Software Design Document

Hazem Alaa, Khaled Waleed, Moataz Samir, Mohamed Tarek
Eng. Hager Sobeah, Dr. Mustafa Abdul Salam

March 8, 2020

1 Introduction

1.1 Purpose

The purpose of this documentation is to present and describe the architecture and system design of our system palm care. Palm care system is a mobile application for detecting palm trees common diseases. This documentation also highlights the system components and how they interacts with each other and defines the functional requirements and their impact on the system architecture and design.

1.2 Scope

A mobile application to help palm tree owners especially palm farm owners to detect palm common diseases such as leaf spots and blight spots by using normal mobile cameras and can detect a lethal pest called Red Palm Weevil by acquiring thermal images of palm trees using thermal USB camera which can be connected to smartphones and also by using hyperspectral cameras for showing palm chlorophyll indices, then combine the results of thermal and hyperspectral imaging to provide reliable results in case of detection of RPW. Moreover providing treatments to the mentioned diseases according to palm health state. This application will ve huge amount of time and money spent on experts and traditional methods and will provide more efficient and accurate results.

1.3 Overview

This documentation 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 application 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 and data flow in details . The fifth section illustrates our component design including the used algorithms, machine learning and image processing techniques. The sixth section illustrates the application design and describes how the user will interact with our system. The seventh section is the requirement matrix that shows which components satisfy each of the functional requirements. The rest of the sections are appendices and references.

1.4 Definitions and Acronyms

Term	**Definition**
RPW	Red Palm Weevil.
SVM	Support vector machine.
CNN	Convolutional neural network
MVC	Model-View- Controller.

2 System Overview

In palm care application the user will be able to capture palm tree leaves using his mobile camera and the application will detect if the palm tree is infected with leaf spots and blight spots diseases. The user can also use the application to capture thermal and hyperspectral images with using external thermal and hyperspectral USB cameras connected to his mobile to detect RPW lethal pest. The user can also upload images instead of capturing them. The application will recommend the user to use thermal and hyperspectral images to increase the accuracy of the results concerning RPW detection. The application enhance the acquired images by using image processing techniques such as histogram normalization and image masking then apply feature extraction techniques to be classified by CNN and SVM algorithms to tell whether the palm is infected or healthy and provide the suitable treatment according to the palm state.

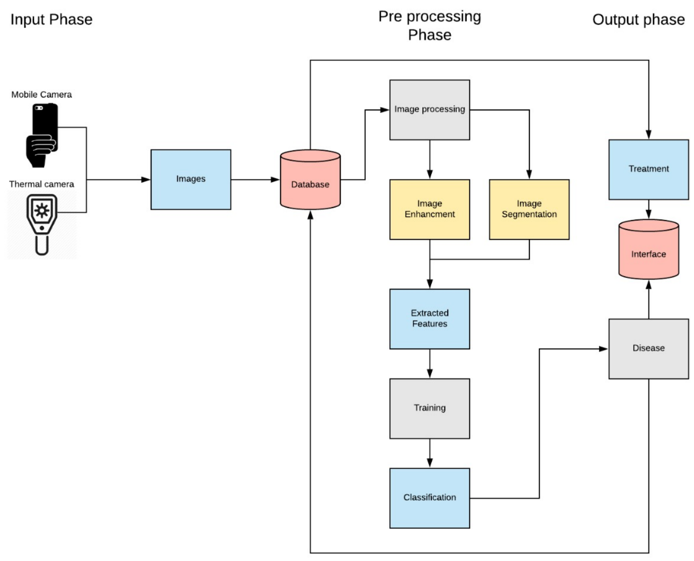


Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

our system is based on the famous design patter (MVC) Model View Controller we used because of it's critical benifits it's offer such as flexibility as the code is separated between three files that make any change won't affect the whole system but one part *Model:* the model sole purpose is to grab the data from the firebase and pass it to the controller *Controller:* the controller is the binder if the design patter as all the logic code is happening and it gets the data to the view *View:* the sole purpose of the view it to display the data after t was proccesed by the controller

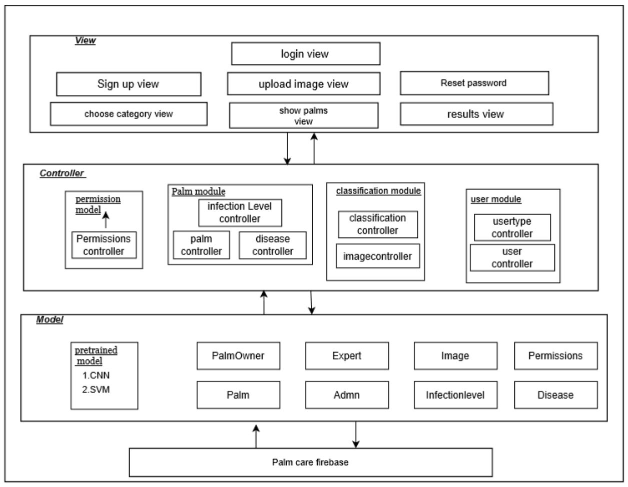


Figure 2: system Architecture Diagram

3.2 Decomposition Description

Our System use the famous design pattern (MVC) in where is the model where all the data flow happens and the view part where the user interface and lastly the controller which combine all o f the together

- *View*
  - *Login View*: where the user will allow to enter his/her username and password to use the application
  - *Sign Up View*: where the user will be able to register for the application
  - *Reset Password View*: where the user will be able to rest the password
  - *Choose category* where the user choose what part of the tree will he upload or capture
  - *Upload image View*: where the user will be able to capture image using the camera or thermal camera or upload a certain image for classification
  - *show palms view*: where the user where able to see his own palms listed
  - *results view*: where the user where be able to see the classification result and the suggested treatment if the palm where ill
- *Controller*:
  - *User Module*:

    - *User Controller*: controller has functions such as
      index() :that is responsible to get all users
      show(userid):show the user data in details
      update(userid ,[]User):is used to update users data
      create():to route to the Sign Up page
      delete(): a function where we can delete the user
      isverified():check if the user is verified return true or false.
      login(email,password):check if the user credentials match what in the database
      logout():end the session of the user

    - *UserType Controller*: controller has function such as index() :that is responsible to get all usertypes
      show(userid):show the usertype data in details
      update(usertypeid ,[]UserType):is used to update users data
      delete(usertypeid): a function where we can delete the usertype
      insert(object[]): a function where tha admin can insert a new usertype
  - *Palm Module*:
    - *Palm Controller*:
      index() :that is responsible to get all palms
      show(palmid):show the palm data in details
      update(palmid ,[]palm):is used to update palm data
      delete(palmid): a function where we can delete any palm
      insert(object[]): a function where tha user can register a new palm

    - *infectionlevel Controller*: index() :that is responsible to get all infection level
      show(infectionlevelid):show the infection level of the palm
      update(infectionlevelid ,[]infectionlevel):is used to update infection level of the palm
      delete(infectionlevelid): a function where we can delete any infection level
      getdisease(Disease disease): a function to get the disease and set it's own infection level
      getimage(Image image):get image of a certain disease with a certain infection level
      create():a route for the page for creating an infection level

    - *Disease Controller*:
      index() :that is responsible to get all diseases
      show(diseaseid):show the diseases of the palm
      update(diseaseid ,[]disease):is used to update diseases of the palm

delete(diseaseid): a function where we can delete any disease
create():a route for the page for creating a disease
insert(object[]): a function where the user can add a disease
getimages(Image []image):get all images of a certain disease
getAllInfectionLevels(InfectionLevel InfectionLevel):InfectionLevel[]: a method to get all infection levels
+getInfectedPalms(palm[] Palm):Palm[]:get all infected palms

- *Permission Module*:
  - *Permission Controller*:
    index() :that is responsible to get all permissions
    show(permissionid):show the permissions of the user
    update(permissionid ,[]Permission):is used to update Permission of the user
    delete(permissionid): a function where we can delete any permission
    create():a route for the page for creating a permission
    insert(object[]): a function where the user can add a permission to a user
    getusertype(usertypeid): a function to get a user type

- *Classification module*:
  - *Classification controller*
    CNN():to invoke the CNN Model
    SVM():to invoke the SVM model

- *Model*:
  - *Pre-trained Models*:
    - CNN: we use the our pre-trained CNN Model to detect normal images
    - SVM: we use the SVM model to classify thermal images
  - *Admin:*
    addUser(email , password , userTypeId , name): admin can add user
    editUser(userId , userTypeId): void:admin can edit user data
    deleteUser(userId): void:admin can delete a user
    viewUserProfile(userId): void: admin can view a certain user profile
    viewAllUsers(): user[]:admin can view all the users on the application
    addUserType(name , parentId): void:admin can add a new usertype
    editUserType(userTypeId , name): void:admin can edit a certain usertype
    deleteUserType(UserTypeId): void: admin can delete a usertype
    viewAllUserTypes(): userTypes[]:admin can view all types of users

on the system
viewDisease(DiseaseId):Disease: admin can view a cetain disease in details
viewAllDiseases():Diseases[]: admin can view all diseases at once
addPalmType(name):void: admin can add anew palm type to the system
editPalmType(palmTypeId , name):void: admin can edit a palm type
viewAllPalmTypes():PalmTypes[]: admin can display all type of palms at once
viewUserTypePermissions(): Permissions[]: view all user type permission
getUserTypeName(): string:get user type name

– *Palm owner:*
addNewPalm(palmTypeId , imageTypeid , File image , objectTypeId):void: the palm owner/owner of the field can add a new palm with an image
updatePalmInfo(palmId ,palmTypeId , imageTypeid , File image , objectTypeId):void: the palm owner can update his palms information
deletePalm(palmId):void:a palm owner can delete any palm he own
addPalmImages(File Images[]):void:he can add a new imges for palm
deletePalmImages(File Images[]):void: a palm owner can delete palm images
viewstatistics(): void: view statistics or results
– *Expert*:
correctResult(Palm PalmId, infectionLevelId): void: an expert can correct a result that was wrong by the model
– *Palm*
getPalmType(PalmType palmType): string:get a certain palm type
add():void: add a palm
getImages(Image obj): Image[]: get images

– *image:*
getImageType(imageType type , typeId): string:get image type to get to classification getDiseases[]:Disease[]:get the diseases in the image getPalm(Palm object): Palm:get the palm that is in the image
getInfectionLevels(InfectionLevel infection): InfectionLevel[]:get infection levels in the image

– *Infectionlevel*
getDisease(InfectionLevelId):Disease:get the disease in an certain infection level
getImages(): Images[]:get images of an infection level diseases

– *Permissions*:
getUserTypes(UserType object):UserType[]:get all usertypes

– *Disease*
getAllInfectionLevels(InfectionLevel InfectionLevel):InfectionLevel[]:get all infected levels in disease
getInfectedPalms(InfectionLevel InfectionLevel):Palm[]:get all infected palms getImages(Image Image):Image[]:get images of the disease

3.2.1 class diagram

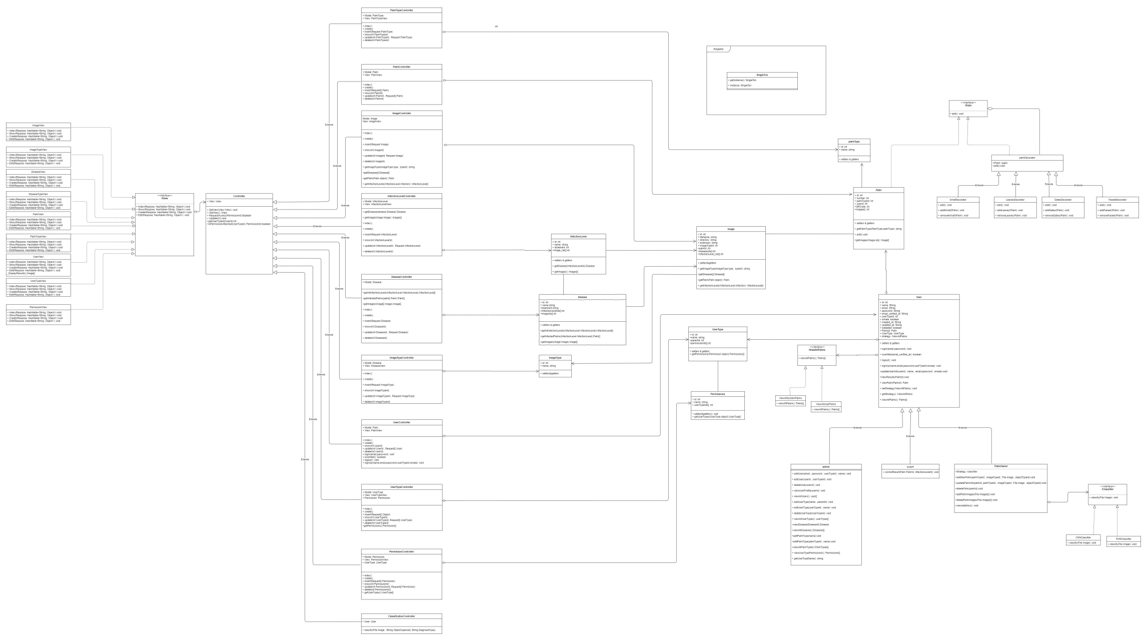


Figure 3: class diagram

Class Name	User
Super Class	None
Subclass	PalmOwner,expert,admin
Purpose	a class model to get all data of the user
Collaboration	aggregate with Palm,IviewAllPalms and usertype
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$_at) : boolean$ 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
Constraints	user is one of the core classess of the application

