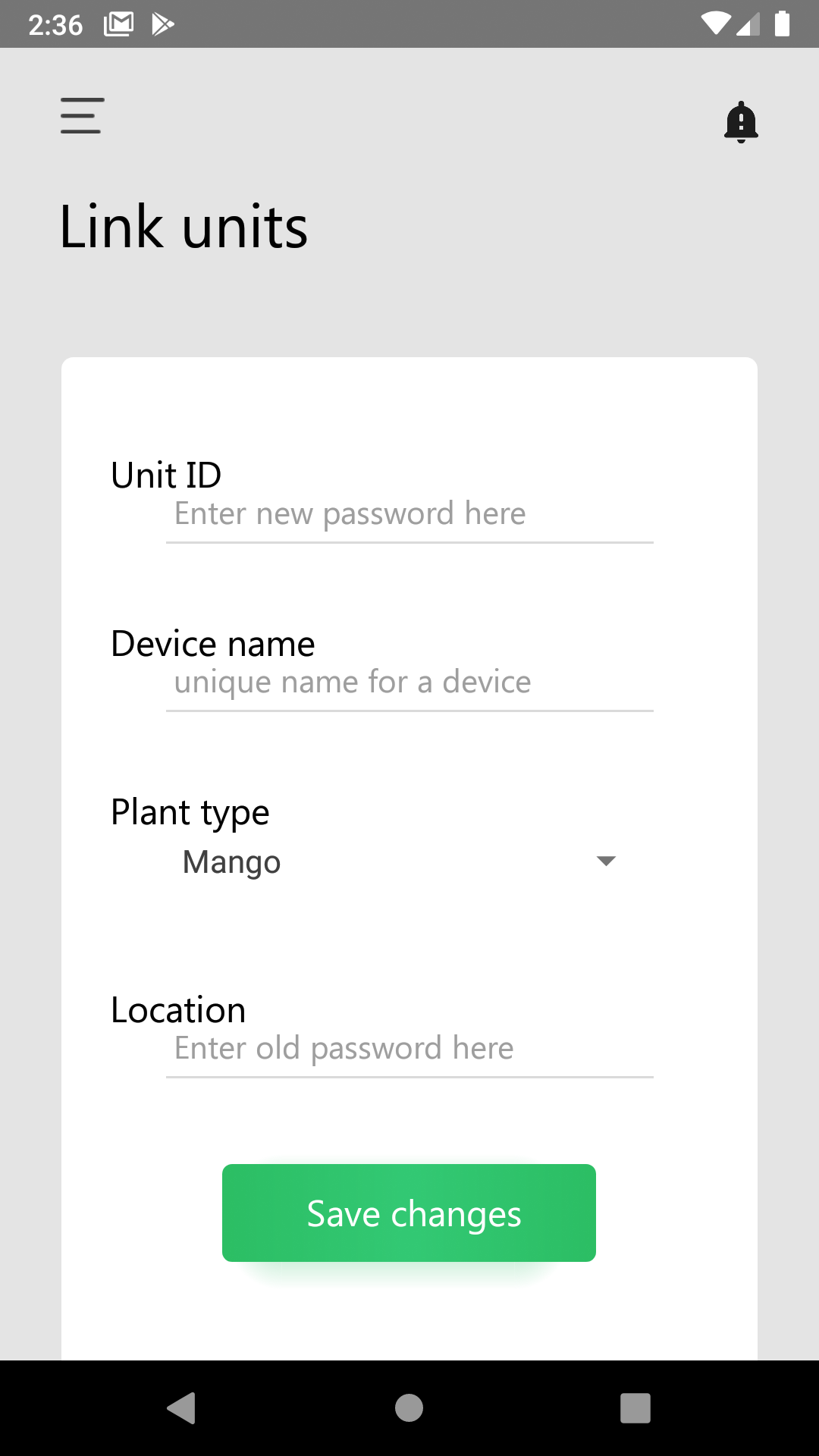
### 5.1.7 Reset Password (FR6)

## Manage Units

* Through these functions users can manage and receive notifications from the IoT units which are connected with his/her account

### 5.2.1 Link Units (FR8)

* Any registered user can link units.
* Upon clicking on the ‘Link Unit’ button in the navigation menu, the following mock-up should be displayed.



|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Unit ID | The unit ID in the IoT device is added here | Field type - Text Box  Field Length - 13 (fixed)  Data type - String  Mandatory |  |
| Device Name | The name of the unit | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default: Unit Id | If the user doesn’t enter a name unit id will be considered as the name |
| Plant Type | To select the plant type | Field Type – Dropdown  Default Value - Select Type  Mandatory |  |
| Location | To enter the location of the unit | Field Type – Dropdown  Default value – Colombo  Mandatory |  |
| Buttons | | | |
| Add | To complete unit linking process | Field Type - Button | Upon click, the unit should be successfully connected with the user and the user is redirected to the view all units screen. |

*Table 5.2.1:1: Input Field Validations for Link Unit*

#### 5.2.1.1 Basic flow of events

* The use case starts when the user clicks on the ‘Link Unit’ button in the Navigation menu
* User enters the Unit ID
* User enters the Device Name
* User enters the Plant Type
* User enters Location
* User clicks on the ‘Add’ button
* The use case ends

#### 5.2.1.2 Alternative Paths

* The user clicks on the back button

#### 5.2.1.3 Validations

* Refer the input field validation table for validations (Refer table 5.2.1:1)

#### 5.2.1.4 Actions

* Refer the input field validation table for button actions (Refer table 5.2.1:1)

#### 5.2.1.5 System integrations

* N/A

#### 5.2.1.6 Negative Paths

* If the user clicks on the ‘Add’ button by leaving the Mandatory fields empty. System should display the error messages

#### 5.2.1.7 Messages

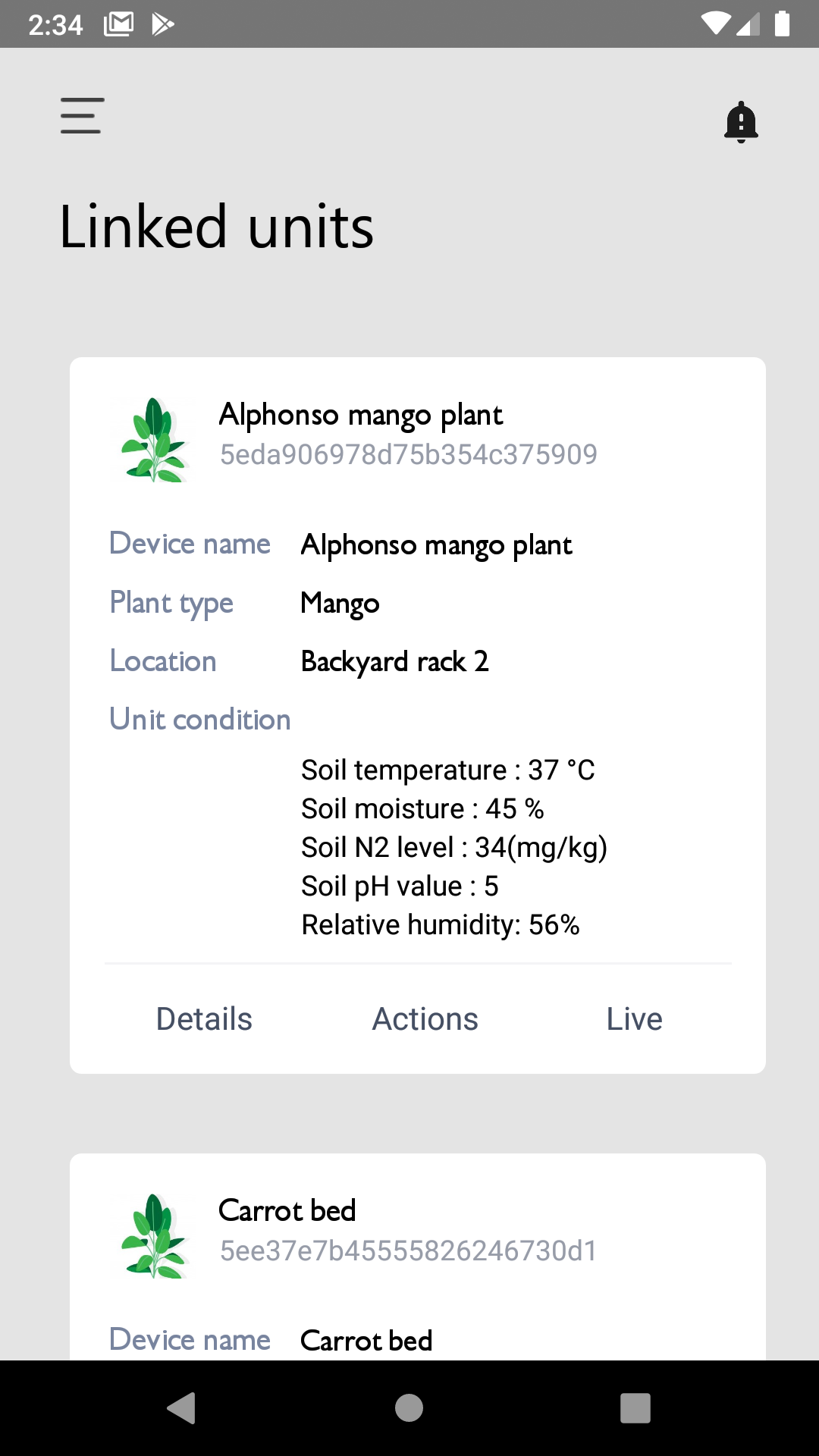
|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR8-MSG1 | Empty Unit ID | In-line message | ‘Unit ID is required’ |
| FR8-MSG2 | Incorrect Unit ID Length | In-line-message | ‘Incorrect Unit ID’ |
| FR8-MSG3 | Empty Plant Type | In-line message | ‘Selection is required’ |

