Senior Design Project

GrapeHealth

Analysis Report

İlhami Özer, İbrahim Berker Kırdök, Murat Süerdem, Osman Sefa İbiş, Göktuğ Özdoğan

Supervisor: Prof. Dr. Halil Altay Güvenir

Jury Members: Prof. Dr. İbrahim Körpeoğlu - Doç. Dr. Özcan Öztürk

Analysis
Report Nov
12, 2018
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Content

1. Introduction

Grape is the most produced fruit in Turkey's market share. Turkey has excessively link field for grape production and has great potential for viticulture. Also, Spain, France, Italy and China are the world leaders of grape production. The importance of detecting grape diseases arises in thinking of the worldwide impact. The main purpose of our innovation is to be able to detect five most common grape diseases to change the viticulture experience of the farmers to easily diagnose their grape products.

Farmers solely depend on agriculturalists to diagnose and cure their products, with our mobile application farmers can detect if their grape has disease or not. The potential users are farmers, agricultural engineers, government's agricultural consultants, agricultural groups, grape merchants and drug companies or chemical stores. Our ambition is to help agricultural specialists and groups in being an additional resource to detect diseases. Grape merchants can decide whether or not the product they will buy has any of the common diseases. Farmers and drug companies can communicate more biologically to decide the accurate drug for the grapes. In some circumstances chemical stores and farmers may not have the information to detect the diseases. Integrating the potential users to diagnose diseases will help grape production sector to communicate better. Detecting diseases will improve efficiency and quality by adding value to fresh and healthy grapes.

In this report, we will provide a comprehensive analysis of the system. Firstly, current system will be described considering the existing systems Then the details of our system and unique features of GrapeHealth will be specified. Functional, non-functional, and pseudo requirements are addressed. Afterwards, the system models of our system are included. Scenarios of GrapeHealth are explained in detail. These scenarios are generalized by common use case descriptions and the use case diagram is given. The object model and the dynamic models of the system are also provided and explained. Finally, the screen mock-ups and the navigational paths are included.

2. Current System

There are no applications in the market share which diagnoses and informs agricultural diseases which can both show notification nearby and weather locally. In the market there are some applications which could find out type of plants or diagnose diseases [1]. None of them warns the user about treatment of the disease or local weather. These applications do not show treatments or recommend any date about treatment to the user.

3. Proposed System

3.1. Overview

GrapeHealth will be a mobile application (Android and iOS) which is designed to serve farmers, agricultural engineers, government's agricultural consultants, agricultural groups, grape merchants and drug companies or chemical stores. The main purpose of the application is recognizing the disease of the grape plants.

The application will use two features to recognize diseases. First feature takes photos of the leaf of the grape plant as input and gives probability of five possible diseases that the plant could have. Leaves are the determining factor to detect diseases. Second feature is a Chatbot that asks specific questions to user to get more knowledge about the situation of the plant. After using these functionalities, the user will be able to see the possible diseases with their potential percentages. The user will also have other functionalities with the application, this report will go in depth by explaining.

3.2. Functional Requirements

- Application requires internet connection and a smartphone or a tablet.
- Application will store login and location information on database.
- Application will store previous images which is uploaded by user and the dedicated results with it.
- Application will give notifications if a contagious disease is present in nearby location.
- Application will continue to learn from images uploaded by users if the photos are validated by the expert.
- Users have to download GrapeHealth from Google Play Store or Apple Store to use application.
- User have to have a valid e-mail account to sign up.
- User’s e-mail account should not be used by another user.
- User’s password should contain at least 6 characters and at least one capital letter and one digit.
- User can add photo from phone library.
- User should give permission to application to use camera and gps for full functionality.
- User should be able to learn disease of grape in three ways.
    - User should upload three photos with specified shooting angle.
    - User should answer questions through chatbot to learn the disease of grape.
    - User should be able to use both images and chatbot to find disease of grape.
- User will be able to see diseases in his/her location.
- Chatbot will ask questions to user related to diseases. Chatbot contains buttons for answers rather than plain text. It is more trustable since user can make spelling mistakes.
- The system will check whether the photo user uploads is an grape leaf or not.
- The system will allow user to see results after validated photos or by the end of using chat-bot.
- User will be able to change profile information.
- User will be able to change his/her password.
- User will be able to logout from his/her account.

3.3. Non-Functional Requirements

Usability

- The user interface should be user-friendly.
- The application should not require any previous knowledge, it should be easy to use.

- Application should be supported most of the Android and iOS versions. Our preliminary examination indicates that Android 4.4 is support %96 of devices and iOS 9 support %98 of devices [4][5].
- Using buttons for chatbot make this part of program easier to you regard to allow user to enter a word/sentence.

Reliability

- The accuracy rate of the application should be higher than %80. If accuracy rate is lower than %80, application should give warning to user. %80 and above can be considered as successful [3].
- If application find any diseases related to provided photo with low probability, it should give warning message.
- Using buttons for chatbot instead of giving change for user to enter sentence/word prevents errors because of spelling mistakes.

Efficiency

- The process of finding disease should not be longer than 60 seconds.

Extensibility

- GrapeHealth will be extensible. For now, it will detect diseases on grapes, however, we can increase the range of diseased plants determined by application.
- The application should be extensible to add new functionalities.

Scalability

- The application gives permission for a user to send only one query. The user could not send another query while his/her query still in progress.

Security

- Application should ask for permission for camera and gps from user when user starts the application.
- Personal data should be secured by the system and images that are received should be used only for train the system.

3.4. Constraints

Implementation Constraints

- Github platform will be used to work efficiently with group members.
- Deep convolutional neural network in Python will be implemented for designing model.
- One of these frameworks will be used: Caffe, Tenserflow, Keras.
- AlexNet or GoogLeNet will be used as deep learning architecture.
- PlantVillage dataset will be used for training, validation and test.
- YOLO3 will be used for edge detection.
- ReactNative will be used for implementation of mobile application.
- Flowchart representation will be implemented for Chatbot.

Economical Constraints

- Training datasets, deep learning tools, frameworks and libraries will be used with no fee.
- Hosting and domain for our website will be bought.
- Server and database will be rent for neural network training.
- Publishing the application to Google Play Store and Apple Store will have fee cost.

Environmental Constraints

- User will be warned to pay attention to leaves while taking photos.

Social Constraints

- User that uploads irrelevant photos will be blocked, thus; user will be encouraged to obey this obligation.
- Personal privacy will be taking in consideration according to EU privacy laws.

Political Constraints

- None.

Health and Safety Constraints

- Detection of diseases of the grapes enhances public safety and health. Therefore, user will be warned to wear gloves while interacting leaves and taking photos.
- Users will be warned about treatment ways, in case of any chemical will be used.

Manufacturability Constraints

- None.

Sustainability Constraints

- Application will improve itself by learning from images uploaded by users, thus; it will give more reliable results to user.

3.5. System Models

3.5.1. Scenarios

<u>Scenario 1</u>

Use Case Name: Display Greetings

Actors: User

Entry Conditions:

- User opens the application

Exit Conditions:

- User closes the application.

Main Flow of Events:

1. GrapeHealth application shows greetings page at first.
2. User will choose from sign-up or sign-in options.

Scenario 2

Use Case Name: Sign-in

Actors: User

Entry Conditions:

● User is on sign-in page

Exit Conditions:

● User click on the sign-in button.

Main Flow of Events:

1. Sign-in page expects user to login with 3 options; Regular login expects E-mail and password access; Google account login and Facebook account login will be redirected.
2. User will be able to have "Forgot password?" option. In this case an email will be sent to change password.
3. User can navigate to sign-up page in case of not having an account.

Scenario 3

Use Case Name: Sign Up

Actors: User

Entry Conditions:

● User selects the sign-up from greetings page.

Exit Conditions:

● From the back-button user could go back to Greetings page or by "Already have an account?" button the user could go to Sign In page.

Main Flow of Events:

1. GrapeHealth sign-up page requires certain information such as name, surname, email and password.
2. User enter name and surname.

3. User enters e-mail. The format of the email will be checked and if the format is wrong e-mail will not be accepted.

4 User enters a password. The format of the password will be checked and if the format is wrong password will not be accepted. Also, information icon will be at the right of password bar to inform the user about the accepted password format.

5. User has to re-type the password. User will not be able to sign-up if it does not match with the above password.

6. After filling the sign-up information correctly user will be able to sign-up.

7. User will able to navigate to sign-in page or will use back button to reach to the sign-in page.

<u>Scenario 4</u>

Use Case Name: Display Old Queries

Actors: User

Entry Conditions:

● After signing-in user will be navigated to this view.

Exit Conditions:

● User selects any other option from side menu.

Main Flow of Events:

1. User will view old queries if any. Old queries will be shown with not detailed versions. Old queries shown at this view will show query number, date of creation and the disease type if any otherwise healthy with the accuracy rate. If user wants detailed information about past queries by selecting any of them the user will be navigated to 'Query Detail' view.

2. User has 3 options to generate a new query. First and suggested one is to use both chat-bot and adding photo option. By this option user will be able to be informed about the accuracy rate more precisely. Second option is to use chat-bot alone. Third option is to take photo.

3. If user selectes chat-bot option the app will view chat-bot view. After answering questions on chat-bot user will be redirected to query view. User will be able to see the results of chat-bot.

4. If the user selects add photo option, the application will open the camera and user need to take pictures. After user inputs a photo the application will validate and analyze the photo to detect whether the grape leaf contains any disease or not. The accuracy rate and other statistics will be shown after user navigates to query detail view.

5. User will be able to navigate to this view also by using the side menu.

6. Query details will be shown if the user selects any of the queries.

Scenario 5

Use Case Name: Nearby Disease Risk View

Actors: User

Entry Conditions:

- User selects ‘Nearby’ option from the side menu.

