

T.C.
HASAN KALYONCU UNIVERSITY



ORCHARD MANAGEMENT

GRADUATION PROJECT REPORT

Eyüp Kazım Göymen

Mehmet TAŞ

Hüseyin Faruk GÖKTAŞ

Supervisor

Assoc. Prof. Dr. Muhammet Fatih HASOĞLU

HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

ORCHARD MANAGEMENT

GRADUATION PROJECT
IN
COMPUTER ENGINEERING

BY
Environmentalists
JAN 2020

JAN2020

Graduation Project in Computer Engineering Department

Environmentalists

Orchard Management

**Graduation Project
in
Computer Engineering
Hasan Kalyoncu University**

**Supervisor
Assoc. Prof. Dr. Muhammet Fatih HASOĞLU**

**By
Environmentalists**

JAN 2020

Copyright © 2020 Orchard Management And Tracking Application

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

REPUBLIC OF TURKEY
HASAN KALYONCU UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Name of the Project: Orchard Management And Tracking Application

Name of the student(s): Hüseyin Faruk Göktaş, Mehmet Taş, Eyüp Kazım Göymen

Exam Date: 01-13-2020

We certify that this project satisfies all the requirements as a project for the graduation project

Assist. Prof. Dr. Ulaş GÜLEÇ

Head of the Computer Engineering Department

This is to certify that we have read this project and that in our consensus/majority opinion it is fully adequate, in scope and quality, as a project for the graduation project.

Assoc. Prof. Dr. Muhammet Fatih HASOĞLU
Supervisor

Examining Committee Members

Signature

We hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. We also declare that, as required by these rules and conduct, we have fully cited and referenced all material and results that are not original to this work.

Hüseyin Faruk Göktaş

Mehmet Taş

Eyüp Kazım Göymen

ABSTRACT

ORCHARD MANAGEMENT AND TRACKING APPLICATION

Göktaş, Hüseyin Faruk

Taş, Mehmet

Göymen, Eyüp Kazım

Graduation Project in Computer Eng.

Supervisor: Assoc. Prof. Dr. Muhammet Fatih HASOĞLU

Jan 2020, XXXX pages

In this report, we are going to talk about plans, requirements, system design about desired application, orchard management application.

In orchard management application, there will be some core features that is going to be explained in Software System Requirements, see Chapter 3.

Briefly, orchard management application is an Android application that simplifies orchard management tracking for long term. Nowadays, it's complex to track entire orchard for long terms with old-fashion techniques. Our aim is to make it simpler and more usable in long term. Also, we are planning to implement some statistics over periods, accounting data such as incomes and expenses to visualize how the orchard has been grown up during time and total incomes/expenses considering what user needs to see. In this project, we are also planning to develop the application and application user interfaces for production environment so that our goal is to deploy our codes to remote server for application program interfaces and provide long-term data support for Android application. Also, in client side, we are going to develop the application as production development with proper architectures such as MVVM etc, explained in System Design, see Chapter 4.

In this project, there'll be some core features and some future works to develop better production environment with several versions that we track from GitHub. In core features, we are going to have some features such as defining orchard, creating harvest periods, adding expenses/incomes that we have to implement in order to release a beta version in our application. For limited time issues, we have defined some feature works that we are going to implement for features such as artificial intelligence based recommendations, ability to track all orchards considering privacy etc.

ÖZET

Göktaş, Hüseyin Faruk

Taş, Mehmet

Göymen, Eyüp Kazım

Bitirme Projesi Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Muhammet Fatih HASOĞLU

Ocak 2020, XXXX Sayfa

Bu raporda, planlar, gereksinimler, istenilen uygulama hakkında sistem tasarımı, meyve bahçesi yönetimi uygulaması hakkında konuşacağız.

Meyve bahçesi yönetimi uygulamasında, Yazılım Sistem Gereksinimleri'nde açıklanacak bazı temel özellikler olacaktır, bkz. Bölüm 3.

Kısaca, meyve bahçesi yönetimi uygulaması, uzun vadede meyve bahçesi yönetim takibini basitleştiren bir Android uygulamasıdır. Günümüzde, tüm meyve bahçelerini eski moda teknikleriyle uzun vadede izlemek karmaşık bir iştir. Amacımız, uzun vadede daha basit ve daha kullanışlı hale getirmektir. Ayrıca, zaman içinde bahçenin nasıl büyüdüğünü ve toplam geliri / gideri görselleştirmek için gelir ve giderler gibi muhasebe verilerini, kullanıcının neyi görmesi gerektiğini göz önünde bulundurarak bazı istatistikler uygulamayı planlıyoruz. Bu projede, üretim ortamı için uygulama ve uygulama kullanıcı arabirimlerini de geliştirmeyi planlıyoruz, böylece hedefimiz uygulama programı arabirimleri için kodlarımızı uzak sunucuya dağıtmak ve Android uygulaması için uzun vadeli veri desteği sağlamaktır. Ayrıca, müşteri tarafında, Sistem Tasarımında açıklanan MVVM vb. Gibi uygun mimarilerle üretim geliştirme olarak uygulamayı geliştireceğiz, bkz. Bölüm 4.

Bu projede, GitHub'dan izlediğimiz çeşitli sürümlerle daha iyi üretim ortamı geliştirmek için bazı temel özellikler ve gelecekteki bazı çalışmalar olacak. Temel özelliklerde, uygulamamızda bir beta sürümü yayınlamak için uygulamamızın tanımlanması, hasat dönemleri oluşturulması, uygulamamız gereken giderler / gelirler gibi bazı özelliklere sahip olacağız. Sınırlı zaman sorunları için, yapay zeka tabanlı öneriler, gizlilik göz önünde bulundurularak tüm meyve bahçelerini izleme yeteneği gibi özellikler için uygulayacağımız bazı özellik çalışmaları tanımladık.

ACKNOWLEDGEMENTS

We would like to thank our supervisor Prof.Dr. M.Fatih Hasoglu for his great effort on orchard management application with critic advices for better product development process with his deep knowledge about orchards.

We also thank to Asst.Prof. Bülent Haznedar and Asst.Prof. Saed Alqaraleh for their great effort for graduation project.

TABLE OF CONTENTS

1.PREFACE	15
2.INTRODUCTION	16
2.1 Purpose and Scope.....	16
2.2 Problem Statement	16
2.3 Solution Statement	16
2.4 Contribution	16
2.5 Glossary	16
3. LITERATURE REVIEW	17
3.1 What is the reason for the application?	17
3.2 Methods used before application?	17
3.3 What are the solutions?	17
3.4What are similar applications?	18
3.4.1 OrchardApp	18
3.4.2 Croptracker	18
3.4.3 Hectre.....	18
3.4.4 Cropdata.....	18
4. SOFTWARE REQUIREMENTS SPECIFICATION	18
4.1 Introduction	18
4.1.1 Purpose	18
4.1.2 Intended Audience And Reading Suggestions	19
4.1.3 Product Scope	19
4.1.4 References.....	19
4.2 The Overall Description.....	20
4.2.1 Product Perspective	20
4.3 Product Functions	21
4.3.1 Initial Application	22
4.3.2 Application Registration.....	22
4.3.3 Specifying a Location in the Application	22
4.3.4 Specifying Orchard Content	22
4.4 User Characteristics	23
4.5 Constraints	23
4.6 Assumptions And Dependencies.....	24
4.7 Apportioning of Requirements	24
4.7.1 Primary	24

4.7.2 Secondary.....	25
4.8 Specific Requirements.....	25
4.8.1 User Requirements	25
4.8.2 System Requirements	27
4.8.3 Non Functional Requirements	30
4.8.4 Logical Database Requirements.....	30
4.8.5 Design Constraints	30
4.8.6 Software System Attributes	31
4.8.7 Organizing the Specific Requirements	31
5. SYSTEM DESIGN SPECIFICATION	32
5.1 Introduction	32
5.1.1 User Interface Upon the First Install.....	32
5.1.2 User Interface with User Interaction and Data Presentation	39
5.2 System Architecture.....	43
5.2.1 System Software Architecture	43
Packages.....	46
Express, Nodemon, Passport, Jsonwebtoken, Babel, Body-parser.....	46
List of endpoints.....	48
POSTMAN.....	48
5.3 Database Design.....	48
5.4 Human-Machine Interface	48
5.4.1 Entire System	49
5.4.2 User Functionalities.....	49
.....	50
5.4.3 Instance Sequence Diagram.....	50
5.4.4 Classes	52
5.5 Detailed Design	52
5.5.1 Software Detailed Design.....	52
6. IMPLEMENTATION	57
6.1 Splash Screen	57
6.2 Onboarding	58
6.3 Sing Up	60
6.4 Forget Password.....	61
6.6 Init Orchard	62
6.6.1 Orchard Selection.....	64

6.6.2 Tree Add	65
6.7 Login Screens.....	66
6.8 Specifying Tree Parameters	67
6.9 Home Page	68
6.10 Update/ Delete Tree	70
6.11 Listing Orchards.....	71
6.12 Creating and Listing Harvest Periods	72
6.13 Creating and Listing Expense/ Income For Periods.....	74
6.14 Creating and Listing Expense / Income For Trees	75
6.15 Updating Harvest Period	76
6.16 Statistics	77
7.RESULT AND CONCLUSION.....	78
8. GLOSSARY	79
9. REFERENCES	80

LIST OF FIGURES

Figure 1: Mock-up of onboarding pages	33
Figure 2: Mock-up of auth pages	34
Figure 3: Selection of Orchard land on map	35
Figure 4: UI for adding the trees to the field	36
Figure 5: Defining attributes for trees	37
Figure 6: General application flow	38
Figure 7: UI for the list of orchards	39
Figure 8: Harvest periods view.....	40
Figure 9: Tree details secreenshots	41
Figure 10: Statistics	42
Figure 11:Structure of MVVM	45
Figure 12:Structure of MVVM	45
Figure 13: Router structure for API.....	47
Figure 14: Database Structure	48
Figure 15: Use case diagram	49
Figure 16:Use case diagram	50
Figure 17: Sequence diagram that shows login state	51
Figure 18: Class UML.....	52
Figure 19: Thread structure	53
Figure 20: Networking structure	54
Figure 21:Networking structure.....	54
Figure 22: MVVM design pattern the application.....	55
Figure 23:Test structure	56
Figure 24:Splash Screen For App loading.....	57
Figure 25:Onboarding Screen-1	58
Figure 26: Onboarding Screen-2	58
Figure 27: Onboarding Screen-3	59
Figure 28: Onboarding Screen-4	59
Figure 29:Register Page.....	60
Figure 30: Forget Password First Stage.....	61
Figure 31: Forget Password Second Stage	61
Figure 32:Orchard Initiliaz Page	62
Figure 33:Location Permission View	63
Figure 34:Location Zoom in.....	64
Figure 35:Orchard Location Specifacation	64
Figure 36:Adding Tree on GoogleMaps	65
Figure 37:Adding Tree Completion	65
Figure 38:Loader View	66
Figure 39:Updating Tree Parameter Pop-up.....	67
Figure 40:Home Page	68
Figure 41:Adding New Tree.....	69
Figure 42:Updating Tree Parameters.....	69
Figure 43:Update or Delete Tree Parameters.....	70
Figure 44: Listing Orchard Map.....	71
Figure 45:Listing Orchards List	71
Figure 46:Create New Period	72

Figure 47:Listing Harvest Periods.....	73
Figure 48:Creating Listing Expense	74
Figure 49:Listing Expense Period	74
Figure 50:Creating Expense For Trees.....	75
Figure 51:Listing Expense For Trees.....	76
Figure 52:Updating Harvest Period.....	76
Figure 53:Statistics	77
Figure 54: Statistics-2	77

1.PREFACE

The presented report is the resulting work of the Graduation Project of Hasan Kalyoncu University Computer Engineering Department. Orchard Management and Tracking Application is the completed project which aims orchard owners to have a better understanding of the harvest period that they work on and give the ability of all the effort spent throughout this period. This project and the idea behind truly belongs to Assoc. Prof. Dr. Muhammet Fatih HASOGLU. The motivation behind the selection of this project is complete matches with his great interests in the problems and the approaches that he proposes a solution. These statements truly motivated us as a group of environmentalists to work on his proposed problem idea. Beyond the project and the required process involved in it, Assoc. Prof. Dr. Muhammet Fatih HASOGLU was an idol for all of us during our Engineering education. Being a good person and doing the best you can on the process of life duties was the main approach that he has taught us. Looking from the bright side of every problem was the main thing that we learn as engineers of the future. During this process Assoc. Prof. Dr. Muhammet Fatih HASOGLU has helped us way more than once in a week, he spends all the time whenever he is available. We would like to thank to Prof. Dr. Muhammet Fatih HASOGLU for his great help and the knowledge and experience he has shared with us. We also would like to thank Asst.Prof. Bülent Haznedar, for his excellent information that he has taught us and we have applied to this project. We also would like to thank Prof.Dr. Veysi İşler for teaching us the most helpful lesson which is Software Engineering and giving the ability to apply it on our Graduation Project. We also would like to thank Dr. Saed Alqaraleh for giving every necessary information throughout this process with the best effort he can.

Hüseyin Faruk Göktaş

Mehmet Taş

Eyüp Kazım Göymen

2.INTRODUCTION

In this section, we have given some introductory information about the project.