*Table 5.2.1:2: Messages for Link Units*

### 

### 5.2.2 View All Units (FR9)

* After pressing view all units in the drawer menu, the system should navigate to this screen



*Table 5.2:2.1: View all linked units screen*

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Card | | | |
| Title | Name of the plant | N/A | The name user set for the unit (Usually a plant name) |
| ID | Unit id | Length: 16 characters | This should be placed right under the plant name |
| Mini image | Image related to the selected plant type | N/A | Align with the title |
| Device name | Same as the title | N/A | Same as the title |
| Location | Location of the unit set by the user to identify a plant specifically | String or null | This could be an empty string if the users didn’t set that |
| Unit condition | Current sensor values of the unit | Data type: Double | This contains some details about current sensor readings of the unit. 6 readings from the sensors |
| Buttons | | | |
| Details | Navigate to the View Single Unit details screen | Field Type – Button | Upon click, the user should be redirected to the view Single Unit details screen |
| Actions | Navigate to the control actuators screen | Field Type – Button | Upon click, the user is redirected to the control actuators screen. |

*Table 5.2.2:1: Input Field Validations for View All Units*

#### 5.2.2.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View all units’ at the drawer menu
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 5.2.2.2 Alternative Paths

* N/A

#### 5.2.2.3 Validations

* As the page loads, the table should be filled with existing Unit details in system

#### 5.2.2.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 5.6.2:1)

#### 5.2.2.5 System integrations

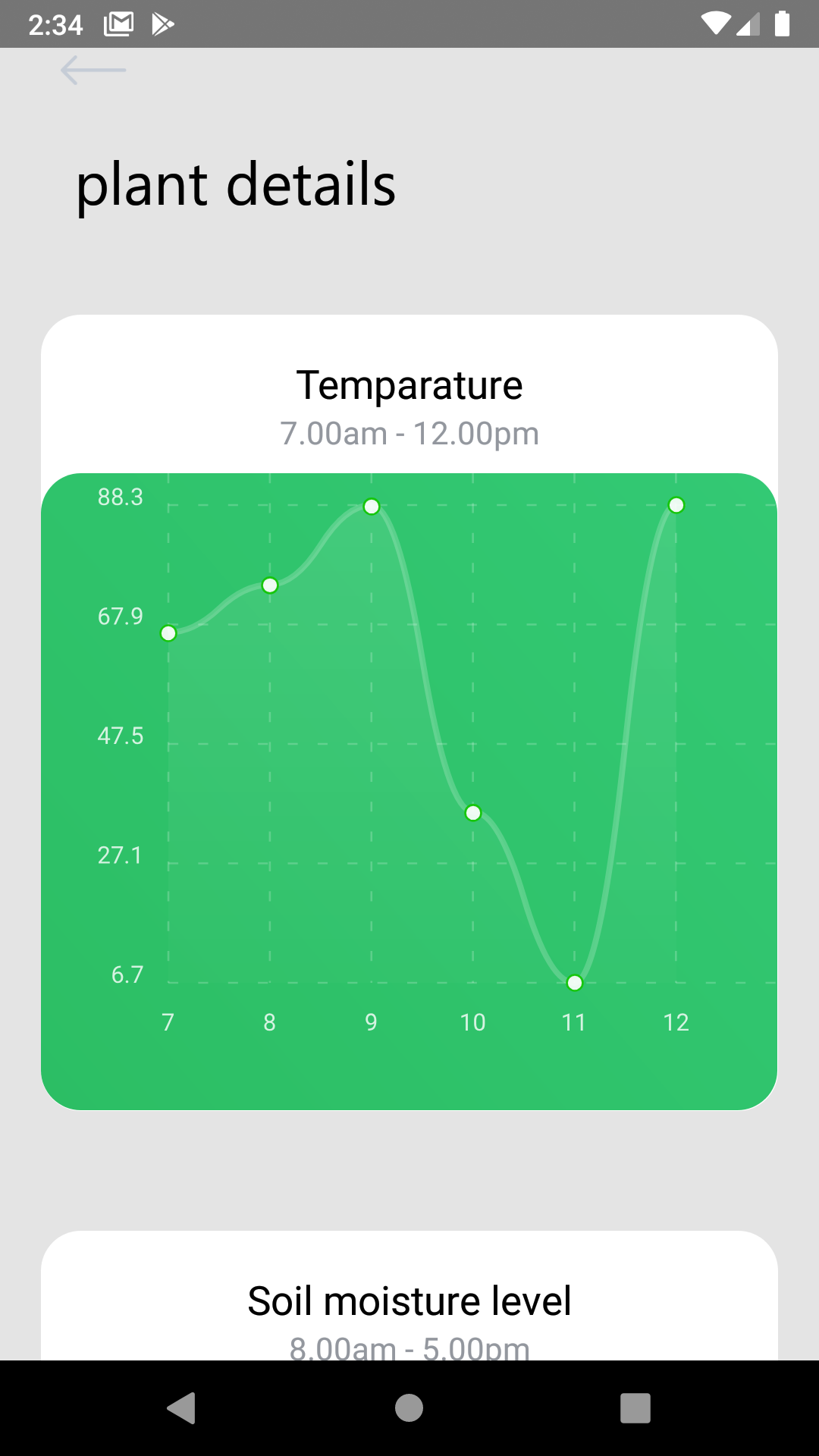
* N/A

#### 5.2.2.6 Negative Paths

* N/A

### 5.2.3 View Single Unit (FR10)

* Upon a press on the ‘Details’ text in a card which is assigned to a device user should be navigated into a screen which consists of several graphs

****

*Table 5.2.3:1: Plant details screen*

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Cards | | | |
| Sensor readings over time | Change of sensor readings over time | N/A | Users can view values related to soil moisture sensor, humidity sensor etc. |
| Fields | | | |
| Title | Title of the sensor reading | N/A | The title should clearly describe the sensor type |
| Time duration | The time gap between the first and the last reading | Should dynamically change according to time | The starting time should be 5 horse before the current time |
| Graph | To show the sensor readings over time | N/A | Readings should evenly be distributed over time |

*Table 5.2.3:1: Fields related to a single card*

#### 5.2.3.1 Basic Flow of events

* When the screen initiated all the graphs should be populated with the values fetched from the server according to units

#### 5.2.3.2 Alternative Paths

* N/A

#### 5.2.3.3 Validations

* If there is no such data present in the API response there should be a message which shows that error in the space dedicated for the graph

#### 5.2.3.4 Actions

* N/A

#### 5.2.3.5 System integrations

* N/A

#### 5.2.3.6 Negative Paths

* N/A

#### 5.2.3.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR10-MSG1 | Empty Result Set | In-line message | ‘No Units’ |

*Table 5.2.3:2: Messages for View All Units*

### Control Actuators

## Chat Module

### 5.3.1 Send New Message (FR12)

* Users can send messages to the admins regarding issues with their units. Upon clicking on the ‘Send Message’ button, the following mock-up should be displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Message | To enter the message | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Null  Mandatory |  |
| Buttons |  |  |  |
| Send | To complete the process | Field Type – Button | Upon click, the message should be successfully sent to user |

*Table 5.3.1:1: Input Field Validations for Send New Message*

#### 5.3.1.1 Basic Flow of events

* The use case starts when the user clicks on the ‘Send Message’ button
* User enters the message
* User clicks on the Send button

#### 5.3.1.2 Alternative Paths

* N/A

#### 5.3.1.3 Validations

* Refer the input field validation table for validations (Refer table 5.3.1:2)

#### 5.3.1.4 Actions

* N/A

#### 5.3.1.5 System Integration

* N/A

#### 5.3.1.6 Negative Paths

* If the user clicks on the ‘Send’ button by leaving the Mandatory fields empty. System should display the error messages

#### 5.3.1.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR12-MSG1 | Empty Message | In-line message | ‘Value is required’ |

*Table 5.3.1:2: Messages in Send New Message*

### 5.3.2 View All Messages (FR13)

* Upon click on the ‘Messages’ tab in navigation menu, following mock-up should be displayed

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| List |  |  |  |
| View Messages | this List should display all messages received or sent by the user. | Field Type – List Item  Order by – Recieved Date  Non-editable | All the users should be able to view the following details in a list item; Sender’s name, Date and Time, View Status |
| Attributes |  |  |  |
| Sender’s name | To display senders (user’s or admin’s) name |  |  |
| Date and Time | To display date and time the message was sent |  |  |
| View Status | To display weather the message is viewed or not |  | List item will be greyed out if it’s viewed by the user |
| Buttons |  |  |  |
| View | To view the complete message loop | Field Type – Button | Upon click, the user should be redirected to the detail view of the message. |

*Table 5.3.2.1:1: Input Field Validations for View All Messages*

#### 5.3.2.1 Basic Flow of events

* The use case starts when the user clicks on the ‘Messages’ tab in the navigation menu
* The user views all the messages in the list
* The use case ends

#### 5.3.2.2 Alternative Paths

* The user clicks on the list item (Message)

#### 5.3.2.3 Validations

* As the page load, the screen should be filled with existing Messages in the system

#### 5.3.2.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 5.3.2:2)

#### 5.3.2.5 System Integration

* N/A

#### 5.3.2.6 Negative Paths

* N/A

#### 5.3.2.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR13-MSG2 | Empty Result Set | In-line message | ‘No Messages’ |

