*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Dr Horticulture v1.0

**Team Members:** Jose Nunez & Cesar Reyes

**Product Owner(s)**: Dr. Khoddamzadeh

**Mentor(s)**:

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of the Dr. Horticulture 1.0 iOS application. The purpose of the system is to decrease the pollution that nitrogen fertilizers produce by detecting how much fertilizer a plant needs based on an image. By accurately measuring the nitrogen content of a plant, farmers can reduce their use of fertilizers. Doing so helps the environment, as the excess nitrogen would otherwise reach masses of water and pollute the marine life. Further, the farmers are able to reduce their fertilizer’s costs by only using the amount of fertilizer the plant actually needs.*

*The levels of nitrogen in a plant are calculated by scanning the color of the leafs. This is done by applying a series of filters to the image in order to remove noise and other objects, and then scanning each leaf’s pixels to get their HSV (Hue, Saturation, Brightness) values. With those values, the pixels colors can be calculated into an index that represents how green the plant is, which then can be used to make recommendations on whether the plant needs fertilizing or not.*

**Table of Contents**

**Introduction** …………………………………………………………………………………………………………………………….. 5

Current System ……….…………………………………………………………………………………………………………... 5

Purpose of New System …………………………………………………..…………………………………………………... 5

**User Stories**

Sprint 2 ………………………………………….………………………………………………………………………………….. 7

Sprint 3 ………………………………………….…………………………………………………………………………………. 12

Sprint 4 ………………………………………….…………………………………………………………………………………. 19

Sprint 5 ………………………………………….…………………………………………………………………………………. 22

Sprint 6 ………………………………………….…………………………………………………………………………………. 33

**Project Plan**

Hardware and Software Resources ……………………………………………………...……………………….… 34

Sprints Plan ……………………………………………………………..………………………………………………………. 35

*Sprint 2*  …………………………………………………………………………………………………………………………... 35

*Sprint 3*  …………………………………………………………………………………………………………………………... 35

*Sprint 4*  …………………………………………………………………………………………………………………………... 36

*Sprint 5*  …………………………………………………………………………………………………………………………... 36

*Sprint 6*  …………………………………………………………………………………………………………………………... 36

**System Design**

Architectural Patterns ………………………………………………………………………………………………….. 37

System and Subsystem Decomposition …………………………………………………………………………….… 37

Deployment Diagram …………………………………………………………………………………………………….…... 38

**System Validation**  …………………………………………………………………………………………………………………….39

**Appendix**  ………………………………………………………………………………………………………………………………….43

Appendix A - UML Diagrams ………………………………………………………………………………………………. 43

Appendix B - User Interface Design …………………………………………………………………………….…... 58

Appendix C - Sprint Review Reports …………………………………………………………………………...…… 69

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents …………………………………………………………………….…………… 75

**References** ……………………………………………………………………………………………………………………...………...76

# Ⅰ-Introduction

Over fertilization is an ever growing issue as excess nitrogen from the fertilizer threatens to harm our ecosystem. Unused nitrogen from plants becomes nitrogen runoff as it get washed away towards water sources. These water sources then become contaminated with toxic algal blooms and death of the aquatic life their. The Dr Horticulture IOS app attempts to solve this issue for flowering plants by analyzing a plants green color from an image to determine its nitrogen levels. Which it can then use to provide an accurate fertilizer recommendation that does not exceeds the plants need nor starves the plant. The app is able to do this thanks to the research on flowering plants done by Dr. Khoddamzadeh.

## Current System

Currently for the domain of flowering plants the only methods of determining a plants nitrogen level are very costly and/or invasive. These methods tend to damage the individual plant through clippings or leaf samples, are to time consuming requiring weeks of lab work, or are to expensive due to lab fees for each plant sample and expensive equipment. This makes it difficult to not over fertilize individual plants since there is no efficient method to determine how much fertilizer a particular plant needs. Outside of the domain of flowering plants there exist a similar application geared towards corn and a handful of other plants related to corn.

## Purpose of New System

The Dr. Horticulture app serves to breach the cost, time, and invasiveness of current systems. Through the use image processing the app is able to provide a fertilizer recommendation without the need to damage the plant itself. The time needed for the results much faster in the scale of minutes rather than weeks. As well as lowering cost since no expensive equipment has to be bought and no laboratory payments are needed. In addition the new system allows for small time growers and home growers who can't justify the cost-benefit of current systems to be able to still properly fertilize.

# Ⅱ-User Stories

The following section provides the detailed user stories that were implemented in this iteration of the Dr Horticulture project. These user stories served as the basis for the implementation of the project’s features. The user stories are displayed along with their corresponding use cases and use case diagrams, sequence diagrams, and class diagrams. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

|  |  |
| --- | --- |
| Sprints | Completed User Stories |
| Sprint 2 | #667-Create Homepage  #668-Create Plant Selection View  #669-Setup Server |
| Sprint 3 | #670-Create Tab-Navigation & Navigation Controllers  #671-Add Camera Function & Image Selection  #675-Update Matlab Image Processing |
| Sprint 4 | #672-Integrate Firebase Database  #673-Improve the App’s Appearance & Layout  #676-Create Wrapper for Server & Image Processing |
| Sprint 5 | #674-Implement User Authentication (Login/Signup) |
| Sprint 6 | #677-Create Fertilizer Recommendation View |

***Table ⅱ.1:*** Implemented User Stories

**Sprint 2:**

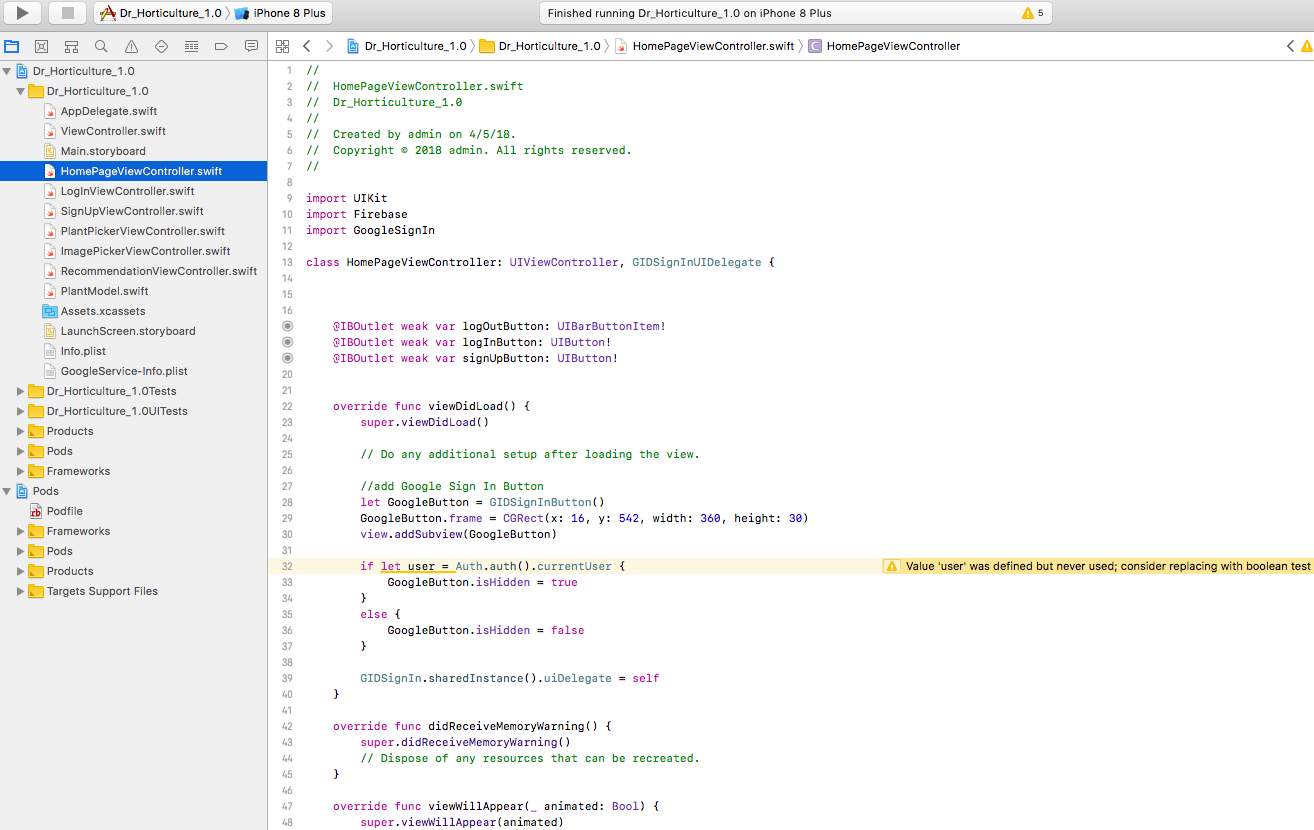
User Story #667: Create Homepage

**Description:** **As a** developer, **I would like to** have the application open to a homepage, **so that** users are greeted with a logo, description, and buttons to login and navigate the app.

**Acceptance Criteria:**

* Loads with proper appearance and layout on all IOS devices
* Button clicks respond with appropriate interactions

**Visual User Guide:**

****

User Story #668: Create Plant Selection View

**Description:**

* **As a** user, **I would like to** have a plant selection page. **so that** I can select the species of plant I plan to use the app for
* **As a** developer, **I would like to** have navigation buttons on the homepage and plant selection view, **so that** users can transition from the home page to the plant selection view and back

**Acceptance Criteria:**

* View loads with proper appearance and layout on all IOS devices
* Transition form the homepage to the plant selection view occurs correctly
* Table view properly loads with the correct data
* Selection is correctly displayed with the corresponding plant name
* back button properly returns to the homepage

**Use Case:**

* Name: Plant Selected
* Actor: User, Database, View-Controller
* Preconditions: 1. User has started the application and is in the homepage view.
* Description: 2. User clicks to transition to Plant Selection View

3. View-controller switches the view displayed

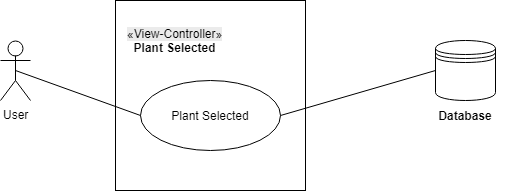
4. Data query made to database for list of plants.

5. User selects a plant

6.View-controller saves plant to var, switches to next view and displays plant

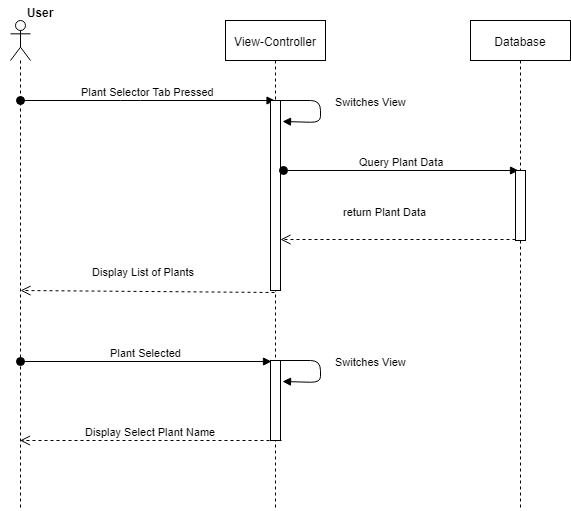
name

**Use Case Diagram:**

****

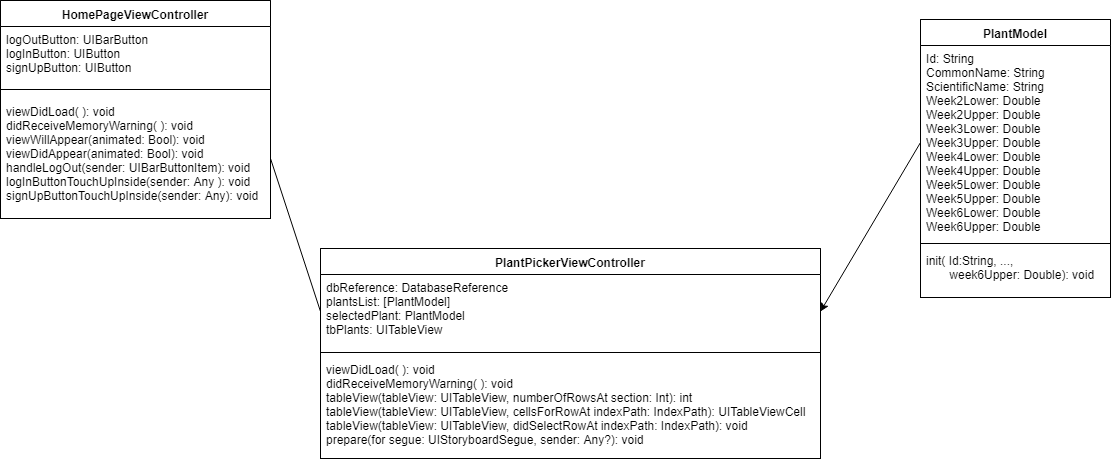
***Figure ⅱ.1:*** Plant Selected, Use Case Diagram

