Bilkent University

Department of Computer Science

Senior Design Project

GrapeHealth

High-Level Design 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

Innovation Expert: Kerem Erikçi (tarla.io)

High-Level Design Report

Dec 31, 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 Contents

List of Figures

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 overview of the architecture and design of the system. First of all, the purpose of the system and the design goals of our project are described. The subsystem decomposition, architectural plans of subsystems, and hardware/software mapping of these components are described. Persistent data management, access control and security, boundary conditions are reported. Finally, the functions of subsystem services and their interactions are outlined.

1.1. Purpose of the system

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 innovative part of GrapeHealth is integrating data, image and agronomy to diagnose grape diseases. The main purpose of the

application is recognizing the disease of the grape plants and informing the users about their grape condition by either making them to take photos of the leaf or using our chat-bot feature or both. By detecting diseases, the target audience will be able to notice the disease after a wider effect might occur and take precautions according to the feedback from our application. Grape is an essential part of our nutrition which has an important place in Turkey's market share and providing an information service to the providers or relevant people will increase healthy grape stock.

1.2. Design Goals

1.2.1. Usability

The user interface should be user-friendly to allow the users to understand how to be integrated with the application to obtain correct results. The application should not require any previous knowledge, it should be easy to use. While user is uploading a photo to the application if the photo is not in the format the application waits, the user will be informed about the correct format. Using buttons for chatbot make this part of program easier to you regard to allow user to enter a word/sentence. 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 [1][2].

1.2.2. 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. Generating reliable results according to user input and performing accurate feedback to the user will be considered.

1.2.3. Efficiency

The process of predicting the disease should not be longer than 60 seconds with reliable service. Transferring data to the server and retrieving the results from the server will be considered as an efficiency concern.

1.2.4. Extensibility

GrapeHealth will be extensible considering many factors but for now, it will detect diseases on grapes, however, we can increase the range of diseased plants determined by application. The application can be adapted to other fruit or vegetables types after GrapeHealth implementation succession. Our application has the potential to become a market factor, considering the value of the data provided from the users, the application can have a feature where we can predict the market value of each plant based nutritions. The application will be extendible considering these future directions.

1.2.5. 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.

1.2.6. Security

The application will ensure security of user's data & privacy by protecting the data management and storage. 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.

1.3. Definitions, Acronyms, and Abbreviations

API: Application Programming Interface
FTP: File Transfer Protocol
HTTP: Hypertext Transfer Protocol
MVC: Model-View-Controller
TCP: Transmission Control Protocol
UI: User Interface
SQL: Structured Query Language

1.4. Overview

GrapeHealth will be an (Android and iOS) application that enables the target audience to obtain information about grape plant condition. 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 and then determine disease by using flowchart representation. User can choose either using one of the features or using them both to have a better result. After using these functionalities, the user will be able to see the possible diseases with their potential percentages. We will use some image dataset of both healthy and diseased grape plants to train our application so that the application could recognize the diseases and can apart healthy and diseased grape plants. The user will also have other functionalities with the application.

Briefly, GrapeHealth will inform the users about the probabilistic analysis of diseases that the grape plant has. The application will prompt user with a plotting of the potential diseases with possibilities considering the query inputs they have given. The users also will be notified if there's a bad weather condition upcoming or contagious disease risk by other nearby users which might damage the plants. Using nearby feature, the user will be able to see disease information about other users' grape plants only within a fixed range.

2. Current Software Architecture

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. PlantNet is an example of them it does not diagnose diseases and none of these applications warn 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 Software Architecture

3.1. Overview

In GrapeHealth, we needed a software architecture design so that we could we could achieve our design goals. GrapeHealth consists of two main parts: Mobile Application and Server. We decided to use Model-View-Controller (MVC) software architecture design for

mobile application part so that our application is going to be reusable, easy to update and safe. We created subsystems so that implementation part would be easier for a team.