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 Permissions , aggregate user and usertype controller
Attributes	id: int name: string parentId: intpermissions[]:Permissions
Operation	getPermissions(Permission object):Permissions[]
Constraints	can't work without user class

Class Name	PalmOwner
Super Class	user
Subclass	None
Purpose	a class represent Palmowner
Collaboration	extends from user.
Attributes	None
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 user

Class Name	Admin
Super Class	user
Subclass	None
Purpose	a class represent Admin user type
Collaboration	extends from user 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 infection level ,image class aggregate infection level controller
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$_Ids[] : int$
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	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	SVM
Super Class	Iclassifier
subclass	None
Purpose extends IClassifier	use SVM classify Collaboration
Attributes	None
Operation	Classify(): File image
Constraints	can't work without Iclassiefy

Class Name	CNN
Super Class	Iclassifier
subclass	None
Purpose extends IClassifier	use CNN classify Collaboration
Attributes	None
Operation	Classify(): File image
Constraints	can't work without Iclassiefy

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

Class Name	Classification Controller
Super Class	Controller
Subclass	None
Purpose	is used to invoke different ai model
Collaboration	extends controller
Attributes	User user
Operation	+ classify(File image , String ObjectCaptured, String DiagnoseType);
Constrains	can't without Iclassifier interface

Class Name	Permission controller
Superclass	Controller
subclass	None
Purpose	used to control user permissions
Collabirations	aggregate usertype, extends controller
Attributes	+ Model: Permission + View: PermissionView + UserType: UserType

operations	+ index()
	+ create()
	+ insert(Request[] Permission)
	+ show(int PermissionId)
	+ update(int PermissionId, Request[] Permission)
	+ delete(int PermissionId)]
	+ getUserTypes():UserType[]

Constrains	can't work without permissions model

Class Name	usertypecontroller
Superclass	Controller
subclass	None
Purpose	used to control user types logic
Collabirations	aggregate usertype, extends controller
Attributes	+ Model: UserType + View: UserTypeView + Permission: Permission

operations	+ index()+ create()
	+ insert(Request[] Object)+ show(int UserTypeId)
	+ update(int UserTypeId, Request[] UserType)
	+ delete(int UserTypeId)
	+getPermissions():Permission[]

Constrains	can't work without usertype model

Class Name	user controller
Superclass	Controller
subclass	None
Purpose	used to control users logic
Collabirations	aggregate user, extends controller
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
operations	+ 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[]
Constrains	can't work without usermodel

Class Name	imagetypecontroller
Superclass	Controller
subclass	None
Purpose	used to control image typelogic
Collabirations	aggregate image type,Disease extends controller
Attributes	+ Model: Disease + View: DiseaseView

operations	+ index()
	+ create()
	+ insert(Request ImageType)
	+ show(int ImageTypeId)
	+ update(int ImageTypeId , Request ImageType)
	+ delete(int ImageTypeId)

Constrains	can't work without image type model

Class Name	diseasecontroller
Superclass	Controller
subclass	None
Purpose	used to control diseases logic
Collabirations	aggregate ,disase extends controller
Attributes	+ Model: Disease

operations	+getAllInfectionLevels(InfectionLevel InfectionLevel):InfectionLevel[]
	+getInfectedPalms(palm[] Palm):Palm[]
	+getImages(Image[] Image):Image[]
	+ index()
	+ create()
	+ insert(Request Disease)
	+ show(int DiseaseId)
	+ update(int DiseaseId , Request Disease)
	+ delete(int DiseaseId)

Constrains	can't work without disease model

Class Name	infectionlevelcontroller
Superclass	Controller
subclass	None
Purpose	used to control infection levels of the diseases logic
Collabirations	aggregate ,infection level extends controller
Attributes	+ Model: infection level + View: InfectionLevelView

operations	+ getDisease(disease Disease):Disease
	+ getImages(mage Image): Images[]
	+ index()
	+ create()
	+ insert(Request InfectionLevel)
	+ show(int InfectionLevelId)
	+ update(int InfectionLevelId , Request InfectionLevel)
	+ delete(int InfectionLevelId)

Constrains	can't work without infectionlevel model

Class Name	imagecontroller
Superclass	Controller
subclass	None
Purpose	used to control imagelogic
Collabirations	aggregate ,infection level extends controller
Attributes	Model: image View: imageView

operations	+ index()
	+ create()
	+ insert(Request Image)
	+ show(int imageId)
	+ update(int imageId, Request image)
	+ delete(int imageId)
	+ getImageType(imageType type , typeId): string
	+getDiseases[]:Disease[]
	+getPalm(Palm object): Palm
	+ getInfectionLevels(InfectionLevel infection): InfectionLevel[]

Constrains	can't work without imagemodel

Class Name	palmcontroller
Superclass	Controller
subclass	None
Purpose	used to control palm logic
Collabirations	aggregate ,palm extends controller
Attributes	+ Model: Palm + View: PalmView

operations	+ index()
	+ create()
	+ insert(Request[] Palm)
	+ show(int PalmId)
	+ update(int PalmId , Request[] Palm)
	+ delete(int PalmId)

Constrains	can't work without palm model

Class Name	palmtypecontroller
Superclass	Controller
subclass	None
Purpose	used to control palmtype logic
Collabirations	aggregate ,palmtype extends controller
Attributes	+ Model: Palmtype + View: PalmtypeView

operations	+ index()
	+ create()
	+ insert(Request[] Palm)
	+ show(int PalmId)
	+ update(int PalmId , Request[] Palm)
	+ delete(int PalmId)

Constrains	can't work without palmtype model

Class Name	controller
Superclass	none
subclass	palmtypecontroller palmcontroller imagecontroller imagetypecontroller usercontroller usertypecontroller
Purpose	a binder for all controllers
Collabirations	all controller extends it
Attributes	+ View: Iview
operations	+ SetView(View IView): void + GetView(): IView + RequestAccess(PermissionId):Boalean + UpdateUI():void +getUserTypeId(UserId):int +isPermissionAttached(UserTypeId, PermissionId):boalean
Constrains	the design pattern and systen won't work without it

Class Name	ImageView
Superclass	implements Iview
subclass	None
Purpose	it's image UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	ImagetypeView
Superclass	implements Iview
subclass	None
Purpose	it's imagetype UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	Diseaseview
Superclass	implements Iview
subclass	None
Purpose	it's disease UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	Diseasetypeview
Superclass	implements Iview
subclass	None
Purpose	it's disase type UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	PalmView
Superclass	implements Iview
subclass	None
Purpose	it's Palm UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	palmtypeView
Superclass	implements Iview
subclass	None
Purpose	it's palmtype UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

3.3 Operational Scenarios

3.3.1 Sequence diagram

Class Name	userview
Superclass	implements Iview
subclass	None
Purpose	it's user UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	usertypeView
Superclass	implements Iview
subclass	None
Purpose	it's usertype UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

Class Name	permissionview
Superclass	implements Iview
subclass	None
Purpose	it's permissions UI
Collabirations	implementsIview
Attributes	None
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without Iview

interfaceName	Iview
Superclass	controller
Purpose	it's views binder
Collabirations	every view implents it
operations	+ Index(Respose: Hashtable<String, Object>):void + Show(Respose: Hashtable<String, Object>) void + Create(Respose: Hashtable<String, Object>):void + Edit(Respose: Hashtable<String, Object>): void
Constrains	can't work without controller

