

Smart Planting

by

Randa Osama
Nour El-Huda Ashraf
Amina Yasser
Salma AbdelFatah

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Computer Science

in

Misr International University

in the

Faculty of Computer Science

of the

Misr International University, EGYPT

Thesis advisor:
Dr. Ashraf AbdelRaouf
Eng. Noha El Masry

(July 2020)

Abstract

Smart planting is a new approach for making use of recent technology in the agriculture sector. Meanwhile, it's a challenging research topic. This document proposes an Automated greenhouse to provide the plants with a favorable atmosphere to make them grow faster and healthier. This has been accomplished by the contribution of both Image Processing techniques and Deep Learning. The system is in action when the real-time cameras start taking frames from the greenhouse then an Arduino reads the DHT11 and soil moisture sensors' readings. After this is done, these frames are passed to the pre-processing stage to detect the desired green range of the plant and the desired color range of the fruit/vegetable. This is done using masking techniques with Hue-Saturation-Value(HSV). By the usage of Histogram of oriented gradient(HOG), the features of fruits/vegetables will be extracted such as it's shape and color. Finally, the classifier detects if there is any fruit/vegetable appearing in the image. This helped the system to detect the correct stage of the plant; whether it's the seeding, flowering or harvesting stage. The classifier used in this process is One-Class Support Vector Machine (OC-SVM). Our experiments were conducted on tomato plants. During the plants' classification stages, the system starts to detect if there are any early recognition of diseases affecting our plants. This was done by using the Deep Learning library called Fastai. By the usage of this library, the system is able to detect most of tomatoes disease such as early blight, late blight, leaf mold, spider mites, target spot, mosaic virus, and yellow curl virus. The proposed approach achieved 81.8% accuracy in the tomato's classification and achieved 98.4% accuracy in detecting 9 types of tomato's diseases. A web application is developed for users to supervise the greenhouses and to be updated with their plants' health.

Acknowledgments

This project would never have been possible without the support and guidance of various people at Misr International University. We are heartily thankful to our supervisors, Prof. Ashraf Abd El-Raouf, and Eng. Noha El-Masry for moral support, expert advice, ideas, guiding and helping us throughout the time as their students, it is truly an honor. We have been extremely lucky to have a supervisors who cared about our work, and who responded to our questions and queries so instantly. We also wish to thank Eng. Hany Youssef (Engineering Management Department, Misr International University, Cairo, Egypt) for providing us with our greenhouse model base. Finally, we wish to thank Dr. Abd Allah A. El-Deeb (Vegetable Breeding Department, Horticulture Research Institute, Agricultural Research Center, Giza, Egypt) for providing assistance in planting our greenhouse.

Contents

Abstract	ii
Acknowledgments	iii
List of Figures	4
List of Tables	6
1 Introduction	7
1.1 Introduction	7
1.1.1 Background	7
1.1.2 Motivation	8
1.1.3 Problem Definition	8
1.2 Project Description	9
1.2.1 Objective	9
1.2.2 Scope	10
1.2.3 Project Overview	10
1.3 Project Management and Deliverable	11
1.3.1 Tasks and Time Plan	11
1.3.2 Budget and Resource costs	12
1.3.3 Supportive Documents	12
2 Literature Work	14
2.1 Related work	14
2.2 Comparison with Proposed Project	18
3 System Requirements Specifications	19
3.1 Introduction	19
3.1.1 Purpose of this chapter	19
3.1.2 Scope of this chapter	19
3.1.3 Overview	20
3.1.4 Business Context	21
3.2 General Description	21
3.2.1 Product Functions	21

3.2.2	User Characteristics	22
3.2.3	User Problem Statement	22
3.2.4	User Objectives	22
3.2.5	General Constraints	22
3.3	Functional Requirements	23
3.4	Interface Requirements	41
3.4.1	User Interfaces	41
3.4.2	Hardware Interfaces	41
3.4.3	Communications Interfaces	42
3.4.4	Software Interfaces	42
3.5	Performance Requirements	42
3.6	Design Constraints	42
3.6.1	Standards Compliance	42
3.6.2	Hardware Limitations	43
3.7	Other non-functional attributes	43
3.7.1	Security	43
3.7.2	Reliability	43
3.7.3	Maintainability	43
3.7.4	Portability	43
3.7.5	Usability	43
3.8	Preliminary Object-Oriented Domain Analysis	44
3.8.1	Inheritance Relationships	45
3.8.2	Class descriptions	45
3.9	Operational Scenarios	50
3.10	Appendices	60
3.10.1	Definitions, Acronyms, Abbreviations	60
4	Software Design Document	61
4.1	Introduction	61
4.1.1	Purpose	61
4.1.2	Scope	61
4.1.3	Overview	62
4.1.4	Definitions and Acronyms	62
4.2	System Architecture	63
4.2.1	Architectural Design	63
4.2.2	Decomposition Description	66
4.2.3	Design Rationale	79
4.3	Data Design	80
4.3.1	Data Description	80
4.3.2	Data Dictionary	82
4.4	Component Design	83
4.4.1	Data Input	83
4.4.2	Preprocessing	83
4.4.3	Processing	83
4.4.4	Classification	84

4.5	Human Interface Design	87
4.5.1	Overview of User Interface	87
4.5.2	Screen Images	87
4.6	Requirements Matrix	99
5	Evaluation	102
5.1	Experiment 1 - Pre-Experiment	102
5.1.1	Setup	102
5.1.2	Goal	102
5.1.3	Task	103
5.1.4	Results	103
5.1.5	Discussion	103
5.2	Experiment 2 - Detecting the tomato	103
5.2.1	Setup	103
5.2.2	Goal	103
5.2.3	Task	103
5.2.4	Results	104
5.2.5	Discussion	104
5.3	Experiment 3 - Detecting the plant disease	104
5.3.1	Setup	104
5.3.2	Goal	104
5.3.3	Task	104
5.3.4	Results	104
5.3.5	Discussion	106
5.4	Experiment 4 - User feedback	106
5.4.1	Setup	106
5.4.2	Results	107
6	Conclusion	108
6.1	Future work	109
	Bibliography	109

List of Figures

1.1	Website Logo	9
1.2	System Overview	11
1.3	purchases	12
1.4	Asking for dataset	12
1.5	Response	13
1.6	Asking for dataset	13
1.7	Our conference paper acceptance	13
2.1	(Data-set of tomato leaves infected with diseases, Jia Shijie et. al. [1]	16
2.2	(figure (a) for a healthy leaf, (b) for an unhealthy leaf, Jinzhu Lu et. al. [2]	16
2.3	Comparison with Proposed Project	18
3.1	System Overview	21
3.2	Laptop	41
3.3	Arduino	41
3.4	Camera	42
3.5	Data Base schema	44
3.6	Class Diagram	44
3.7	Inheritance Relationship	45
3.8	Usecase	50
4.1	Architectural Design	63
4.2	Class Diagram	66
4.3	Singleton design pattern	66
4.4	Singleton design pattern	67
4.5	Observer design pattern	67
4.6	MVC design pattern	68
4.7	Activity Diagram	73
4.8	Landowner adds new request	74
4.9	Landowner requests to delete the land	75
4.10	Landowner requests to update the land	76
4.11	Preprocessing	77
4.12	Automated and Classifier	78
4.13	AcceptRejectLand	79

4.14 Database Schema	80
4.15 Before and after the HSV masking effect on tomatoes' testing image	84
4.16 flow chart	85
4.17 Login-Page	87
4.18 SignUp	88
4.19 Admin Updating his/her INFO	88
4.20 Admin Viewing All Landowners	89
4.21 Admin Viewing All landowner's requests for adding new greenhouse	89
4.22 Admin update Plants	90
4.23 Admin View All Plant	90
4.24 Admin view statistics	91
4.25 Admin adding a new admin	91
4.26 Led light colors	92
4.27 Admin assign led light color to each plant is each stage	92
4.28 Sensors	93
4.29 Admin view update/delete requests of the greenhouse	93
4.30 Landowner making a new request with a new greenhouse	94
4.31 Landowner viewing all his/her land requests in the system	94
4.32 Landowner viewing all his/her lands in the system	95
4.33 Landowner update his/her lands request	95
4.34 Landowner view statistics	96
4.35 Landowner view greenhouse images	96
4.36 Landowner view notifications	97
4.37 Required Matrix 1	99
4.38 Required Matrix 2	100
4.39 Required Matrix 3	101
5.1 Confusion matrix	105
5.2 Plotting	106

List of Tables

3.1	Definitions, Acronyms and Abbreviations	60
4.1	Definitions and Acronyms	62
5.1	Experiments algorithms accuracy	104
5.2	Experiments algorithms accuracy	105
5.3	User study	107

Chapter 1

Introduction

1.1 Introduction

1.1.1 Background

Plants in all of its types; whether fruits or vegetables, is one of the main courses in human life. Nowadays, due to some climate changes and geographical locations, some plants are endangered of extinction. Due to these climate-changes [3–5], in some countries the sun could appear for a shorter or longer time than the time needed for the plant to grow in a healthy life cycle. Even though, the sun plays an important role in a plant's growth, but not all of the 7 spectrum colors produced from the sun are important for the plant. There are only three important spectrum colors, which are: green, blue, and red. Green and blue spectrum are essential for producing strong roots and leaves while red spectrum for making the plants more blossomed [6].

While each plant has its specific growing season. Still some farmers want to grow plants at any season they want and to have the fastest growth rate. Consequently, a greenhouse is used for growing plants during their off-seasons; As they are a well-known isolated environment that helped farmers to grow all their needed plants. Meanwhile, for increasing the growth rate, we had recommend the usage of supplementary lights as several researchers had reported that they improved the growth rate of vegetables/fruits [7].

Farmers and landowners could be provided with all of their needed plants. Meanwhile, greenhouses are not only a solution for protecting plants but also they are a solution for an economical increase, as it helped in increasing Egypt's economy [8].

By the usage of Technology and greenhouses, an automated greenhouse was created. As that allowed us to have a better control over the processes and maintaining a favorable atmosphere inside the greenhouse. By the addition of using Image processing and deep learning together, the appearance of any diseases on the plants was detected, which led to high quality yield by reducing the wastage.

1.1.2 Motivation

This project takes into consideration that it's hard to distinguish the plant's stages manually whether it's in the seeding, flowering or harvesting stages.

Due to the different weather conditions that any plant could be exposed to, the plant could be affected by any kind of disease. Usually, these diseases can't be recognized by naked eyes of landowners' and farmers' in their early stages. This may cause the death of the plant and its' surroundings. Instead of curing the infected plant, the landowner would be in threat of losing all the other plants and crops in the greenhouse. Therefore, if an automated system is implemented in their greenhouse it would help in saving the plants from dying.

Although COVID-19 has a huge impact on the economy of the agriculture due the lockdown and curfew. Greenhouses supported with smart planting system won't be affected by the lockdown of the workers since the system operates dynamically and allows the user to check the crops and their growth, also it notifies the user if there is any disease found on the leaves of the plant. So, the system doesn't need the human power and the crops in the greenhouse won't be affected by the pandemic. Some economists has early estimates that throughout 2020, the global economic growth will loss almost 2.4% of the value of the gross domestic product (GDP).[9].

1.1.3 Problem Definition

The agricultural sector in Egypt faces major challenges. Some crops take too much time to be ready to be harvested, and not necessarily all the crops are efficiently harvested. Therefore, leaving a negative impact on the economy of the agricultural sector. The high humidity caused by the sunlight can damage and cause some diseases to the plants. The market currently lacks a system which automatically detects these diseases with high accu-

racy and makes an artificial environment to the plants to grow faster and healthier to save time, money and effort.

1.2 Project Description

Smart planting is an automated system implemented inside a greenhouse that would help in increasing the growth rate of plants and detecting the early stages of diseases affecting the plants. That was achieved by the installing some LED lights (red, green and blue), real-time camera for monitoring and protecting plants, and sensors(DHT11 and soil moisture) that would help in creating the favorable weather conditions for the plants to grow. Also, a web-application has been developed for users to monitor the growth rate and to be notified if there are any diseases in their greenhouse. The website logo is shown in Figure 1.1.



Figure 1.1: Website Logo

1.2.1 Objective

The proposed system is designed to automatically detect the plants in the greenhouse to classify its stages. LED lights are added to the green house which will be automatically switched on for the specified hours and accordingly increase the growth rate of the plant, reduce human effort and save time. Also, automatic detection for the diseases that affect the plant's leaves and notify the user using web application. The early detection of it, allows the landowner to use the right treatment for this disease and also allows save the other crops from infection.

1.2.2 Scope

1. The system will detect the plants diseases for tomatoes accurately which are: early blight, late blight, leaf mold, spider mites, target spot, mosaic virus, septoria, bacterial spot and yellow leaf curl virus.
2. The system will classify the growth stage that the plant had been reached to turn on the needed LED light color.
3. The landowner will get notified with their land's important updates.
4. The system have the ability to learn different types of diseases to enhance the accuracy in the future.

1.2.3 Project Overview

Our proposed system consists of four linked stages.

The first stage is the data input, the real time camera starts to capture frames for the greenhouse every 5 minutes without any human intervention and these frames are saved directly in the database.

Secondly, those images enter the pre-processing stage which is converting the image frame from RGB to HSV to detect the desired colors for each plant type.

The third stage is the processing, which is defined in feature extraction that is used to extract the plants' features using histogram of oriented gradient (HOG) such as: it's size, shape and color, after this process all the data are saved in the database.

Moving to the classifying stage using One-Class Support Vector Machine, it is used to detect the growth stage that the plant had been reached. Also detecting if any disease affected the plant using Fastai deep learning model.

Then, the results will make the system take it's decision to generate the suitable LED light needed to be turned on. At the end the landowner will be notified when the greenhouse is ready to be harvested and if theirs any disease appeared on the plants.

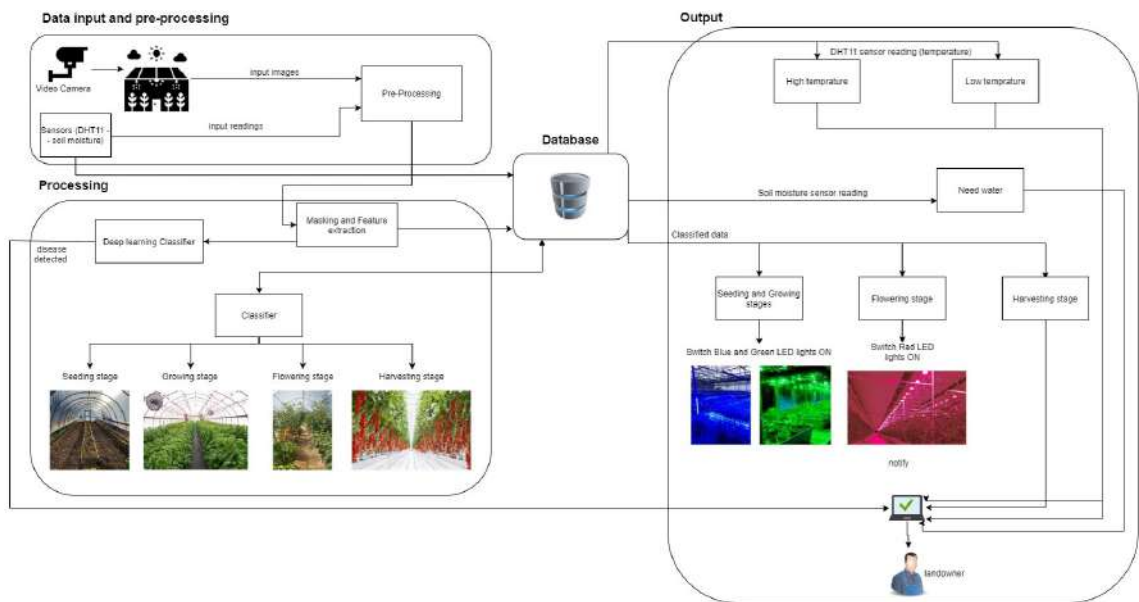
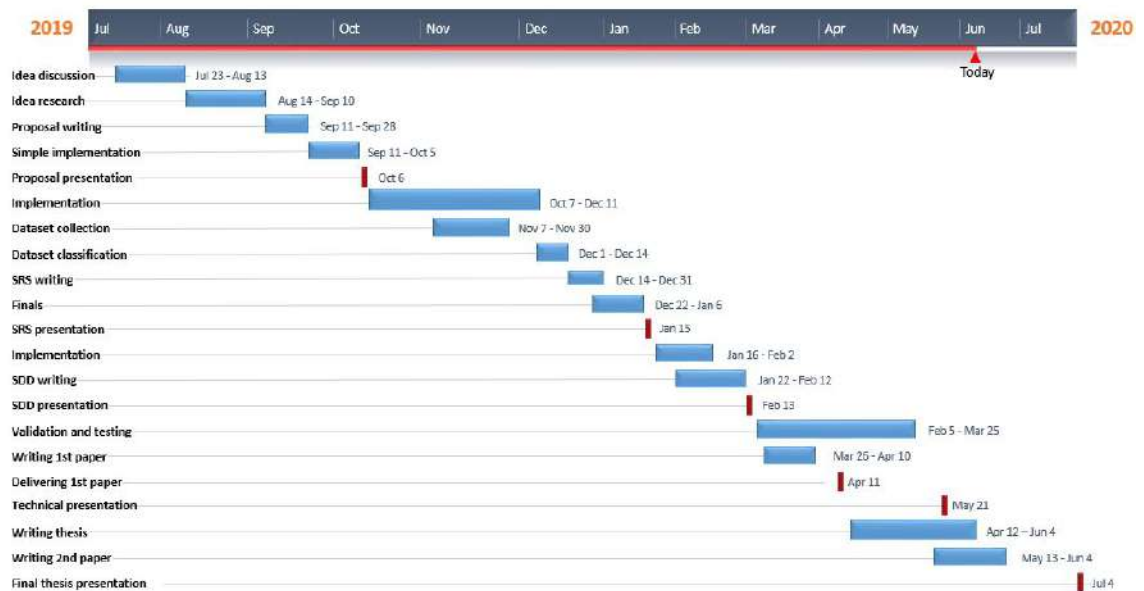


Figure 1.2: System Overview

1.3 Project Management and Deliverable

1.3.1 Tasks and Time Plan



1.3.2 Budget and Resource costs

Product	Price
Arduino Mega 2560	EGP 300.00
Peat Moss	EGP 250.00
Vermiculite	EGP 50.00
Perlite	EGP 250.00
Tomatoes Seedlings	EGP 50.00
12x12 Pc Fans	EGP 80.00
Logitech Webcam C21	EGP 450.00
5m RGB LED Strip	EGP 400.00
DHT11 and Soil Moisture Sensors	EGP 100
Greenhouse wooden box	EGP 600
Extra fees of the greenhouse	EGP 800

Figure 1.3: purchases

1.3.3 Supportive Documents



Figure 1.4: Asking for dataset

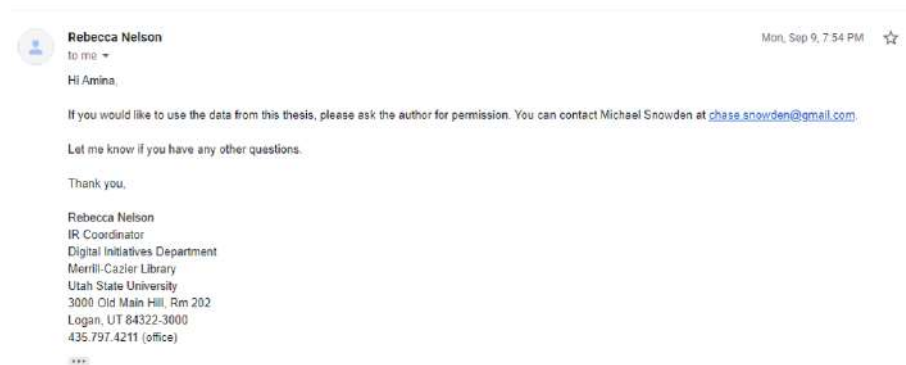


Figure 1.5: Response



Figure 1.6: Asking for dataset



Figure 1.7: Our conference paper acceptance

Chapter 2

Literature Work

2.1 Related work

Several researches proposed different systems related to an automated greenhouses. Digging deep into some of these researches. A greenhouse system proposed by Khamis et. al. [10] is controlled remotely and their goal is to enhance the plant growth by using the artificial lights. Their objective is to use LED as the artificial light, to develop an automated system to control the greenhouse and to improve the sensors reliability by using the Internet. They used temperature, humidity, oxygen sensors with the help of water value and micro-controller. Their software is implemented using Python. Their data are stored in SQL database, then it's processed to be displayed in CHartJS and PHP graphs. This paper aimed to enhance the plant growth using the LED lights and automate the whole system so the farmer won't need to go to the greenhouse, he can track the plant's growth remotely. Another research by Drakulić and Mujčić [3] proposed a system with a greenhouse of size 120x60 cm. They used the optimum conditions that is the same needed in the real greenhouse. They applied their experiment on two types of flowers, strawberry and pepper. Their objective is to control all the optimum needed parameters in the greenhouse in respect of minimum energy consumption rate. They used TFT LCD touch display to control the used parameters so the users can review and monitor the work done in the greenhouse. They saved the parameters data in the cloud.

