Software Design Document for project Smart Planting

Randa Osama, Nour El Huda Ashraf, Amina Yasser, Salma Abd El Fatah,
Supervised by Dr. Ashraf Abd El Raouf, Eng. Noha El Masry

March 6, 2020

1 Introduction

1.1 Purpose

The main purpose of this document is to represent the architecture and the system design of our Smart Planting system. Our proposed system is an automated greenhouse system that control the LED lights and fans using real time cameras. Our system is accompanied with a web application that enables the user to monitor their plant's growth health. We also provide a fulfilled illustration about each stage inputs and outputs, along with the development process and a full illustration about the system components and their interaction together.

1.2 Scope

This document targets farmers and landowners that would use the Smart Planting system to help them monitoring their plants' growth rate and to get notified with their land's important updates which will save much time instead of visiting their land constantly. Meanwhile, the system provides a less cost equipment for producing more plants while farmers and landowners having double of their normal income, with less percentage of plants loss during their growth. Our proposed system tends to automate the greenhouse for speeding up the plant growth using LED lights, monitoring the plant health, increasing their production and detecting some certain diseases. Frames are extracted from the real time cameras inside the greenhouse then performs pre-processing to enhance the image, converting the image from RGB into HSV color format and providing all the needed masking, then feature extraction takes place using HOG then classification using One-Class SVM. Meanwhile, The system have the ability to learn different types of diseases to enhance the accuracy in the future.

1.3 Overview

This SDD document includes 8 main sections. The first section is an introduction to our system including our scope and purpose. The second section is the system overview illustrating our system workflow. The third section includes the architecture design of the system, activity diagram, sequence diagram, state diagram and class diagram. The fourth section illustrates the database design in details. The fifth section illustrates our component design including the used algorithms and techniques. The sixth section illustrates the human interface design and describes how the user will interact with our system. The seventh section is the requirement matrix that shows which components satisfy each of the functional requirements. The rest of the sections are appendices and references.

1.4 Definitions and Acronyms

Term	Definition
LED	Light-Emitting Diode
RGB	Red-Green-Blue color Model
HSV	Hue-Saturation-Value
HOG	Histogram of oriented gradients
MVC	Model-View-Controller
OC-SVM	One-Class Support Vector Machine
DHT11	Sensor used for measuring temperature and humidity
LDR	Sensor used for measuring the light intensity

2 System Overview

In Smart Planting system, in order to monitor the way of growth and the needs of any plant they should be monitored by a video camera and sensors (DHT11 for measuring temperature and humidity, soil moisture for measuring the water content in soil and LDR for measuring the light intensity), they are placed inside the green house.
In the data input stage, the system extracts 7 frames of the greenhouse every 5 minutes, then these frames and sensors reading will be saved in the database.

Moving to the processing stage, by applying masking Hue-Saturation-Value (HSV) is used to detect the desired color range of the plant and the desired color range of the fruit/vegetable. Meanwhile, extracting features from the frames such as the size, shape and color of the fruit/vegetable, in order to detect the stage of the plant and if there is any diseases effected the plant then the landowner will be notified. The results will help us classifying the plant stage to generate the suitable LED light needed to be turned on.
These stages are:

1. Seeding stage, which seeds are being added to the ground but still no green leave being produced.
2. Growing stage, which some green leaves are produced.
3. Flowering stage, which the plant starts to blossom.
4. Harvesting stage and that where the plants' fruit is ready to be collected.

Ending with notifying the user if there is any diseases appeared on the plants or saving the outputs in our database, if there is no diseases being detected.
Finally in the final output stage, the system starts doing it's job, in providing the plant it's suitable environment to grow.
According to the readings from the sensors, if the temperature is higher than the threshold value then the fans start to work automatically. otherwise the fans will be turned off.
The output coming from the classification, as mentioned above, it helped in detecting the plant stage, if the plant is still in the Seeding or Growing stages then the Blue and Green LED lights are turned on, and if the current stage is the flowering stage then the Red LED light will be turned on, moving to the last plant stage which is the harvesting stage, the system goes to notify the user that the crops are ready to be harvested.

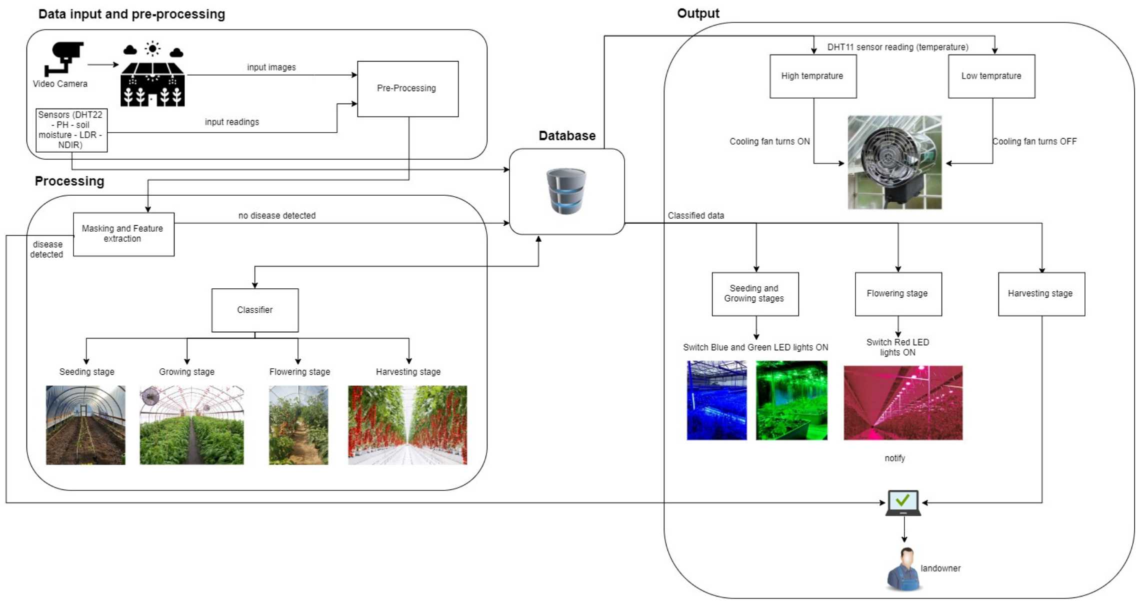


Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

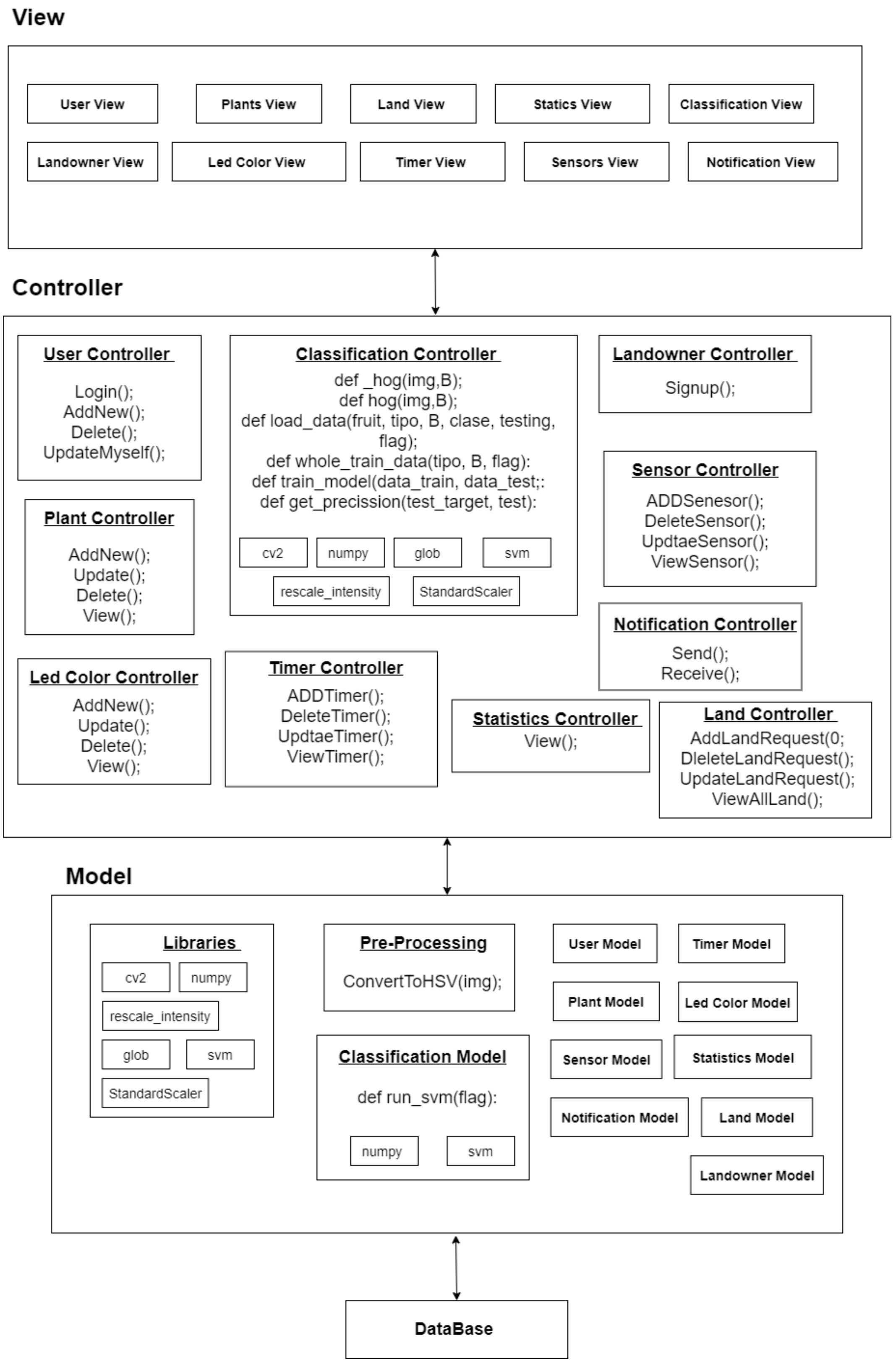


Figure 2: Architectural Design

3.1.1 View

It is responsible for presenting the data for the user in a User Interface(UI), to make it easy for the user to interact with our system. We have two different interfaces with different actions, one for the admin and another one for the landowner, in order to different between both of the users, first they have to pass by User View, which is a login page through which both users log into our system, opening to a update info page which allows the user (both admin and landowner) to see all their data and update any if needed. First, we start by the main Admin pages. As an admin have got more views pages for hem/her than the landowner, as the admin could go through the plant view, which gives the ability to add, delete and view all available plants in our system. As well as led color view, the admin could add, delete and view all available led colors. Time view, as this view page gives the ability for the admin to add, delete, view and update time intervals in hours, minutes and seconds, these time intervals are adjusted for the led strips on every king of plant being added in to the system. Land view and this view have to different jobs, first the admins, it allows them to view all lands registered in the system, view there requests, as for these requests it could be accepted or rejected according to the kind of the request, second the landowners,they could only view their land and make some requests on them only, as well as they could make a new request about adding a new land to the system. Statics view, this is a common interface view between the admin and the landowner, both view statics about the their plant growth weekly, while the admin view for the whole lands in the system while the landowner for his land only. Sensor view, is for allowing the admin to add, delete and view all king of sensors being used inside the greenhouses registered inside the system.

3.1.2 Controller

It is responsible for connecting both the View and Model together. All user interactions and requests made in the view are sent to the database in order to be fetched, this is done by the usage of the models. If these requests require a response it will be forward to the user through the view. Some controllers like the User Controller is responsible for only the user actions such as: login, adding a new user, deleting a user and updating him/her self. Plants Controller, Timer Controller, Led Color Controller, Sensor Controller and Land Controller all of the previous Controller are responsible for actions related to them as adding, deleting, viewing and updating.

3.1.3 Model

prepossessing:
A video camera is adjusted inside the greenhouse, which extracted image frames from it, these frames requires some preprocessing to be done to it, which are converting the image to an HSV image. This is done by the usage a library called cv2, this library is used in image processing, video capturing and analysis that includes feature as feature extraction, which helped in getting the perfect data for the system by which helped to move to the next phase in the system.
Libraries:
numpy: It's a core library which stands for Numerical Python that used as an efficient multi-dimensional container for generic data.
rescale-intensity: It's a library that is used to change the intensity range of an image according to the desired range.
glob:It's a library that is used to define techniques to match a specific pattern.
StandardScaler: it's a library that will transfer the data to have a mean equal 0 and the standard deviation equal 1 Classification Model:
This model takes image frames from the database, after being preprocessed, it's main role is to classify the stage of the plant whether it's seeding, flowering or harvesting.
The rest of the models as the User, Timer, Plant, Led color,Statistics, Notification, Sensor and Landowner all of these models are responsible to talk to the database and get all the needed and data of there function to be viewed for the user correctly.

