Palm Tree Disease Detection Using Artificial Intelligence and Image Processing

Hazem Alaa, Khaled Waleed, Moataz Samir, Mohamed Tarek
Supervised by Dr. Mostafa Abdelsalam, Eng. Hager Sobeah

January 16,2020

1 Introduction

1.1 Purpose of this document

The purpose of this software specification document is to demonstrate a palm tree disease detection system. The system will be able to differentiate between three common diseases that infect the palm trees which are; leaf spot,leaf blight and the red palm weevil, Also our system will be built using OpenCv, CNN, SVM, Android Java, and Firebase Database.

1.2 Scope of this document

This document targets farm owners and agricultural experts who are going to use the application in order to detect whether their palms are infected or not; which will save them a lot of money and time as some of the diseases are hardly detected at an early stage.

1.3 Overview

In our project the user has the choice to upload three types of images; a thermal and a normal image. Every image type carries it's own unique features, for example, thermal images; they carry temperature information for every pixel and that will be used as a main feature to detect Red Palm Weevil existence. On the other hand, normal images will be used to detect other diseases such as leaf spots and leaf blights . As a first step in using our application, the user will be asked to either upload an image or capture an new image. So that image processing techniques can be applied on the captured or imported image such as histogram normalization and image masking, as well as applying some feature extraction techniques and passing these features to our classifiers to tell whether the tree is infected or healthy.

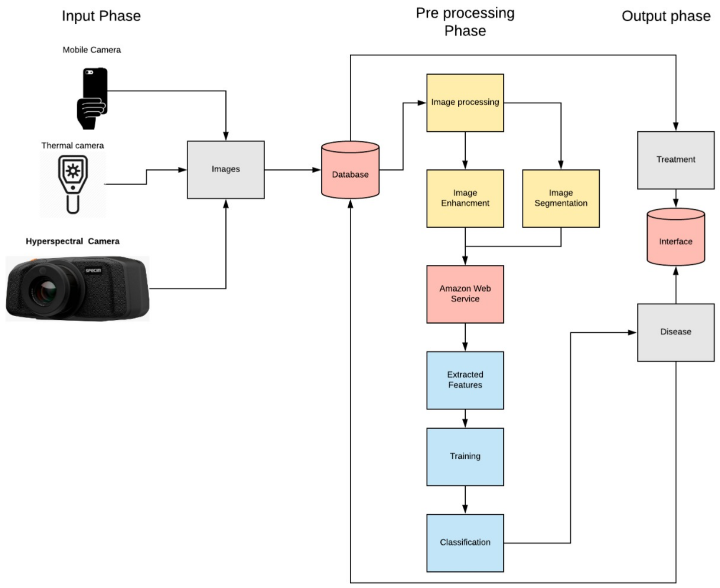


Figure 1: System Overview

1.3.1 Context Diagram

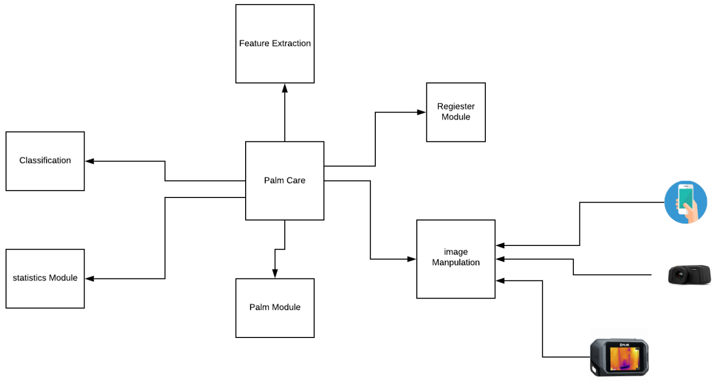


Figure 2: context Diagram

in [Fig3]

- Login Module: Is the module responsible for the login operation of admin,expert and palm owner
- User Manipulation Module: Is responsible for the admin to edit user information or delte it or view all of it
- Image Manipulation Module: this Module is responsible for the image processing phase and is explained in [Fig1].
- statistics Module: is responsible for display the reports generated by the system.
- Feature extraction Module: is responsible for extracting the features required for training in the data-set system can pass this features to classification
- Classification Module : we use the features to compare the training labels such as csv files
- Palm Module: user can choose a specific palm type to add and then import/capture an image and have the privilege to edit the palm data and delete palm if he want to.also he can check if the palm is infected and on which level.

1.4 Business Context

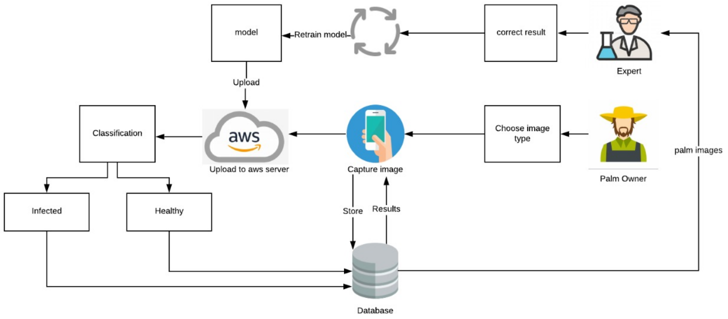


Figure 3: Bussnies Model

Palm trees are considered of great importance to the agricultural economy of many countries, especially in the Middle East. In 2006, productive date palm trees were estimated at 11,888,020 million trees and produced 1,328,720 tones of dates. Due to the great importance of palm economy to the middle east we have chosen to address the three top dangerous diseases among all of the palm diseases. At the top rank is our Red Palm Weevil (RPW), a worldwide destructive tiny insect that faces 17 palm species in 35 countries and is considered the most threatening palm pest according to the Food and Agriculture Organization of the United Nations (FAO). RPW severely damage the palm trees whileliving its life cycle inside the core of the palm itself, its symptoms cannot be visible to the naked eyes during the early stages of infection but it only appears when the palm is nearly dead and incurable. Second comes the Leaf blights and Leaf spots which are not only common diseases to palm trees, but they are also common diseases for many other plants such as tomato, cucumber, and mango plants. Accordingly, we aim to protect palm trees from these infections in order to maintain our huge wealth by assuring the quality and quantity of the palm trees in the middle east.

2 General Description

2.1 Product Functions

Register module:is the module where the user insert his personal data to make an account such as email,password,gender and name and then the user get verified and login using his email and password

Palm module: this the module where any user with this type of permission the palm owner enter his palm trees data and can update this data or even delete or see all the palm trees that he own and what are there current infection level

image manipulation module: is where all the image processing and enhancement is done we done some resizing using keras built in attributes to help improve the image. feature extraction module: feature extraction depend on the algorithm we in the time we use svm we extracted the feature ourselves unlike cnn where the algorithm extracted the features by itself classification module: is the module where the algorithm decide where is the palm infected with a certain disease or it's healthy and what infection level this disease is. Report module: is where we gather all the data that availabele and give the user the final answer where the palm is infected or healthy and on which level the disease is and how to treat the disease

2.2 Similar System Information

1. Plantix Application

Plantix is an application which only needs a picture of a sick plant to receive a diagnosis and the appropriate treatment tips. Plantix recognizes specific patterns of each disease, pest and deficiency leaves. Moreover, Plantix social network provides exchanging experiences and information with like-minded people in the immediate area as well as with international experts which can help users to get helpful answers and practical solutions for disease,pests and deficiency symptoms which means that Plantix gets better with each new user.