2.1 Purpose and Scope

Orchard Management and Tracking Application is aiming the orchard owners to have a better the management of their orchards as well as tracking of all data that comes out in each harvest period. During the use of this system, the target stakeholders which are the Orchard owners will be more capable of balancing their both budget and defining the requirements of each harvest period with better-experienced harvest periods.

2.2 Problem Statement

Orchard Management and Tracking Application wants all orchard owners to produce their products while having the awareness of all procedures that they apply during the harvest period.

Today we have many people gets participated in this industry making a life living and most of them having no clue of which process to apply in how much density and how to predict the current harvest's period operations with the data of the previous harvest period. When having this problem, the operation that they apply to their products on their orchards just relies on the estimation that they have experienced in the previous years or it is all written on to the paper and this may causes many misunderstand results. If we ignore this problem, resources that the orchard owners spend will increase and they may miss critical operations where they suppose to apply and productivity of the orchards that they own may dramatically change.

We have develop this system in a way that it will handle this problem with most accurate operations being applied.

2.3 Solution Statement

Using the Orchard Management and Tracking Application, each data and the implementation of this data to the field with aspects that the many constraints including geographical location, the weather and aimed product will be collected based on time and recorded into the system. The system will be creating guidance to the owner with processing all this data with time and many other bases (product, harvest time, tree etc.)

2.4 Contribution

Current technologies in the industry limited with focusing on general scope of the orchard of the owner. Either they have chosen to use technologies that reveal and transfer specific data with using lot technology and this is a big cost for the small orchard owners and the owners who just want to keep track of specific tree as well as while viewing general data of the field. Contribution of our project into this aspects is the features that enables the orchard owner specify the tracking method and limitations as he desires and being able to manipulate this data, get more accurate statistics of the data as he collects more data into the system. The extended perspective that we try to contribute to our project will result in more comfortable and flexible management of the orchard.

2.5 Glossary

The glossary will be list at the end of the report.

3. LITERATURE REVIEW

Orchard follow-up management project is aimed to be used by people interested in agriculture. Briefly speaking, the application will have many features such as garden planning, tracking of tree health status, amount of product obtained and so on using the mobile application in real-time.

3.1 What is the reason for the application?

Nowadays, smart systems, which have become widespread in almost every sector and need to be used, have shown their effect on the agricultural sector. In the agricultural sector, there are some systems and applications developed for job tracking and easy accessibility. The purpose of this mobile application is to facilitate the work of people interested in agriculture and to manage the work by managing it more easily and efficiently. It is to save costs and time as well as facilitate the operation of the works. It will allow garden owners to store information about their gardens with the mobile application so that they can easily see the data for each harvest period.

3.2 Methods used before application?

Before these systems and applications, they were managed by complex, primitive and sensory methods. As a result of these unsystematic and primitive methods, the opportunity to get jobs easier and fewer costs has not been utilized. Those interested in agriculture tried to manage it through the notes they took in a book. It is difficult for a garden owner made up of different products to easily calculate the income-expense balance and display it on a schedule. It is also difficult and tedious for the owner to calculate how much yield each tree receives for the harvest period.

3.3 What are the solutions?

This mobile application will be available to those engaged in agriculture, farmers, and people interested in agriculture. It will have a simple interface and will be easily used by people with or without knowledge of agriculture. If we want to talk about the functional solutions of the application, these are the solutions that are based on the problems described in the section 2.2. With this application, the user can monitor the health and productivity status of the network in real-time and add comments and notes according to the health status of the trees.

Another feature and solution to the problem is the knowledge of the amount of product per tree for each harvest period. It was not easy and accessible to use this data with the traditional note-taking methods used before. With this mobile application, the user will be able to see the number of products falling on each tree upon entering the data of the products collected at the end of the harvest. Another solution to the problem is spending, income, and expense balance. The solution offered for these situations in practice will be able to record user expenses and get their graphs. Thus, the user will be able to see how much income and expenses are generated during each harvest period. If the spread of this situation to a period of three years, the data obtained will be separated into years. In general, this mobile application focused on finding solutions to the problems experienced by the garden owners and tried to solve the problems.

3.4 What are similar applications?

There have been a few previous applications, but they are not identical. Because of the general structure of applications running on the web. Our application will be used as a mobile application. The references to these products similar to our application are as follows;

3.4.1 OrchardApp

In this application, users can follow the activities in the garden. With the application can learn about costs. It enables irrigation using smart irrigation systems. In certain periods and according to weather conditions, mold and fungus warnings. It is trying to make agriculture a lot smarter. In addition, users can access this application from the web [1].

3.4.2 Croptracker

In this application, users can produce different products in their gardens. But they can do this by dividing their gardens into shapes they want. Processes that start with production, spraying, irrigation, product separation according to quality (with image processing), packaging, transportation, etc. can be done. This application is available via the web [2].

3.4.3 Hectre

Both IOS and android users can use mobile applications. It is used as harvest management. With this application, users can monitor the trees in the garden via GPS. It provides the user with warnings for the work done as a timeline. To make it easier for users to manage their trees, each tree has the ability to add notes and photos separately. You also have to use powerful devices for this application which takes up quite a lot of space on both systems (IOS / Android). Otherwise, the application cannot benefit [3].

3.4.4 Cropdata

This application, which supports its users through the web, serves garden owners with a simple kiwi garden. CROPDATA, which is a very limited application, does not have many features. According to the information about the garden, the estimated profit and annual budget are created. Users can view yield records based on trees' performance history. We can say that this application, which has limited transactions, does not appeal to many audiences [4].

4. SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Introduction

In this SRS part, we will explain technical details about orchard management projects considering its score features, user requirements and system requirements.

4.1.1 Purpose

Orchard Management and Follow-up Application allow fruit owners to monitor the products they intend to produce at the end of each harvest period with a better understanding of each tree and the whole orchard. It also saves both expenses and time. Orchard management and monitoring application aim to minimize the workload of those engaged and interested in agriculture. It presents graphs of data more easily with the generated algorithms.

4.1.2 Intended Audience And Reading Suggestions

This document can be used by readers of orchard owners who aim to see this project as a real-life application by reading the general scope of the application, which will be explained in detail in the remainder. Developers interested in the platform (android) will refer to the general idea and flow of the architectural design of the project. The person responsible for managing this project development process will refer to this document to manage each of this team.

4.1.3 Product Scope

The aim of the project as stated above, we aim to benefit the orchard owners and viewers interested in the scope of this business to achieve a better understanding of the product. The amount of crop harvested by each tree and the entire area will be compared to each tree's income, weight, price, cost by specifying parameters such as income and all of these costs. Both financially and time will be saved. With the mobile application, users will be able to enter and store information about their garden effortlessly. This way, users can easily access valuable data about their garden. Besides, with this mobile application, the user will be able to see the graphs that are important to the user based on the data entered.

4.1.4 References

Some of the utilities we use to develop the application are presented below.

4.1.4.1 What is MongoDB?

MongoDB [5] is an open-source NoSQL database. There are many NoSQL databases such as Cassandra, BigTable, Dynamo. In MongoDB, each record is expressed as a document and these documents are stored in JSON format. The table structure in relational databases is taken here by collection, row by document and column by field.

4.1.4.2 What is Android?

It is a free and free Linux-based operating system developed by Android [6], Google and the Open Handset Alliance for mobile devices. Although the system is open source, a small but very important part of its code is kept closed by Google.

4.1.4.3 What is Nodejs?

Node.js [7] is JavaScript. It can work anywhere without the need for web browsers. It is the easiest way to write both backend and frontend codes of software. Today, most applications spend most of their time requesting databases or various services on the Internet and waiting for the results. Node.js is by nature asynchronous. It makes requests in parallel instead of making them one by one. When the requests are finished, they N call back N Node. This allows you to quickly process a large number of requests.

4.1.4.4 What is Npm?

NPM, short for Node Package Manager, is two things: first and foremost, it is an online repository for the publishing of open-source Node.js projects; second, it is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management. A plethora of Node.js libraries and applications are published on npm, and many more are added every day.

4.2 The Overall Description

This section is going to provide background for external systems that related to orchard management system. Briefly, Orchard management system is an independent system so that there will be soon backgrounded about internet provider, host provider etc...

4.2.1 Product Perspective

For the users who will use this application, we can compare it with the applications mentioned in the similar applications section. Let's do a separate review for a few of them. First of all, we can compare this application with the "Farmable" application which was made and used before. The Farmable application can be used by our garden owners as well as ours. First of all, the Farmable application allows users to control their garden via GPS. As another feature, users can control the trees in the garden and have access to the applied operations in the application. The difference between the Farmable application and our application is that the application does not show a general statistical result to the user. Another difference is that the application will be presented to users as a mobile application. The Farmable application is used as a desktop application. In addition, the user can store product photos in the Farmable application.

Secondly, if we compare our application with another example, we can give "Orchard Management" application as an example. In Orchard Management, it makes it easy for users to save all orchard information in one place. The application is perfect for small or medium-sized businesses and allows uploading images, comments, and dates for later redirection of information. It also has the ability to track sales, expenses and how much harvest it makes in each season. This is a paid app and is available on Android operating systems. However, our application is a mobile application and will be offered to users free of charge.

4.2.1.1 System Interfaces

In orchard management software, there's only one dependency that is going to be used to fetch weather per each day to show several statistics to the orchard owners. Except that, the system is going to be a totally independent system.

4.2.1.2 Interfaces

In this section, we are going to talk about entire interfaces that effect the orchard management application.

4.2.1.3 Hardware Interfaces

The system has hardware requirements in both the android side and server-side. Since, we use the free plan for the server side, we don't deal with the hardware of the server-side that much, and obviously it mostly about the number of the users so that it's not necessary to concern server side hardware requirements for the beginning. In the client-side, there won't be crucial hardware interfaces since we serve entire data in the server-side.

4.2.1.4 Software Interfaces

Orchard management project will have consisted of two basic parts, android side and server-side so that there will be some software interfaces or tools that rarely used to develop the systems that the project needs.

So, in order to develop android application, we shall use the last stable version for macOS, 3.5.1. This tool provides several SDK in order to improve code quality for android applications and improves efficiency at all.

In backend side, there will be some components that help to make application program interface such as Visual Studio Code, MongoDB, Robo3T etc. We are going to use VS Code with 1.39.1 version which is the most reliable version at all [9], the tool provides the developers to write script language codes efficient way, means with several extensions. For instance, we are going to use VS Code terminal extension in order to see the terminal screen in the bottom of the window without moving another screen at all. The other tool is the MongoDB 4.0, a tool to create NOSQL the database in local and remote servers. The last one is Robo3T is the application of MongoDB that shows entire collections in database such as SQL Server for SQL systems.

4.2.1.5 Communication Interfaces

In the orchard management application, the system will have two backgrounds: client-side (android) and server-side (API). For sure, there will be some communication interfaces, protocols that enable the system to interact with all subsystems in a concurrent way. The server-side will have consisted of RestFul API, that uses entire HTTP methods in order to communicate through server-side and client-side. These methods are briefly: get, post, put and delete. In order to send a data to server-side, client-side sends a post request, simply for the updating the data in a specific record, client-side sends put request. Also, the client side will use the header of the request in order to define content type and bearer token too.

4.2.1.6 Memory Constraints

As a priority, Android users must have at least Android 4.4 to download the application. In addition, to use the application in a healthy way, the user must have 256-512M RAM in the device. On devices with lower RAM, the application will be difficult to use. Because there are not enough conditions in practice, contraction and strain will occur.

4.2.1.7 Operations

In orchard management application, there will be some operations related with back-up via Heroku (PaaS) service. Basically, all back-end necessary functions, databases will be served on heroku services so that this service will be responsible to back-up operations in the orchard management application.

4.2.1.8 Site Adaptation Requirements

In the orchard management application, there will be no site adaptation requirements at all since the Heroku will handle all the entire requirements about adaptation.

4.3 Product Functions

To clarify the flow of systems and product functions, this description will be made as used in the program. The application will host to user information will be comfortable. When all the redirects he needs to use the application, and later on, he will be able to help. The user will be able to understand each step easier, and with a simple and simple interface, the user will be able to understand it.

4.3.1 Initial Application

Users of the system will be greeted by screens that describe the system with its features. The application will be introduced to the user with easy to understand sentences and visuals.

- Features will be described with descriptive illustrations.
- The user will be given explanations and instructions.
- Introduction of the application.
- Information about what can be done with the application.

4.3.2 Application Registration

The user will be authenticated to the system to have unique access to different devices and fulfil their security reasons. With this verification, problems that may arise between users will be resolved.

- Registration will be completed with the necessary information.
- Each user will have their username and password.
- The user is logged in with correct information.
- When authentication is successful, the user is directed to the system interface.
- The user cannot use the application unless registered.

4.3.3 Specifying a Location in the Application

When the user first opens the system interface, in satellite mode, they are guided by instructions intended to help them identify orchards with the boundaries on the actual map. The application interface, which will provide various options to the user, can be assisted by the ready moulds and will also be able to draw (determine) the garden environments by the user.

- Users can search location on the map in real-time.
- The user will find the location request requested by the application.
- Both users will draw the map by hand and set boundaries.
- The specified location will be saved to the application and the location will belong to the user.