3.2 Decomposition Description

3.2.1 Class Diagram

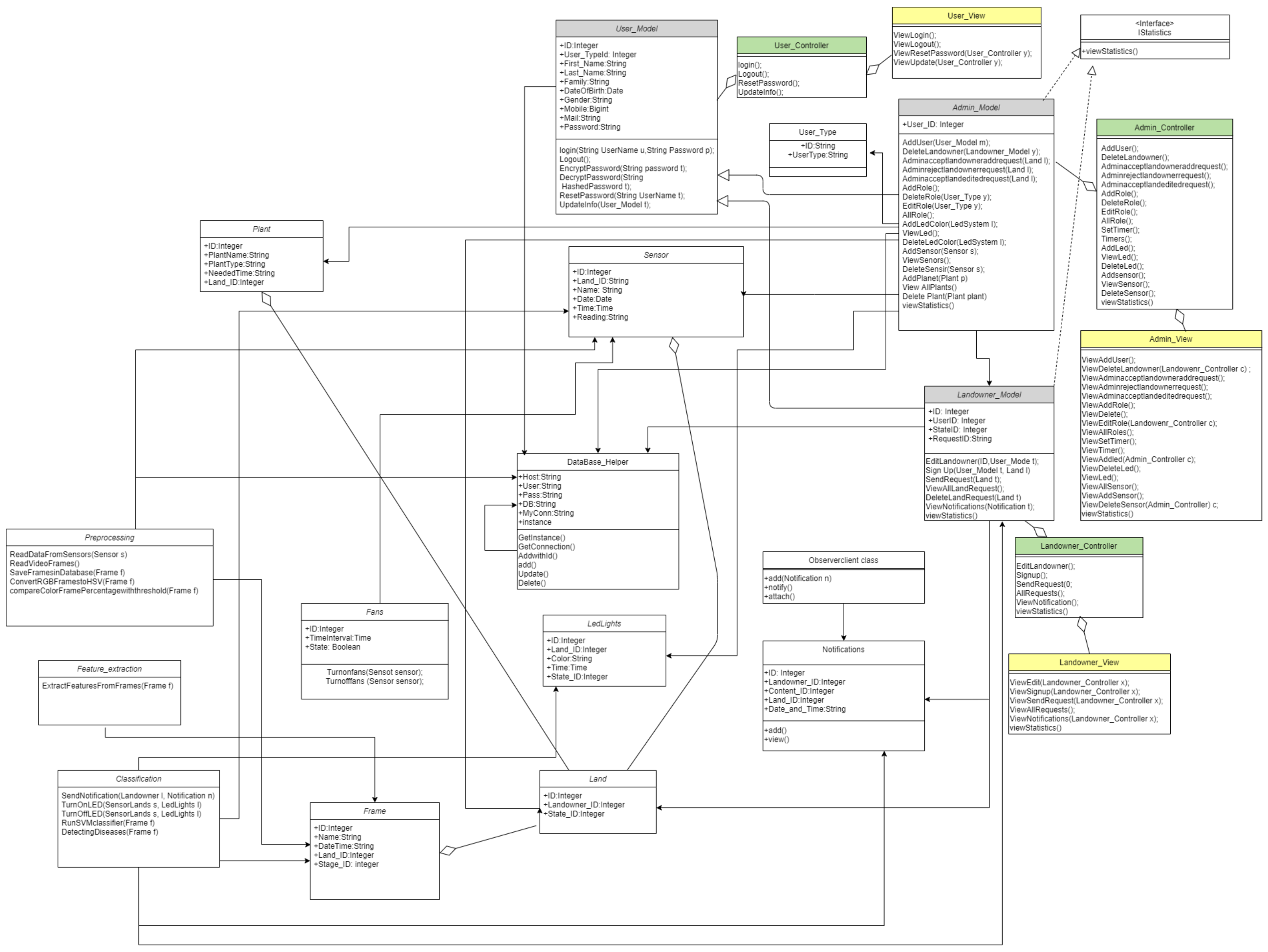


Figure 3: Class Diagram

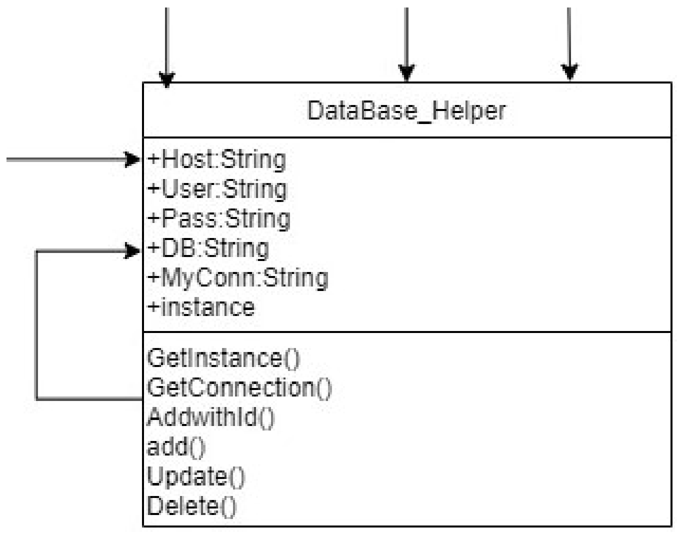


Figure 4: Singleton design pattern

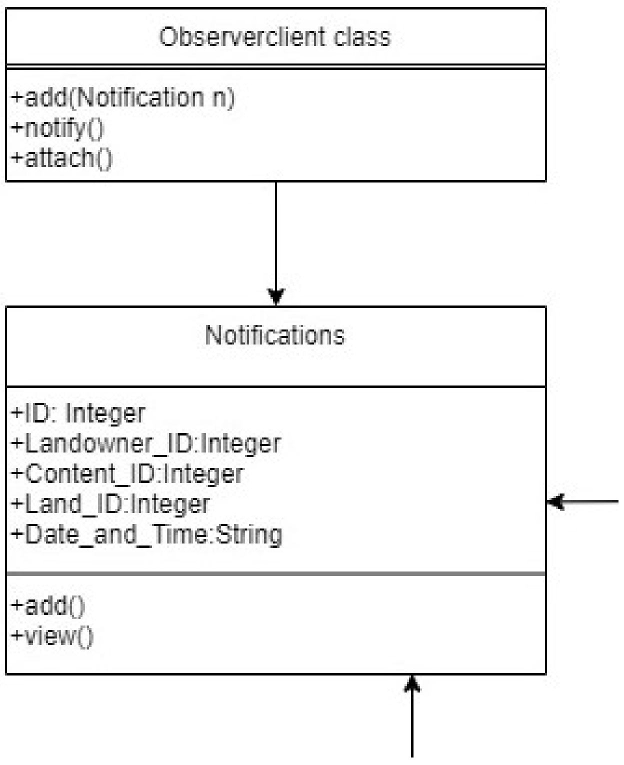


Figure 5: Observer design pattern

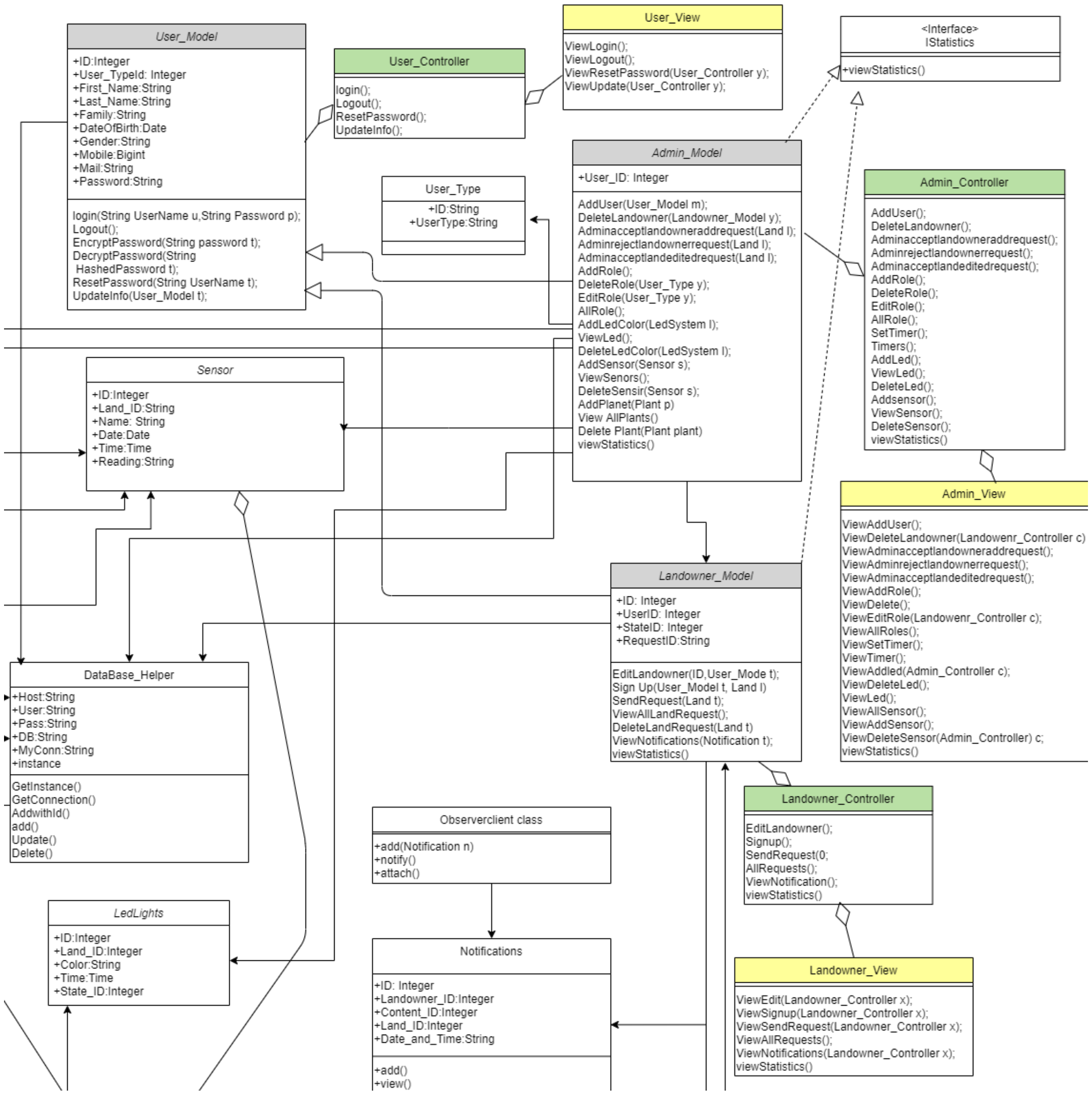


Figure 6: MVC design pattern

Class Name	User_Model
Super Class	None.
Sub Class	Admin_Model, Landowner_Model,Datebase_Helper classes
Purpose	Main class that is used to encapsulate different user types with their common attributes.
Collaborations	Admin_Model and Landowner_Model inherit from it. This class associated with database_Helper class. User_Controller aggregates from it.
Attributes	ID, UserTypeId,FirstName,LastName,FamilyName,DateOfBirth,Gender,Mobile,Mail, Password.

Operations	Login(String UserName u,String Password p)
	Logout()
	Encrypt password(String password t)
	Decrypt password(String Hashed_password t)
	Resetpassword(String UserEmail t)
	Updateinfo(User_Model t)

Class Name	Admin_Model
Super Class	User_Model
Sub Class	User_Type,Database_Helper,Sensor,Timer,Plant,Land
Purpose	This class is used to represent the admin.
Collaborations	This class inherits from User_Model. Admin_Controller class aggregates from it. This class associated with Database_Helper,User_Type, Sensor,Timer,Plant and Land classes.
Attributes	UserID.

Operations	AddUser(User_Model m)
	DeleteLandowner(Ladowner_Model y)
	Admin_accept_land_add_request(Land l)
	Admin_reject_land_add_request(Land l)
	Admin_accept_land_edited_request(Land l)
	AddRole()
	EditRole(User_Type y)
	DeleteRole(User_Type y)
	ViewAll()
	SetTimer(Timer t)
	ViewTimer();
	AddLedColor(LedSystem l);
	ViewLed ();
	DeleteLedColor(LedSystem l);
	AddSensor();
	ViewSenors();
	DeleteSensor(Sensor s);
	AddPlanet(Plant p)
	View AllPlants()
	Delete Plant(Plant plant)
	viewStatistics()

Class Name	Landowner_Model
Super Class	User_Model
Sub Class	Database_Helper,Notification,Land,Landowner_Controller
Purpose	This class is used to represent the landowner.
Collaborations	This class inherits from User_Model class. Landowner_Controller class aggregates from it. Database_Helper,Notification and land classes are associated with it.
Attributes	ID, StateID, UserID, RequestID.

Operations	EditLandowner(ID,User_Mode t);
	Sign Up(User_Model t, Land l)
	SendRequest(Land t);
	ViewAllLandRequest();
	DeleteLandRequest(Land t)
	ViewNotifications(Notification t);
	viewStatistics()

Class Name	UserType
Super Class	None.
Sub Class	Admin_Model
Purpose	This class is used to differentiate between user roles.
Collaborations	This class inherits from User_Model class and State class.
Attributes	ID, Role.
Operations	None.

Class Name	Land
Super Class	None.
Sub Class	Landowner_Model.
Purpose	This class is used to represent the land owned by which landowner in the system.
Collaborations	This class aggregates with sensor,Plant,Frame This class associate with Fans and Admin_Model
Attributes	ID, LandownerID,StateID.
Operations	None.

Class Name	Notification
Super Class	Content
Sub Class	Landowner_Model.
Purpose	This class is used to send notifications to landowners.
Collaborations	Land class is associated by Landowner_Model, Observation client class.
Attributes	ID,ContentID,LandID,LandownerID,DateTime.

Operations	Add()
	View()

Class Name	Plant
Super Class	None.
Sub Class	Land,Admin_Model.
Purpose	This class is used to represent the different type of plants.
Collaborations	Plant class associated with Admin_Model while it aggregates with Land class.
Attributes	ID,Name,PlantType,PlantNeededTimeInterval,Land_ID.
Operations	None.

Class Name	Timer
Super Class	None.
Sub Class	Plant.
Purpose	This class is used to set the timer to switch on/off the LED strips.
Collaborations	Timer class associated with Admin_Model class.
Attributes	ID,Duration.
Operations	None.

Class Name	Fans
Super Class	None.
Sub Class	Land,Sensors.
Purpose	This class is used to switch on/off fans.
Collaborations	Fans class associate with land and sensor Classes.
Attributes	ID, TimeIntervals,StateID.

Operations	TurnOnFans(LandID,Sensor)
	TurnOffFans(LandID,Sensor)

Class Name	LED Lights
Super Class	Admin_Model
Sub Class	None.
Purpose	This class is used to adjust the suitable led color to the land.
Collaborations	Admin_Model associate with this class.
Attributes	ID,Color,LandID,StateID,Time.
Operations	None.

Class Name	Sensors
Super Class	Admin_Model,Fans
Sub Class	Land
Purpose	This class is used to get the sensors readings from the database.
Collaborations	It aggregates with land class and associate with Admin_Model,Fans classes
Attributes	ID,Name,Land_ID,DataTime,Readings.
Operations	None.

Class Name	Frames
Super Class	None
Sub Class	Land.
Purpose	This class is used to take images from a real time camera inside lands and convert it into frames.
Collaborations	This class is aggregated with land class.
Attributes	ID,Name,TimeDate,LandID,Stage_ID.
Operations	None.