2. Detection of tomatoes diseases using thermal imaging [5].

Early time detection of plant diseases can mitigate the worldwide losses in agriculture.Thermal imaging provides a fast and non-destructive way of scanning plants for diseased regions but some environmental conditions can make a lot challenges due to canopy architecture, leaf angles, sunlit and shaded regions and the depth (distance) of plant regions from the camera . This system provide combining of thermal and visible light image data with depth information and develop a machine learning system to remotely detect plants infected with the tomato powdery mildew fungus (Oidium neolycopersici).The system extract feature set from the image data using local and global statistics and show that by combining these with the depth information will improve the accuracy of detection of the diseased plants.The detection algorithm consists of registration, depth estimation, feature extraction and classification.The classification algorithm which is used is support vector machine (SVM) which shows a high accuracy (80%) in detection of tomato powdery mildew fungus but in day 9, and can show higher accuracy more than 90% but after day 9 from infection, however this is not very beneficial at the commercial scale as the disease might spread across the crop after day 9. **3. Detection of red palm weevil infestation on palm species using**
thermal imaging [3].

The date palm (Phoenix dactylifera L.) and other palm species have recently been threatened by the red palm weevil (RPW) Rhynchophorus ferrugineus Olivier, which is very difficult to be detected at early stage.Thermal camera it is a tool used for years in agriculture science to detect plant stress and at experimental level to find hidden cavities inside trunks of woody plants.Thermal measurement should be taken at the same time and with the same view angle and this is important because different weather conditions, the reduction of sunlight, sun reflections and glitters may alter the results, which rely on plant tissues transpiration.The thermal camera was used to compare internal temperatures of neighbouring palm trees, only a comparison between nearby palm trees shows if a palm is warmer than the others. If a palm is warmer than the others, at the same climate and solar conditions, it is probably infested. The thermal camera showed a good accuracy (77.73%) better than the digital camera that showed a lower accuracy of 66.67% due to the fact that the red weevil mainly attacks the base of the stem and therefore there are no visible symptoms on the crown shape that would be picked up in image analysis.

4. Early Detection and Quantification of Verticillium Wilt in Olive Using Hyperspectral and Thermal Imagery over Large Areas [1].

Automatic methods for an early detection of plant diseases using remotesensing are critical for crop protection.Verticillium wilt (VW) of olive caused by Verticillium dahliae can be controlled only if detected at early stages of development.High-resolution thermal and hyperspectral imagery were acquired with a manned platform which flew a 3000-ha commercial olive area. These methods are rapid and reliable, allowing real-time plant disease monitoring for disease control and management V. dahliae infects the plant through the roots and colonizes its vascular system, blocking water flow and eventually inducing wilt symptoms, This damage results in a significant reduction in leaf transpiration rate which finally leads to leaf chlorosis and defoliation, causing a change of spectral reflectance.Chlorophyll content tends to decrease in infected plants, showing a higher reflection in the visible (VIS) green (550 nm) and red-edge (650–720 nm) regions, and here comes the turn of the hyperspectral imaging to show these changes which detect the V. dahliae.In addition, the thermal-infrared (TIR) region (8000–15,000 nm) is highly suitable for the detection of V.dahliae infection due to the decrease in transpiration rate which induces stomata closure, reducing evaporative cooling and increasing canopy temperature.Linear discriminant analysis (LDA) and support vector machine (SVM) classification methods were applied to classify V. dahliae disease, LDA reached an overall accuracy of 59.0% while SVM obtained a higher overall accuracy, 79.2%.

5. Detection of Chimaera and Anthracnose diseases in palm oil tree using image processing techniques[2].

Disease in palm oil sector is one of the major concerns because it affects the production and economy losses to Malaysia. Diseases such as Chimaera and Anthracnose appear as spots on the leaf and if not treated on time, cause growth problems of the palm oil tree.The Chimaera disease is caused by the genetic problem of the oil palm tree seeds. The symptoms of the disease are the leaves have white stripe or yellowish-white, and the lack of chlorophyll. Anthracnose disease can affect all palm oil trees at any growth stages. The symptom is most visible on leaves and also ripe fruit.There are five main steps used for classification of palm oil leaf diseases, classification consists of image acquisition through digital camera, image enhancement, clustering and classification.By going through this processes, the presence of diseases on the palm oil leaf can be identified.The classifier in this system is support vector machine (SVM), The classification shows that SVM achieves accuracy of 97% for Chimaera and 95% for Anthracnose.

6. Real-Time Detection of brown spot Apple Leaf Disease Using Deep Learning Approach Based on Improved Convolutional Neural Networks[4].

Alternaria leaf spot, Brown spot, Mosaic, Grey spot, and Rust are five common types of apple leaf diseases that severely affect apple yield. various spectroscopic and imaging techniques have been studied for detecting plant diseases. However, they require precise instruments and bulky sensors, which lead to high cost and low efficiency. In recent years, with the popularization of digital cameras and other electronic devices, automatic plant disease diagnosis via machine learning has been widely applied as a satisfactory alternative.Based on this, a new apple leaf disease detection model that uses deep-CNNs is proposed as a classifier for detecting apple leaf diseases.CNN classifier shows 80.45% for detection of brown spots using DSSD model.

7. Leaf blight Disease Detection in tomatoes leaves and Classification based on CNN with LVQ Algorithm[6].

The early detection of diseases is important in agriculture for an efficient crop yield. The bacterial spot, late blight, septoria leaf spot and yellow curved leaf diseases affect the crop quality of tomatoes. Automatic methods for classification of plant diseases also help taking action after detecting the symptoms of leaf diseases.Leaf Blight is first seen as large brown spots with greengray edges on old leaves. As the disease matures, the spots become darker. Eventually the disease infects the whole plant and causes the plant to be seriously damaged. In the proposed paper they developed a CNN model based on RGB components of the tomato leaf images on PlantVilliage dataset.They preferred Learning Vector Quantization (LVQ) algorithm as classifier due to its topology and adaptive model.One of the main challenges in disease detection and classification for this study is that the leaves with different diseases are very similar to each other.

2.3 User Characteristics

- Farm owner:Basic knowledge of using an android phone
- Expert:Basic knowledge of using an android phone

2.4 User Problem Statement

Plant disease detection are one of the biggest problems world wide and one of the most expensive problems in our project we fix that problem by making the process far cheaper than it use to be.

- the main problem for the palm owner that is it so expensive to hire an expert to monitor the field of palms monthly
- for the expert some disease are really hard to detect in the early stages of the diseases

2.5 User Objectives

users will be able to view the classified image if the palm is infected or healthy and what infection level the disease is and see what treatment it needs

2.6 General Constraints

1. the application work on android devices only
2. User must have internet connection
3. user must have a good mobile camera average 8 MGP or above
4. user must have a portable thermal camera like seek pro

3 Functional Requirements

1.server :

ID	SR1
Title	user registration
Description	user register an account in Firebase
Input	name, email, password, user Type, gender
Action	User register an account
Output	Authentication Token
Pre-condition	A developer account must be created in firebase web application
Post-condition	None
Dependencies	None

ID	SR2
Title	Authenticate User
Description	The server will search in firebase for an email matching the email entered by user and find if the entered password matches the password connected to the registered email.
Input	email and password.
Action	Authenticating user in firebase.
Output	Authentication Token.
Pre-condition	Pre-condition: A developer account must be created in firebase web application
Post-condition	None
Dependencies	SR1, AC2.

ID	SR3
Title	update Profile
Description	user update his own information in his profile and the new information are updated in the firebase
Input	name, email, password, gender.
Action	user update his profile information
Output	Status Message.
Pre-condition	user must be authenticated, server must be connected to the Firebase.
Post-condition	None
Dependencies	SR1, SR2, AC1

ID	SR4
Title	Password Hashing
Description	once the user register successfully the server hashes the password so it could be secured.
Input	user's password
Action	hashing the user password
Output	The Encrypted Password
Pre-condition	user must enter his password
Post-condition	None
Dependencies	SR1

2.Admin

ID	AD3
Title	deleteUser
Description	Admin can delete any user from the storage
Input	userId
Action	admin deletes a user
Output	Status Message.
Pre-condition	aimed user must not be already deleted , user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD4
Title	viewUserProfile
Description	Admin can view any user profile in the storage
Input	userId
Action	Admin views user profile
Output	user profile
Pre-condition	aimed user must not be already deleted, user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD5
Title	viewAllUsers
Description	Admin can view a table of all users in the storage
Input	None
Action	admin view all users
Output	array of users
Pre-condition	user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD6
Title	addUserType
Description	can add a new user Type to the storage
Input	name , parentId
Action	admin adds new user type
Output	Status Message
Pre-condition	user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD7
Title	editUserType
Description	admin can edit the name of a user type in the storage
Input	userTypeId , name
Action	admin edit a user type
Output	user type name
Pre-condition	there must at least one user type added to the storage, user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD8
Title	deleteUserType
Description	can delete a user Type from the storage
Input	UserTypeId
Action	admin can delete a user Type
Output	Status Message.
Pre-condition	there must at least one user type added to the storage , user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD9
Title	viewAllUserTypes
Description	admin can view a table of all the user stored in the storage.
Input	None
Action	admin view all users
Output	array of users
Pre-condition	user authentication token, action must be made by user of user type admin
Post-condition	None
Dependencies	SR1, SR2, AC1.