*Table 5.3.2:2: Messages for View All Messages*

### 5.3.3 View Single Message (FR14)

* Users can view a single message by clicking the message in View All Messages screen. Accessible for all the users having existing messages.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| List | | | |
| Name | To display user’s name | Field type – Navbar heading |  |
| Message/Reply | To display the message or replies | Field type – List Item |  |
| Date and Time | To display the message sent time |  |  |
| Sent by | To display the user that sent the message or reply |  |  |
| Buttons |  |  |  |
| Back | Return to the previous screen | Field Type – Button | Upon click, the user should be redirected to the previous page. |

*Table 5.3.3:1: Input Field Validations for View Single Message*

#### 5.3.3.1 Basic Flow of events

* The use case starts when the user clicks on a message in the View All Messages screen
* The user views all the replies and the message on the screen
* The use case ends

#### 5.3.3.2 Alternative Paths

* Reply Message (FR15)

#### 5.3.3.3 Validations

* Refer the input field validations in the table for validations (Refer Table 5.3.3:2)
* As the page load, the screen should be filled with existing replies and the message

#### 5.3.3.4 Actions

* N/A

#### 5.3.3.5 System Integration

* N/A

#### 5.3.3.6 Negative Paths

* N/A

#### 5.3.3.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR14-MSG1 | Empty Result Set | In-line message | ‘No Messages’ |

*Table 5.3.3:2: Messages for View Single Message*

### 

### 5.3.4 Reply Message (FR15)

* Users can reply to received messages regarding their unit statuses. Upon clicking on the Message in ‘View All Messages screen’, the following mock-up should be displayed along with the ‘View Single Message’ screen.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Reply | To enter the reply | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Null  Mandatory |  |
| Buttons |  |  |  |
| Send | To complete the process | Field Type – Button | Upon click, the reply should be successfully sent to the admin |

*Table 5.3.4:1: Input Field Validations for Reply Message*

#### 5.3.4.1 Basic Flow of events

* The use case starts when the user clicks on a message in the ‘View All Messages’ screen
* User enters the reply
* User clicks on the ‘Send’ button

#### 5.3.4.2 Alternative Paths

* N/A

#### 5.3.4.3 Validations

* Refer the input field validation table for validations (Refer table 5.3.4:2)

#### 5.3.4.4 Actions

* N/A

#### 5.3.4.5 System Integration

* N/A

#### 5.3.4.6 Negative Paths

* If the user clicks on the ‘Send’ button by leaving the Mandatory fields empty. System should display the error messages

#### 5.3.4.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR35-MSG1 | Empty Reply | In-line message | ‘Value is required’ |

*Table 5.3.4:2: Messages in Reply Message*

## Alerts

**5.4.1** **View All Alerts (FR16)**

# Functional Requirements – Web App

## Manage Users

* Through this function Users can be viewed (Existing users in the system), created, updated, and deleted from the system

### Register User (FR17)

* Admins should be able to register in the system. It is mandatory for the admin to register in the system to use the system functions and all the fields are mandatory.
* All the Admins can edit/update his details by accessing the ‘Edit User’ function (explained under the FR22)
* Upon clicking on the ‘Register’ button (figure), the following mock-up should be displayed. After verifying the email, the rest of the user registration process will be carried out.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Name | To enter User’s name | Field type - Text Box  Field Length - Maximum 255  Data type - String  Mandatory |  |
| Email | To enter user’s email address | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory | When email is already exists in the database,  “Email Already exists”  inline error message will be prompt. |
| Password | To enter user’s password | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory |  |
| Confirm Password | To enter the user’s confirmed password. | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory |  |
| Buttons | | | |
| Signup | To save the users details | Field Type - Button | Upon click, the user should be successfully saved in the system. |

*Table 6.1.1:1: Input Field Validations for Signup-1*

#### 6.1.1.1 Basic flow of events

* The use case starts when the user clicks on the ‘Signup’ button.
* User views the ‘Sign Up’ screen
* User Enter the Name, email, password, confirm password
* User clicks on the ‘Sign Up’ button
* User directs to ‘login’ screen
* The use case gets the end

#### 6.1.1.2 Alternative Paths

* When User clicks on “Already have an account? Click here to Sign In” text, user will navigate to Login Screen.

#### 6.1.1.3 Validations

* Refer the input field validation table for validations (Refer table 6.1.1:1)

#### 6.1.1.4 Actions

* Refer the input field validation table for button actions (Refer table 6.1.1:1)

#### 6.1.1.5 System integrations

* N/A

#### 6.1.1.6 Negative Paths

* If the user clicks on the ‘Signup’ button by leaving the Mandatory fields empty. System should display the error messages

#### 6.1.1.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR20-MSG1 | Empty User Name | In-line message | ‘Name is required’ |
| FR20-MSG2 | Empty Email | In-line message | ‘Email is required’ |
| FR20-MSG3 | Empty Password | In-line message | ‘Password is required’ |
| FR20-MSG4 | Empty confirm password | In-line message | ‘Passwords doesn’t match’ |

*Table 6.1.1:2: Messages in Register Users*

Email Verification mockup

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Buttons | | | |
| Verify Email | To send a verification email | Field Type - Button | Upon click, the user will be sent a verification mail |
| Help Center | To access Help Center | Field Type – Button | Upon click, the user will be redirected to Help Center |

*Table 6.1.1:3: Input Field Validations for Verify Email*

#### 6.1.1.1 Basic flow of events

* The use case starts when the user clicks on the ‘Signup’ button (in figure 5.3.1:1)
* User clicks on the ‘Verify Email’ button
* The use case ends

#### 6.1.1.2 Alternative Paths

* N/A

#### 6.1.1.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.3:3)

#### 6.1.1.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.3:3)

#### 6.1.1.5 System integrations

* Email Integration

#### 6.1.1.6 Negative Paths

* N/A

#### 6.1.1.7 Messages

* N/A

Resend email verification mockup is as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Buttons | | | |
| Change | Allow the user to change the email | Field Type - Button | Upon click, the user should be redirected to the Signup page |
| Resend Link | To email the verification link again | Field Type – Button | Upon click, a verification link will be sent to the user’s email |

*Table 6.1.1:4: Input Field Validations for Resend Email*

#### 6.1.1.1 Basic flow of events

* The use case starts when the user clicks on the ‘Verify Email’ button (in figure 5.1.3:2)
* User clicks on the ‘Resend link’ button
* The use case ends

#### 6.1.1.2 Alternative Paths

* User clicks on the ‘Change’ button

#### 6.1.1.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.3:4)

#### 6.1.1.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.3:4)

#### 6.1.1.5 System integrations

* Email Integration

#### 6.1.1.6 Negative Paths

* N/A

#### 6.1.1.7 Messages

* N/A

### User (Admin) Login (FR18)

* Any user registered in the system should be able to login to the system.
* Users can log in to the system by entering valid email and password.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Email | To enter user’s email address | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory  Valid email address |  |
| Password | To enter password | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory  Data type - String  Mandatory |  |
| Buttons | | | |
| Sign in | To log in the to the system | Field Type – Button | Upon click, the user should be successfully logged in to the system |

*Table 6.1.2:1: Input Field Validations for Login*

#### 6.1.2.1 Basic flow of events

* The use case starts when the user opens the app.
* User views the ‘Login' screen
* User Enter the email, password.
* User clicks on the ‘Login' button
* User directs to ‘Dashboard’ screen
* The use case gets the end

#### 6.1.2.2 Alternative Paths

* Users can click on “Don’t have an account click here to Sign Up’ text available in the ‘Sign Up screen, the user will navigate to Sign Up Screen.
* Users can click on the “forgot password” user will direct to the forgot password screen.

#### 6.1.2.3 Validations

* Refer the input field validation table for validations (Refer table 5.1.4:1)

#### 6.1.2.4 Actions

* Refer the input field validation table for button actions (Refer table 5.1.4:1)

#### 6.1.2.5 System integrations

N/A

#### 6.1.2.6 Negative Paths

* If the user clicks on the ‘Login’ button by leaving the Mandatory fields empty. System should display the error messages.
* If a user enters an incorrect email or password, System should display the error messages.

#### 6.1.2.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR2-MSG1 | Incorrect Email and password | Alert message | “Incorrect email or password” |

*Table 6.1.2:2: Messages in User Login*

### Forgot Password (FR22)

* When the user forgets his/her password, the user can click on “forgot password” in the Login screen and request for a password reset.
* Then the system will send an email including a verify token.
* Users can enter the verify token in “verify token” screen and navigate to “reset password” screen.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Verify Token | To enter verify token | Field type - Text Box  Field Length - Maximum 4  Data type - String/Number  Mandatory |  |
| Buttons | | | |
| Submit | To send the token for password reset request | Field Type - Button | Upon click, the user should navigate to reset password screen. |

Table 6.1.3:1 Input Field Validations for Forgot Password

Then the system navigates to the “Reset Password” screen

### 6.1.4 Rest Password (FR21)

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| New Password | To enter new password | Field type - Text Box  Field Length - Maximum 16  Data type - String  Mandatory |  |
| Re Enter New Password | To re enter the new password | Field type - Text Box  Field Length - Maximum 1  Data type - String  Mandatory | Check whether the two passwords entered are the same. If not display error message |
| Buttons | | | |
| Save Changes | To save the updated password | Field Type - Button | Upon click, the password should be successfully updated in the system |

*Table 6.1.4:1: Input Field Validations for Change Password*

#### 6.1.4.1 Basic flow of events

* The use case starts when the user clicks on the ‘Forgot Password’ button available in the ‘Login’ screen
* User views the ‘Verify Token’ screen
* User enter verify Token
* User click on the “Submit” button
* User navigates to the “Reset Password” screen
* User enters the new password
* User reenters the new password
* User clicks on the ‘Save Changes’ button
* User navigate to the login screen
* The use case gets the end