**Sequence Diagram:**

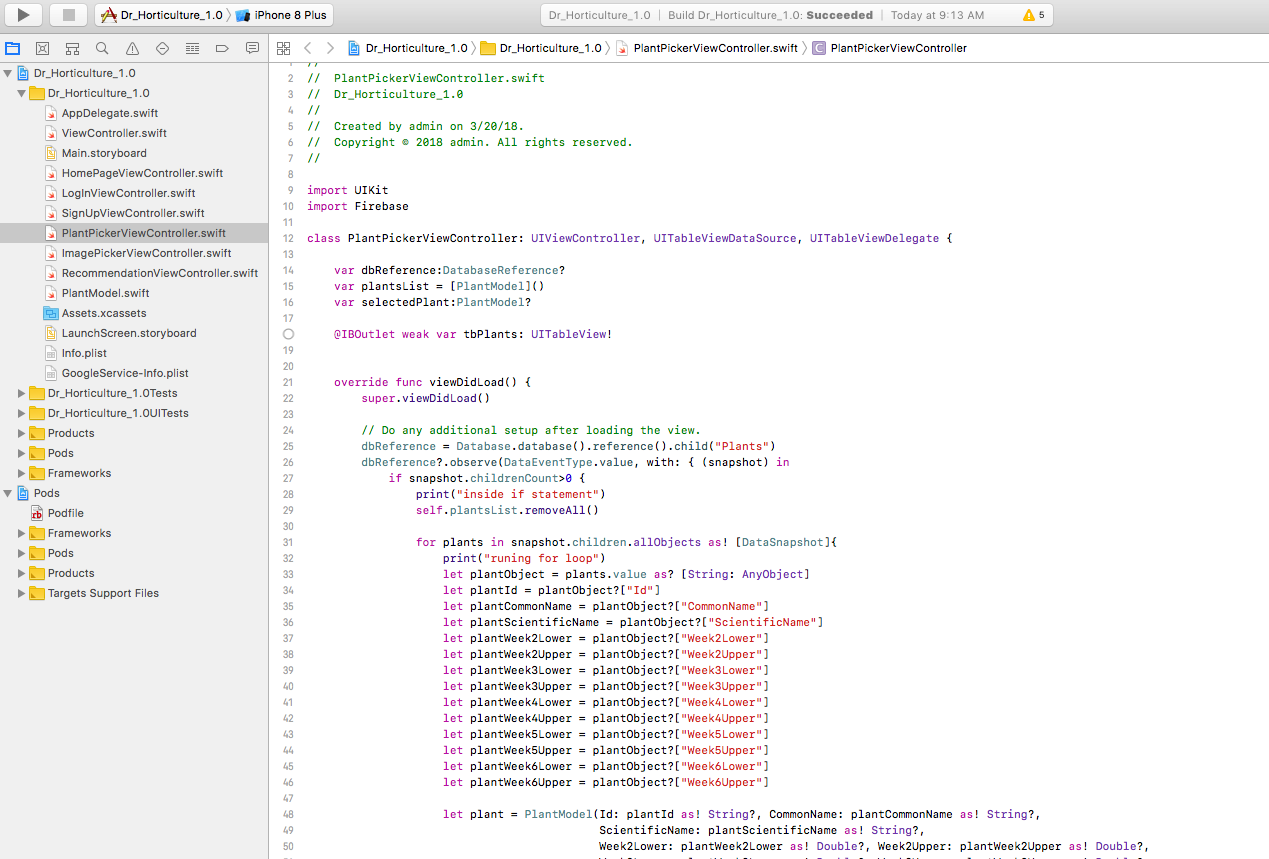


***Figure ⅱ.2:*** Plant Selected, Sequence Diagram

**Class Diagram:**

***Figure ⅱ.3:*** User Story #668, Class Diagram

**Visual User Guide:**



**Sprint 3:**

User Story #670: Create Tab-Navigation & Navigation Controllers

**Description: As a** Developer, **I would like to** create a tab-bar navigation & navigation controllers, **so that** users can seamlessly transition from one view to another and back throughout the application

**Acceptance Criteria:**

* Tab view appears in all views with appropriate titles and images for the corresponding tabs
* Clicking on the tab changes view to appropriate tab and clicking back on previous tab returns it to the state in which it had previously been in
* Navigation title and return buttons properly appear at the top of the application for each view
* return buttons on the navigation bar perform transition back to previous view

**Visual User Guide:**

User Story #671: Add Camera Functionality and Image Selection

**Description:**

* **As a** Developer, **I would like to** allow users select existing images from their photo gallery, **so that** they can upload it to the server for results.
* **As a** Developer, **I would like to** allow users invoke the camera app., **so that** they can take a photo of the plant and upload it to the server for results.

**Acceptance Criteria:**

* View for image selection/tacking is rendering properly
* User is able to set the weeks since planted dropdown menu
* User is able to select an image from their photo library
* Image selected appears in the image view as a preview

**Use Case:**

* Name: Select Image from Photo Library
* Actor: User, View-Controller, IOS Photo Library
* Preconditions: Plant was selected in PlantPicker view and ImagePicker view is rendered
* Description: 1. User clicks import image button

2. The View-Controller opens IOS photo library, asking for permission to access photos if not previously granted

3. User Selects an image from their photo library

4. View-Controller switches back to application and displays the image in the UIImageView of the ImagePicker view

**Use Case:**

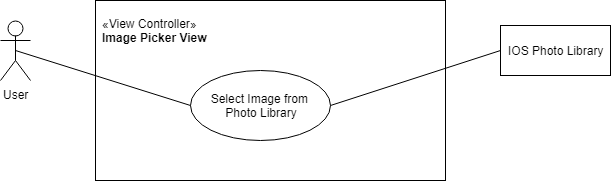
* Name: Take Photo Through IOS Camera Application
* Actor: User, View-Controller, IOS Camera App
* Preconditions: Plant was selected in PlantPicker view and ImagePicker view is rendered
* Description: 1. User clicks import image button

2. The View-Controller opens IOS camera application, asking for permission to access camera if not previously granted

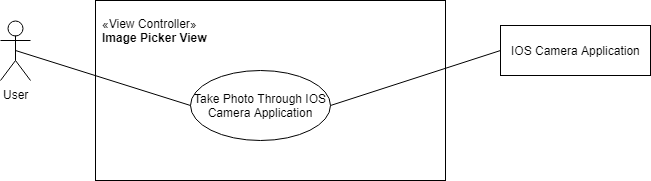
3. User takes a photo of the plant

4. View-Controller switches back to application and displays the image in the UIImageView of the ImagePicker view

**Use Case Diagrams:**

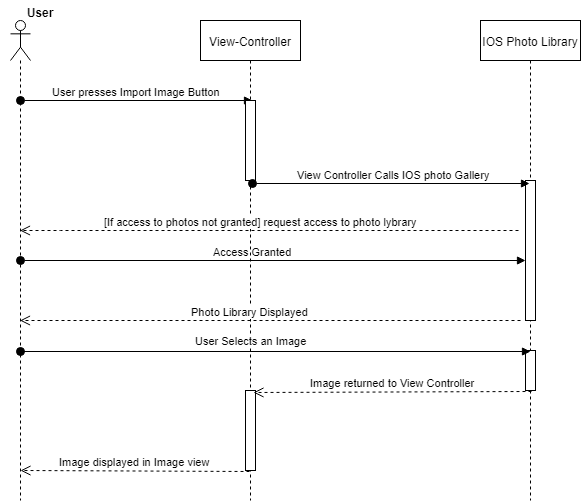


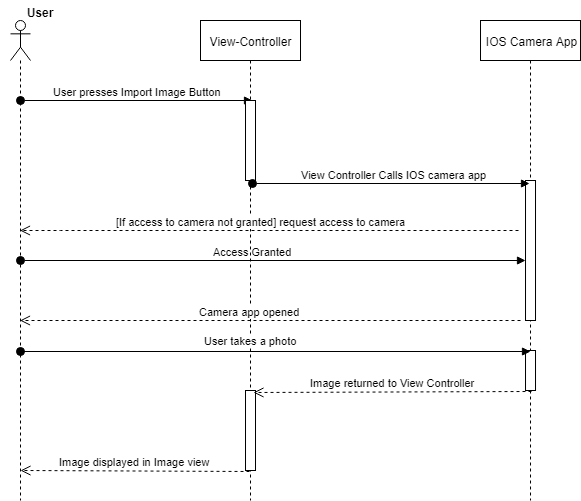
***Figure ⅱ.4:*** Select Image from Photo Library, Use Case Diagram

****

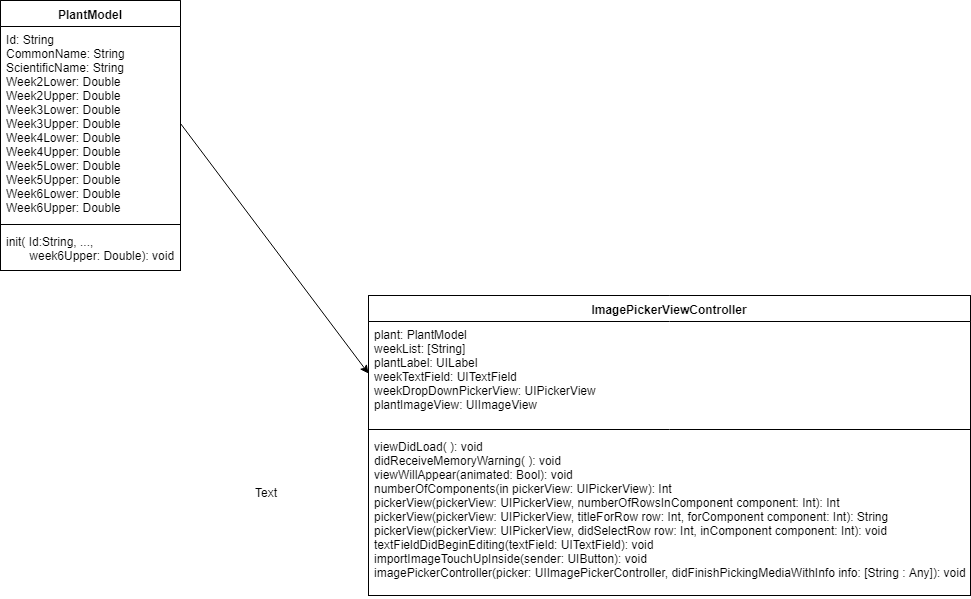
***Figure ⅱ.5:*** Take Photo Through IOS Camera Application, Use Case Diagram