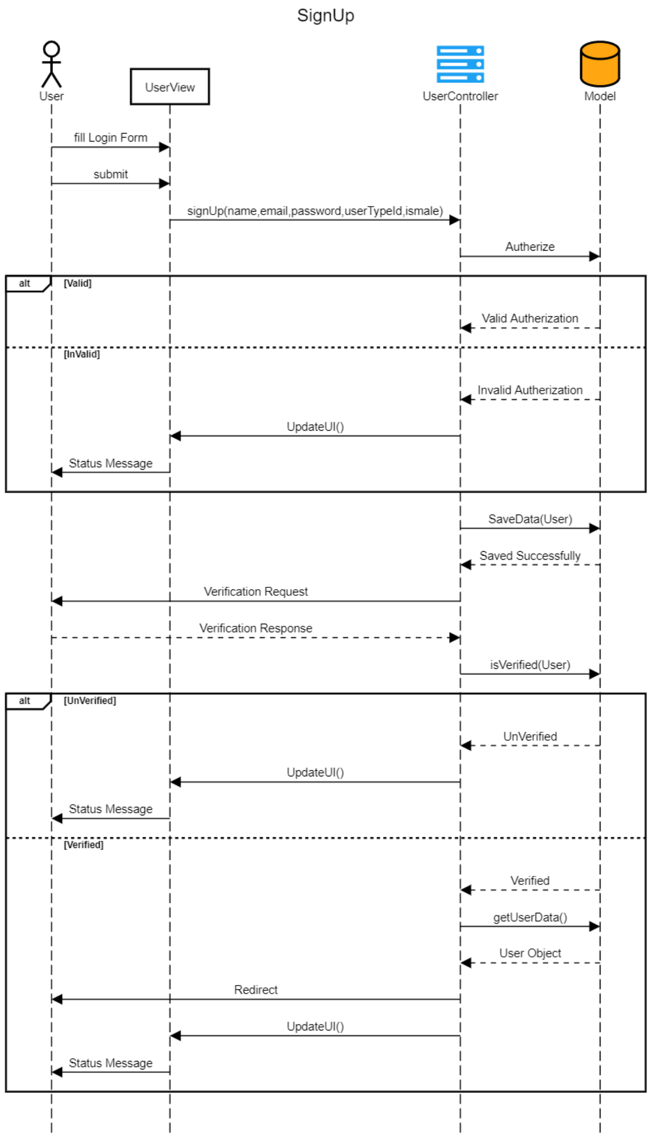


Figure 4: Sign up

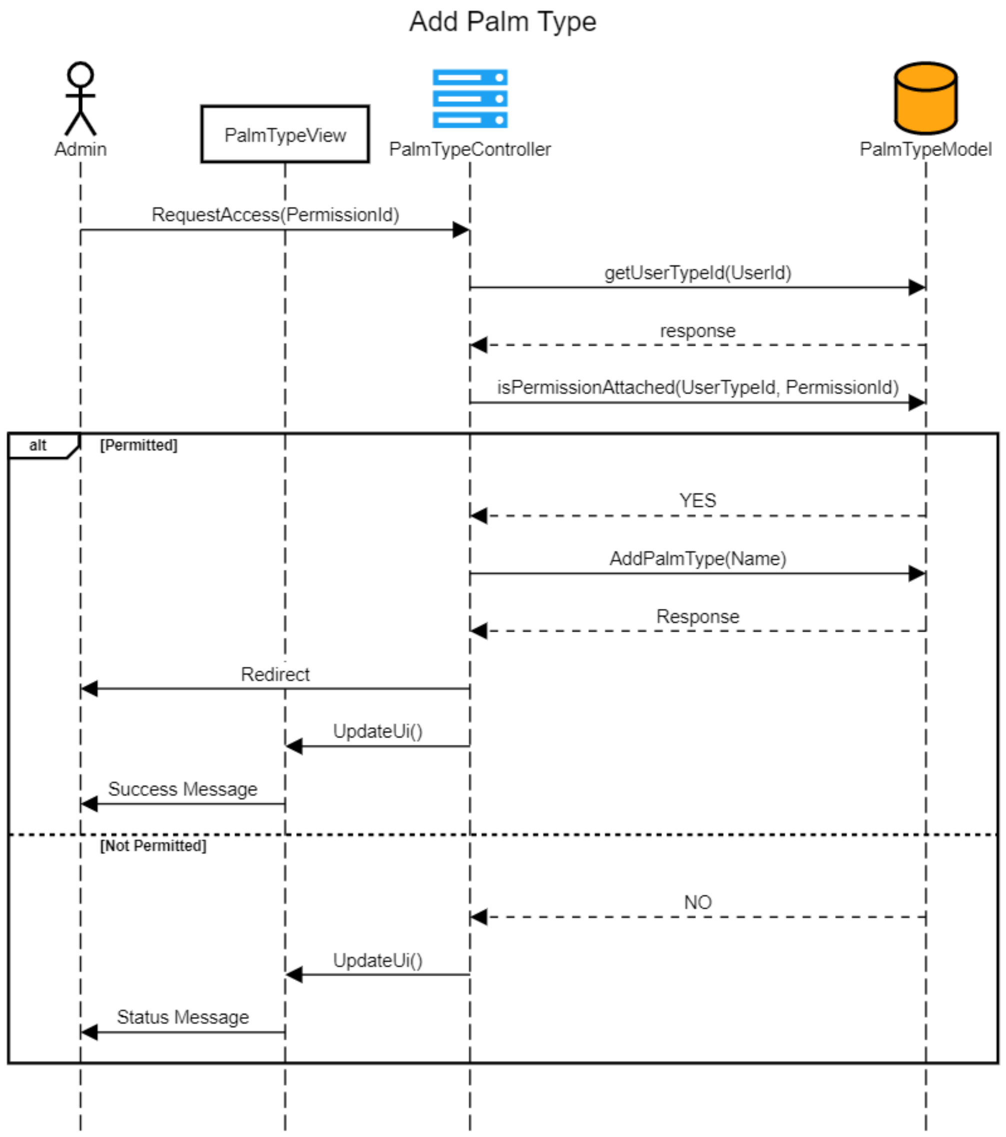


Figure 5: Add Palm Type

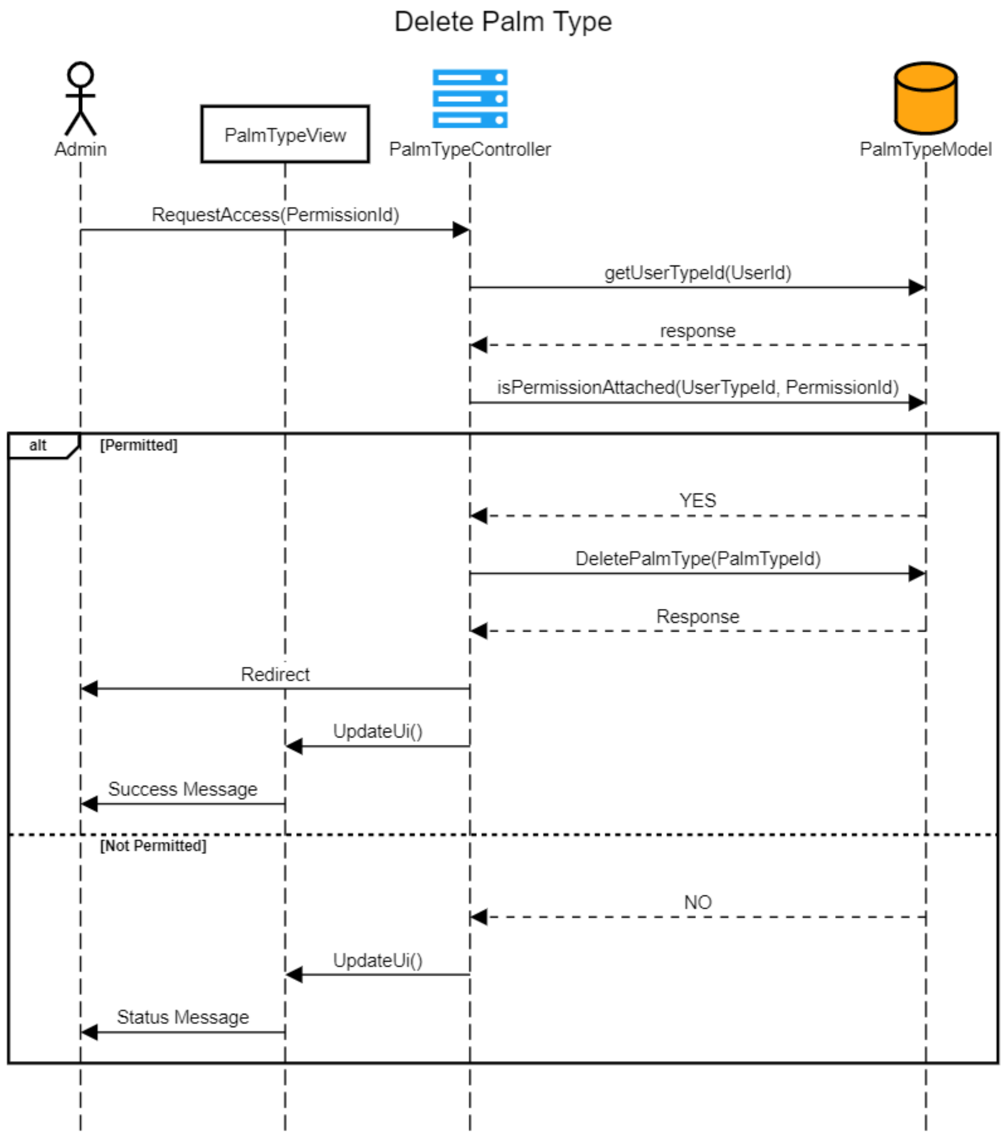


Figure 6: Delete Palm type

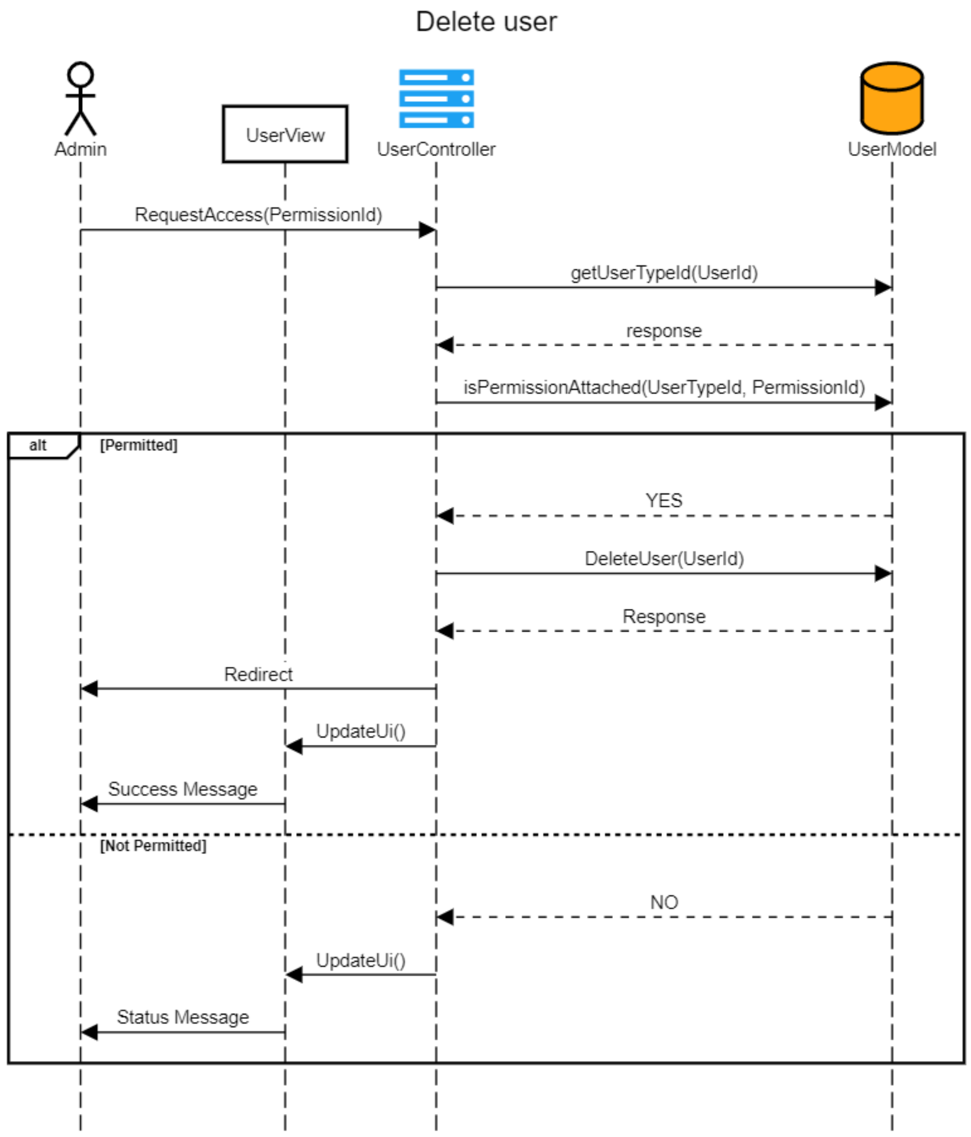


Figure 7: Delete user

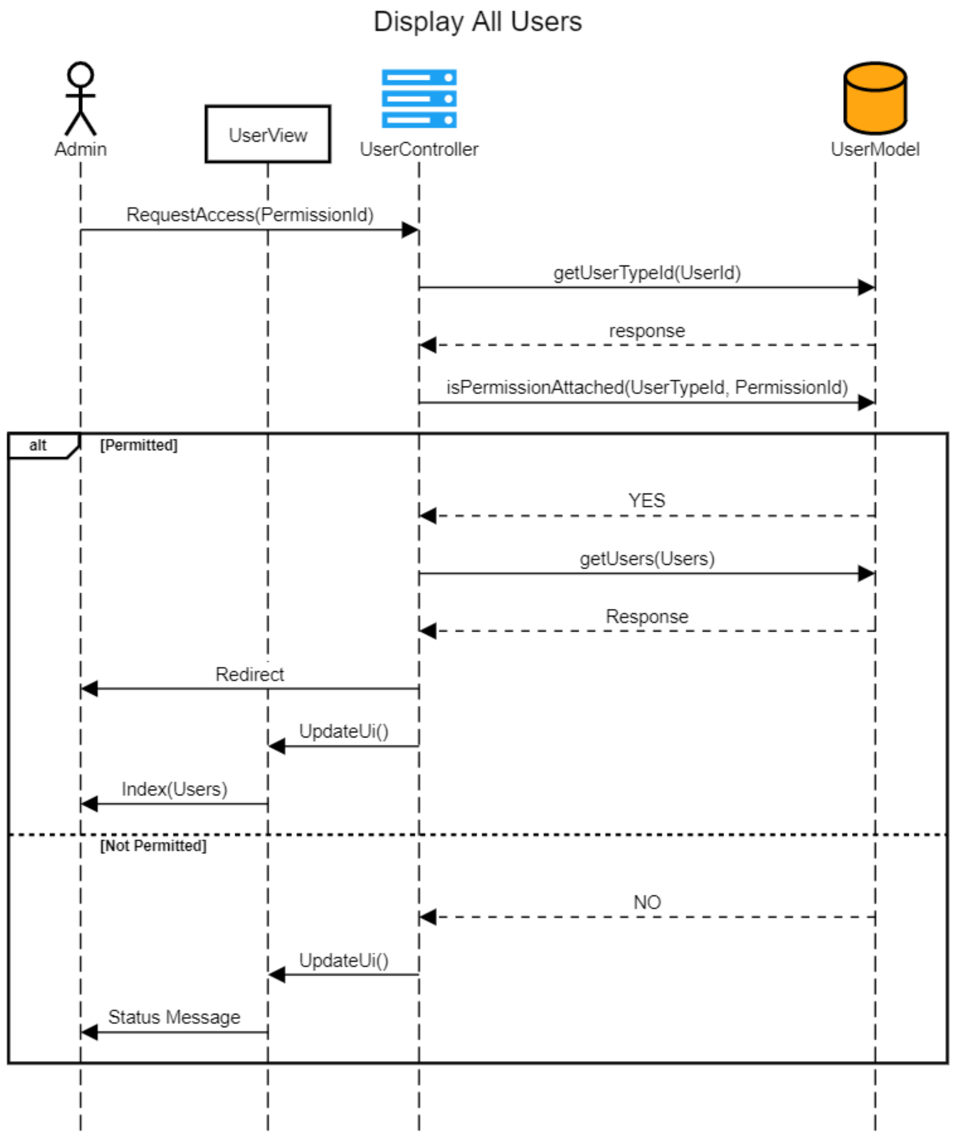


Figure 8: Display all users

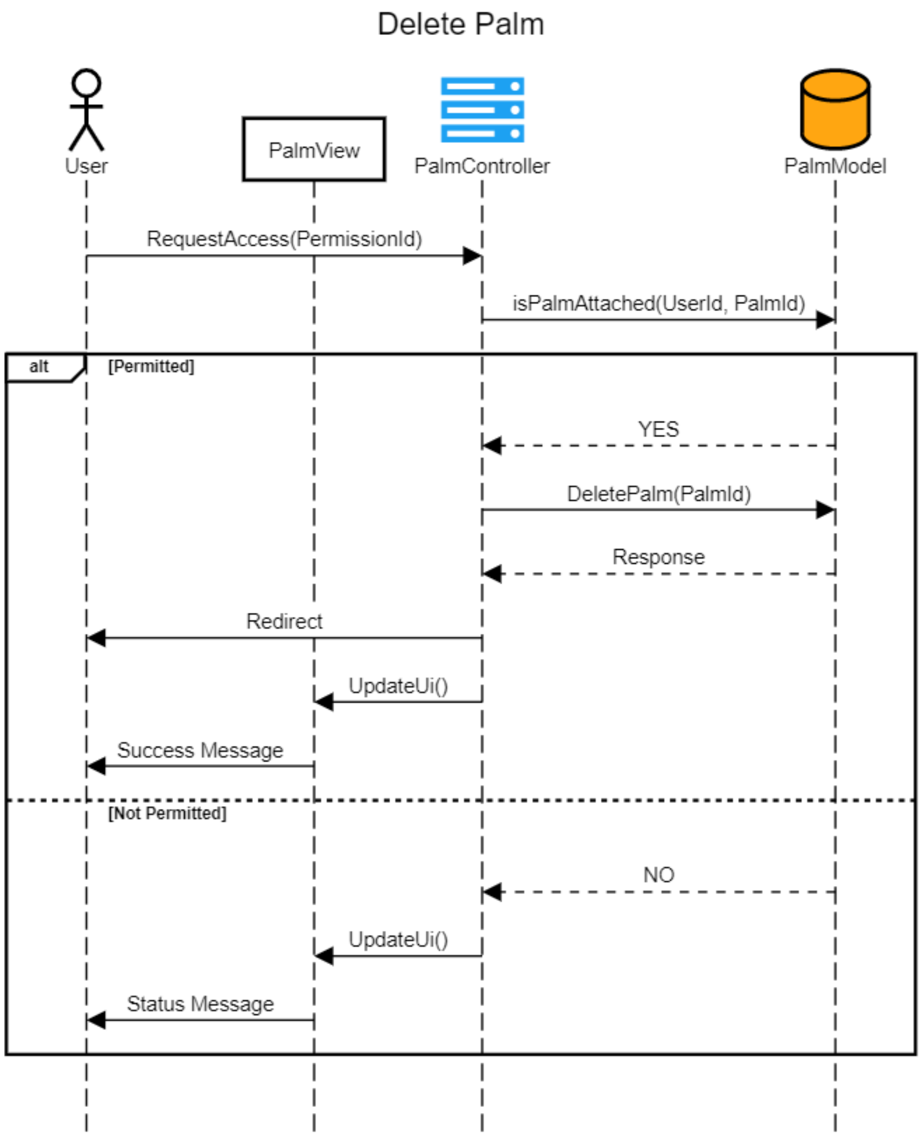


Figure 9: Delete Palm

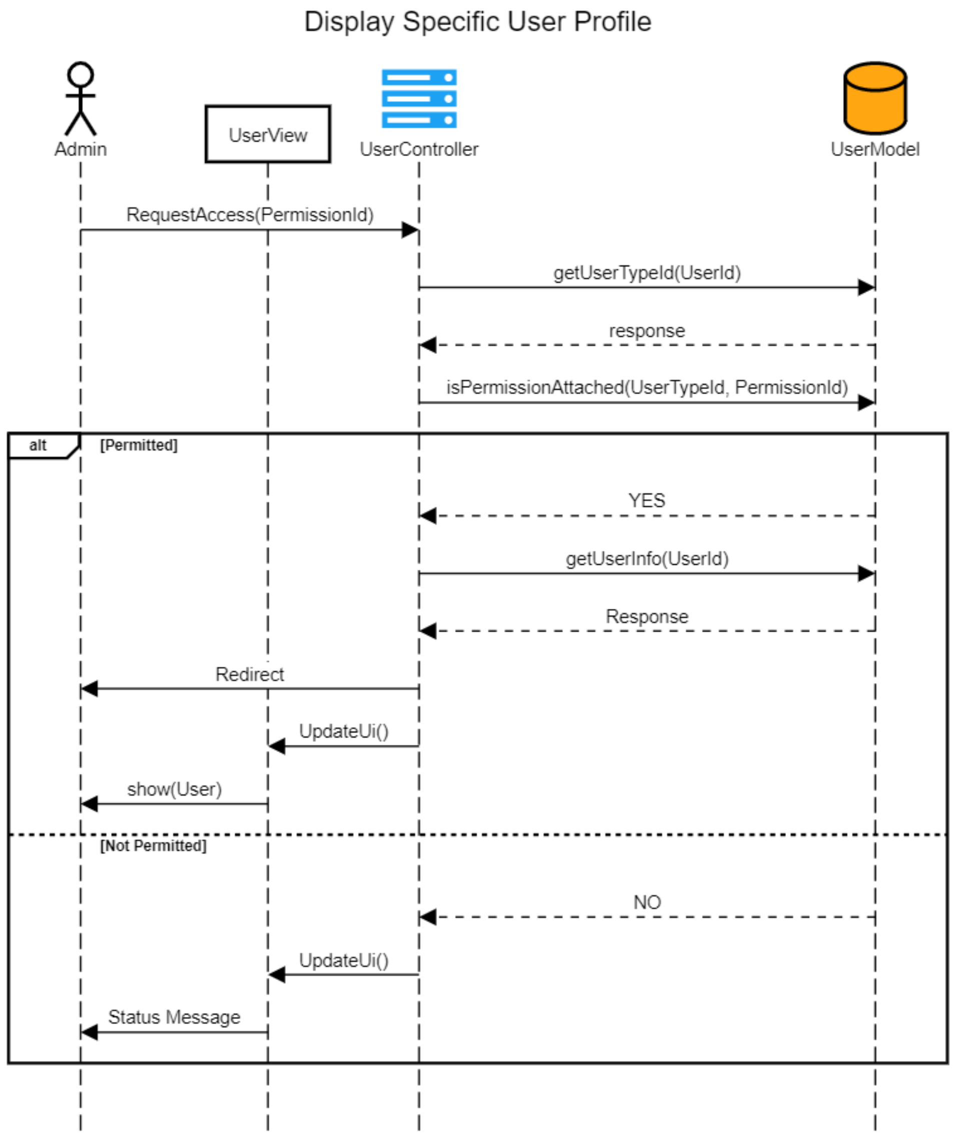


Figure 10: Display specific user profile

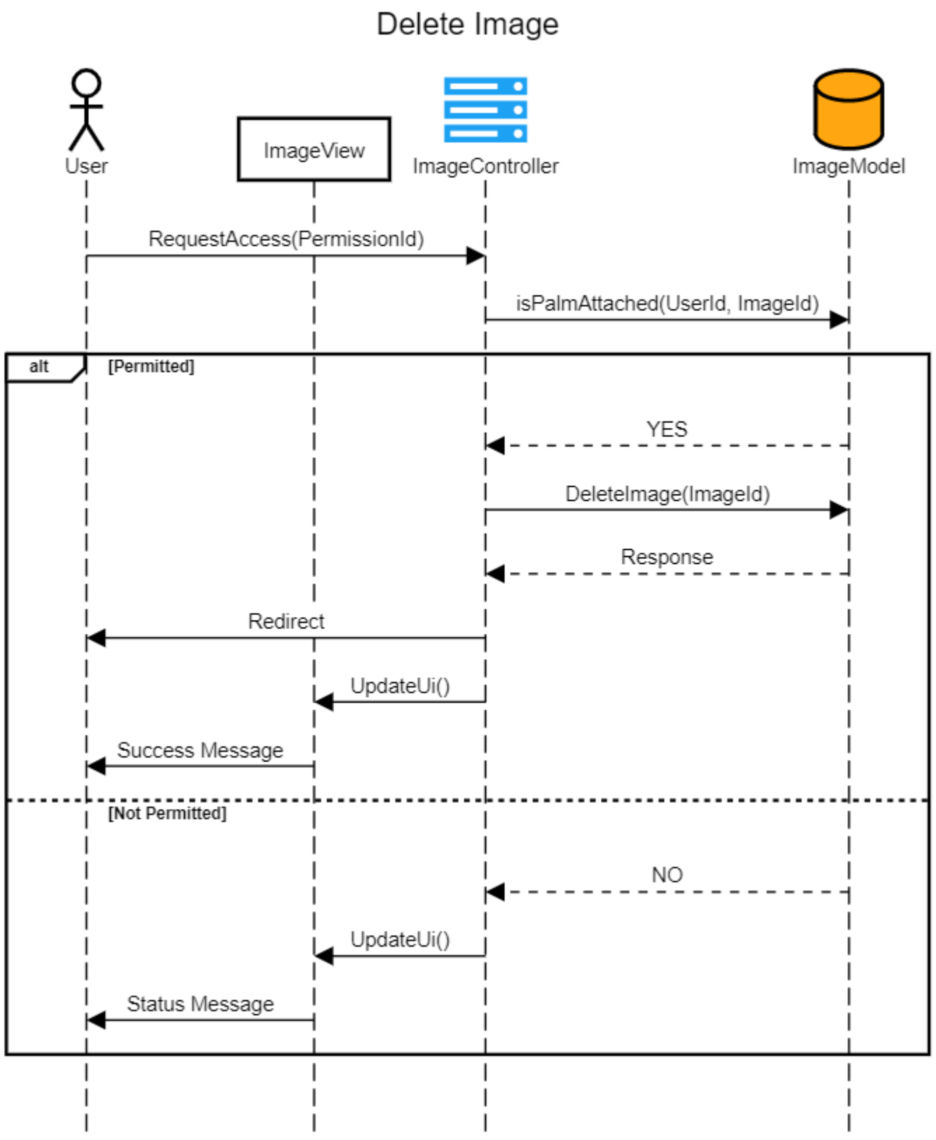


Figure 11: Delete Image

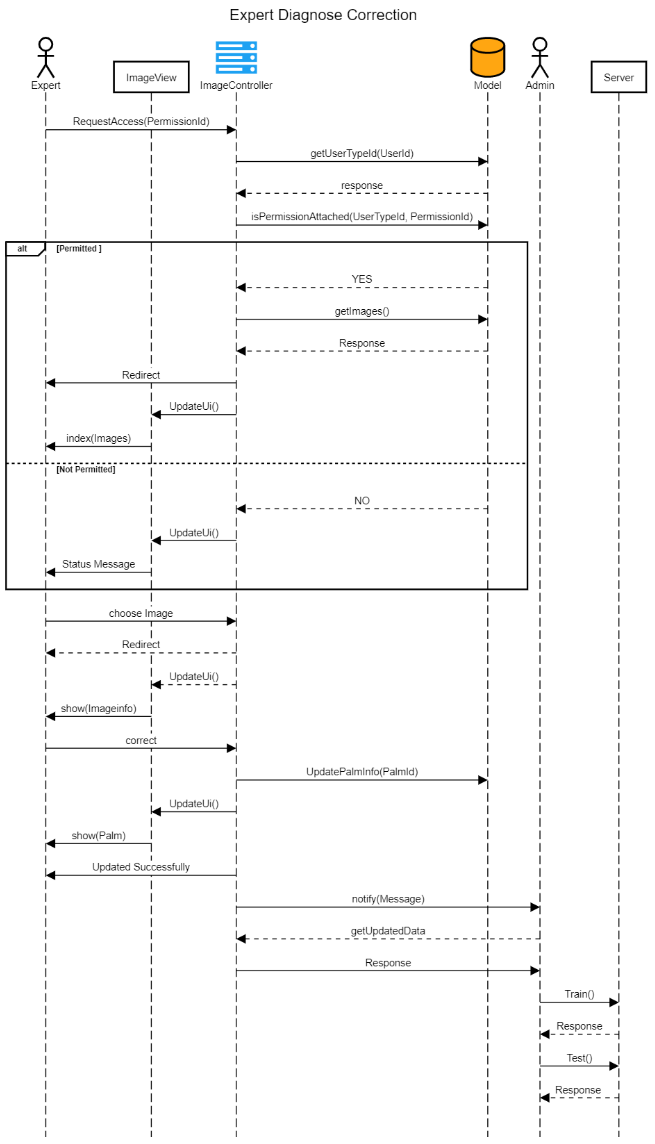


Figure 12: Expert Diagnose Correction

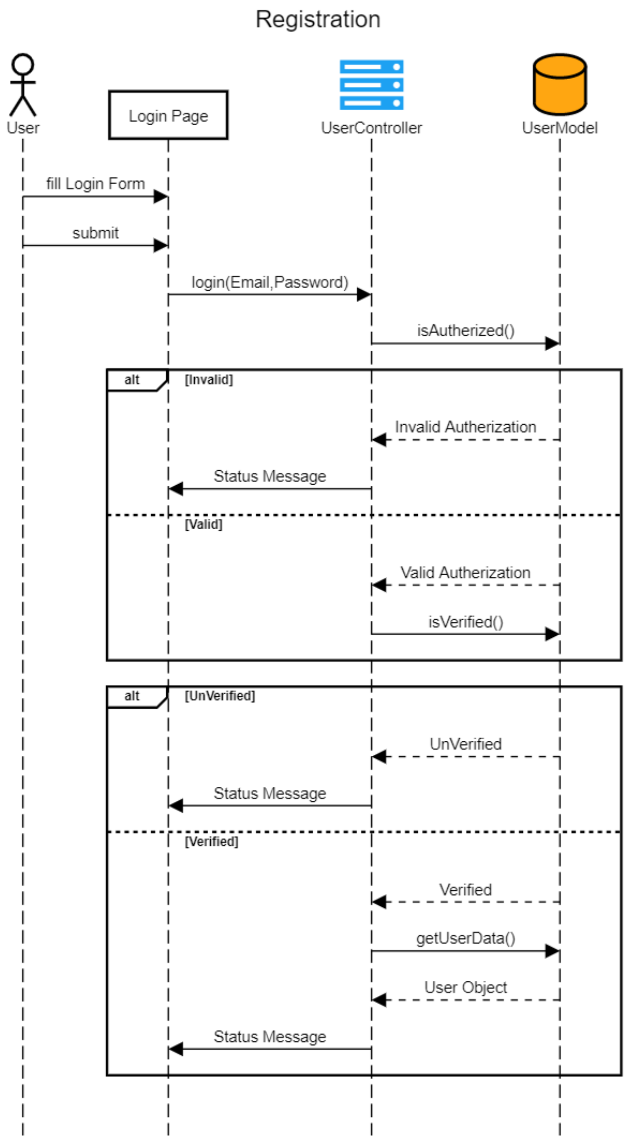


Figure 13: Registration

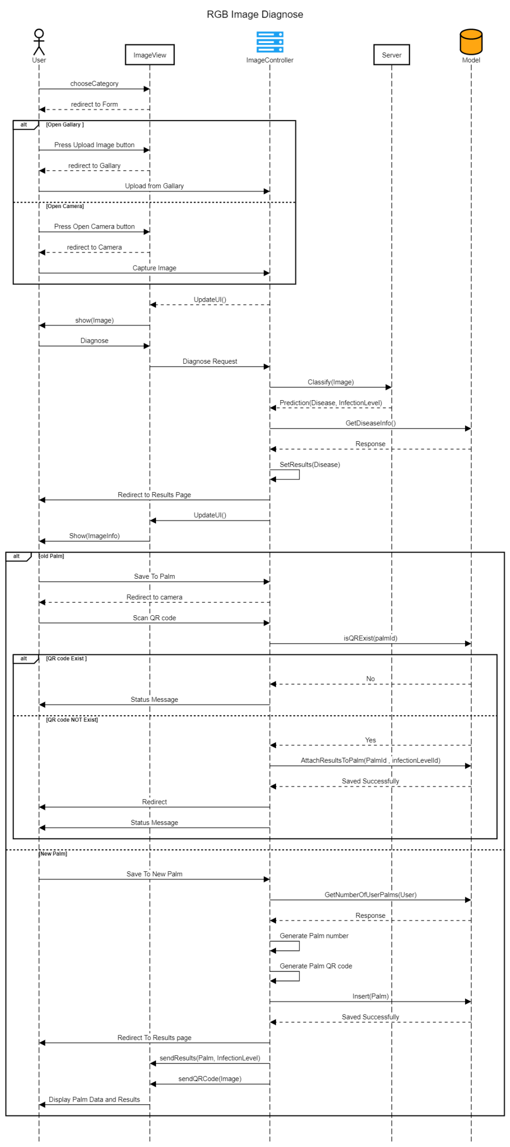


Figure 14: RGB Image Diagnose

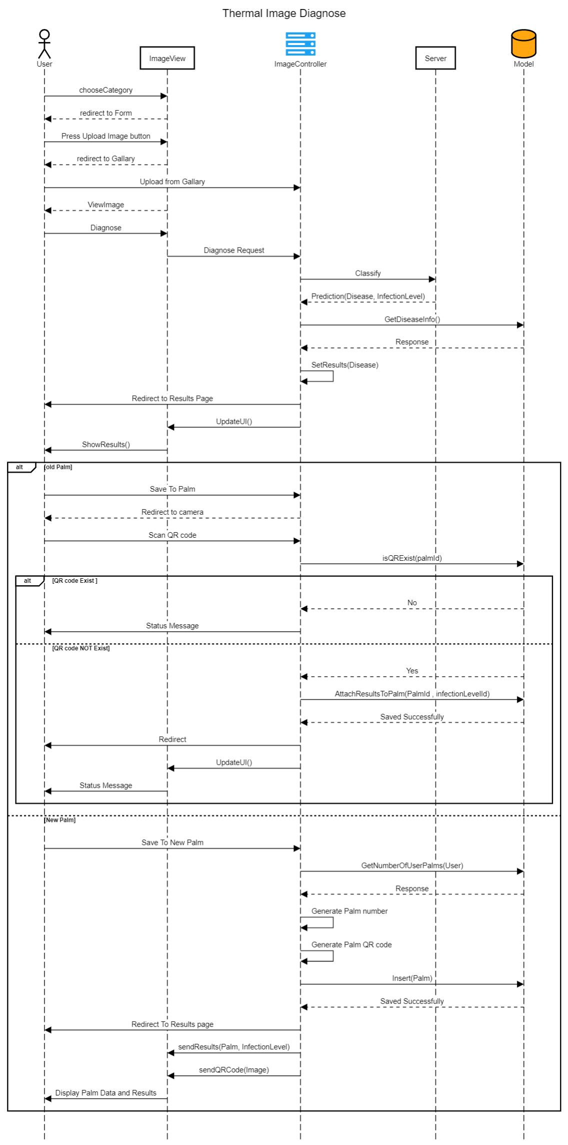


Figure 15: Thermal Image Diagnose

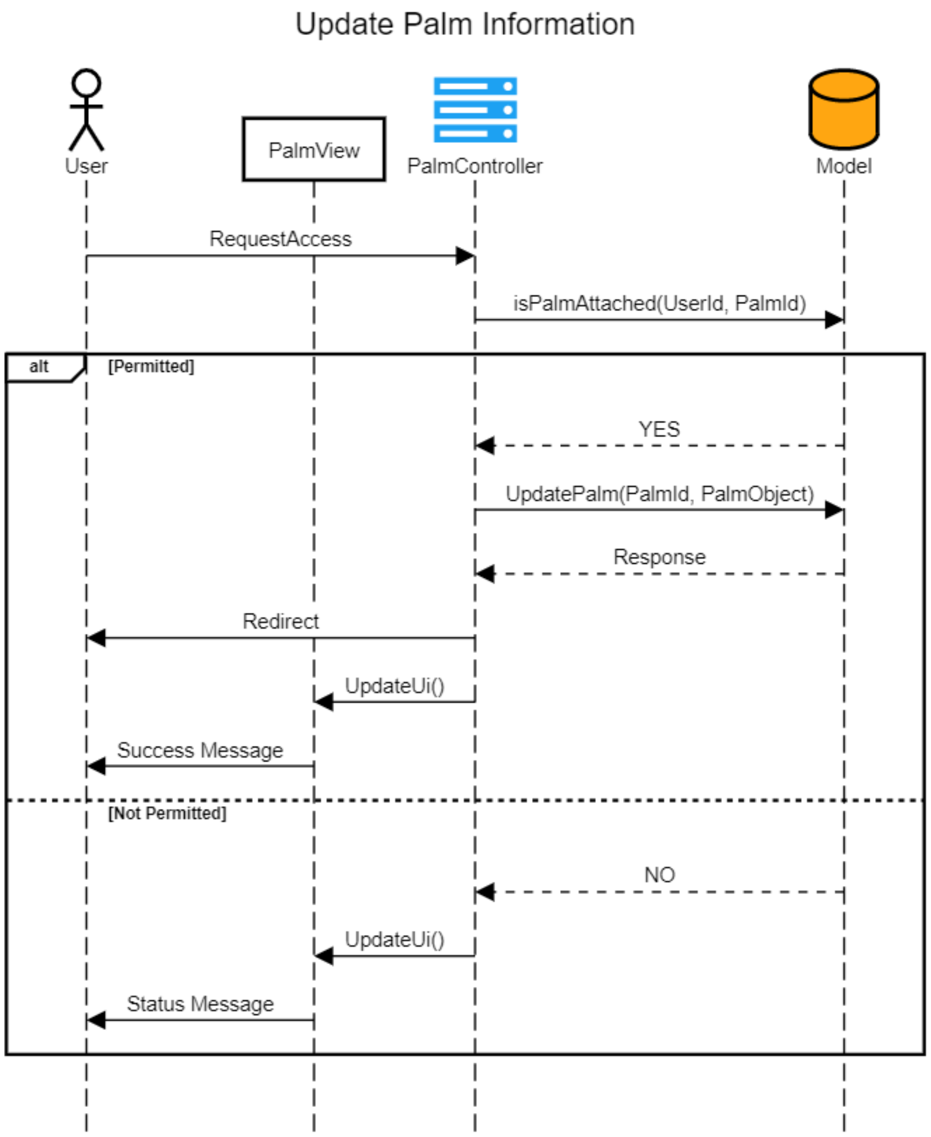


Figure 16: Update Palm Information

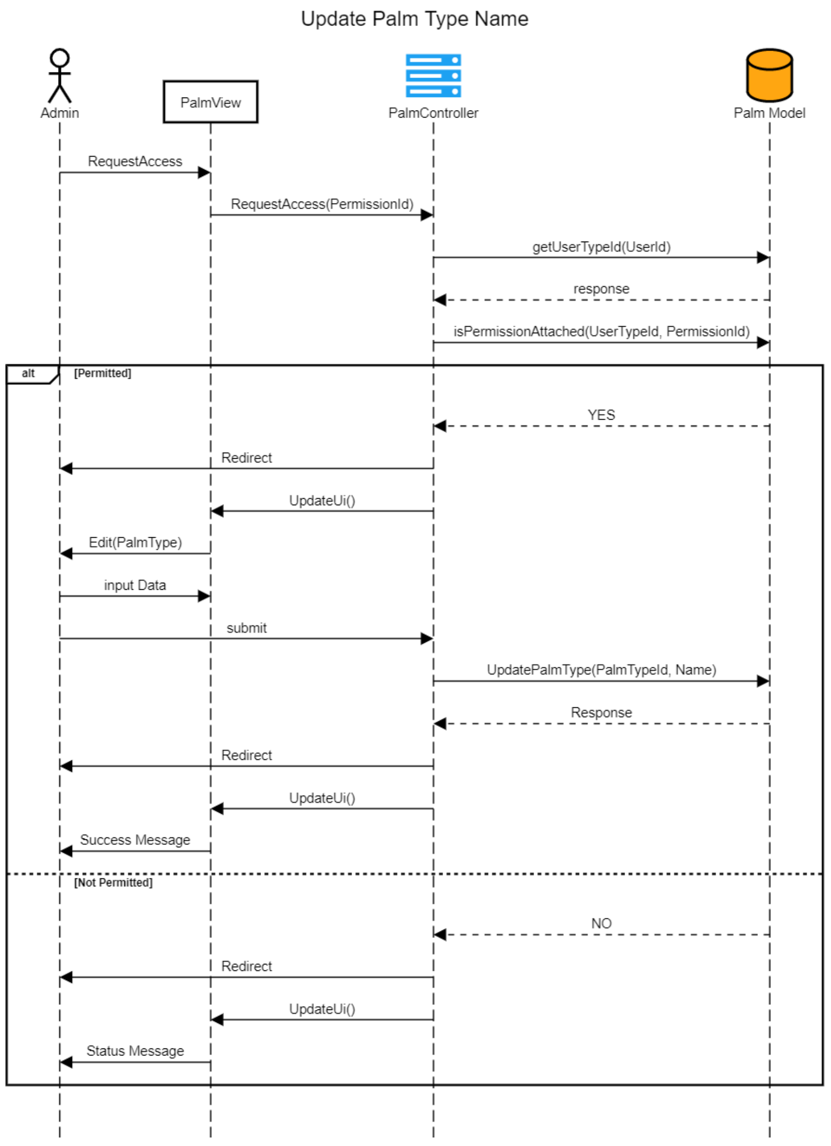


Figure 17: Update Palm Type Name

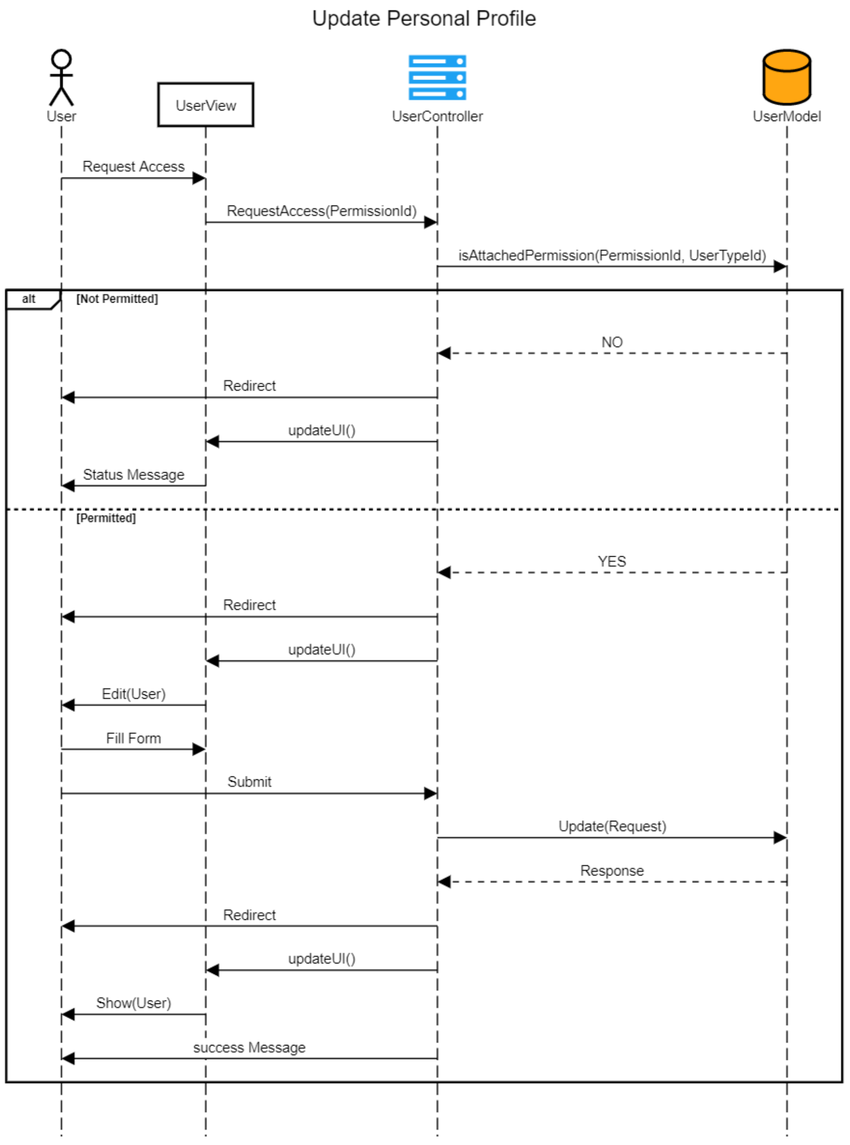


Figure 18: Update Personal Profile

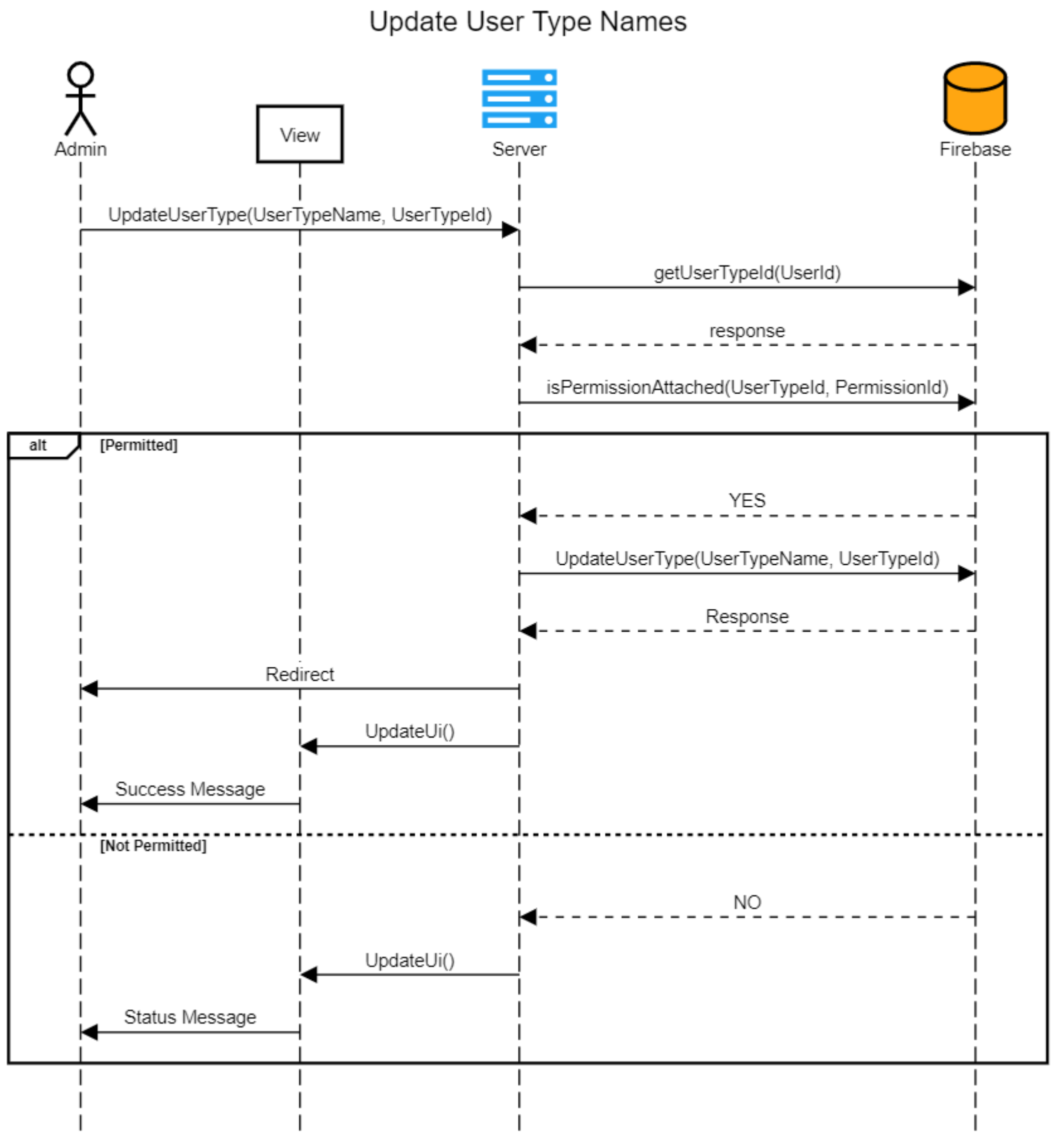


Figure 19: Update User Type Names

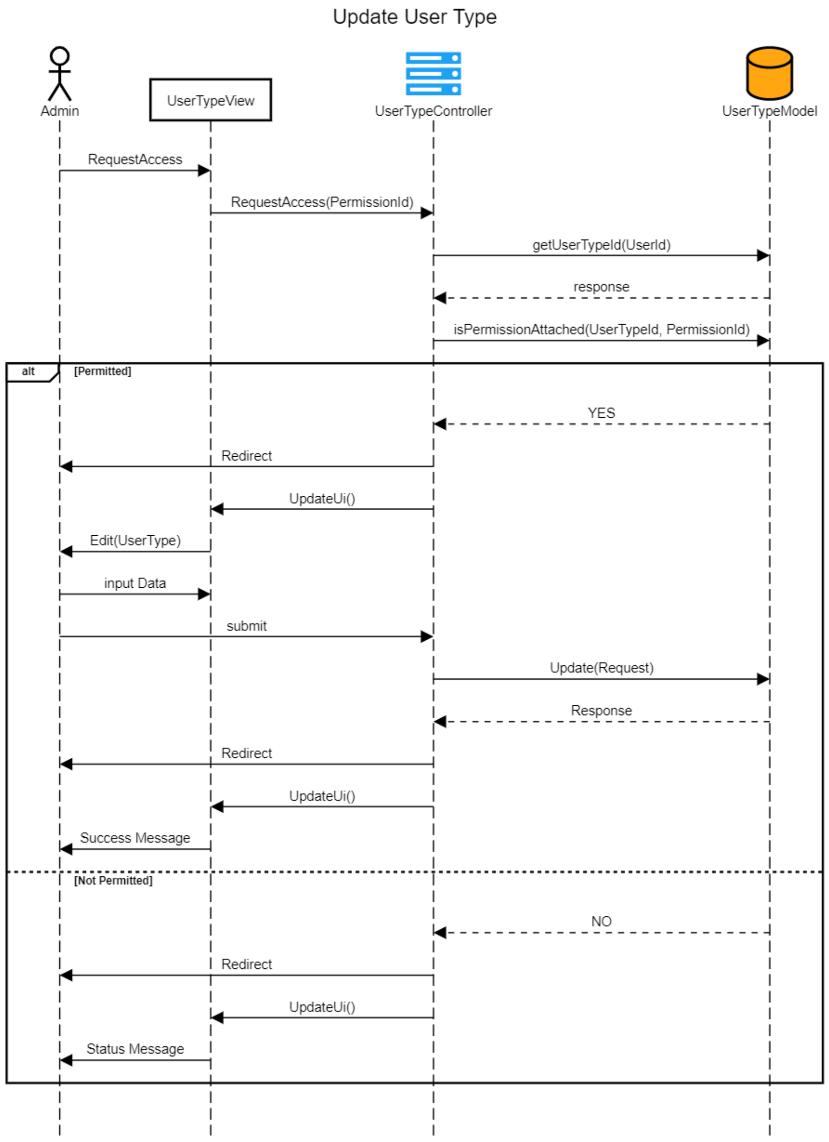


Figure 20: Update User Type

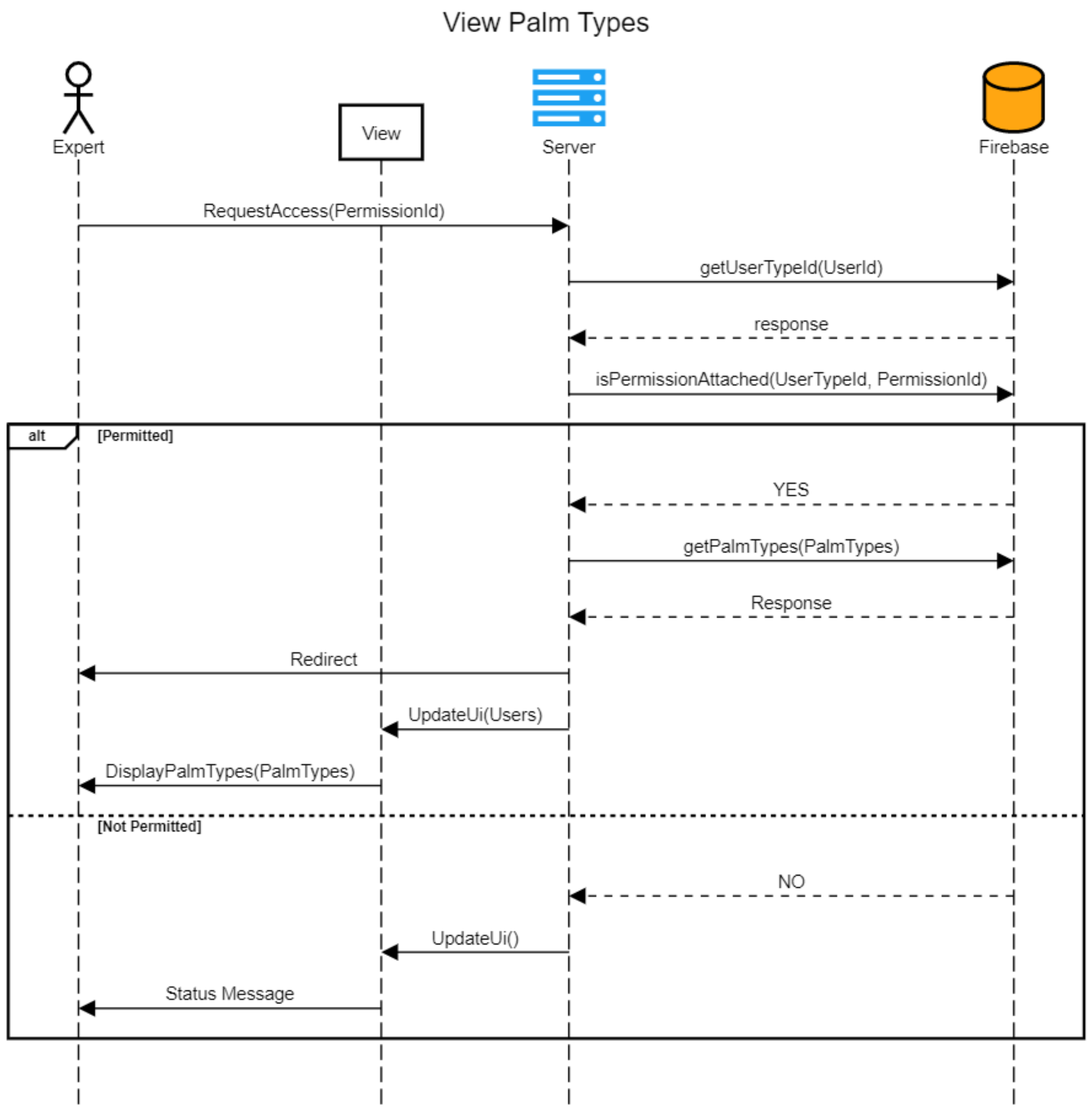


Figure 21: View palm types

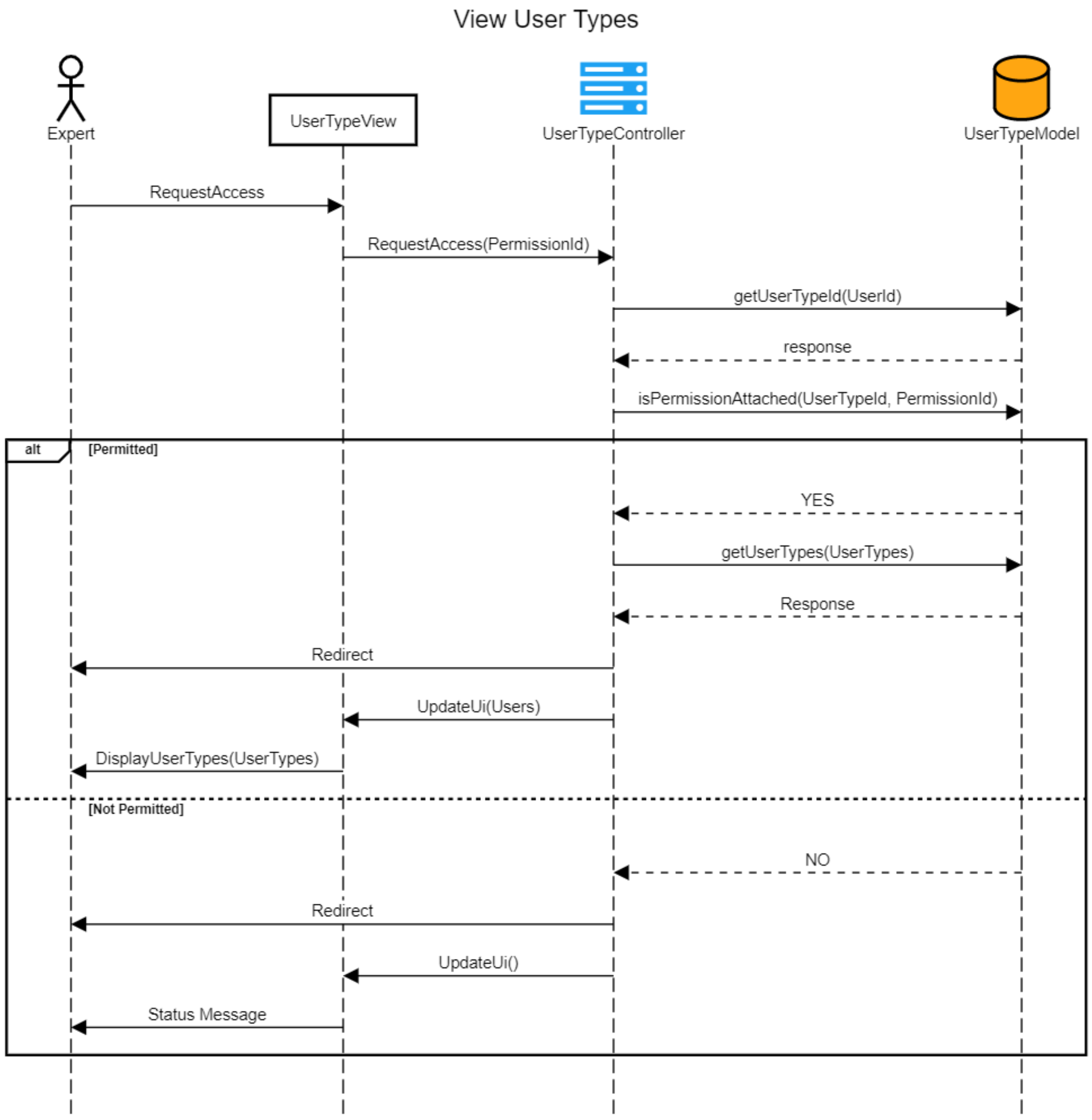
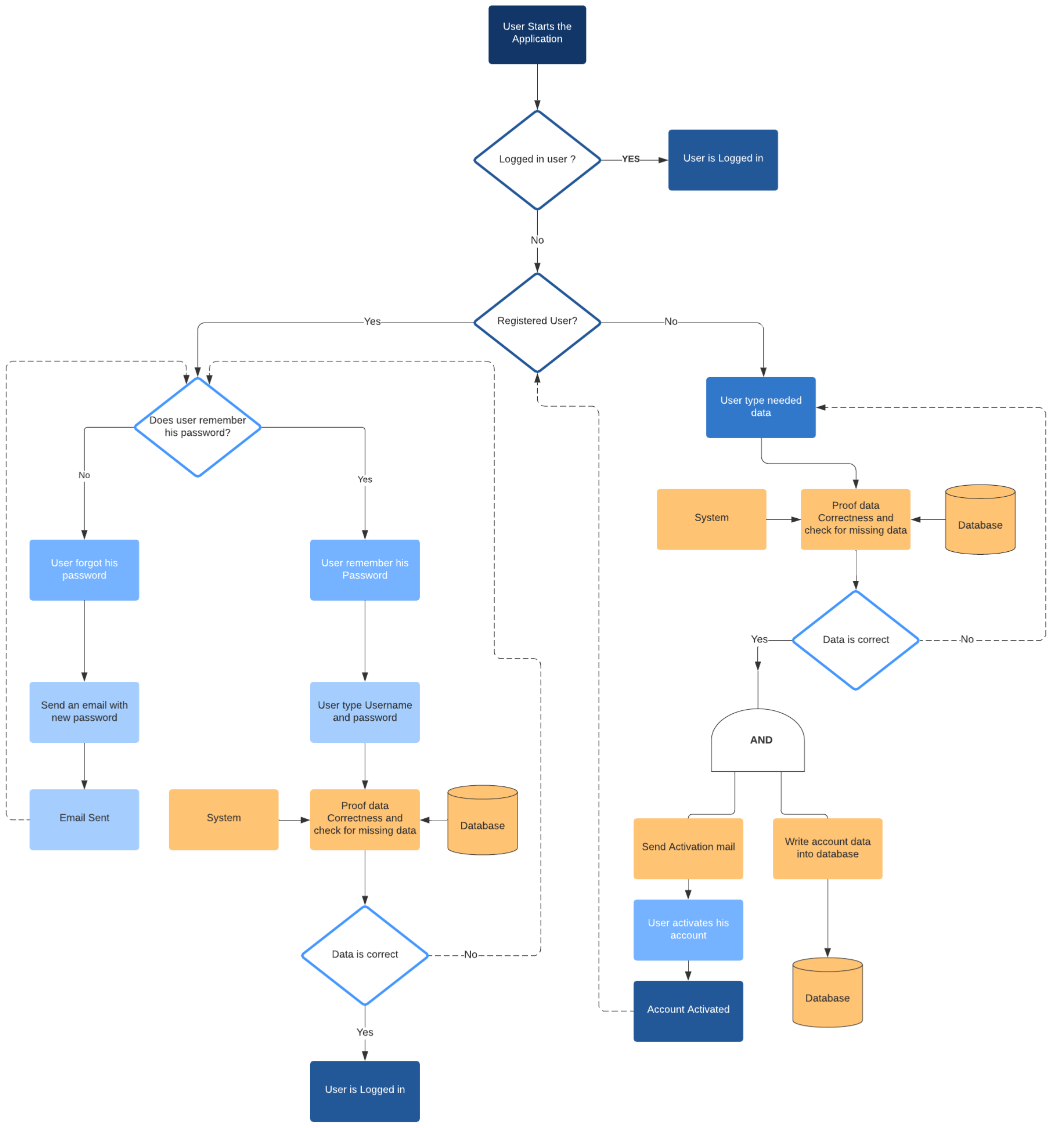


Figure 22: View users types

3.4 Process Diagram



User Starts the Application
Logged in user ?
YES