Exit Conditions:

- Selecting another option from side menu bar.

Main Flow of Events:

1. If the user allowed the application to get GPS information, the application will view the user about any possible nearby disease if any.

2. User will be able to see what disease nearby is. İnformation provided will be disease name, distance to the diseases and the report date.

Scenario 6

Use Case Name: Viewing Query Details

Actors: User

Entry Conditions:

- After adding a photo or using chat-bot or by using them both user will generate a query.

- User can also access this view by ‘Queries’ option in the side bar, under the old queries by selecting any this view will be shown.

Exit Conditions:

- User selecting another option in side menu.

Main Flow of Events:

1. User will view information about query generated by the inputs provided by them.

2. Query name or number with reporting date will be shown.

3. Statistical graph will be shown relying on the probabilistic rates of potential diseases.

4. User will be able to see the the photo they had uploaded by ‘Show Photo’ button.

5. User will be able to see the possible treatment options by selecting ‘Show Treatment’ button.

6. The diseases will be in a ranked order of most probable disease rate. The percentage and the disease name will be shown.

7. The user will be able to select add photo or use chat-bot functionalities if any of them is not used. If the user has already used these functionalities the application will not allow.

8. User will be able to view the GPS location of the query.

Scenario 8

Use Case Name: Show Treatment

Actors: User

Entry Conditions:

● Within 'Query Details' view the user can select 'Show Treatment' button to navigate to this view.

Exit Conditions:

● User selecting another option in side menu.

Main Flow of Events:

1. The disease name will be visible. GPS information and suggested treatment dates will be reported due to weather conditions.

2. User will view the suggested treatment.

Scenario 9

Use Case Name: Notification

Actors: User

Entry Conditions:

● User selecting 'Notifications' from side menu.

Exit Conditions:

● User selecting another option in side menu.

Main Flow of Events:

1. User will view notifications about nearby diseases and risky weather conditions. Data will be same as 'Nearby' view but in the notifications the data will be stored.
2. User will be notified with weather information if there is a potential risk with risk type (ex: Heavy rain) and the possible time periods it effects.

Scenario 10

Use Case Name: Settings View

Actors: User

Entry Conditions:

● User will select settings from side menu.

Exit Conditions:

● User selecting another option in side menu.
User can delete account from this view.

Main Flow of Events:

1. User can change information such as name, surname, email, password.
2. To edit the changes the user should select save button.
3. The user can delete account with ‘Delete Account’ button.

Scenario 11

Use Case Name: Chat-Bot

Actors: User

Entry Conditions:

● User can use the functionality of chat-bot by the ‘Display Old Queries’ view, there is a button which navigates to this view.
Also, user can navigate to this view by ‘Queries’ option in the side menu. From there we navigate to ‘Query Details’ view and we have the option to add chatbot if the user has not used it while creating query.

Exit Conditions:

● User selecting another option in side menu.
After user finishes the chatbot questioning the navigation will flow through ‘Query Details’ view.

Main Flow of Events:

1. User will answer chatbot questions by selecting answers with buttons. After answering all the questions, the functionality directly navigate to ‘Query Details’ view to show the results.

Scenario 12

Use Case Name: Help View

Actors: User

Entry Conditions:

- User selects it from the side menu.

Exit Conditions:

- User selecting another option in side menu.

Main Flow of Events:

1. User will be able to seek help within the application. How to use the application tutorial will help user to understand the application.
2. Edit profile, upload photo, use chatbot, delete account will be options to help the user.

Scenario 13

Use Case Name: About Us

Actors: User

Entry Conditions:

- Select it from side menu.

Exit Conditions:

- User selecting another option in side menu.

Main Flow of Events:

1. User will view information provided by the developer team.

Scenario 14

Use Case Name: Upload Photo
Actors: User

Entry Conditions:

- User can upload photos from the camera or library of their phone. By ‘Displaying Old Queries’ view user can select to add photo for a new query.
  User can upload a photo also from ‘Query Details’ view if the user only used chatbot to get results.

Exit Conditions:

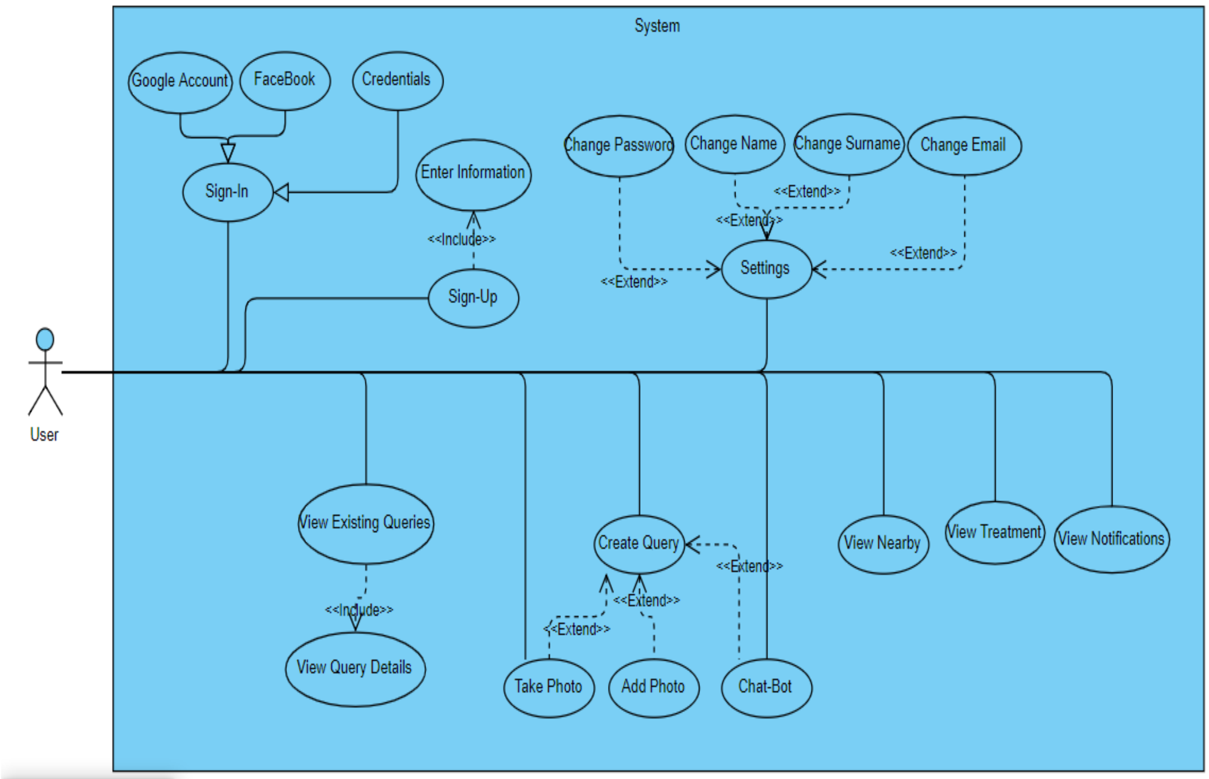
- User closing the camera or cancelling to upload photo from library.

After taking photo the user will be viewing a prompt suggesting to ‘Continue with Chatbot?’. If yes button is selected the application will navigate to chatbot view otherwise the navigation will go to ‘Query Details’.

Main Flow of Events:

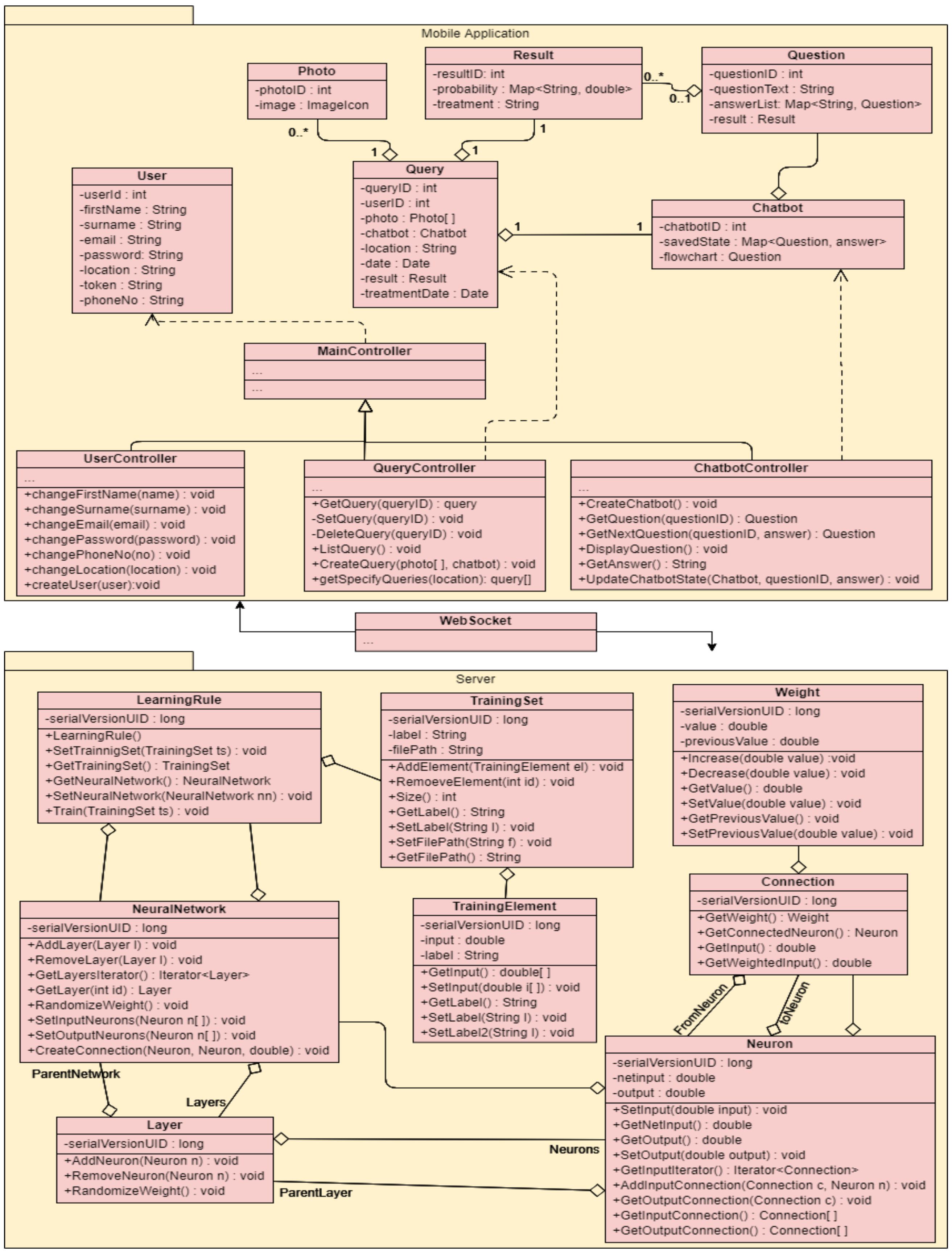
1. The user will take photo and if the photo is validated by the system the photo will be analyzed for results.