**Sequence Diagram:**

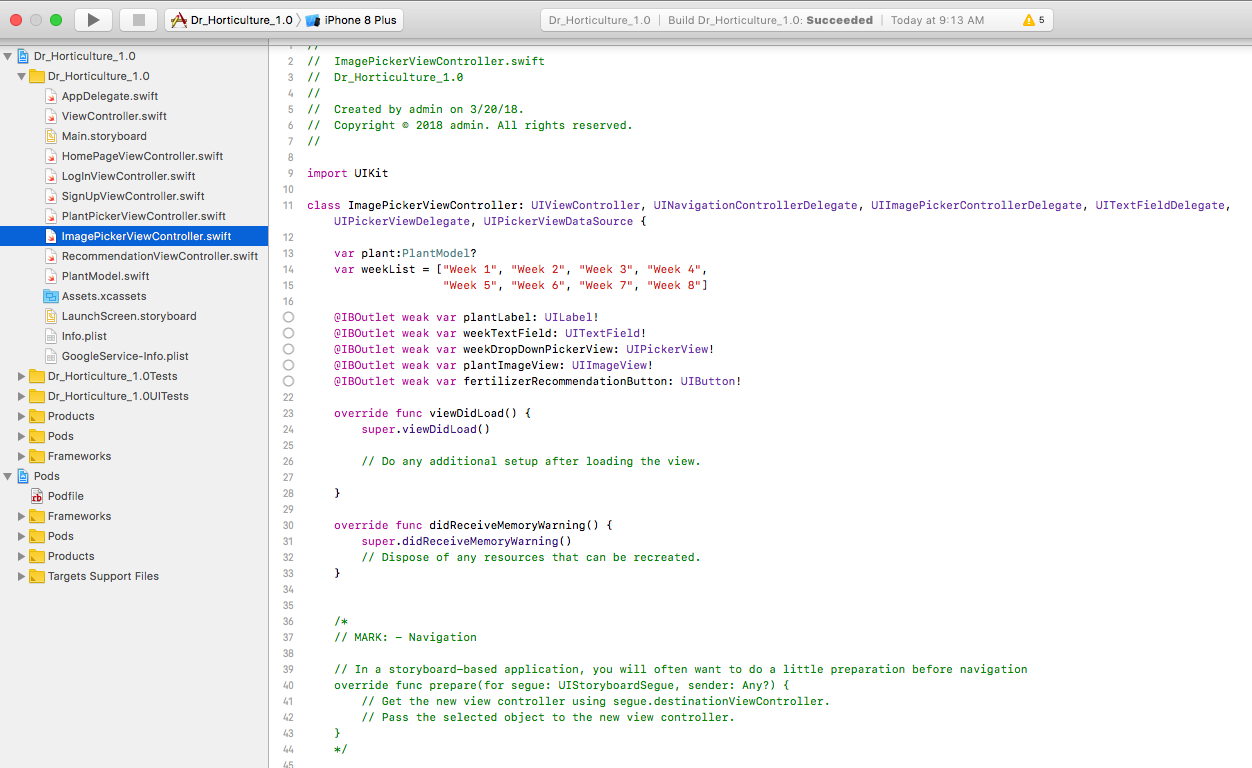
***Figure ⅱ.6:*** Select Image from Photo Library, Sequence Diagram

***Figure ⅱ.7:*** Take Photo Through IOS Camera Application, Sequence Diagram

**Class Diagram:**

***Figure ⅱ.9:*** User Story #671, Class Diagram

**Visual User Guide:**



**Sprint 4:**

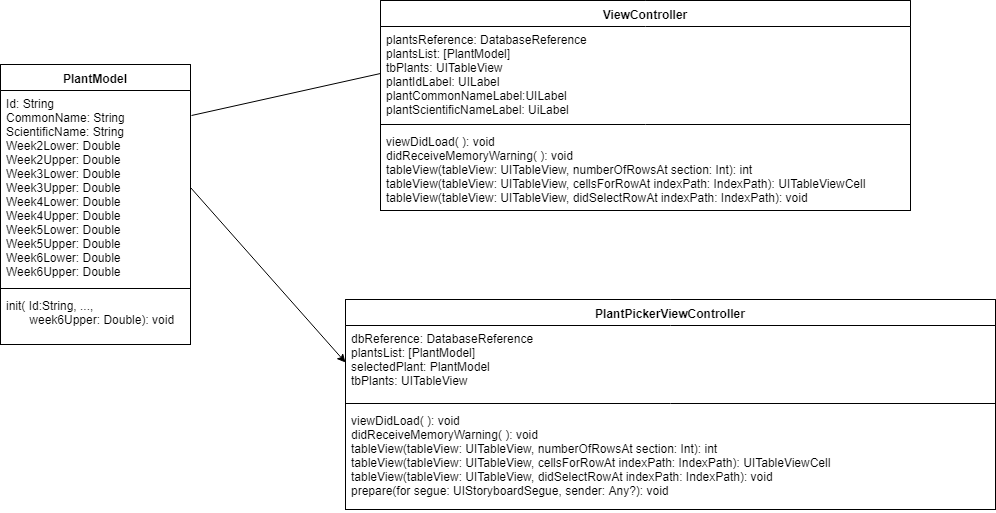
User Story #672: Integrate Firebase Database

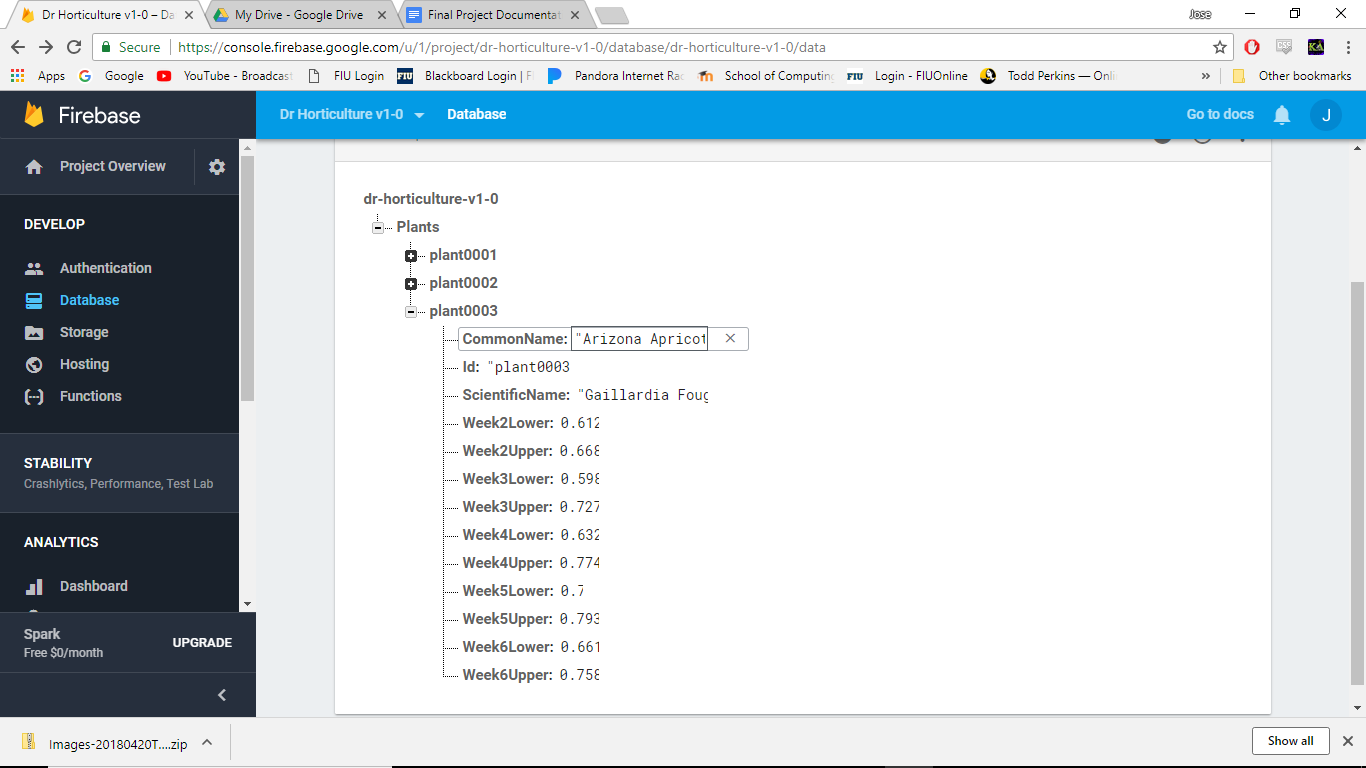
**Description:** **As a** developer, **I would like to** have a database setup with the plant data, **so that** the application can load the latest and most up to date plant data for plant selection and fertilizer recommendations

**Acceptance Criteria:**

* Firebase properly installed in the system along with the corresponding cocoa pods
* The real time database is set up on the Firebase console
* The data is queried and displayed on the application

**Class Diagram:**

***Figure ⅱ.10:*** User Story #672, Class Diagram

**Visual User Guide:** 

**User Story #673: Improve the App’s Appearance and Layout**

**Description :As a** product owner, **I would like the** look and layout of the app to better represent represent the purpose of the app, **so that** users find confidence in using the app and its fertilizer recommendations

**Acceptance Criteria:**

* The app better reflects the feel and image of the power-point provided by the product owner
* The descriptions on the app are better representative of the apps intentions as desired by the product owner

**Visual User Guide**

**Sprint 5:**

**User Story #674: Implement User Authentication (login/Signup)**

**Description:** **As a** product owner, **I would like to** have users login, **so that** monetization can be implemented later on as well as allow for user statistics to be generated

**Acceptance Criteria:**

* Users are able to signup
* Users are able to login
* Users can logout

**Use Case:**

* Name: Signup with Email
* Actor: User, View-Controller, Database
* Preconditions: User is in homepage and has not created an account
* Description: 1. User clicks Login button

2. View-Controller switch to Login View

3. User Enters Credentials (email, password, and a reentering of the password) and clicks continue

4. View-Controller checks if fields have been entered and if passwords match

5. View-Controller sends create user request to Database

6.Database returns success if successful or error otherwise

**Use Case:**

* Name: Login with Email
* Actor: User, View-Controller, Database
* Preconditions: User is in homepage and has previously created an account
* Description: 1. User clicks Login button

2. View-Controller switch to Login View

3. User Enters Credentials (email and password) and clicks continue

4. View-Controller checks if fields have been entered

5. View-Controller sends signin user request to Database

6.Database returns success if successful or error otherwise

**Use Case:**

* Name: Sign-in with Google
* Actor: User, View-Controller, Database, IOS Safari web-browser
* Preconditions:
* Description: 1. User clicks Google Sign in button

2. View-Controller opens Google URL in Safari web-browser

3. User Enables Google authentication by signing into Google account and allowing app access

4. Web-browser returns to Homepage View with google account instance

5. View-Controller sends create user request to Database with google account instance

6.Database returns success if successful or error otherwise

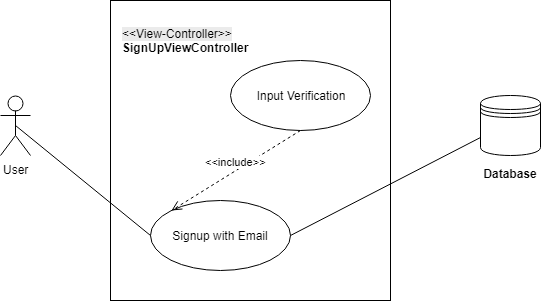
**Use Case:**

* Name: Logout
* Actor: User, View-Controller, Database
* Preconditions: User is currently logged into an account
* Description: 1. User clicks Logout bar-button

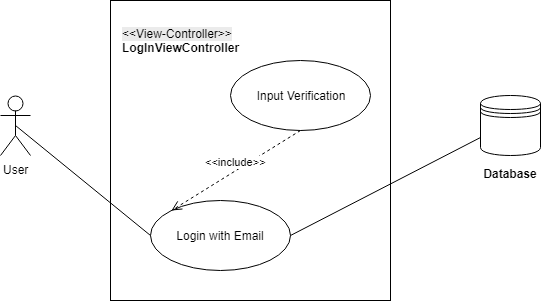
2. View-Controller sends logout request to database