Danila et. al. [6] proposed a system that measures the effect of different color of LED lights in an restrictive conditions for a greenhouse construction. By the usage of LED Spectra dedication software. Their objective is to find the suitable lighting and spectrum color for

plant growth. It was obtained by the usage of dedication software; that Red and Blue lights are proven to be the most effective on plants, as for the lights, LED is the best compared with fluorescent, metal halide and natural light. Their system shows that the production of plants in greenhouses can be obtained faster by the help of artificial lighting system than outdoors planting.

An experiment that aims to the highly effect of greenhouse agriculture on food production using LED lights by Watson et. al. [11]. The system can be controlled automatically, also it reduces lighting cost around 60%. The paper clarifies the plant's physiology which is converting photons such as water, co2 and the three main lights into sugars and oxygen. At the end, The solution has been implemented through a system and sensors in a greenhouse and produces savings of 64 percent with LED lights.

While other researches were done on how to detect plants' diseases with the usage of different image processing techniques. As a system that was proposed by Jia Shijie et. al. [1] Proposed a system for detecting certain kinds of diseases that seriously affects tomatoes, such as a) Tomato bacterial spot, b) Tomato early blight, c) Tomato late blight, d) Tomato leaf mold e) Tomato septoria leaf spot, f) Tomato two spotted spider mite, g) Tomato target spot h) Tomato mosaic virus, i) Tomato yellow leaf curl virus and j) Tomato gray spot. The effect of these diseases is shown in the following collected data-set of the tomatoes leaves. 2.2. Detecting is done by deep convolution neural network to identify 26 diseases of 14 crops. There are two algorithms that are employed, respectively. The first(VGG16+SVM)) is to employ VGG16 as image feature extraction, with the combine SVM classifier in detecting tomato pests and diseases on leaf images. The second (Fine-tuning) is using fine-tuning to construct an end-to-end classification model based on the original VGG16 model.

Another proposed system by H. Sabrol and K. Satish [12]. Proposed a system based on image processing for detecting plant diseases and classifying them. Their system only classify 5 different types of diseases that are tomato late blight, Septoria spot, bacterial spot, bacterial canker, tomato leaf curl. The data-set was a combination of images of the plant leaves and stem, for the plants that are affected and not affected plants. Classification was done on six types of tomato images yielded that had reached an accuracy of 97.3%. The infected plant pattern recognition was done by the usage of some techniques such as "neural networks", "support vector machine", and classification was done by the usage of classifi-



Figure 2.1: (Data-set of tomato leaves infected with diseases, Jia Shijie et. al. [1])

cation tree.

Another proposed system by Jinzhu Lu et. al. [2] Proposed a new technique in recognizing yellow leaf curl disease on tomato plants, and that was by their usage of hyper-spectral imaging technique. Images of the leaves were taken to get the selected sensitive bandwidth and band ratios. They had selected Hyper-spectral imaging technology as it's well known of combining the advantages of machine vision and spectroscopy. Detection accuracy had reached 92%. But by the usage of SVM, it had a short operating time then without SVM, and their accuracy in detecting yellow leaf curl had increased to reach to be 95%. In the following figure, it shows the difference between a healthy leaf and an unhealthy leaf.

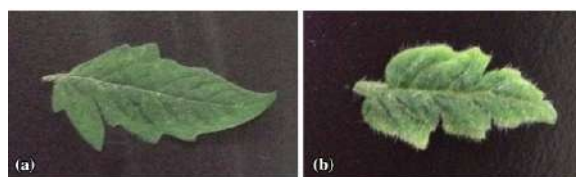


Figure 2.2: (figure (a) for a healthy leaf, (b) for an unhealthy leaf, Jinzhu Lu et. al. [2])

Chit Su Hlaing and Sai Maung Maung Zaw [13] proposed a system that can be able to distinguish between six tomato diseases. The system is simply working by taking a plant leaf image using phone cameras whereas the output will be the disease's name. First, the

input image passes by preprocessing algorithm to remove the background of the leaf's image. By using SIFT features that are extracted from the previous stage, then it passes by Generalized Extreme Value (GEV) Distribution to represent information about the image in a small number of dimensions. These diseases are Leaf Mold, Septoria Leaf Spot, Two Spotted Spider Mite, Late Blight, Bacterial Spot and Target Spot.

Reza Ghaffari et. al [14] proposed a solution to reduce the environmental pressure, cost and resource use by making an automatic diagnosis system for plant diseases. They say that it's a major challenge to detect the diseases before they spread throughout the whole greenhouse. A statistical and intelligent systems techniques are used to detect plants' diseases with a very high accuracy. Many clustering algorithms were used to visualise any clusters within the dataset. Multi-Layer Perceptron (MLP), Learning Vector Quantization (LVQ) and Radial Basis Function (RBF) based Artificial Neural Network (ANNs) were used to learn to classify and categorise the datasets. 94,96 and 98% classification accuracy are achieved by using the RBF,MLP and LVQ techniques for the healthy,powdery mildew and spider mite infected plants.

2.2 Comparison with Proposed Project

	Our Proposed system	Similar system1	Similar system2	Similar system3	Similar system4
Detection	Plant stages and Tomatoes diseases: early blight, late blight, leaf mold, spider mites, target spot, mosaic virus, and yellow curl virus. Also, temperature, humidity and water content.	Detecting diseases that seriously affects tomatoes, such as bacterial spot, early and late blight, leaf mold, septoria leaf spot, two spotted spider mite, target spot, Tomato mosaic virus, yellow leaf curl virus and gray spot.	Their system only classify 5 different types of diseases that are tomato late blight, septoria spot, bacterial spot, bacterial canker, tomato leaf curl.	Their system detects tomato yellow leaf curl virus.	Their system detect any plant diseases that affect plant leaves.
Pre-processing	RGBtoHSV color Sensor Readings using Arduino	The input images passes through a stack of convolution layers.	computed total nine features from RGB healthy and unhealthy images of the tomato plant. The R, G, B color components extracted from the RGB images.	Mathematical equations was operated on hyperspectral images.	only extract green region and brown region to get whole leaf image with removed background. Bit OR operation is performed. Median filtering.
Segmentation	Masking using Hue-Saturation-Value(HSV)	three Fully-Connected (FC) layers	The Otsu's segmentation	Spectral dimension analysis	k-means clustering
Feature	Feature extraction using HOG Histogram of oriented gradient	(VGG16-SVM) and Fine-tuning	Extracted eleven shape features: area, Euler number, orientation extent, perimeter, convex area, filled area, eccentricity, majoraxis length, equidiameter, and minaxislength. another way using a Feature Extraction Module	Was done by 2 ways: 1- Otsu's method 2-perform the logical AND operator between the mask and both selected sensitive wavelength images. Texture features were extracted using grey level co-occurrence matrix (GLCM).	SIFT Algorithm.
Classification	One-Class SVM for plants stages and FastAI for classifications of diseases	SVM	Classification tree	SVM	Support Vector Machine (SVM)
Learning and finding	Yes	Yes	Yes	No	Yes
Accuracy	31.8% in plants stages 98.4% in detecting Tomatoes diseases	Accuracy of 69%	Accuracy of 97.3%	Accuracy of 97.2%	Accuracy of 84.7%
Organization	-----	IEEE 2017	IEEE 2018	Springer 2017	IEEE 2017

Figure 2.3: Comparison with Proposed Project

Chapter 3

System Requirements Specifications

3.1 Introduction

3.1.1 Purpose of this chapter

The main purpose of this System Requirements Specifications document is to represent a detailed description of our system (Smart Planting) requirements. Smart Planting system mainly monitor and control the plant under the effect of LED lights to increase the plants' growth rate, also detecting some certain diseases. This documentation will present a fully description about our system's web application and back-end using Python, Arduino and sensors. We also provide a fulfilled illustration about each stage inputs and outputs.

3.1.2 Scope of this chapter

Smart Planting system scope is to help Farmers and land owners increase their plants production using the LED lights; The system is not only working for increasing the growth rate but it also do protect the plants from two types of diseases (Early Blight and Late Blight). This system is a composite of a green house monitored with cameras and sensors; and LED lights which is the main source of the plants' growth, associated with a website, for farmers and landowner to communicate with the system.

3.1.3 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 (DHT22 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 on the database.

Moving to the processing stage, starting with extracting features from the frames in order to detect if there is any green colored fruit/vegetable is found or any diseases effecting the plant. Then using masking with Hue-Saturation-Value (HSV) is used to detect the desired green range of the plant and the desired color range of the fruit/vegetable. 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 had appeared.
2. Growing stage, which some green leaves are being 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 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 high 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 were 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. They system overview is shown at fig 3.1

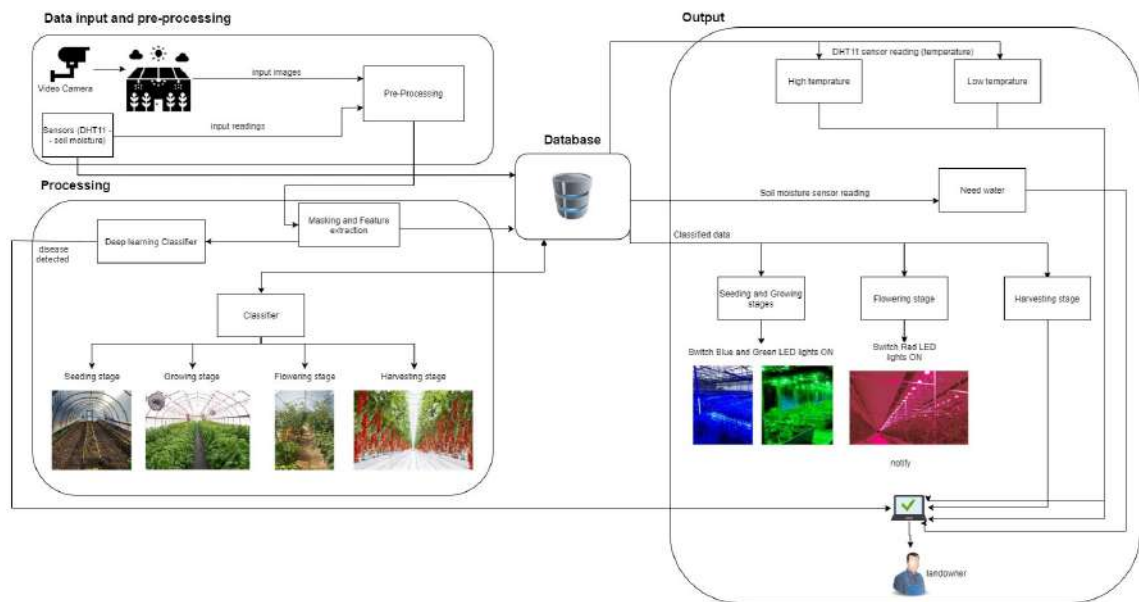


Figure 3.1: System Overview

3.1.4 Business Context

There are some plants that do leave a negative impact on the economy of the country; as plants do take their time in order to get ready to be harvested, and some of the plants do die before being harvested which made a huge effect on the market. Our system is striking to increase the amount of plants and protecting plants by giving them their suitable environment. Meanwhile, it provides a less cost equipment for producing more plants while land owners and farmers having double of their normal income, with less percentage of plants loss during their growth.

3.2 General Description

3.2.1 Product Functions

The system aims to increase the plants' growth rate. The main idea of the system is to automate the greenhouse with video cameras, sensors, LED lights, and fans for assisting the plant to grow faster than the effect of the sunlight on it. The system will increase plant production in a shorter time, so the economy of the country will increase. The user can monitor/track the parameters and the plant's growth inside the greenhouse using a website.

3.2.2 User Characteristics

There are 2 types of users that interact with the system: Admins and Landowners. Both types of users have different use of the system so each of them has his own requirements.

1. Admin:

- Must have domain knowledge.
- Must be able to interact with the UI.
- Must be able to manage and monitor the database.
- Must be able to insure the security of the data.

2. Landowner:

- Must be able to monitor the greenhouse.
- Must be able to interact with the UI to receive notifications, view some statistics and send requests to add, edit or delete land.

3.2.3 User Problem Statement

The agricultural sector in Egypt faces major challenges. Farmers and landowners wait for the land too much time to be harvested. The high humidity caused by sunlight can damage and cause some diseases to the plants. The market currently lacks a system that automatically detects these diseases with high accuracy and makes an artificial environment for the plants to grow faster and healthier to save time, money, and effort.

3.2.4 User Objectives

- 1- Automated system to accurately detect plant diseases.
- 2- Automated greenhouse with video cameras, LED lights and fans for speeding up the plant growth and increasing the production.
- 3- Notifying the landowner when the greenhouse is ready to be harvested.
- 4- Notifying the landowner when their is a strange behavior or disease appeared on the plants.

3.2.5 General Constraints

One of the main constraints of the system it the variant light conditions that the camera could face, as the video cameras should capture each stage of the plant's growth so if there is a low lightening, the main features that we want won't be detected clearly.

3.3 Functional Requirements

ID	1
Function	Read real-time video frames.
Description	This function extracts frames from real-time video.
Action	Frames will be extracted.
Requirements	Performed by the system.
Input	Real Time video.
Output	Return true or false if the image was taken successfully or not.
Precondition	Camera is set to capture the real time video.
Postcondition	Save the frames into database.
Dependencies	None

ID	2
Function	Save frames into database.
Description	This function is to store frames into the database.
Action	A defined number of frames will be saved into the database.
Requirements	Performed by the system.
Input	Frames extracted from real time video.
Output	Return true or false if the image was saved successfully or not.
Precondition	Connect to the database and the frames was extracted successfully from the real time video.
Postcondition	The Frames are saved in database.
Dependencies	1

ID	3
Function	Retrieve frames from database.
Description	This function is to get the saved frames from database.
Action	The frames will get retrieved.
Requirements	Performed by the system.
Input	Time of the frames captured.
Output	Frames.
Precondition	Connect to the database.
Postcondition	The Frames are retrieved from database.
Dependencies	2

ID	4
Function	Convert RGB images to HSV
Description	This function is to convert the images from RGB to HSV images.
Action	The images are converted into HSV images.
Requirements	Performed by the system.
Input	Frames
Output	Masked images
Precondition	Images are retrieved successfully from the database.
Postcondition	Image is converted into HSV.
Dependencies	3

ID	5
Function	Extract features from images
Description	This function is to extract features from the HSV images.
Action	Features extracted from the images.
Requirements	Performed by the system.
Input	HSV images.
Output	Return the features extracted from the images.
Precondition	Images are converted from RGB to HSV.
Postcondition	Features are extracted
Dependencies	4

ID	6
Function	Run_SVMClassifier
Description	Classify the extracted features of the images by the usage of the OneClassSvm classifier to show if there are any tomatoes in the land.
Action	Show if there are any tomatoes in the land.
Requirements	Performed by the system.
Input	Features extracted from the images
Output	Return true or false if the image contains any tomatoes in the land.
Precondition	Feature are extracted successfully from the input images.
Postcondition	Show if there are any tomatoes in the land.
Dependencies	5

ID	7
Function	Compare testing percentage with Threshold
Description	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.
Action	Specific LED lights will be turned on according to the stage of the plant.
Requirements	Performed by the system.
Input	Percentage of the desired green range of the plant, percentage of the desired color range of the fruit/vegetable.
Output	A specific LED lights will be turned on by calling the Arduino pin number connected to the specific light in the LED strip
Precondition	Connect to database to get the threshold value, images are converted into HSV, the classification must be done and LED strips must be connected to the Arduino.
Postcondition	Specific LED lights will be turned on.
Dependencies	6

ID	8
Function	Detecting diseases
Description	This function is used to detect if there is a specific disease starts to appear on the fruit/vegetable.
Action	Detecting if there is one of the defined diseases is shown on the HSV images.
Requirements	Performed by the system.
Input	HSV images.
Output	Return true or false if a disease detected or not.
Precondition	Converting from RGB to HSV images.
Postcondition	Notification will be sent to the landowner if a disease is detected.
Dependencies	6

ID	9
Function	Login
Description	This function is used to let users get into the system.
Action	Users will login successfully.
Requirements	Performed by the user.
Input	Username and password.
Output	Logged in successfully or try again.
Precondition	The username and password should be found in the database.
Postcondition	Return true or false if the logged in successfully or not.
Dependencies	None

ID	10
Function	Delete notification content
Description	The admin can delete a notification content that can be sent to the landowner.
Action	Deleting a notification content from the database
Requirements	Performed by the Admin.
Input	Content id.
Output	Return true or false if the content was deleted or not.
Precondition	Admin must be logged in. Validate data entered.
Postcondition	Object is deleted from database.
Dependencies	9

ID	11
Function	Read data from sensors
Description	This function is used to take the readings from the used sensors on our system.
Action	Control the led lights intensity and the temperature according to the readings.
Requirements	Performed by the system.
Input	The Arduino pin number connected to the sensors.
Output	The sensors readings.
Precondition	The sensors must be connected to the Arduino.
Postcondition	Control the led lights intensity and the temperature according to the readings.
Dependencies	Sensors

ID	12
Function	Turn on Fans
Description	This function is used to turn on the fans according to the readings of the DHT11 sensor.
Action	The fans will be turned on.
Requirements	Performed by the system.
Input	The sensor reading.
Output	The fans will be turned on.
Precondition	The temperature is higher than the normal.
Postcondition	The temperature will be normal.
Dependencies	11

ID	13
Function	Turn off the fan
Description	This function is used to turn off the fans according to the readings of the DHT11 sensor.
Action	The fans will be turned off.
Requirements	Performed by the system.
Input	The sensor reading.
Output	The fans will be turned off.
Precondition	The temperature is normal or lower than the normal.
Postcondition	The temperature will be normal.
Dependencies	11

ID	14
Function	Turn off LED lights
Description	This function is used to turn off the LED lights after the specified time ends or the plant is at the harvesting stage
Action	LED lights will be turned off.
Requirements	Performed by the system.
Input	Arduino pin numbers connected to the LED strip.
Output	Lights will be turned off.
Precondition	The LED lights specified time ends or the plant is at the harvesting stage.
Postcondition	LED strips will be turned off.
Dependencies	6

ID	15
Function	Add notification content
Description	The admin can add new notification content that can be sent to the landowner.
Action	Adding the new content in the database.
Requirements	Performed by the Admin.
Input	New content.
Output	Return true or false if the content was added or not.
Precondition	Admin must be logged in. Validate data entered.
Postcondition	Object is added in database.
Dependencies	9