#### 6.1.4.2 Alternative Paths

* User can click on “Resend Token” button in the “Verify Token” screen

#### 6.1.4.3 Validations

* Refer the input field validation table for validations (Refer table 6.1.4:1)

#### 6.1.4.4 Actions

* Refer the input field validation table for button actions (Refer table 6.1.4:1)

#### 6.1.4.5 System integrations

* N/A

#### 6.1.4.6 Negative Paths

* If the user clicks on the ‘Save Changes’ button in the “Reset password” screen by leaving the Mandatory fields empty. System should display the error messages
* If user clicks on the “Submit” button in the “Verify Token” screen by leaving it empty, system should display the error messages

#### 6.1.4.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR5-MSG1 | Empty New Password | In-line message | ‘New Password is required’ |
| FR5-MSG2 | Empty Re Enter New Password | In-line message | ‘Reenter password’ |
| FR5-MSG3 | Empty verification Token | In-line message | ‘Verification Token is required’ |

Table 6.1.4.7 Messages for Change Password

### View User (Admin) Details (FR19)

* Upon a click on the ‘Manage Users’ function, the list of existing users in the system should be displayed along with the search facility.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table | | | |
| User search results | This table should display a list of all user details that are already registered in the system. | Field Type – Table  Length – Should have pagination and 10 records per page  Order by – User ID (Default)  Non-editable | The user should be able to view the following details on the table; Email, Username, Full Name, Address, Contact Number, User role and gender |
| Columns | | | |
| Email | To display the list of user emails | Field type - table column  Non-editable  Display only |  |
| Username | To display usernames of users | Non-editable  Display only |  |
| Password | To display passwords of users | Non-editable  Display only |  |
| Buttons | | | |
| View | To view the detail view of the user | Field Type – Button | Upon click, the user should be redirected to the detail view of the user profile. |
| Edit | Edit details of the user. | Field Type – Button | Upon click, the user is redirected to the edit user information screen. |
| Delete | Delete details of the user. | Field Type – Button | Upon click, a popup message is shown to confirm delete or cancel the action. |

*Table 6.1.5:1: Input Field Validations for View User*

#### 6.1.5.1 Basic Flow of events

* The use case starts when the user clicks on the ‘Manage Users’ function
* The user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 6.1.5.2 Alternative Paths

* The user clicks on the reset button
* The user clicks on the Edit button

#### 6.1.5.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.2.1:1)
* As the page load, the table should be filled with existing users in system

#### 6.1.5.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.2.1:1)

#### 6.1.5.5 System integrations

* N/A

#### 6.1.5.6 Negative Paths

* N/A

### 6.1.6 Update User (Admin) Details (FR20)

* User details must be able to edit upon click on the ‘Edit’ button that is available in front of the User icon, in the Nav Bar.
* This function is accessible to all the registered users.
* Through the edit user function, the First Name Last Name, Gender, Contact Number, Username, and Password should be able to update.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| First Name | To update the first name | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Existing  Mandatory |  |
| Last Name | To update the last name | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Existing  Mandatory |  |
| Gender |  | Field Type - Radio Button  Default value – Existing  Mandatory |  |
| Contact Number | To edit contact number | Field Type - Text Box  Field Length - 12  Data Type - String  Default Value - Existing  Mandatory |  |
| Username | To enter username | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Existing  Mandatory |  |
| Password | To update password | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Existing  Mandatory |  |
| Reenter Password | To reenter the password to check if it’s correct | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value - Null  Mandatory |  |
| Buttons | | | |
| Update | To save the user details | Field Type - Button | Upon click, the user should be successfully saved in the system and display the success message |
| Cancel | To abort the user update process | Field Type – Button | Upon click, the user registration process should be abort and direct user to the previous screen |

*Table 6.1.6:1: Input Field Validations for Update User (Admin) Details*

#### 6.1.6.1 Basic flow of events

* The use case starts when the user clicks on the ‘Edit’ button available next to the user that needs to be updated
* User views the ‘Edit User’ screen
* User updates the required fields
* User clicks on the update button and view the success message
* The use case gets the end

#### 6.1.6.2 Alternative Paths

* The user clicks on the cancel button

#### 6.1.6.3 Validations

* Refer the input field validation table for validations (Refer table 6.1.6:1)

#### 6.1.6.4 Actions

* Refer the input field validation table for button actions (Refer table 6.1.6:1)

#### 6.1.6.5 System integrations

* N/A

#### 6.1.6.6 Negative Paths

* If the user clicks on the ‘Update’ button by leaving the Mandatory fields empty. System should display the error messages.

#### 6.1.6.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR20-MSG1 | Empty First Name | In-line message | ‘Value is required’ |
| FR20-MSG2 | Empty Last Name | In-line message | ‘Value is required’ |
| FR20-MSG3 | Empty Gender | In-line message | ‘Value is required’ |
| FR20-MSG4 | Empty Contact Number | In-line message | ‘Value is required’ |
| FR20-MSG5 | Empty username | In-line message | ‘Value is required’ |
| FR20-MSG6 | Empty Password | In-line message | ‘Value is required’ |
| FR20-MSG7 | Empty Reenter Password | In-line message | ‘Value is required’ |
| FR20-MSG8 | Mismatching Passwords | In-line message | ‘Incorrect Password’ |
| FR20-MSG9 | Success Message | Message Box | ‘User updated successfully’ |

*Table 6.1.6:2: Messages in Update User (Admin) Details*

### View Users (FR24)

* Upon a click on the ‘Manage Users’ function, the list of existing users in the system should be displayed along with the search facility.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table | | | |
| User search results | This table should display a list of all user details that are already registered in the system. | Field Type – Table  Length – Should have pagination and 10 records per page  Order by – User ID (Default)  Non-editable | The user should be able to view the following details on the table; Email, Username, Full Name, Address, Contact Number , User role and gender |
| Columns | | | |
| Email | To display the list of user emails | Field type - table column  Non-editable  Display only |  |
| Username | To display usernames of users | Non-editable  Display only |  |
| Full name | To display users’ full names | Non-editable  Display only |  |
| Role | To display user role details | Non-editable  Display only |  |
| Gender | To display users’ gender information details | Non-editable  Display only |  |
| Buttons | | | |
| View | To view the detail view of the user | Field Type – Button | Upon click, the user should be redirected to the detail view of the user profile. |
| Edit | Edit details of the user. | Field Type – Button | Upon click, the user is redirected to the edit user information screen. |
| Delete | Delete details of the user. | Field Type – Button | Upon click, a popup message is shown to confirm delete or cancel the action. |

*Table 6.1.7:1: Input Field Validations for View User*

#### 6.1.7.1 Basic Flow of events

* The use case starts when the user clicks on the ‘Manage Users’ function
* The user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 6.1.7.2 Alternative Paths

* The user clicks on the reset button
* The user clicks on the Edit button

#### 6.1.7.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.1.8:1)
* As the page load, the table should be filled with existing users in system

#### 6.1.7.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.1.8:1)

#### 6.1.7.5 System integrations

* N/A

#### Negative Paths

* N/A

### 6.1.8 Search Users (FR25)

* To view a user’s details, initially the relevant user’s details should be able to filter through a search function. As the search criteria email and Username should be available on the search screen.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Search By | To display list of categories required for searching | Field Type – Drop Down  Single selection  Default value - search by  Order by - alphabetical order  Mandatory for searching | Drop down should contain a list of components in user records to specify the field or type of data the user will be entering in the search field for searching records (Username & email) |
| Search | To input a string for searching | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value – Type Here  Mandatory for searching | The string that is entered here should match with the type specified in the search by field.  If the entered string is matched, the matching records are shown in the same interface.  If the entered string does not match with the search by field an error message is shown in the same interface. |
| Buttons | | | |
| Search | To search for users by submitting the values for the selected search criteria | Field Type – Button | Upon click, relevant search results should be populated on the same screen under the search function.  Upon click, the selected values in the search parameters should reset to default values.  Upon click if users are not available for the selected search criteria, should display an error message FR25-MSG1 |
| Reset | To reset the search criteria | Field Type – Button | Upon click, input search fields should get cleared and reset to default values |

*Table 6.1.8:1: Input Field Validations for User Search*

#### 6.1.8.1 Basic Flow of events

* The use case starts when the user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 6.1.8.2 Alternative Paths

* N/A

#### 6.1.8.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.1.9:1)
* As the page load, the table should be filled with existing users in system

#### 6.1.8.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.1.9:1)

#### 6.1.8.5 System integrations

* N/A

#### 6.1.8.6 Negative Paths

* If user clicks on the ‘Search’ button, without selecting any value from the Search by dropdown, should display the error message FR25-MSG2

#### 6.1.8.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR25-MSG1 | No matching users found | In-line message | ‘No matching users found’ |
| FR25-MSG2 | Empty Search field | In-line message | ‘Search Value is required’ |

*Table 6.1.8:2: Messages in Search Users*

### 6.1.9 View Single User (FR26)

## Manage Units

* Admin should have permission to access the ‘Manage Units’ function.
* Through this function unit details can be viewed (Existing units of registered users in the system), from the system.