In MVC software architecture, Model parts are also entities for us to keep information in database. Queries, chatbots, results and user data compose GrapeHealth mobile application's Model part. On the other hand, user controller, query controller and chatbot controller forms Controller part. It fetches data from view part and send them to model part so that necessary modifications could be done.

Mobile application needs to communication with server to send queries and fetch user data and results of queries. Thus, we are going to use a WebSocket to ensure communication between server and mobile application.

3.2. Subsystem Decomposition

Our system can be decomposed into four main subsystems; Client, Server, Database and WebSocket. We preferred Client/Server architectural style, since data in database is private and should not be seen by third party.

In Client side, we use MVC software architecture. The View component will be used for all UI logic of the mobile application. User directly interact with UI and see necessary changes on screen.

In Server side, Communication between mobile application and server will be provided by WebSocket. Network Controller manages location suggestions by using Database and send some suggestions to the users in Client side through WebSocket. Suggestions are based on location of the user. Firebase Realtime Database aimed to use for storing relevant data. Also, Network Controller manages query results; Network Controller take input data(photo) and give it to the Learning Model in order to take prediction which represents query result.

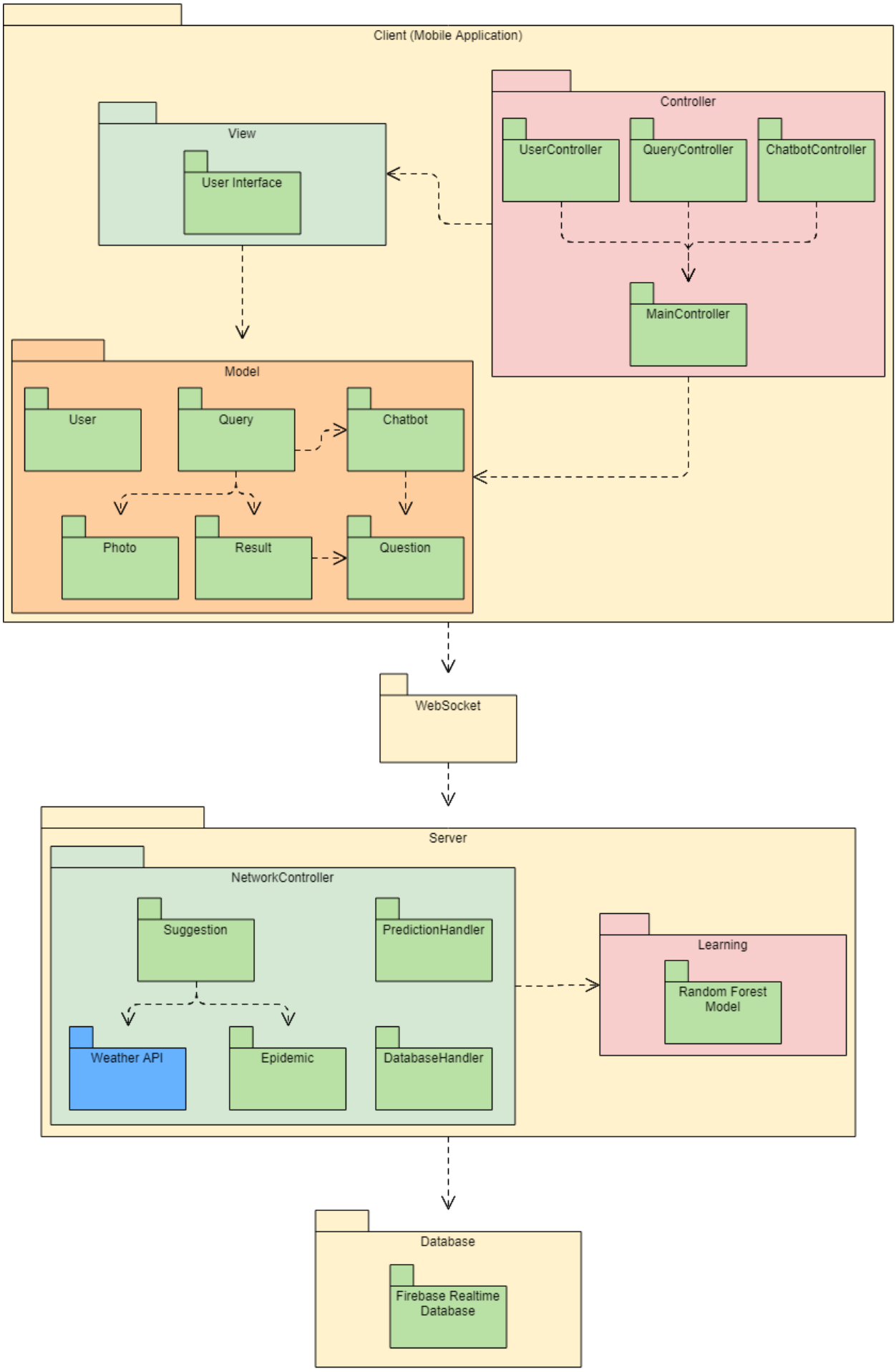

Figure 1: System Decomposition

3.3. Hardware/Software Mapping

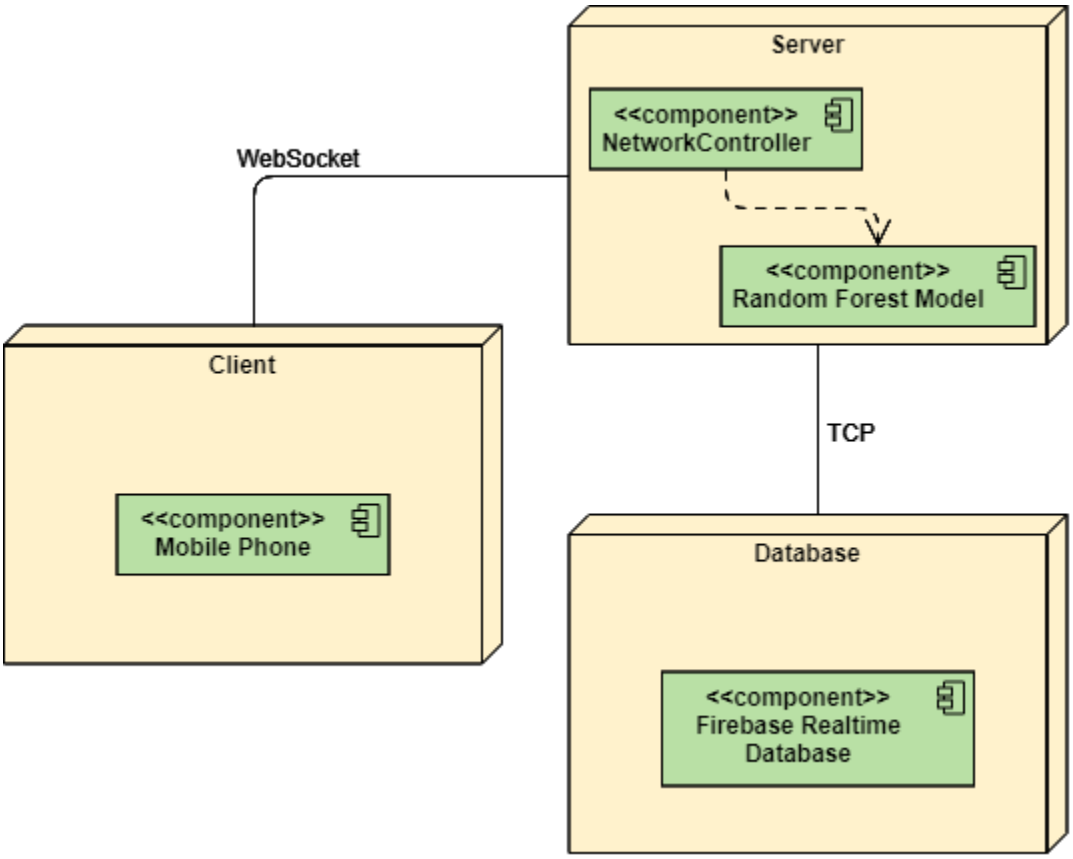

Figure 2: Hardware/Software Mapping

GrapeHealth will be run on both iOS and Android Phones. GrapeHealth will mainly uses the mobile phones; camera, GPS, connectivity hardware and memory to detect the diseases of the grapes. On the mobile phones, GrapeHealth will take the photo from users and send it to the server. Random Forest Algorithm will be applied on the server side. Therefore, Random Forest Model will not use the user's mobile phone to train and test. However, to