ID	AD10
Title	viewDisease
Description	admin can view from database all the data of a disease with the palm images images attached to it.
Input	DiseaseId
Action	admin view disease data.
Output	object of class Disease.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.there must have been previously at least one disease added in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1, AD14.

ID	AD11
Title	viewAllDiseases
Description	Admin can view a prototype of all the disease recorded in database
Input	None.
Action	admin view all diseases
Output	array of objects of class disease.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.at least one disease must be previously recorded in the Firebase.
Post-condition	None.
Dependencies	SR2

ID	AD12
Title	addPalmType
Description	Admin can add a new palm type to be recorded in the database.
Input	name
Action	admin adds new palm type.
Output	Status Message.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase.
Post-condition	New Palm Type Added to the system.
Dependencies	SR1, SR2, AC1.

ID	AD13
Title	editPalmType
Description	Admin can edit the name of a palm type recorded in the database.
Input	palmTypeId , name
Action	admin edits the palm type's name
Output	Status Message.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.Admin must have previously added at least one palm type to be recorded in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1, AD14.

ID	AD14
Title	viewAllPalmTypes
Description	Admin can view all palm types recorded in the database.
Input	None
Action	admin view all palm types
Output	array of objects of class PalmType
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.Admin must have previously added at least one palm type to be recorded in the Firebase.
Post-condition	None
Dependencies	: SR1,SR2.

ID	AD15
Title	getUserTypeName
Description	Admin can view the name of a specific object of class UserType
Input	None.
Action	Admin views User Type name.
Output	User Type Name.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.Admin must have previously added user types to be recorded in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1.

ID	AD16
Title	viewUserTypePermissions
Description	Admin can view the All permissions related to a specific user type.
Input	None.
Action	When admin selects a user type he can view all permissions related to the selected user type.
Output	array of objects of class Permissions.
Pre-condition	1.Admin must have an authenticated account. 2.Admin must enter from admin account. 3.Admin Device must be connected to the Internet. 4.Server must be connected to the Firebase. 5.Admin must have previously added user types to be recorded in the Firebase. 6.All permissions must be previously recorded in the Firebase.
Post-condition	User Type's permissions are displayed in a table.
Dependencies	SR1, SR2, AC1.

3.Expert

ID	EXP1
Title	Correct Result
Description	Expert can correct all results obtained by the model.
Input	PalmId, infectionLevelId
Action	Expert
Output	None
Pre-condition	1.Admin must have an authenticated account 2.Admin must enter from admin account 3.Admin Device must be connected to the Internet 4.Server must be connected to the Firebase 5.Admin must have previously added at least one palm to be recorded in the Firebase 6.server must be connected to the AWS
Post-condition	results
Dependencies	SR1, SR2, AC1, AC6

4.PalmOwner

ID	PO1
Title	AddNewPalm
Description	Palm Owner can add new palm to his collection
Input	palmTypeId , imageTypeid , image
Action	Palm Owner will upload image for palm and add its information { type of palm , image , type of image} and submit the form
Output	Status Message
Pre-condition	1.Palm Owner must have an authenticated account 2.Palm Owner Device must be connected to the 3.server must be connected to the Firebase 4.server must be connected to the AWS
Post-condition	1.New palm data will be added to the Firebase 2.New palm images will be added to the AWS
Dependencies	SR1, SR2, AC1

ID	PO2
Title	Update Palm Info
Description	Palm Owner can update an existing palm data
Input	palmId ,palmTypeId , imageTypeid , image
Action	Palm Owner views a form where he can submit changes to the palm type and image
Output	Status Message
Pre-condition	1.Palm Owner must have an authenticated account 2.Palm Owner Device must be connected to the Internet 3.server must be connected to the Firebase 4.server must be connected to the AWS 5.Palm Data must be already existed in the Firebase
Post-condition	1. palm data will be changed in the Firebase 2.New palm images will be changed in the AWS
Dependencies	SR1, SR2, AC1, AC6

ID	PO3
Title	delete Palm
Description	Palm Owner can delete a palm
Input	PalmId
Action	Palm Owner Selects a Palm and press Delete
Output	Status Message
Pre-condition	1.Palm Owner must have an authenticated account 2.Palm Owner Device must be connected to the Internet 3.Server must be connected to the Firebase 4.Palm Data must be already existed in the Firebase
Post-condition	Palm Data will be deleted from database
Dependencies	SR1, SR2, AC1, PO6

ID	PO4
Title	addPalmImages
Description	Palm Owner can add an array of images added to the palm data
Input	File Images[]
Action	Palm owner press add button and upload the images he selects
Output	Status Message
Pre-condition	1.Palm Owner must have an authenticated account 2.Palm Owner Device must be connected to the Internet 3.Server must be connected to the Firebase 4.Palm Data must be already existed in the Firebase
Post-condition	1.Palm new images are inserted in the Firebase 2.Palm new images are inserted in the AWS
Dependencies	SR1, SR2, AC1, AC6, AC1

ID	PO5
Title	deletePalmImages
Description	Palm Owner can delete number of palm images
Input	File Images[]
Action	Palm Owner enters the page of a specific palm information
Output	Status Message
Pre-condition	1.Palm Owner must have an authenticated account 2.Palm Owner Device must be connected to the Internet 3.Server must be connected to the Firebase 4.Palm Data must be already existed in the Firebase
Post-condition	palm images must be deleted from database
Dependencies	SR1, SR2, AC1, AC6, AC1.

5.Account

ID	AC1
Title	login
Description	The system searches in the database for the email and password entered by the User and opens to user his account in case the data entered matches the database records.
Input	email and password
Action	User can login to the system.
Output	None.
Pre-condition	1.User must have an authenticated account. 2.User must previously register an account in the system. 3.User Device must be connected to the Internet. 4.Server must be connected to the Firebase.
Post-condition	1.Password entered must not less than 8 characters. 2.email entered must have a correct domain name.
Dependencies	SR1, SR2, AC1, SR4, AC2.

ID	AC2
Title	isVerified
Description	It checks if the current user object is verified.
Input	email_verified_at
Action	It return the value of isVerified column for the targeted object in user table.
Output	TimeStamps
Pre-condition	1.User must previously register an account in the system. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase.
Post-condition	None.
Dependencies	SR1.

ID	AC3
Title	Logout
Description	User logouts of the System.
Input	None.
Action	Closes the current opened user session.
Output	Status Message.
Pre-condition	1.User must be previously signed up or signed in to the system. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase.
Post-condition	Server must not view the user profile after successfully loging out.
Dependencies	SR1, AC1.

ID	AC34
Title	updateUserInfo
Description	User can update his information.
Input	userId, name, email, password, gender.
Action	Updates the recorded data for the user in Database.
Output	Status Message.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase.
Post-condition	User profile view must be updated with previous data.
Dependencies	SR1, SR2, AC1.

ID	AC5
Title	viewPalm
Description	User can view specific palm.
Input	PalmId
Action	search in database the object data of a specific palmId.
Output	Object of class Palm.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	user view a page containing palm data.
Dependencies	SR1, SR2, AC1.