User is Logged in
No
Registered User?
Yes
No
Does user remember his password?
No
Yes
User forgot his password
User remember his Password
Send an email with new password
User type Username and password
Email Sent
System
Proof data Correctness and check for missing data
Database
Data is correct
No
Yes
User is Logged in
User type needed data
System
Proof data Correctness and check for missing data
Database
Yes
Data is correct
No
AND
Send Activation mail
Write account data into database
User activates his account
Database
Account Activated

Figure 23: Registration and Login process

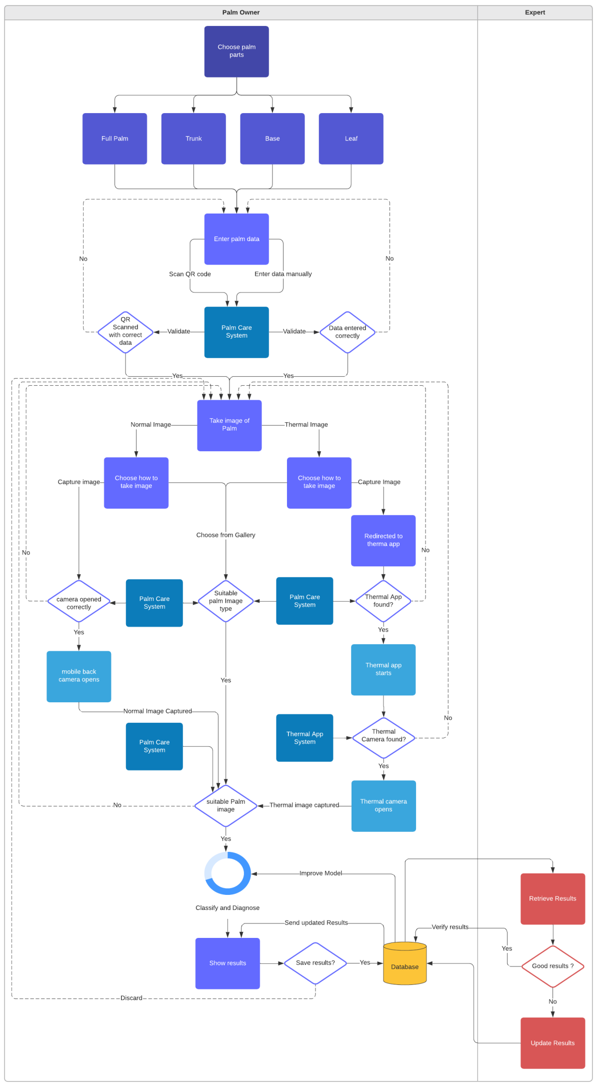


Figure 24: Palm care Process

3.5 Design Rationale

We use MVC architectural model (MVC) to make our code easy to maintain and to handle any change of a requirement and it also helps deal with data and presentation in a separate way ,also we are developing a unique software that tackles one of the biggest agriculture problems that's why our software must be accurate, sophisticated and user friendly as it target a wide demographic of people from experts to normal field owners and farmers, we use two of the best algorithms to tackle our problem the CNN :deep learning algorithm which considered to be one of the best image classifier but it's only problem that it need large amount of images to work on and the other algorithm is the SVM is one of the oldest and one of the best classifiers as it use arithmetic approaches to label or classify.

4 Data Design