4.3.4 Specifying Orchard Content

Once the orchard is successfully displayed on the map in Satellite mode, the user will be prompted to add some sample trees. There are many situations for users who are directed to add them to satellite mode. Therefore, the application provides many scenarios for user satisfaction in this regard, leaving the user flexible (free) in the garden layout. Each user will be able to create his garden as he wishes and plant various trees by dividing his garden as he wishes. If the user wants to create a garden of mixed trees, he will be able to create a garden of mixed trees by naming them separately for each tree, rather than a single species garden.

- The trees of the garden will be positioned according to the mobile application's location (location) request when this feature is applied, the manual system will be dragged and released.
- One of the key features of this section is that it allows the user to create ready-to-use templates and apply them to all trees considered the default tree.

- Specific information for specific trees that appear special by the user will be provided manually.
- There will be a health indicator for each tree indicating health status. In this way, the user can easily see the location of the tree, the health status of the trees can easily specify and write notes about these trees.
- Enter the characteristics of each tree (planting year, type, etc.).
- The user will follow each tree privately and closely, thus achieving a more regular and more efficient harvest period.

4.4 User Characteristics

System users will benefit from the mobility of this system. This opportunity system will operate at 7/24. Data entry time depends on the user. The user can use the application at any time. The user can make any changes after registering and after making the garden formation. Also, expenditures and revenues can be entered instantly. The user can keep special notes about the trees he owns and makes changes to the tree health at any time. Use of the application is entirely dependent on the user. A user with many gardens and trees will have more use of time. Because you will enter data based on the garden and trees.

The data collected during this period will generally be processed at the end of the harvest period and the processing of the data will be provided as the desired user during the harvest period. Every data that the user enters into the application plays a very important role for the user to benefit from the application. The user must enter the data correctly and completely in order for the results to be formed by the user to be entered correctly. Because all the graphs that will appear at the end of the harvest period will consist of the data that the user will enter during the harvest period.

The most needed user class to define the system success, which is the processing of data collected from the user, will be processed by the backend to be operated and the data interface is given to the mobile application. Since the application depends entirely on the performance of the user, the better the user performs, the better the result.

Therefore, if the user starts to use the application in terms of the character and then the missing data and incorrect data are entered, the user will get a result accordingly.

4.5 Constraints

- Application is an Android device that users need to use. Users with Android will be able to download the app from Google PlayStore.
- Another requirement after downloading the application is to register the application. It is an important stage for user security and each user will have their own username and password. Candidates will not be able to benefit from the application without registration.
- After creating a registration for the application, they should access the location of the garden they have via GPS. However, they can perform operations in this area by registering the accessed location. They will not be able to perform an operation without specifying a position via GPS. In addition, users will not be able to choose a garden that was previously selected.
- The correct and complete entry of data into the application is also an important stage. If the user wants to obtain an accurate and healthy result from his / her transactions, he/she must enter the data accurately and completely.

- The data that will be displayed as a result will come out entirely from the data entered by the user. The application will not add data to itself and make calculations.

4.6 Assumptions And Dependencies

In orchard management system, there are a lot of constraints that might affect the entire SRS overtime such as preferable operating systems, some specific user requirements and some individual system requirements that are affected by user requirements too.

The orchard management system will develop in the Android operating system, rather than both Android and iOS. There is also one more constraint, web applications. So, those operating system changes might affect the entire SRS in terms of system design, no doubt. The Application Program Interface (API) will be designed considering that issue, there will be some open endpoints that can be reachable in any platform so that there will be only changes in system design, not in database design.

Some individual and extra user requirements might affect some other constraints and system design, so that this report might be needed to be changed over time in the case of user requirements have been changed in a crucial way.

Scheduling accuracy is one of the other constraints that might need to be considered in terms of the priority of some core features. In a case that project scheduling is longer than what it's given, some features works might be added into SRS. For instance, in the orchard management system, there will be some feature works related to data processing that is depending on individual tree, orchard and harvest period that might be added into the entire project. But this constraint might affect the whole SRS, so that the aim of the project might be defined with some other terms, not as an only tracking system.

Human resource availability is another dependency that might affect the entire SRS documentation. This metric is the one the most crucial constraints that might affect project scheduling directly, if there will be some developers concerns with RestFul API development and web development, then the project might serve for web platform and the overall system design and the SRS might be needed to be covered once again to supply concurrency for all platforms that project supports and for some security issues as well.

The availability and performance of the hosting that is served in 'Heroku 'might affect this report in terms of performance, response time and etc. Depending on some parameters that can evaluate the entire back-end system, there might be some changes in back-end side and some endpoints and this might affect the entire report too.

The number of users is another assumption that can cause some problems that might be considered immediately. In order to increase the number of users support in the system, we might need to consider some user features and test all the performances and redefine the priority of all functions.

4.7 Apportioning of Requirements

In this section, we will explain primary and secondary requirements.

4.7.1 Primary

In the orchard management process, there will be some data processing features for future work considering schedule accuracy and the number of core features that have to be

implemented in the orchard management application. This data processing topic might be divided into two main parts: future value expectations and current data split such as the number of chemicals that help to produce the best product considering the pricing, health of the trees and so on.

There will be some platforms that can serve users to reach the orchard management system such as iOS and web browsers. As we mentioned above, the project has limited time and there are some core features that have to be done in defined deadlines. This state is important to increase number of users that uses the orchard management system in several platforms.

4.7.2 Secondary

In orchard management system, there will be some sharing features between orchard owners so that each orchard owner will be able to see the status, statistics of the orchards that is close to the it's orchards and see the overall statistics such as expenses in all individual type of trees and communicate with orchard owners to make partnerships too. This will affect the entire system in terms of the definition of the application and system design, they all need to be recovered and define new boundaries and specifications to make a concurrent application. This feature might be visible in 2.x.x versions in the application.

4.8 Specific Requirements

This requirements section has split into two main parts: user requirements and system requirements.

4.8.1 User Requirements

In the user requirements, there will be all core features that orchard management system application will show up.

4.8.1.1 Tutorials

The application shall show tutorials about how to use orchard management system in depth. For instance, there will be roughly 3-4 pages that introduce the core features what application does.

4.8.1.2 Register

In the application, users shall register to the system in order to save orchard and status of the orchard without losing any data.

4.8.1.3 Login

Orchard owners who have already registered to the system shall log in to the system when they logout and try to reach their data in any other android device.

4.8.1.4 Adding Orchard

Orchard owners shall add orchards to the system, define the tree type and select the location via the map.

4.8.1.5 Listing Orchard

Orchard owners shall list all the orchards that they have with general pieces of information such as number of trees, number of harvest periods etc.

4.8.1.6 Defining Trees

Orchard owners shall define the tree types that they are going to use and add some default values in order to make overall calculations etc.

4.8.1.7 Adding Trees into Orchard

After defining the appropriate orchard boundaries, the owner shall define the trees over the orchard by selecting specific tree type which has already defined earlier.

4.8.1.8 Listing Trees

Orchard owners shall see all the trees over the specific orchard through harvest periods.

4.8.1.9 Defining Harvest Periods

As a nature of orchard processing, there will be some periods that owner gets the product in some specific periods. User shall define that harvest period with start-end finish dates per each year.

4.8.1.10 Expenses

Orchard owner shall define expenses per each harvest period in order to track the revenue model in terms of years and periods. Also, orchard owner should add expense individually for each treatments.

4.8.1.11 Adding Harvest Periods For Orchards

Orchard owners shall add several periods for their own orchards. For instance, the orchard should be trackable for several periods or years depending on how the user defines the harvest periods over time.

4.8.1.12 Listing Harvest Periods

Orchard owners should list all harvest periods and see revenues per each period and check out some details per each period depending on the orchard itself.

4.8.1.13 Adding Notes

Orchard owners shall add custom notes to each tree per each period. This note might help the owners to track the tree over time.

4.8.1.14 End of Periods

Orchard owners shall add extra parameters to each individual harvest period such as weights of the products, quality of the product etc.

4.8.1.15 Accounting Details

In each individual harvest periods, there will be some expenses, no doubt. User shall see all accounting details over periods with details, orchard name etc.

4.8.1.16 Revenue Statistics

Orchard owners shall see statistics related to revenue models over harvest periods. Also, orchard owner should select the period/time range in order to see the details in a specific time range as well.

4.8.1.17 Accounting Statistics

Orchard owners shall see all expenses over harvest periods.

4.8.1.18 Overall Statistics

Orchard owners shall list overall statistics over harvest periods which are predefined in the system. These statistics shall include period comparisons, quality of product's comparisons, cost comparisons all over periods etc.

4.8.2 System Requirements

This section contains some detailed pieces of information about user requirements in terms of system requirements.

4.8.2.1 Tutorials

- In the tutorials page, there will be scrollable sections that user can scroll and see the next tutorial about the application itself.
- There will be no functions related to the database.
- In the next tutorial, the user will be direct to the login/register screen if the user is not authorized, otherwise, user will be direct to the main pages.

4.8.2.2 Register

- In the register page, there will be three fields: email, password, re-password fields.
- The application will check if the fields are empty or not. All the fields must be filled. Otherwise, the user will see an alert.
- All the information that user typed will be check via register endpoint.
- There will be a check statement if the typed email is already registered, the user will see an alert error.
- If the registration process is successful, the user will be direct to the main pages.

4.8.2.3 Login

- In the login page, there will be two fields: email, password fields.
- The application will check if the fields are empty or not. All the fields must be filled. Otherwise, the user will see an alert.
- All the information that user typed will be check via login endpoint.
- If the login process is successful, the user will be direct to the main pages and the token that produced via back-end will be saved into android application.

4.8.2.4 Adding Orchard

- In the orchard page, the user will be able to add orchard via google maps. The user will define some pins to draw out the shape of the orchard and the system will automatically generate the boundaries via google maps.
- During that process, there will be some required fields: Name of the orchard and type of the tree that orchard has got.
- All the pin locations and other typed fields will be saved to the database through API.

4.8.2.5 Listing Orchard

- User will be able to see all the orchards that the user has added to the system. All orchards will be seen as a list with names.
- By clicking the specific orchard, the user will be navigated through orchard detail on map and see the all trees on the orchard.

4.8.2.6 Defining Trees

- The orchard management system will manage several trees on an orchard with separated harvest periods. Users will be able to
- User also will be able to update or delete the defined tree type over the system via API.

4.8.2.7 Adding Trees into Orchard

- User will be able to define tree type with extra parameters such as average cost, average revenue, expected production in terms of kg.
- User also will be able to update or delete the defined tree type over the system via API.
- User can remove or update the tree over orchard over the harvest period.

4.8.2.8 Listing Trees

- User will be able to see all trees over harvest periods.
- User will be able to select the individual tree and see details over harvest periods.

4.8.2.9 Defining Harvest Periods

- User will be able to add specific time periods, in other words, harvest periods with specific start and end dates.
- Each harvest period will have revenue parameter.

4.8.2.10 Expenses

- User will be able to add extra expenses by either clicking the tree and adding to the individual tree or to the orchard in a specific harvest period.
- Each expense will be special to specific time period. That means when the user adds a new harvest period, all parameters related to expenses will be zero.
- The expense that the user has added to the tree will also affect the entire harvest period too.

4.8.2.11 Adding Harvest Periods For Orchards

- User will be able to change, add harvest periods for specific orchards. In the process of harvest period changes, all the tree pieces of information will be copied into the new period with default values.

4.8.2.12 Listing Harvest Periods

- User will be able to see the harvest periods for specific orchards.
- First, all orchards will be listed in a list structure and harvest periods will be listed by clicking the individual orchard.

4.8.2.13 Adding Notes

- In the orchard management process, it's crucial to add some notes to track the tree by one by. So, the user will be able to add notes to each tree one by one.
- All notes will be visible for each harvest periods.
- All notes will be visible via tree detail.

4.8.2.14 End of Periods

- User will be able to define the end structure for periods individually in order to see all details that define the entire harvest period.
- There will be some required fields such as the weight of the product in total, total revenue and costs as well.
- All data will be saved in the database via API.

4.8.2.15 Accounting Details

- User will be able to define the end structure for periods individually in order to see all details that define the entire harvest period.
- There will be some required fields such as weight of the product in total, total revenue and costs as well.
- All data will be saved in database via API.

4.8.2.16 Revenue Statistics

- User will be able to see revenue model over periods with the data that we got from 3.9.1.15 in-depth. All these statistics will be shown with graphics over periods.
- All values will be calculated in server-side.

4.8.2.17 Accounting Statistics

- User will be able to see the accounting model over periods with the data that we got from 3.9.1.15 in-depth. All these statistics will be shown with graphics over periods.
- All values will be calculated in server-side.

4.8.2.18 Overall Statistics

- In orchard management system application, there will be several parameters, data that has got from in the entire application so that there will be several overall statistics such as the comparison of the product weights, accounting/product weights etc.
- All values will be calculated in server-side.

4.8.3 Non Functional Requirements

This section explains the non-functional requirements.

4.8.3.1 Performans Requirements

%99 of all request shall proceed in the correct order with less than 2 seconds response time. This is one of the most crucial parameters in the application that might decrease retention in a good way.

The android application shall support roughly 10,000 users concurrently. Only 2% of fatal errors are acceptable in the overall system. That metric is needed to be tested accurately to test out the entire client-side performance.