ID	AC6
Title	ViewResults
Description	View the palm classification results.
Input	PalmId
Action	the function executes the the classification saved results in database.
Output	Array of Objects
Pre-condition	1.User must have an authenticated account. 2.User must previously register an account in the system. 3.User Device must be connected to the Internet. 4.Server must be connected to the Firebase.
Post-condition	user can view the palm results.
Dependencies	SR1, SR2,AC1.

ID	AC7
Title	setStrategy
Description	set the strategy pattern value
Input	IVeiwAllPalms
Action	the function sets the value of IViewAllPalms variable.
Output	None.
Pre-condition	1.User must have an authenticated account. 2.User must previously register an account in the system. 3.User Device must be connected to the Internet. 4.Server must be connected to the Firebase.
Post-condition	the strategy pattern value is set.
Dependencies	SR1, SR2,AC1.

ID	AC8
Title	getStrategy
Description	get the strategy pattern value
Input	IVeiwAllPalms
Action	the function sets the value of IViewAllPalms variable.
Output	IVeiwAllPalms
Pre-condition	1.User must have an authenticated account. 2.User must previously register an account in the system. 3.User Device must be connected to the Internet. 4.Server must be connected to the Firebase.
Post-condition	the strategy pattern value is returned.
Dependencies	SR1, SR2,AC1.

5.Image

ID	Img1
Title	getImageType
Description	view image type name from a specific typeId in database.
Input	imageTypeObject, typeId
Action	use the given typeId to retrieve the inage type name from database.
Output	image type name
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Image Data must be already existed in the Firebase. 5.Image Type Data must be already existed in the Firebase.\par
Post-condition	attached ype is shown.
Dependencies	SR1, SR2, AC1.

ID	Img2
Title	getDiseases
Description	get diseases attached to the image.
Input	None.
Action	gets Disease ids attached to the image and uses it to retrieve diseases from disease table in database.
Output	array of objects of type Disease.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Image Data must be already existed in the Firebase. 5.Image Type Data must be already existed in the Firebase. 6.Diseases Data must be already existed in the Firebase.
Post-condition	retrieve disease attached to the image.
Dependencies	SR1, SR2, AC1.

ID	img4
Title	+ getInfectionLevels
Description	retireve all infection levels attached to the image
Input	InfectionLevelObject
Action	get specific infection levels data from database.
Output	array of InfectionLevels
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Image Data must be already existed in the Firebase. 5.Image Data must be already existed in the Firebase. 6.Infection Level Data must be already existed in the Firebase.
Post-condition	retrieve infection levels attached to the image.
Dependencies	None.

ID	img5
Title	getPalm
Description	the function gets images palm
Input	Palm object
Action	retrieving the images related palm from database.
Output	Palm object
Pre-condition	1.Server must be connected to the Firebase. 2.Palm Data and its images must be already existed in the Firebase.
Post-condition	palm object retieved.
Dependencies	None.

6.InfectionLevel

ID	IL1
Title	getDisease
Description	get the disease attached to specific infection level
Input	InfectionLevelId
Action	retrieve data from disease table from database.
Output	Disease
Pre-condition	1.Server must be connected to the Firebase. 2.infection level Data and its images must be already existed in the Firebase.
Post-condition	None.
Dependencies	None.

ID	IL2
Title	getImages
Description	get images attached to an infection level
Input	None.
Action	get attached images.
Output	array of Images
Pre-condition	1.Server must be connected to the Firebase. 2.infection level Data and its images must be already existed in the Firebase.
Post-condition	the output of infection level's images
Dependencies	None.

7.Disease

ID	D1
Title	getAllInfectionLevels
Description	gets all infection level attached to the disease.
Input	InfectionLevel
Action	uses specific DiseaseId to retrieve
Output	array of InfectionLevels
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Diseases Data must be already existed in the Firebase. 5.InfectionLevel Data must be already existed in the Firebase
Post-condition	infection levels of the disease are retrieved from database.
Dependencies	DEP: SR1, SR2, AC1.

ID	D2
Title	getInfectedPalms
Description	get all palms infected with this disease.
Input	InfectionLevel object
Action	get all palms attached to the diseaseId.
Output	array of Palms
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palms Data must be already existed in the Firebase. 4.InfectionLevel Data must be already existed in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1.

ID	D3
Title	getImages
Description	get all images attached to this Disease.
Input	ImageObject
Action	use DiseaseId to get all images attached to this Disease.
Output	array of PalmsImageObjects
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Disease Data must be already existed in the Firebase
Post-condition	None.
Dependencies	SR1, SR2, AC1.

8.Palm

ID	PM2
Title	getImages
Description	get images attached to an infection level
Input	ImageObject
Action	get attached images.
Output	PalmTypeName
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data and its images must be already existed in the Firebase.
Post-condition	the retrieved of palm's images
Dependencies	SR1, SR2, AC1.

ID	PM3
Title	getPalmType
Description	get the palm type Name from its Palm Type Object.
Input	PalmTypeObject
Action	use the palmTypeId recorded in Palm data to get the palm type name.
Output	PalmTypeName
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.Palm Type Data must be already existed in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1.

ID	PM4
Title	add
Description	user can add palm parts to his palm.
Input	None.
Action	User chooses Palm decorator parts {Kirnaf, Dates, Faseel, Leaves} to decorate a specific palm
Output	Status Message.
Pre-condition	1.User must have an authenticated account 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.Palm Type Data must be already existed in the Firebase.
Post-condition	None.
Dependencies	SR1, SR2, AC1.

9.Permissions

ID	PS1
Title	getUserTypes
Description	gets attached usertypes.
Input	UserType object
Action	gets user types of specific permissionId.
Output	array of UserTypes
Pre-condition	1.User must have an authenticated account 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.userType and permission data must be already previously added in the Firebase..
Post-condition	show permission's user types/
Dependencies	SR1, SR2, AC1.

10.Usertype

ID	UT1
Title	getPermissions
Description	gets attached Permissions.
Input	Permission object
Action	gets user types of specific userTypeId.
Output	array of Permissions
Pre-condition	1.User must have an authenticated account 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.userType and permission data must be already previously added in the Firebase.
Post-condition	show permission's user types/
Dependencies	SR1, SR2, AC1.

11.CnnClassifier

ID	CNN
Title	+ classify
Description	classify image using CNN Model.
Input	Image file
Action	executing CNN trained model.
Output	None.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

12.SVMClassifier

ID	SVM
Title	+ classify
Description	classify image using SVM Model.
Input	Image file
Action	executing SVM trained model.
Output	None.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

13.Kirnaf

ID	KR1
Title	add
Description	inform that the palm has kirnaf.
Input	None.
Action	decorate palm with kirnaf.
Output	Status message.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

14.Leaves

ID	LV1
Title	add
Description	inform that the palm has Leaves.
Input	None.
Action	decorate palm with Leaves.
Output	Status message.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

15.Dates

ID	DA1
Title	add
Description	inform that the palm has Dates.
Input	None.
Action	decorate palm with Dates.
Output	Status message.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

16.Faseel

ID	FA1
Title	add
Description	inform that the palm has Faseel.
Input	None.
Action	decorate palm with Faseel.
Output	Status message.
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

17.ViewAllUserPalms

ID	VUP1
Title	viewAllPalms
Description	View all palms for one user.
Input	None.
Action	retireve palm data of a specific userId
Output	array of Palms
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

18.ViewSystemPalms

ID	VSP1
Title	viewAllPalms
Description	View all palms for All the system.
Input	None.
Action	retireve all palms data in the system.
Output	array of Palms
Pre-condition	1.User must have an authenticated account. 2.User Device must be connected to the Internet. 3.Server must be connected to the Firebase. 4.Palm Data must be already existed in the Firebase. 5.server must be connected to the AWS.
Post-condition	None.
Dependencies	None.