3.Database returns success if user logged out or error otherwise

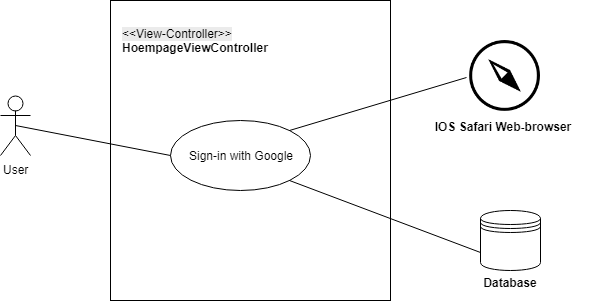
**Use Case Diagrams:**

****

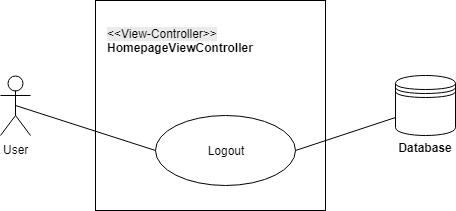
***Figure ⅱ.11:*** Signup with Email, Use Case Diagram

****

***Figure ⅱ.12:*** Login with Email, Use Case Diagram

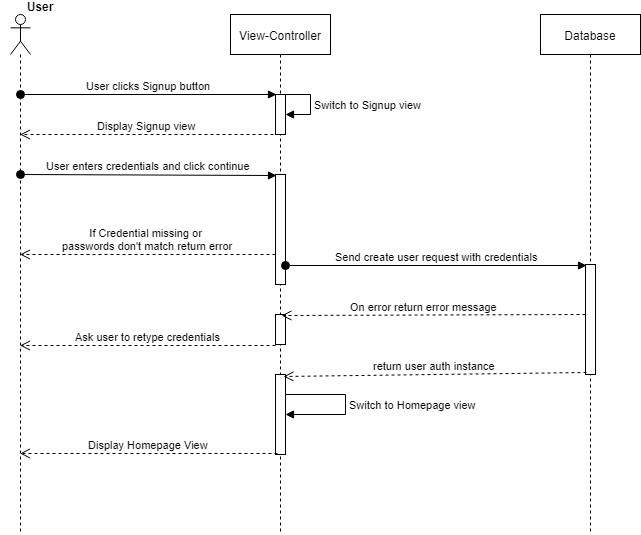
****

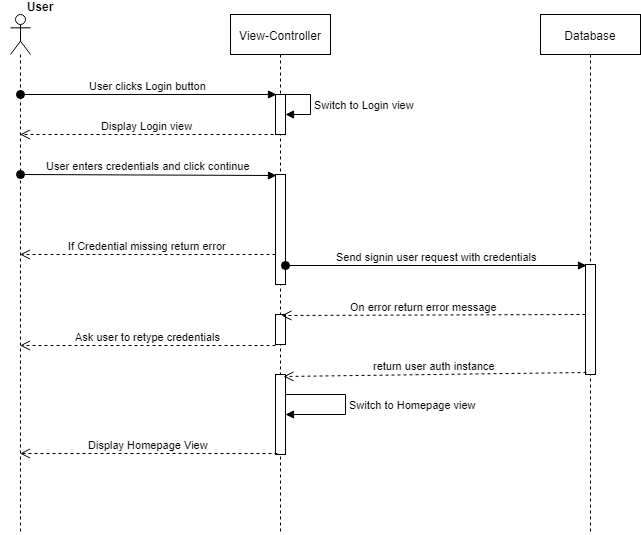
***Figure ⅱ.13:*** Sign-in with Google, Use Case Diagram

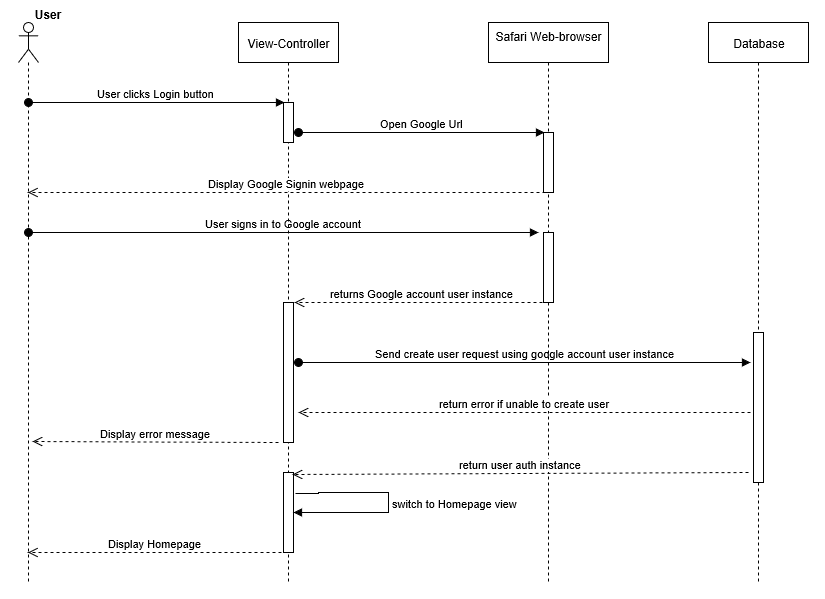
****

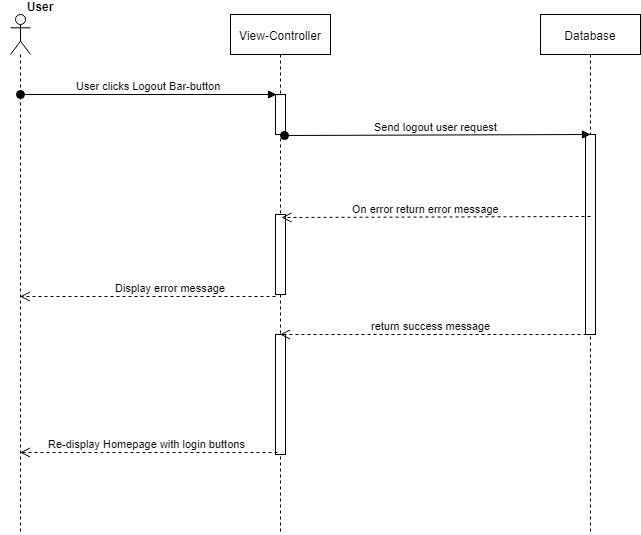
***Figure ⅱ.14:*** Logout, Use Case Diagram

**Sequence Diagrams:**

 ***Figure ⅱ.15:*** Signup with Email, Sequence Diagram

***Figure ⅱ.16:*** Login with Email, Sequence Diagram

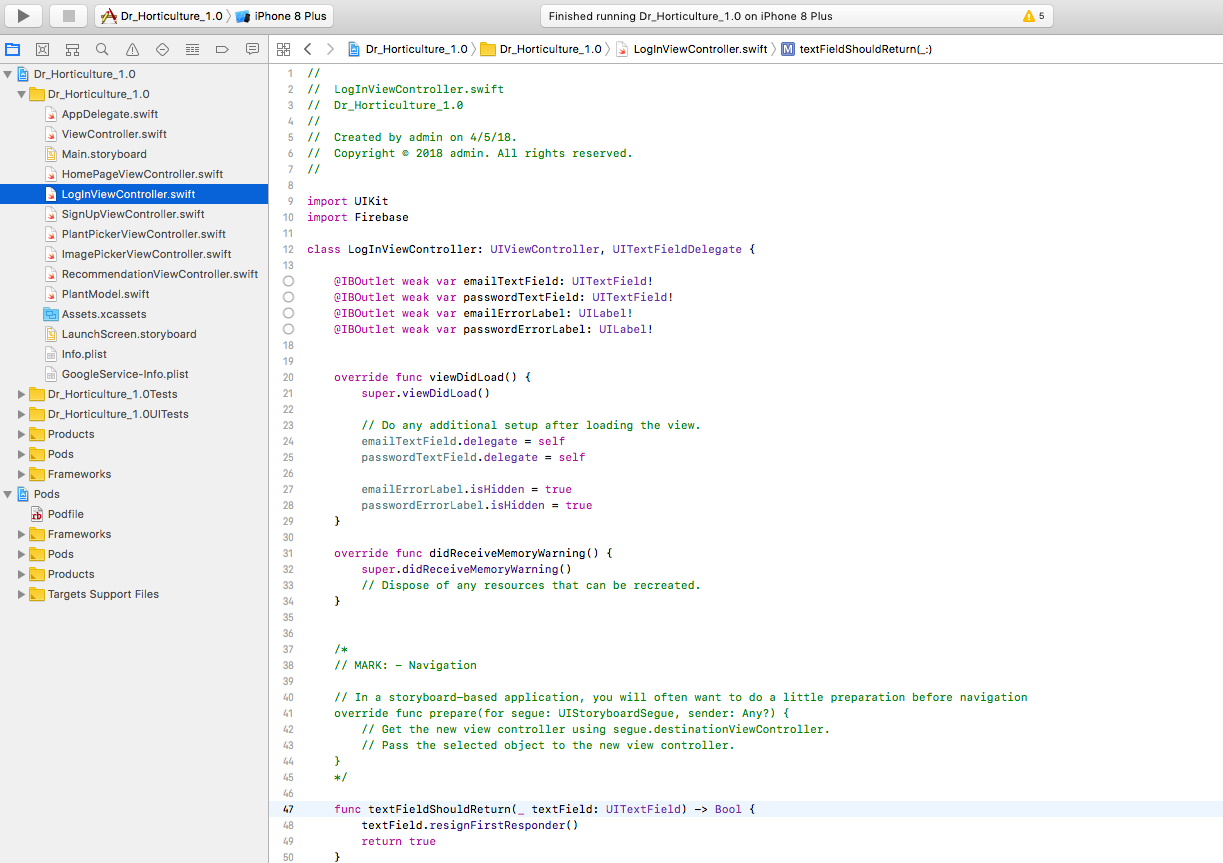
***Figure ⅱ.17:*** Sign-in with Google, Sequence Diagram

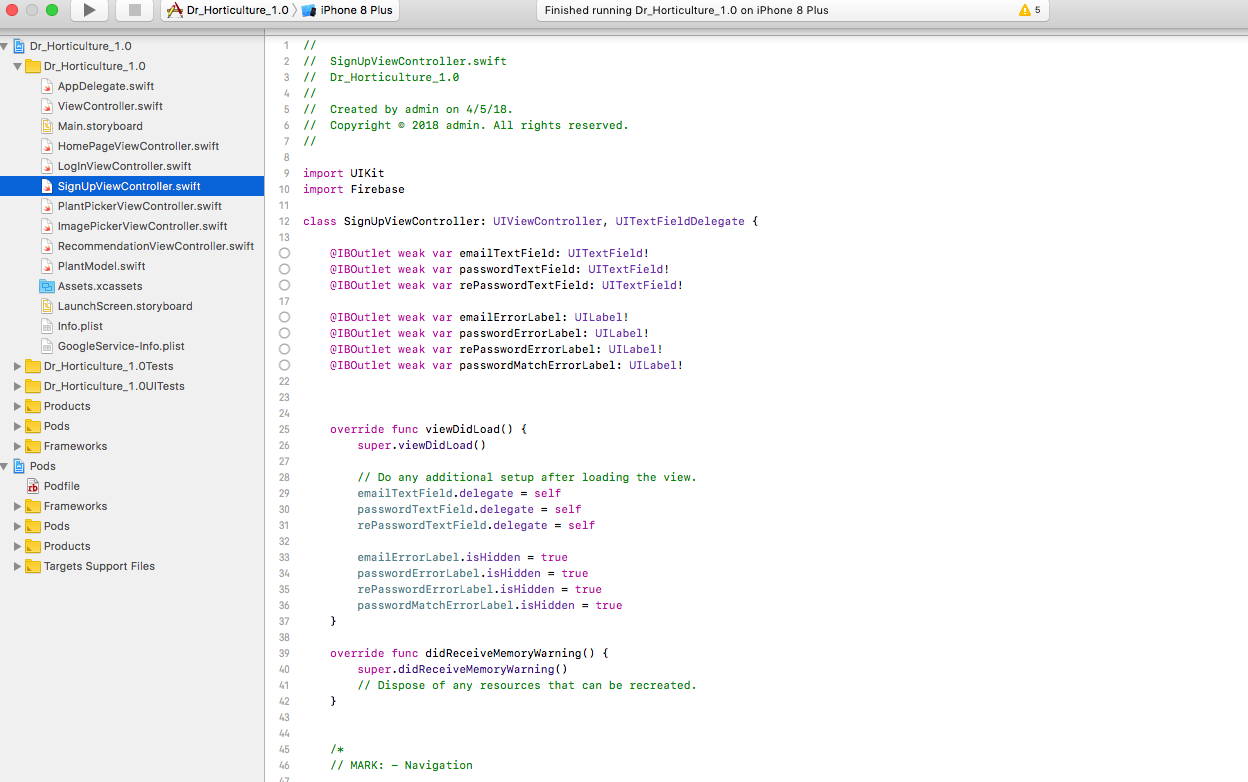
***Figure ⅱ.18:*** Logout, Sequence Diagram

**Class Diagram:**

***Figure ⅱ.19:*** User Story #672, Class Diagram

**Visual User Guide:**

****

****

**Sprint 6:**

# User Story #677 Create Fertilizer Recommendation view

**Description: As a** developer, **I would like for** there to be a view to display the fertilizer recommendation, **so that** the servers response can be displayed to the user

**Acceptance Criteria:**

* View load properly on device sizes
* Response is according to specific plant data and server results

# Ⅲ-Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

* **Xcode-** Apple’s IDE for IOS
* **Swift-** The programing Language used to program the app’s front-end
* **Firebase-** A Google real-time Database system

## 

## 

## Sprints Plan

### Sprint 2

Meeting 01/31/18

Attendees: <Dr. K, Jose Nunez, Cesar>

Start time: <3:00pm>

End time: <3:30am>

After discussion, the velocity of the team were estimated to be <50 hours>.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* #667-Create Homepage
* #668-Create Plant Selection View
* #669-Set up Server

The team members indicated their willingness to work on the following user stories.