able to take a picture and upload it to the program, mobile phone must have enough storage. React-native-camera library enables the program to use mobile phone's camera. Communication between server and client will be provided via WebSocket. WebSocket is a communications protocol, bi-directional, full duplex TCP connection from a user's web browser to a server [4]. WebSocket uses the handshake request from a browser's HTTP connection and handshake request includes a 64-bit Sec-WebSocket-Key header [4]. Database interaction will be established via Google Firebase Realtime Database. Firebase is management tool for android developers. Creation of databases will be done on Firebase.

3.4. Persistent Data Management

The GrapeHealth application need to have a database to store queries (created by user to learn disease of specified grape by uploading photo of grape's leaf or answering questions in chatbot) and user identities. The quickness is very important for an application and user need swift access to his or her old queries with its multiple instances like result, photo, flowchart of chatbot answered by specified user and treatment. Therefore, we will use Firebase Realtime Database to store and interact with relevant data quickly. Data is stored as JSON and concurred in real-time to every connected client in Firebase Realtime Database, and it save a tremendous amount of custom development time.

3.5. Access Control and Security

Access control is a security technique which regulates who or what can view or use resources in a computing environment [5]. GrapeHealth has one type user, it does not have admin users. Therefore, this part will explain the security issues related to GrapeHealth application. In this part, authentication mechanism, SQL Injection and encryption will be discussed.

By using application, users generally provide their private information. If user's data stored in an unsecured manner, it creates the potential risk of the data leak. For the iOS part of the GrapeHealth in order to protect user's data, iOS provide Keychain. Keychain is the system which enables to program store logins, passwords, keys inside of it. It is password management system which