The API shall support 4,000 requests concurrently, this is one of the other crucial evaluation metrics that needs to be tested.

4.8.4 Logical Database Requirements

In the orchard management system, there will be several models that have relations with each other. In the API development stage, MongoDB will be used in order to handle good performance and better structure.

Depending on the application usage, the database must meet the all required requests in a concurrent way. The location of the database should be close to Turkey and has at least 4 cores for better performance.

There will be seven general models in database: User, Orchard, Harvest, Tree, Tree type and Accounting. The relation between the models is not much complex considering required models that are defined in future works.

Each user will have several orchards, each orchard might have several harvest periods, each harvest period might have several trees. That is the summary of the entire database diagram obviously. Besides, there will be some relations between tree type and accounting models as well. For instance, Tree's might have several expanses or orchard must have specific tree type.

4.8.5 Design Constraints

This section defines design constraints that can be imposed by other standards, hardware limitations, etc.

4.8.5.1 Standart Compliance

Specify the requirements derived from existing standards or regulations.

All data naming issues will be related to special to each platform. So that, client-side developers will decide the naming issue for their own, back-end developers will decide the naming issue for their own.

In the orchard management project, there will be no expectations about accounting for the whole project. So, there will be no procedures that deal with accounting.

4.8.6 Software System Attributes

This section will introduce non-functional attributes.

4.8.6.1 Reliability

The entire development process will be under TDD (test-driven development) principles to test out each module individually in order to avoid crashes. That means, each of the modules will have their own test targets and also the project itself will contain entire test cases with integrated modules with both client and server-side as well.

Each usage case will be tested individually and carefully such as network connection disconnection issues etc. We can taste all page actings according to the connection issues.

4.8.6.2 Availability

The system shall run 24/7 but there is an issue in our database side. In the orchard management project, the Heroku, PaaS service for hosting, will be used with free plans. The issue that the server gets offline in the 30 minutes time range without any request so that it requires a little bit time, roughly 1-2 seconds to get online again. In the upcoming versions of the app, the free planning might up upgraded.

In each week, there will be some checkpoints in order to match the requests that send from the application and the requests handled by API. Then, the missing part will be calculated and test the individual test modules in order to get better results.

4.8.6.3 Security

Security is the most crucial criteria in the orchard management application. So that, JWT(json web token) will be used to handle all security stuffs in the API side. Basically, when the user logged in, there will be some tokens on the client-side. So that each request will include bearer header that got from login request, if the header is empty, the server will return 401 status code, unauthorized. So, no one can reach the database without any application such as postman.

In future work, there will be also SSL-Pinning processes in order to create invisible requests as well. The case that the person who has a proxy app such as Charles can see all details of request including body of the request and header of the request. That's why we are going to implement SSL-Pinning process in the client side in future works.

All logs related with the user will be saved via Fabric, that means all user interactions with the user will be saved through google servers and see detailed reports in order to provide a more secure application.

4.8.6.4 Maintainability

All modules in the client and backend side will be open to scale, each structure will be created considering the future works and some constraints that we defined in the earlier of this report. So that in a case that a module needs to maintain will not be depending on any other modules so that it's going to be easy to maintain the individual modules one by one.

4.8.7 Organizing the Specific Requirements

In this section, we are going to describe non-functional requirements.

4.8.7.1 System Mode

In the orchard management, there is going to be only the Android platform, so that the developers will not deal with several modes.

The only issue on this state is that client-side might face different response times considering the server-side.

4.8.7.2 User Class

In this section, we will explain user classes.

- Orchard owners who are the end users that will use the entire system.
- Developers, who develop and test the entire system
- Lecturer who is responsible to read the entire report and test the application as well.

4.8.7.3 Objects

There will be some services such as Heroku in the application in order to use concurrent hosting in the application. Basically, there will be no other services that we need to use in the orchard management system.

4.8.7.4 Feature

All features are explained in functional requirements in section 3.8.

4.8.7.5 Response

Responses are going to be detailed in system design in section 4.2.1.2.

4.8.7.6 Functional Hierarchy

Functionally related topics will be covered in system design with several UML's.

5. SYSTEM DESIGN SPECIFICATION

5.1 Introduction

In this chapter, system requirements will be detailed for the following titles; Operating environment, system and subsystem architecture files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

5.1.1 User Interface Upon the First Install

User Interface of the Orchard Management and Tracking Application has been completed as mockup files based on the final requirements that we have specified. Since the system will be an Android Application, images below represent the mockups designs of our system based on the android (mobile) platform interface and its components.

Each Interface will be defined below with its mockup as well as the flow of the app given in detail. Processing login will be explained in each step and will be generalized at the end of the definition of user interface mockups completed.

5.1.1.1 On-Boarding

Users of the system will be welcomed by the screens that explain the system with its features

- Features will be explained by the defining images.



Figure 1: Mock-up of onboarding pages

5.1.1.2 Authentication

- User will be authenticated to the system to have unique access over the different devices and satisfy their own security reasons
- The register will be completed by needed information
- Login will be achieved by providing the correct informations from the user
- When authentication becomes successful, the user will be directed to the sytem interface



Figure 2: Mock-up of auth pages

5.1.1.3 Main Page

When the user first opens up the system interface, they will be directed by the instructions which will be aiming to help them define their orchard with the boundaries on the real map in satellite mode in Figure 3.

- User will locate the location request that is requested by the application.
- Either user will locate the boundaries by manually drawing on the map.

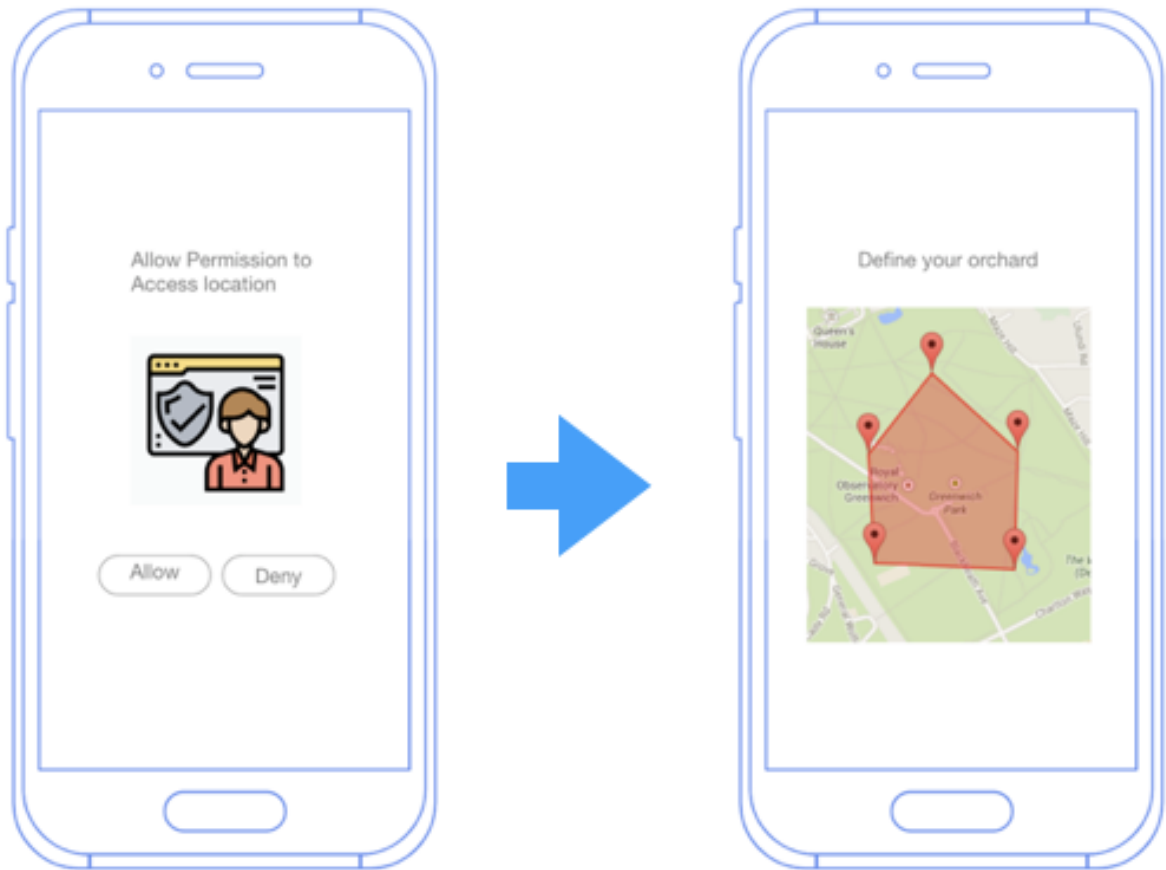


Figure 3: Selection of Orchard land on map

The process of defining the orchard will be defined as defining the corner points of the orchard as shown above and a polyline will be created with these locations. The location will be in precise values so that we can define it as accurately as we can as shown above. Also, the harvest time and the important values regarding this attribute will be entered.

5.1.1.4 Adding Trees

Once the orchard has been successfully shown on the Map in satellite mode, the user will be directed for adding some sample trees as shown in Figure 4.

Trees of the orchard will be located by location request of the mobile application either drag-drop of manual system will be accomplished when applying this feature

One of the important features of this section is that the user will be given the opportunity to create some ready to use templates and apply them to all trees that considered as default trees. Specific information for specific trees that seem specific by the user will be given manually.

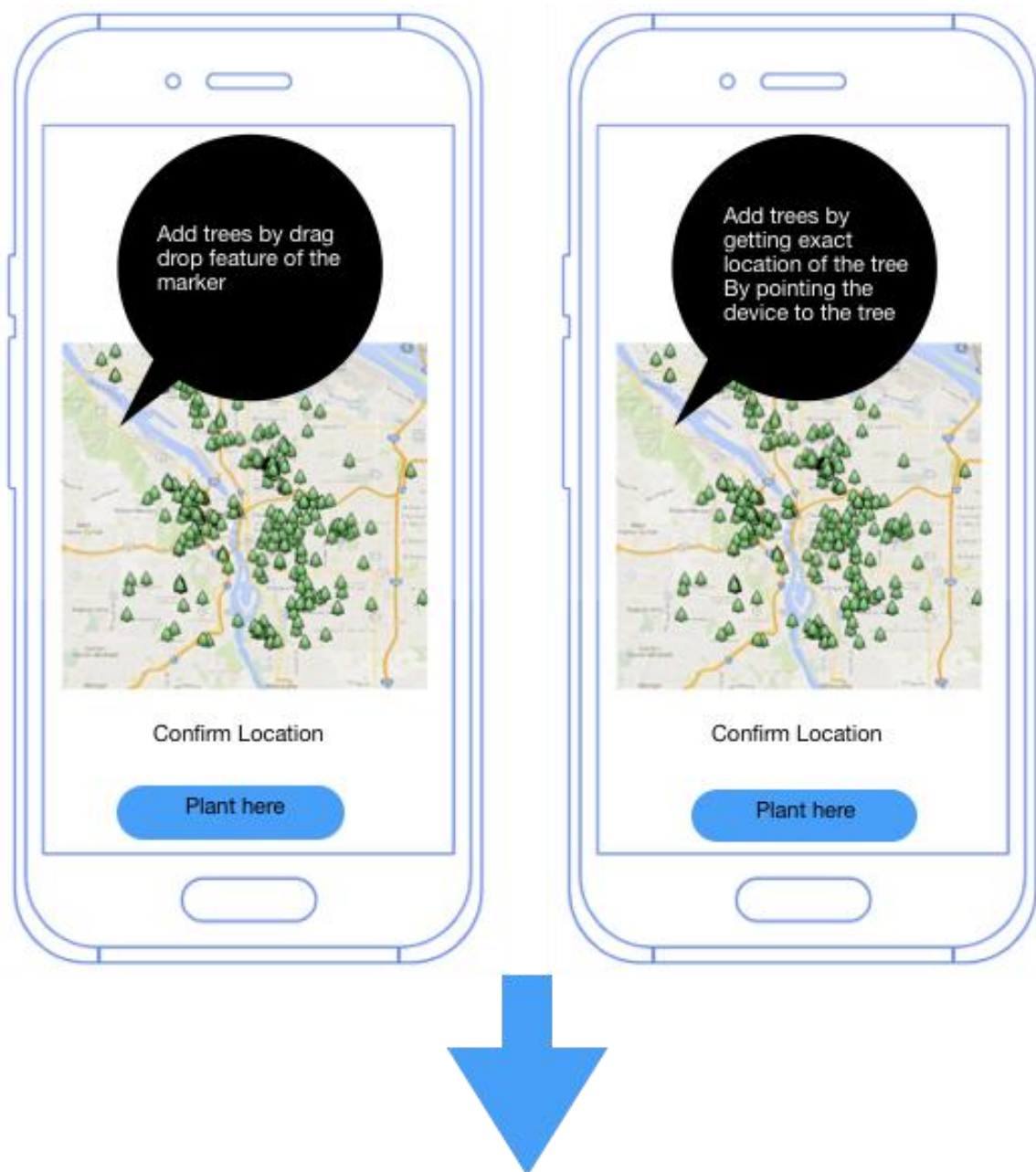


Figure 4: UI for adding the trees to the field

While the user was able to add the generalized template for the all over the trees in the orchard in order to not to enter the same information for hundreds of trees, orchard owner also will be able to create a specific template with the specific attributes for a specific target product producer individual(tree etc..)