* Jose Nunez
  + #667-Create Homepage
  + #668-Create Plant Selection View
* Cesar Reyes
  + #669-Set up Server

### ***Sprint 3***

Meeting 02/15/18

Attendees: <Dr. K, Jose Nunez, Cesar>

Start time: <10:50am>

End time: <11:30am>

After discussion, the velocity of the team were estimated to be <80 hours>.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* #670-Create Tab-Navigation & Navigation Controllers
* #671-Add Camera Functionality and Image Selection
* #675-Update MATLAB image processing code

The team members indicated their willingness to work on the following user stories.

* Jose Nunez
  + #670-Create Tab-Navigation & Navigation Controllers
  + #671-Add Camera Functionality and Image Selection
* Cesar Reyes
  + #675-Update MATLAB image processing code

***Sprint 4***

Meeting 02/26/18

Attendees: <Dr. K, Jose Nunez, Cesar>

Start time: <10:00am>

End time: <10:45am>

After discussion, the velocity of the team were estimated to be <60 hours>.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* #672-Integrate Firebase Database
* #673-Update the App Look and Layout

The team members indicated their willingness to work on the following user stories.

* Jose Nunez
  + #672-Integrate Firebase Database
  + #673-Update the App Look and Layout

***Sprint 5***

***Sprint 6***

Meeting 04/05/18

Attendees: <Dr. K, Jose Nunez, Cesar>

Start time: <4:15pm>

End time: <4:45pm>

After discussion, the velocity of the team were estimated to be <60 hours>.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* #677-Create Fertilizer Recommendation View

The team members indicated their willingness to work on the following user stories.

* Jose Nunez
  + #677-Create Fertilizer Recommendation View

# Ⅳ-System Design

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

The project was primarily designed with the use of two architectural patterns. Those architectural patterns are Model-View-Controller pattern and Client-Server pattern. The Model-View-Controller pattern was used for the design and creation of the front end. With the use of model-view-controller the separation of what is presented to the user along with user inputs, how the inputs are handled, and the functionality of the data. As for the Client-Server pattern, it was used in the designing of back-end. The pattern played a roll with both the design of the database and the server.

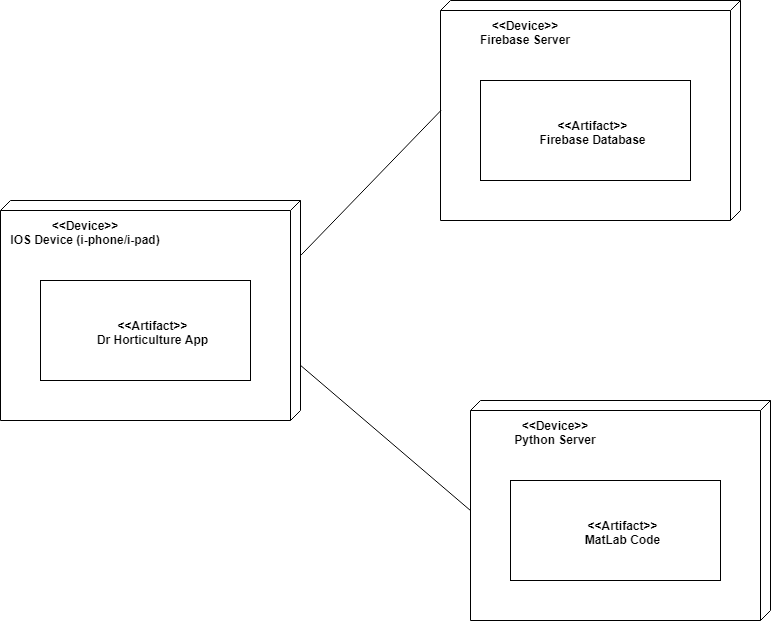
## System and Subsystem Decomposition

The front-end of the system is divided into views, which could be thought of as similar to web pages. Each view is composed of of the view as displayed in the main storyboard and the View-Controller of the view. The view in the story board has is part of Xcodes IDE and displays how the elements the user sees. The View-Controller determines the state of the app, data present in the view, and how the view itself changes or switches to a different view.

The back-end is divided into the database side and the server side. The database side is composed of a real-time Firebase database that is composed of user data and plant data. The plant data contains the the names of plants and the research data of Dr. Khoddamzadeh. The user data contains just the users email and password data along with user key.

## Deployment Diagram

The deployment diagram below shows the relation of the subsystems to one another.

***Figure ⅳ.1:*** Deployment Diagram

# Ⅴ-System Validation

|  |  |
| --- | --- |
| Test Case ID | Load\_TableView\_Data |
| Description/Summary of Test | Test to ensure that the database data loads in TableView of plant picker view |
| Pre-Condition | User click on the plant picker tab button |
| Expected Result | The Common names of the plants in the database are displayed |
| Actual Result | The Common names of the plants in the database were displayed |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | Select\_Plant\_From\_TableView |
| Description/Summary of Test | Test to see if Table view responds properly to plant selection |
| Pre-Condition | The user is in the plant picker view and clicks on “Amico Bronze” in the TableView |
| Expected Result | View switches to image picker view and displays “Amico Bronze” |
| Actual Result | View switched to image picker view and “Amico Bronze” was displayed |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | Week\_Since\_Planted\_Selected |
| Description/Summary of Test | Test to ensure that PickerView for week after planted appears when textbox is clicked and disappears when week selection made |
| Pre-Condition | Image picker view is present and PickerView is hidden |
| Expected Result | PickerView is made visible when user clicks on text box and the becomes hidden again when user scrolls down and makes week selection |
| Actual Result | PickerView became visible during selection and returned to being hidden when selection was complete |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | Back\_Button\_to\_Plant\_Picker\_View\_from\_Image\_Picker\_View |
| Description/Summary of Test | Test to see if the back button on the navigation bar of image picker switches view back to plant picker view |
| Pre-Condition | The user is in the image picker view |
| Expected Result | View switches to plant picker view |
| Actual Result | View switched to plant picker view |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | ImportImage\_Button\_Pressed |
| Description/Summary of Test | Test to see if ImportImage button callas IOS photo library |
| Pre-Condition | The user is in the image picker view and clicks Import Image |
| Expected Result | IOS photo library is opened up |
| Actual Result | IOS photo library was opened up |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | Photo\_Lyrary\_Permisions |
| Description/Summary of Test | Test to ensure that app properly request for permission to view photo library |
| Pre-Condition | The user is in the image picker view and clicks Import Image, & and has not previously granted the app permission |
| Expected Result | IOS photo library is opened up and request notification pop-up displayed |
| Actual Result | IOS photo library was opened up with request notification |
| Status (Pass/Fail) | Pass |

|  |  |
| --- | --- |
| Test Case ID | ImportImage\_Button\_Pressed |
| Description/Summary of Test | Test to see if ImportImage button callas IOS photo library |
| Pre-Condition | The user is in the image picker view and clicks Import Image |
| Expected Result | IOS photo library is opened up |
| Actual Result | IOS photo library was opened up |
| Status (Pass/Fail) | Pass |

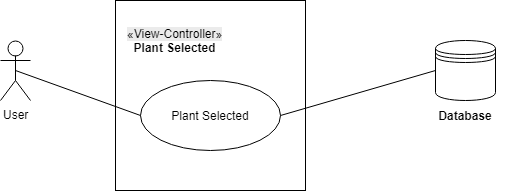
# 

# 

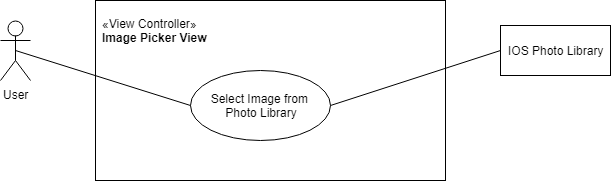
# Ⅶ-Appendix

## Appendix A - UML Diagrams

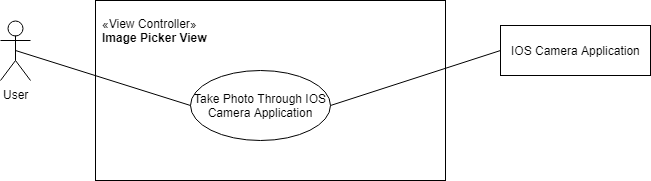
## Use Case Diagrams

****

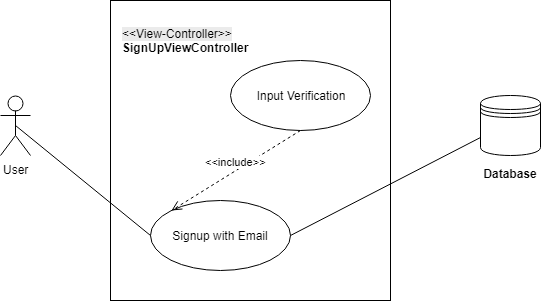
***Figure ⅱ.1:*** Plant Selected, Use Case Diagram



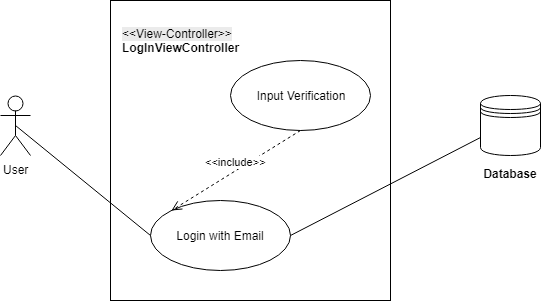
***Figure ⅱ.4:*** Select Image from Photo Library, Use Case Diagram

****

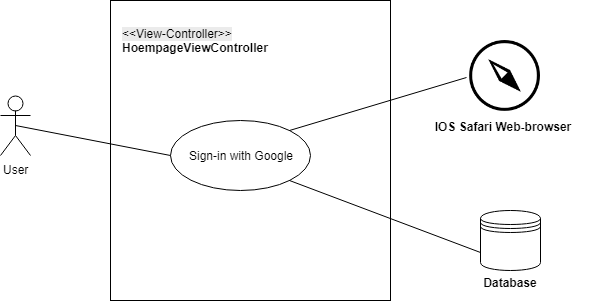
***Figure ⅱ.5:*** Take Photo Through IOS Camera Application, Use Case Diagram

****

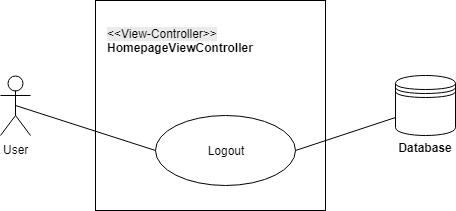
***Figure ⅱ.11:*** Signup with Email, Use Case Diagram