developed by Apple and distributed iOS [6]. Therefore, in the GrapeHealth the user's keys, passwords will be stored in Keychain systems. As the majority of the mobile applications run on HTML5 technology, this created potential risk of the SQL injection. To prevent SQL injection, respected SQL commands will be written in safer manner. That is, SQL command will not allow an attacker to inject code that can be executed on our database systems.

In the android part, android security model will be used. Android Security model provides, Robust Security at the OS level through the Linux Kernel, Mandatory application sandbox for all applications, secure inter process communication, Application signing,

Application-defined and user-granted permissions [7]. To prevent SQL injection, SQL commands will be written in a safer way.

For encryption part, React Native crypto-js library will be used for both iOS and Android. As the GrapeHealth will be written on React Native, encryption will be done on React Native. After React Native parts completed, specified features above will be applied for both iOS and android.

3.6. Global Software Control

For GrapeHealth application event-driven control mechanism will be used. In our project, the system is designed to respond to user actions within the system. Each event that user performs corresponds to a request creation and the requests runs code segments in the application. After event occurs, the server responds to that request and waits for another request.

3.7. Boundary Conditions

3.7.1. Initialization

In order to use the GrapeHealth application, depending on their mobile devices operating system (iOS or Android), user must download the application from App Store or Google Play Application store. After the installation is finished the user can open the app. Greetings page will display sign-in and sign-up options. If the user does not have an account, by selecting sign-up the user will be accepted only by filling the necessary information. After user creates an account or already has with email and password information the user can sign-in. Sign-in preferences are by directly entering login information or via Google account or Facebook account.