Figure 5: Defining attributes for trees

5.1.1.5 General Application Flow

The actions and the required interactions for the first install and use are defined with mockup design. The general application flow of the project is shown in Figure 6.

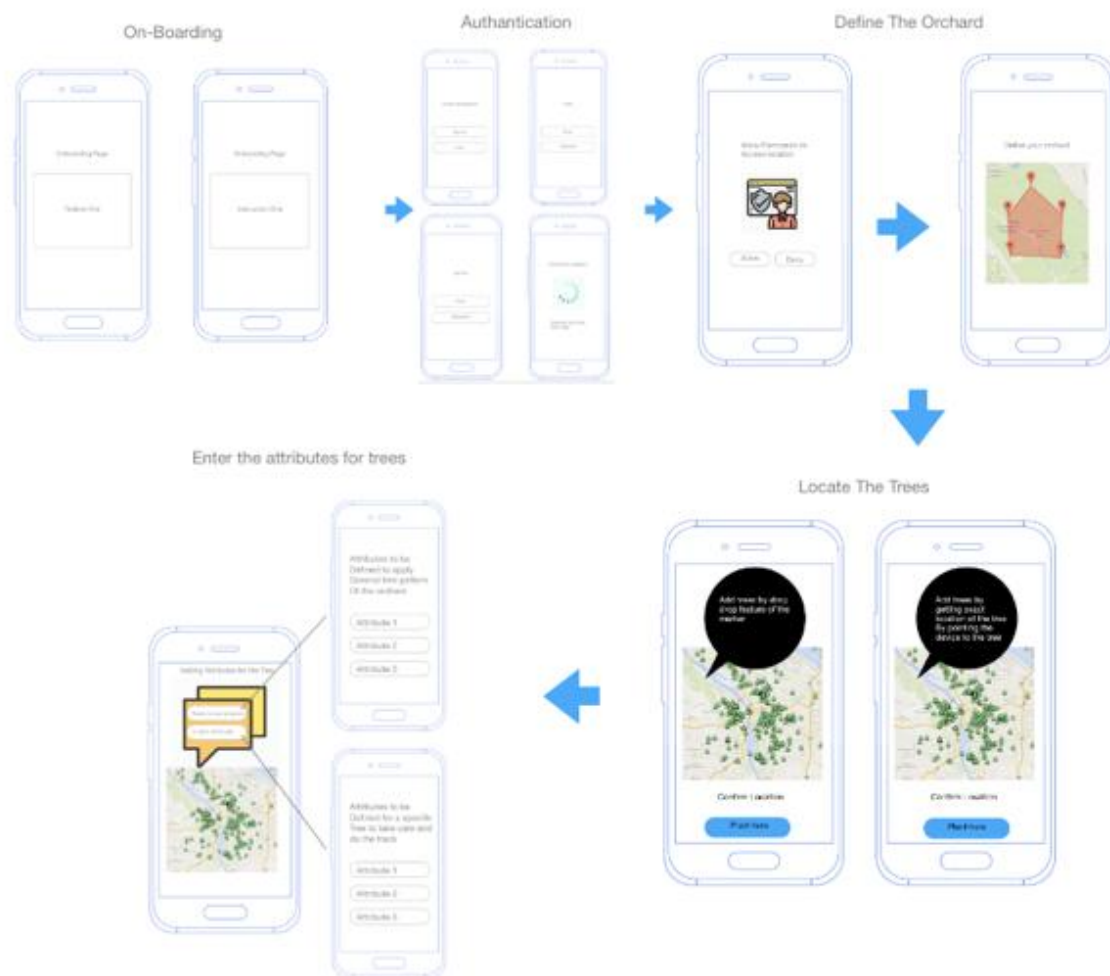


Figure 6: General application flow

5.1.2 User Interface with User Interaction and Data Presentation

5.1.2.1

Users can add multiple orchards and have multiple harvest times for each orchard. This will be represented on a separate screen with a design that navigates the user easily. The same user interface with the below mockups is also applicable for the harvest times and listing of all harvest periods.



Figure 7: UI for the list of orchards

5.1.2.2

User will have access to list all the trees that he has. The distribution of this list won't depend on one constraint, will be defined on different constraints such as a show by harvest periods, by performance etc. Each tree will have its own attributes, will show the quality rating. User may perform to add note and direct the interface to statistics of this tree based on the parameters as shown below in Figure 8.

Trees list UI with the data that is filtered by the parameters. Throug our program, it is possible to list data for individual tree filtered by the parameters for any harvest period(See Figure 8).

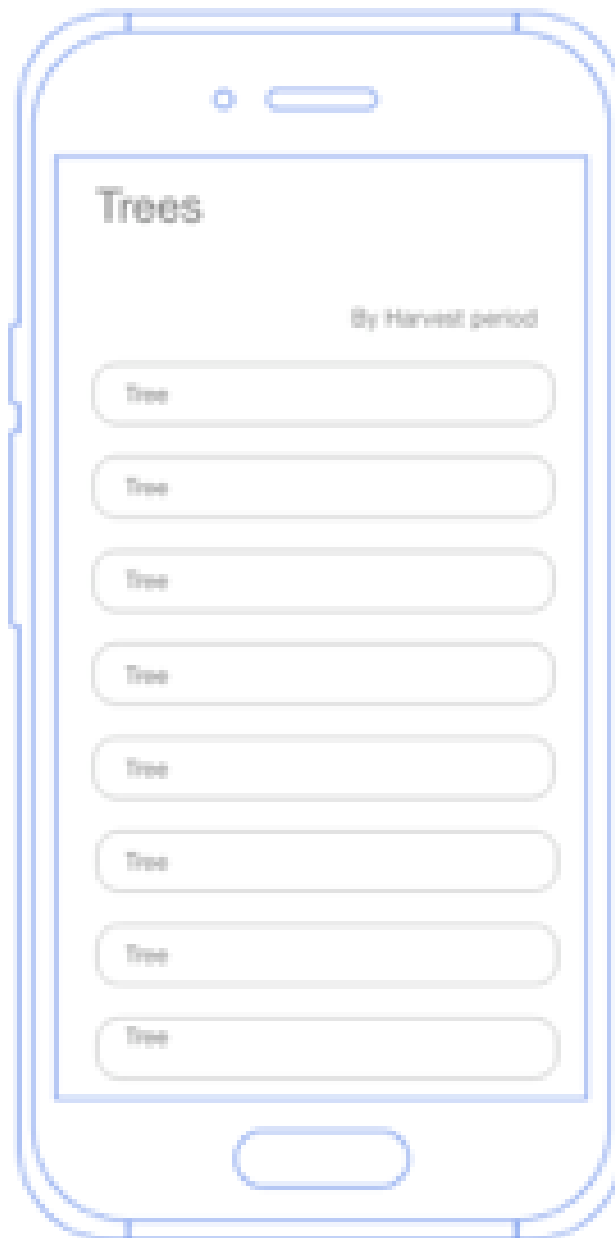


Figure 8: Harvest periods view

4.1.2.3 Trees detail page and statistics given by parameters



Figure 9: Tree details screenshots

5.1.2.4 Accounting Statistics

User will be able to see the overall accountings data on this interface as well as the revenue.

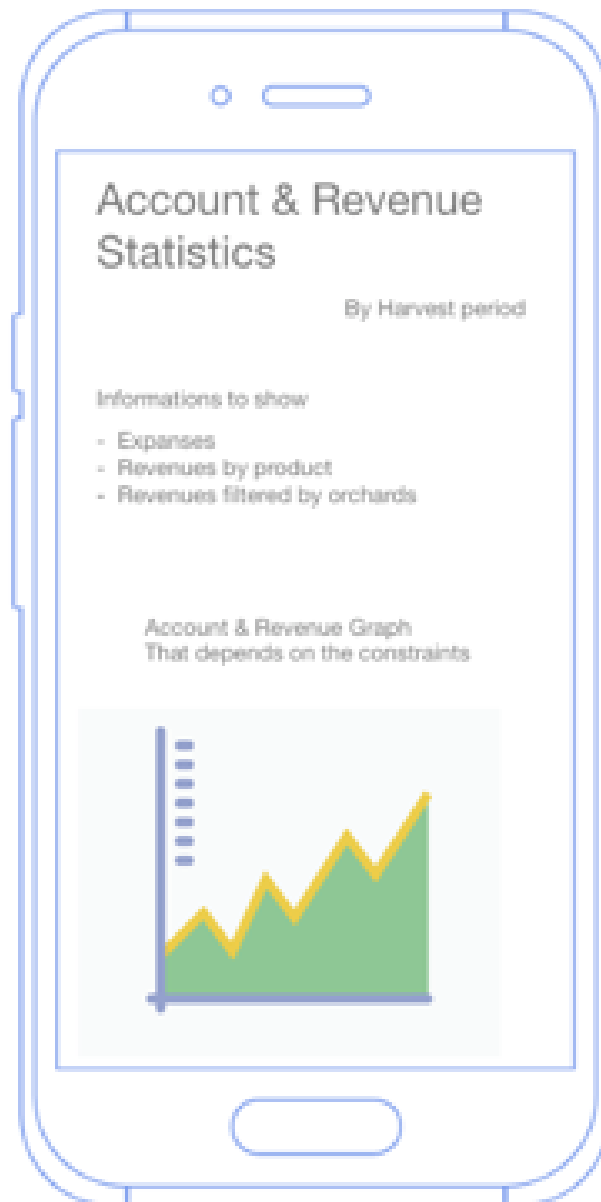


Figure 10: Statistics

5.2 System Architecture

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 4.5, External Interfaces.

5.2.1 System Software Architecture

The system will operate on the Android platform which works on the Android operating system and the data will be provided by the back-end. It will be retrieved from ... platform.

5.2.1.1 Android Platform and Architectures That Will Be Used

Since the system is going to be operating on the android platform, The design of the architecture libraries to use, development environment and the languages are explained below in detail.

Development Environment	Android Studio
-------------------------	----------------

The development environment for the system will be Android Studio it is based on the IntelliJ IDEA. It is a java integrated development environment. Enables the developers to develop on both java and kotlin programming languages.

It uses Gradle based build system to develop support application development in this operating system which is an Android Operating System thanks to this we can execute emulators and prevent the environment as real as the Android Operating System environment.

System Development	Kotlin
Interface Design and Implementation	XML

Languages to be used in the android platform will be the Kotlin & XML. Kotlin is the latest technology language on the Android platform that is announced by google in 2017 and it is a new language that is proposed and advised by the many industry-leading individuals. Many of the coder's startups adopted to the Kotlin programming language for android development.

In our project for the overall development, Kotlin language will be used. For creating the interfaces in the application, XML which is the only advised language to create interfaces in android development will be used.

Dependency Injection	Kotlin
----------------------	--------

When we develop our application when it becomes bigger in time which is a problem that we can see on any platform, not just on the Android platform, we will see that we will have a dependency de-coupling problem. The dependencies on our project will increase and the performance of the system will dramatically change.

When the dependencies increase, it'll be a nightmare to add new features and do some maintenance on the system.

To better manage this highly possible problem, we'll be using the Koin library to manage the dependencies as much as possible. The classes and the modules that we created with the Koin will be injected into the classes that we needed. We will try to stay away from not needed uses of dependencies.

Asynchronous - Non-blocking programming Programming	Coroutines
-----------------------------------------------------	------------

While we are developing the system on the android platform, it is not the only point to make the system functional. It is also important to provide an experience to the user that is scalable when needed. In this case, we should consider that our project should support Asynchronous non-blocking programming. For this statement, we'll be using coroutines which are also provided by Kotlin.

Android Architecture Components	Livedata - ViewModel - Databinding
---------------------------------	------------------------------------

When we write our application just with the pure knowledge of the language, the system won't as functional as we wanted it to be. There are some libraries that are supported by the Android platform itself which are Android Architecture Components. These components help us to write maintainable, robust apps.

For example one of these components is the LiveData which we will be using for most of the features in our app. When there is an update on the database or the remote source we don't need to put additional requests and use the memory, with LiveData all we need to do the changes when we notified by the LiveData when there is a change occurs on the database.

Android Software Architecture Pattern - MVVM

MVVM (Model - View - View Model) is one of the patterns that we apply to our software. The software which is developed without the Separation of Concerns considered is not that scalable and functional, to not live this problem in our project we'll apply MVVM to our system while we develop. The business logic and the user interface login will be separated thanks to this pattern. When we apply this pattern the UI code will be simple and the operations that are included in the business logic will be much easier to manage end edit.

We have three layers to apply in our project for the MVVM as shown in Figure 11.

1-Model: The data of our Orchard Management and Tracking system will be managed on this layer of the pattern of the software. When we search for the suggested implementation of this later from the Android Official codelabs, we'll see that it is better to implement the Model layer with the observables. That is why we use Architecture components which will make this observer structure of the app way easier and more maintainable. Observer structure detailed information is given in the Detailed Software Design of this report.

2- ViewModel: Our ViewModel layer will be working as a bridge between our data and the View. It will be interacting between these two and it prepares the observables that will serve, and be observable by the View itself. It should be noted that in our project ViewModel has

nothing to do with the View about the awareness. It just deals with the view when it comes to observing its data by the view.