****

***Figure ⅱ.12:*** Login with Email, Use Case Diagram

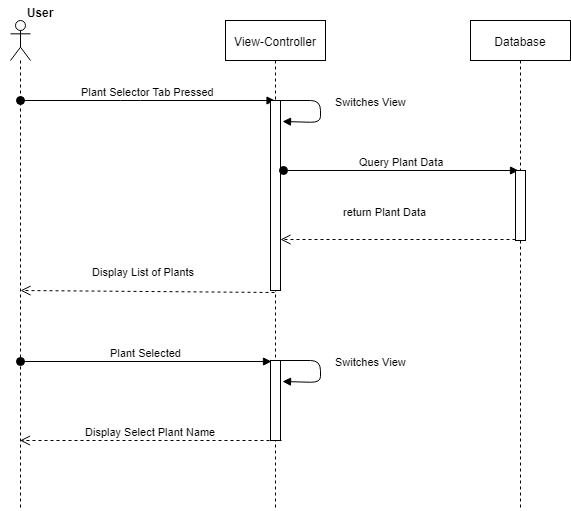
****

***Figure ⅱ.13:*** Sign-in with Google, Use Case Diagram

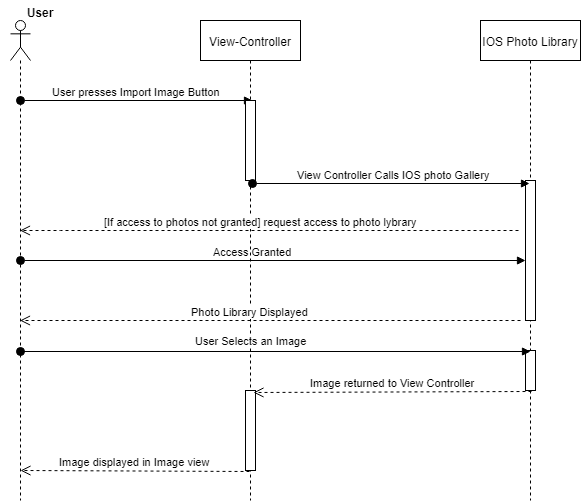
****

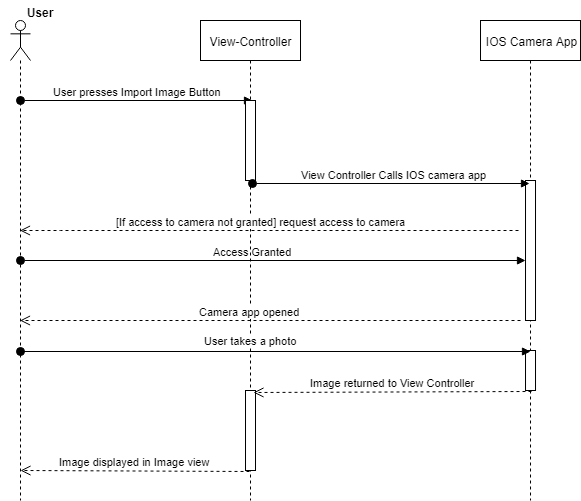
***Figure ⅱ.14:*** Logout, Use Case Diagram

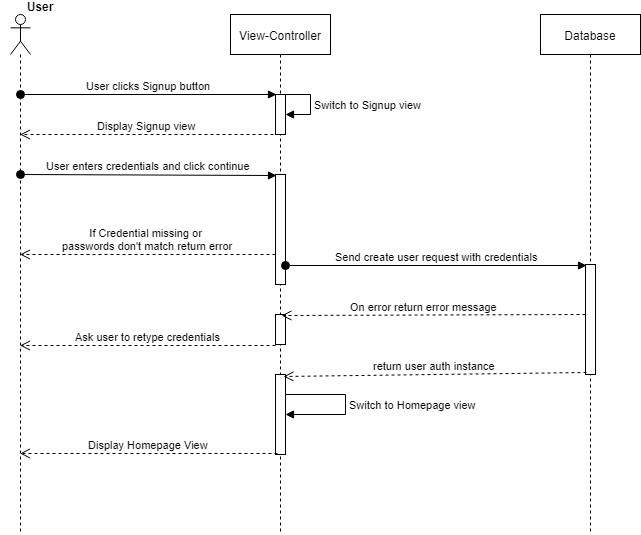
## **Sequence Diagrams**

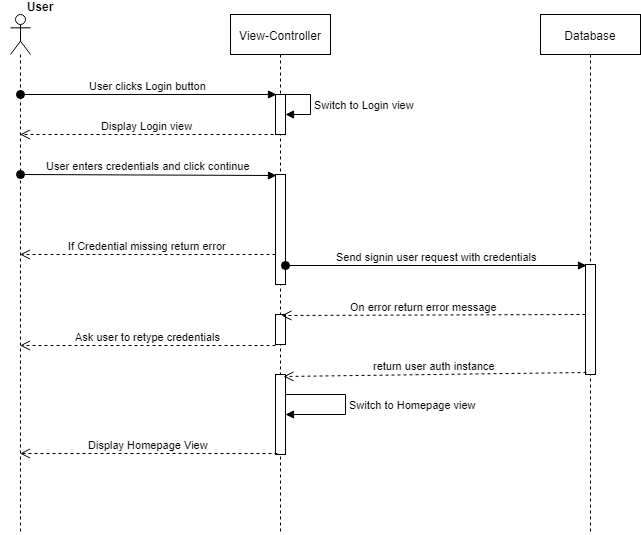


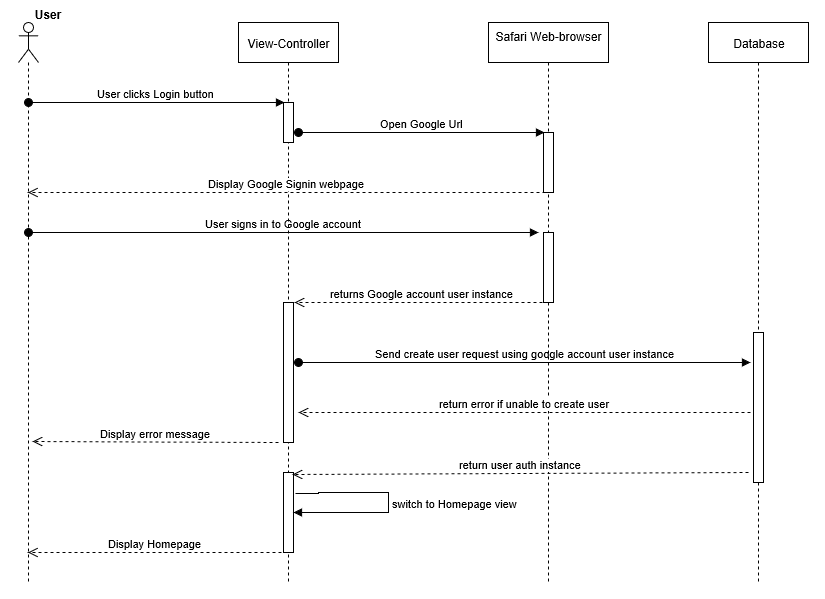
***Figure ⅱ.2:*** Plant Selected, Sequence Diagram

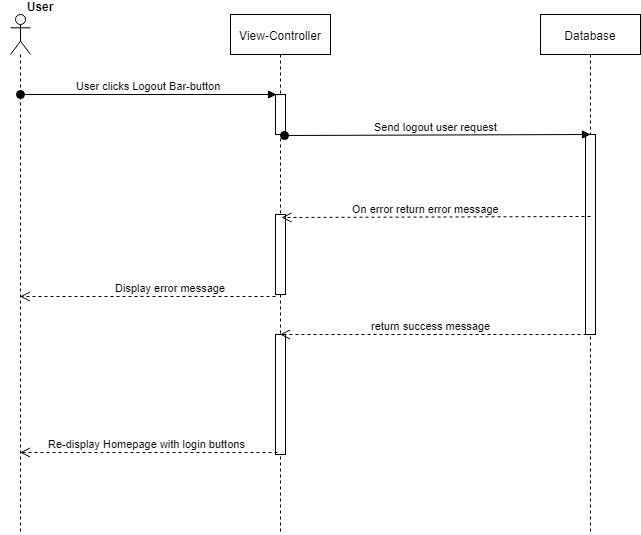
***Figure ⅱ.6:*** Select Image from Photo Library, Sequence Diagram

***Figure ⅱ.7:*** Take Photo Through IOS Camera Application, Sequence Diagram

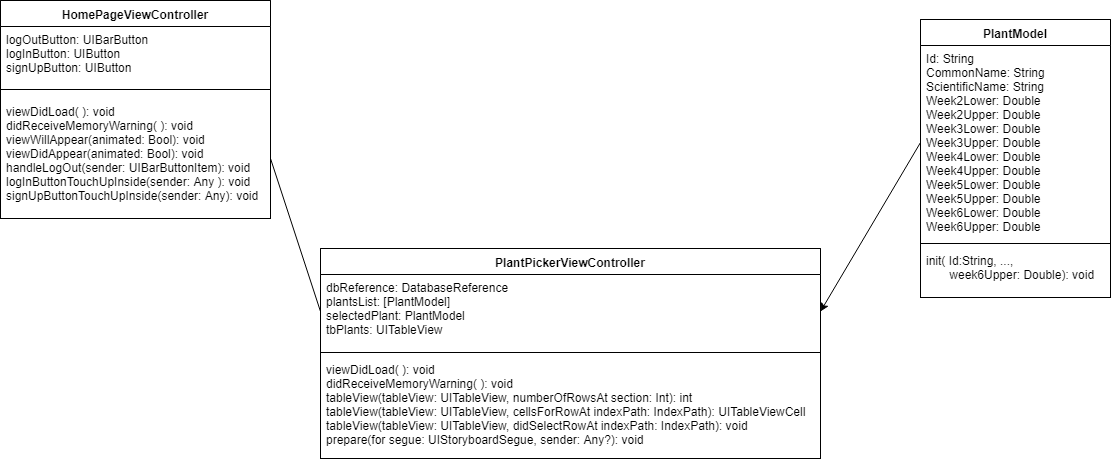
 ***Figure ⅱ.15:*** Signup with Email, Sequence Diagram

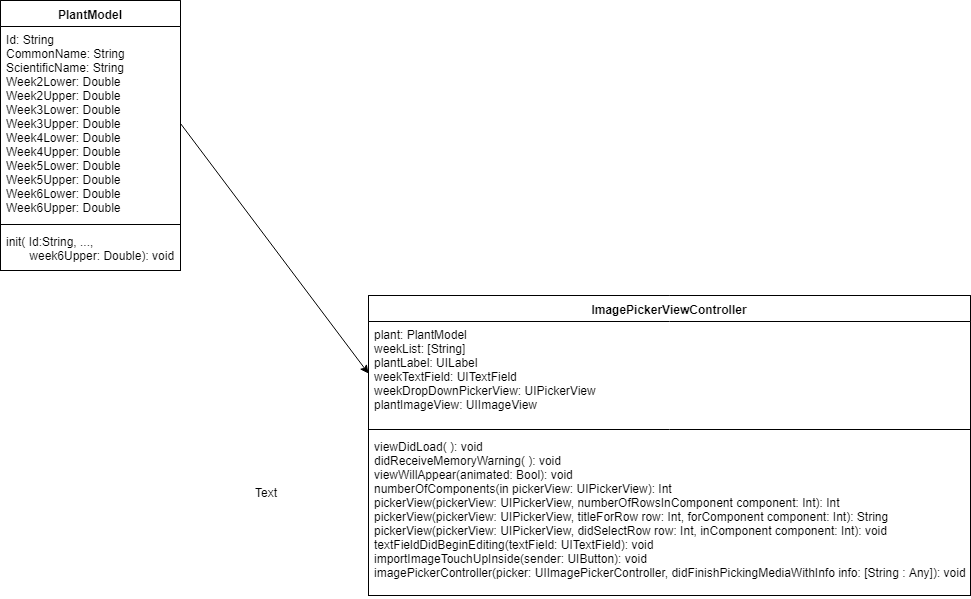
***Figure ⅱ.16:*** Login with Email, Sequence Diagram