Interface name	IStatistics
Super Class	None
Sub Class	Admin_Model,Landowner_Model
Purpose	This interface is used to view statistics in different user views.
Collaborations	Admin_Model and Landowner_Model Implements from this interface
Attributes	None
Operations	ViewStatistics()

3.2.2 Activity Diagram

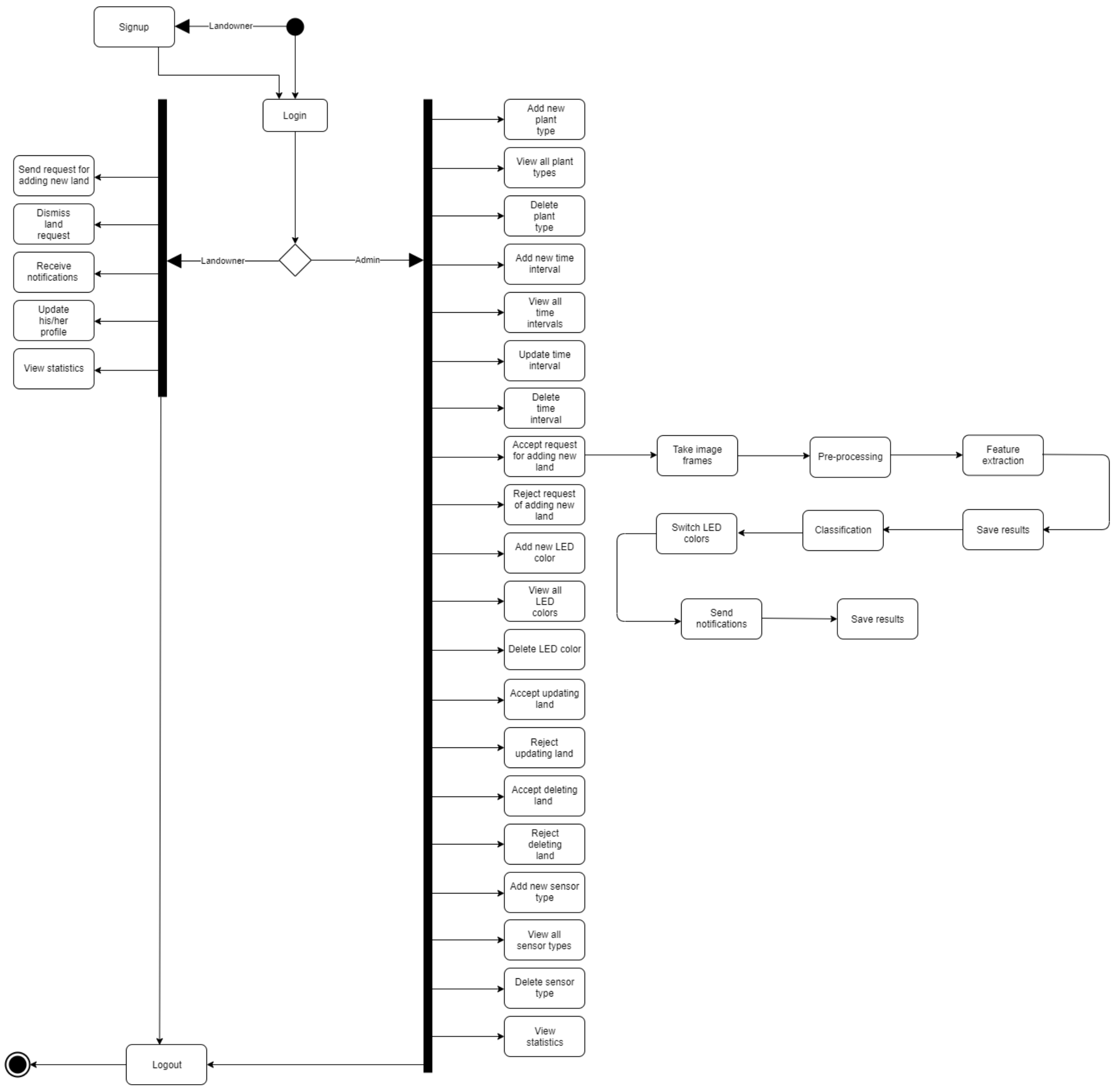


Figure 7: Activity Diagram

3.2.3 Sequence Diagram

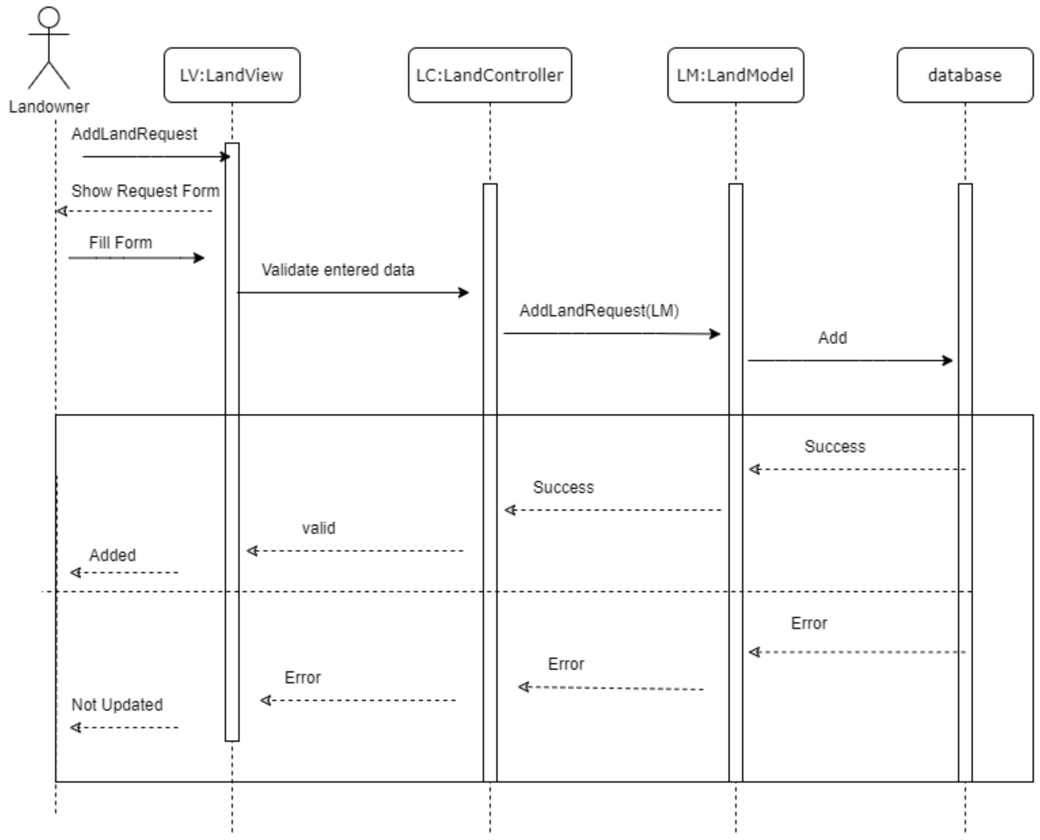


Figure 8: Landowner adds new request

The sequence diagram in Figure 8 views how a request done by the landowner to add a new greenhouse moves inside the system. First, the landowner fill the request form viewed in the landView page, the data taken from the form goes to the land Controller which is consider as a connection tube between the model and view, then to the land Model, to end up with saving the data in the database.

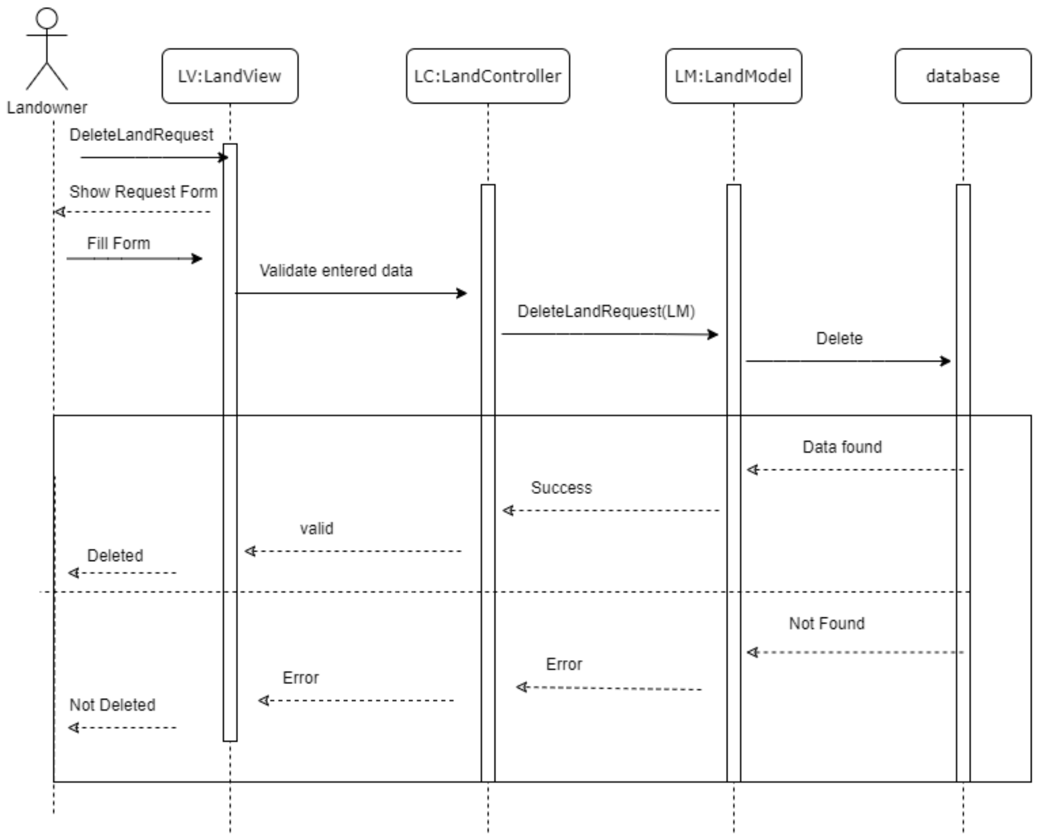


Figure 9: Landowner requests to delete the land

The sequence diagram in Figure 9 explains how the landowner can send request to delete his greenhouse from the system. First, the landowner sends a request to the admin to delete his greenhouse by filling the required form which is viewed in LandView page,validation is done on the data entered by the landowner, then this data will move to the LandController, then to LandModel ending to be saved the data in the database that his greenhouse is deleted successfully.

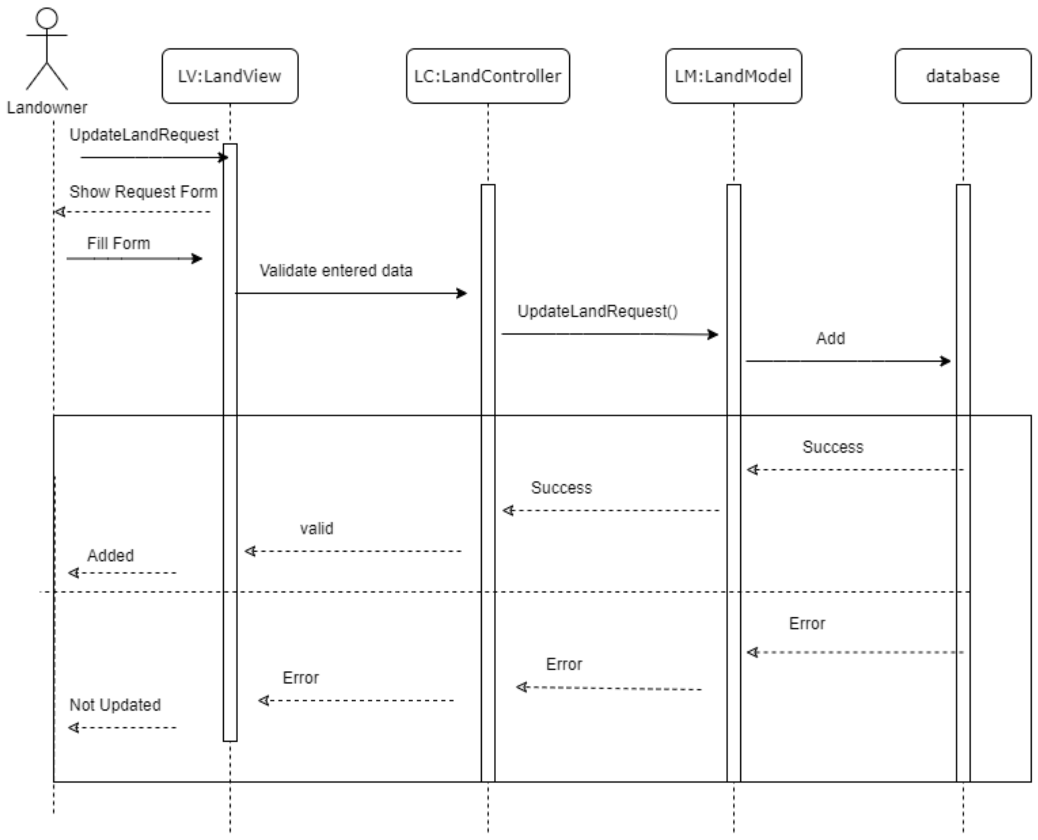


Figure 10: Landowner requests to update the land

The sequence diagram in Figure 10 explains how the landowner can send request to update his greenhouse's information in the system. First, the landowner sends a request to the admin to make some modifications to his greenhouse by filling the required form viewed in the landview page,then the system will make validation on the entered data, then the data taken from this form will move to the landController, then moves to LandModel, to end up with saving the new data in the database successfully.

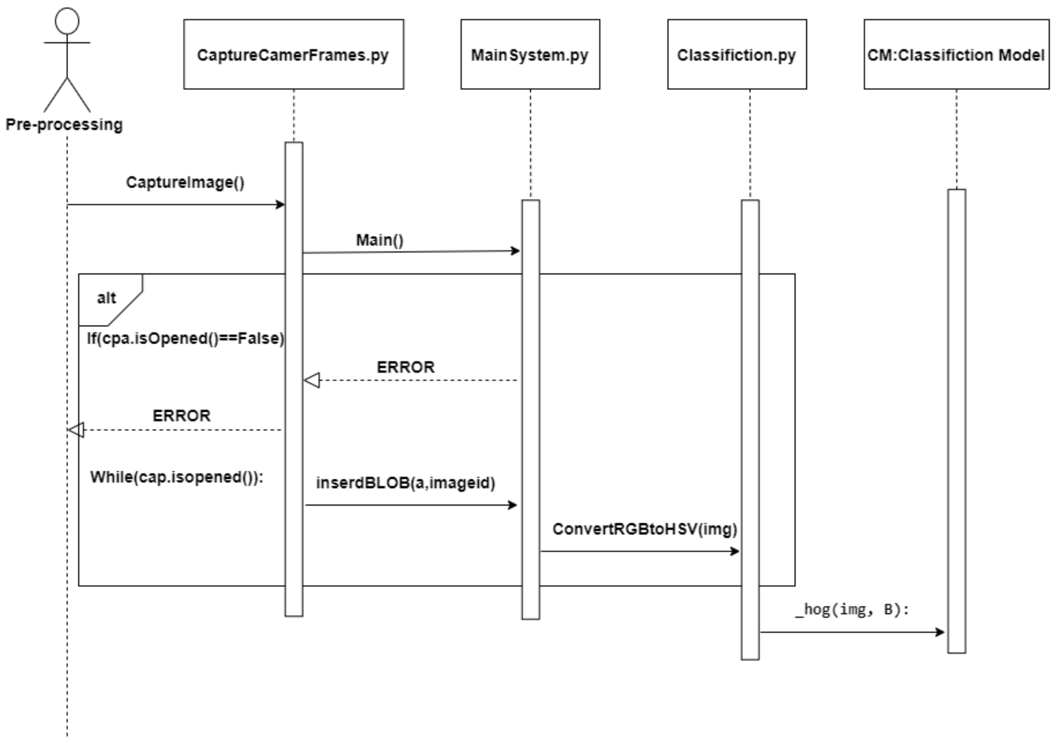


Figure 11: Pre-Processing

The sequence diagram in Figure 11 explains how preprocessing is done in the system, as first images are being captured through the CaptureCameraFrames python class, moving to the MainSystem python class through the main function, while moving the images passes through some conditions which are checking that the camera is opened, if it's opened then an error message is sent, else the images passes through MainSystem python class through inserBLOB(a,imageid) that takes the image id saved in database. Moving to the Classification python class through convertRGBtoHSV(img) and this function do convert the image from RGB to HSV, ending the Classification Model through the -hog(img,B)