User needs to login to the system in order to use the functionalities of the application. After login the user will be able to upload photos of their grape plant leaves or use chat-bot feature. By using the feature, the system will generate a query by considering user input. Interacting with the application requires internet connection otherwise user will not be able to use the features. User also needs to give access to viewing camera roll and GPS to use the

application for full potential. Therefore, the application will prompt the user to obtain access to enable using the feature and user needs to have internet connection.

3.7.2. Termination

User can logout from the system by simply tapping on the logout button on the side bar menu. After the user terminates from the system, to access his/her account user must login again. When user decides to close the application without logout, user will still be logged in to the system in when user decides to launch the application again.

3.7.3. Failure

Failure of the application can occur in four cases.

First case can be while uploading the photo. If the user uploads an image not related with the expectation, the system will warn the user to upload with correct format and the feature will fail with warning. While the user is taking the photo, the application will validate the photo.

Second case of the failure will be seen when the user closes the internet connection within the application. If the user closes internet connection while uploading photos to the system and is waiting for a response from the system, the system will fail with warning. Without internet connection user will not be able to use the features.

Third case of failure is seen if one of the functioning modules stops to function due to a bug in code or other problems in the server subsystems. In this case, the application will inform the users that the feature is not available. After fixing the problem the users will be notified.

The fourth case of failure is seen if the user closes the app during processing photo or calculating the predictions, the system will fail and the state of the system will not be saved in this case.