### Search Units (FR28)

* To view a unit, initially the relevant unit should be able to filter through a search function, upon click on the ‘units’ function, the following search screen should be displayed. As the search criteria Name, and other fields should be available on the search screen.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Search By | To display a list of categories required for searching. | Field Type – Drop Down  Single selection  Default value - search by  Order by - alphabetical order  Mandatory for searching | Drop down should contain a list of components in units records to specify the field or type of data the user will be entering in the search field for searching records. |
| Search | To input a string for searching | Field Type - Text Box  Field Length - 255  Data Type - String  Default Value – Type Here  Mandatory for searching | The string that is entered here should match with the type specified in the search-by field.  If the entered string is matched, the matching records are shown in the same interface.  If the entered string does not match with the search by field an error message is shown in the same interface. |
| Buttons | | | |
| Search | To search for units by submitting the value for the selected search criteria | Field Type – Button | Upon click, relevant search results should be populated on the same screen under the search function.  Upon click, the selected values in the search parameters should be reset to default values.  Upon click if unit is not available for the selected search criteria, should display an error message FR27-MSG1 |
| Reset | To reset the search criteria | Field Type – Button | Upon click, input search fields should get cleared and reset to default values |

*Table 6.2.1:1: Input Field Validations for Search Unit*

#### Basic Flow of events

* The use case starts when the user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### Alternative Paths

* N/A

#### Validations

* Refer the input field validations in the table for validations (Refer Table 6.2.1:1)
* As the page load, the table should be filled with existing users in system

#### Actions

* Refer the input field validations in the table for button actions (Refer Table 6.2.1:1)

#### System integrations

* N/A

#### Negative Paths

* If user clicks on the ‘Search’ button, without selecting any value from the Search by dropdown, should display the error message FR28-MSG2

#### Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR27-MSG1 | No matching units found | In-line message | ‘No matching units found’ |
| FR27-MSG2 | Empty Search field | In-line message | ‘Search Value is required’ |

*Table 6.2.1:2: Messages in Search Units*

### View All Units (FR27)

* Upon a click on the ‘View Units’ function, the list of existing units in the system should be displayed along with the search facility.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table | | | |
| Unit search results | This table should display a list of all units details that are already registered in the system. | Field Type – Table  Length – Should have pagination and 20 records per page  Order by – Unit ID (Default)  Non-editable | The unit should be able to view the following details on the table; Title, Location, Soil Moisture readings, Temperature readings, Humidity readings, Light intensity sensor readings and Water motor actuator. |
| Columns | | | |
| Title | Name of the plant | N/A | The name user set for the unit (Usually a plant name) |
| ID | Unit id | Length: 16 characters | This should be placed right under the plant name |
| Mini image | Image related to the selected plant type | N/A | Align with the title |
| Device name | Same as the title | N/A | Same as the title |
| Location | To display units’ locations | Non-editable  Display only |  |
| Soil Moisture readings | To display soil moisture level last reading with time and current reading with time. | Non-editable  Display only |  |
| Temperature readings | To display temperature level last readings with time and current reading with time. | Non-editable  Display only |  |
| Humidity readings | To display humidity level last readings with time and current reading with time. | Non-editable  Display only |  |
| Light intensity sensor readings | To display light intensity sensors’ last readings with time and current reading with time. | Non-editable  Display only |  |
| Water motor actuator readings | To display water motor actuator activated or not activated. | Non-editable  Display only |  |
| Buttons | | | |
| View | To view the detail view of the unit | Field Type – Button | Upon click, the unit should be redirected to the detail view of the unit profile. |
| Edit | Edit details of the unit. | Field Type – Button | Upon click, the unit is redirected to the edit unit information screen. |
| Delete | Delete details of the unit. | Field Type – Button | Upon click, a popup message is shown to confirm delete or cancel the action. |

*Table 6.2.2:1: Input Field Validations for View All Units*

#### 6.2.2.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View units’ function
* The user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 6.2.2.2 Alternative Paths

* N/A

#### 6.2.2.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.2.2:1)
* As the page load, the table should be filled with existing units in system

#### 6.2.2.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.2.2:1)

#### 6.2.2.5 System integrations

* N/A

#### 6.2.2.6 Negative Paths

* N/A

### View Single Unit (FR29)

* Upon a press on the ‘Details’ function, should be navigated into a page which consists of more details about the particular unit

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Cards | | | |
| Sensor readings over time | Change of sensor readings over time | N/A | Users can view values related to soil moisture sensor, humidity sensor etc. |
| Fields | | | |
| Title | Title of the sensor reading | N/A | The title should clearly describe the sensor type |
| Time duration | The time gap between the first and the last reading | Should dynamically change according to time | The starting time should be 5 horse before the current time |
| Graph | To show the sensor readings over time | N/A | Readings should evenly be distributed over time |

*Table 6.2.3:1: Fields related to view a single unit*

#### 6.2.3.1 Basic Flow of events

* When the screen initiated all the graphs should be populated with the values fetched from the server according to units

#### 6.2.3.2 Alternative Paths

* N/A

#### 6.2.3.3 Validations

* If there is no such data present in the API response there should be a message which shows that error in the space dedicated for the graph

#### 6.2.3.4 Actions

* N/A

#### 6.2.3.5 System integrations

* N/A

#### 6.2.3.6 Negative Paths

* N/A

#### 6.2.3.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR29-MSG1 | Empty Result Set | In-line message | ‘No Units’ |

*Table 6.2.3:2: Messages for View a unit*

### 

### View Vulnerable Units (FR30)

* Upon a click on the ‘View Vulnerable Units’ function, the list of existing vulnerable units in the system should be displayed when the button in the navigation menu is clicked.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table | | | |
| Unit search results | This table should display a list of all vulnerable units details that are already registered in the system. | Field Type – Table  Length – Should have pagination and 20 records per page  Order by – Unit ID (Default)  Non-editable | The unit should be able to view the following details on the table, Title, Location, Soil Moisture readings, Temperature readings, Humidity readings, Light intensity sensor readings and Water motor actuator. |
| Columns | | | |
| Title | Name of the plant | N/A | The name user set for the unit (Usually a plant name) |
| ID | Unit id | Length: 16 characters | This should be placed right under the plant name |
| Mini image | Image related to the selected plant type | N/A | Align with the title |
| Device name | Same as the title | N/A | Same as the title |
| Location | To display vulnerable units’ locations | Non-editable  Display only |  |
| Soil Moisture readings | To display soil moisture level last reading with time and current reading with time. | Non-editable  Display only |  |
| Temperature readings | To display temperature level last readings with time and current reading with time. | Non-editable  Display only |  |
| Humidity readings | To display humidity level last readings with time and current reading with time. | Non-editable  Display only |  |
| Light intensity sensor readings | To display light intensity sensors’ last readings with time and current reading with time. | Non-editable  Display only |  |
| Water motor actuator readings | To display water motor actuator activated or not activated. | Non-editable  Display only |  |
| Buttons | | | |
| View | To view the detail view of the vulnerable unit | Field Type – Button | Upon click, the unit should be redirected to the detail view of the vulnerable unit profile. |
| Edit | Edit details of the vulnerable unit. | Field Type – Button | Upon click, the vulnerable unit is redirected to the edit unit information screen. |
| Delete | Delete details of the vulnerable unit. | Field Type – Button | Upon click, a popup message is shown to confirm delete or cancel the action. |

*Table 6.2.4:1: Input Field Validations for View All Vulnerable Units*

#### 6.2.4.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View units’ function
* The user selects a search criterion from the ‘Search by’ dropdown list
* The user clicks on the search button
* The user views the search results on the table
* The use case gets the end

#### 6.2.4.2 Alternative Paths

* N/A

#### 6.2.4.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.2.4:1)
* As the page load, the table should be filled with existing units in system

#### 6.2.4.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.2.4:1)

#### 6.2.4.5 System integrations

* N/A

#### 6.2.4.6 Negative Paths

* N/A

## Chat Module

### 6.3.1 Send Message (FR33)

* Admin can send messages to users regarding their unit statuses. Upon clicking on the ‘Send Message’ button, the following mock-up should be displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Message | To enter the message | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Null  Mandatory |  |
| Buttons |  |  |  |
| Send | To complete the process | Field Type – Button | Upon click, the message should be successfully sent to user |

*Table 6.3.1:1: Input Field Validations for Send Message*

#### 6.3.1.1 Basic Flow of events

* The use case starts when the user clicks on the ‘Send Message’ button
* User enters Message
* User clicks on the Send button

#### 6.3.1.2 Alternative Paths

* User clicks on the cancel button

#### 6.3.1.3 Validations

* Refer the input field validation table for validations (Refer table 6.3.1:1)

#### 6.3.1.4 Actions

* N/A

#### 6.3.1.5 System Integration

* N/A

#### 6.3.1.6 Negative Paths

* If the user clicks on the ‘Send’ button by leaving the Mandatory fields empty. System should display the error messages

#### 6.3.1.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR31-MSG1 | Empty Message | In-line message | ‘Value is required’ |
| FR31-MSG2 | Success Message | Message Box | ‘Message sent successfully’ |

*Table 6.3.1:2: Messages in Send Message*

### 

### View All Messages (FR31)

* Upon click on the ‘View Messages’ button, following mock-up should be displayed

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table |  |  |  |
| View Messages | this table should display all messages received by the admin. | Field Type – table  Order by – Date  Non-editable | All the admins should be able to view the following details on the table; User’s name, Date and Time, View Status, Priority |
| Columns |  |  |  |
| Name | To display senders (user’s) name | Field type - table column |  |
| Date and Time | To display date and time the message was sent | Field type – table column |  |
| View Status | To display weather, it is viewed or not | Field type - table column |  |
| Priority | Message Priority | Field type - table column |  |
| Buttons | | | |
| View | To view the complete message loop | Field Type – Button | Upon click, the user should be redirected to the detail view of the message. |
| Delete | Delete Message. | Field Type – Button | Upon click, a popup message is shown to confirm delete or cancel the action. |

*Table 6.3.2.1:1: Input Field Validations for View All Messages*

#### 6.3.2.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View Messages’ function
* The user views all the messages on the table
* The use case ends

#### 6.3.2.2 Alternative Paths

* The user clicks on the View button
* The user clicks on the Delete button

#### 6.3.2.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.3.2:2)
* As the page load, the table should be filled with existing Messages in the system