3- View: All the role of the View layer will be to observe the data from a specific ViewModel and update the UI elements that it includes itself in a proper way without putting the business logic inside.

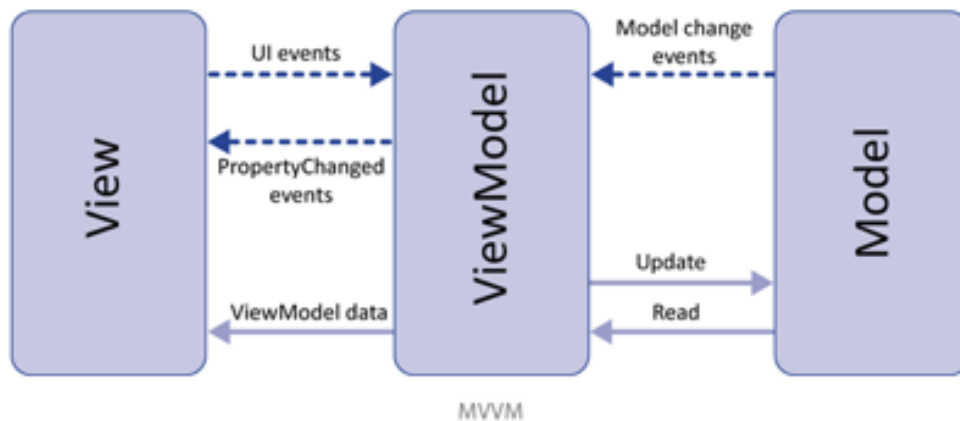


Figure 11:Structure of MVVM

The above diagram shows the interaction over the MVVM architecture pattern (See Figure 11). As we can see, View observes the data from the ViewModel, ViewModel will have nothing to do with the data on the View. As well as the same concept, ViewModel becomes the observer when it comes to the connection between the ViewModel and the Model. We can show this flow in detail with the diagram shown in Figure 12.

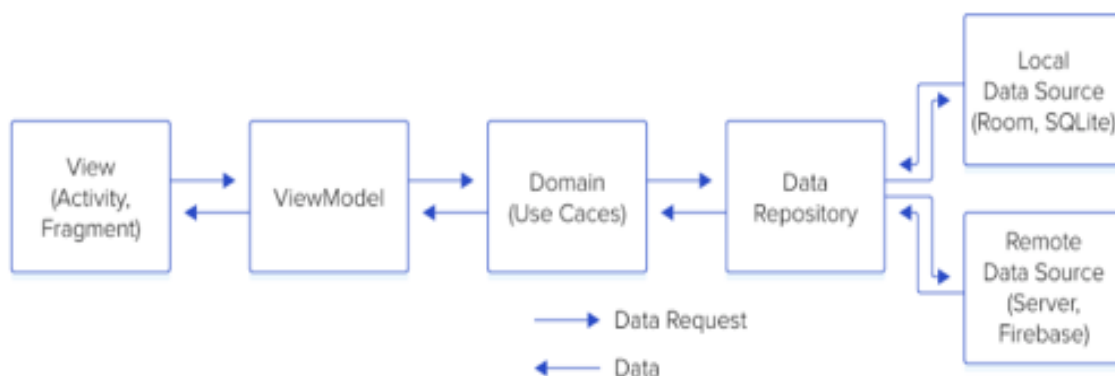


Figure 12:Structure of MVVM

The flow of our software will be just like the flow shown above. We will define our remote requests by creating a package under the model layer. This package will have all the remote services. Data consumed from these services will be transferred into the data model. This model will be consumed by the repository.

The repository is the place where we differentiate the data as remote or local. It is the place where we decide where to consume the data. When the data consumed from the remote service or the local databases, it will be transferred into the view model through our repository because in this case, ViewModel will specify the request.

When the request completed, the requested data will set on the LiveData property variable, since this variable is an observable variable that we have created the structure in our app, it will be observed by our UI which is the View layer in this case in our project and all the UI components will be updated.

5.2.1.2 Application Program Interface(API) Design

Since the API (Application Program Interface) layer exists in the orchard management application, the design patterns that decided to use, libraries that will help us and programming language will be explained in detail below.

Development Enviroment	Visual Studio Code
------------------------	--------------------

The development environment for the API system will be Visual Studio Code, source code editor, since the API is going to be written in a script language. Visual Studio Code, support many programming languages including javascript, golang etc. It's super fast and there are several plugins that help us to develop faster, such as VSCode terminal that enables developers to use terminals inside of the Visual Studio Code.

Programming Language	Javascript, NodeJS
Database System	MongoDB

Nowadays, javascript is the most popular languages in both backend and frontend technologies and used a lot, so that there are a lot of external libraries which are very useful and improve coding process such as NodeJS. NodeJS is the most popular backend library that has been produced for Javascript and quite used by developers. NodeJS has several packages to support easy, maintainable, testable backend codes.

In database design, the orchard management system is going to be built with MongoDB, a NoSQL database structure. It's rarely used in large scale applications that don't require that much response speed. It allows developers to create some documents/models and define relations inside the models not like ordinary SQL systems. By the way, it's much faster than SQL systems approximately 25% faster for queries.

Packages	Express, Nodemon, Passport, Jsonwebtoken, Babel, Body-parser
----------	--------------------------------------------------------------

There will be several packages that will help us to write better, testable API for the orchard management system. The entire system will be managed via Express, it's a node package that enables modern API structure. The other dependency is nodemon, it's a powerful developer dependency that runs server when any of the API files has been changed so that developer doesn't have to restart the node js's local server.

In the security section, we have mentioned the token-based API development structure, in order to implement that structure, we use passport and JSON web token (JWT) to produce safe token each time the user tries to log in to the orchard management system. In this case, we use passport to manage the entire token system, parsing without showing the developer.

The last dependency is body-parser, is a layer for HTTP requests, very powerful to collect all requests from HTTP and move to the router structure.

Architecture	MVC
--------------	-----

In the backend development era, MVC pattern is rarely used to implement a large system. So that there are a lot of sources about MVC with any programming language, that's why we prefer to use MVC architecture, it's testable and easy to manage each module.

In MVC structure, we have 4 basic layers: Model, View and Controller with router structure. So, we are going to deal with all these layers in-depth with such examples.

Model is a briefly MongoDB schema that is consisted of several variables to manage the model. For instance, in the orchard management system, there will be user schema which some parameters: email, password, hash, orchards etc. So that we define all required parameters, essential ones, in that model.

A Controller is the async functions that manage each endpoint, such that user models. There will be three main endpoints for users: login, register and logout. Each of them will be managed via the controller. So, the controller will have many functions that need to be responded via API.

A Router is the manager structure that routes each request to the related controllers. Such as the request that contains /user, will be managed into the user controller. That's the entire logic structure in the API development process.

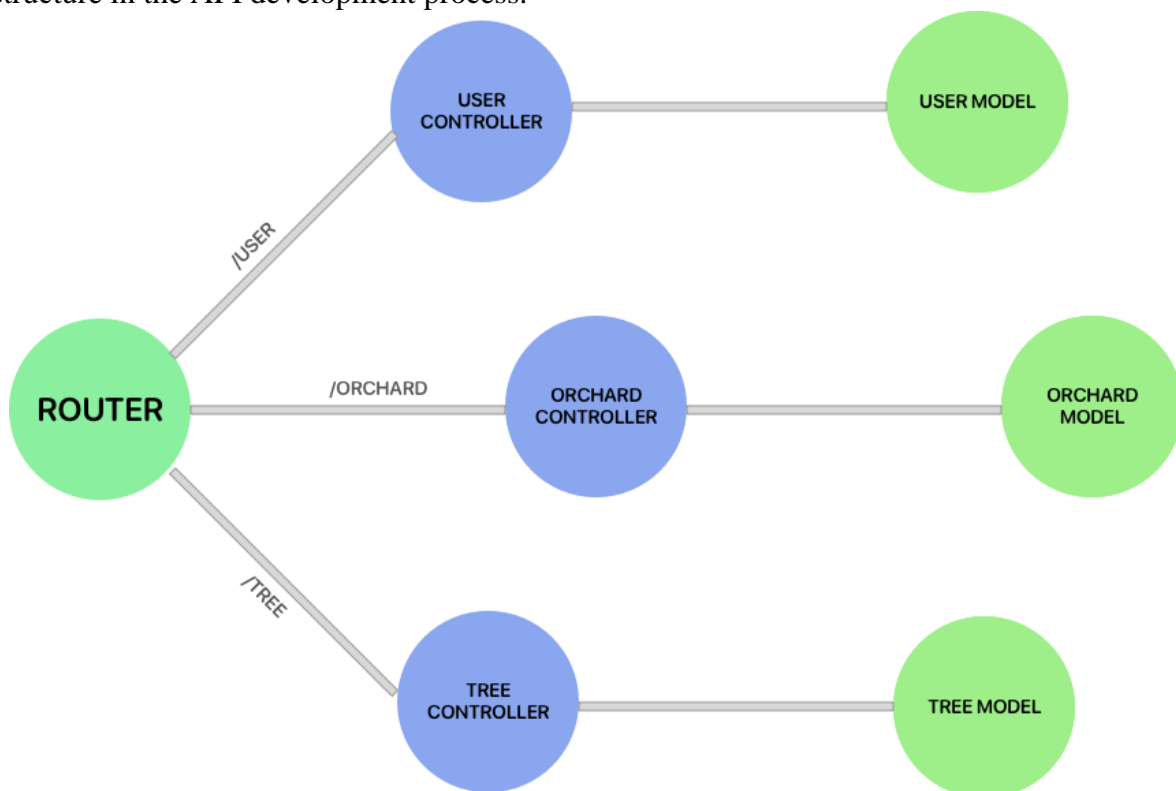


Figure 13: Router structure for API

List of endpoints	POSTMAN
-------------------	---------

Postman is rarely used to manage API documents, in the orchard management system, we are going to communicate with android developers and API developers together.

5.3 Database Design

In this section, we are going to present the database design in the orchard management system in depth.

Basically, there will be 6 models in order to handle all possible data structure in API. So that we use MongoDB to create all relations and database structure in the application.

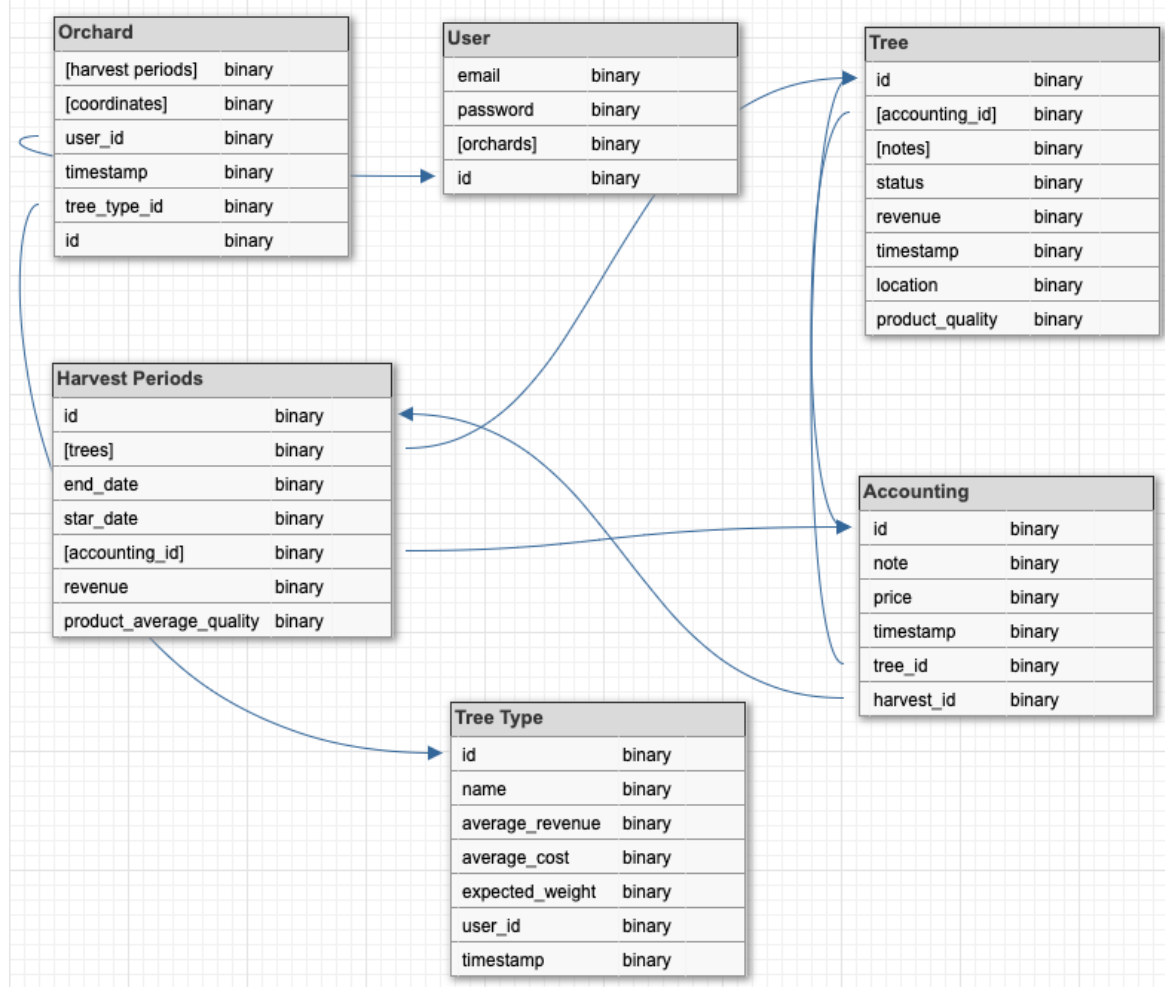


Figure 14: Database Structure

5.4 Human-Machine Interface

In this section, we are going to cover some system design instances with several UML's.