4.1 Data Description

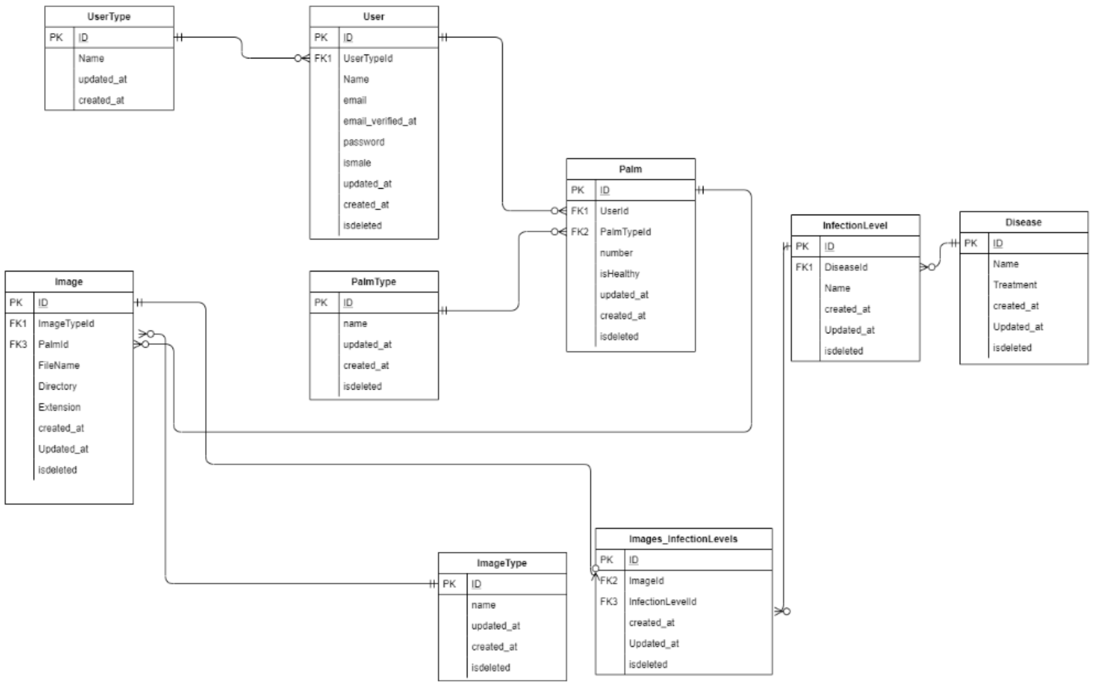


Figure 25: ER-Diagram

4.2 Data Dictionary

Table Name	**Description**
User	This table describes user information it is attributes are : (1)ID (2)UserTypeId related to UserType it specifies the user type either palm owner or admin or expert (3)Name specifies first and last name of the user (4)email (5)email_verified_at specifies the date of mail verification (6)Password (7)ismale; boolean attribute which specifies the gender of the user (8)Updated_at used to track the date of any update occurs to user information, used for security reasons (9)Created_at used to know the date of any new account creation (10)isDeleted used to know if user information is deleted or not, mainly used for user information restoration.

Table Name	**Description**
UserType	This table describes user Types information it is attributes are : (1)ID (2)Name either admin or palm owner or expert (3)Updated_at used to track the date of any change in user type (4)Created_at used to know the date of any new user type creation (5)isDeleted boolean used to know if user information is deleted or not, mainly used for usertype restoration and for security.

Table name	**Description**
Palm	This table describes palm information it is attributes : (1)ID (2)UserID related to user table, each user have many palms which can view,update or delete according to his usertype (3)Number, each palm has unique QR code number (4)isHealthy, Boolean which describes whether