4 Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

4.1 User Interfaces

Use some software for primitive plan of your project. Describes how this product interfaces with the user.

4.1.1 GUI

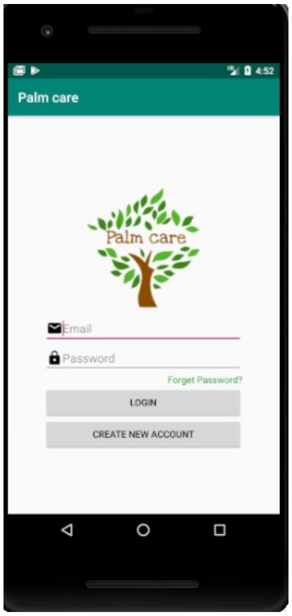


Figure 4: Login

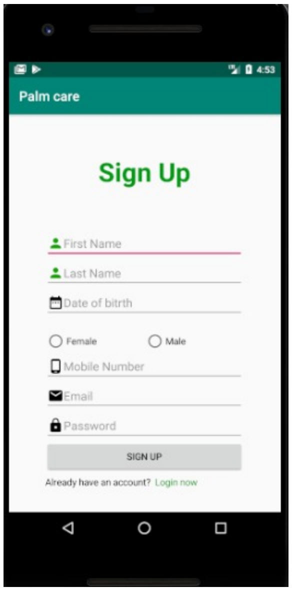


Figure 5: Sign up

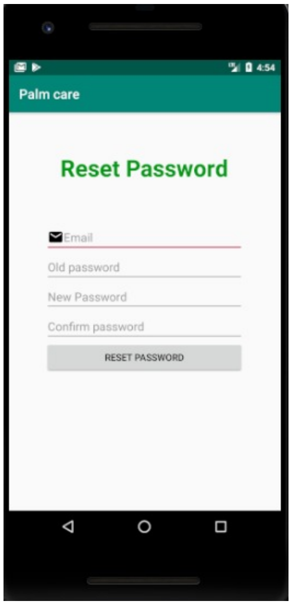


Figure 6: Reset Password

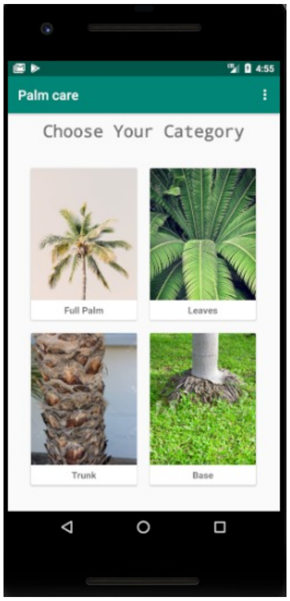


Figure 7: Choose palm parts

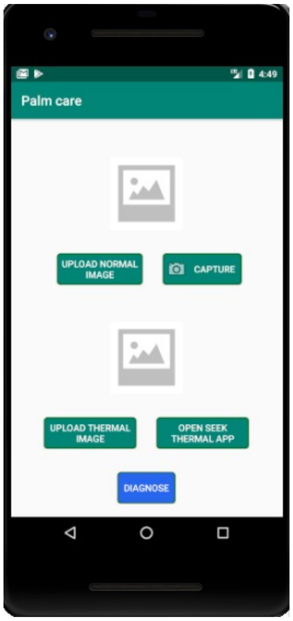
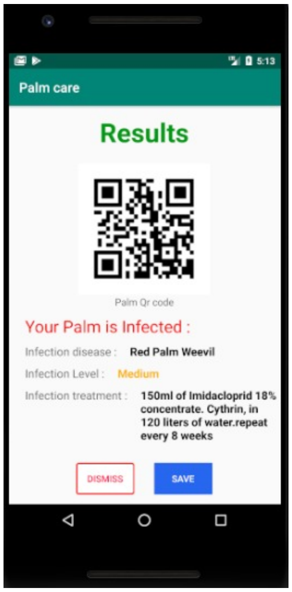


Figure 8: upload image



Results

Palm Qr code

Your Palm is Infected :

Infection disease : Red Palm Weevil

Infection Level : Medium

Infection treatment : 150ml of Imidacloprid 18% concentrate. Cythrin, in 120 liters of water.repeat every 8 weeks

Figure 9: Results

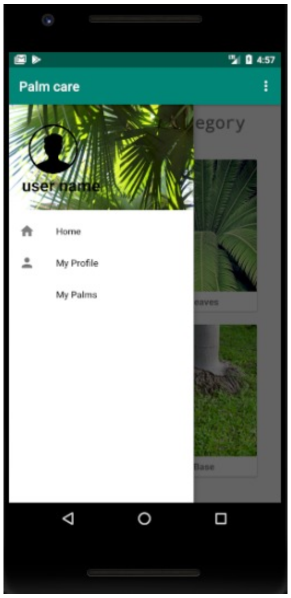
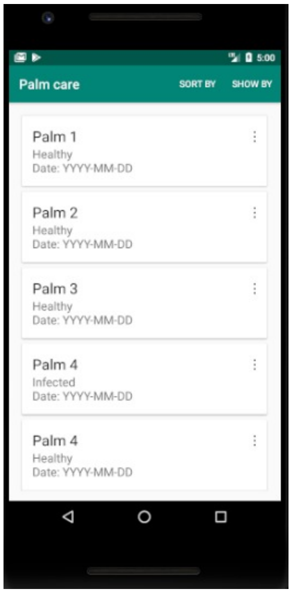


Figure 10: Sidebar



Palm 1
Healthy
Date: YYYY-MM-DD

Palm 2
Healthy
Date: YYYY-MM-DD

Palm 3
Healthy
Date: YYYY-MM-DD

Palm 4
Infected
Date: YYYY-MM-DD

Palm 4
Healthy
Date: YYYY-MM-DD

Figure 11: Show user palms

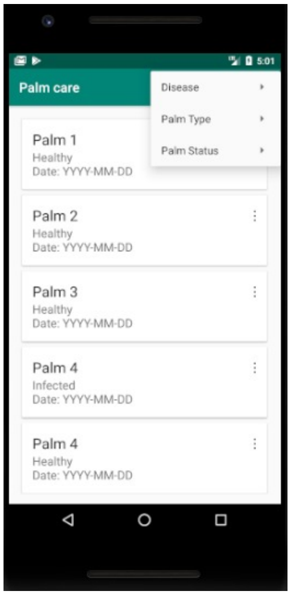


Figure 12: Show by menu

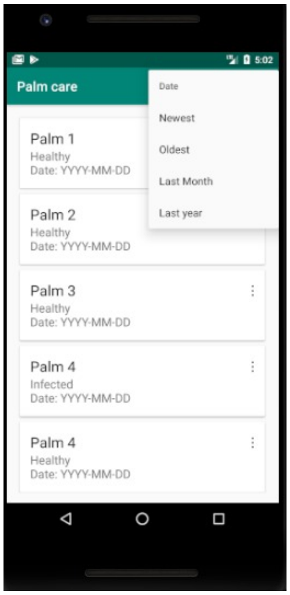


Figure 13: Sort by date menu

4.1.2 CLI

- OpenCV python installation
  - pip install numpy
  - pip install opencv
- jupyter notebook

4.1.3 API

- Keras api is used to implement CNN algorithm and use it's attributes such as resize
- sklearn is used to implemnt is used implemnt svm algorithm
- Opencv for imageprocessing techniuqes

4.1.4 Diagnostics or ROM

Not Available

4.2 Hardware Interfaces

N.A

- first normal phone camera
- secomdly a thermal seek pro portable camera to capture thermal camera

4.3 Communications Interfaces

N.A

4.4 Software Interfaces

N.A

5 Performance Requirements

image will be processed and classified in 30 seconds

6 Design Constraints

user interface must be friendly following nelson's 10 heuristics

6.1 Hardware Limitations

we need to use seekpro thermal camera which is aportable usb camera

7 Other non-functional attributes

7.1 Security

Security:

In the user registration the system hashes the user passsword and after a successfull signin/signup The system request the user to Authenticate his registered account to begin using the system options.