5.4.1 Entire System

In the orchard management application, there will be four basic systems that will be responsible to manage each subsystem such as orchard system, statistics system, tree system and finally weather API which is a dependent system.

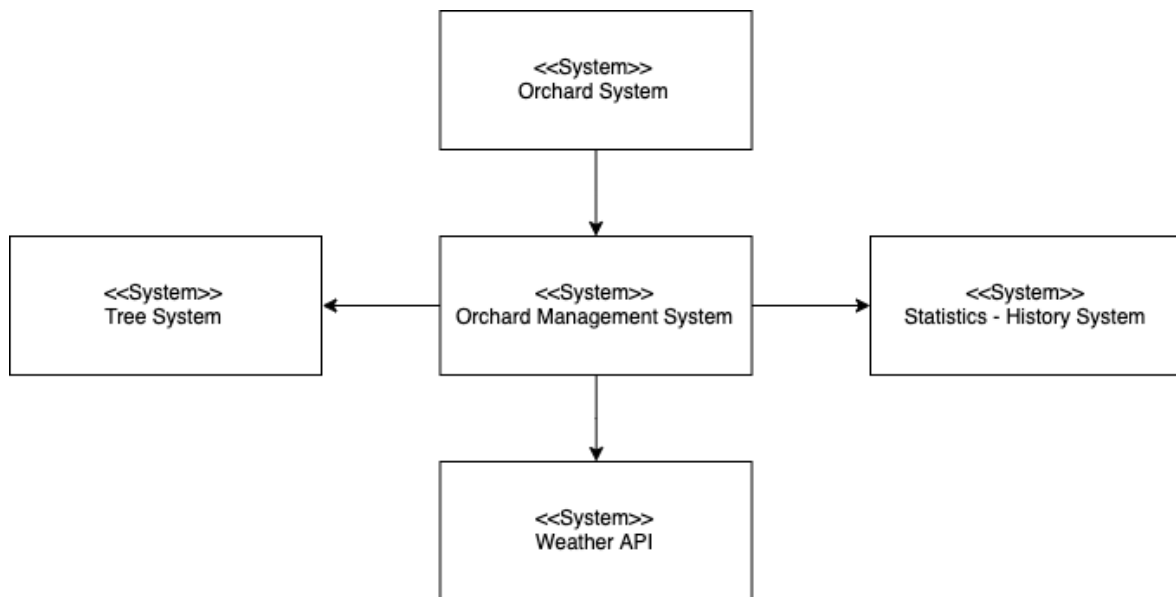


Figure 15: Use case diagram

In the system, there are three crucial systems that interact with each other a lot. These are orchard system, tree system and statistics system. The main feature that orchard management application offers is the statistics so that proper data that we need to handle via orchard owners is crucial to calculate proper/efficient statistics and show the orchard owners. There is also a dependency, weather API, is necessary to show statistics based on weekly weather statistics for orchards.

5.4.2 User Functionalities

Orchard management system offers several functionalities to orchard owners:

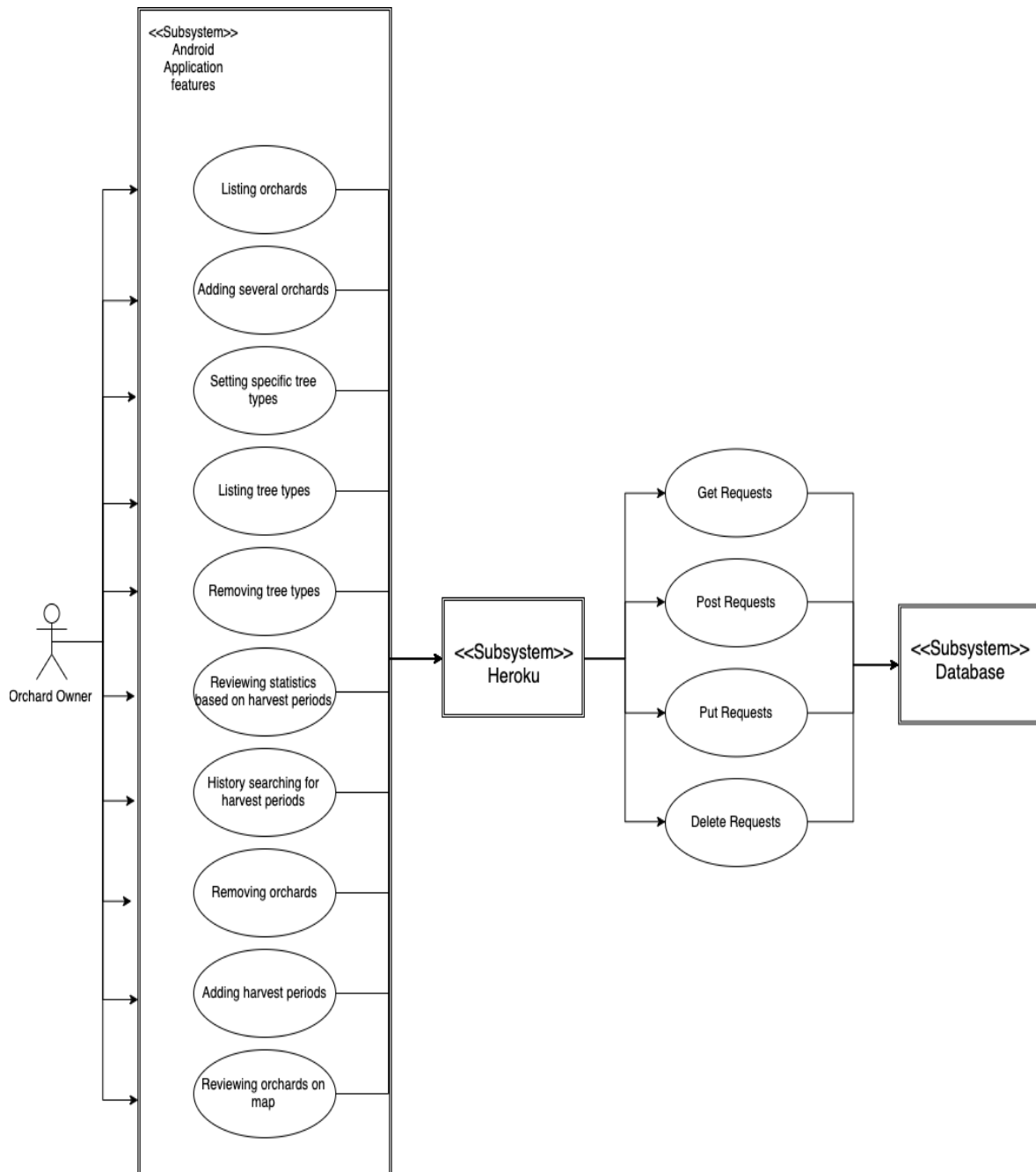


Figure 16: Use case diagram

As you can see from the UML, the orchard owner has a lot of features that easy to manage orchards. The next system is the Heroku, PaaS service that we hold our backend related codes and database data. The system gets several requests and directs to database depending on the type of the requests.

5.4.3 Instance Sequence Diagram

There will be some static ways to communicate between client-side and server-side. Let's take a look simple sequence diagram with HTTP methods is shown in Figure 17.

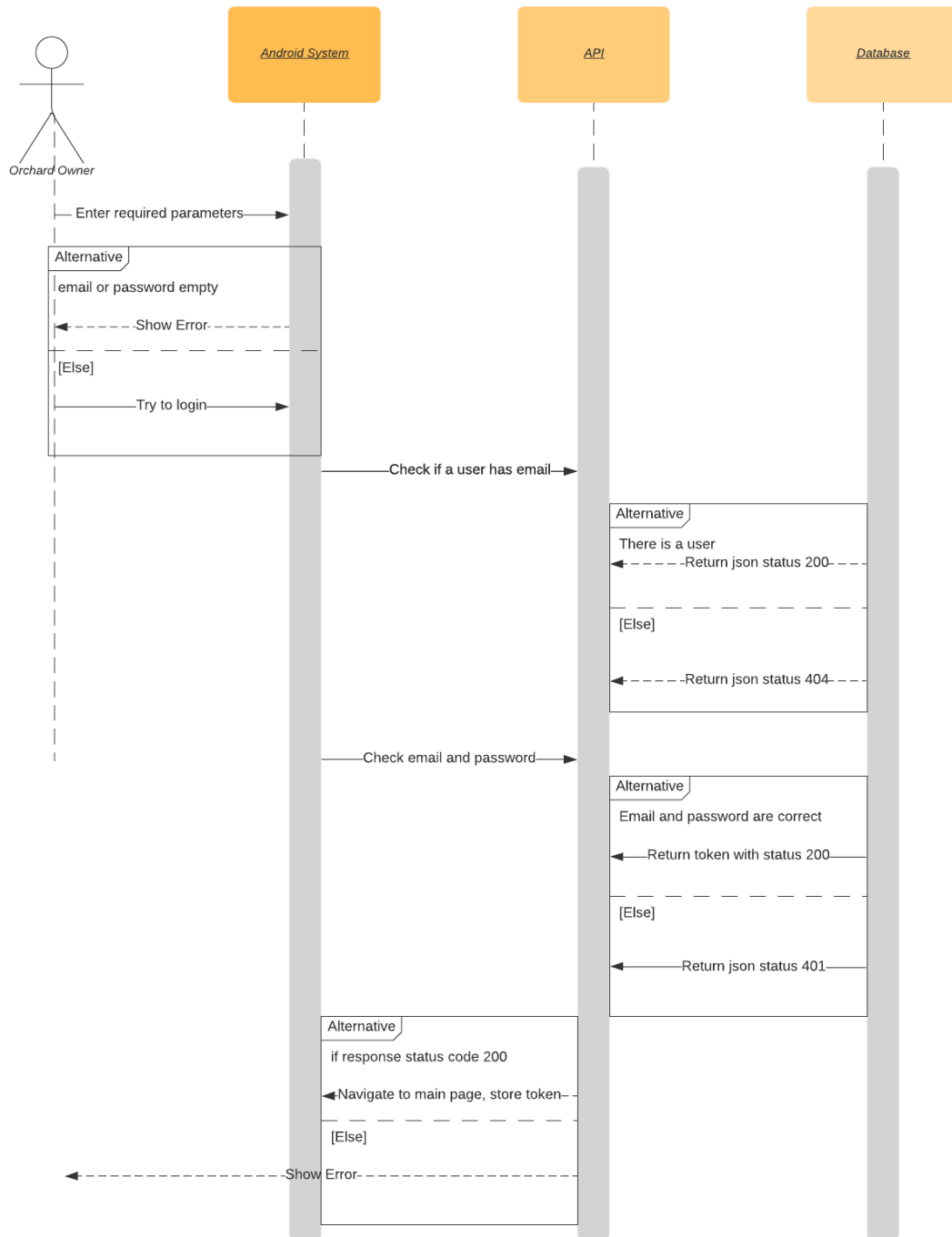


Figure 17: Sequence diagram that shows login state

5.4.4 Classes

Let's take a look how class structure UML's in android side is shown in Figure 18.