ID	16
Function	SignUp
Description	This function is to create accounts for landowners.
Action	The user will be set up successfully into pending state to be accepted by the admin.
Requirements	Performed by the user.
Input	Registration information, land information.
Output	Return true or false if the signup request has been sent to the admin successfully or not.
Precondition	None.
Postcondition	The user will be set up successfully into pending state to be accepted by the admin or have to re-enter any information again.
Dependencies	None

ID	17
Function	Encrypt password.
Description	This function is used to translate password into another form to keep it secured.
Action	Encrypt data using MD5 algorithm.
Requirements	Performed by the system.
Input	Password.
Output	Encrypted password
Precondition	None.
Postcondition	Password is added in the database in the hashed form successfully.
Dependencies	None

ID	18
Function	Decrypt password.
Description	This function is used to translate the password back to its original form.
Action	Decrypt data using MD5 algorithm.
Requirements	Performed by the system.
Input	Hashed Password.
Output	None.
Precondition	Decrypted password
Postcondition	Password is returned to its normal state.
Dependencies	17

ID	19
Function	Add user.
Description	This function is used to add users in the database.
Action	The user is added into the database.
Requirements	Performed by the Admin.
Input	User information.
Output	Return true or false if the user is added successfully or not.
Precondition	Validate data entered. Admin Logged in.
Postcondition	User is added into the database.
Dependencies	9

ID	20
Function	Edit land owner.
Description	This function enables the admin to edit a landowner's information.
Action	Information will be edited.
Requirements	Performed by an admin.
Input	Landowner ID, Edited information.
Output	Return true or false if the user was edited successfully or not.
Precondition	Landowner ID is valid, edited information is valid to be added and the system must be connected to the database.
Postcondition	Landowner will be edited.
Dependencies	9

ID	21
Function	Delete land owner
Description	This function enables the admin to delete a landowner.
Action	Landowner will be edited.
Requirements	Performed by an admin.
Input	Landowner ID.
Output	Return true or false if the user was deleted successfully or not.
Precondition	Landowner ID is valid and the system must be connected to the database.
Postcondition	Landowner will be deleted.
Dependencies	9

ID	22
Function	Reset password
Description	Enable the user to reset his/her password.
Action	User's password will be reset.
Requirements	Performed by a user.
Input	User email.
Output	Confirmation that the password is reset successfully.
Precondition	User email already exists.
Postcondition	User password is changed in the database.
Dependencies	9

ID	23
Function	Logout
Description	Enable the user to logout.
Action	User will be logged out.
Requirements	Performed by a user.
Input	None.
Output	Confirmation that the user has logged out successfully.
Precondition	User is already logged in.
Postcondition	User will be logged out.
Dependencies	9

ID	24
Function	View growth statistics
Description	Shows the rate of plant's growth across all the stages it passes by.
Action	Show statistical graph for the user.
Requirements	Performed by the system.
Input	None.
Output	Plant growth graph.
Precondition	System should be connected to the database. Admin/User Logged in
Postcondition	Statistical graph will be shown to the user.
Dependencies	9

ID	25
Function	View all landowners' information.
Description	Admin will be able to view all landowners' information.
Action	All landowners' information will be shown to the admin.
Requirements	Performed by the admin.
Input	None.
Output	All landowners' information.
Precondition	The system must be connected to the database. Admin Logged in.
Postcondition	All landowners' information will be shown to the admin.
Dependencies	9

ID	26
Function	Add sensor type
Description	Admin can add a new sensor to the system.
Action	New sensor type will be added into the database.
Requirements	Performed by the admin.
Input	Sensor data.
Output	Confirmation that the new sensor is added successfully.
Precondition	No repeated sensor types. Admin Logged in.
Postcondition	New sensor details added in database.
Dependencies	9

ID	27
Function	Delete sensor type
Description	Admin can delete a sensor from the system.
Action	The sensor will be deleted from the database.
Requirements	Performed by the admin.
Input	Sensor data.
Output	Confirmation that the sensor is deleted successfully.
Precondition	Sensor data already exists. Admin Logged in.
Postcondition	Sensor will be deleted.
Dependencies	9

ID	28
Function	View all sensor types
Description	Admin can view all the sensors data in the system.
Action	The sensor will be showed from the database.
Requirements	Performed by the admin.
Input	None.
Output	Show all sensors.
Precondition	System connected to the database. Admin Logged in.
Postcondition	All sensors will be shown.
Dependencies	9

ID	29
Function	Add plant type
Description	Admin can add a new plant type to the system.
Action	New plant type will be added into the database.
Requirements	Performed by the admin.
Input	Plant data.
Output	Confirmation that the new plant is added successfully.
Precondition	No repeated plant data. Admin Logged in.
Postcondition	New plant details added in database.
Dependencies	9.

ID	30
Function	Delete plant type
Description	Admin can delete a plant from the system.
Action	The plant will be deleted from the database.
Requirements	Performed by the admin.
Input	Plant data.
Output	Confirmation that the plant is deleted successfully.
Precondition	Plant data already exists. Admin logged in
Postcondition	Plant will be deleted.
Dependencies	9.

ID	31
Function	View all plant types
Description	Admin can view all the plant data in the system.
Action	The plants will be showed from the database.
Requirements	Performed by the admin.
Input	None.
Output	Show all plants.
Precondition	System connected to the database. Admin logged in.
Postcondition	All plants will be shown.
Dependencies	9.

ID	32
Function	Add user role
Description	Admin can add a new user role to the system.
Action	New user role will be added into the database.
Requirements	Performed by the admin.
Input	User role data.
Output	Confirmation that the new user role is added successfully.
Precondition	No repeated user role data. Admin logged in.
Postcondition	New user role added in database.
Dependencies	9.

ID	33
Function	Delete role
Description	Admin can delete a user role from the system.
Action	The user role will be deleted from the database.
Requirements	Performed by the admin.
Input	User role data.
Output	Confirmation that the user role is deleted successfully.
Precondition	User role already exists. Admin logged in.
Postcondition	User role will be deleted.
Dependencies	9.

ID	34
Function	View all user roles
Description	Admin can view all the user roles data in the system.
Action	The user roles will be showed from the database.
Requirements	Performed by the admin.
Input	None.
Output	Show all user roles.
Precondition	System connected to the database. Admin logged in.
Postcondition	All user roles will be shown.
Dependencies	9.

ID	35
Function	Add LED color
Description	Admin can add a new LED color to the system.
Action	New LED color will be added into the database.
Requirements	Performed by the admin.
Input	LED color data.
Output	Confirmation that the new LED color is added successfully.
Precondition	No repeated LED color data. Admin logged in.
Postcondition	New LED color added in database.
Dependencies	9.

ID	36
Function	Delete LED color
Description	Admin can delete a LED color from the system.
Action	The LED color will be deleted from the database.
Requirements	Performed by the admin.
Input	LED color data.
Output	Confirmation that the LED color is deleted successfully.
Precondition	LED color already exists. Admin Logged in.
Postcondition	LED color will be deleted.
Dependencies	9.

ID	37
Function	view LED colors
Description	Admin can view all the LED colors in the system.
Action	The LED colors will be showed from the database.
Requirements	Performed by the admin.
Input	None.
Output	Show all LED colors.
Precondition	System connected to the database. Admin Logged in.
Postcondition	All LED colors will be shown.
Dependencies	9.

ID	38
Function	Add time interval
Description	Admin will able to set a time interval of a LED color to be turned on in a specific land.
Action	The time interval will be added in the database.
Requirements	Performed by the admin.
Input	LED ID, Land ID and the time in hours.
Output	Confirmation that the data is added successfully.
Precondition	Land ID and LED ID already exists. Validate data entered. Admin Logged in.
Postcondition	The data will be added in database.
Dependencies	9.

ID	39
Function	Delete time interval
Description	Admin will be able to delete a time interval of a LED color to be turned on in a specific land.
Action	The time interval will be deleted.
Requirements	Performed by the admin.
Input	Time interval ID.
Output	Confirmation that the data is deleted successfully.
Precondition	Time interval ID is found. Admin Logged in.
Postcondition	The time interval will be deleted in the database.
Dependencies	9.

ID	40
Function	view timer interval details
Description	Admin will be able to view all time intervals in the system
Action	Time intervals will be shown.
Requirements	Performed by the admin.
Input	None.
Output	Time interval details.
Precondition	Existence of timer details in database. Admin Logged in.
Postcondition	Time intervals will be retrieved from the database.
Dependencies	9.

ID	41
Function	Update time interval
Description	Admin will be able to update the time interval of a LED color to be turned on in a specific land.
Action	The time interval is updated.
Requirements	Performed by the admin.
Input	Timer ID.
Output	Confirmation that the data is updated successfully.
Precondition	Time interval ID exists. Admin Logged in.
Postcondition	The time interval will be updated in database.
Dependencies	9.

ID	42
Function	Add land request
Description	Land owner will make a request to add a new land.
Action	The land state will be pending until the admin accepts/rejects the request.
Requirements	Performed by the landowner.
Input	Land details.
Output	Confirmation that a request has been sent to the admin.
Precondition	User Logged in. Validate data entered.
Postcondition	The request will be sent to the admin.
Dependencies	9

ID	43
Function	Land edit request
Description	The landowner will send a request with an information to be edited in a certain land that he owns.
Action	The land state will be pending until the admin accepts/rejects the update request.
Requirements	User must be a landowner.
Input	Land ID, Information to be edited.
Output	Confirmation that a request has been sent to the admin.
Precondition	Land is already found. User Logged in.
Postcondition	The request will be sent to the admin.
Dependencies	9

ID	44
Function	Land delete request
Description	The landowner will send a request to delete a certain land that he owns.
Action	The land state will be pending until the admin accepts the deletion request.
Requirements	User must be a landowner.
Input	Land ID.
Output	Confirmation that a request has been sent to the admin.
Precondition	Land is already found. User Logged in.
Postcondition	The request will be sent to the admin.
Dependencies	9

ID	45
Function	View all lands
Description	The landowner views all his lands registered in the system.
Action	All land's information will be shown.
Requirements	User must be a landowner.
Input	None.
Output	All his registered lands will be shown.
Precondition	The land owner already have a registered land in the system.
Postcondition	The lands are showed to the user.
Dependencies	9

ID	46
Function	Admin accept land add request
Description	The Admin will accept the request of the landowner to add a new land.
Action	Land will be added in the system.
Requirements	User must be an Admin.
Input	None.
Output	Confirmation that the land has been added.
Precondition	A request from the landowner must be sent first. Admin logged in.
Postcondition	Land will be added and the landowner will be notified with the acceptance.
Dependencies	9

ID	47
Function	Admin reject land add request
Description	The Admin will reject the request of the landowner to add a new land.
Action	Land won't be added in the system.
Requirements	User must be an Admin.
Input	None.
Output	None.
Precondition	A request from the landowner must be sent first. User Logged in.
Postcondition	The landowner will be notified with the rejection.
Dependencies	9.

ID	48
Function	Admin accept land edit request
Description	The Admin will accept the request of the landowner to update an information in his land.
Action	Land information will be updated in the database.
Requirements	User must be an Admin.
Input	None.
Output	Confirmation that the information has been updated.
Precondition	Land is already found. Admin logged in.
Postcondition	Update request will be accepted.
Dependencies	9.

ID	49
Function	Admin reject land edit request
Description	The Admin will reject the request of the landowner to update an information in his land.
Action	Land information won't be updated in the database.
Requirements	User must be an Admin.
Input	None.
Output	Alert that the information won't be updated.
Precondition	Land is already found. Admin logged in.
Postcondition	Update request will be rejected.
Dependencies	9.

ID	50
Function	Admin accept land delete request
Description	The Admin will accept the request of the landowner to delete his registered land.
Action	Land information will be deleted from the database.
Requirements	User must be an Admin.
Input	None.
Output	Confirmation that the land has been deleted.
Precondition	Land is already found. Admin logged in.
Postcondition	Delete request will be accepted.
Dependencies	9.

ID	51
Function	Receive notification
Description	Landowner will receive an email in every growth stage or if there is a disease in the plant.
Action	None.
Requirements	User must be a Landowner.
Input	None.
Output	Land ID, Date and time and Content of the notification message.
Precondition	The growth stage has changed or a disease has been detected. User Logged in.
Postcondition	None.
Dependencies	9.

ID	52
Function	Send notification
Description	System will send an email in every growth stage or if there is a disease in the plant.
Action	Email will be sent.
Requirements	Performed by system.
Input	Land ID, Date and time and Content of the notification message.
Output	Confirmation that the email has been sent.
Precondition	Land is already found the growth stage has changed or a disease has been detected. User Logged in.
Postcondition	Email will be sent.
Dependencies	9.

ID	53
Function	User Edit
Description	This function enables the user to edit his info.
Action	Information will be edited.
Requirements	Performed by a user.
Input	Edited information.
Output	Return true or false if the user was edited successfully or not.
Precondition	Edited information is valid to be added and the system must be connected to the database. User logged in
Postcondition	User will be edited.
Dependencies	9.

3.4 Interface Requirements

3.4.1 User Interfaces

Our system's user interface is easy to use and very usable. There are two main user-types who can login in our system, an admin or a landowner. The system shows the user duties in the navigation menu according to the user-type logged in.

3.4.2 Hardware Interfaces

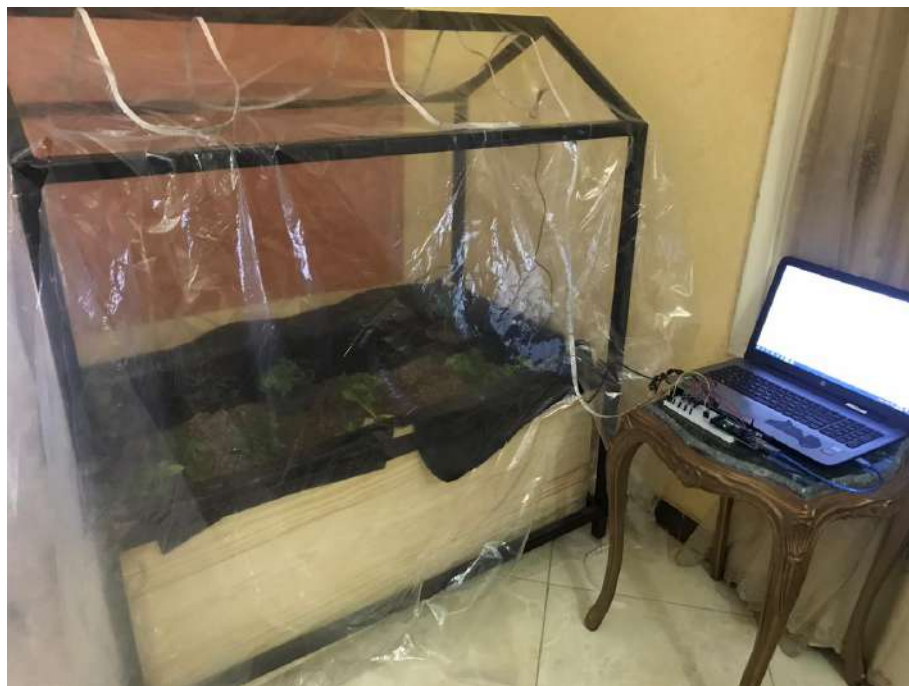


Figure 3.2: Laptop

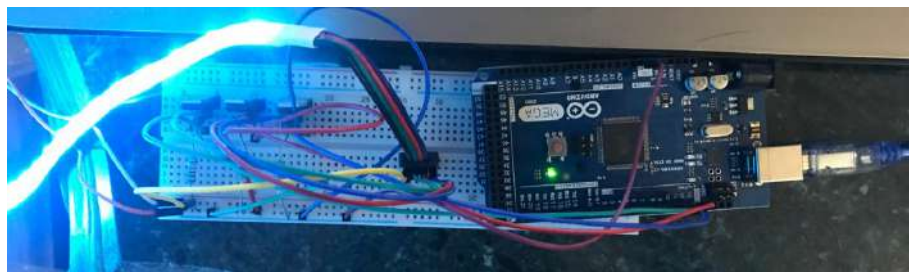


Figure 3.3: Arduino



Figure 3.4: Camera

3.4.3 Communications Interfaces

N/A

3.4.4 Software Interfaces

Arduino that connects between sensors and LED lights with our system.

3.5 Performance Requirements

For monitoring the greenhouse, the system shall be able to process at least 7 frames every 5 minutes. The system also must be able to handle large training dataset.

3.6 Design Constraints

3.6.1 Standards Compliance

64-bit operating system, x64 based processor.

3.6.2 Hardware Limitations

1. Arduino mustn't be nano.
2. Camera mustn't be less than 8 megapixel

3.7 Other non-functional attributes

3.7.1 Security

The password of the user is encrypted before being saved in the database.

3.7.2 Reliability

The system needs electricity all the time, so if there is a power cut the generators will work automatically.

3.7.3 Maintainability

The code is implemented by using MVC, Singleton and observer design patterns so the system could be improved by different developers easily.

3.7.4 Portability

It is a web-based system so it can be deployed on any device.

3.7.5 Usability

The system functionalities doesn't need time to be learned by the user.