4. Subsystem Services

4.1. Client Subsystem

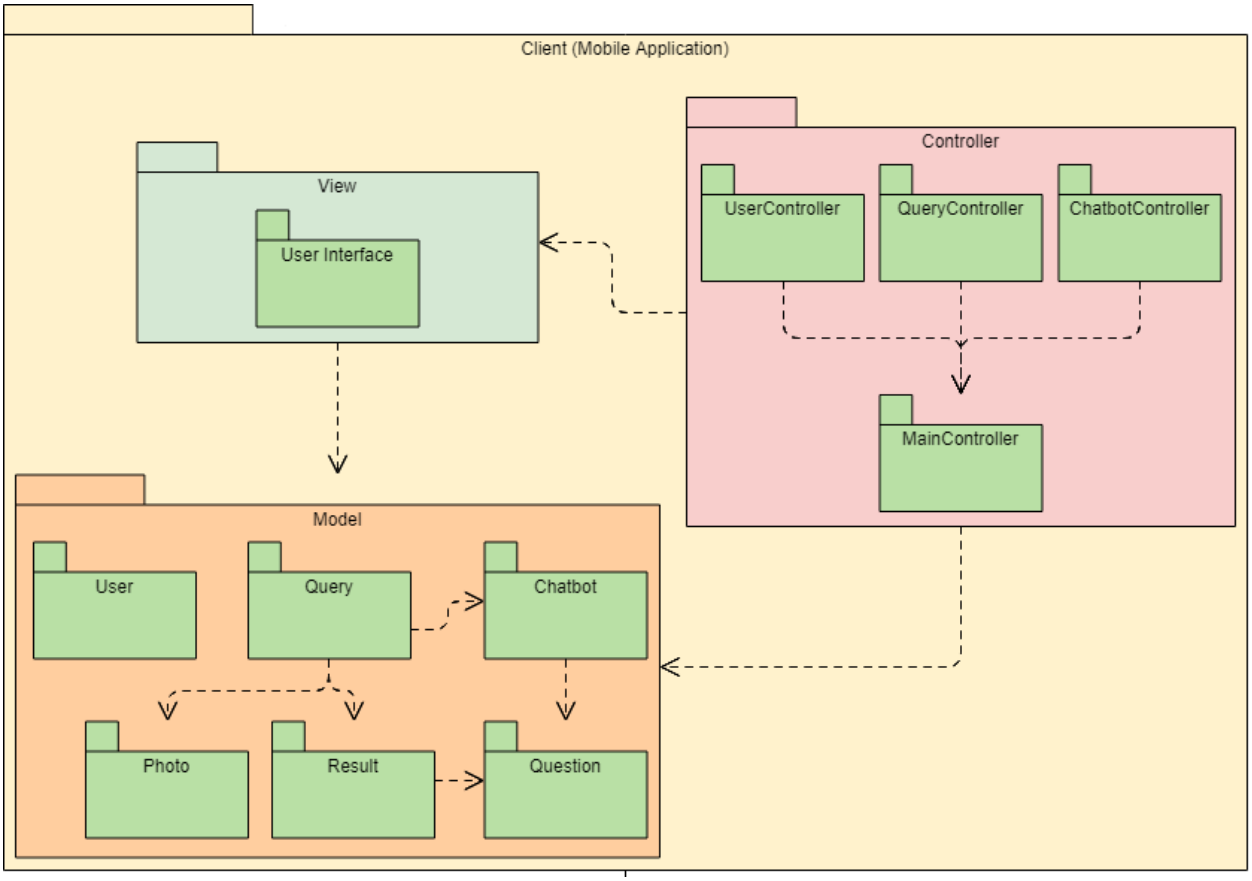

Figure 3: Client Subsystem

4.1.1. View

The view component will be used for all UI logic of the mobile application. View is attached to its model and gets the necessary data to present. User directly interact with UI and see necessary changes on screen.

4.1.2. Controller

MainController: Main controller class is abstract parent class which control and manages operations in all other controller classes.

UserController: It is responsible controlling for user identity relate actions.

QueryController: Handles and manages operations about queries that represents inquiries created by user to learn disease of specified grape by uploading photo of its leaf or answering questions in chatbot.

ChatbotController: Handles and manages flowcharts while the user is using chatbot and answering questions to learn disease of specified grape.

4.1.3. Model

User: The system allows to the user to access program after creating his or her account.

Query: Query can be created by user to use chatbot by answering questions, or to use trained deep learning model by uploading photo of diseased grape leaf, or both of them.

Chatbot: Chatbot represent stored questions and answers in the map. According to the order of the questions, it highlights them in sequence of flowchart from start state to the final states (final state determine the disease).

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.

Question: Question is a class which represents the states in chatbot flowchart.

4.2. Server Subsystem

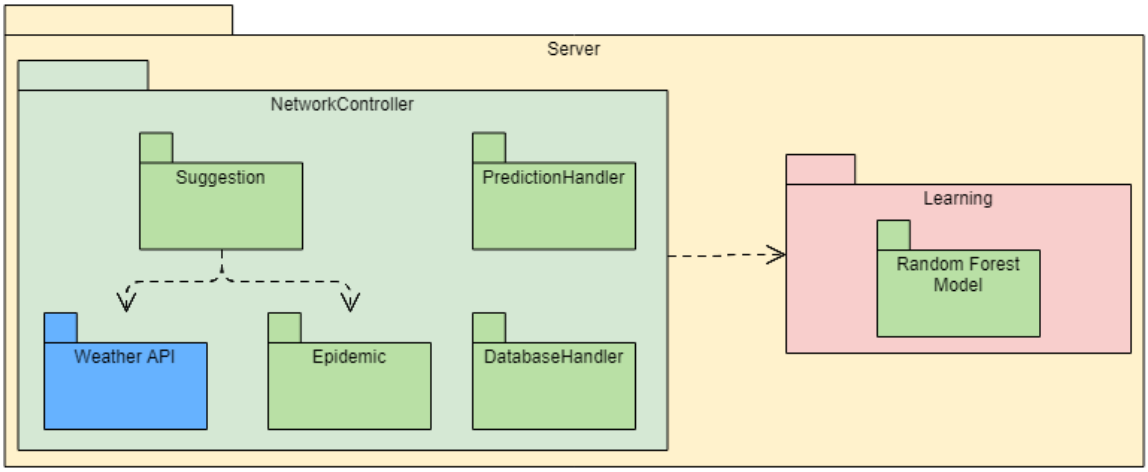

Figure 4: Server Subsystem

4.2.1. Learning Model

Trained random forest model take input (photo of grape leaf) through the WebSocket, and it analyze data and deliver prediction which is shown to the user through web application.