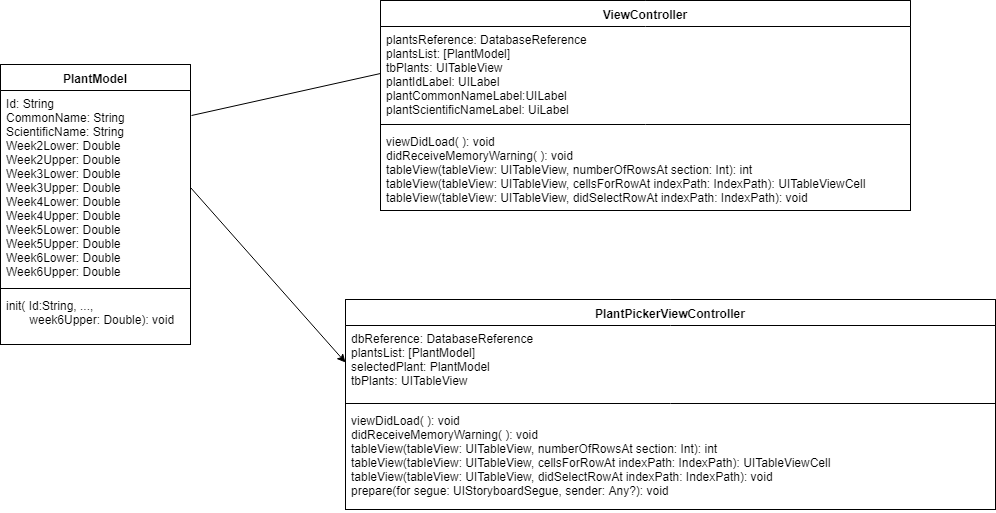
***Figure ⅱ.17:*** Sign-in with Google, Sequence Diagram

***Figure ⅱ.18:*** Logout, Sequence Diagram

## **Class Diagrams**

***Figure ⅱ.3:*** User Story #668, Class Diagram

***Figure ⅱ.9:*** User Story #671, Class Diagram

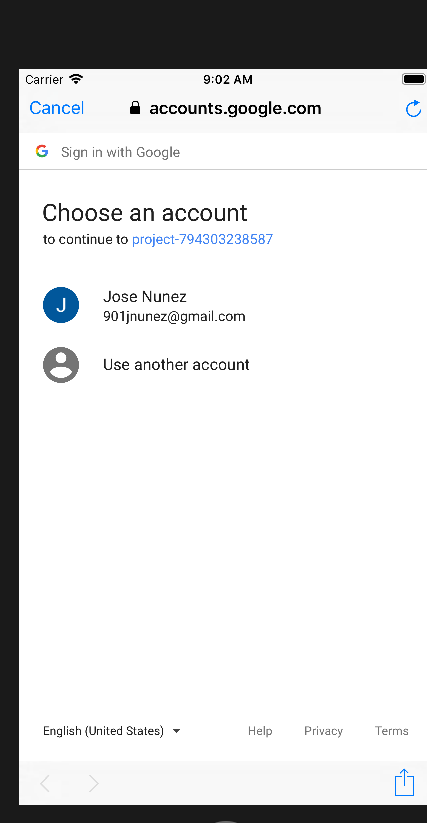
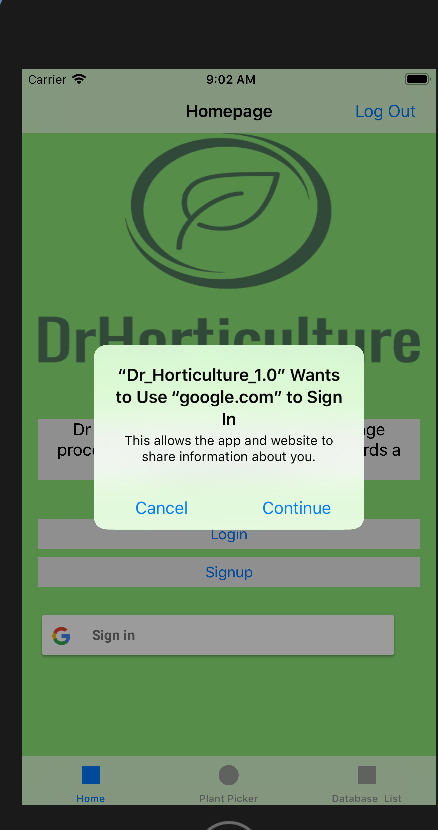
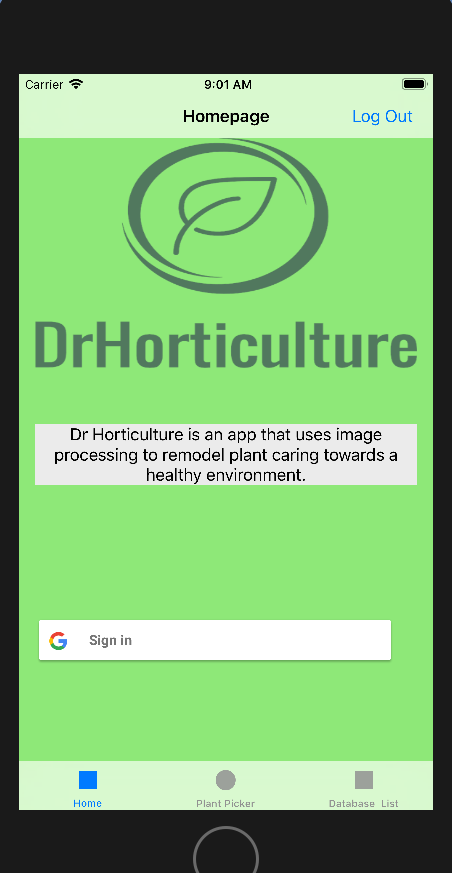
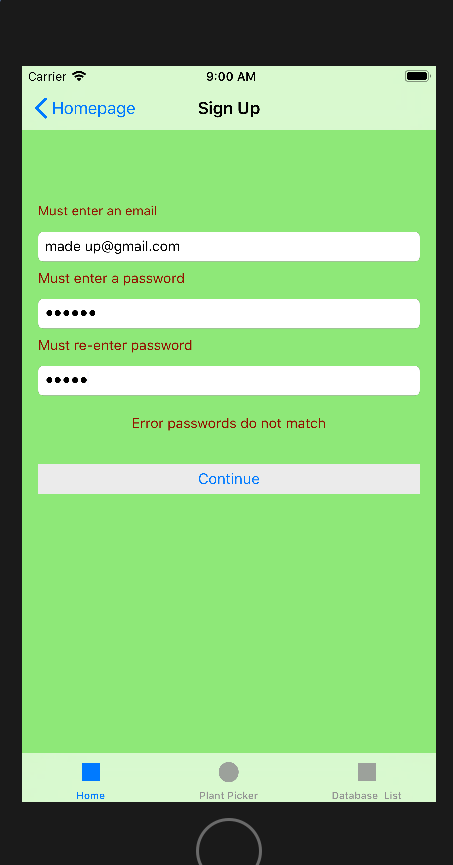
***Figure ⅱ.10:*** User Story #672, Class Diagram