3.8 Preliminary Object-Oriented Domain Analysis

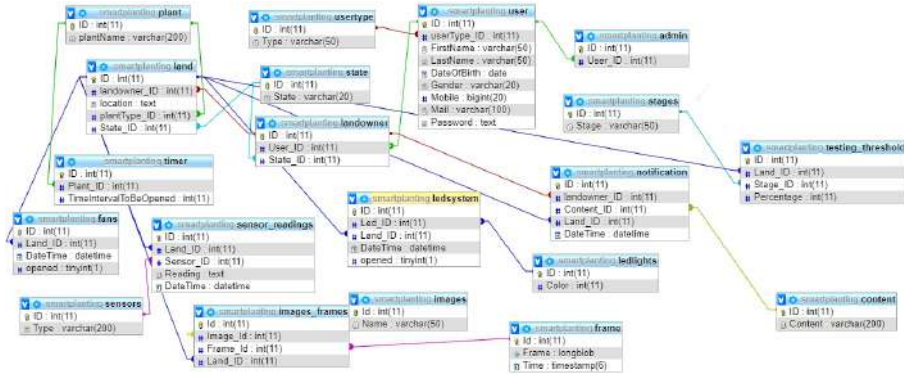


Figure 3.5: Data Base schema

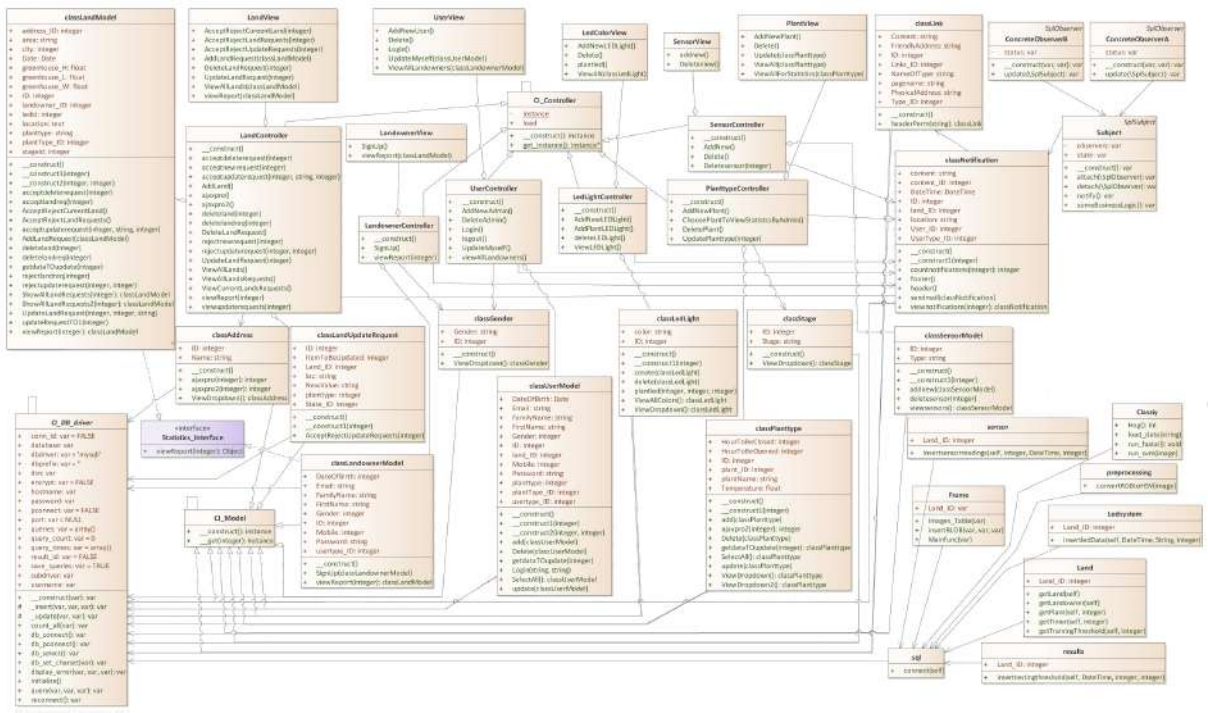


Figure 3.6: Class Diagram

3.8.1 Inheritance Relationships

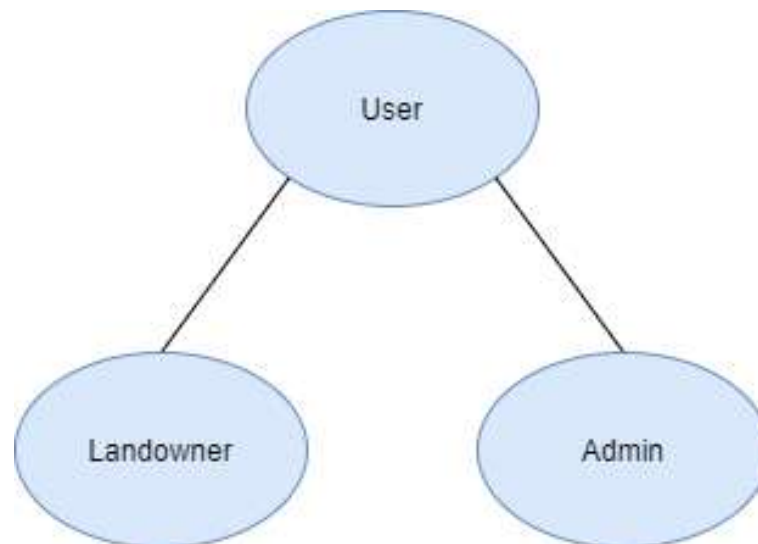


Figure 3.7: Inheritance Relationship

3.8.2 Class descriptions

Class Name	User_Model
Super Class	None.
Sub Class	Admin_Model, Landowner_Model, Database_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(Landowner_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(); ViewSensors(); 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_Model 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()

3.9 Operational Scenarios

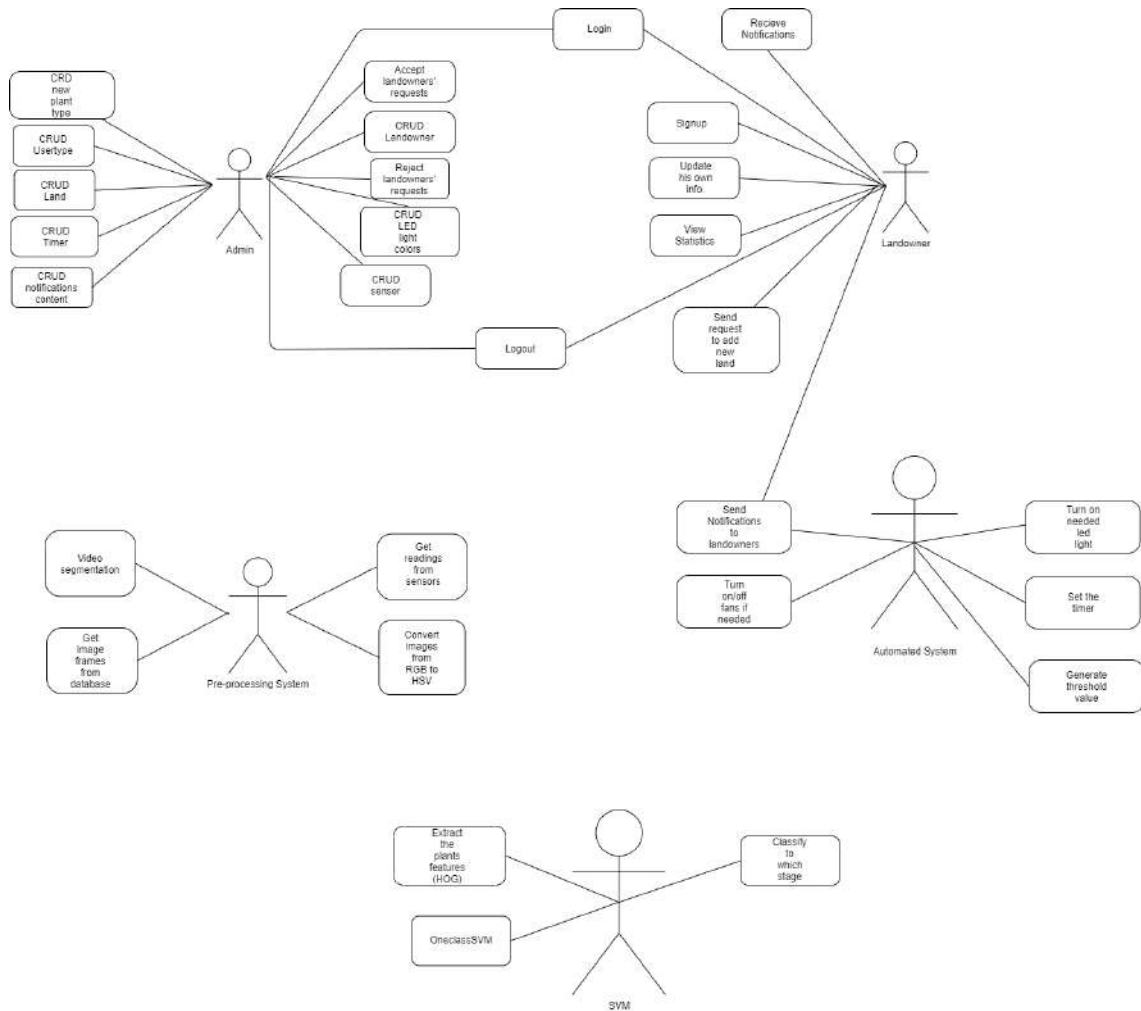


Figure 3.8: Usecase

Function	Add New Plant Type.
Actors	Admins.
Description	Admins can add new plant type to the system.
Data	Name of the plant type.
Response	Name of the plant added to the system.
Comments	Admin must be signed in.

Function	Delete Plant Type.
Actors	Admins.
Description	Admins can delete plant type to the system.
Data	Search on the name of the plant type.
Response	Name of the plant deleted from the system.
Comments	Admin must be signed in.

Function	Add New User Type.
Actors	Admins.
Description	Admins can add new user type to the system.
Data	Information about the new user.
Response	The new user added to the system.
Comments	Admin must be signed in.

Function	Update User Type.
Actors	Admins.
Description	Admins can update user type's information in the system.
Data	Information about the new user.
Response	The user type's info. is updated in the system.
Comments	Admin must be signed in.

Function	Delete User Type.
Actors	Admins.
Description	Admins can delete user type from the system.
Data	Information about the new user.
Response	The user type's info. are updated in the system.
Comments	Admin must be signed in.

Function	Add Land.
Actors	Admins.
Description	Admins can add new land to the system.
Data	Information about the new land.
Response	The new land is added to the system.
Comments	Admin must be signed in.

Function	Delete Land.
Actors	Admins.
Description	Admins can delete lands from the system.
Data	Search on the name of the land.
Response	The land deleted from the system.
Comments	Admin must be signed in.

Function	Add Timer.
Actors	Admins.
Description	Admins can Add timer in the system.
Data	Time needed for each plant.
Response	The timer added to the system.
Comments	Admin must be signed in.

Function	Update Timer.
Actors	Admins.
Description	Admins can update timer in the system.
Data	Time needed for each plant.
Response	The timer updated in the system.
Comments	Admin must be signed in.

Function	Delete Timer.
Actors	Admins.
Description	Admins can Delete timer in the system.
Data	Time needed for each plant.
Response	The timer deleted from the system.
Comments	Admin must be signed in.

Function	Delete LED Light Colors
Actors	Admins.
Description	Admins can delete led light colors from the system.
Data	Name of the new color.
Response	The name of the LED light deleted from the system.
Comments	Admin must be signed in.

Function	Add Notification's Content
Actors	Admins.
Description	Admins can add new notification content to the system.
Data	The content of the notification he would like to send.
Response	The notification's content added to the system.
Comments	Admin must be signed in.

Function	Update Notification's Content
Actors	Admins.
Description	Admins can update notification content to the system.
Data	The content of the notification he would like to send.
Response	The notification content updated in the system.
Comments	Admin must be signed in.

Function	Delete Notification's Content
Actors	Admins.
Description	Admins can Delete notification content to the system.
Data	The content of the notification he would like to send.
Response	The notification content deleted from the system.
Comments	Admin must be signed in.

Function	Add New Sensor.
Actors	Admins.
Description	Admins can add new sensor to the system.
Data	The name of the sensor and its description.
Response	The name of the sensor is added to the system.
Comments	Admin must be signed in.

Function	Delete Sensor.
Actors	Admins.
Description	Admins can delete sensor from the system.
Data	The name of the sensor and its description.
Response	The name of the sensor is deleted from the system.
Comments	Admin must be signed in.

Function	Accept Landowner's requests.
Actors	Admins.
Description	Admins can accept landowner's requests to apply the system on his land.
Data	The location of the land and its measurements.
Response	The landowner's request is accepted.
Comments	Admin must be signed in.

Function	Reject Landowners' Requests
Actors	Admins.
Description	Admins can reject landowner's requests to apply the system on his land.
Data	The location of the land and its measurements.
Response	The landowner's request is rejected.
Comments	Admin must be signed in.

Function	Create Landowner's Account.
Actors	Admins.
Description	Admins can create landowner's account
Data	The information about the landowner and his land.
Response	The landowner's account is created.
Comments	Admin must be signed in.

Function	Update Landowner's Account.
Actors	Admins.
Description	Admins can update landowner's information.
Data	The information about the landowner and his land.
Response	The landowner's account is updated.
Comments	Admin must be signed in.

Function	Delete Landowner's Account.
Actors	Admins.
Description	Admins can delete landowner's account.
Data	The information about the landowner and his land.
Response	The landowner's account is deleted.
Comments	Admin must be signed in.

Function	Login.
Actors	Admins and Landowners.
Description	Admins and landowners are being able to login to the system.
Data	Username and password of the user.
Response	The admin/landowner is logged in if the required information <u>are</u> correct.
Comments	Admin/landowner must exist on the system.

Function	Logout.
Actors	Admins and Landowners.
Description	Admins and landowners are being able to log out of the system.
Data	None.
Response	The admin/landowner is logged out of the system.
Comments	Admin / landowner must be signed in.

Function	Sign-up.
Actors	Landowners.
Description	Landowners are being able to sign-up for the system.
Data	Some information about the landowner.
Response	The landowner is signed up for the system.
Comments	Landowner mustn't be exist on the <u>system</u> .

Function	Receive Notification.
Actors	Landowners.
Description	Landowners are being able to receive notifications from the system.
Data	Notifications content.
Response	The landowner received the notification from the system.
Comments	Landowner must be signed in.

Function	View Statistics.
Actors	Landowners.
Description	Landowners are being able to view statistics of the productivity of his land.
Data	Bar charts.
Response	The landowner can view statistics about his land.
Comments	Landowner must be signed in.

Function	Update his own information.
Actors	Landowners.
Description	Landowners are being able to update his own information that <u>are</u> saved in the system.
Data	Information about the landowner.
Response	The landowner's information are updated in the system.
Comments	Landowner must be signed in.

Function	Send Request To Add New Land
Actors	Landowners.
Description	Landowners are being able to send requests to add another land to apply the system on it.
Data	Information about the landowner.
Response	The landowner's request will be pending until the admin accepts it.
Comments	Landowner must be signed in.

Function	Extract The Plants Features.
Actors	Classification System.
Description	The classification system will be able to extract the plant features using masking and feature extraction (HOG).
Data	Frames that are captured from the video cameras.
Response	The needed features in plants will be extracted clearly.
Comments	Make sure about the availability of image frames.

Function	Classify To Which Stage.
Actors	Classification System.
Description	The classification system will be able to classify which growth stage the plants reach using SVM.OneClassSVM.
Data	Frames that are captured from the video cameras.
Response	The plants' growth stage will be classified accurately.
Comments	Make sure about the availability of image frames.

Function	Get Image Frames.
Actors	Pre-processing system.
Description	The pre-processing system will be able to get the image frames from the database.
Data	Image frames that are captured from the video cameras.
Response	The pre-processing system got the image frames to send it to the classification system.
Comments	Make sure about the availability of image frames in database.

Function	Get Readings From Sensors.
Actors	Pre-processing system.
Description	The pre-processing system will be able to get readings from sensors to make a suitable environment for plants.
Data	Readings from sensors that are saved in the database.
Response	The pre-processing system got readings from sensors to use them then.
Comments	Make sure about the availability of readings of sensors in the database.

Function	Convert Image to HSV
Actors	Pre-processing system.
Description	The pre-processing system will be able to convert images from RGB to HSV to use it in the classification system.
Data	RGB test images.
Response	The image is converted.
Comments	None.

Function	Send Notifications To Landowners.
Actors	Automated system.
Description	The automated system will be able to send notifications automatically when the crops are ready to be harvested or if there is any abnormality or disease appeared on the plants.
Data	Notification content.
Response	Notifications are sent automatically.
Comments	None.

Function	Turn on/off fans
Actors	Automated system.
Description	The automated system will be able to turn on/off fans automatically if needed to provide a suitable environment for the plants.
Data	Readings from DHT22 sensor.
Response	Fans will be turned on/off automatically.
Comments	None.

Function	Turn on/off LED lights
Actors	Automated system.
Description	The automated system will be able to turn on/off the LED lights according to the plant's needs automatically.
Data	Image frames from database.
Response	LED lights will be turned on/off automatically.
Comments	None.

Function	Generate Threshold Value
Actors	Automated system.
Description	The automated system will be able to generate the threshold value to be used in the testing stage.
Data	Image frames from database.
Response	The threshold value will be calculated accurately.
Comments	The dataset must be divided into training and testing images.

Function	Set Timer
Actors	Automated system.
Description	The automated system will be able to set the timer automatically after getting it from the database according to the plants' name and type.
Data	Time for each plant type.
Response	The automated system will set the timer automatically.
Comments	None.

3.10 Appendices

3.10.1 Definitions, Acronyms, Abbreviations

Table 3.1: Definitions, Acronyms and Abbreviations

Term	Definition
UI:	User interface is the space where interactions between humans and machines occur.
Hue-Saturation-Value (HSV)	is a color model that is used to place the RGB (Red-Green-Blue) color model in graphics. Is used to make it easily color adjustments.

Chapter 4

Software Design Document

4.1 Introduction

4.1.1 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.

4.1.2 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 Preprocessing to enhance the image, con-

verting 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.

4.1.3 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.

4.1.4 Definitions and Acronyms

Table 4.1: 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

4.2 System Architecture

4.2.1 Architectural Design

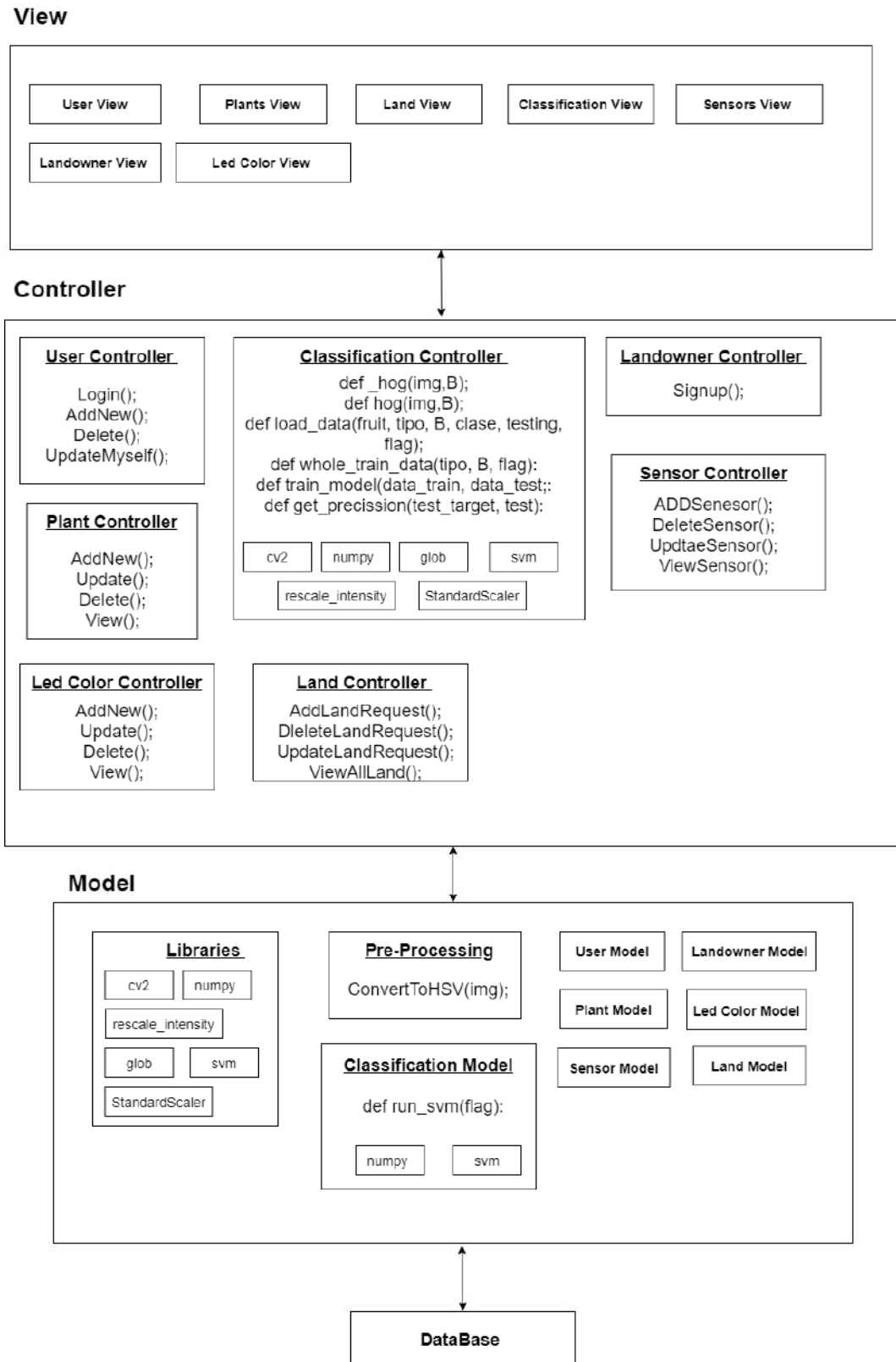


Figure 4.1: Architectural Design

4.2.1.1 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 differentiate 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 an 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 has more views/pages for him/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. Land view, appears for both the admin and the landowner. First the admin, it allows them to view all lands registered in the system, view their 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 lands and make some requests on them, as well as they could make a new request about adding a new land to the system. Sensor view, is for allowing the admin to add, delete and view all kinds of sensors being used inside the greenhouses registered inside the system. Plant view is for the admin to add/delete plants.