#### 6.3.2.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.3.2:3)

#### 6.3.2.5 System Integration

* N/A

#### 6.3.2.6 Negative Paths

* N/A

#### 6.3.2.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR32-MSG2 | Empty Result Set | In-line message | ‘Required results are empty’ |

*Table 6.3.2:2: Messages for View All Messages*

### 6.3.3 View Single Message (FR32)

* Users can view a single message by clicking the view button on the relevant message in the View Messages page. Accessible for all the admins.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| List | | | |
| Name | To display user’s name | Field type – Heading |  |
| Message/Reply | To display the message or replies | Field type – List Item |  |
| Date and Time | To display the message sent time |  |  |
| Sent by | To display the user that sent the message or reply |  |  |
| Buttons |  |  |  |
| Back | Return to the previous page | Field Type – Button | Upon click, user should be redirected to the previous page. |

*Table 6.3.3:1: Input Field Validations for View Single Message*

#### 6.3.3.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View’ button
* The user views all the replies and the message on the screen
* The use case ends

#### 6.3.3.2 Alternative Paths

* N/A

#### 6.3.3.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.3.3:2)
* As the page load, the screen should be filled with existing replies and the message

#### 6.3.3.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.3.3:3)

#### 6.3.3.5 System Integration

* N/A

#### 6.3.3.6 Negative Paths

* N/A

#### 6.3.3.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR32-MSG2 | Empty Result Set | In-line message | ‘Required results are empty’ |

*Table 6.3.3:2: Messages for View Single Messages*

### 

### 6.3.4 Reply Message (FR34)

* Admin can reply to received messages regarding their unit statuses. Upon clicking on the ‘View’ button in all messages screen, the following mock-up should be displayed.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Fields | | | |
| Reply | To enter the reply | Field type - Text Box  Field Length - Maximum 255  Data type - String  Default Value - Null  Mandatory |  |
| Buttons |  |  |  |
| Send | To complete the process | Field Type – Button | Upon click, the reply should be successfully sent to user |

*Table 6.3.4:1: Input Field Validations for Reply Message*

#### 6.3.4.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View’ button
* User enters the reply
* User clicks on the Send button

#### 6.3.4.2 Alternative Paths

* N/A

#### 6.3.4.3 Validations

* Refer the input field validation table for validations (Refer table 6.3.2:2)

#### 6.3.4.4 Actions

* N/A

#### 6.3.4.5 System Integration

* N/A

#### 6.3.4.6 Negative Paths

* If the user clicks on the ‘Send’ button by leaving the Mandatory fields empty. System should display the error messages

#### 6.3.4.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR33-MSG1 | Empty Reply | In-line message | ‘Value is required’ |
| FR33-MSG2 | Success Message | Message Box | ‘Reply sent successfully’ |

*Table 6.3.4:2: Messages in Reply Message*

### 

### View New Messages (FR34)

* Upon click on the ‘View New Messages’ button, following mock-up should be displayed. Also this should be available in the dashboard.

|  |  |  |  |
| --- | --- | --- | --- |
| Screen/Report Field | Description | Type/Value/Validations | Remarks/Special Validations |
| Table |  |  |  |
| View New Messages | this table should display all the new messages received by the admin which were not read. | Field Type – table  Order by – Date  Non-editable | All the admins should be able to view the following details on the table; User’s name, Date and Time and Priority |
| Columns |  |  |  |
| Name | To display senders (user’s) name | Field type - table column |  |
| Date and Time | To display date and time the message was sent | Field type – table column |  |
| Priority | Message Priority | Field type - table column | This should be denoted by using colored dots,  High priority: Red  Medium: Yellow  Low: Green |
| Buttons | | | |
| View | To view the complete message loop | Field Type – Button | Upon click, the user should be redirected to the detail view of the message. |

*Table 6.3.5:1: Input Field Validations for View All Messages*

#### 6.3.5.1 Basic Flow of events

* The use case starts when the user clicks on the ‘View New Messages’ function
* The user views all the new (unread) messages on the table
* The use case ends

#### 6.3.5.2 Alternative Paths

* The user clicks on the View button

#### 6.3.5.3 Validations

* Refer the input field validations in the table for validations (Refer Table 6.3.5:1)
* As the page load, the table should be filled with existing new messages in the system

#### 6.3.5.4 Actions

* Refer the input field validations in the table for button actions (Refer Table 6.3.5:1)

#### 6.3.5.5 System Integration

* N/A

#### 6.3.5.6 Negative Paths

* N/A

#### 6.3.5.7 Messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message ID | Description | Message Type | Content |
| FR35-MSG1 | Empty Result Set | In-line message | ‘Required results are empty’ |

*Table 6.3.5:2: Messages for View New Messages*

# Non-Functional Requirements

### 6.1 Availability

* System will be available on both mobile and web with any internet accessible device.

### 6.2 Usability

* Simple and easy to use interface throughout the mobile and web application.

### 6.3 Dependability

* Backups backend database to a database in the cloud periodically

### 6.4 Security

* Only administrators can access the web application and only registered users can enter the mobile app.
* Assumption- Administrator is a trustworthy person.

### 6.5 Maintainability

* Mobile app can be updated easily through the Play Store.
* Repairs on the IoT device can be done easily because of the modular nature of the IOT device.

### 6.6 Performance

* Database will update its data almost instantly when an action takes place. So, the users will always get real-time information from the database. Response time of the system is minimized

### 6.7 Security

* Control of access to the Admin System’s resources, its data and operating system files to prevent unauthorized persons from accessing sensitive data of the system. Sensitive data is backed up frequently

### 6.8 Capacity

* Any number of users can login at the same time. Allocated storage space per user is unlimited.

# System Requirements

**Server for hosting.**

* 4 x 1.6 GHz Processor
* 16 GB RAM
* 50 GB of Minimum HDD
* Unlimited Bandwidth
* Internet Connection

# Stakeholders & Use Case Diagrams

## Stakeholders

### Admin

* Login
* Manage Users
* Manage Units
* Send Messages
* Reply Messages

### Unregistered User

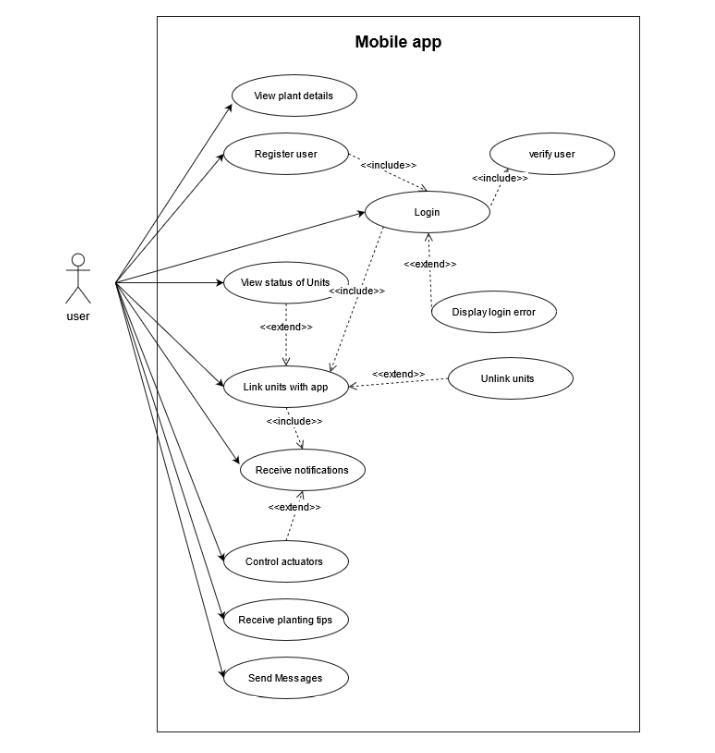
* Register to the system

### Registered User

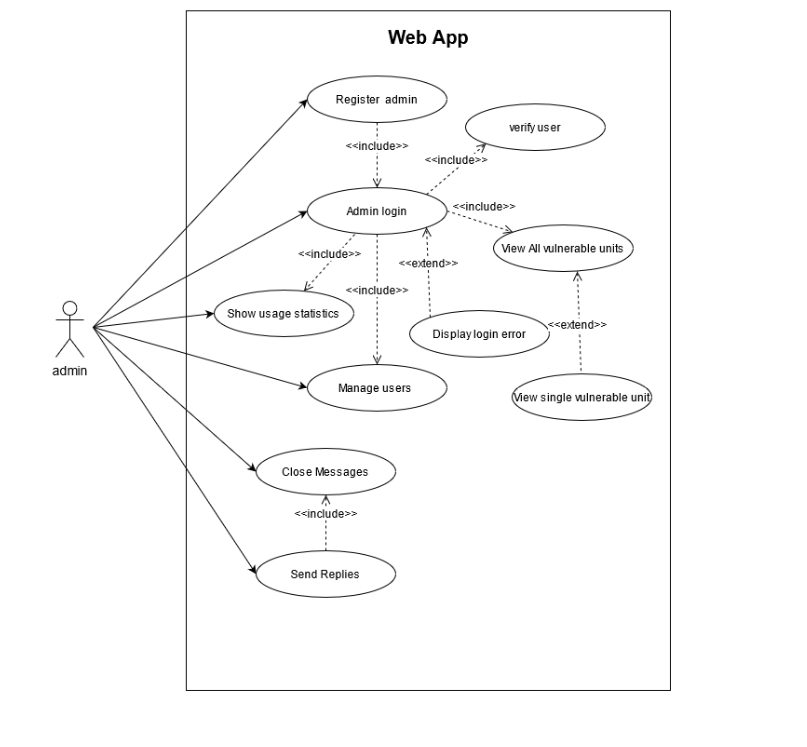
* Login
* Add units
* Manage User profile
* View Units and details
* Control actuators
* Send Messages
* Reply Messages