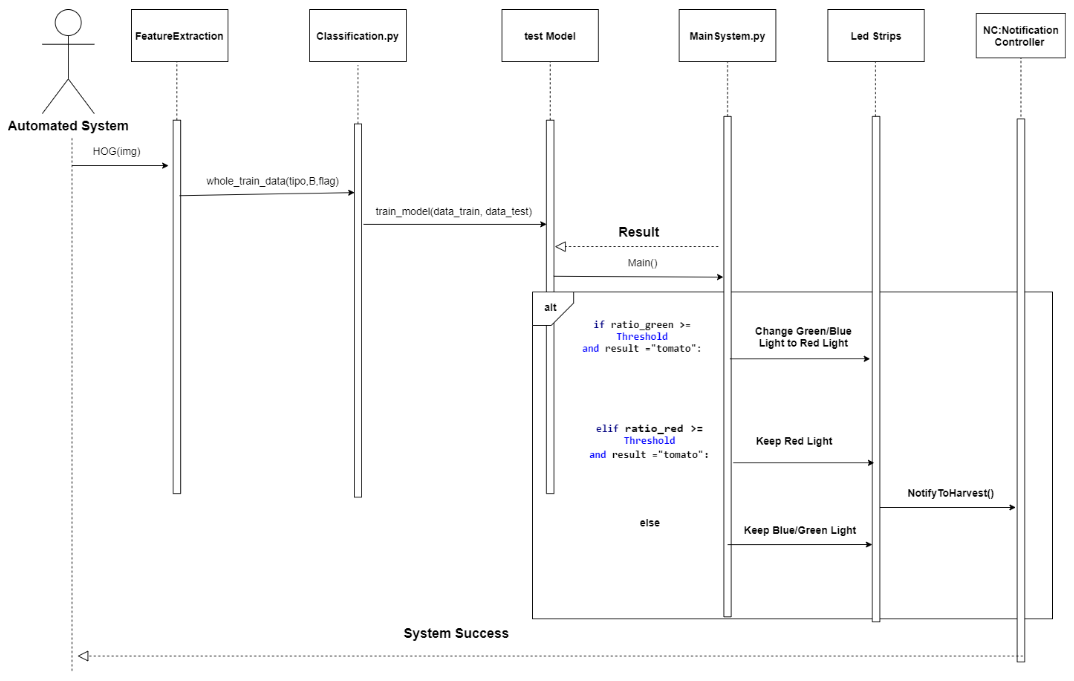

Automated System
FeatureExtraction
Classification.py
test Model
MainSystem.py
Led Strips
NC:Notification Controller
HOG(img)
whole_train_data(tipo,B,flag)
train_model(data_train, data_test)
Result
Main()
alt
if ratio_green >= Threshold and result ="tomato":
Change Green/Blue Light to Red Light
elif ratio_red >= Threshold and result ="tomato":
Keep Red Light
NotifyToHarvest()
else
Keep Blue/Green Light
System Success

Figure 12: Automated and Classifier

The sequence diagram in Figure 12 explains how our system is fully automated syetm, as first we start by feature extraction this is done through the function -hOG(img), moving to the Classifiaction python class through the function whole-tain-data(ripo,B,flag), then the test Model through the tarin-model(data-tarin,data-test), while moving to the Main System Python class a Result is being send back, this result is a string saying if there is tomatoes or not. Moving to led strips by some functions depending on some if conditions as showen in the above diagram. Ending with a system success statement.

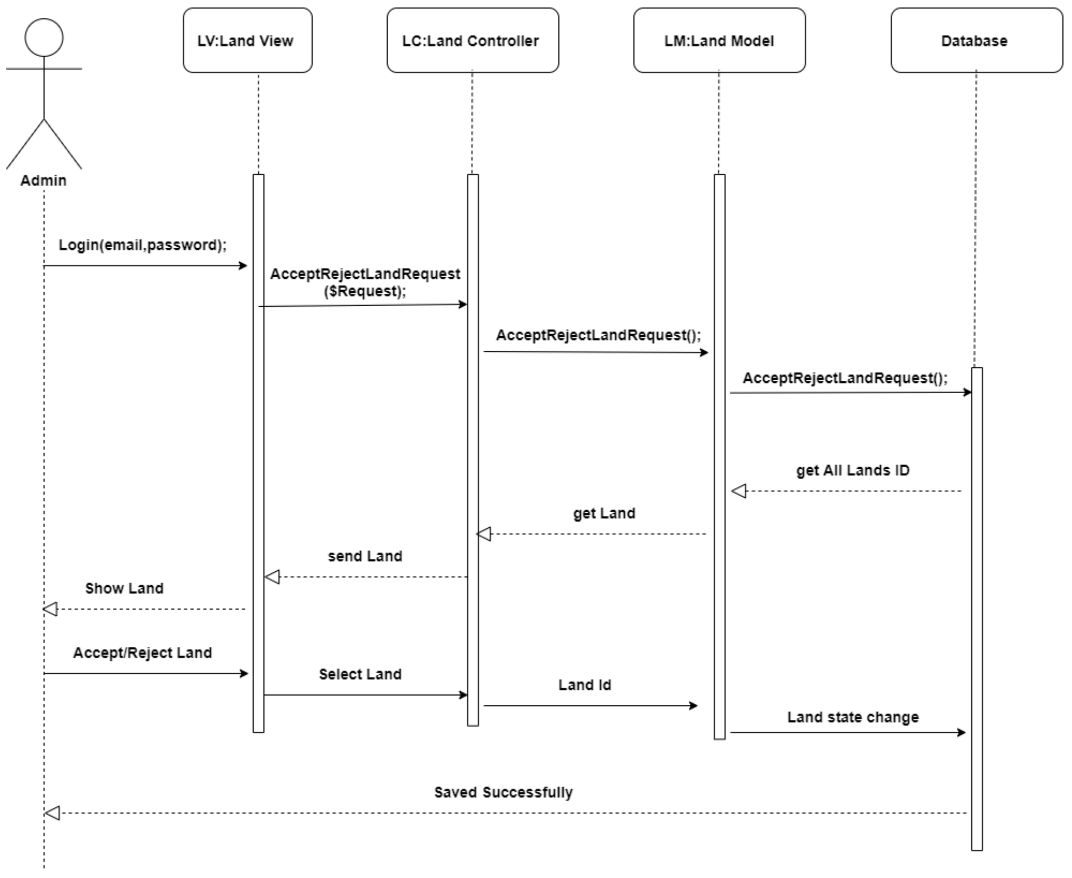


Figure 13: AcceptRejectLand

The sequence diagram in Figure 13 explains when the admin accepts or rejects a request coming from the Landowner. First the admin have to login with there mail and password, the admin chooses the view request link page to move to the Land controller, sending a request to the model to fetch all the requests from the database, and then getting all the land ids' that have requests only to be sent to the controller moving to the view, so the admin could accept or reject, when the action is done by the admin, this action is passed from the view to the controller then the model to be saved in the database.

3.3 Design Rationale

We have some design patterns to make our system maintainable. As we used Model-View-Controller (MVC) as helped us to make modifications easily, Single-Tone design pattern for reducing the overhead while connecting to the database and observer design patter for allowing different notification content for the users as not all of our users receives the same notification message. Our system is very accurate as it takes some actions at a specific time. So, it should be developed in an efficient and reliable way. We used some algorithms to make the accurate detection, feature extraction and classification of fruits/vegetables. Those algorithms are HSV, HOG and OC-SVM.

4 Data Design

4.1 Data Description

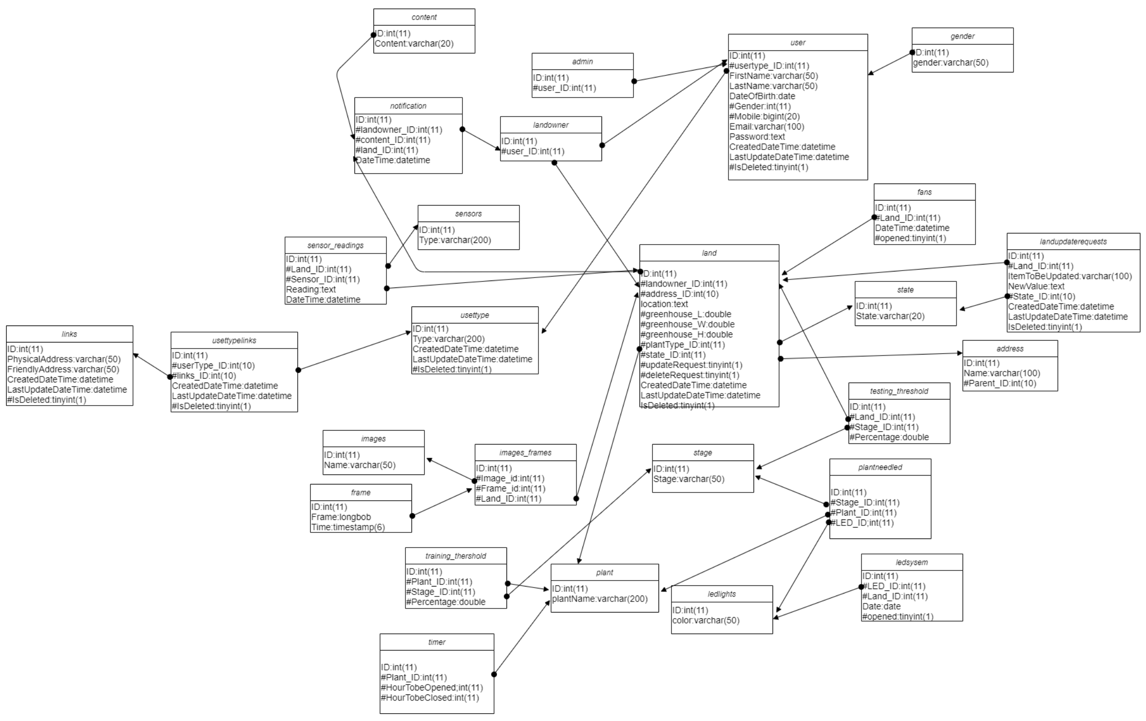


Figure 14: Database Schema

user: This table contains:id, userType-ID as we have two users in our system they are Admin and Landowner, FirstName, LastName, DateOfBirth, Gender, Mobile, Email, Password, CreateDateTime,LastUpdateDateTime andIsDeleted.

userType: This table contains: ID, Type, CreateDateTime,LastUpdateDateTime and IsDeleted. landowner:This table contains: ID and user-ID.

admin: This table contains: ID and user-ID.

land: This table contains: ID, landowenr-ID, address-ID, location,greenhouse-L, greenhouse-W, greenhouse-H and they represent the greenhouse size, plantType-ID as this system is trained to have different types of plants, state-ID, updateRequest, deleteRequest as a landowner they have the ability to make a request about the greenhouse if there is a new one to be added to the system or a new update of an already existing greenhouse in the system, CreateDateTime,LastUpdateDateTime and IsDeleted.

state: This table contains: ID and State.

content:This table contains:ID and content.

address:This table contains: ID, Name and Parent-ID.

frame: This table contains: id, Frame and Time.

ledsystem: This table contains: ID, LED-ID, land-ID,Date, opened.

plantneededled: This table contains: ID, Stage-ID and Plant-ID, LED-ID.

image-frames: This table contains: Id,Image-Id,Frame-Id and Land-ID.

landupdaterequests: This table contains: ID, Land-ID as every land have got it's own request, ItemToBeUpdated and that represent the item inside the greenhouse that will be updated, NewValue, State-ID, CreateDateTime,LastUpdateDateTime and IsDeleted.

timer: This table contains: ID, pant-ID, HourTobeOpened and HourTobeClosed.

fans: This table contains: ID, Land-ID, DateTime and opened.

ledlights: This table contains: ID and color of the led lights that will be used in he system.

usertypelinks: This table contains: ID, userType-ID,links-ID, CreateDateTime,LastUpdateDateTime and IsDeleted. This table is done to separate between the links that the Landowner and Admin will be able to view in the web applicationn.

stages: This table contains: ID and Stage.

plant: This table contains: ID and plantName.

links: This table contains: ID, PhysicalAddress, FriendlyAddress, CreateDateTime,LastUpdateDateTime and IsDeleted. This table is done to save the links that will be used in our web application. The PhysicalAddress is the real ink address while the FriendlyAddress is the address that the user will see, it's written in a way the user would understand.

gender: This table contains: ID and gender.

images: This table contains: Id and Name.

notification: This table contains: ID, landowner-ID, content-ID, land-id and DateTime.

4.2 Data Dictionary

Security is achieved through our web application, as the user which is the Landowner should first register in the system, so both of the Admin and Landowner have their own account and no one could access it, only if they have the password, so passwords are being encrypted and decrypted in the system.
Reliability is achieved through our greenhouse, as any greenhouse in our system is supplied by an Electric generators to insure that if the power goes off, generators will support with the needed electricity.
Maintainability is achieved through our web application as it is programmed using MVC to apply any changes by the developer easily, Single-Tone as to insure that there is only one connection to the database and that helps in avoiding any over head on the system, Observer so the user could be notified with any notification inside the system and design patterns.
Portability is achieved as our web application could be viewed on different platforms. Usability is achieved through our web application as it's easy for the user to learn and interact with it.

5 Component Design

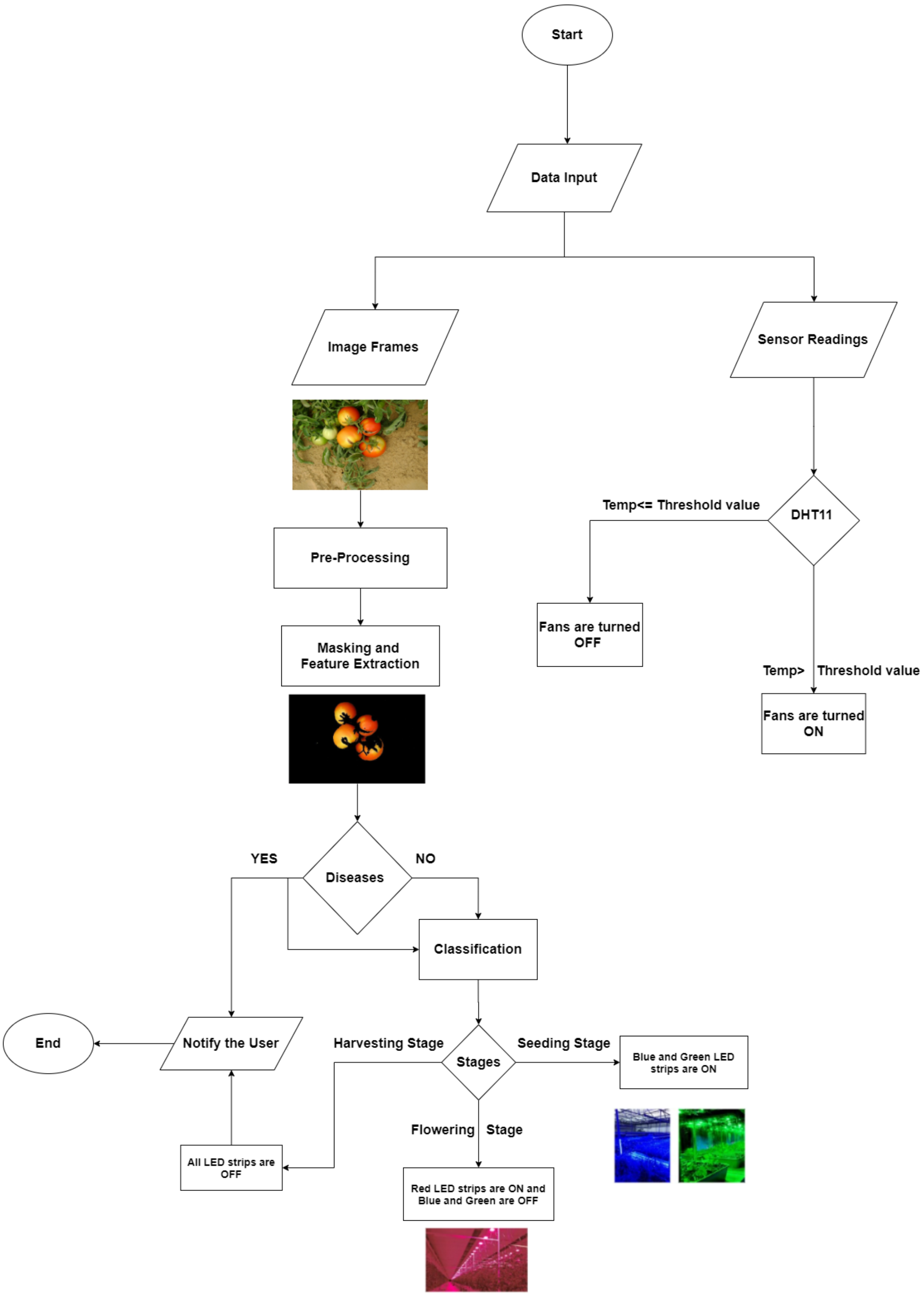


Figure 15: Flow chart of the system approach

5.1 Data Input

In this phase, there are two types of data inputs. First, the data coming from the sensors by the usage of an Arduino are passed and saved in the database. First we check on the temperature by the DH11 sensor, the readings of the sensor is compared with an adjusted threshold value. This Threshold value is changeable according to the plant type inside the greenhouse. Since our experiment is done on tomato so, the threshold value of the temperature will be 21–29.5°C during the day and 18.3–21.1°C during the night. According to the readings, fans will be turned ON/OFF to make a suitable temperature in the greenhouse. The second data input, is a collection of image frames coming from a real time camera settled in our greenhouse. Pre-processing and processing are operated on these frames as Enhancements, Masking and Feature extraction.