4.2.1.2 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 forwarded 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, Led Color Controller, Sensor Controller and Land Controller all of the previous Controllers are responsible for actions related to them as adding, deleting, viewing and updating.

4.2.1.3 Model

preprocessing:

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, Plant, Led color, Sensor and Landowner all of these models are responsible to talk to the database and get all the needed data to be viewed for the user correctly.

4.2.2 Decomposition Description

4.2.2.1 Class Diagram

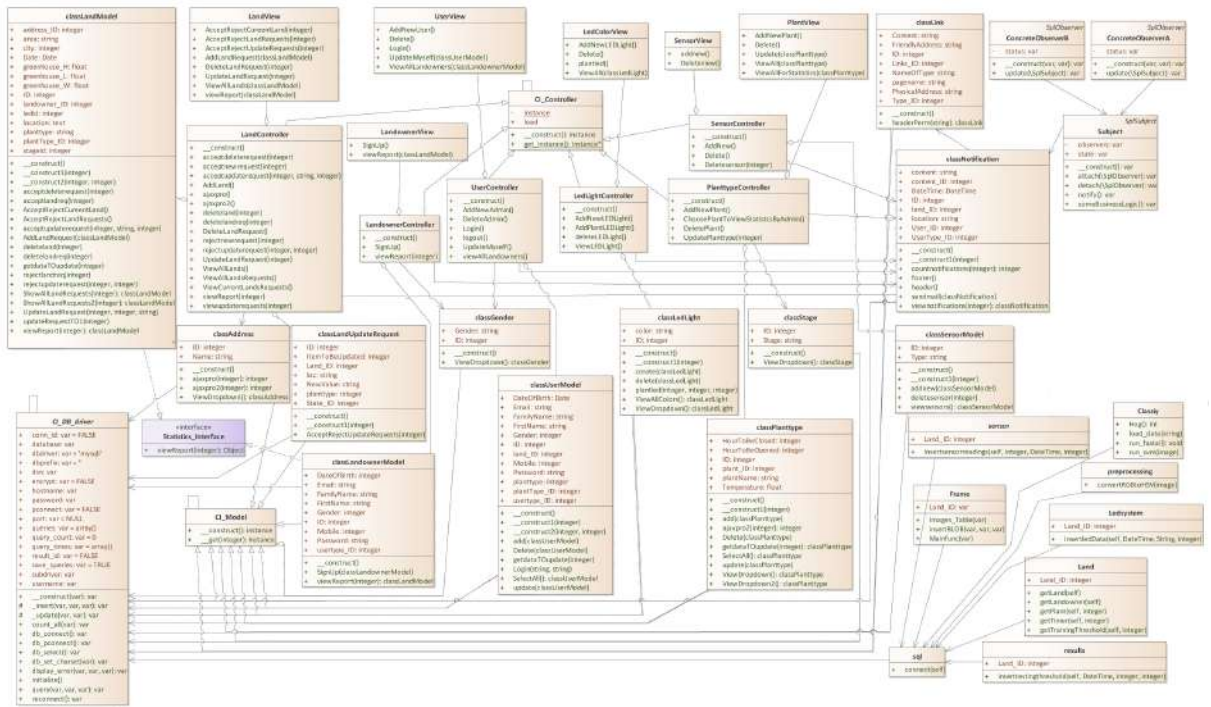


Figure 4.2: Class Diagram

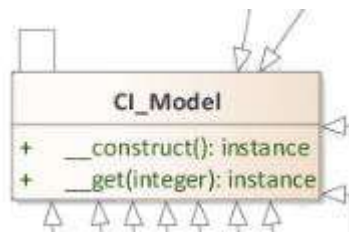


Figure 4.3: Singleton design pattern

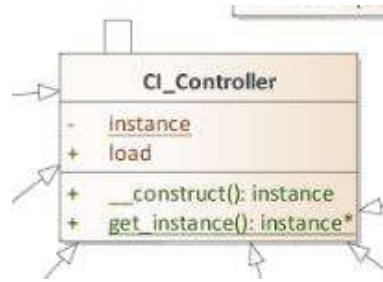


Figure 4.4: Singleton design pattern

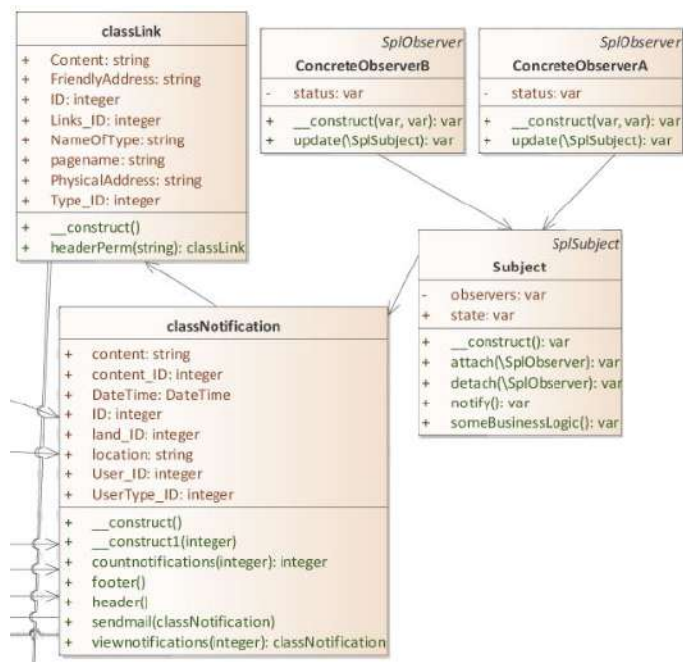


Figure 4.5: Observer design pattern

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_Model 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	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,DateTime,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()

Class Name	Land
Super Class	None.
Sub Class	None.
Purpose	This class is used to represent the land owned by which landowner in the system.
Collaborations	This class associate with sql class
Attributes	Land_ID.
Operations	None.

Class Name	Results
Super Class	None.
Sub Class	None.
Purpose	This class is used to save the results of the threshold value of each land
Collaborations	This class associate with sql class
Attributes	Land_ID.
Operations	None.

Class Name	Address
Super Class	LandOwnerController
Sub Class	None
Purpose	This class is used to get the address of the lands
Collaborations	None
Attributes	ID, Name
Operations	Ajaxpro() Ajaxpro2() viewDropdown()

Class Name	classLandModel
Super Class	None.
Sub Class	Landowner_Model.
Purpose	This class is used to allow the landowner make requests for the land and allow the user to accept/reject
Collaborations	This class aggregates with LandController Realization connection with StatisticsInterface
Attributes	Address_ID, area,city,date,GreenHouse_H,GreenHouse_W, GreenHouse_L,ID,Landowner_ID,ledID,location,PlantType,PlantType_ID,Stage_ID
Operations	Acceptdeletelandrequest(),Acceptlandrequest() Acceptrejectcurrentland(),Acceptupdaterequest() Addlandrequest(),deletelandrequest() deleteland(),getdatatouupdate() rejectlandrequest(),ShowAllLandRequests () ShowAllLandRequests2 (),UpdateRequestTO1() ViewReport(classLandModel)

4.2.2.2 Activity Diagram

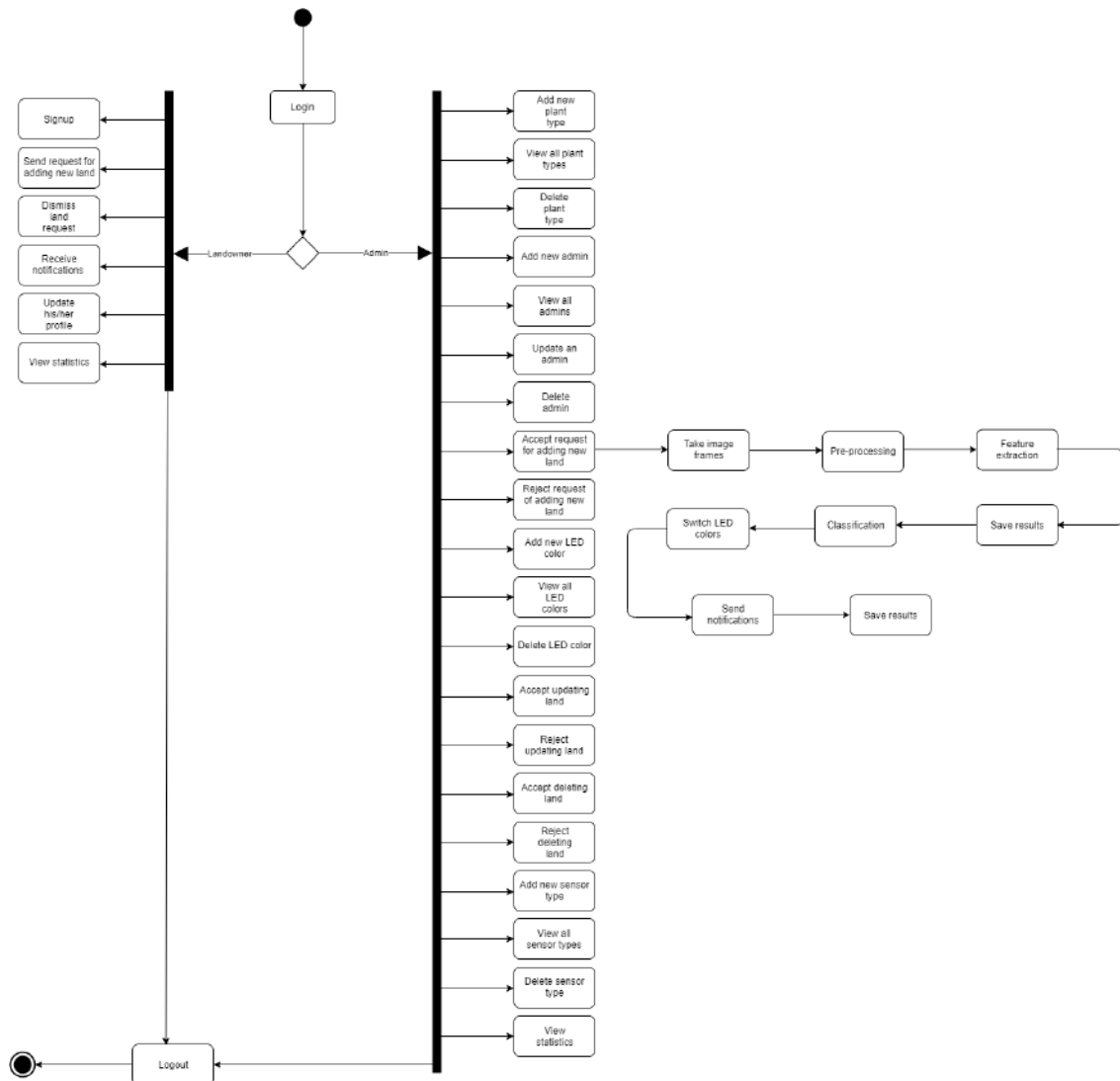


Figure 4.7: Activity Diagram

4.2.2.3 Sequence Diagram

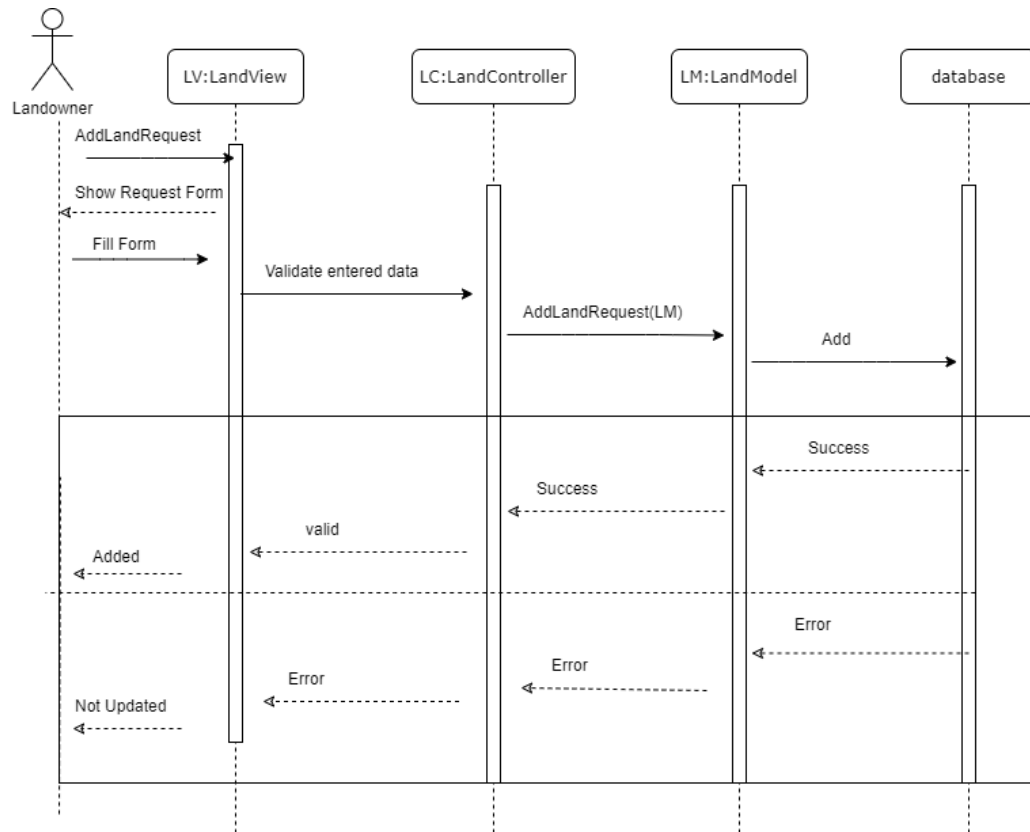


Figure 4.8: Landowner adds new request

The sequence diagram in Figure 4.8 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.

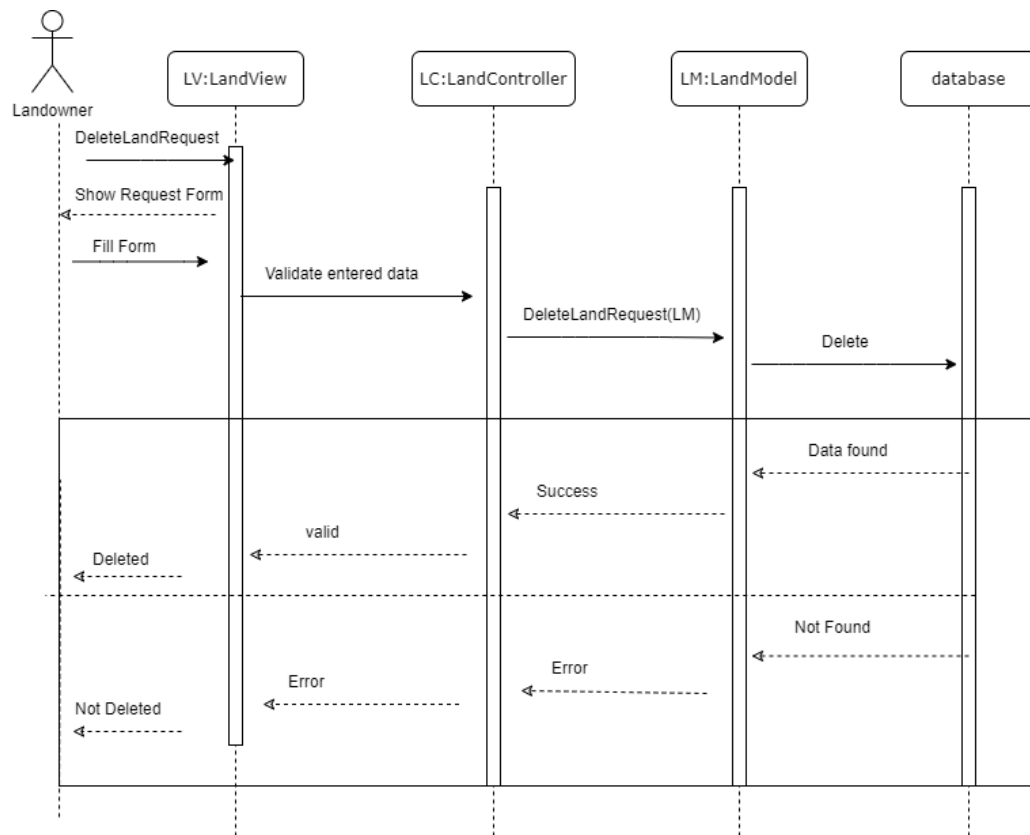


Figure 4.9: Landowner requests to delete the land

The sequence diagram in Figure 4.9 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.

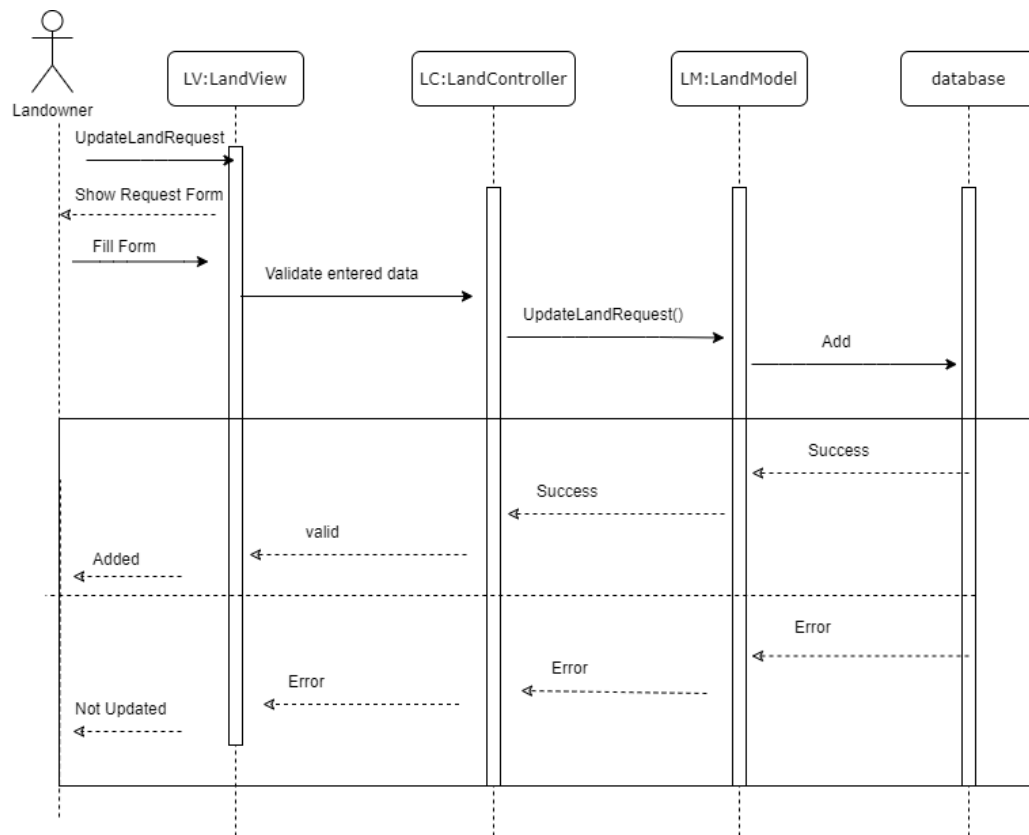


Figure 4.10: Landowner requests to update the land

The sequence diagram in Figure 4.10 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.