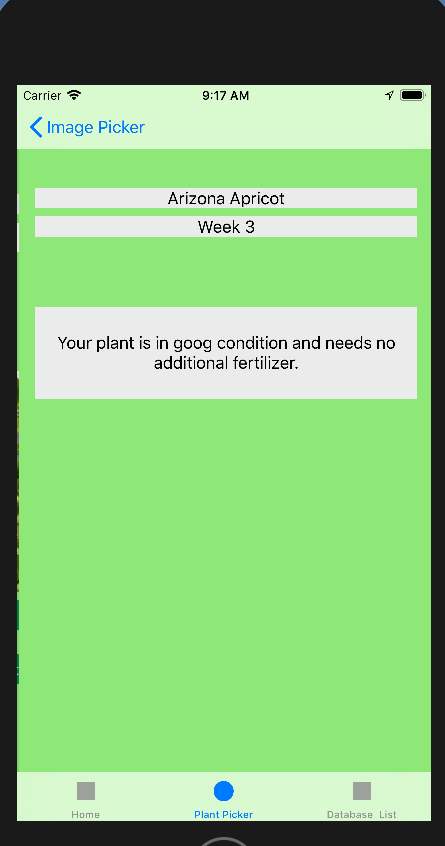
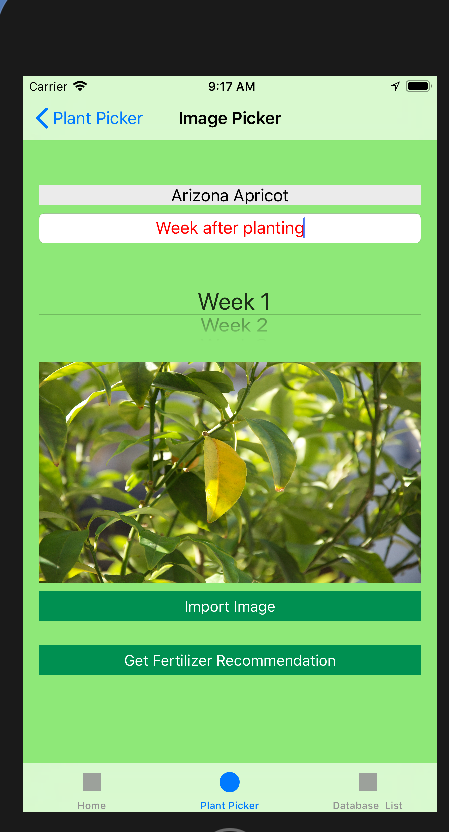
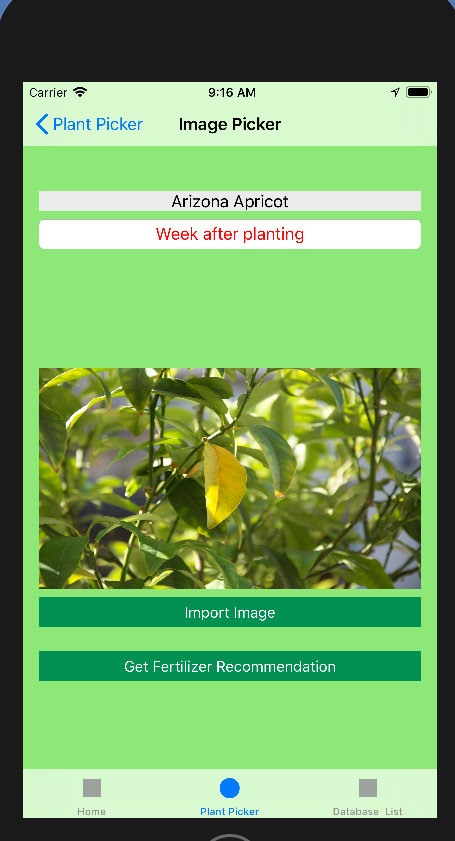
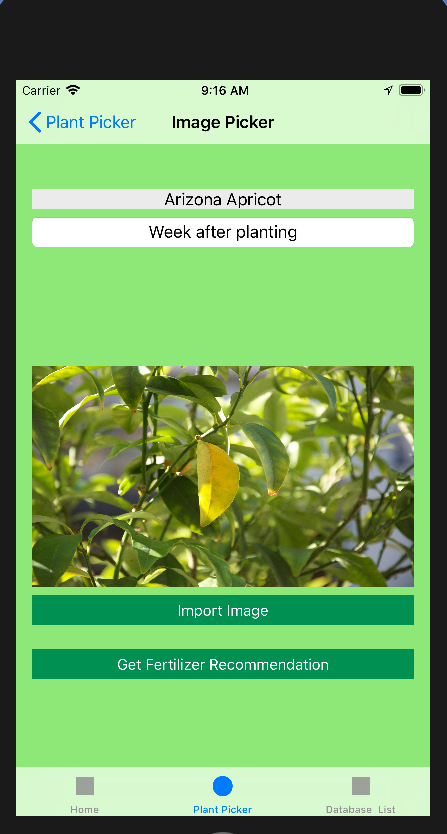
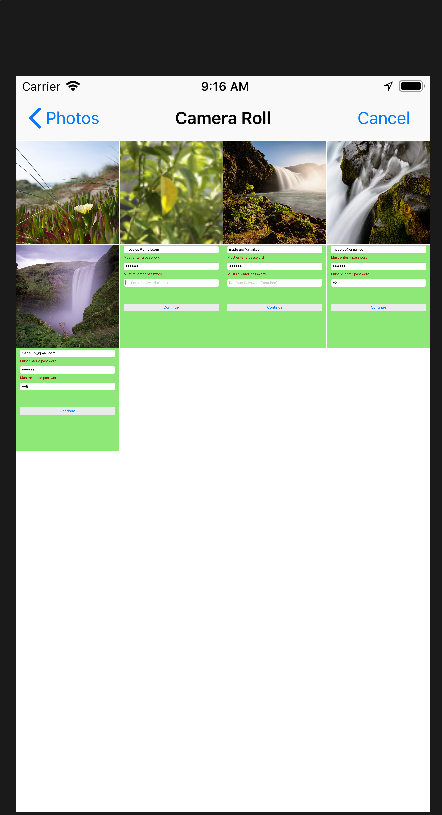
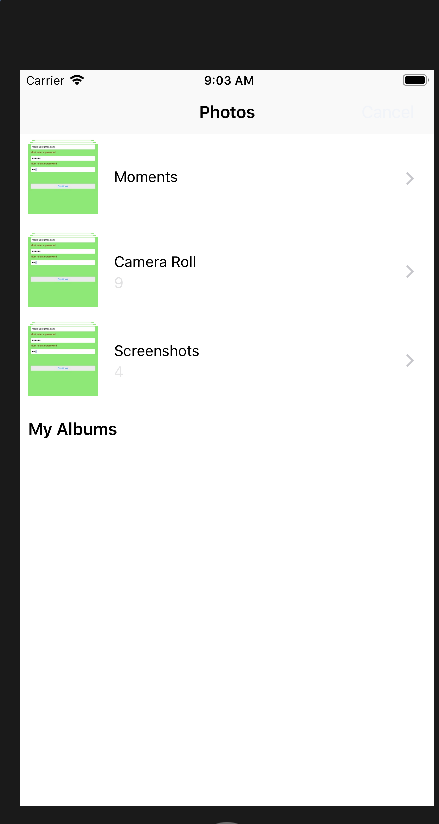
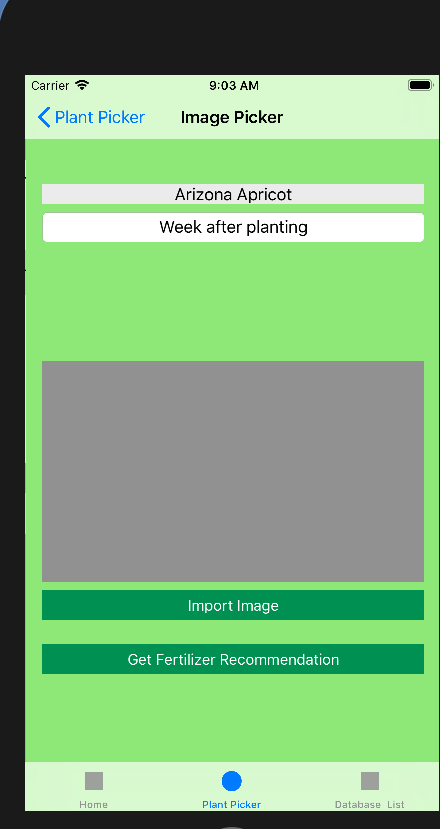
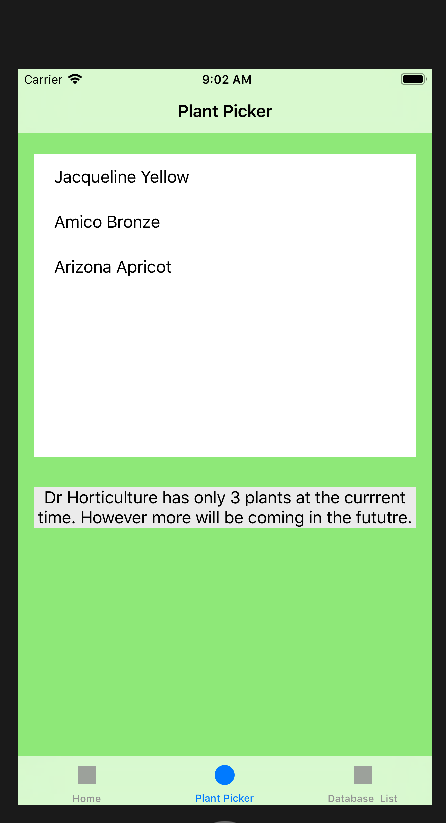
***Figure ⅱ.19:*** User Story #672, Class Diagram

## Appendix B - User Interface Design

## 

## 





## Appendix C - Sprint Review Reports

**Sprint 2 Review** **Meeting Minutes:**

Date: <02/15/18>

Attendees: <Dr. K, Jose Nunez, Cesar >

Start time: <10:00am>

End time: <10:45am>

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* #667, Create Homepage
* #668, Create Plant Selection View
* #669, Setup Server

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* No user story was rejected in this sprint meeting

**Sprint 2 Retrospective Meeting Minutes:**

Date: <02/02/18>

Attendees: <Jose , Cesar>

Start time: <7pm>

End time: <7:30pm>

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Our estimation were under estimations for our team’s velocity
* Did we do a good job estimating the points (time required) for each user story?
  + We were not able to estimate the points accurately
* Did each team member work as scheduled?
  + Yes

What went right?

* We were able to be on time and meet with our product owner
* Am learning Xcode and swift at a good pace

How to address the issues in the next sprint?

* How to improve the process?
  + Being more proactive for documentations and meetings
* How to improve the product?
  + Getting more knowledge in order to implement the user stories.

**Sprint 3 Review** **Meeting Minutes:**

Date: <02/23/18>

Attendees: <Dr. K, Jose Nunez, Cesar >

Start time: <1:00pm>

End time: <1:45pm>

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* #670, Create Tab-Navigation & Navigation Controller
* #668, Add Camera Usage and Image Upload

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* #669, Set Up server
* How this should be reflected on the user story definition in Mingle:

**Sprint 3 Retrospective Meeting Minutes:**

Date: <02/23/18>

Attendees: <Jose , Cesar>

Start time: <7pm>

End time: <7:30pm>

What went wrong?

* Did we do a good job estimating our team's velocity?
  + Our estimation were over estimations for our team’s velocity
* Did we do a good job estimating the points (time required) for each user story?
  + We were not able to estimate the points accurately
* Did each team member work as scheduled?
  + Yes

What went right?

* We were able to be on time and meet with our product owner
* We are learning how to manage the project better

How to address the issues in the next sprint?

* How to improve the process?
  + Being more proactive for documentations and meetings
  + Scheduling meeting times to better fit the the course requirements
* How to improve the product?
  + Getting more knowledge in order to implement the user stories.

**Sprint 4 Review** **Meeting Minutes:**

Date: <03/09/18>

Attendees: <Dr. K, Jose Nunez>

Start time: <12:00pm>

End time: <1:00pm>

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* #672-Integrate Firebase Database

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* #673-Update the App Look and Layout
* How this should be reflected on the user story definition in Mingle:
  + The applications still needs further updating to gain the feel as

desired

**Sprint 4 Retrospective Meeting Minutes:**

Date: <03/10/18>

Attendees: <Jose , Cesar>

Start time: <7pm>

End time: <7:30pm>

What went wrong?

* Did we do a good job estimating our team's velocity?
  + We still need to improve on estimating our team’s velocity
* Did we do a good job estimating the points (time required) for each user story?
  + Point estimations were on the lower side due to complications encountered with finding a mac to migrate the front end of the project
* Did each team member work as scheduled?
  + Yes

What went right?

* We were able to schedule meetings that better fit the requirements of the course
* The database was able to be integrated into the system

How to address the issues in the next sprint?

* How to improve the process?
  + Documenting more as the process unfolds rather than after
* How to improve the product?
  + Adding authentication through ability to login/signup

**Sprint 5 Review** **Meeting Minutes:**

Date: <04/05/18>

Attendees: <Dr. K, Jose Nunez, Cesar reyes >

Start time: <3:30pm>

End time: <4:15pm>

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* #674-Implement User Authentication (Login/Signup)

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* N/A

**Sprint 5 Retrospective Meeting Minutes:**

Date: <04/05/18>

Attendees: <Jose , Cesar>

Start time: <7pm>

End time: <7:30pm>

What went wrong?

* Did we do a good job estimating our team's velocity?
  + We are still a little bit off on estimating our team’s velocity
* Did we do a good job estimating the points (time required) for each user story?
  + Point estimations were on the lower side due to complications encountered with authentication
* Did each team member work as scheduled?
  + Yes

What went right?

* User authentication was properly implemented
* Google sign in was proper implemented

How to address the issues in the next sprint?

* How to improve the process?
  + Documenting more as the process unfolds rather than after
* How to improve the product?
  + Improve the appearance of the app to a more aesthetically pleasing look
  + Add in the recommendations view

**Sprint 6 Review** **Meeting Minutes:**

Date: 04/13/18

Attendees: <Dr. K, Jose Nunez, Cesar reyes >

Start time: <5:00pm>

End time: <5:45pm>

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* #677-Create Fertilizer Recommendation View

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* N/A

**Sprint 6 Retrospective Meeting Minutes:**

Date: 04/13/18

Attendees: <Jose , Cesar>

Start time: <7pm>

End time: <7:30pm>

What went wrong?

* Did we do a good job estimating our team's velocity?
  + We are still a little bit off on estimating our team’s velocity
* Did we do a good job estimating the points (time required) for each user story?
  + Point estimations were only a tad bit off
* Did each team member work as scheduled?
  + Yes

What went right?

* Final view properly added
* Documentation being done properly

How to address the issues in the next sprint?

* How to improve the process?
  + Continue to work on documentation
* How to improve the product?
  + Get the server and app to communicate

## Appendix D - Manuals

## Installation/Maintenance Document

Dr Horticulture v1.0 Used the following programs and thus require the following to be installed:

1. Xcode v9.2 & Swift 4

Use the following link to find apples IDE Xcode and install version 9.2 (Note: Xcode 9.2 already comes with Swift 4 so no further instalment is needed with regards to swift)

<https://developer.apple.com/download/more/>

1. Firebase

Firebase can be accessed from the firebase console. You can reach the firebase console by searching for it on google or following the link provided below. To have access to firebase you will have need login with your google email or corresponding email with which you have an account as well as have been granted access to the project with a given role. This process is explained in the “InstallMaintenanceGuide” video.

1. Cocoapods

In order for firebase portion of the app to run on xcode, cocoapods has to be installed. You can check if cocoapods is installed by running the command “pod --version”. It will return the version of cocoapods if it is installed or command not found otherwise. If not already installed, then run the command “sudo gem install cocoapods”.

Now that cocoapods is installed it is necessary to install the apps pod files. To do this navigate within terminal to where the app files are located. In the directory with podfile in it, simply run “pod install” and the pods mentioned within podfile will be installed. If future pods are to be installed simply add in their pod names under

“# Pods for Dr\_Horticulture\_1.0”

And then rerun “pod install” from terminal.

1. Python v2.7 & Visual Studios Community 2017

The server portion of the app was developed on python through the use of visual studios community. As such the links to both the download of visual studios community 2017 as well as the guide for installing python within visual studios.

<https://www.visualstudio.com/vs/features/python/>

<https://docs.microsoft.com/en-us/visualstudio/python/installing-python-support-in-visual-studio#visual-studio-2017>

# Ⅷ-References

Youtube Playlist For Videos:

https://www.youtube.com/playlist?list=PLG3lylzOg8Fho0qglNgpQy0e0BkbYIObX