5.2 Pre-Processing

In this phase, Enhancements could be applied to image frames if needed, and that in order to remove any added noise in the frame to make it prepared for the processing stage in the system.

5.3 Processing

In this phase, both Masking and feature extraction are done on the input frames. First, we start by the masking, since all the input frames are RGB we start by converting them into HSV, as HSV separates image luminance from color information. Which makes it easier to deal with all image frames,as shown in figure 16.

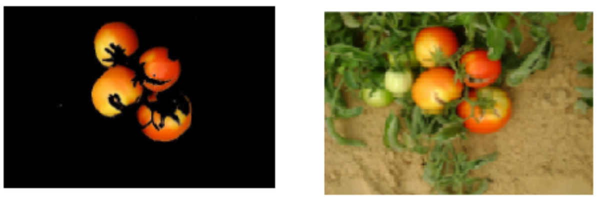


Figure 16: Before and after the HSV masking effect on tomatoes' testing image

Second, Feature extraction. Feature extraction is done by the usage of HOG.

$$\begin{aligned} gx &= ones((8,8)) * cos(pi/4) \\ gy &= ones((8,8)) * sin(pi/4) \\ h &= HOG8x8(gx, gy) \end{aligned} \tag{1}$$

First, the Image will be converted into grey value image. Then get the HOG value (h) in 8x8 cell using equation (1), where the gx is the x-directional derivative, gy the y-directional derivative.

While masking and feature extraction are in process, a disease could be detected, since our experiment is on tomato, so the most common diseases are the late Blight and the Early Blight, as shown in both figure: 5 and 6 . When any of these two diseases are detected the user will be notified, while the system is still in process.

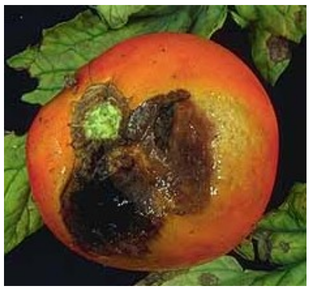


Figure 17: Tomato effected by Early Blight

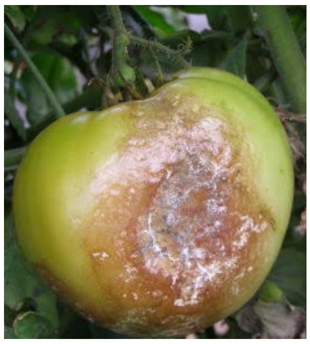


Figure 18: Tomato effected by Late Blight

5.4 Classification

In this phase, the model starts by detecting stages of the plants growth. We used One-class Support Vector Machine (OC-SVM) classifier as it shows great accuracy [1]. It is a statistical machine learning algorithm applied on data that has only one class, which is the "normal" class. OC-SVM basically separates all the data point from the origin, then maximizes the distance from this hyper plane to the origin. The function returns 1 if there is any fruits/vegetables appeared while it returns -1 elsewhere.

$$f(x) = sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) - p) \tag{2}$$

This method in equation (2) creates a hyper plane characterized by p which separates all the data points from the origin, αi is the Lagrange multipliers computed for each distance and *K(x,xi)* is the Kernel.

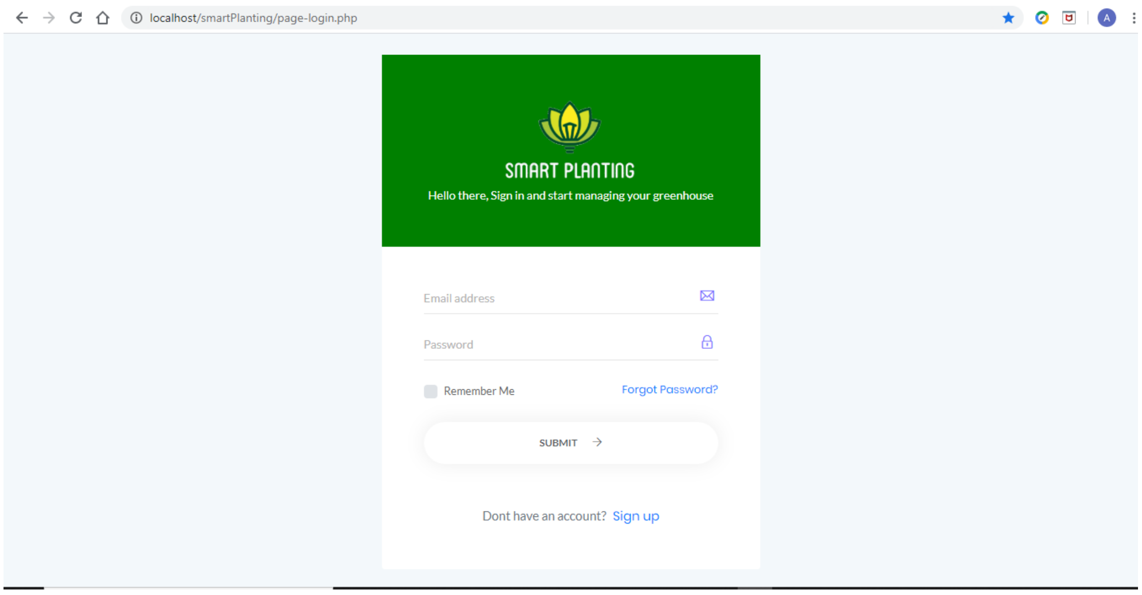
This phase classify the stages of the plant, whether it's still in the seeding stage, or the flowering stage, or it's ready to be harvested. When the plants reach the harvest stage, the system automatically notifies the user, and the system goes down.

6 Human Interface Design

6.1 Overview of User Interface

Our system Smart Planting user interface is easy to be used. You can login whether you're a Landowner or an admin. The system leads you to different screens depends on your role. Admins will be able to CRUD on most of the system such as, accepting/rejecting landowners' requests to add new greenhouses. While landowners are responsible for dealing with their greenhouses. In fact, representation for the whole system will be shown in the upcoming sections (6.2, 6.3).

6.2 Screen Images


localhost/smartPlanting/page-login.php

SMART PLANTING

Hello there, Sign in and start managing your greenhouse

Email address

Password

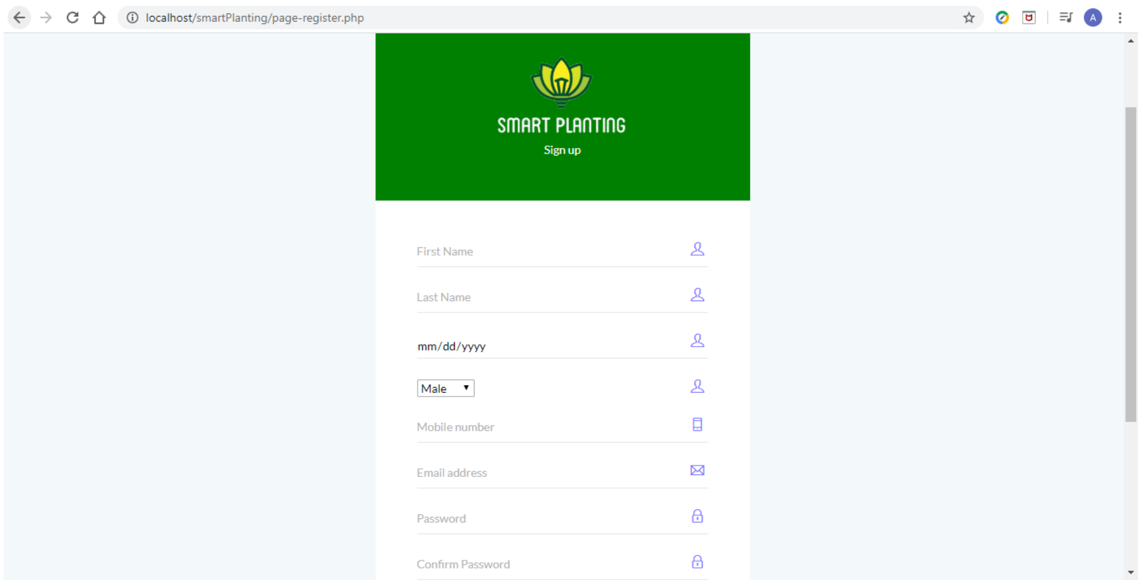
Remember Me

Forgot Password?