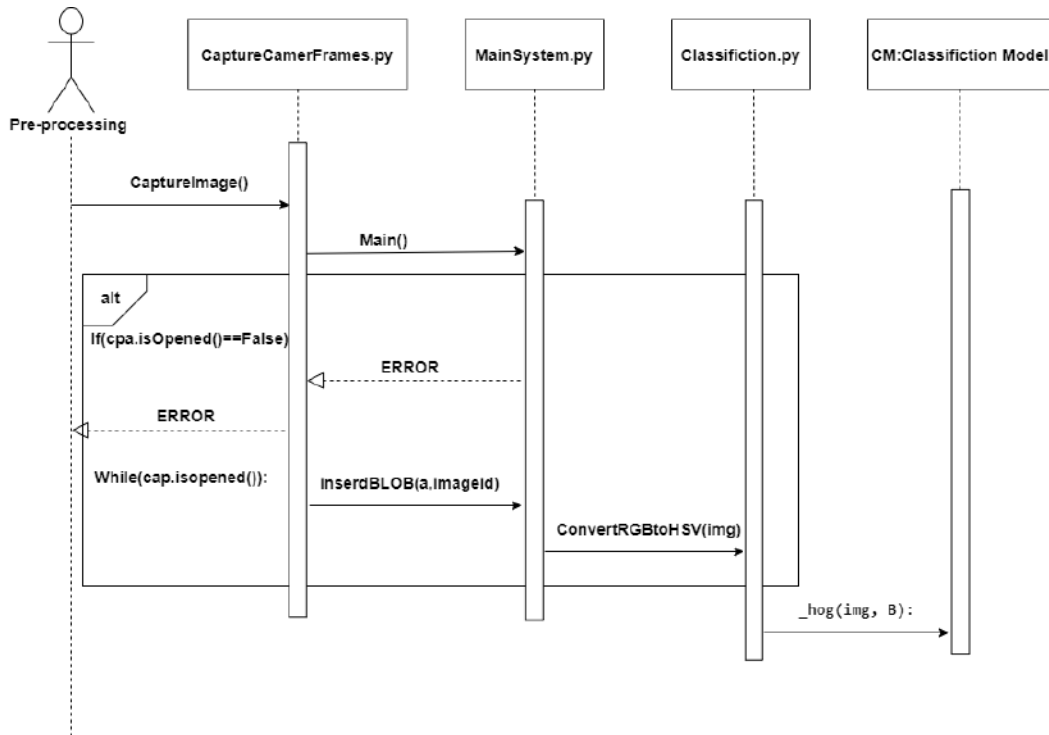


Figure 4.11: Preprocessing

The sequence diagram in Figure 4.11 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 insertBLOB(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)

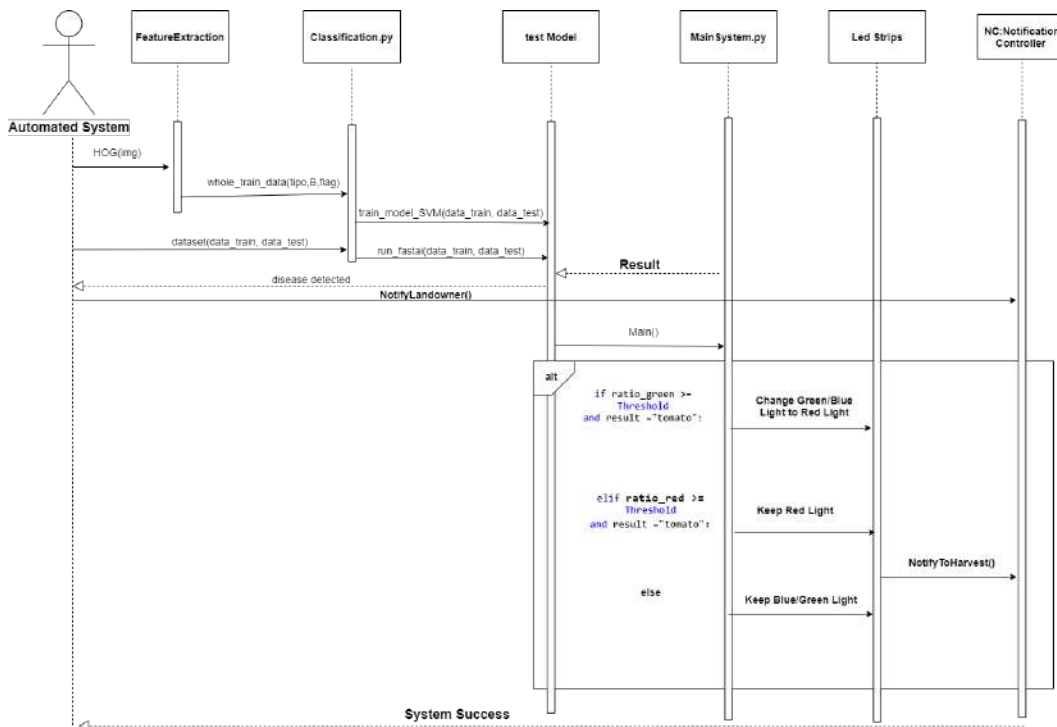


Figure 4.12: Automated and Classifier

The sequence diagram in Figure 4.12 explains how our system is fully automated system, it's divided into two parts. First we start by feature extraction this is done through the function `-hOG(img)`, moving to the Classification python class through the function `whole_train_data(ripo,B,flag)`, then the test Model through the `train_model_SVM(data_train,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 shown in the above diagram. Ending with a system success statement. Second the `fastai(data_train,data_test)` function is passed to the classification class to detect if there is any detected diseases. If it detected a disease a notification will be sent to the landowner.

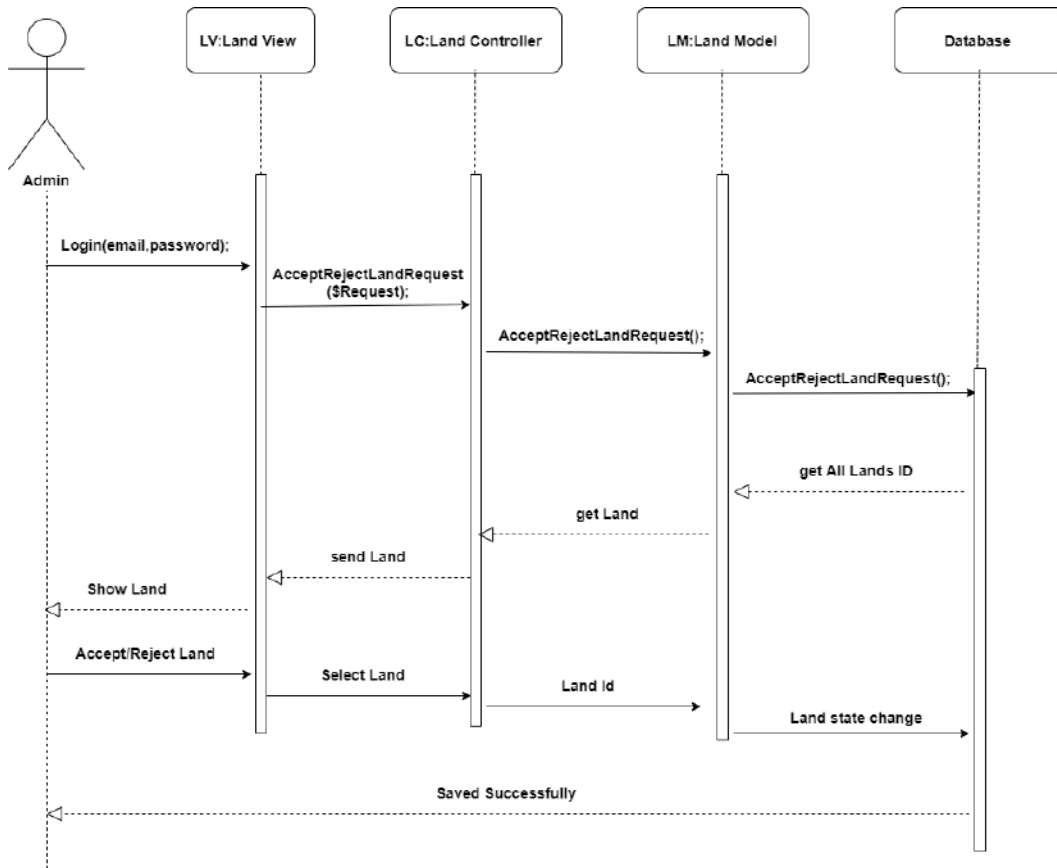


Figure 4.13: AcceptRejectLand

The sequence diagram in Figure 4.13 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.

4.2.3 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.

4.3 Data Design

4.3.1 Data Description

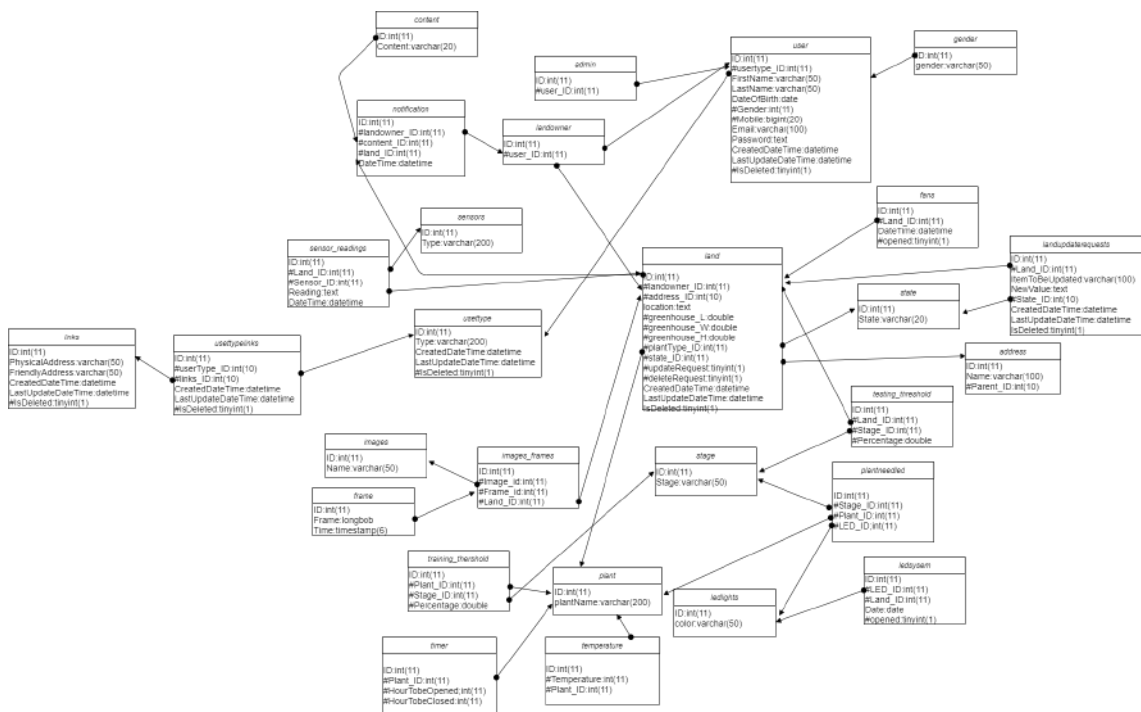


Figure 4.14: Database Schema

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 and IsDeleted.

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, landowner-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 the 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 application.

stages: This table contains: ID and Stage.

plant: This table contains: ID and plantName.

temperature: This table contains: ID, temperature and plantID. This table is used to save the suitable temperature of each plant.

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.

4.3.2 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.

4.4 Component Design

4.4.1 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, the user will be notified to turn the fan 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. Preprocessing and processing are operated on these frames as Enhancements, Masking and Feature extraction.

4.4.2 Preprocessing

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.

4.4.3 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 4.15.



Figure 4.15: Before and after the HSV masking effect on tomatoes' testing image

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

$$\begin{aligned}
 gx &= \text{ones}((8, 8)) * \cos(\pi/4) \\
 gy &= \text{ones}((8, 8)) * \sin(\pi/4) \\
 h &= \text{HOG}_{8 \times 8}(gx, gy)
 \end{aligned} \tag{4.1}$$

First, the Image will be converted into grey value image. Then get the HOG value (h) in 8x8 cell using equation (4.1), where the gx is the x-directional derivative, gy the y-directional derivative.

4.4.4 Classification

In this phase, it is divided into two parts. First detecting the tomato where 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 [15]. 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. 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.

$$f(x) = \text{sgn}\left(\sum_{i=1}^n \alpha_i K(x, x_i) - p\right) \tag{4.2}$$

This method in equation (4.2) creates a hyper plane characterized by p which separates all the data points from the origin, α_i is the Lagrange multipliers computed for each distance and $K(x, x_i)$ is the Kernel.

Second detecting 9 different types of plant diseases which are: Bacterial spot, Early blight, Late blight, Leaf mold, Septoria leaf spot, Spider mites, Target spot, Yellow curl virus and Mosaic virus. Fastai deep learning library is used. Flow chart for the proposed system is shown in Fig.4

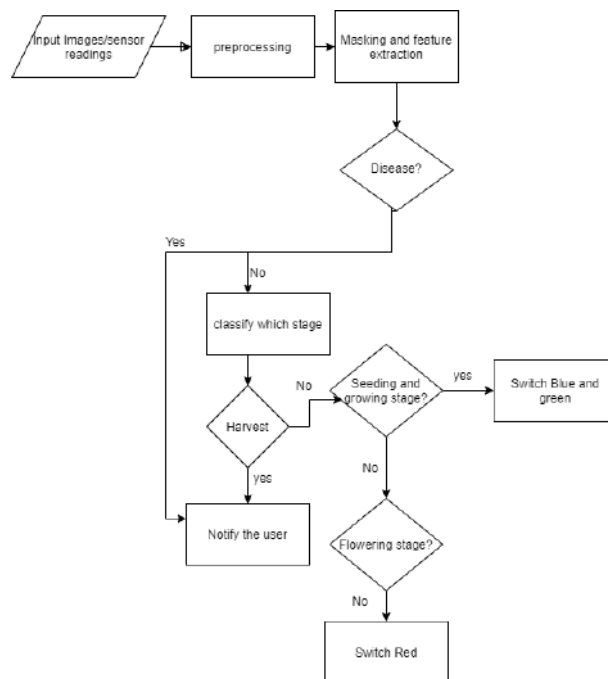


Figure 4.16: flow chart

Deep learning methods automatically select the suitable features and extract them to be used which any instructions given by the user. Since deep learning is one of the machine learning techniques, it is used frequently in computer vision field [16]. Fastai is a deep learning model which is on the top of pytorch, it is recommended because of its simplicity ,ease of use for its implemented functions and faster than other models ,also giving higher accuracy. The main methods used to classify the disease are:

1. “databunch” identifies the needed batch size which is no. of training samples from the training dataset utilized in one iteration to make prediction then calculate error.

2. “Cnn-learner” method is used to train the model on a very large dataset already identified before and then adapt it to the given dataset.
3. “lr_find” is used to get the suitable range of learning rates for the model
4. “recorder.plot” is used to get the exact learning rate which give the least loss.
5. “fit_one_cylcle” to specify the number of epochs which is passing through the training dataset.
6. “get_preds” to get predictions on the validation dataset

4.5 Human Interface Design

4.5.1 Overview of User Interface

Our system Smart Planting user interface is easy to be used and implemented using Codeigniter php framework. 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 subsection.

4.5.2 Screen Images

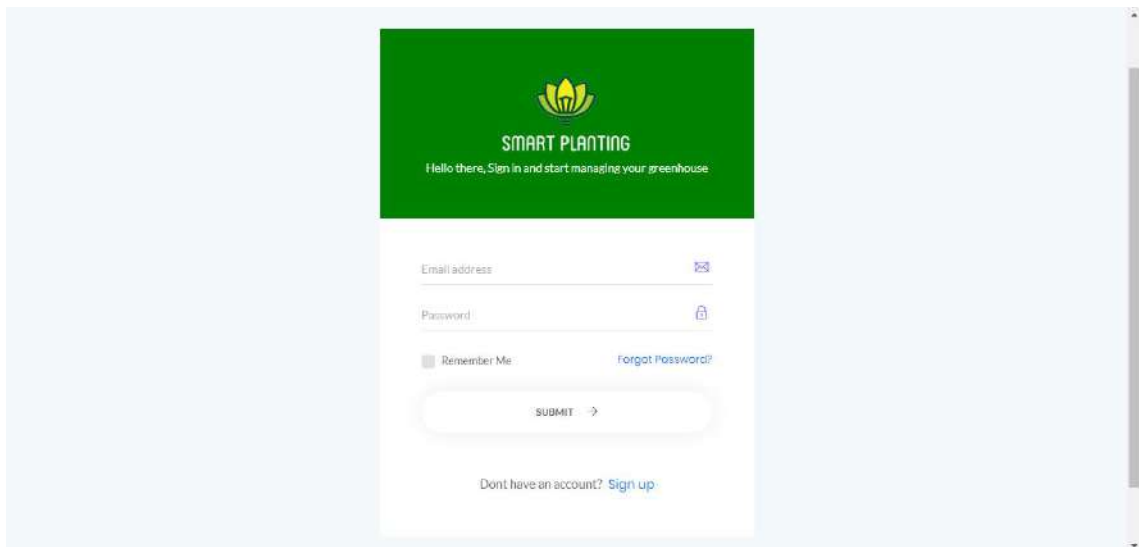
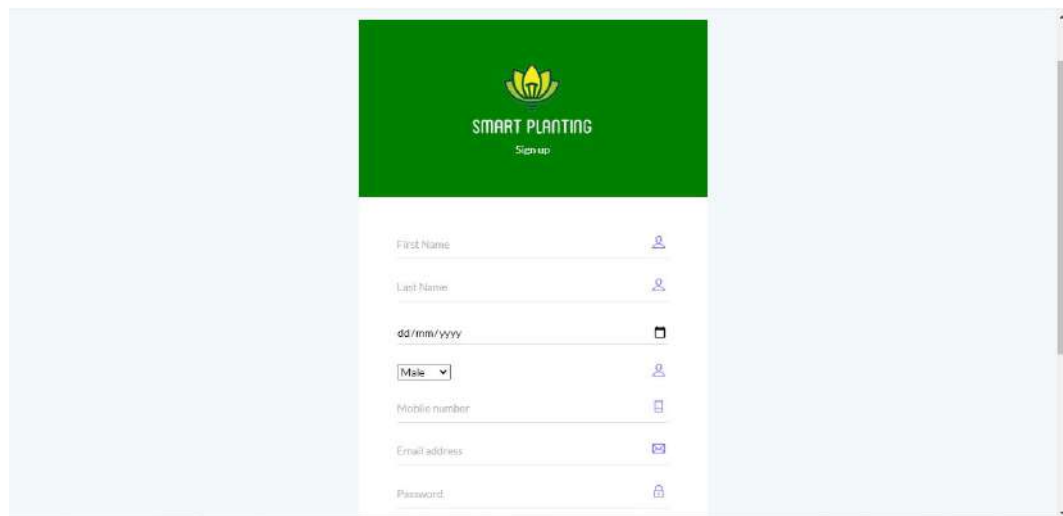
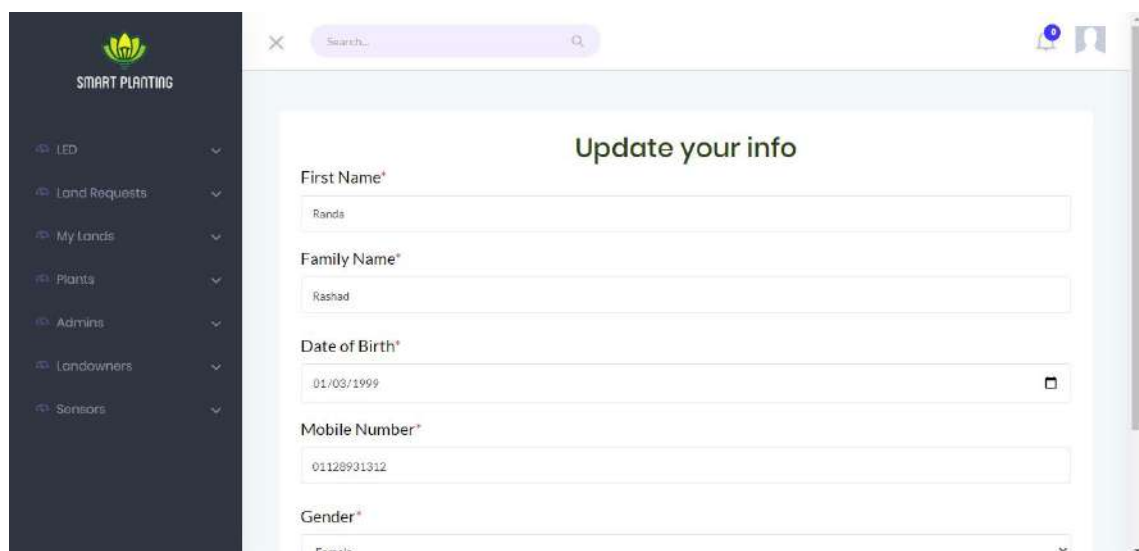


Figure 4.17: Login-Page



The screenshot shows a sign-up form for SMART PLANTING. At the top, there is a green header with the SMART PLANTING logo and the text "Sign up". Below the header, the form contains several input fields: First Name, Last Name, Date of Birth (dd/mm/yyyy), Gender (a dropdown menu with "Male" selected), Mobile number, Email address, and Password. Each field has a small icon to its right indicating the type of input required.