the palm is infected or not (5)Updated_at used to track the date of any update occurs to Palm information, used for security reasons. (6)Created_at used to know the date of any new palm created (7)isDeleted used to know if palm information is deleted or not, mainly used for palm information restoration or for security reasons

Table name	**Description**
PalmType	This table describes palm types information it is attributes : (1)ID (2)Name (5)Updated_at used to track the date of any update occurs to PalmType information, used for security reasons. (6)Created_at used to know the date of any new palm type created. (7)isDeleted, boolean used to know if any palm type is deleted or not, mainly used for palm types restoration or for security reasons

Table name	**Description**
Image	This table describes Image information it is attributes : (1)ID (2)ImageTypeID, related to imagetype table, used to know the type of the image (3)PalmId, used to know which image related to which palm (4)FileName, used to know the file the image stored in. (5)Directory, used to know the Directory the image stored in. (6)Extension, used to know the image extension either JPG, PNG, etc.. (7)Updated_at used to track the date of any update occurs to Image information, used for security reasons. (8)Created_at used to know the date of any new Image captured. (9)isDeleted, boolean used to know if any image is deleted or not, mainly used for palm types restoration or for security reasons.

Table name	**Description**
ImageType	This table describes Image Type information it is attributes : (1)ID (2)Name, either RGB or Thermal (3)Updated_at used to track the date of any update occurs to Image Type information, used for security reasons. (4)Created_at used to know the date of any new Image Type is created . (5)isDeleted, boolean used to know if any image type is deleted or not, mainly used for palm types restoration or for security reasons.

Table name	**Description**
Disease	This table describes Diseases information it is attributes : (1)ID (2)Name, either Leaf spots, blight spots, or RPW (3)Treatment, used to know the information about how to treat the palm according to the palm state (3)Updated_at used to track the date of any update occurs to diseases information, used for security reasons. (4)Created_at used to know the date of any new disease is created . (5)isDeleted, boolean used to know if any disease is deleted or not, mainly used for palm types restoration or for security reasons.

Table name	**Description**
InfectionLevel	This table describes the disease Infection Level it is attributes : (1)ID (2)DiseaseId, related to Disease table. (3)Name (4)Updated_at used to track the date of any update occurs to Infection Level information, used for security reasons. (5)Created_at used to know the date of any new Infection Level is created . (6)isDeleted, boolean used to know if any Infection Level is deleted or not, mainly used for palm types restoration or for security reasons.