3.5.2. Use Case Model



Sign-in: User can login via using Google account, Facebook or by credentials.
Sign-up: User must enter information to sign-up.
Settings: User can change settings such as password, name, surname and email as wanted.
View Existing Queries: User can view old queries and the query details by selecting one of them.
Create Query: User can create query by using cht-bot, taking a photo or adding a photo by camera roll.
View Nearby: User can view nearby potential risks if any.
View Treatment: User can view treatment.
View Notifications: User can view notifications about possible nearby risks and weather conditions.
Take photo: User can directly take a photo to create a query.
Chat-bot: User can directly use chat-bot to create query.

3.5.3. Object and Class Model



Mobile Application
Photo
-photoID : int
-image : ImageIcon
Result
-resultID: int
-probability : Map<String, double>
-treatment : String
Question
-questionID : int
-questionText : String
-answerList: Map<String, Question>
-result : Result
0..*
0..1
0..*
1
1
1
1
User
-userId : int
-firstName : String
-surname : String
-email : String
-password: String
-location : String
-token : String
-phoneNo : String
Query
-queryID : int
-userID : int
-photo : Photo[ ]
-chatbot : Chatbot
-location : String
-date : Date
-result : Result
-treatmentDate : Date
Chatbot
-chatbotID : int
-savedState : Map<Question, answer>
-flowchart : Question
1
1
MainController
...
...
UserController
...
+changeFirstName(name) : void
+changeSurname(surname) : void
+changeEmail(email) : void
+changePassword(password) : void
+changePhoneNo(no) : void
+changeLocation(location) : void
+createUser(user):void
QueryController
...
+GetQuery(queryID) : query
-SetQuery(queryID) : void
-DeleteQuery(queryID) : void
+ListQuery() : void
+CreateQuery(photo[ ], chatbot) : void
+getSpecifyQueries(location): query[]
ChatbotController
...
+CreateChatbot() : void
+GetQuestion(questionID) : Question
+GetNextQuestion(questionID, answer) : Question
+DisplayQuestion() : void
+GetAnswer() : String
+UpdateChatbotState(Chatbot, questionID, answer) : void
WebSocket
...
Server
LearningRule
-serialVersionUID : long
+LearningRule()
+SetTrainnigSet(TrainingSet ts) : void
+GetTrainingSet() : TrainingSet
+GetNeuralNetwork() : NeuralNetwork
+SetNeuralNetwork(NeuralNetwork nn) : void
+Train(TrainingSet ts) : void
TrainingSet
-serialVersionUID : long
-label : String
-filePath : String
+AddElement(TrainingElement el) : void
+RemoeveElement(int id) : void
+Size() : int
+GetLabel() : String
+SetLabel(String l) : void
+SetFilePath(String f) : void
+GetFilePath() : String
Weight
-serialVersionUID : long
-value : double
-previousValue : double
+Increase(double value) :void
+Decrease(double value) : void
+GetValue() : double
+SetValue(double value) : void
+GetPreviousValue() : void
+SetPreviousValue(double value) : void
NeuralNetwork
-serialVersionUID : long
+AddLayer(Layer l) : void
+RemoveLayer(Layer l) : void
+GetLayersIterator() : Iterator<Layer>
+GetLayer(int id) : Layer
+RandomizeWeight() : void
+SetInputNeurons(Neuron n[ ]) : void
+SetOutputNeurons(Neuron n[ ]) : void
+CreateConnection(Neuron, Neuron, double) : void
TrainingElement
-serialVersionUID : long
-input : double
-label : String
+GetInput() : double[ ]
+SetInput(double i[ ]) : void
+GetLabel() : String
+SetLabel(String l) : void
+SetLabel2(String l) : void
Connection
-serialVersionUID : long
+GetWeight() : Weight
+GetConnectedNeuron() : Neuron
+GetInput() : double
+GetWeightedInput() : double
FromNeuron
toNeuron
ParentNetwork
Layers
Layer
-serialVersionUID : long
+AddNeuron(Neuron n) : void
+RemoveNeuron(Neuron n) : void
+RandomizeWeight() : void
Neurons
ParentLayer
Neuron
-serialVersionUID : long
-netinput : double
-output : double
+SetInput(double input) : void
+GetNetInput() : double
+GetOutput() : double
+SetOutput(double output) : void
+GetInputIterator() : Iterator<Connection>
+AddInputConnection(Connection c, Neuron n) : void
+GetOutputConnection(Connection c) : void
+GetInputConnection() : Connection[ ]
+GetOutputConnection() : Connection[ ]

Mobile Application:

User: The system allows to user to access program after user create his or her account.
Query: Query can be created by user to use chatbot by answering questions or to use deep learning model by uploading photo of diseased grape leaf, or both of them. These queries are kept in this class with its results for each specified user.
Photo: Photo is the model class, which enables system to create entity on the database system. It specifies the unique Id for each Photo and it enables to store them as an Input Stream.
Result: This class keeps result of each query created by user, also each result has a treatment suggestion.
Chatbot: Chatbot is a class which enables systems to operate chatbot algorithm. It stores questions and answers in the map. According to the order of the questions, it highlights them in sequence of flowchart.
Question: Question is a class which operates the chatbot questions. All questions have unique Id, and all questions have answer related to specific diseases. Question class provides to system, specify the unique question Id, new questions and answers of them.
MainController: This abstract parent class for UserController, QueryController and ChatbotController classes.
UserController: UserController manages user identity activities like creating account or updating profile.
QueryController: QueryController manages activities related to Query class, like creating, deleting, getting or listing queries which are belong to specified user and used for learning disease of the grape.
ChatbotController: ChatbotContoller is a class which manages and operates chatbot activitives which are create chatbot, get questions of chatbot and display the question of chatbot.
WebSocket: WebSocket is protocol which provides full-duplex communication channels over a single TCP connection [2]. It enables the system to communicate app domain and server domain.

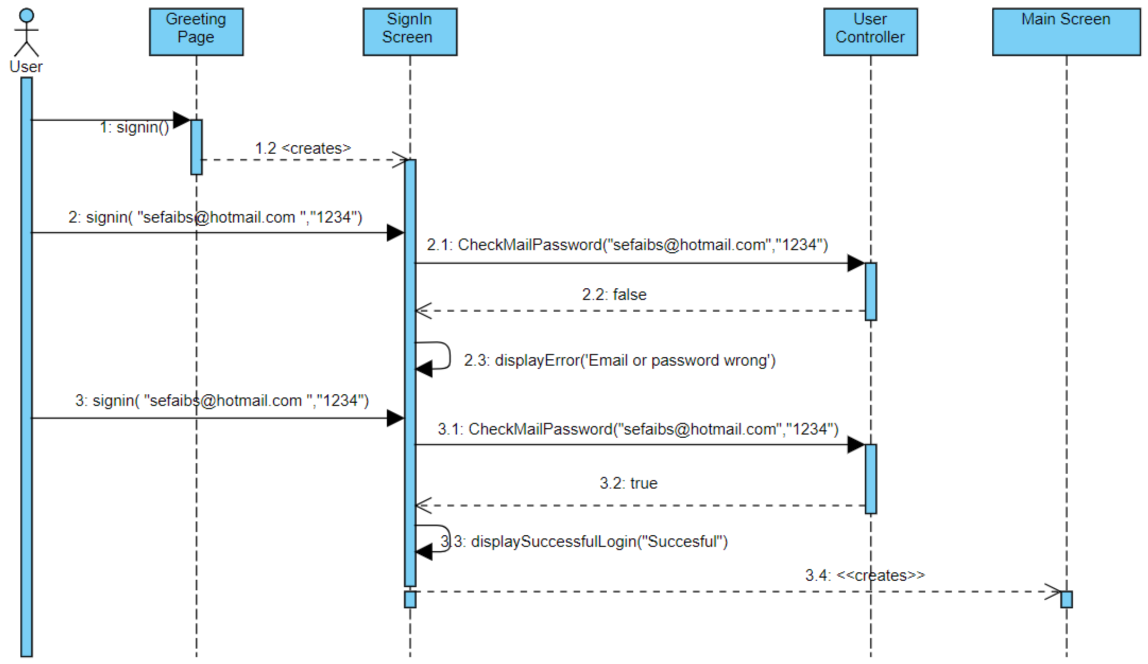
Server:

LearningRule: It determines how to use saved neural network. It has operations to learn by using neural network and training set. Learning occurs as weights of the connection between neurons are updated according to labeled input data which is given to neural network.
TrainingSet: Different types of training inputs are clustered in different classes.
TrainingElement: Each training input data element from specified TrainingSet is used for learning.
NeuralNetwork: It is saved computational model inspired by human brain and contains layers, neurons, connections and weights.
Layer: Different levels of layers which involve many neurons constitute neural network.
Neuron: Each neuron takes input from its connections with previous layer and gives output to the neurons in next layer through connections between them.
Connection: It represents connections between neurons to transport outputs, each neuron has connection with all neurons in previous layer and next layer.
Weight: Each connection has a weight and this weight is increased or decreased as labeled input data is used.