Furthermore, every record change is recorded through the Created_at, Updated_at and isDeleted Columns.

7.2 Reliability

1. The system operations are backed up through adding boolean isdeleted column in all database tables,once a delete operation is executed the only change in database row will be the isdeleted value.

2. The system uses realtime database through connecting to Firebase.

3. The system uses Amazon Web Service for model training to provide a classification for the infection level of infestation through the model's high accuracies achieved.

7.3 Maintainability

The system is maitained through using list of design patterns (MVC, Single Tone, Decorator , Strategy).

7.4 usability

The system will be user friendly and straight forward as palm owner may not be familiar with modern technologies, that is achieved through the following:

1. Mobile Application:
    1.1. user friendly
    1.2. multilingustic
    1.3. easy sequential capture image method.
2.nelson usability hieristics methods:
    2.1. visibilt of system status.
    2.2. matching between system and real world.
    2.3. user control and freedom.
    2.4. consistency and standards.
    2.5. Error prevention.
    2.5. Help user recognise diagnose and recover from errors.

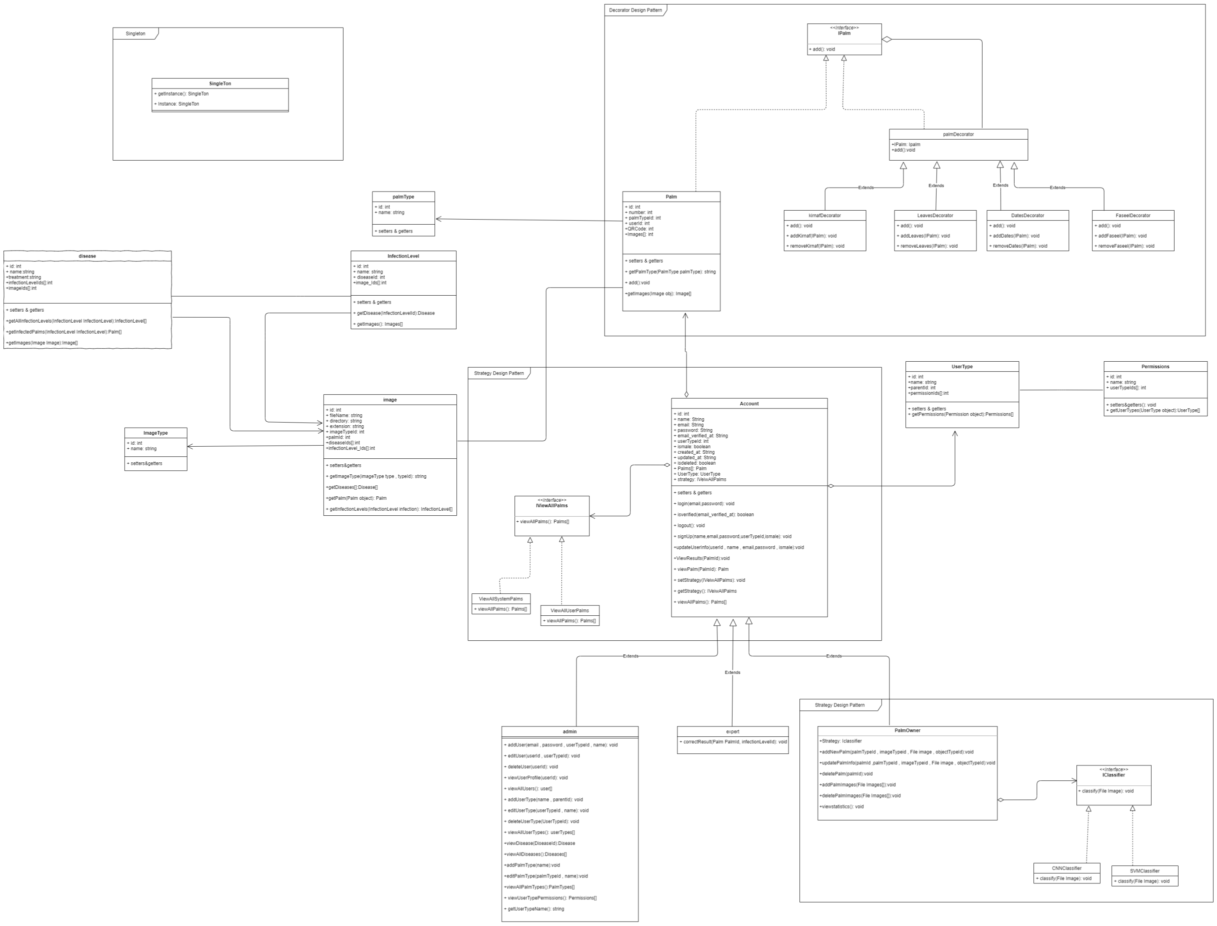
2.5. Flexibility and effiency of use.

7.5 Resource Utilization

1. Every piece of text that a user might see can be modified without changing in code.

2. Most of computations are made on the server not on the smartphone due to the leak of smartphone utilities in handling all needed computations.

8 Preliminary Object-Oriented Domain Analysis



Singleton
SingleTon
+ getInstance(): SingleTon
+ Instance: SingleTon
Decorator Design Pattern
<<Interface>>
IPalm
+ add(): void
palmDecorator
+IPalm: Ipalm
+add():void
Extends
Extends
Extends
Extends
kirnafDecorator
+ add(): void
+ addKirnaf(IPalm): void
+ removeKirnaf(IPalm): void
LeavesDecorator
+ add(): void
+ addLeaves(IPalm): void
+ removeLeaves(IPalm): void
DatesDecorator
+ add(): void
+ addDates(IPalm): void
+ removeDates(IPalm): void
FaseelDecorator
+ add(): void
+ addFaseel(IPalm): void
+ removeFaseel(IPalm): void
Palm
+ id: int
+ number: int
+ palmTypeId: int
+ userId: int
+QRCode: int
+Images[]: int
+ setters & getters
+ getPalmType(PalmType palmType): string
+ add():void
+getImages(Image obj): Image[]
palmType
+ id: int
+ name: string
+ setters & getters
disease
+ id: int
+ name:string
+treatment:string
+infectionLevelIds[]:int
+imageIds[]:int
+ setters & getters
+getAllInfectionLevels(InfectionLevel InfectionLevel):InfectionLevel[]
+getInfectedPalms(InfectionLevel InfectionLevel):Palm[]
+getImages(Image Image):Image[]
InfectionLevel
+ id: int
+ name: string
+ diseaseId: int
+image_Ids[]:int
+ setters & getters
+ getDisease(InfectionLevelId):Disease
+ getImages(): Images[]
ImageType
+ id: int
+ name: string
+ setters&getters
image
+ id: int
+ fileName: string
+ directory: string
+ extension: string
+ imageTypeId: int
+palmId: int
+diseaseIds[]:int
+infectionLevel_Ids[]:int
+ setters&getters
+ getImageType(imageType type , typeId): string
+getDiseases[]:Disease[]
+getPalm(Palm object): Palm
+ getInfectionLevels(InfectionLevel infection): InfectionLevel[]
Strategy Design Pattern
Account
+ id: int
+ name: String
+ email: String
+ password: String
+ email_verified_at: String
+ userTypeId: int
+ ismale: boolean
+ created_at: String
+ updated_at: String
+ isdeleted: boolean
+ Palms[]: Palm
+ UserType: UserType
+ strategy: IVeiwAllPalms
+ setters & getters
+ login(email,password): void
+ isverified(email_verified_at): boolean
+ logout(): void
+ signUp(name,email,password,userTypeId,ismale): void
+updateUserInfo(userId , name , email,password , ismale):void
+ViewResults(PalmId):void
+ viewPalm(PalmId): Palm
+ setStrategy(IVeiwAllPalms): void
+ getStrategy(): IVeiwAllPalms
+ viewAllPalms(): Palms[]
<<Interface>>
IViewAllPalms
+ viewAllPalms(): Palms[]
ViewAllSystemPalms
+ viewAllPalms(): Palms[]
ViewAllUserPalms
+ viewAllPalms(): Palms[]
UserType
+ id: int
+name: string
+parentId: int
+permissionIds[]:int
+ setters & getters
+ getPermissions(Permission object):Permissions[]
Permissions
+ id: int
+ name: string
+ userTypeIds[]: int
+ setters&getters(): void
+ getUserTypes(UserType object):UserType[]
Extends
Extends
Extends
admin
+ addUser(email , password , userTypeId , name): void
+ editUser(userId , userTypeId): void
+ deleteUser(userId): void
+ viewUserProfile(userId): void
+ viewAllUsers(): user[]
+ addUserType(name , parentId): void
+ editUserType(userTypeId , name): void
+ deleteUserType(UserTypeId): void
+ viewAllUserTypes(): userTypes[]
+viewDisease(DiseaseId):Disease
+viewAllDiseases():Diseases[]
+addPalmType(name):void
+editPalmType(palmTypeId , name):void
+viewAllPalmTypes():PalmTypes[]
+ viewUserTypePermissions(): Permissions[]
+ getUserTypeName(): string
expert
+ correctResult(Palm PalmId, infectionLevelId): void
Strategy Design Pattern
PalmOwner
+Strategy: Iclassifier
+addNewPalm(palmTypeId , imageTypeid , File image , objectTypeId):void
+updatePalmInfo(palmId ,palmTypeId , imageTypeid , File image , objectTypeId):void
+deletePalm(palmId):void
+addPalmImages(File Images[]):void
+deletePalmImages(File Images[]):void
+viewstatistics(): void
<<Interface>>
IClassifier
+ classify(File Image): void
CNNClassifier
+ classify(File Image): void
SVMClassifier
+ classify(File Image): void