Figure 4.18: SignUp



The screenshot shows the "Update your info" form for an admin user in the SMART PLANTING system. The form is displayed in a web browser window with a search bar at the top. On the left, there is a dark sidebar with the SMART PLANTING logo and a list of menu items: LED, Land Requests, My Lands, Plants, Admins, Landowners, and Sensors. The main content area has a light blue background and contains the following fields: First Name* (Randa), Family Name* (Rashad), Date of Birth* (01/03/1999), Mobile Number* (01126931312), and Gender* (Female). Each field is marked with an asterisk to indicate it is required.

Figure 4.19: Admin Updating his/her INFO

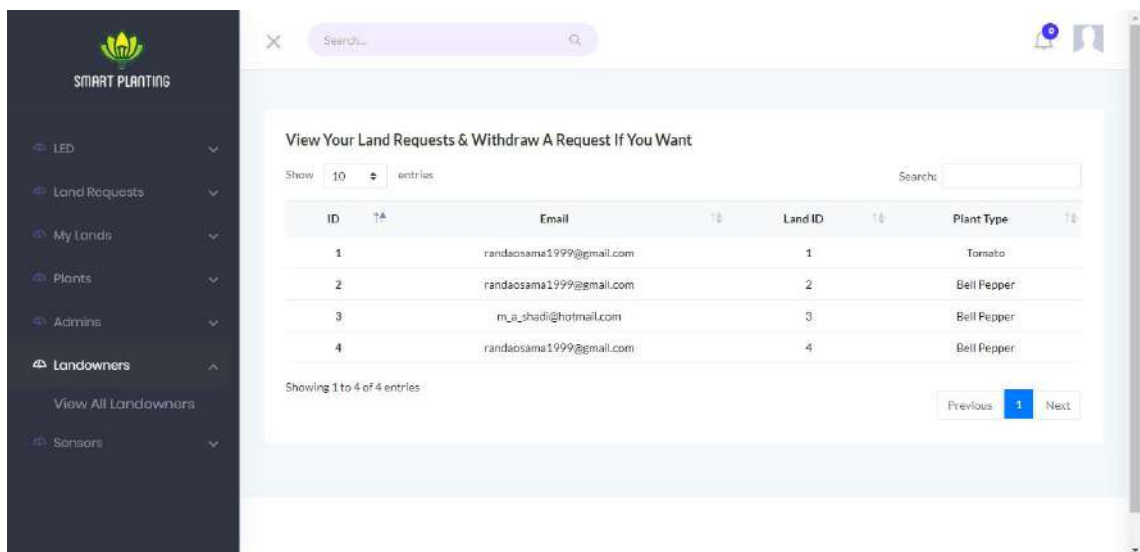


Figure 4.20: Admin Viewing All Landowners

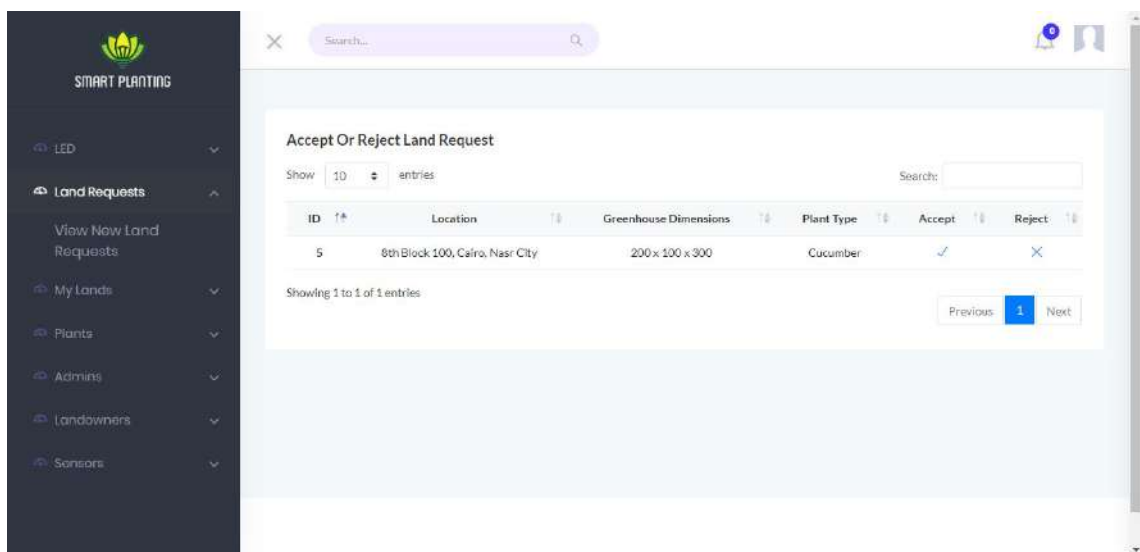


Figure 4.21: Admin Viewing All landowner's requests for adding new greenhouse

The screenshot shows the 'Update Plant' form in the SMART PLANTING admin interface. The form is titled 'Update Plant' and contains three input fields: 'Hours to be opened*' with the value '14:00:00', 'Hours to be closed*' with the value '23:30:00', and 'The suitable temperature*' with the value '34'. There is an 'Add' button at the bottom of the form. The left sidebar shows the navigation menu with 'Plants' selected.

Figure 4.22: Admin update Plants

The screenshot shows the 'View Plants Statistics' table in the SMART PLANTING admin interface. The table has 7 columns: ID, Plant Type, Temperature, Hour LED To Be Opened, Hour LED To Be Closed, Update, and View Statistics. There are 3 entries in the table. The table is titled 'View Plants Statistics' and has a search bar and a 'Show 10 entries' dropdown. The table is displayed with 3 entries.

ID	Plant Type	Temperature	Hour LED To Be Opened	Hour LED To Be Closed	Update	View Statistics
1	Tomato	34	14:00:00	23:30:00		
2	Bell Pepper	35	20:00:01	25:00:17		
3	Cucumber	36	20:00:01	25:00:17		

Showing 1 to 3 of 3 entries

Previous 1 Next

Figure 4.23: Admin View All Plant

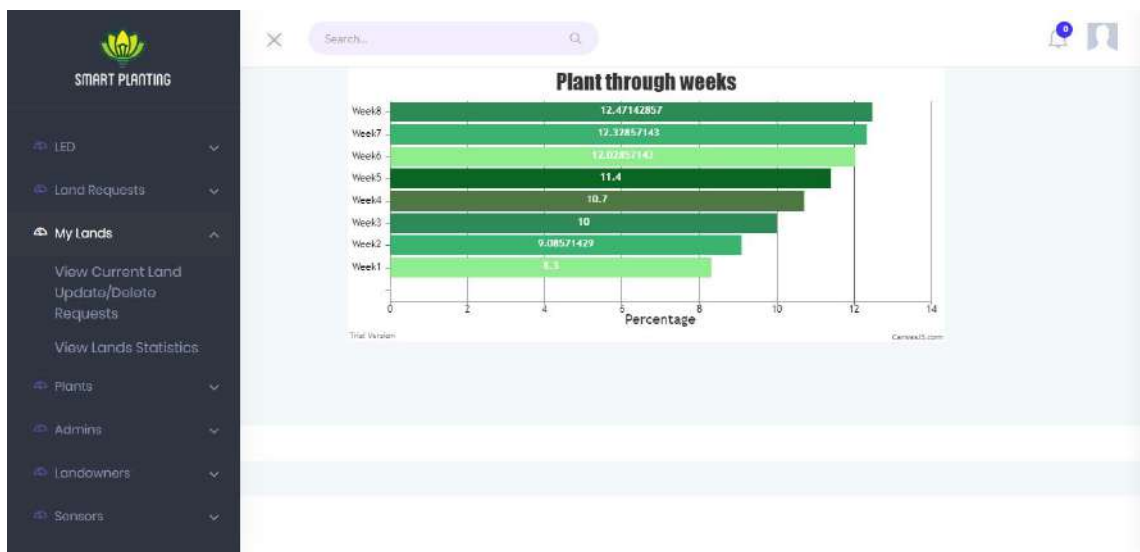


Figure 4.24: Admin view statistics

Add new Admin

First Name*

Family Name*

Mobile*

Email*

Password*

Figure 4.25: Admin adding a new admin

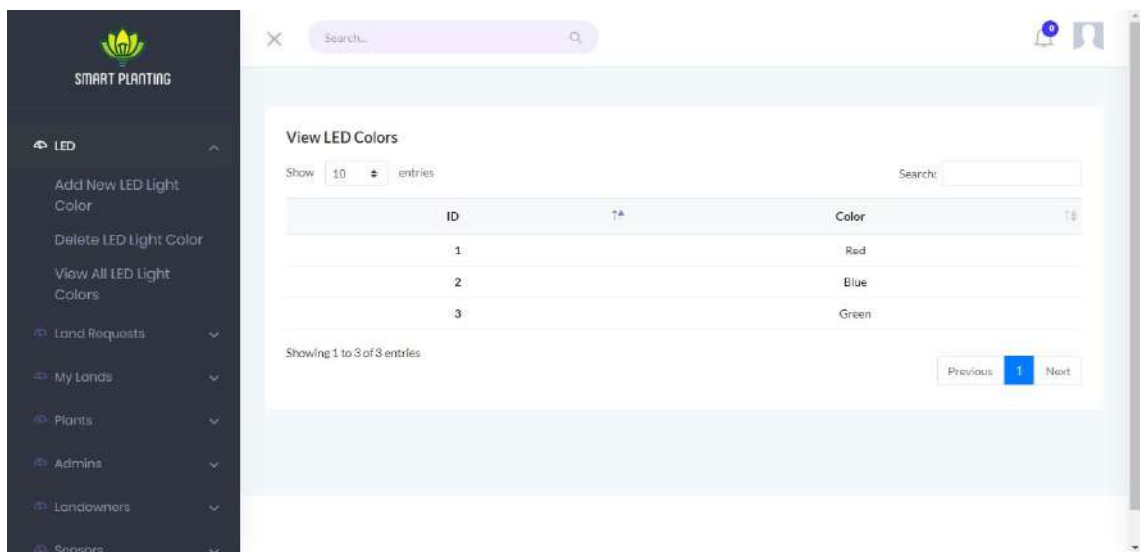


Figure 4.26: Led light colors

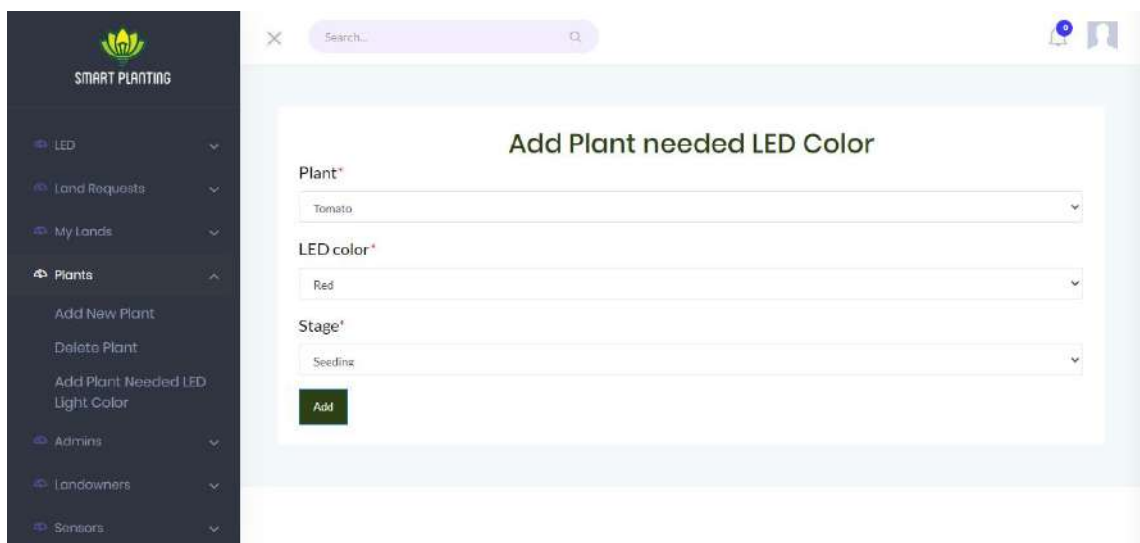


Figure 4.27: Admin assign led light color to each plant is each stage

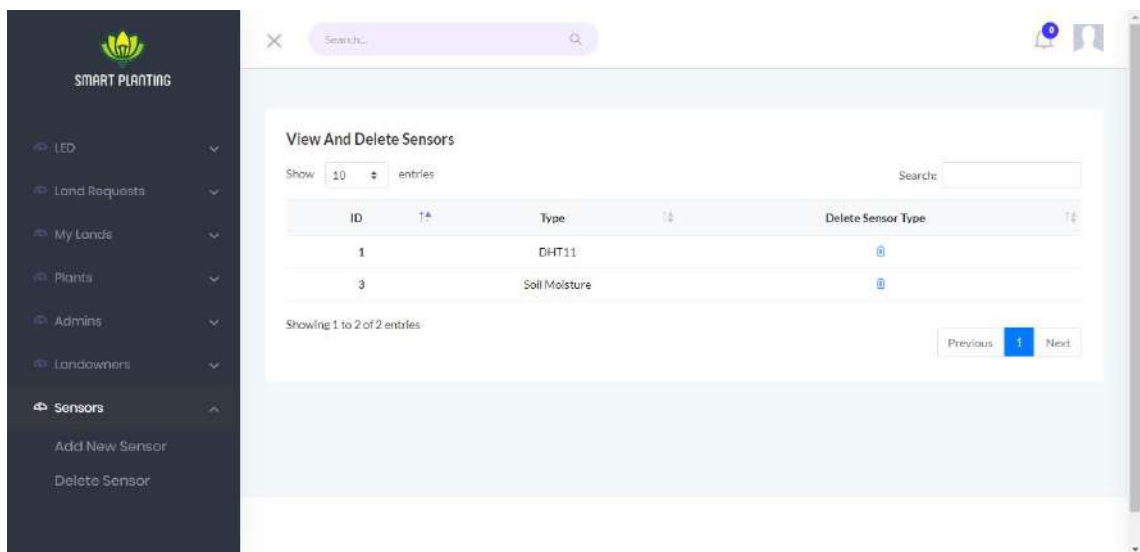


Figure 4.28: Sensors

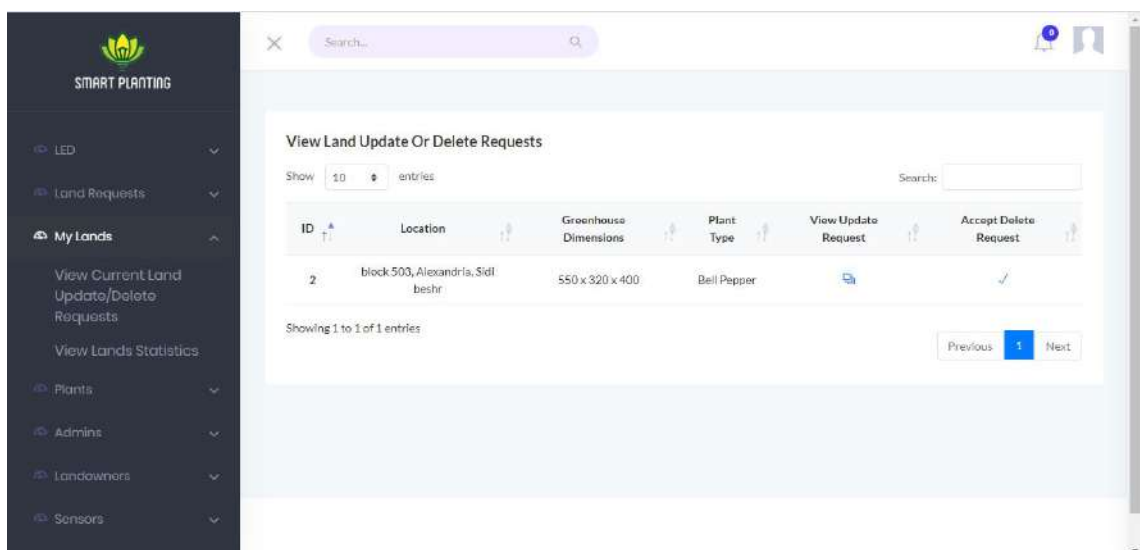


Figure 4.29: Admin view update/delete requests of the greenhouse

The screenshot shows a web application interface for adding greenhouse details. On the left is a dark sidebar with the 'SMART PLANTING' logo and navigation options: 'Land Requests' (with sub-options 'Add New Land Request' and 'Delete Land Request'), and 'My Lands'. The main content area is titled 'Add the greenhouse details' and contains several form fields: 'City*' (a dropdown menu with the placeholder 'Choose the greenhouse city'), 'Area*' (a dropdown menu), 'Location in details*' (a text input field), 'Plant Type*' (a dropdown menu with 'Tomato' selected), and 'Greenhouse dimensions' (three input fields for 'Length', 'Width', and 'Height'). A green 'send' button is located at the bottom left of the form area.

Figure 4.30: Landowner making a new request with a new greenhouse

The screenshot shows the 'View And Delete Land Requests' page. It features a search bar at the top right and a table of requests. The table has columns for ID, Location, Greenhouse Dimensions, Plant Type, and Withdraw. A single entry is visible with ID 5, Location '8th Block 100, Cairo, Nasr City', Greenhouse Dimensions '200x100x300', and Plant Type 'Cucumber'. The 'Withdraw' column contains a trash icon. Below the table, it says 'Showing 1 to 1 of 1 entries' and includes 'Previous', '1', and 'Next' navigation buttons.