Table name	**Description**
Images_InfectionLevel	This table to connect between Image and Infection Level tables it is attributes : (1)ID (2)Image Id, related to Image table. (3)Infection Level Id, related to infection_level table. (4)Updated_at used to track the date of any update occurs to this table information, used for security reasons. (5)isDeleted, boolean used to know if any Infection Level or image deleted or not, mainly used for palm types restoration or for security reasons.

5 Component Design

5.1 Image Acquisition

We have acquired a data set of 90 thousand images for leaf spots and blights from **'Kaggle'** data sets which is an augmented data set that makes a variety of features in the images for a good feature extraction process.

5.2 Pre Processing

we use a pretraied model VGG16 to help improve the data set with techniques such as subtracting the mean value RGB then the image is passed through a stack of conventional layers the n we pass it to small receptive 3x3

5.2.1 feature extraction

as was said we used VGG16[4] which can used to do the feature extraction for us the needed features extraction the architecture use max pooling layers, Convolution Layers,Dense layer is used and FC 1000(REF FC) and 4096 layers and softmax at the end

- *Convolution* layer:
  we used a 13 layers of conventional layers those layers are the one responsible for feature extraction [5]

- *Max Pooling* layer:
  we used 5 max pooling layers and these layers are responsible for summarizing or down sampling the feature map that is extracted from the conventional layer to get the highest features using a max pooling [1]

- *Dense* layer:
  we used our dense layer to match our classification problem as the dense layer is the layer to put a limit of classification labels we put a limit of three choices [2].

5.3 Classification Algorithm

Building a CNN Model upon Keras library [3] that is a deep neural network well known for it's best performance with complex image classification and its capability of improving the efficiency and accuracy of the model when dealing with huge data sets. Our CNN model is built on pre-structured VGG16 Network that's known for its well measured layers and parameters. We made some customizations in some layers in the VGG16 architecture to fit our plant disease classification case and added some features to the code that makes it computationally unexpensive, efficiently feature extracting, and predicting the outcomes in a maximum accuracy depending on the variation and the size of the given data set.

6 Human Interface Design

6.1 Overview of User Interface

The proposed application allows the user to create an account. The application will allow the user to sign in with his account and then will show a quick tour through the application to know how to use it. The user will choose which part of the palm tree he would like to take an image for it (Full palm tree, Trunk, base, Leaves), this will save processing time. The user will enter palm information manually or can scan palm QR code to easily fill palm information. Then the user is going to choose which type of image he is going to deal with, the application will recommend him to take the palm image using both thermal and normal images for better results. The user can either take thermal and normal images using cameras or import his images to the application. The application will show the results if the palm is infected or not, and infection level in case of palm infection, also the application will provide some solutions to the user according to the infection level. Palm information and results will be saved into the user account and the user can show them at any time.

6.2 Screen Images

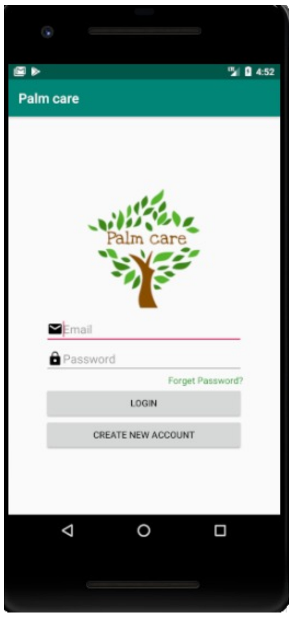


Figure 26: Login

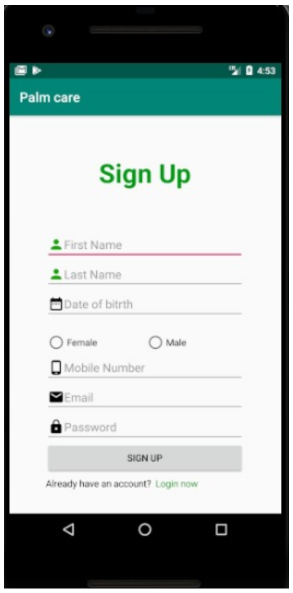


Figure 27: Sign up

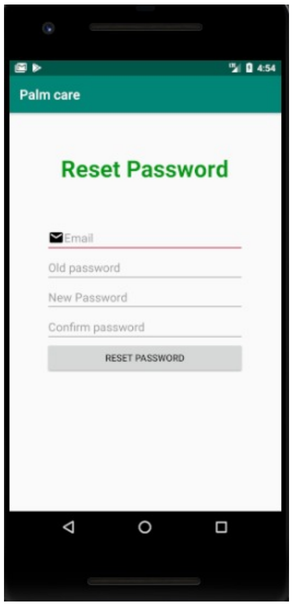


Figure 28: Reset Password

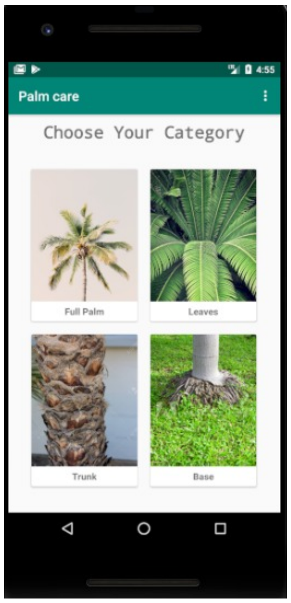


Figure 29: Choose palm parts

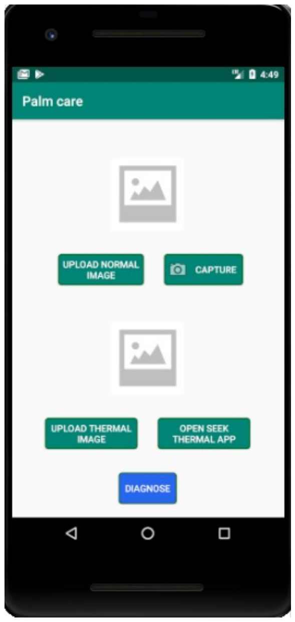
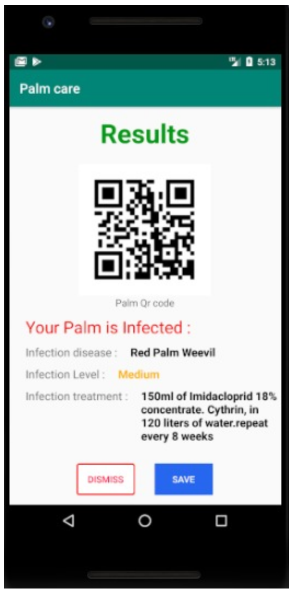


Figure 30: upload image



Palm care

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 31: Results

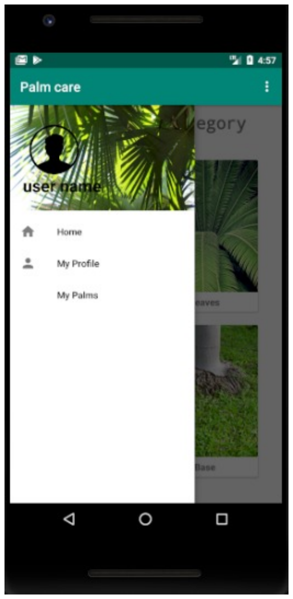
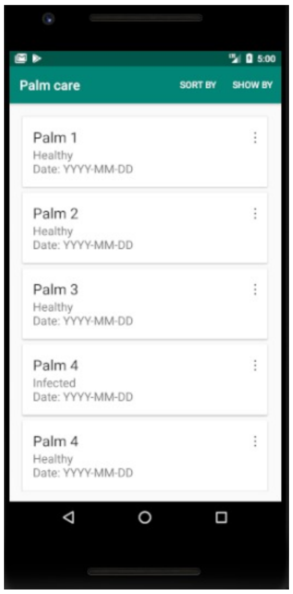


Figure 32: Sidebar



Palm 1
Healthy
Date: YYYY-MM-DD

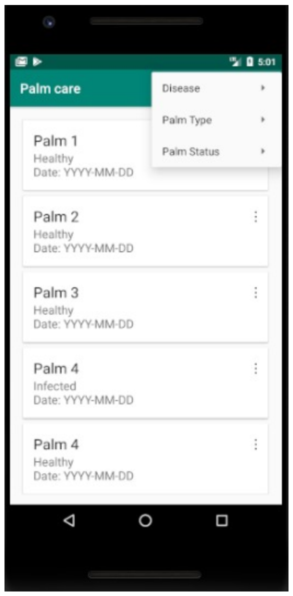
Palm 2
Healthy
Date: YYYY-MM-DD

Palm 3
Healthy
Date: YYYY-MM-DD

Palm 4
Infected
Date: YYYY-MM-DD

Palm 4
Healthy
Date: YYYY-MM-DD

Figure 33: Show user palms



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 34: Show by menu

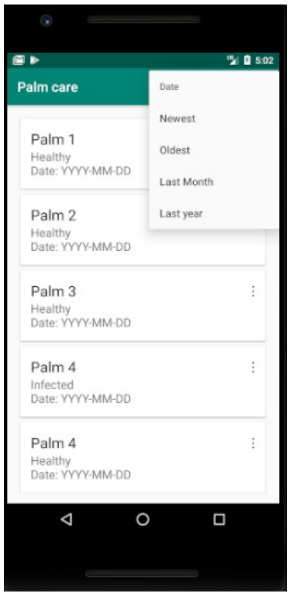
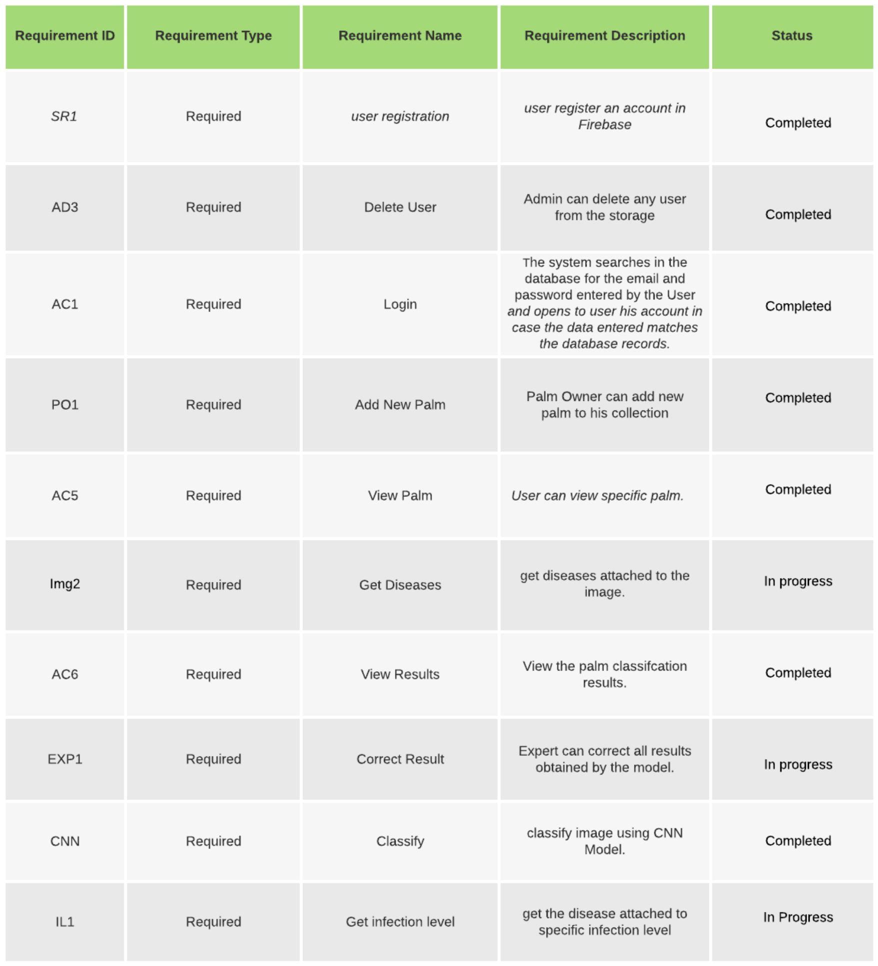


Figure 35: Sort by date menu

7 Requirements Matrix



Requirement ID	Requirement Type	Requirement Name	Requirement Description	Status
SR1	Required	*user registration*	*user register an account in Firebase*	Completed
AD3	Required	Delete User	Admin can delete any user from the storage	Completed
AC1	Required	Login	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.*	Completed
PO1	Required	Add New Palm	Palm Owner can add new palm to his collection	Completed
AC5	Required	View Palm	*User can view specific palm.*	Completed
Img2	Required	Get Diseases	get diseases attached to the image.	In progress
AC6	Required	View Results	View the palm classifcation results.	Completed
EXP1	Required	Correct Result	Expert can correct all results obtained by the model.	In progress
CNN	Required	Classify	classify image using CNN Model.	Completed
IL1	Required	Get infection level	get the disease attached to specific infection level	In Progress

Figure 36: Requirement Matrix

References

[1] Jason Brownlee. "A Gentle Introduction to Pooling Layers for Convolutional Neural Networks". In: (2019).

[2] Hunter Heidenreich. "Understanding Keras — Dense Layers". In: (2019).

[3] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[4] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[5] Matthew Stewart. "Simple Introduction to Convolutional Neural Networks". In: (2019).