## Use Case Diagrams

### Users



### Admin



# System Architecture

## Architecture Diagram

## Component Diagram

* + 1. **Components and their responsibilities**

|  |  |
| --- | --- |
| Components | Responsibilities |
| User Component | * Add User * Edit User * Search User * View User * Register User * Login User |
| Units Component | * View Units * Edit Units * Control Actuators |
| Notification Component | * Notify users about garden status * Reminders |
| Security Component | * Verifies users based on their credentials * System security |
| Statistical Component | * Generate line charts |
| Admin Component | * Manage User * Manage Units |
| Chat Component | * Send Message * Reply Message * View Messages |

*Table 9.2.1:1: Components and their responsibilities*

* + 1. **Component Diagram**

# System Design

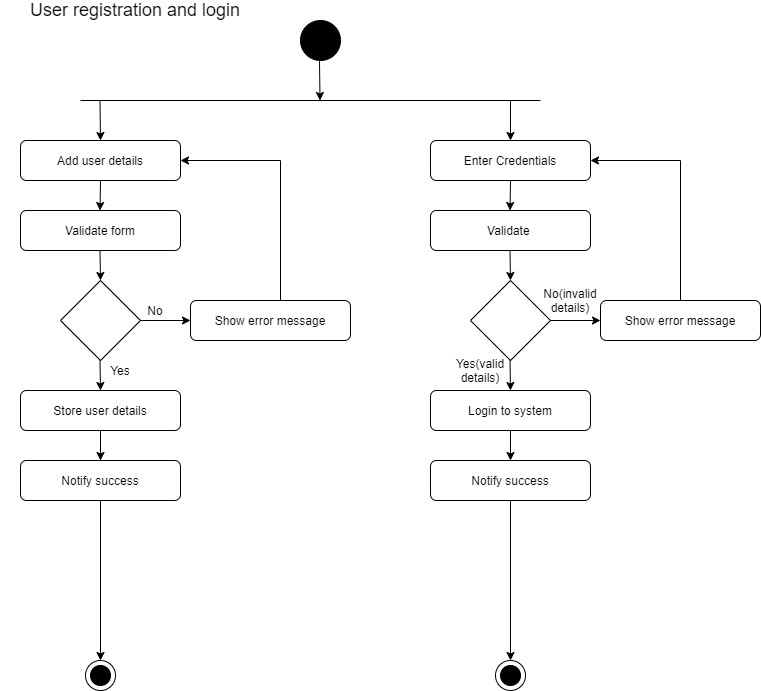
## Class Diagram

## Sequence Diagrams

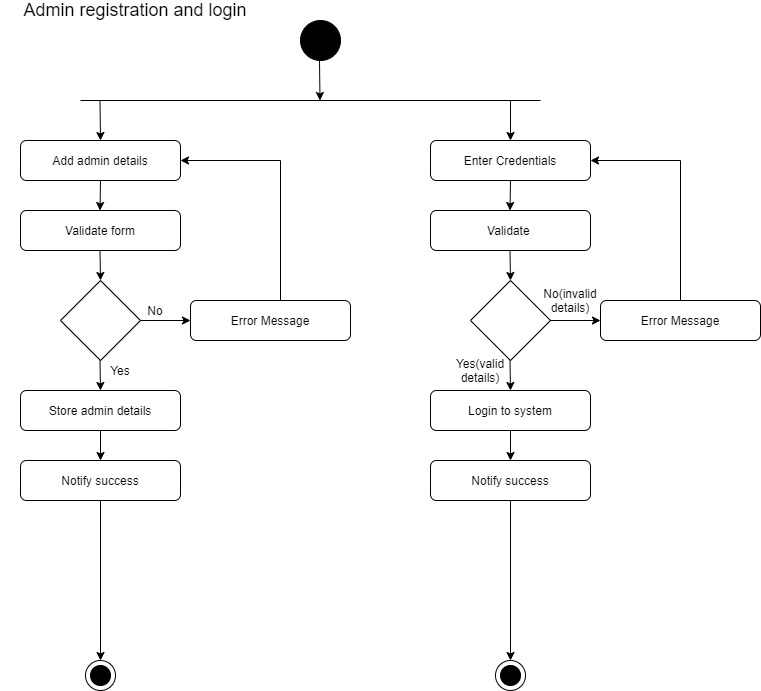
* + 1. **Sequence diagram 1**
    2. **Sequence diagram 2**