3.5.4. Dynamic Models

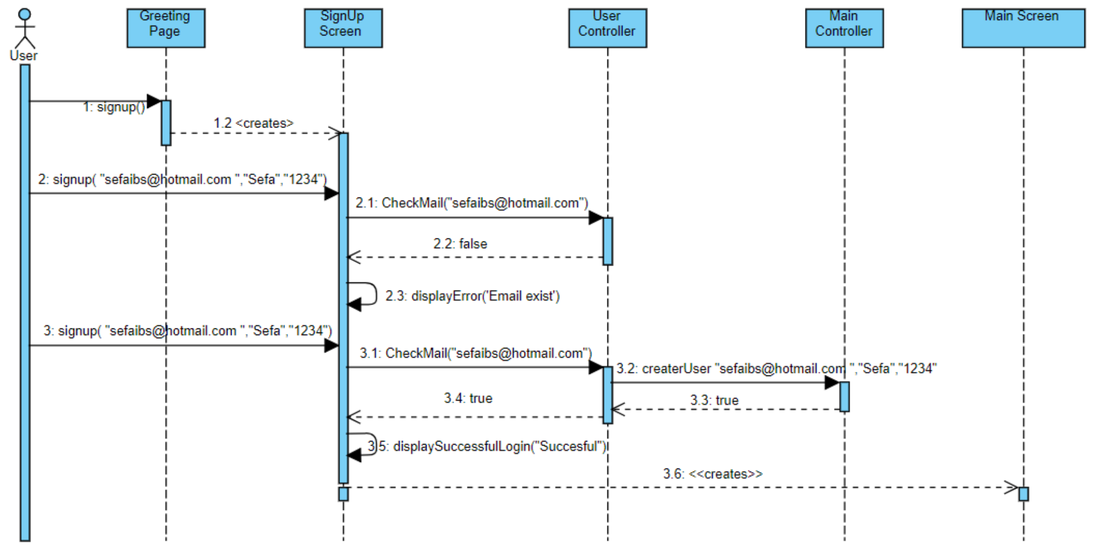
3.5.4.1. Sequence Diagrams

Sign-In Sequence Diagram


User
Greeting Page
SignIn Screen
User Controller
Main Screen
1: signin()
1.2 <creates>
2: signin("sefaibs@hotmail.com ","1234")
2.1: CheckMailPassword("sefaibs@hotmail.com","1234")
2.2: false
2.3: displayError('Email or password wrong')
3: signin("sefaibs@hotmail.com ","1234")
3.1: CheckMailPassword("sefaibs@hotmail.com","1234")
3.2: true
3.3: displaySuccessfulLogin("Succesful")
3.4: <<creates>>

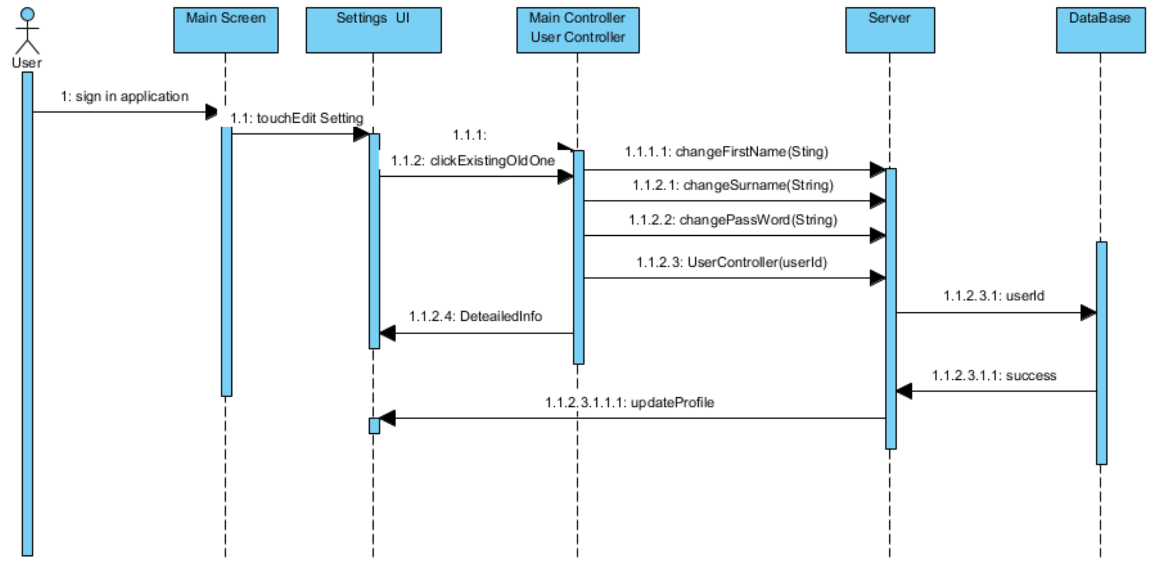
For sign-in user will specify email and password. The system will check for validation. Then user will be able to use the application.

SignUp Sequence Diagram


User
Greeting Page
SignUp Screen
User Controller
Main Controller
Main Screen
1: signup()
1.2 <creates>
2: signup("sefaibs@hotmail.com ","Sefa","1234")
2.1: CheckMail("sefaibs@hotmail.com")
2.2: false
2.3: displayError('Email exist')
3: signup("sefaibs@hotmail.com ","Sefa","1234")
3.1: CheckMail("sefaibs@hotmail.com")
3.2: createrUser "sefaibs@hotmail.com ","Sefa","1234"
3.3: true
3.4: true
3.5: displaySuccessfulLogin("Succesful")
3.6: <<creates>>

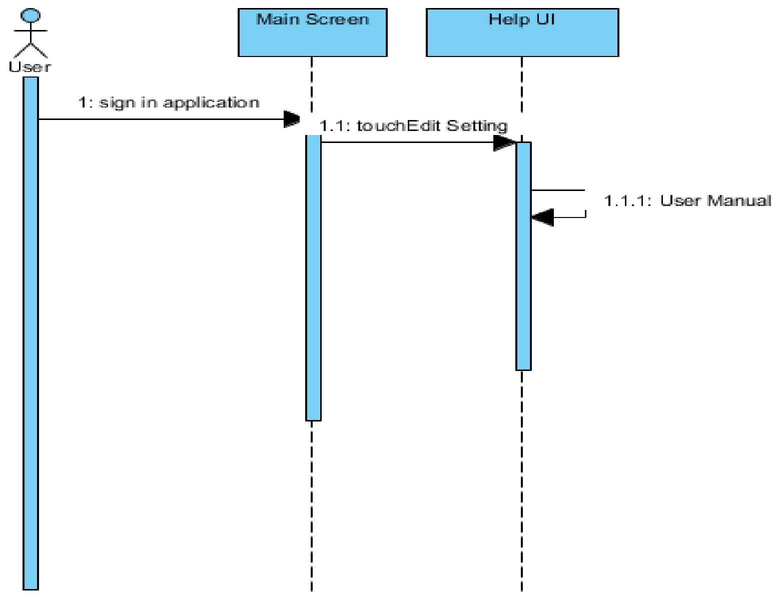
User will sign-up by specifying information such as email, name and password. The system will check for the email then after validating the user can finish the sign-up process.

Settings Sequence Diagram



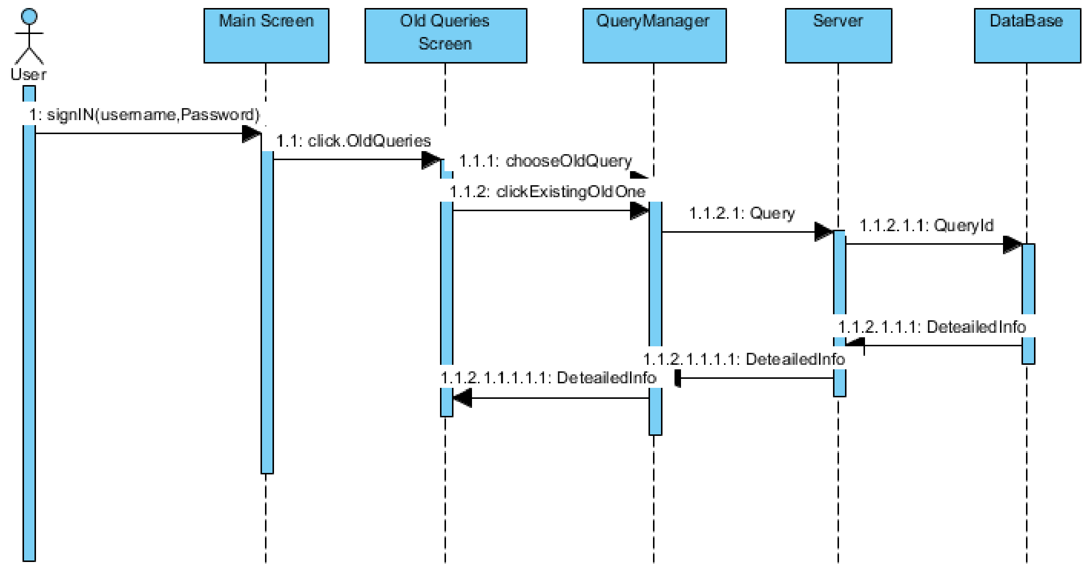
User touch setting bar on the main screen, after specified changes on the Settings Screen, when user touch ok bar on the screen, provided changes will send to database with unique userId via server. Necessary changes on databases is performed, with detailed information provided by user.