4.2.2. Network Controller

Network Controller manages query results and location suggestions between Client, Database and Learning Model. Suggestions are based on location of the user. It notifies user about epidemics or weather. When an epidemic occurred in a field, it notifies all users close to there. We are planning to use OpenWeatherMap API to get current weather and forecasts [8]. It shows weather conditions to user while suggesting it is good or bad time to spraying. Windy or rainy days are not suitable for spraying. PredictionHandler and DatabaseHandler manage requests coming from Client side and inform it back.

4.3. Database Subsystem

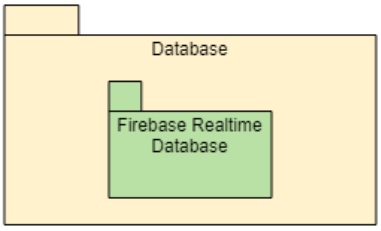

Figure 5: Database Subsystem

We decided to use Firebase Realtime Database to store relevant data. Database can be access only by Server. User's' profile information will be stored in database as well as all old queries. Queries will be stored in database in order to show user his/her passed queries as well as predict possible epidemics. If user gives permission to GPS, we use location information to detect epidemics in area. Also, photos taken by users can be used for increasing our dataset.

4.4. WebSocket

WebSocket provide communication between mobile application and server. Mobile application should send some data to server such that user login credentials, queries, chatbot answers and location whereas server should send results of queries and chatbot answers, profile information, notifications and treatments to mobile application. In order to achieve this communication, WebSocket will be used. It will be a bidirectional bridge.

Glossary

Client	The part of the system the users interact with.
Server	The part of the system responsible from logical operations, scheduling, and data management.
MVC	Model–view–controller is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts; Model, View, Controller.
React Native	React Native is a JavaScript framework for building native mobile apps. It uses the React framework and offers large amount of inbuilt components and APIs.
Keychain	Keychain is the password management system in macOS, developed by Apple.
PlantNet	PlantNet is an image sharing and retrieval application for the identification of plants.

References

[1] "iOS Distribution and iOS Market Share," *Apteligent | Mobile Application Performance and User Experience*. [Online]. Available: https://data.apteligent.com/ios/. [Accessed: 15-Oct-2018].

[2] "Distribution dashboard | Android Developers," *Android Developers*. [Online]. Available: https://developer.android.com/about/dashboards/. [Accessed: 15-Oct-2018].

[3] K. Erikçi, "Interview with Co-founder of ErikTronik," 10-Oct-2018.

[4] "What is WebSocket? - Definition from WhatIs.com," *WhatIs.com*. [Online]. Available: https://whatis.techtarget.com/definition/WebSocket. [Accessed: 30-Dec-2018].

[5] "What is access control? - Definition from WhatIs.com," *SearchSecurity*. [Online]. Available: https://searchsecurity.techtarget.com/definition/access-control. [Accessed: 30-Dec-2018].

[6] "iOS App Security – How to Protect Users' Sensitive Data in Mobile Applications," *Netguru Blog on Machine Learning*. [Online]. Available: https://www.netguru.co/blog/ios-app-security-how-to-protect-users-sensitive-data-in-mobile-applications. [Accessed: 30-Dec-2018].

[7] S. Srivatsa, *A Summary of Network Traffic Monitoring and Analysis Techniques*. [Online]. Available: https://www.cse.wustl.edu/~jain/cse571-14/ftp/android_security/index.html#sec2.1. [Accessed: 30-Dec-2018].

[8] OpenWeatherMap.org, "Weather API," *openweathermap*. [Online]. Available: https://openweathermap.org/api. [Accessed: 25-Dec-2018].