## Activity Diagrams

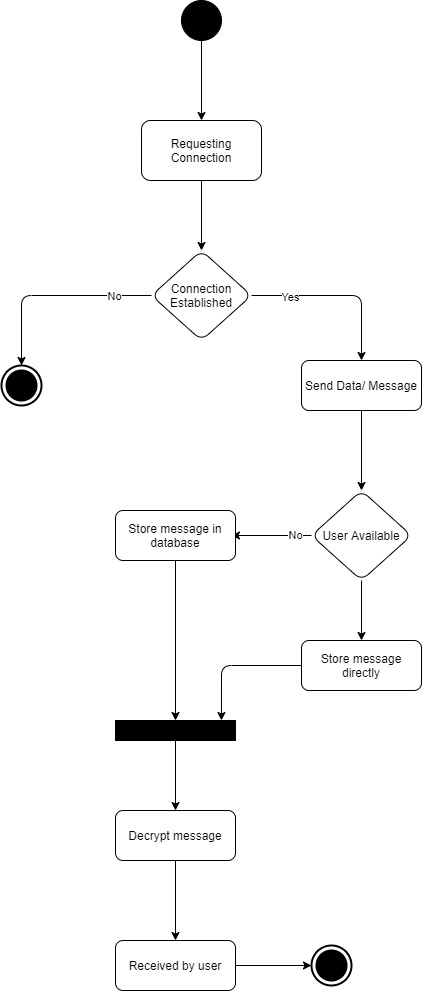
### Login Activity Diagram-Mobile App



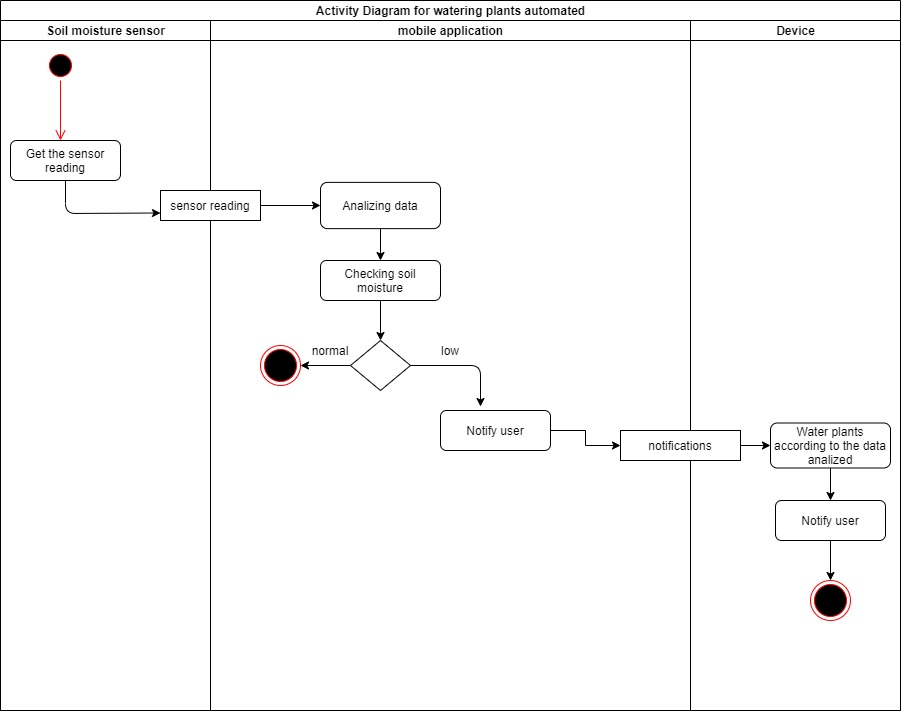
### Login Activity Diagram-Web App



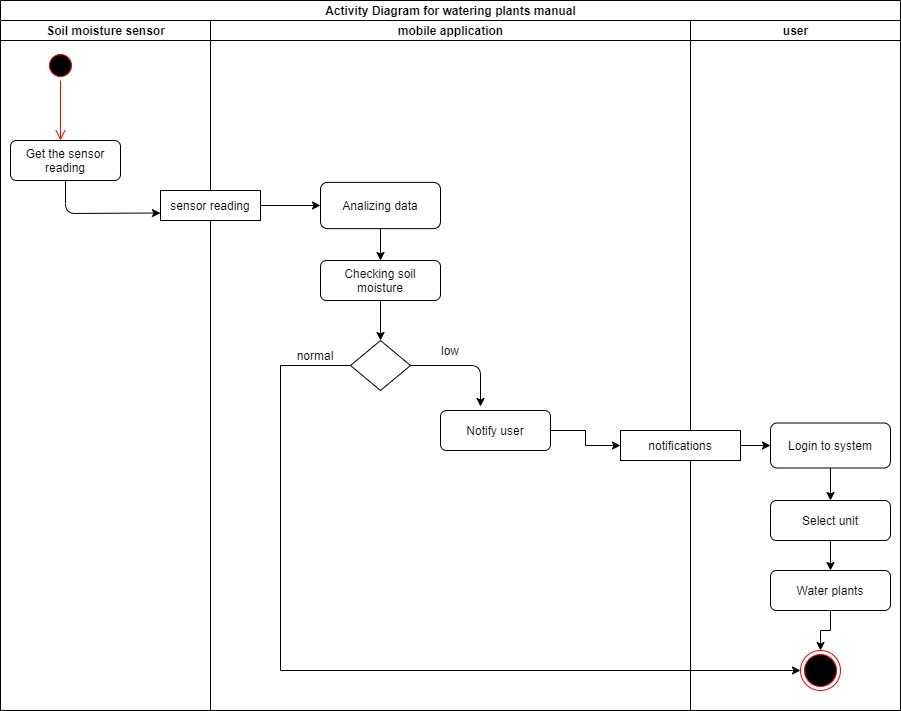
### Chat module Activity Diagram



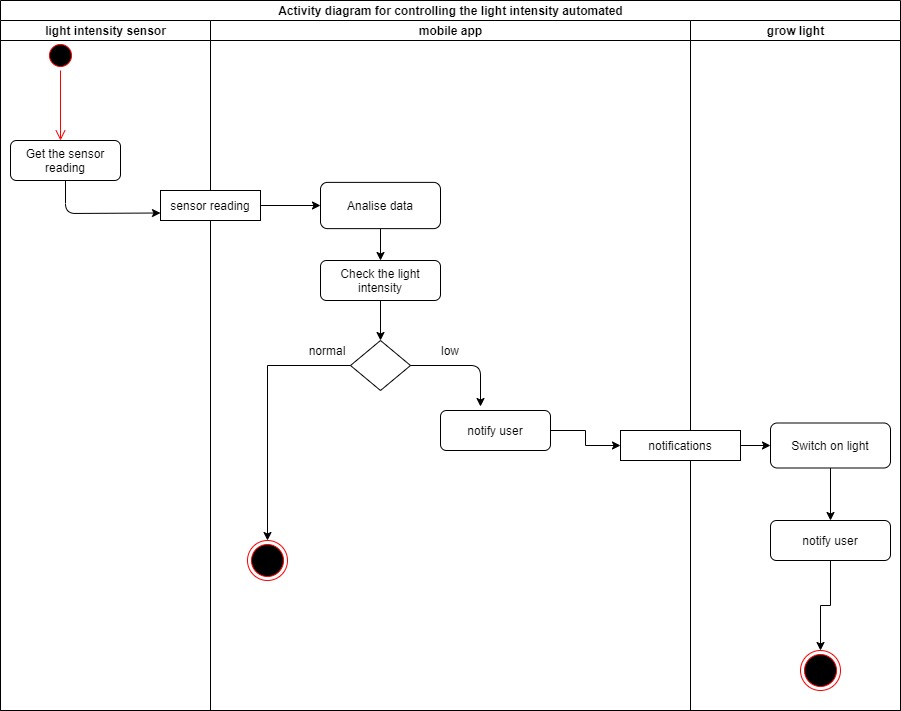
### Automated Watering of Plants Activity Diagram



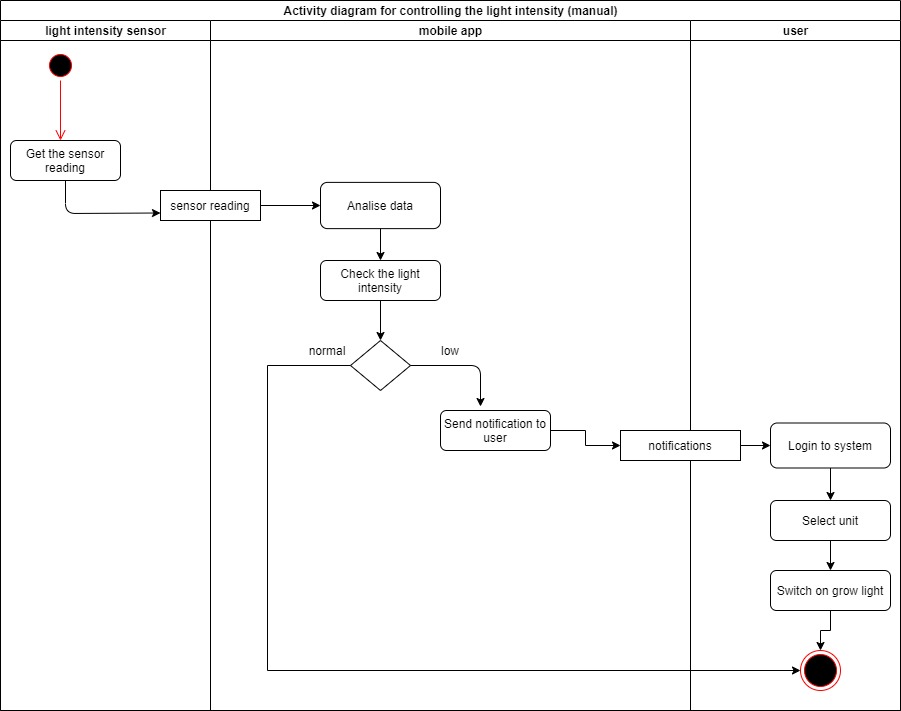
### Manual Watering of Plants Activity Diagram



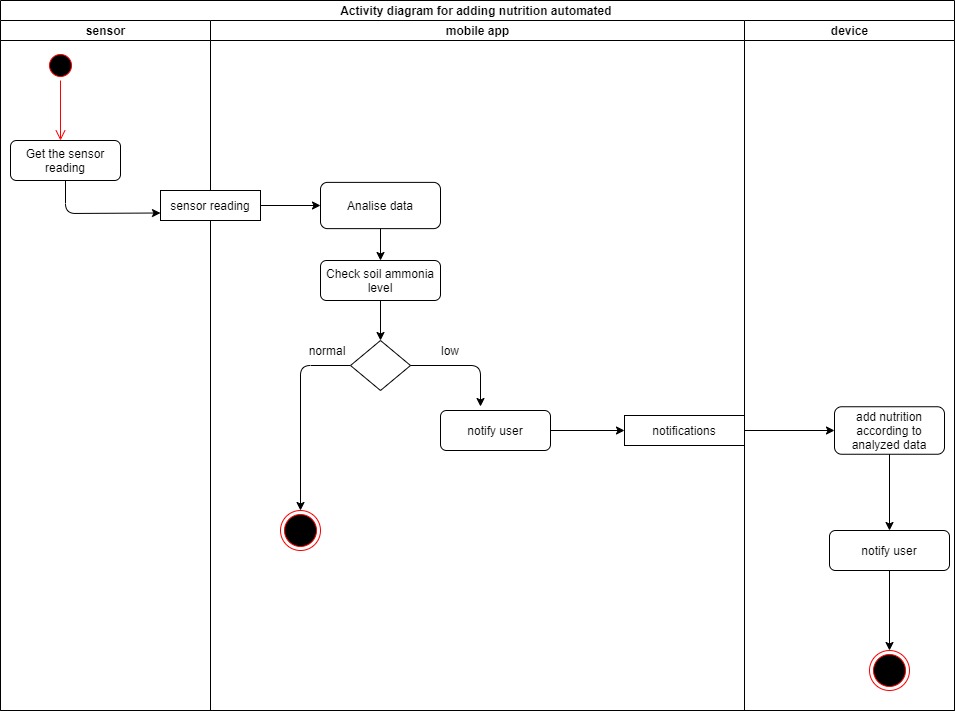
### Controlling light intensity automated Activity Diagram



### Controlling light intensity manual Activity Diagram

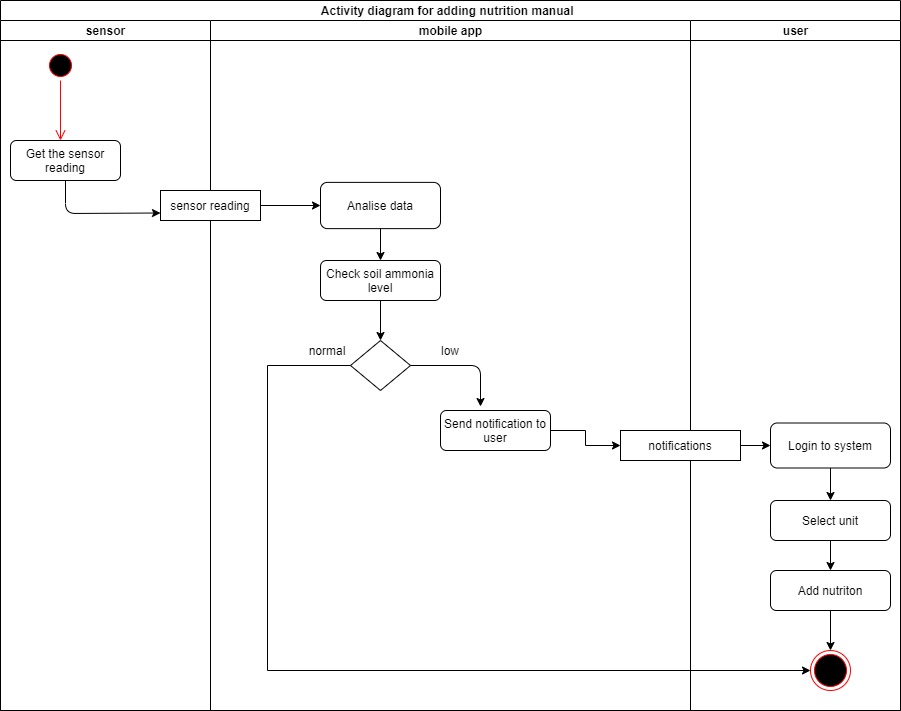


### Adding nutrients automated Activity Diagram



### 

### Adding nutrition manual Activity Diagram



# User Interface Flow Diagram

# Approval

|  |  |  |  |
| --- | --- | --- | --- |
| Stakeholder Name | Designation | Date | Signature |
| Dr Dinuni K Fernando |  |  |  |