Help Sequence Diagram



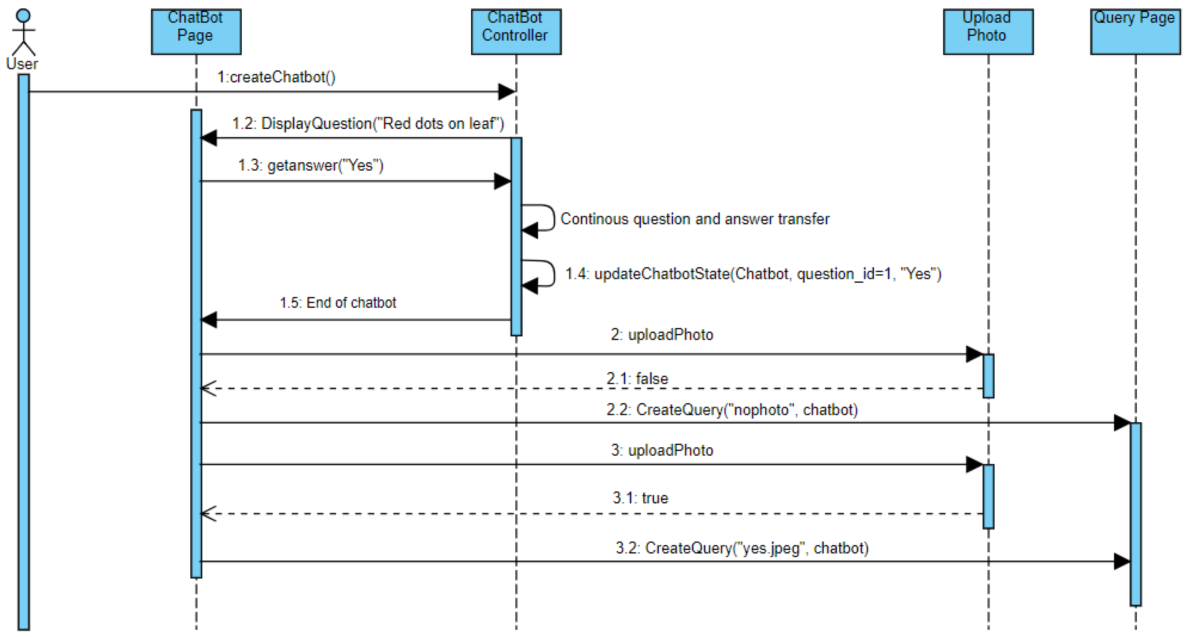
User touches help bar on the main screen, after that program displays the User Manual.

Display Old Queries Sequence Diagram


User
Main Screen
Old Queries Screen
QueryManager
Server
DataBase
1: signIN(username,Password)
1.1: click.OldQueries
1.1.1: chooseOldQuery
1.1.2: clickExistingOldOne
1.1.2.1: Query
1.1.2.1.1: QueryId
1.1.2.1.1.1: DeteailedInfo
1.1.2.1.1.1.1: DeteailedInfo
1.1.2.1.1.1.1.1: DeteailedInfo

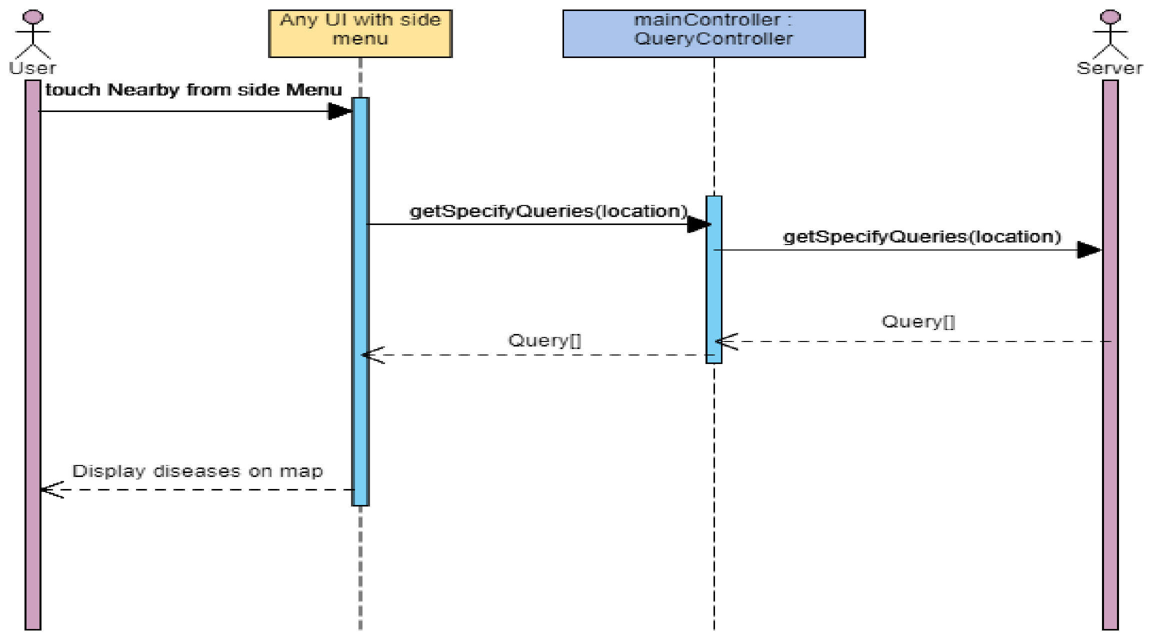
After user sign in to application, user will view old queries if any. Old queries will be shown with not detailed versions. When user click old queries, query id of clicked queries will send to server with QueryManeger. In the databases with the provided QueryId, detailed information of specified query will be sent OldQuery Screen.

ChatBot Sequence Diagram


User
ChatBot Page
ChatBot Controller
Upload Photo
Query Page
1:createChatbot()
1.2: DisplayQuestion("Red dots on leaf")
1.3: getanswer("Yes")
Continous question and answer transfer
1.4: updateChatbotState(Chatbot, question_id=1, "Yes")
1.5: End of chatbot
2: uploadPhoto
2.1: false
2.2: CreateQuery("nophoto", chatbot)
3: uploadPhoto
3.1: true
3.2: CreateQuery("yes.jpeg", chatbot)

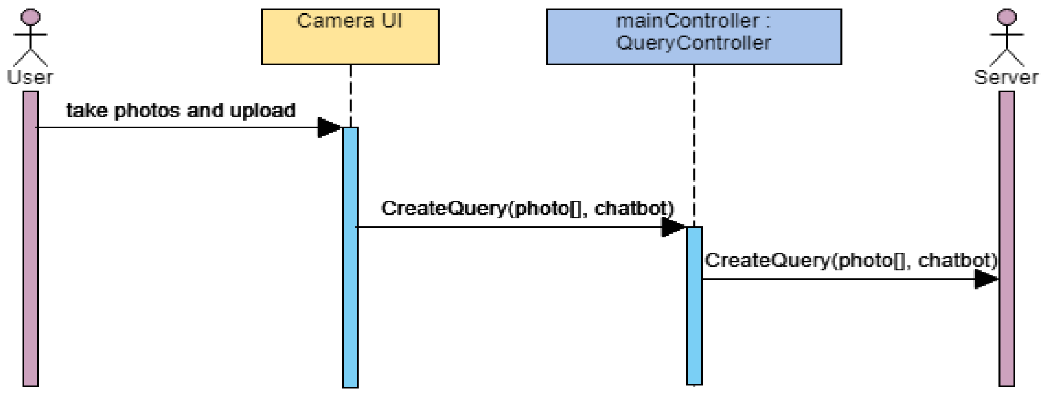
User will answer questions for chatbot to diagnose the diseases of the grapes. After the questioning and answering process ends the user will have the option to upload a photo or not. Then the user will be navigated to the query page.

Nearby Sequence Diagram


User
Any UI with side menu
mainController : QueryController
Server
touch Nearby from side Menu
getSpecifyQueries(location)
getSpecifyQueries(location)
Query[]
Query[]
Display diseases on map

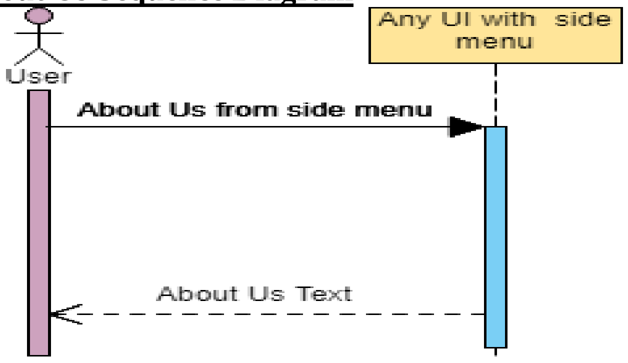
When a user touch on the "Nearby" menu item on side menu, Query Controller sends request to database to get nearby diseases. After return from database, according to information diseases are shown on the map with map marker. Farness of diseases and type of the disease is also shown when a user touch on a map marker for a disease.