Figure 14: Class Diagram

8.1 Inheritance Relationships

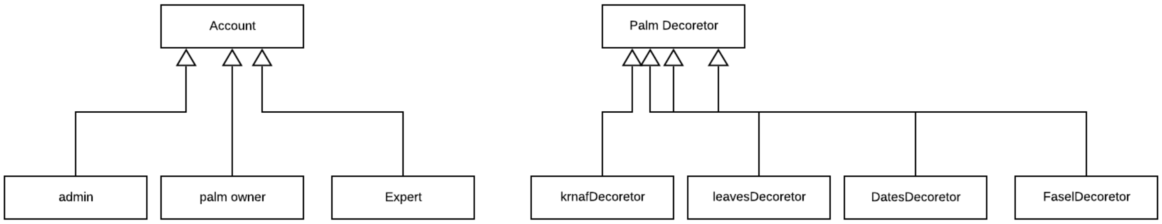


Figure 15: inheritance relation

8.2 Class descriptions

Class Name	Account
Super Class	None
Subclass	PalmOwner,expert,admin
Purpose	a class represent Account
Collaboration	aggregate with Palm,IviewAllPalms
Attributes	id: int, name: String, email: String, password: String email_verified_at: String, userTypeId: int,ismale: boolean created_at: String, updated_at: String,isdeleted: boolean Palms[]: Palm,UserType: UserType,strategy: IVeiwAllPalms
Operation	login(email,password): void, isverified(email,verified$_{a}t) : boolean$ logout(): void signUp(name,email,password,userTypeId,ismale): void,updateUserInfo(userId , name , email,password , ismale):void ViewResults(PalmId):void,viewPalm(PalmId): Palm setStrategy(IVeiwAllPalms): void,getStrategy(): IVeiwAllPalms
Constraints	user can't use the application without account

Class Name	SingleTone
Super Class	None
Subclass	None
Purpose	design pattern for database instance
Collaboration	None
Attributes	Instence:SingleTone
Operation	getInstance(): SingleTone
Constraints	None

Class Name	UserType
Super Class	None
Subclass	None
Purpose	represent different types of users
Collaboration	association with Permission and aggregate account
Attributes	id: intname: string parentId: intpermissions[]:Permission
Operation	getPermissions(Permission object):Permissions[]
Constraints	can't work without account class

Class Name	PalmOwner
Super Class	Account
Subclass	None
Purpose	a class represent Palmowner
Collaboration	extends from Account.
Attributes	Strategy: iclassifier
Operation	addNewPalm(palmTypeId , imageTypeid , File image , objectTypeId):void updatePalmInfo(palmId ,palmTypeId , imageTypeid , File image , objectTypeId):void deletePalm(palmId):void addPalmImages(File Images[]):void viewstatistics()void deletePalmImages(File Images[]):void
Constraints	user can't work without a Account

Class Name	Admin
Super Class	Account
Subclass	None
Purpose	a class represent Admin usertype
Collaboration	extends from Account class
Attributes	None
Operation	addUser(email , password , userTypeId , name): void,editUser(userId , userTypeId): void deleteUser(userId): void,viewUserProfile(userId): void viewAllUsers(): user[],addUserType(name , parentId): void, editUserType(userTypeId , name): void deleteUserType(UserTypeId): void, viewAllUserTypes(): userTypes[], viewDisease(DiseaseId):DiseaseviewAllDiseases():Diseases[], addPalmType(name):void,editPalmType(palmTypeId , name):void, viewAllPalmTypes():PalmTypes []viewUserTypePermissions(): Permissions[],getUserTypeName(): string
Constraints	Admin is the main usertype of application

Class Name	Palm
Super Class	None
subclass	None
Purpose	represent the palms
Collaboration	uses IPalm interface and aggregate Account.
Attributes	id: int, number: int palmTypeId: int, int,userId: int QRCode: File,Images[]: int
Operation	getPalmType(PalmType palmType): string add():void getImages(Image obj): Image[]
Constraints	None

Class Name	Disease
Super Class	None
subclass	None
Purpose	represent the diseases
Collaboration	assisted by infectionlevel and image class
Attributes	id: int, name:string treatment:string,infectionLevelIds[]:int imageIds[]:int
Operation	getAllInfectionLevels(InfectionLevel InfectionLevel):InfectionLevel[] getInfectedPalms(InfectionLevel InfectionLevel):Palm[] getImages(Image Image):Image[]
Constraints	can't without image

Class Name	PalmType
Super Class	None
Subclass	None
Purpose	class represent the palm types
Collaboration	associated by palm
Attributes	id: int ,name: string
Operation	None
Constraints	can't work without class palm

Class Name	Infection Level
Super Class	None
Subclass	None
Purpose	class represent the infection level of a disease
Collaboration	associated by palm,associated by Disease
Attributes	id: int,name: string,diseaseId: int,image$_{I}ds[] : int$
Operation	getInfectionLevel(Disease),getImages():Images[]
Constraints	can't work without class palm

Class Name	image
Super Class	None
Subclass	None
Purpose	a class represent Admin image class
Collaboration	association with palm,disease,infection level and imagetype
Attributes	id: int, fileName: string, directory: string extension: string, imageTypeId: int palmId: int,infectionLevel$_{I}ds[] : int, diseaseIds[] : int$
Operation	getImageType(imageType type , typeId): string, getDiseaseIds[]:Disease[],getPalm(Palm object): Palm getInfectionLevel(Infection infection): string
Constraints	can't work without palm class

Class Name	Permissions
Super Class	None
Subclass	none
Purpose	represents users Permissions
Collaboration	association with Permission
Attributes	id: intname: string name: string userTypeIds[]: int
Operation	getPermissions(Permission object):Permissions[]
Constraints	can't work without account class

Class Name	imageType
Super Class	None
Subclass	None
Purpose	class represent image different types
Collaboration	associated by Image
Attributes	id: int,name: string
Operation	None
Constraints	can't work without class image