ID	Location	Greenhouse Dimensions	Plant Type	Withdraw
5	8th Block 100, Cairo, Nasr City	200x100x300	Cucumber	

Figure 4.31: Landowner viewing all his/her land requests in the system

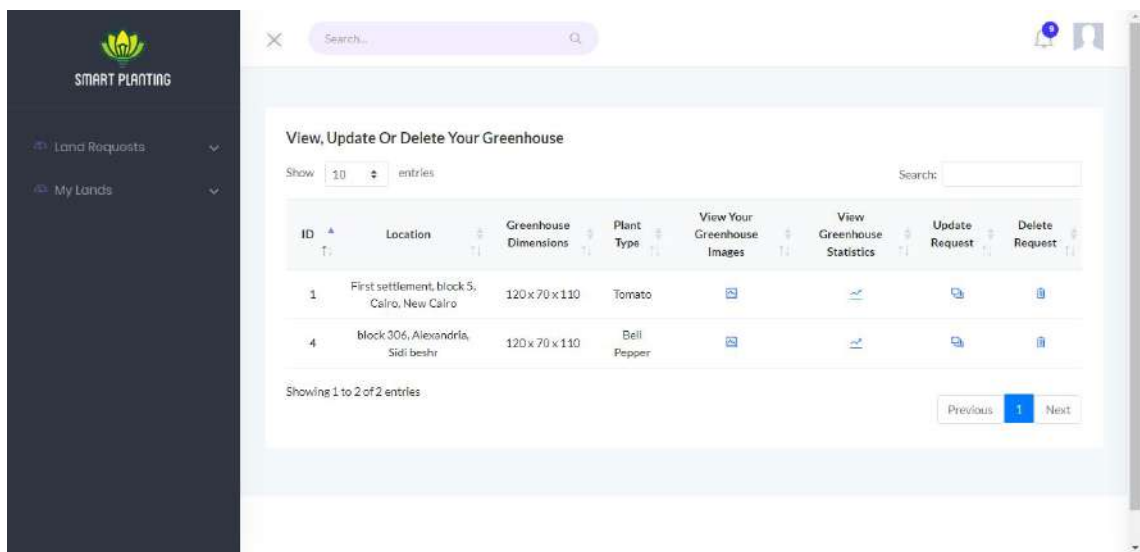


Figure 4.32: Landowner viewing all his/her lands in the system

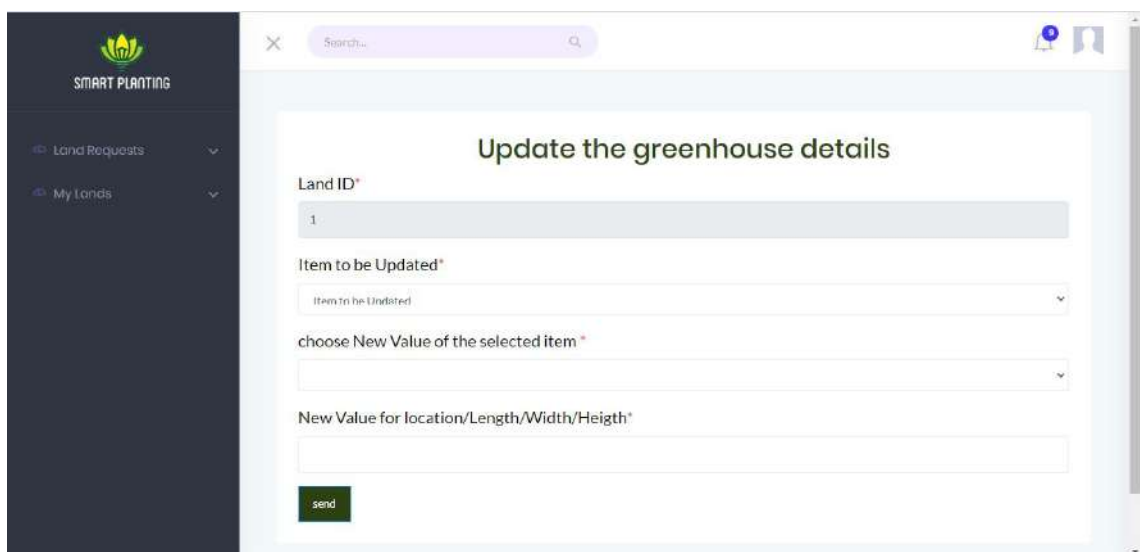


Figure 4.33: Landowner update his/her lands request



Figure 4.34: Landowner view statistics

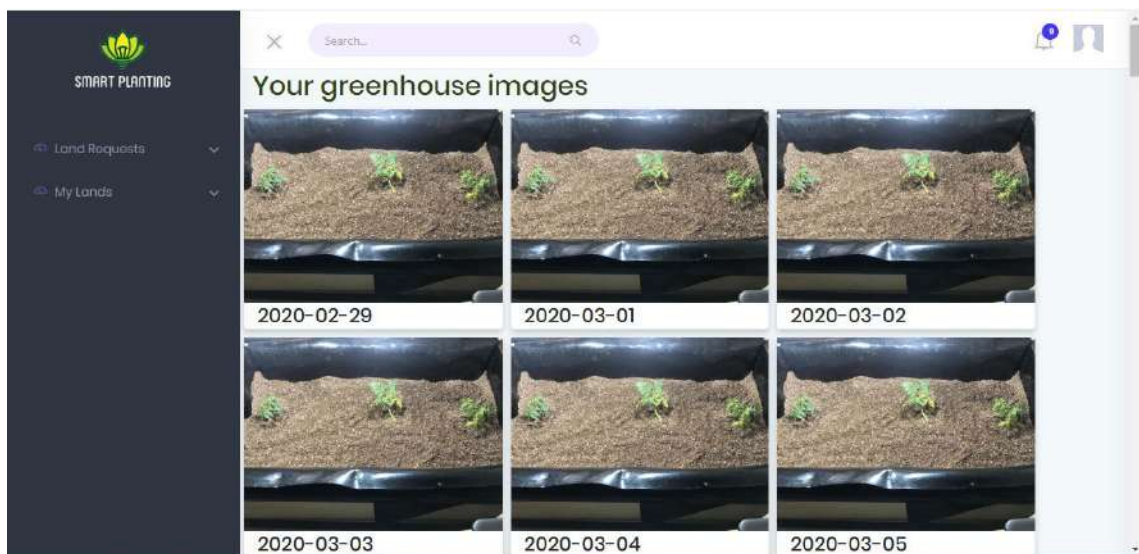


Figure 4.35: Landowner view greenhouse images

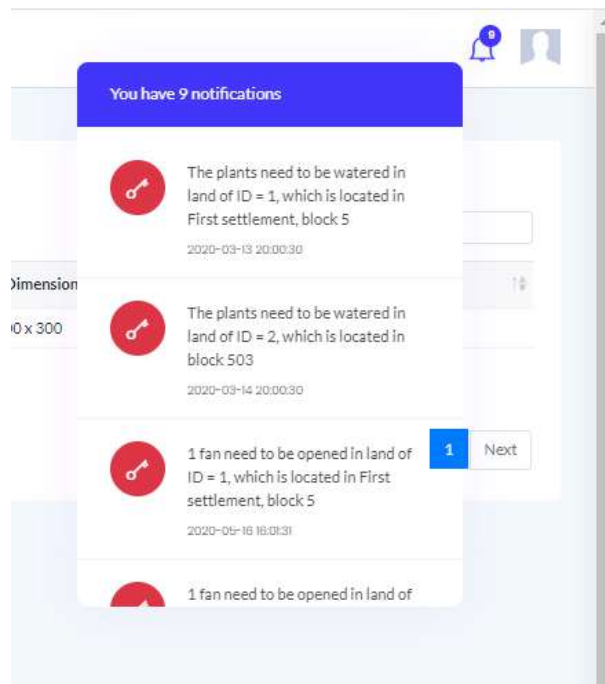


Figure 4.36: Landowner view notifications

Fig. 4.17 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. 4.19, 4.20, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29 and if you logged in as a landowner you will access the screens in Fig. 4.30, 4.31, 4.32, 4.33, 4.35, 4.36. 4.18 shows sign-up page to let landowners register to the system. Fig. 4.19 shows how the admin will be able to edit his/her personal information such as (Username, password, mobile number, ..). Fig. 4.20 show all landowners with their lands. Fig. 4.21 shows that the admin can view all requests that the landowner sends to add new greenhouse. Fig. 4.22 shows that the admin can add new plant type by typing it's name. Fig. 4.23 shows that the admin will be able to view all the plant types and it's id. Fig. 4.24 show admin the plant statistics in all lands. Fig. 4.25 shows that the admin can add new admins. Fig. 4.26 shows the admin all the led light colors. Fig. 4.27 shows that the admin can assign led light color for each plant in each stage. Fig. 4.28 shows the admin all system sensors. Fig. 4.29 shows that the admin can view update/delete land requests. Fig. 4.30 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. 4.31 shows

that the landowner will be able to view all his land requests. Fig. 4.32 shows that the landowner will be able to view all information about his greenhouses. Fig. 4.33 shows the landowner his land statistics. Fig. 4.35 shows the landowner his greenhouse images. Fig. 4.36 shows the landowner his greenhouse notifications.

4.6 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

Figure 4.37: Required Matrix 1

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

Figure 4.38: Required Matrix 2

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

Figure 4.39: Required Matrix 3

Chapter 5

Evaluation

After the implementation process of the system functionalities, the system is tested by four different experiments.

The first experiment is the masking phase of the frame.

The second experiment is for the plant classification.

The third experiment is for plant diseases classification.

The fourth experiment is the users' feedback about our system interface that is used to track their greenhouse statistics and notifications.

Our experiments were made on tomato plants.

5.1 Experiment 1 - Pre-Experiment

5.1.1 Setup

Greenhouses filled with tomato plant in different stages images were obtained from the internet.

5.1.2 Goal

This experiment aims to detect the current plant's stage by obtaining a fixed threshold percentage value of the tomato plant from the dataset images. The frame is originally in the RGB color space; which is converted to HSV color space to separate the needed colors from the image. The lower and upper boundaries are adjusted according to the fruit color and the plant's green color.

5.1.3 Task

The image with HSV color format of both the plant green color and the tomato color is compared to the resulted threshold percentage value. If the green color percentage was less than the resulted threshold percentage value then the plants are on the seeding stage which need blue and green LED lights to be turned on which helps getting the roots to be stronger. Else if the red or the green color percentage was greater than the resulted threshold percentage value then the plants are on the flowering stage which need red LED lights to be turned on to help making the tomato to grow faster.

5.1.4 Results

After testing this experiment several times, we achieved an accurate range of tomatoes colors in it's all stages(yellow-yellowish red-red). Also we got a threshold percentage value equal 40%.

5.1.5 Discussion

By comparing the the resulted threshold percentage value(40%) with the HSV color format of both the plant green color and the tomato color we get the current stage of the plant whether it's seeding, flowering or harvesting stage.

5.2 Experiment 2 - Detecting the tomato

5.2.1 Setup

We used Fruit360 [17] dataset, as it contains a mixture of tomatoes images in different stages.

5.2.2 Goal

This experiment aims to detect the tomato from the testing image frame.

5.2.3 Task

The testing image is passed to extract features from using HOG then pass it to the One Class SVM classifier to detect if there is any tomatoes or not.

5.2.4 Results

After applying different feature extractors and classifiers, Histogram of oriented gradients(HOG) with One Class SVM achieved the highest accuracy which is 81.8% as shown in Table 5.1.

Table 5.1: Experiments algorithms accuracy

algorithm	Accuracy
OC-SVM And HOG	81.8 %
KNN and ORB	60.3%

5.2.5 Discussion

This experiment helps in turning on the needed LED light with the first experiment. As when the tomatoes starts to appear this means that we're in the flowering stage which needs the red led light to be turned on.

5.3 Experiment 3 - Detecting the plant disease

5.3.1 Setup

We used PlantVillage[18] dataset, as it has 9 types of tomato diseases and a healthy class. The dataset was divided into training and testing subsets with a ratio of 3:1 respectively.

5.3.2 Goal

This experiment help us in detecting our plant diseases.

5.3.3 Task

Fastai is used to detect if there is an infected tomato with one of our 9 trained diseases or not. Our 9 trained diseases are: Bacterial spot, Early blight, Late blight, Leaf mold, Septoria leaf spot, Spider mites, Target spot, Yellow curl virus and Mosaic virus.

5.3.4 Results

After applying different classifiers, Fastai achieved the highest accuracy which is 94.8% as shown in Table 5.2. Fastai confusion matrix and plotting are shown in images 5.1 5.2.

Table 5.2: Experiments algorithms accuracy

algorithm	Accuracy
Fastai	94.8%
Keras	86.3%

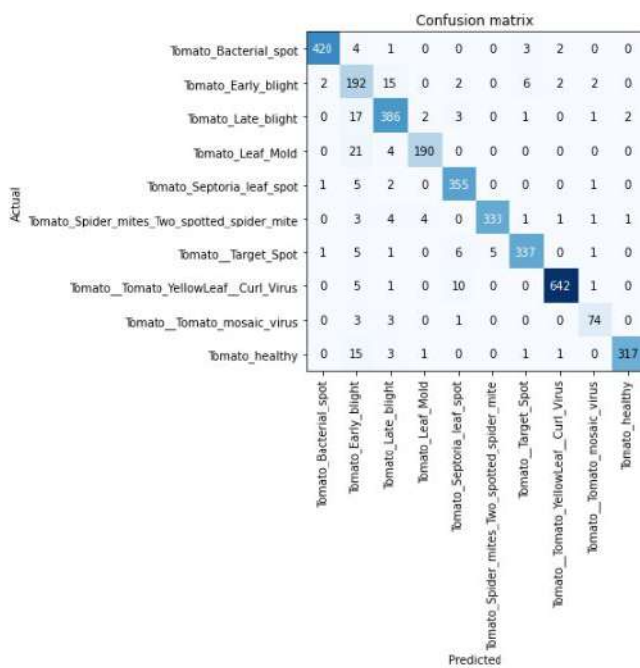


Figure 5.1: Confusion matrix

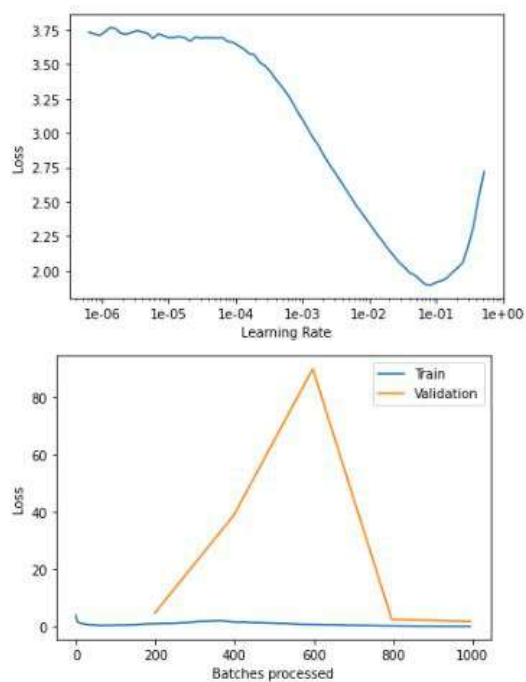


Figure 5.2: Plotting

5.3.5 Discussion

This experiment helps in an early and easily prediction of the diseases. By the early detection, our automated system will notify the landowner to maintain the disease before spreading throughout the whole greenhouse, by this way we ensure that our plants are growing in a good environment.

5.4 Experiment 4 - User feedback

This experiment was made to evaluate the system and to take the user feedback.

5.4.1 Setup

We explained our idea and send our website to 2 experts in the agriculture field. Table 5.3 shows the feedback of those experts on the system.

Table 5.3: User study

Name	Title	Useful	Easy to use	Applicable
Eng. Ahmed Mohamed Saeed	Owner of Oasis company	Yes	Yes	Yes
Eng. Youssef El-Shawekh	Free lancer engineer and owner of El-Shawekh company	Yes	Yes	Yes

5.4.2 Results

We get a positive feedback from both engineers and they assured that our system should be applied in Egypt.

Chapter 6

Conclusion

In this document, we have presented the design, development, and evaluation of an automated greenhouse system that helps the plants in all of its' types having a healthy life-cycle. That was achieved by detecting and classifying both plants' stages and diseases. The system mainly capture frames using a real-time camera that is installed inside the greenhouse model. Then the system starts to proceed into pre-processing, segmentation, feature extraction and finally the classification. It classifies plant stages to Seeding, Flowering, and Harvesting. Upon these stages, the system makes its decision to turn on the needed led light (Red, Blue, Green). During the classification process of the plant growth's stage, it can detect if there's any disease appeared on the plants. The system detects and classifies Tomatoes' diseases into early blight, late blight, leaf mold, spider mites, target spot, mosaic virus, bacterial spot, septoria and yellow curl virus. To get the highest accuracy in classifying both stages and diseases, we had used two different algorithms, as for classifying growth's stages we had used HOG and One-class SVM, that had recorded an accuracy of 81.8%, while in detecting diseases we used Fastai deep learning library that achieved an accuracy of 94.8%.

6.1 Future work

In the near future, we aim to accomplish a fully automated greenhouse system, that doesn't only classify, detect stages and diseases but also to control water pumps, fans and heaters. As that would provide every plant type the suitable atmosphere for growing in a healthy life-cycle. Also we aim to increase our classification model accuracy.

Bibliography

- [1] J. Shijie, J. Peiyi, H. Siping *et al.*, “Automatic detection of tomato diseases and pests based on leaf images,” in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 2537–2510.
- [2] J. Lu, M. Zhou, Y. Gao, and H. Jiang, “Using hyperspectral imaging to discriminate yellow leaf curl disease in tomato leaves,” *Precision agriculture*, vol. 19, no. 3, pp. 379–394, 2018.
- [3] U. Drakulić and E. Mujčić, “Remote monitoring and control system for greenhouse based on iot,” in *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*. Springer, 2019, pp. 481–495.
- [4] R.-a. Li, X. Sha, and K. Lin, “Smart greenhouse: A real-time mobile intelligent monitoring system based on wsn,” in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2014, pp. 1152–1156.
- [5] M. F. Siddiqui, N. Kanwal, H. Mehdi, A. Noor, M. A. Khan *et al.*, “Automation and monitoring of greenhouse,” in *2017 International Conference on Information and Communication Technologies (ICICT)*. IEEE, 2017, pp. 197–201.
- [6] E. Dănilă and D. D. Lucache, “Efficient lighting system for greenhouses,” in *2016 International Conference and Exposition on Electrical and Power Engineering (EPE)*. IEEE, 2016, pp. 439–444.
- [7] Z. Tang, J. Yu, J. Xie, J. Lyu, Z. Feng, M. M. Dawuda, W. Liao, Y. Wu, and L. Hu, “Physiological and growth response of pepper (*capsicum annum* l.) seedlings to supplementary red/blue light revealed through transcriptomic analysis,” *Agronomy*, vol. 9, no. 3, p. 139, 2019.

- [8] [Online]. Available: <http://www.fao.org/3/v9978e/v9978e0e.htm#targetText=Tomatoes> are grown in three, and late blight, and nematodes
- [9] E. Duffin, "Impact of the coronavirus pandemic on the global economy-statistics & facts," *Statistica Report, April*, vol. 3, p. 2020, 2020.
- [10] M. N. Khamis, N. F. Ismail, N. A. M. Yunus, and D. Ahmad, "Led lighting with remote monitoring and controlling system for indoor greenhouse," in *2017 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*. IEEE, 2017, pp. 81–84.
- [11] R. T. Watson, M.-C. Boudreau, and M. W. van Iersel, "Simulation of greenhouse energy use: An application of energy informatics," *Energy Informatics*, vol. 1, no. 1, p. 1, 2018.
- [12] H. Sabrol and K. Satish, "Tomato plant disease classification in digital images using classification tree," in *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2016, pp. 1242–1246.
- [13] C. S. Hlaing and S. M. M. Zaw, "Model-based statistical features for mobile phone image of tomato plant disease classification," in *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. IEEE, 2017, pp. 223–229.
- [14] R. Ghaffari, F. Zhang, D. Iliescu, E. Hines, M. Leeson, R. Napier, and J. Clarkson, "Early detection of diseases in tomato crops: An electronic nose and intelligent systems approach," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–6.
- [15] X. E. Pantazi, D. Moshou, and A. A. Tamouridou, "Automated leaf disease detection in different crop species through image features analysis and one class classifiers," *Computers and electronics in agriculture*, vol. 156, pp. 96–104, 2019.
- [16] A. Adedoja, P. A. Owolawi, and T. Mapayi, "Deep learning based on nasnet for plant disease recognition using leave images," in *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE, 2019, pp. 1–5.

-
- [17] H. Mureşan and M. Oltean, “Fruit recognition from images using deep learning,” *Acta Universitatis Sapientiae, Informatica*, vol. 10, no. 1, pp. 26–42, 2018.
- [18] spMohanty, “spmohanty/plantvillage-dataset,” Sep 2018. [Online]. Available: <https://github.com/spMohanty/PlantVillage-Dataset>