Upload Photo Sequence Diagram


User
Camera UI
mainController : QueryController
Server
take photos and upload
CreateQuery(photo[], chatbot)
CreateQuery(photo[], chatbot)

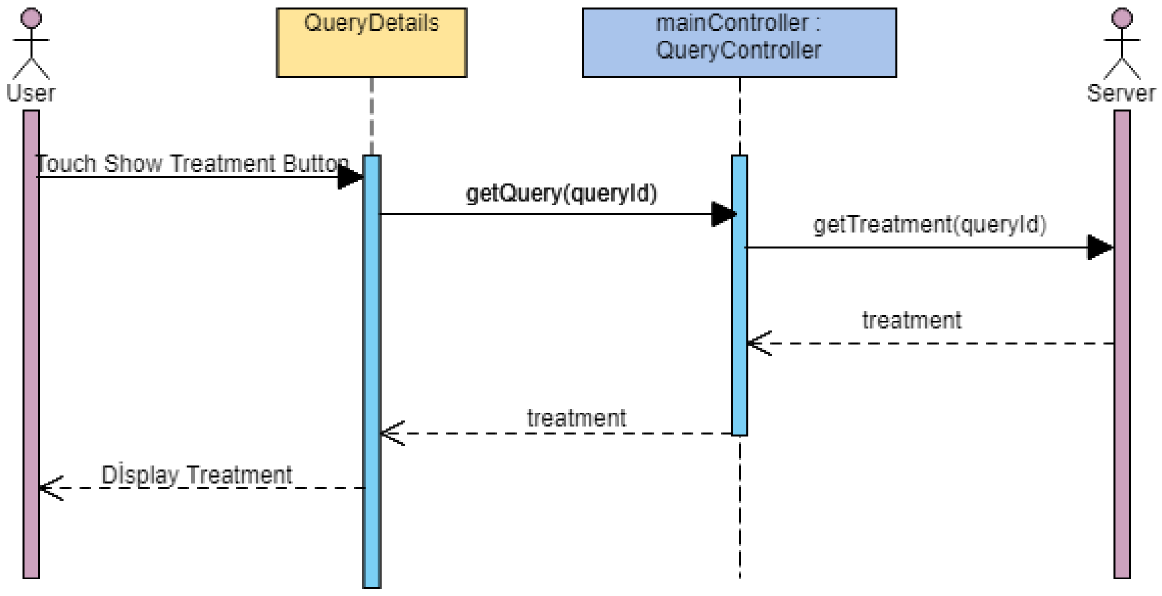
When a user wants to upload photo from "Camera" UI, the system creates a new query and sends its information to server side for both evolution and for database record.

About Us Sequence Diagram



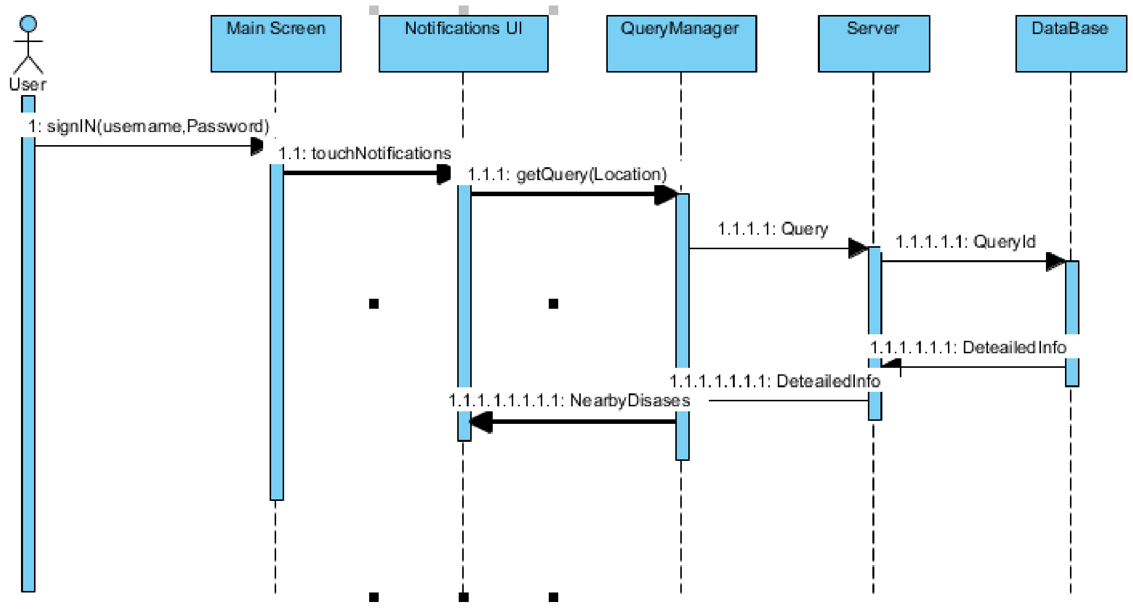
When a user touch on “About Us” menu item on side menu, he/she is forward to “About Us” UI and text about developers and the application.

Treatment Sequence Diagram



When a user wants to see treatment about the disease that query returns as a result, he/she touches the “Show Treatment” button on QueryDetails page and recommended treatment is shown on the screen.

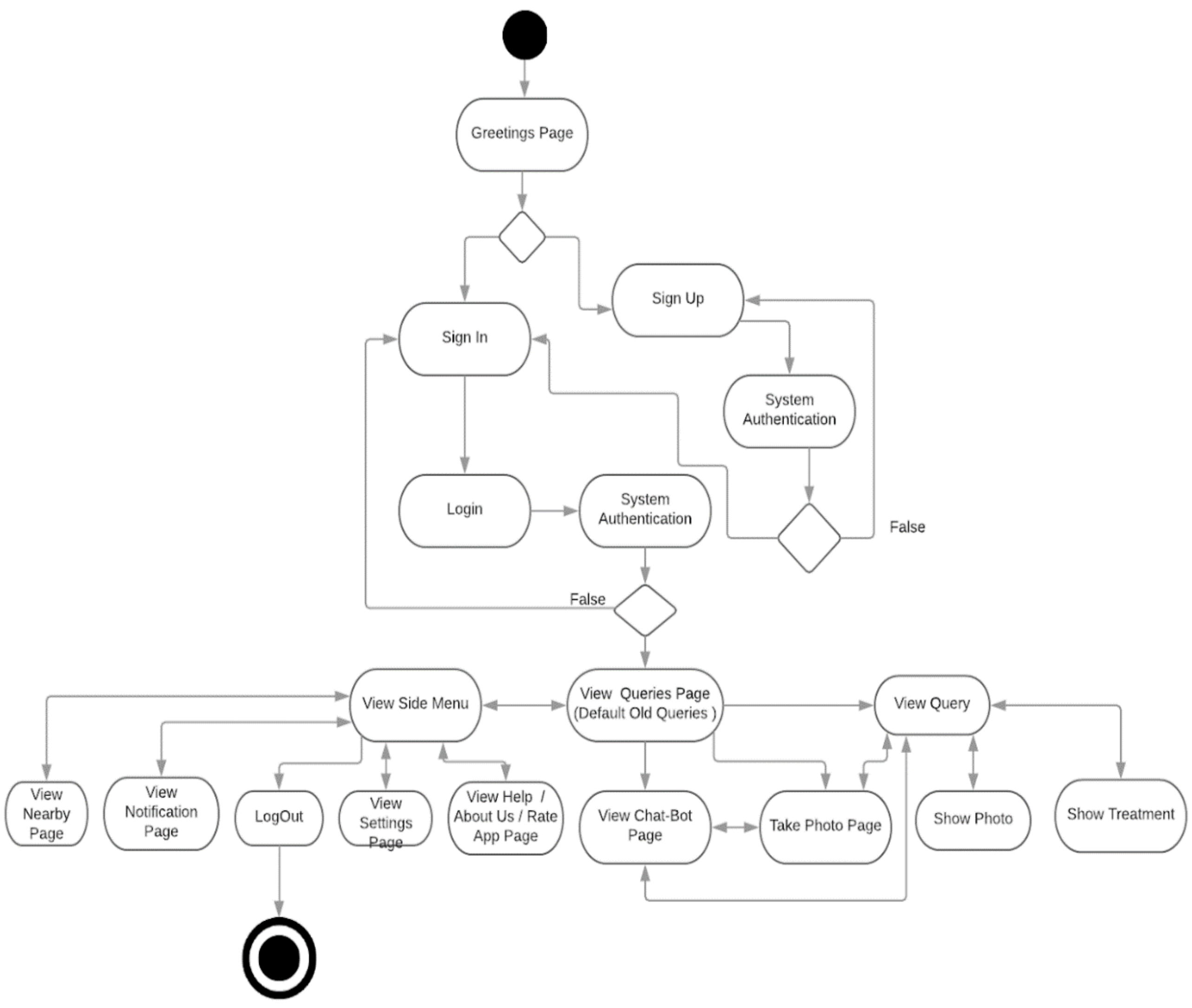
Notification Sequence Diagram


User
Main Screen
Notifications UI
QueryManager
Server
DataBase
1: signIN(username,Password)
1.1: touchNotifications
1.1.1: getQuery(Location)
1.1.1.1: Query
1.1.1.1.1: QueryId
1.1.1.1.1.1: DeteailedInfo
1.1.1.1.1.1.1: DeteailedInfo
1.1.1.1.1.1.1.1: NearbyDisases

When a user touch “Notitication” menu item on side menu, all notifications about nearby diseases and local weather will be shown on the “Notification” page.