Class Name	PalmDecroter
Super Class	None
Subclass	KernafDecroter,FasselDecroter,DatesDecroter,leavesDecroter
Purpose	class to get components of a palm
Collaboration	aggregates IViewallpalms()
Attributes	None
Operation	palmDecorator(IPalm): void,add():void
Constraints	can't work without class Palm

Class Name	KernafDecroter
Super Class	PalmDecroter
Subclass	None
Purpose	class to get kernaf component
Collaboration	inherits from PalmDecroter
Attributes	None
Operation	addKirnaf(IPalm): void,void,add(),removeKirnaf(IPalm): void
Constraints	can't work without class PalmDecorator

Class Name	leavesDecroter
Super Class	PalmDecroter
Subclass	None
Purpose	class to get leaves component
Collaboration	inherits from PalmDecroter
Attributes	None
Operation	addleaves(IPalm): void,void,add(),removeleaves(IPalm): void
Constraints	can't work without class PalmDecorator

Class Name	CNNClassifier
Super Class	implemnts IClassifier
Subclass	None
Purpose	class to implemnt
Collaboration	implents IClassifier
Attributes	None
Operation	classify(File Image)
Constraints	can't work without interface IClassifier

Class Name	SVMClassifier
Super Class	implemnts IClassifier
Subclass	None
Purpose	class to implemnt
Collaboration	implents IClassifier
Attributes	None
Operation	classify(File Image)
Constraints	can't work without interface IClassifier

Class Name	DatesDecroter
Super Class	PalmDecroter
Subclass	None
Purpose	class to get leaves component
Collaboration	inherits from PalmDecroter
Attributes	None
Operation	addDates(IPalm): void,void,add(),removeDates(IPalm): void
Constraints	can't work without class PalmDecorator

Class Name	ViewAlluserPalms
Super Class	IViewAllPalms
subclass	None
Purpose	view all palms owned by the user
Collaboration	extends IViewAllPalms
Attributes	None
Operation	viewAllPalms(): Palms[]
Constraints	can't work without IViewAllPalms

Class Name	ViewAllSystemPalms
Super Class	IViewAllPalms
subclass	None
Purpose	view all palms in the system
Collaboration	extends IViewAllPalms
Attributes	None
Operation	viewAllPalms(): Palms[]
Constraints	can't work without IViewAllPalms

Interface Name	IClassifier
Super Class	None
subclass	SVMClassifier,CNNClassifier
Purpose	interface to make use multiple calssifiers
Collaboration and aggregate palm owner implemented in	SVMClassifier CNNClassifier extends from it Palm Owner
Operation	classify(File image):
Constraints	None

Interface Name	IViewAllPalms
Super Class	None
subclass	ViewAllSystemPalms,ViewallUserPalms
Purpose	interface to view all palms
Collaboration	aggregates with account ViewAllSystemPalms ,ViewAlluserpalms extends it
implemented in	infection level
Operation	viewAllPalms(): Palms[]
Constraints	None

Interface Name	IPalm
Super Class	None
subclass	None
Purpose	help to specifypalm parts
Collaboration	aggregates with PalmDecroter
implemented in	Palm.Palm Decroter
Operation	add(): void
Constraints	can't work without palm

9 Operational Scenarios

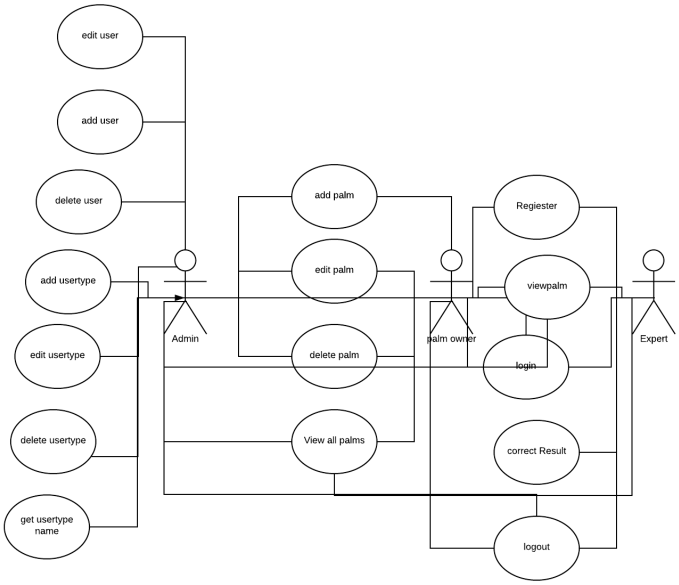


Figure 16: Use Case

-Admin: the admin have many privileges such as

- the admin can add user
- the admin can edit user
- the admin can delete user
- the admin can add a different user type
- the admin can edit any user type
- the admin can delete any user type
- the admin can add palm
- the admin can edit palm
- the admin can delete palm
- the admin can add a different palm type
- the admin can edit any palm type
- the admin can delete any palm type

- the admin can login
- the admin can logout

-Palm owner:

- the palm Owner can add palm
- the palm owner can edit palm
- the palm owner can delete palm
- the palm owner can View results of the classification
- the palm owner can Sign-up
- the palm owner can login
- the palm owner can logout

-Palm owner:

- the palm Owner can add palm
- the palm owner can edit palm
- the palm owner can delete palm
- the palm owner can view all the palms that he own
- the palm owner can View results of the classification
- the palm owner can Sign-up
- the palm owner can login
- the palm owner can logout

-Expert:

- Expert can add palm
- Expert can Sign-up
- expert can view all palms on the system
- Expert can correct the result
- the palm owner can login
- the palm owner can logout

10 Preliminary Schedule Adjusted

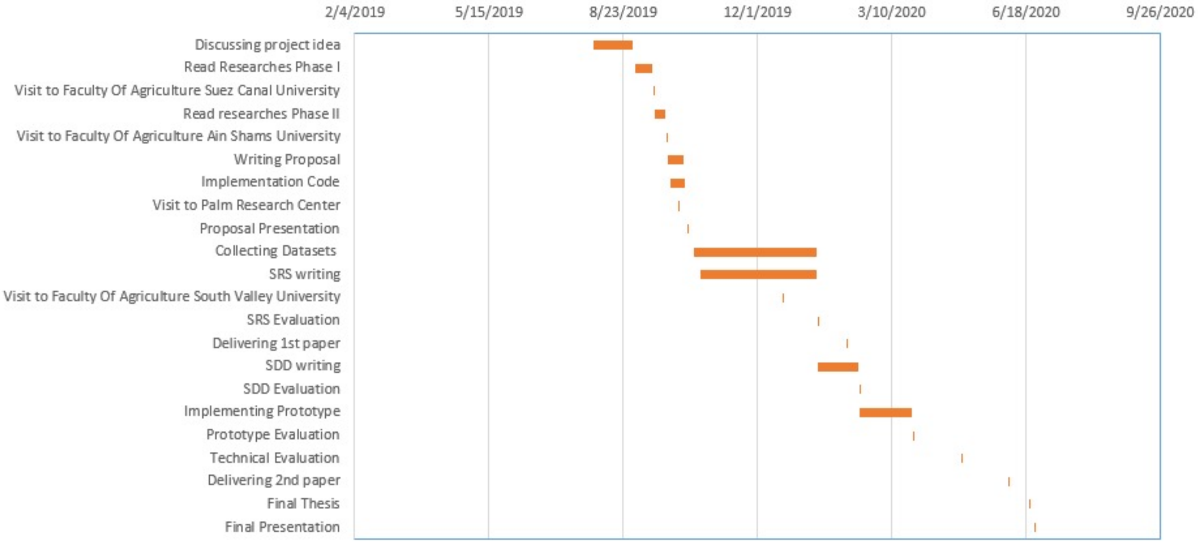


Figure 17: Gunt chart

11 Preliminary Budget Adjusted

Name	Price
Seek Thermal Compact Pro Android	EGP 7.932.00
Amazon web Service for machine learning	EGP 1.586.00
Sum	EGP 958

12 Appendices

SVM:support vector Machine.
CNN:Convolution Neural Network