SUBMIT

Dont have an account? Sign up

Figure 19: Login-Page


localhost/smartPlanting/page-register.php

SMART PLANTING

Sign up

First Name

Last Name

mm/dd/yyyy

Male

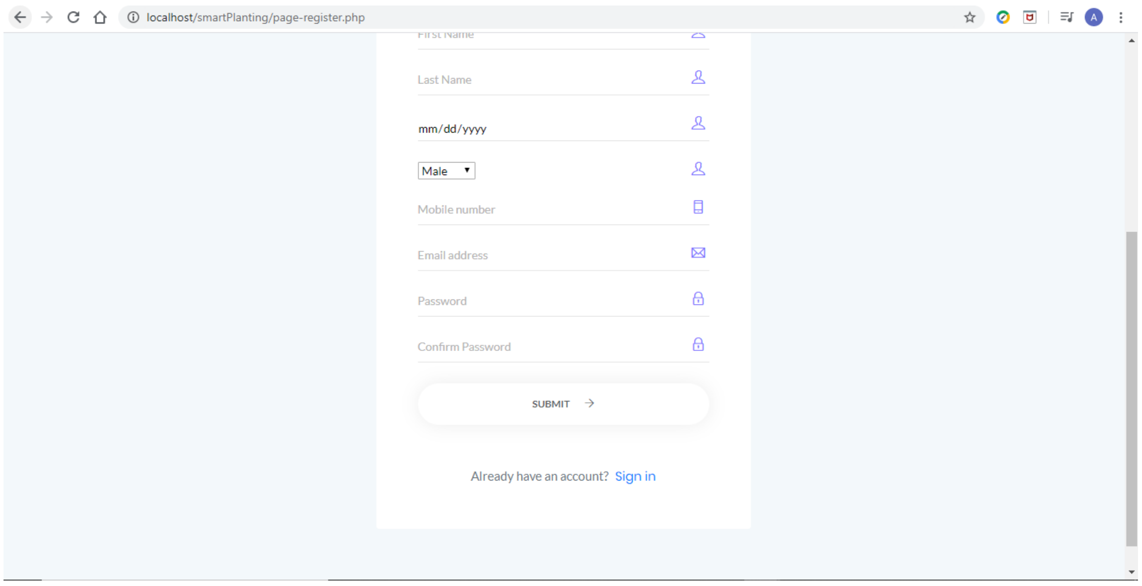
Mobile number

Email address

Password

Confirm Password

Figure 20: SignUp page one


localhost/smartPlanting/page-register.php

First Name

Last Name

mm/dd/yyyy

Male

Mobile number

Email address

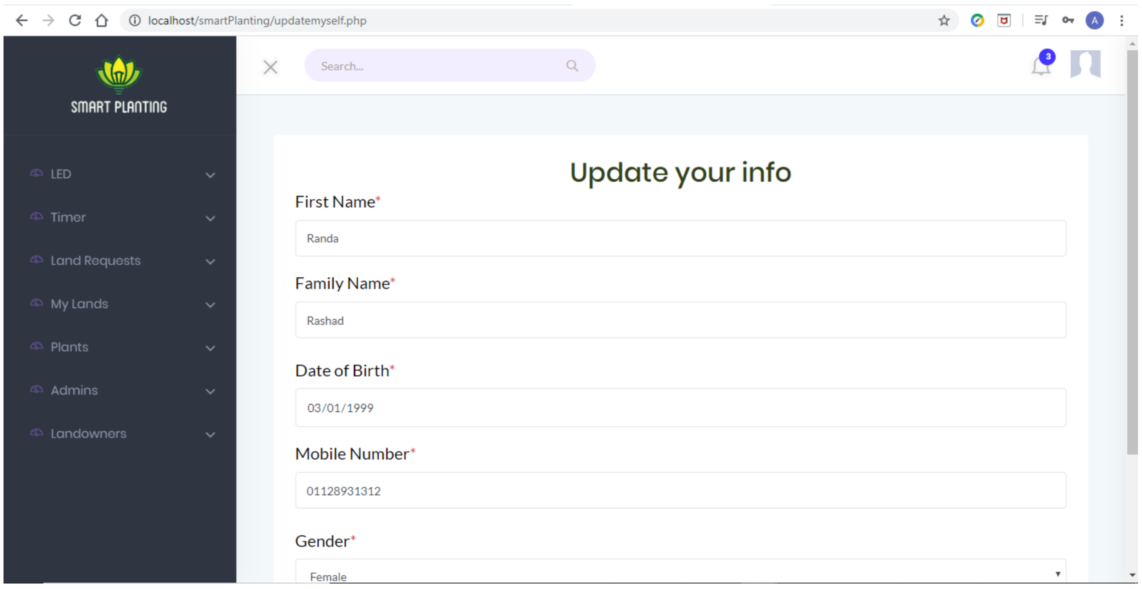
Password

Confirm Password

SUBMIT

Already have an account? Sign in

Figure 21: SignUp page two


localhost/smartPlanting/updatemyself.php

Update your info

First Name*

Randa

Family Name*

Rashad

Date of Birth*

03/01/1999

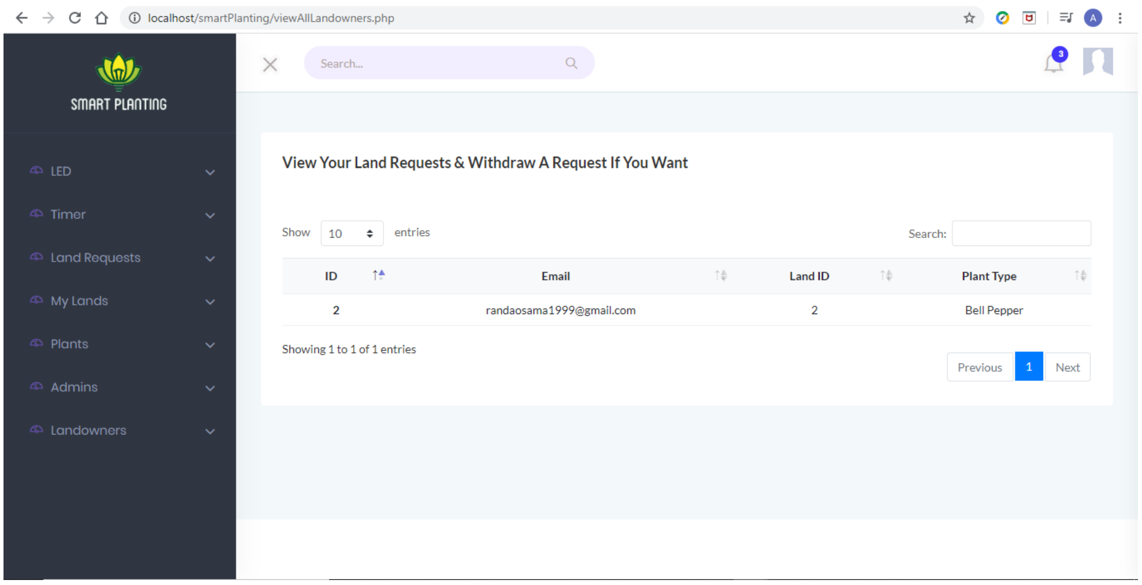
Mobile Number*

01128931312

Gender*

Female

Figure 22: Admin Updating his/her INFO



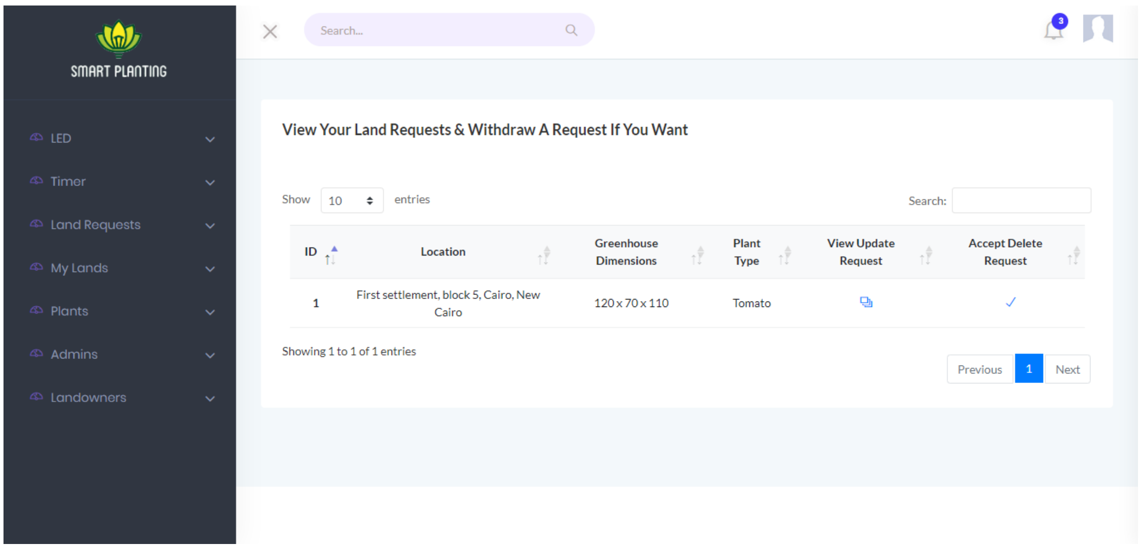
View Your Land Requests & Withdraw A Request If You Want

Show 10 entries Search:

ID	Email	Land ID	Plant Type
2	randaosama1999@gmail.com	2	Bell Pepper

Showing 1 to 1 of 1 entries

Figure 23: Admin Viewing All Landowners



View Your Land Requests & Withdraw A Request If You Want

Show 10 entries Search:

ID	Location	Greenhouse Dimensions	Plant Type	View Update Request	Accept Delete Request
1	First settlement, block 5, Cairo, New Cairo	120 x 70 x 110	Tomato		✓

Showing 1 to 1 of 1 entries

Figure 24: Admin Viewing All Requests

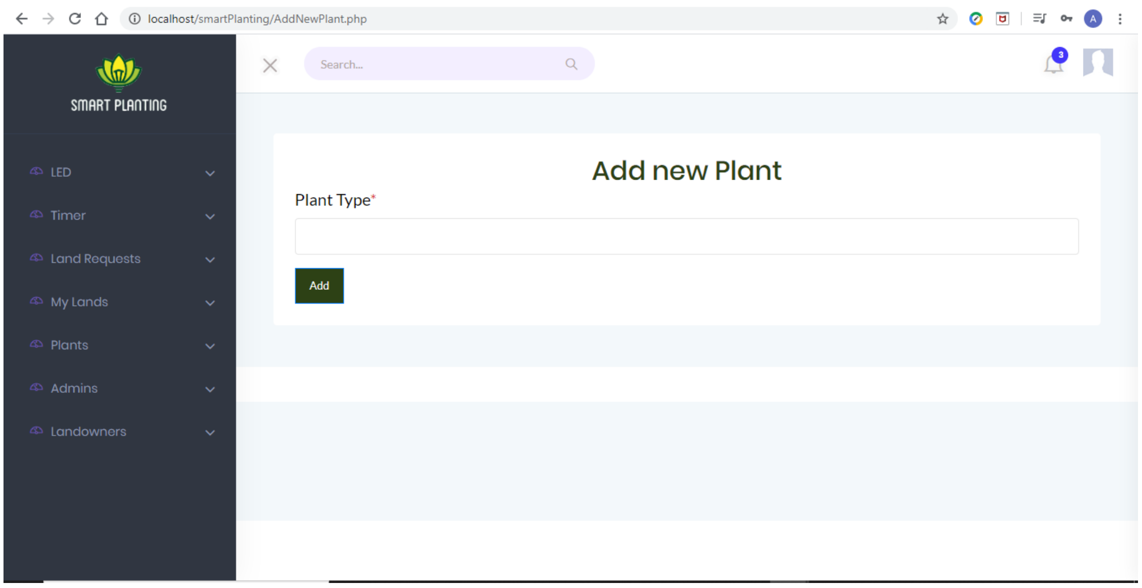


Figure 25: Admin Adding New Plants

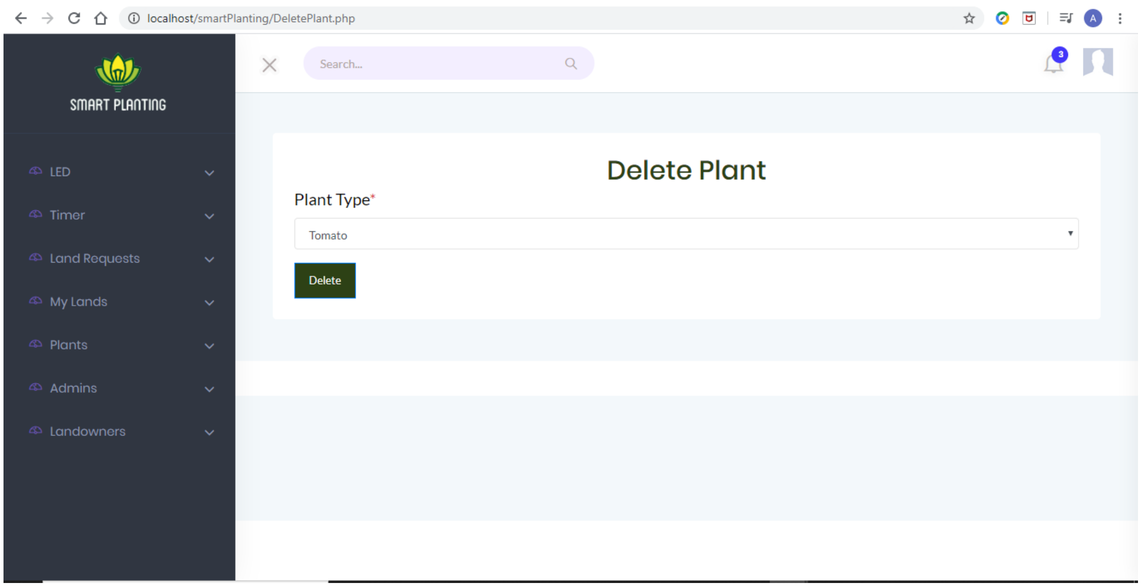
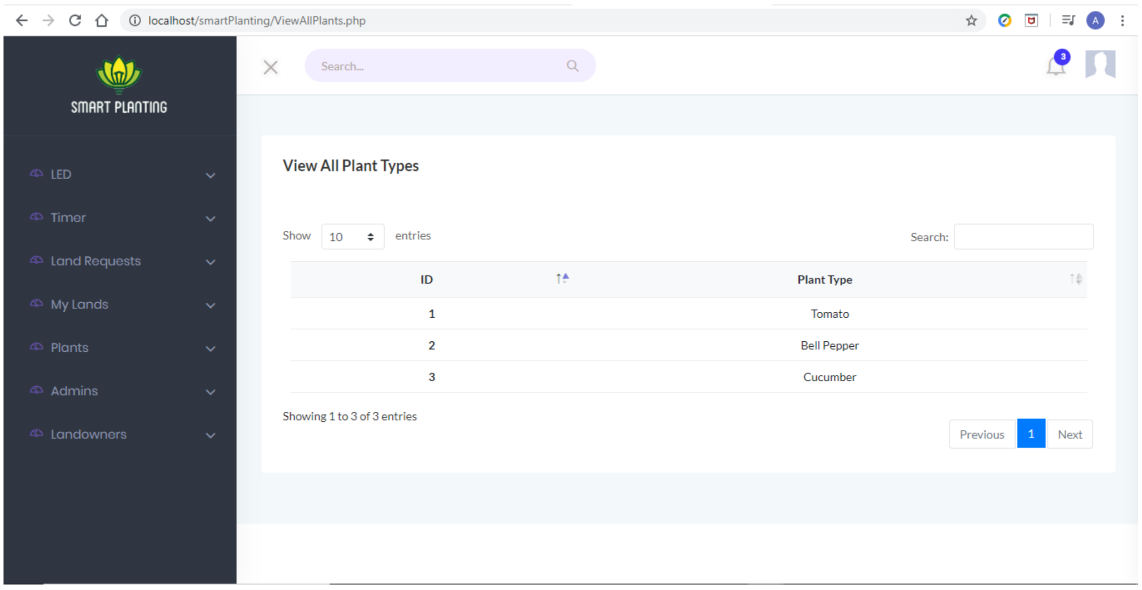


Figure 26: Admin Deleting Plant


localhost/smartPlanting/ViewAllPlants.php

View All Plant Types

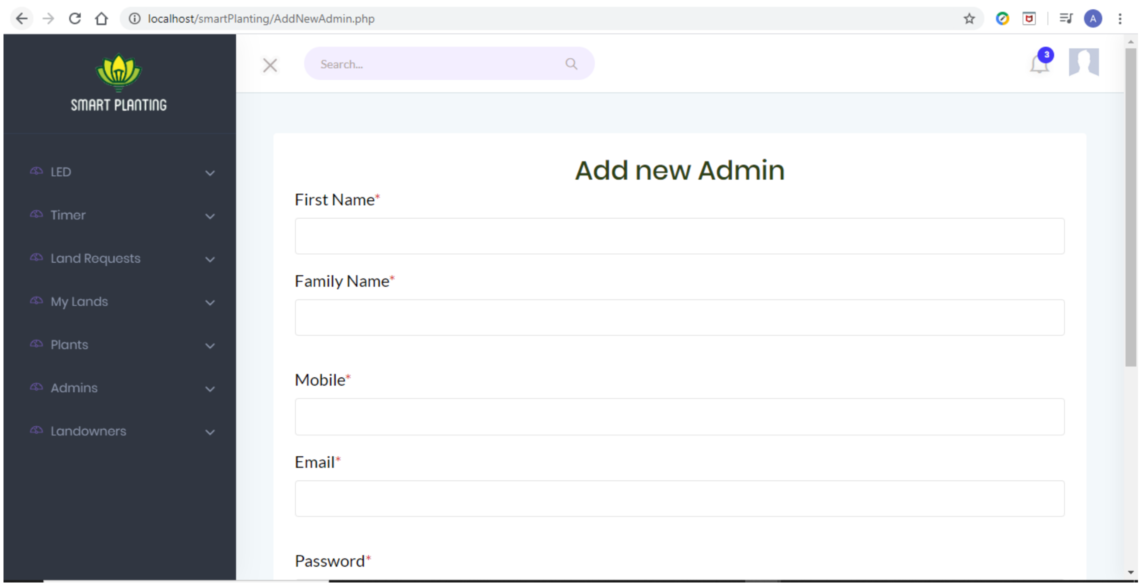
Show 10 entries

Search:

ID	Plant Type
1	Tomato
2	Bell Pepper
3	Cucumber

Showing 1 to 3 of 3 entries

Figure 27: Admin View All Plant


localhost/smartPlanting/AddNewAdmin.php

Add new Admin

First Name*

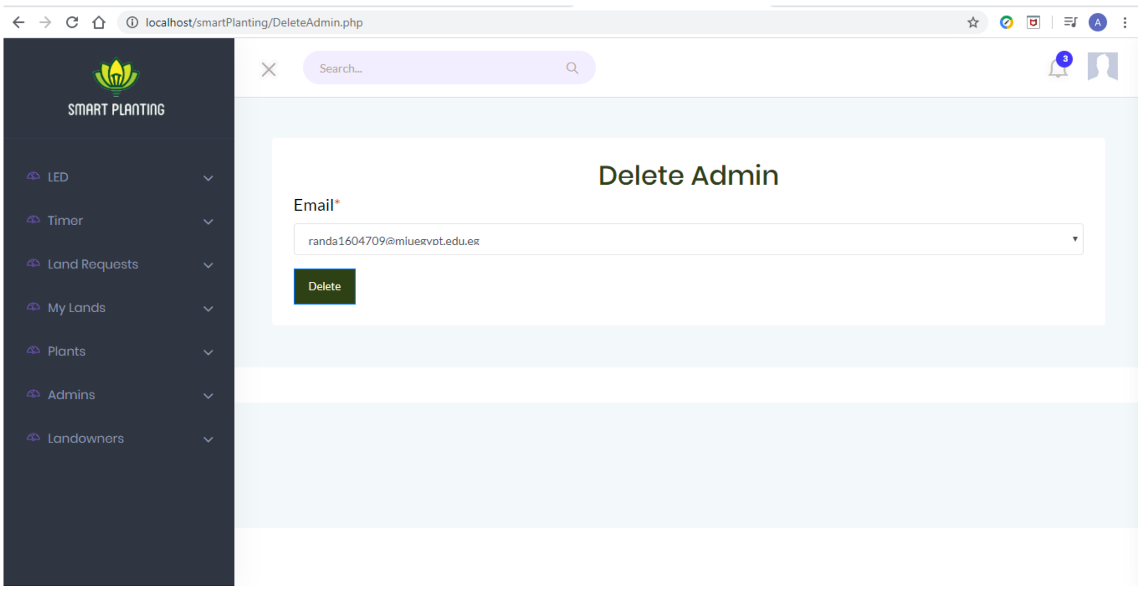
Family Name*

Mobile*

Email*

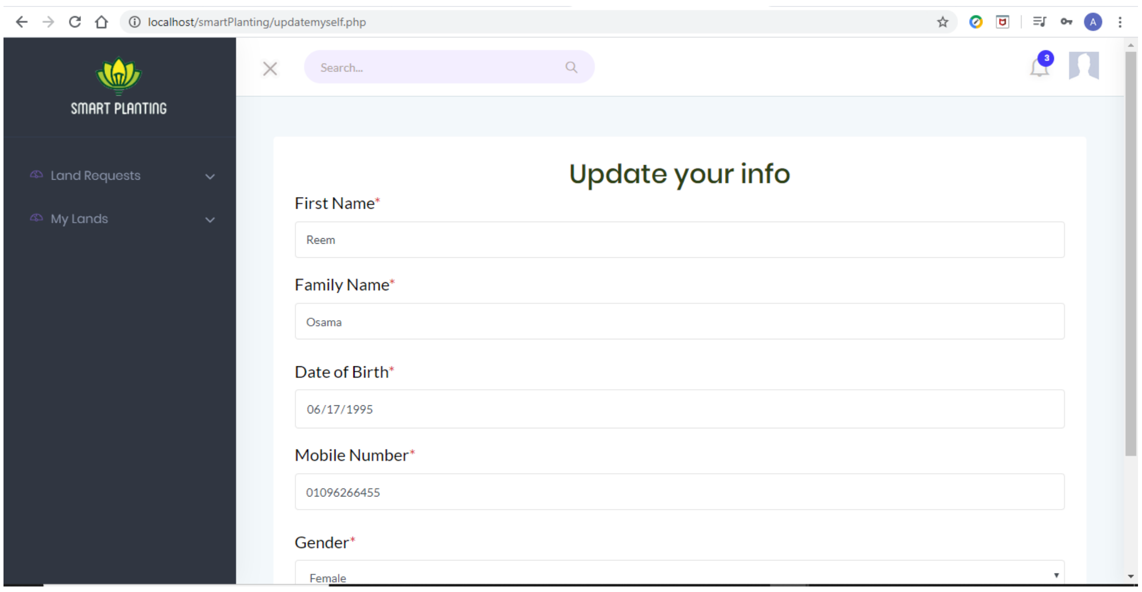
Password*

Figure 28: Admin adding a new admin



randa1604709@miuegypt.edu.eg

Figure 29: Admin deleting an old Admin



First Name* Reem

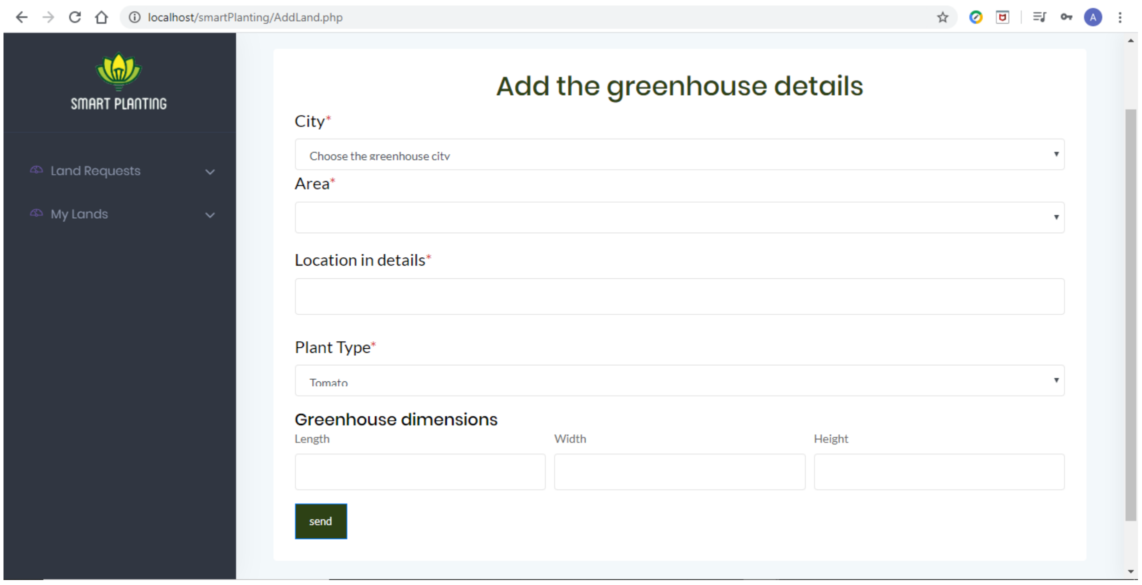
Family Name* Osama

Date of Birth* 06/17/1995

Mobile Number* 01096266455

Gender* Female

Figure 30: Landowner Updating her/his INFO


localhost/smartPlanting/AddLand.php

SMART PLANTING

Land Requests

My Lands

Add the greenhouse details

City*

Choose the greenhouse city

Area*

Location in details*

Plant Type*

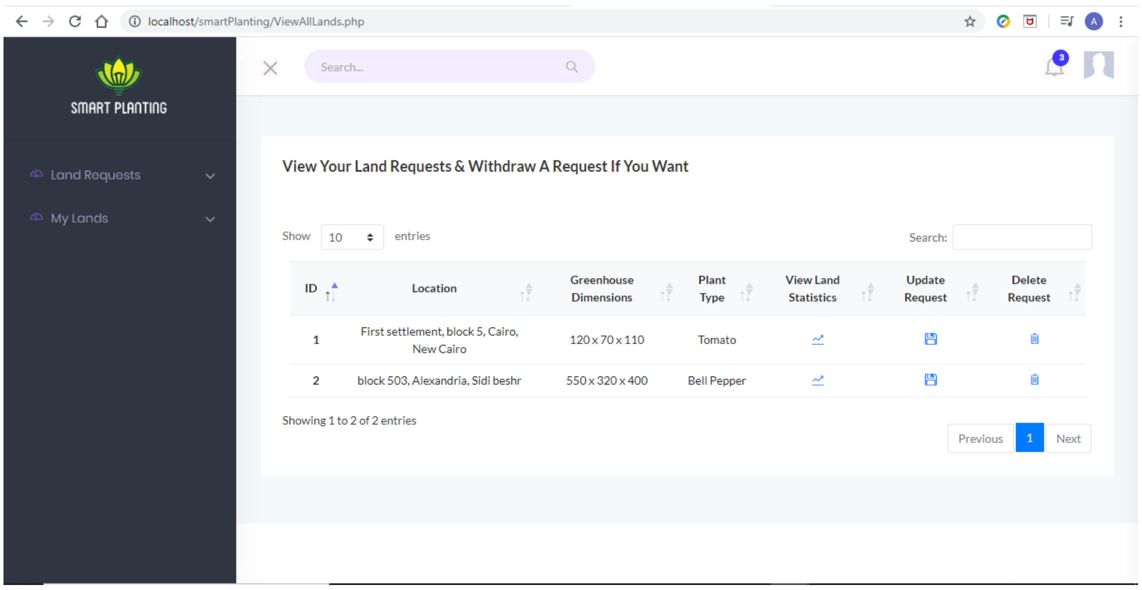
Tomato

Greenhouse dimensions

Length | Width | Height

send

Figure 31: Landowner making a new request with a new land


localhost/smartPlanting/ViewAllLands.php

SMART PLANTING

Land Requests

My Lands

Search...

View Your Land Requests & Withdraw A Request If You Want

Show 10 entries

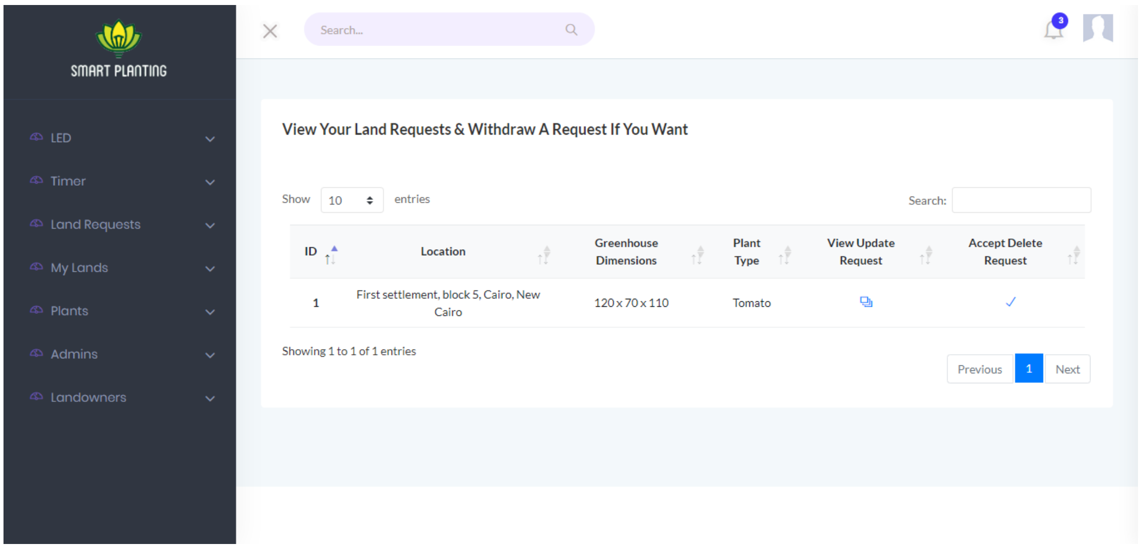
Search:

ID	Location	Greenhouse Dimensions	Plant Type	View Land Statistics	Update Request	Delete Request
1	First settlement, block 5, Cairo, New Cairo	120 x 70 x 110	Tomato			
2	block 503, Alexandria, Sidi beshr	550 x 320 x 400	Bell Pepper			

Showing 1 to 2 of 2 entries

Previous 1 Next

Figure 32: Landowner viewing all his/her lands in the system



View Your Land Requests & Withdraw A Request If You Want

Show 10 entries Search:

ID	Location	Greenhouse Dimensions	Plant Type	View Update Request	Accept Delete Request
1	First settlement, block 5, Cairo, New Cairo	120 x 70 x 110	Tomato		✓

Showing 1 to 1 of 1 entries

Previous 1 Next

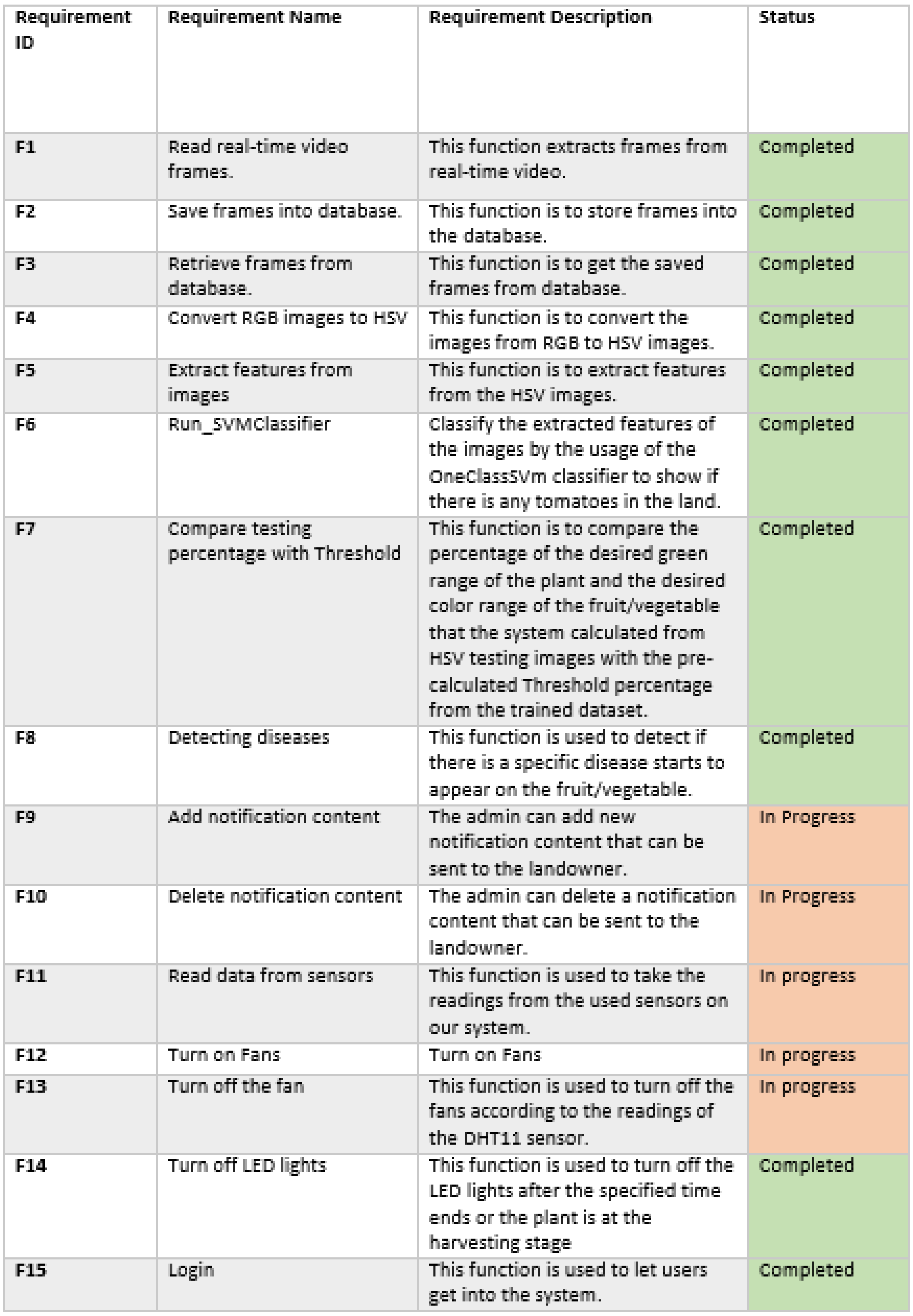
Figure 33: Admin Accept/Reject Landowner's request for adding new greenhouse

6.3 Screen Objects and Actions

Fig. 17 shows the login page, it has two boxes. The first box is for entering the email address and the second box is for the password. If you logged in as an admin you will access the screens in Fig. 20, Fig. 21, Fig. 22, Fig. 23,Fig. 24,Fig. 25,Fig. 26, Fig. 27, Fig. 31 and if you logged in as a landowner you will access the screens in Fig. 28, Fig. 29, Fig. 30.