3.5.4.2. Activity Diagrams

Application Flow Activity Diagram



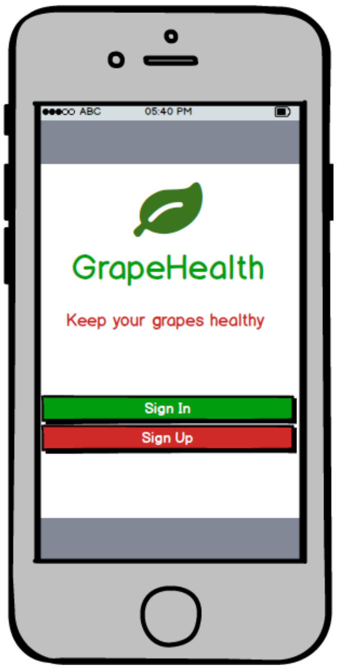
In this diagram, the user first views the greeting page. Sign-in and sign-up option is up to the user. After the system authentication the user will first view the queries page. If the user is new this view will not show queries, user needs to add. If the user had already created queries user can see old queries. From queries page user can create a new query by adding/taking photo or using chatbot. By side menu of the application the user can navigate through the app as shown above in the figure.

3.5.5. User Interface

The GrapeHealth application is a mobile application for both Android and iOS devices. The application aims to have easy to use user interface. We will have 15 different frames.

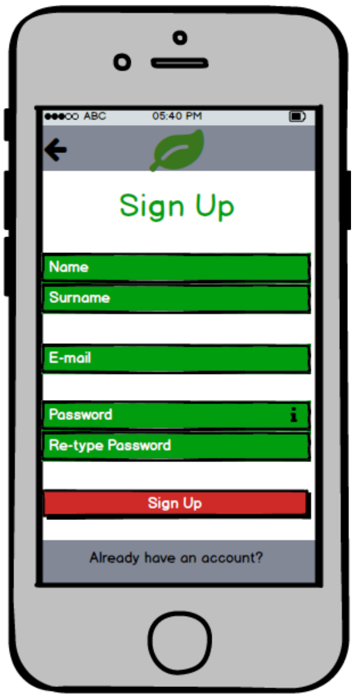
3.5.5.1. Greetings

This panel welcomes users and forward them to “Sign In” or “Sign Up” panels.



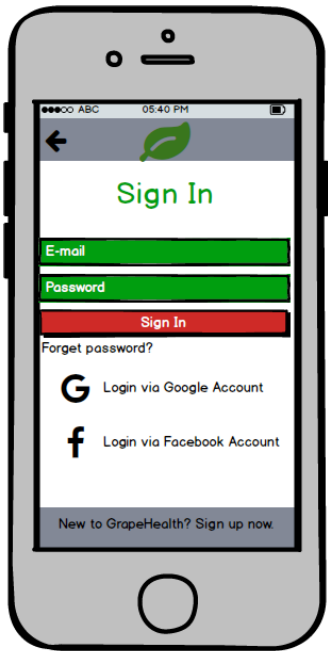
3.5.5.2. Sign Up

Users could sign up the application by using this panel. This panel requires from user to enter Name, Surname, E-mail information and create a password for his/her account. If the user has already an account by “Already have an account?” link he/she is forward to “Sign In” panel.



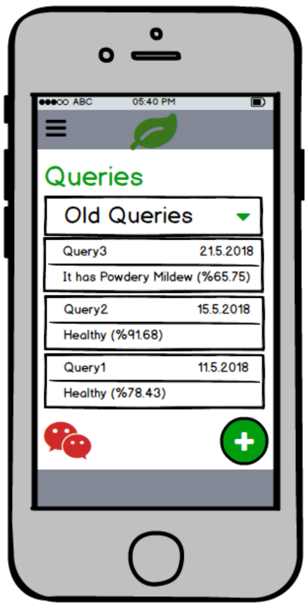
3.5.5.3. *Sign In*

A user could sign in to his/her account by using this panel. The user could sign in by using email/password information, Google account or Facebook account. If the user does not have a account yet, he/she could forward to “Sign Up” page by using “New to GrapeHealth? Sign up now.” link. When the used sign in, he/she is forwarded to “Queries” panel.



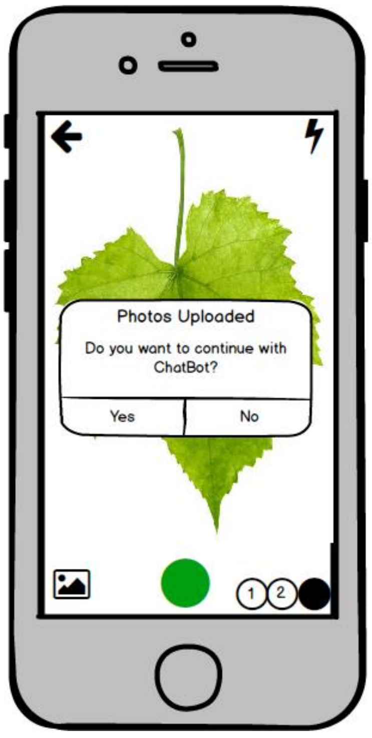
3.5.5.4. *Queries*

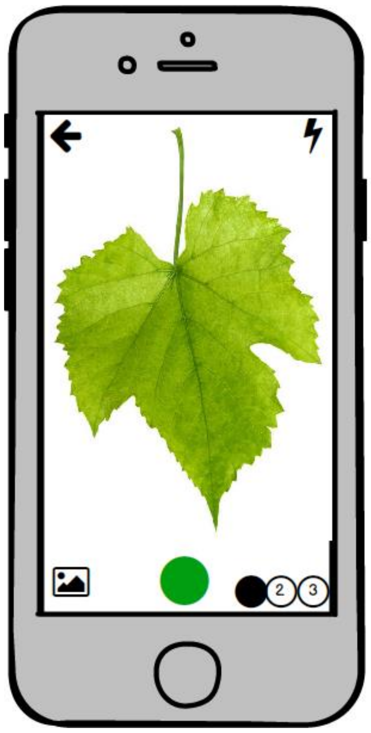
In this panel old queries will be listed. The user could see old queries date and briefly information about them. For detailed information the user could click on a query and program will be forwarded to “Query Details” page. Moreover, the user could send a new query by clicking plus sign at the right down corner. This button forwards him/her to “Camera” panel. There will be chatbot button at the left down corner. The user will be forward to “ChatBot” panel by this button. This panel also has a menu logo at the left upper corner. This button opens a menu component.



3.5.5.5. *Camera*

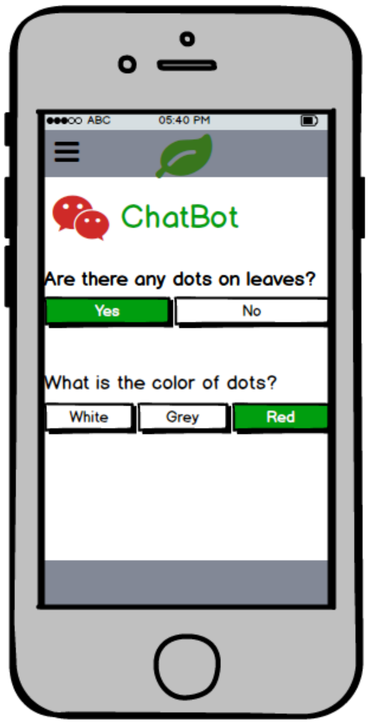
In this page a user could take photos of leaves to upload a query. After taking third photo, the user is asked to if he/she wants to continue with a chatbot.





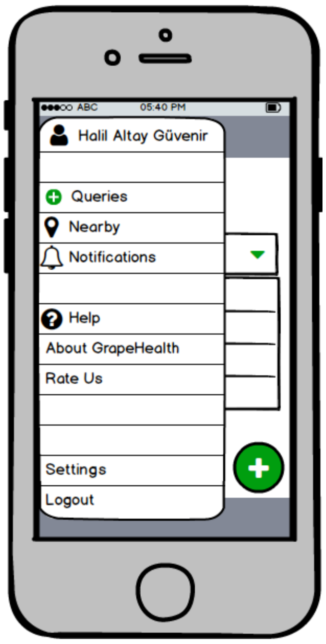
3.5.5.6. *Chatbot*

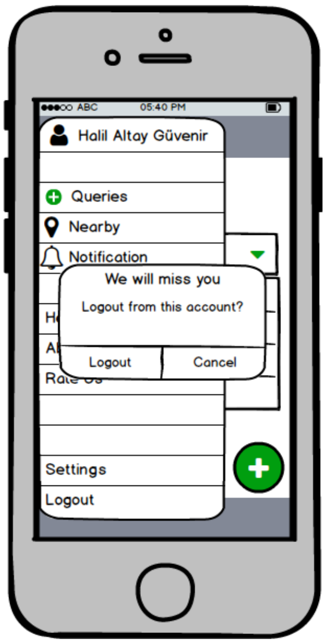
In this page a user is asked some questions so that, disease of the grape plant could be determined.



3.5.5.7. Menu

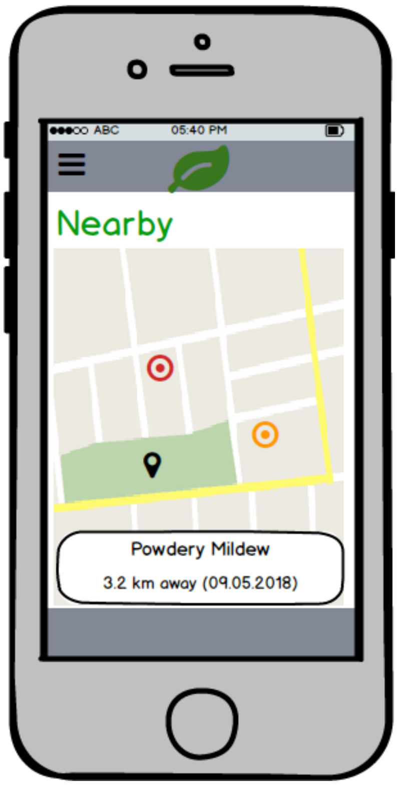
Actually “Menu” is not a distinct page. It is a navigator so that user could change to other pages. Menu option will be active in these pages: Queries, Nearby, QueryDetails, Notification, Settings, ChatBot, Help, About and by using this panel user could logout.