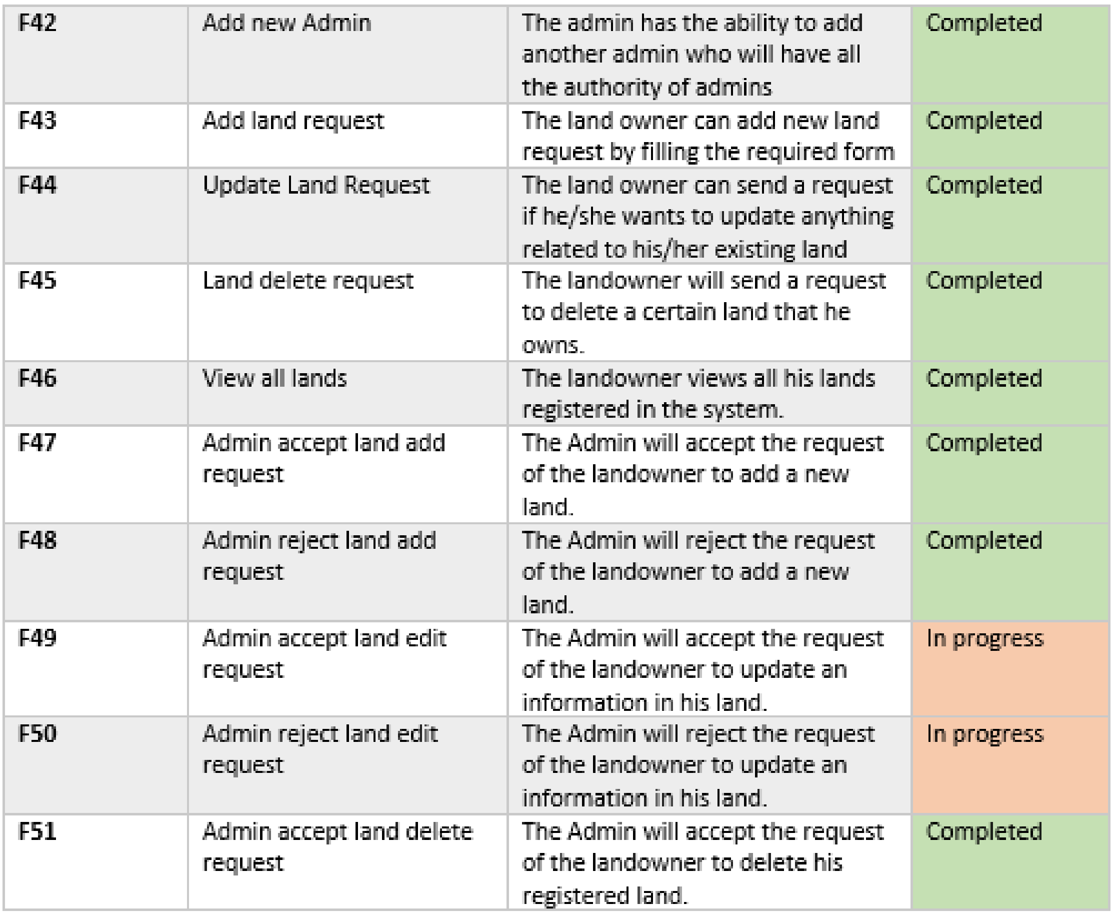
Fig. 18, Fig. 19 shows sign-up page to let landowners register to the system. Fig. 20 shows how the admin will be able to edit his/her personal information such as (Username, password, mobile number, ..). Fig. 21 shows that the admin can view all landowners' information. Fig. 22 shows that the admin can view all requests that the landowner sends to add new greenhouse. Fig. 23 shows that the admin can add new plant type by typing it's name. Fig. 24 shows that the admin will be able to delete plant type by searching on it's name. Fig. 25 shows that the admin will be able to view all the plant types and it's id. Fig. 26 shows that the admin can add new admins. Fig. 27 shows that the admin can delete current admins. Fig. 28 shows that the landowner will be able to edit his/her personal information such as(password, mobile number,...). Fig. 29 shows that the landowner can send request to the admin to add new greenhouse by sending the greenhouse's address, which plant type he/she wants to plant and the greenhouse's dimensions. Fig. 30 shows that the landowner will be able to view all information about his greenhouses. Fig. 31 shows that the admin can accept/reject the landowner's requests to add new greenhouse to the system.

7 Requirements Matrix



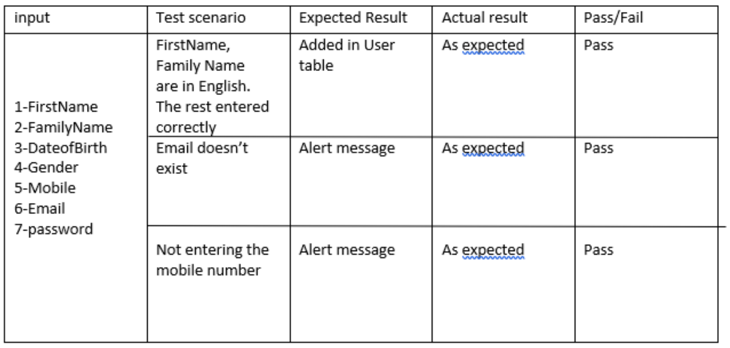
Requirement ID	Requirement Name	Requirement Description	Status
F1	Read real-time video frames.	This function extracts frames from real-time video.	Completed
F2	Save frames into database.	This function is to store frames into the database.	Completed
F3	Retrieve frames from database.	This function is to get the saved frames from database.	Completed
F4	Convert RGB images to HSV	This function is to convert the images from RGB to HSV images.	Completed
F5	Extract features from images	This function is to extract features from the HSV images.	Completed
F6	Run_SVMClassifier	Classify the extracted features of the images by the usage of the OneClassSVm classifier to show if there is any tomatoes in the land.	Completed
F7	Compare testing percentage with Threshold	This function is to compare the percentage of the desired green range of the plant and the desired color range of the fruit/vegetable that the system calculated from HSV testing images with the pre-calculated Threshold percentage from the trained dataset.	Completed
F8	Detecting diseases	This function is used to detect if there is a specific disease starts to appear on the fruit/vegetable.	Completed
F9	Add notification content	The admin can add new notification content that can be sent to the landowner.	In Progress
F10	Delete notification content	The admin can delete a notification content that can be sent to the landowner.	In Progress
F11	Read data from sensors	This function is used to take the readings from the used sensors on our system.	In progress
F12	Turn on Fans	Turn on Fans	In progress
F13	Turn off the fan	This function is used to turn off the fans according to the readings of the DHT11 sensor.	In progress
F14	Turn off LED lights	This function is used to turn off the LED lights after the specified time ends or the plant is at the harvesting stage	Completed
F15	Login	This function is used to let users get into the system.	Completed

F16	Sign Up	This function is to create accounts for landowners.	Completed
F17	Encrypt password.	This function is used to translate password into another form to keep it secured.	Completed
F18	Decrypt password.	This function is used to translate the password back to its original form.	Completed
F22	Reset password	Enable the user to reset his/her password.	Completed
F23	Logout	Enable the user to logout.	Completed
F24	View growth statistics	Shows the rate of plant's growth across all the stages it passes by.	Completed
F25	View all landowners' information.	Admin will be able to view all landowners' information.	Completed
F26	Add sensor type	Admin can add a new sensor to the system.	In Progress
F27	Delete sensor type	Admin can delete a sensor from the system.	In Progress
F28	View all sensor types	Admin can view all the sensors data in the system.	In Progress
F29	Add plant type	Admin can add a new plant type to the system.	Completed
F30	Delete plant type	Admin can delete a plant from the system.	Completed
F31	View all plant types	Admin can view all the plant data in the system.	Completed
F32	Add user role	Admin can add a new user role to the system.	In Progress
F33	Delete role	Admin can delete a user role from the system.	In Progress
F34	View all user roles	Admin can view all the user roles data in the system.	In Progress
F35	Add LED color	Admin can add a new LED color to the system.	In Progress
F36	Delete LED color	Admin can delete a LED color from the system.	In Progress
F37	view LED colors	Admin can view all the LED colors in the system.	In Progress
F38	Add time interval	Admin will able to set a time interval of a LED color to be turned on in a specific land.	In Progress
F39	Delete time interval	Admin will able to delete a time interval of a LED color to be turned on in a specific land.	In Progress
F40	view timer interval details	Admin will able to view all time intervals in the system	In Progress
F41	Update time interval	Admin will able to update the time interval of a LED color to be turned on in a specific land.	In Progress



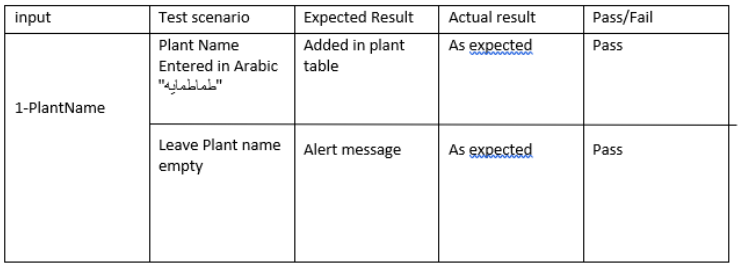
F42	Add new Admin	The admin has the ability to add another admin who will have all the authority of admins	Completed
F43	Add land request	The land owner can add new land request by filling the required form	Completed
F44	Update Land Request	The land owner can send a request if he/she wants to update anything related to his/her existing land	Completed
F45	Land delete request	The landowner will send a request to delete a certain land that he owns.	Completed
F46	View all lands	The landowner views all his lands registered in the system.	Completed
F47	Admin accept land add request	The Admin will accept the request of the landowner to add a new land.	Completed
F48	Admin reject land add request	The Admin will reject the request of the landowner to add a new land.	Completed
F49	Admin accept land edit request	The Admin will accept the request of the landowner to update an information in his land.	In progress
F50	Admin reject land edit request	The Admin will reject the request of the landowner to update an information in his land.	In progress
F51	Admin accept land delete request	The Admin will accept the request of the landowner to delete his registered land.	Completed

7.1 Test cases



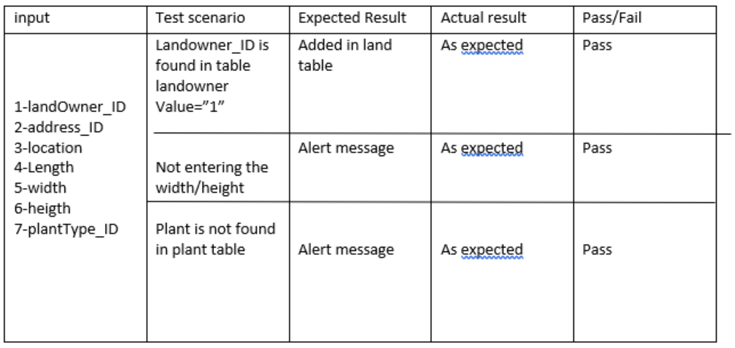
input	Test scenario	Expected Result	Actual result	Pass/Fail
1-FirstName 2-FamilyName 3-DateofBirth 4-Gender 5-Mobile 6-Email 7-password	FirstName, Family Name are in English. The rest entered correctly	Added in User table	As expected	Pass
	Email doesn't exist	Alert message	As expected	Pass
	Not entering the mobile number	Alert message	As expected	Pass

Figure 34: Add new admin



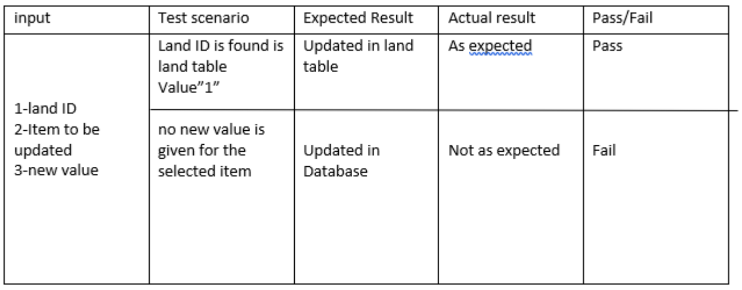
input	Test scenario	Expected Result	Actual result	Pass/Fail
1-PlantName	Plant Name Entered in Arabic "طماطمايه"	Added in plant table	As expected	Pass
	Leave Plant name empty	Alert message	As expected	Pass

Figure 35: Add new plant



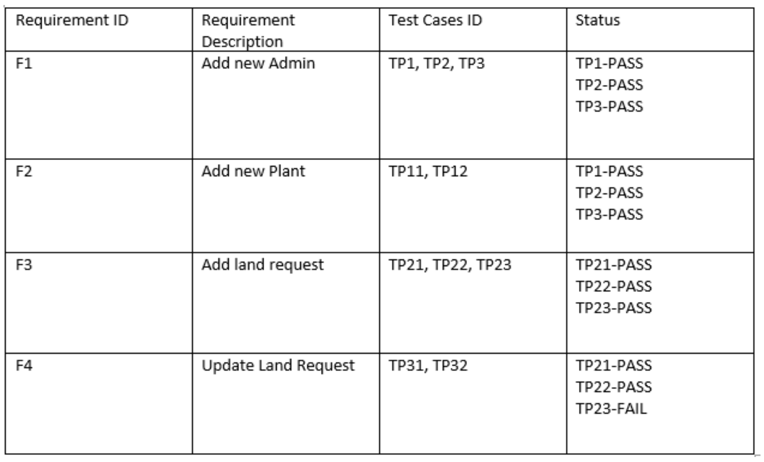
input	Test scenario	Expected Result	Actual result	Pass/Fail
1-landOwner_ID 2-address_ID 3-location 4-Length 5-width 6-heigth 7-plantType_ID	Landowner_ID is found in table landowner Value="1"	Added in land table	As expected	Pass
	Not entering the width/height	Alert message	As expected	Pass
	Plant is not found in plant table	Alert message	As expected	Pass

Figure 36: Add new land request



input	Test scenario	Expected Result	Actual result	Pass/Fail
1-land ID 2-Item to be updated 3-new value	Land ID is found is land table Value"1"	Updated in land table	As expected	Pass
	no new value is given for the selected item	Updated in Database	Not as expected	Fail

Figure 37: Update land request



Requirement ID	Requirement Description	Test Cases ID	Status
F1	Add new Admin	TP1, TP2, TP3	TP1-PASS TP2-PASS TP3-PASS
F2	Add new Plant	TP11, TP12	TP1-PASS TP2-PASS TP3-PASS
F3	Add land request	TP21, TP22, TP23	TP21-PASS TP22-PASS TP23-PASS
F4	Update Land Request	TP31, TP32	TP21-PASS TP22-PASS TP23-FAIL

Figure 38: Requirements

References

[1] Xanthoula Eirini Pantazi, Dimitrios Moshou, and Alexandra A Tamouridou. Automated leaf disease detection in different crop species through image features analysis and one class classifiers. *Computers and electronics in agriculture*, 156:96–104, 2019.