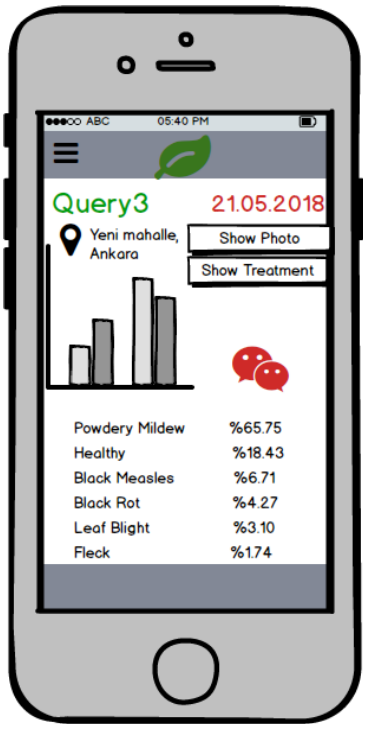
3.5.5.8. Nearby

“Nearby” page shows detected grape diseases nearby the user on the map. When the user tounch on a map marker, name of the disease, its farness to current location and date will be display.



3.5.5.9. QueryDetails

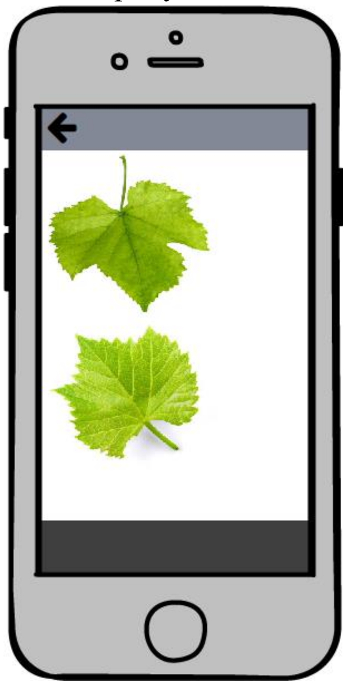
When a user wants to see details about old queries, he/she can touch on the query on "Queries" page. QueryDetails page display a graph and probability of diseases that the plant could has. Moreover, if the user did not use chatbot when he/she upload the photo by using chatbot, the user could give more information about the query via chatbot. In this page, show photo button forwards user to "OldPhotos" page and show treatment button forwards user to "Treatment" page. This page also displays date and location that the query was sent.



Powdery Mildew	%65.75
Healthy	%18.43
Black Measles	%6.71
Black Rot	%4.27
Leaf Blight	%3.10
Fleck	%1.74

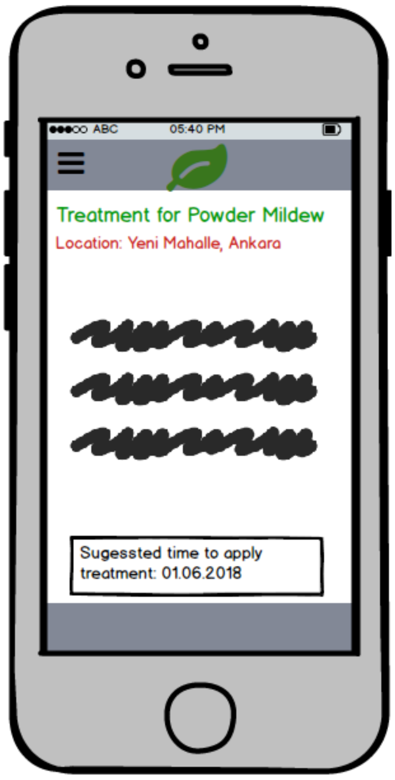
3.5.5.10. OldPhotos

This page displays old photos the query that is selected in "QueryDetails" page.



3.5.5.11. Treatment

In this page suggested treatments for found disease is given according to location and time of the year. Moreover, this page also displays best time to apply treatment according to location and weather of the location.



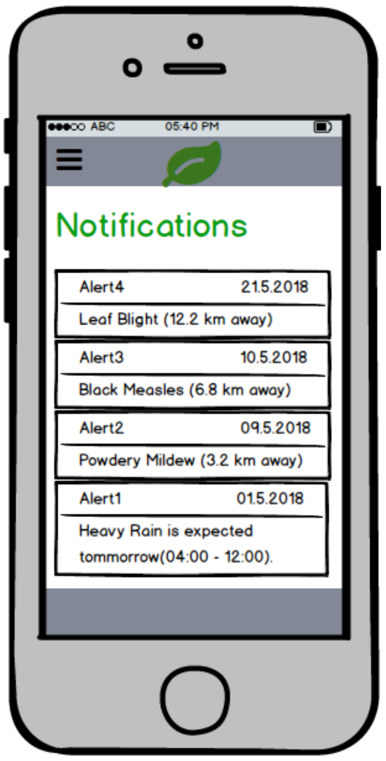
Treatment for Powder Mildew

Location: Yeni Mahalle, Ankara

Sugessted time to apply treatment: 01.06.2018

3.5.5.12. Notifications

"Notifications" page displays alerts about nearby diseases and weather conditions.



Notifications

Alert4 21.5.2018
Leaf Blight (12.2 km away)

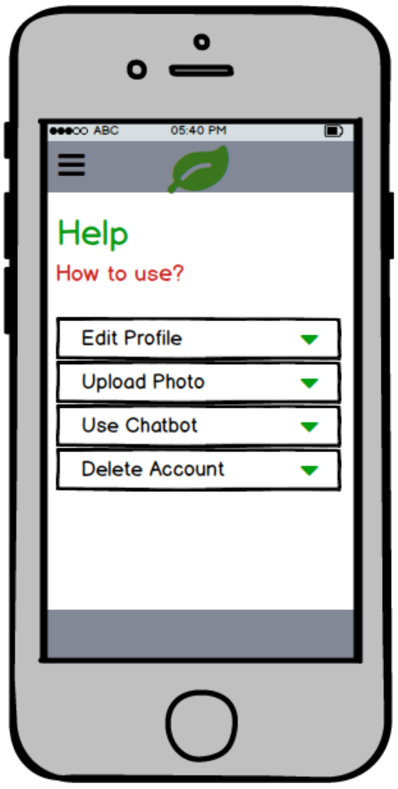
Alert3 10.5.2018
Black Measles (6.8 km away)

Alert2 09.5.2018
Powdery Mildew (3.2 km away)

Alert1 01.5.2018
Heavy Rain is expected tommorrow(04:00 - 12:00).

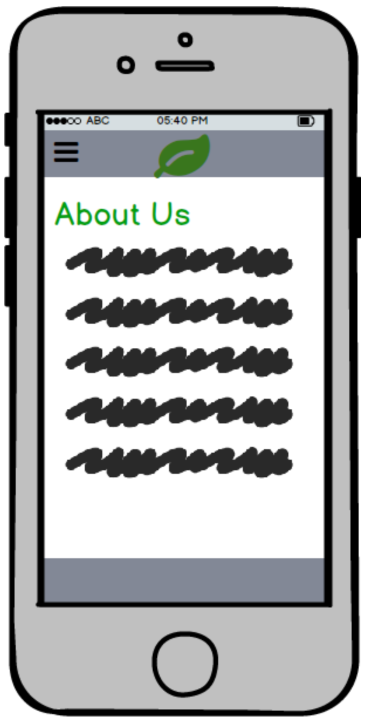
3.5.5.13. Help

This page helps users to use the application. This page has information about "How to edit profile?", "How to upload photo?", "How to use chatbot?", and "How to delete an account?".



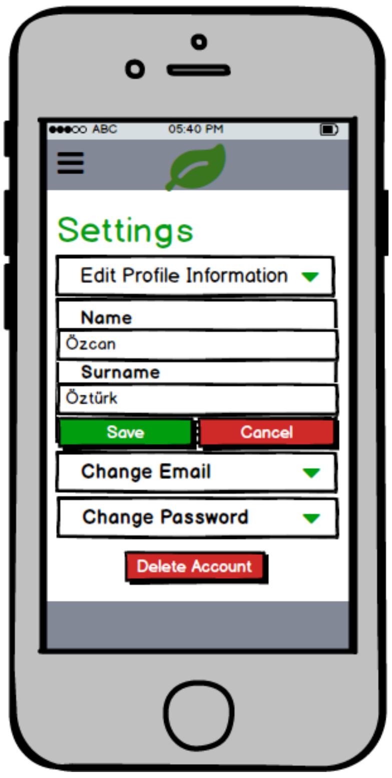
3.5.5.14. About

This page gives information about developers and the application.



3.5.5.15. *Settings*

A user could change profile information, email address and password by using this page. In this page the user could also delete his/her account.



REFERENCES

[1] PlantNet Plant Identification
https://play.google.com/store/apps/details?id=org.plantnet&hl=tr
[2] Saurel, Sylvain. “Learn to Use WebSockets on Android with OkHttp – Sylvain Saurel – Medium.” *Medium*, Medium, 14 Feb. 2017, medium.com/@ssaurel/learn-to-use-websockets-on-android-with-okhttp-ba5f00aea988.