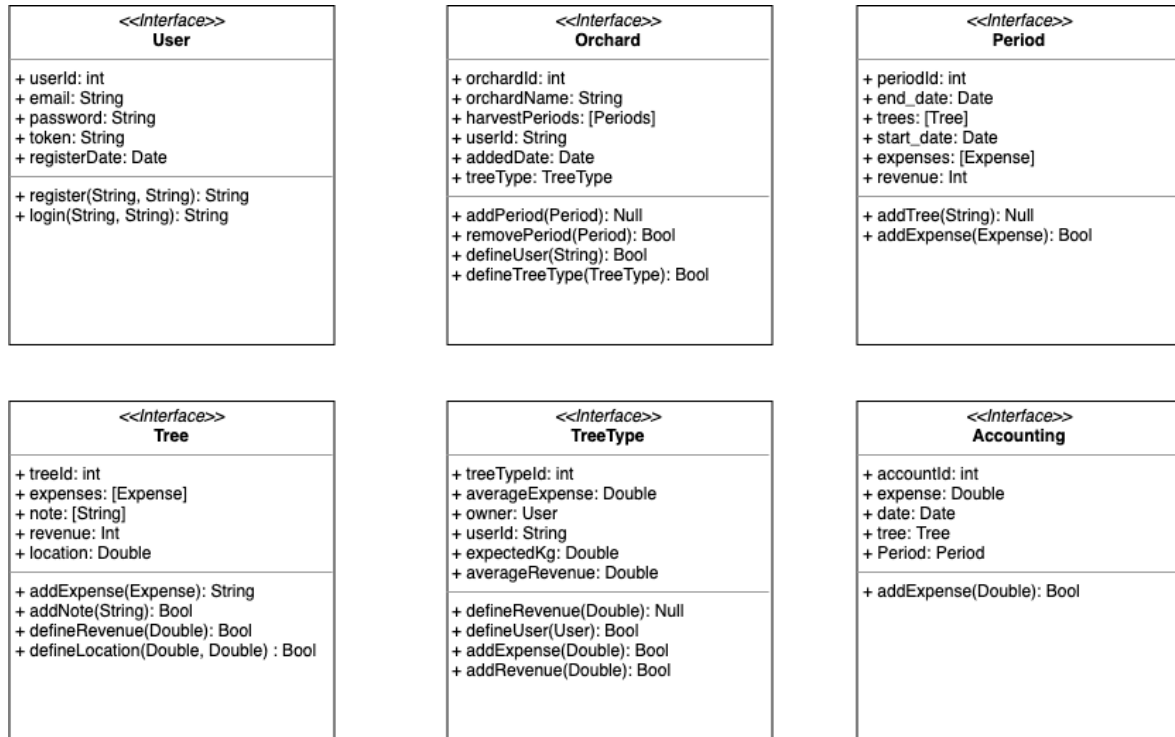


Figure 18: Class UML

5.5 Detailed Design

5.5.1 Software Detailed Design

Modules that are used for the Android platform is listed and detailed information given for each of these modules.

Kotlin Coroutines (Asynchronous Programming)

Kotlin itself as the language provides us with the non-blocking programming option with the module “Coroutines”. In our project for every service call and data retrieval operation, we will be using coroutines to do asynchronous calls. When we do the coding and this data retrieval operation without the coroutines, we have to use asynchronous and wait for calls a lot but with coroutines this module will enable us to not to have this complicated call blocks and we don't have to manage them. We will be adding the implementations for this module as follows;

implementation 'org.jetbrains.kotlin:kotlinx-coroutines-core:1.2.1'
implementation 'org.jetbrains.kotlin:kotlinx-coroutines-android:1.2.1'

As we can see from above, it also submodule of the JetBrains company which is the creator of the Kotlin. While we prioritize a function, we can easily suspend other function with coroutines in our Orchard Management system remote data calls.

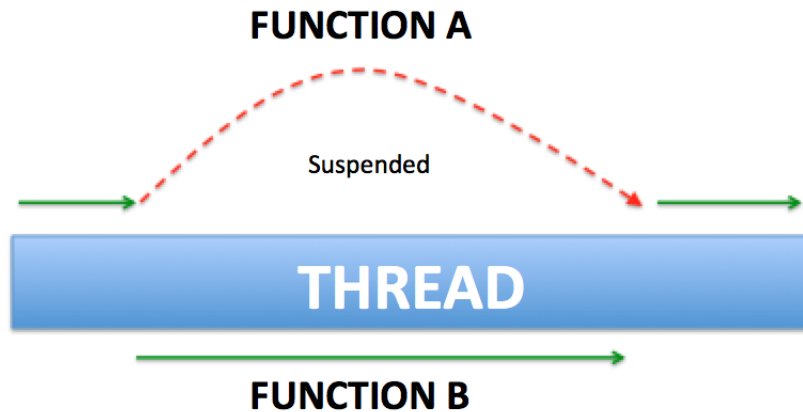


Figure 19: Thread structure

Retrofit (HTTP Client)

Retrofit is an HTTP client (a type of **COTS**) that we'll be using in our app to complete all our service calls from android to the back-end. It will help us to retrieve our data in JSON or XML format. In our case, the specified format to be used will be the JSON (JavaScript Object Notation) format. Simply will turn our REST API which will be prepared by the Back-end team. And will be integrated into our project. In general, we'll have 4 methods to complete this calls (calls are also the main important concept of the Retrofit)

We can list this calls with request types that we will use on the android side as following;

@GET @POST @PUT @DELETE

When it comes to the detailed design that will be considered in our MVVM pattern architecture used the android app, the Repository module will include all remote and local models(data models) this models will be filled by the local or remote data retrieval operations. In our case, most of the operations will be with remote REST API and the reason we complete this call with the Retrofit, the retrofit will execute on the remote module, and it will direct the calls through our Web service classes. We can easily summaries this as shown in the Figure 20.

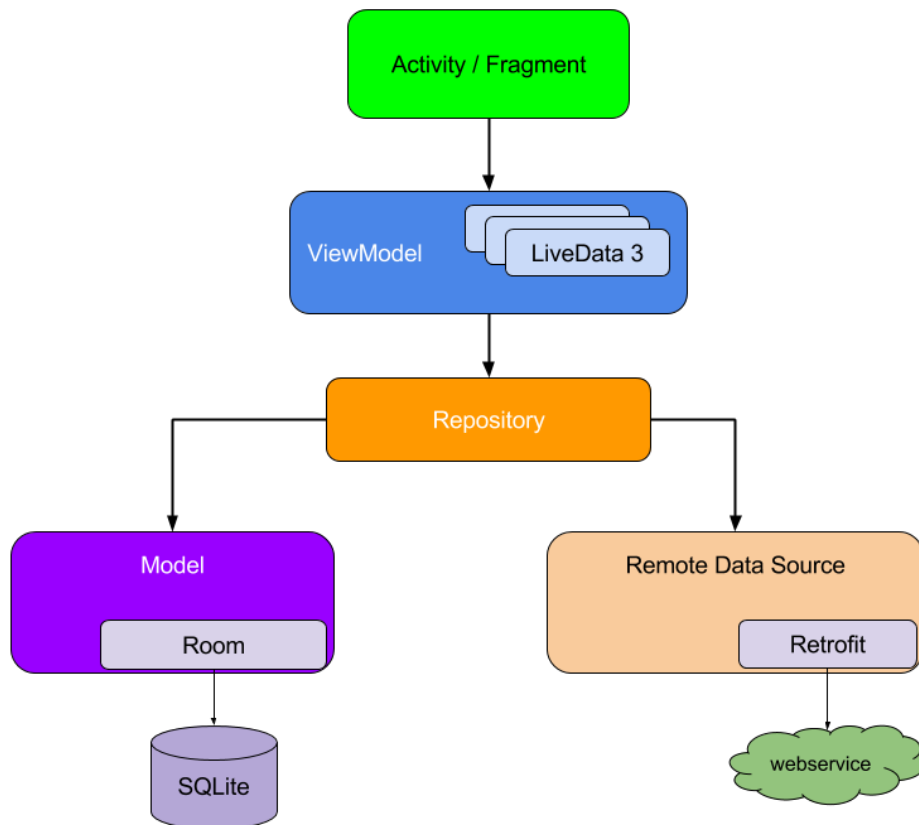


Figure 20: Networking structure

- Retrofit is under the Remote Data Source module.
- We can show a small snippet that explains how the data that we retrieve from the remote services (REST API) will look like. The data shown in the snippet is sample data from the Weather Rest API.

```

<  >  ↻  🔒 samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22
{
  "coord": {
    "lon": -0.13,
    "lat": 51.51
  },
  "weather": [
    {
      "id": 300,
      "main": "Drizzle",
      "description": "light intensity drizzle",
      "icon": "09d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 280.32,
    "pressure": 1012,
    "humidity": 81,
    "temp_min": 279.15,
    "temp_max": 281.15
  },
  "visibility": 10000,
  "wind": {
    "speed": 4.1,
    "deg": 80
  },
  "clouds": {
    "all": 90
  }
}
  
```

Figure 21: Networking structure

Kotlin (Dependency Injection)

The dependency between our classes is an important approach to take care of. The more classes and the higher amount of hierarchy we have in our project will result in more dependency coupling. This will cause our project to manage harder, do small modification with doing a lot of change. To get rid of this problem as much as we can, we will use Koin to decrease these dependencies and inject the dependencies we want easily.

We can easily show how we will implement this dependency injection in our MVVM design pattern used the application as shown in Figure 22.

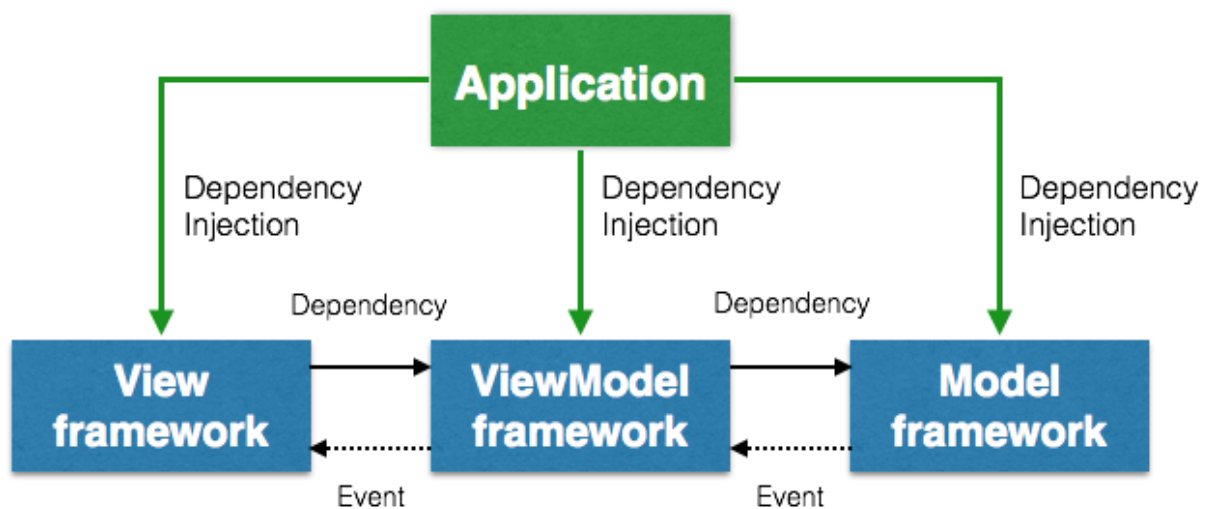


Figure 22: MVVM design pattern the application

To explain the above figure with koin, we will have our created modules and components that contain the ViewModel and Activity classes references and these references given to the Core class of the app. When the app started, the first thing to do will be creating instances of this these classes for once or multiple times depending on the usage. Then created instances will be injected to the classes that require this injection.

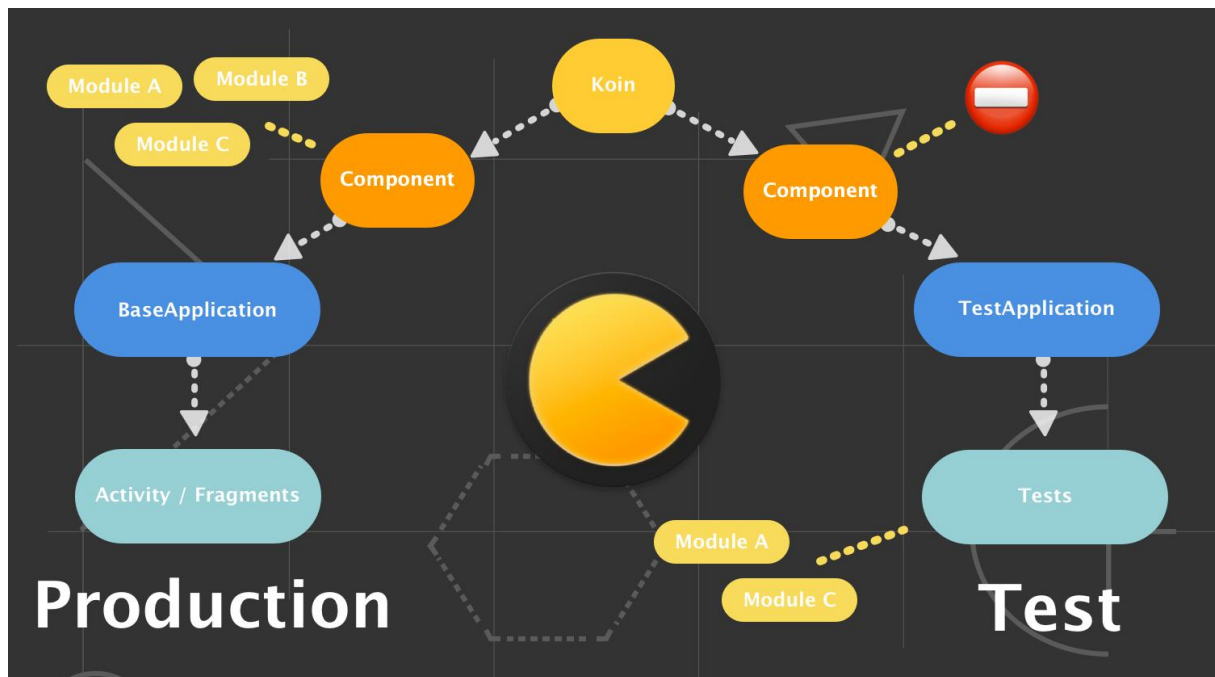


Figure 23:Test structure

6.IMPLEMANTATION

In this section, we describe what we have done for orchard management application including screenshots, detail information about each screen, flow and endpoints that we have used per each page individually.

6.1 Splash Screen

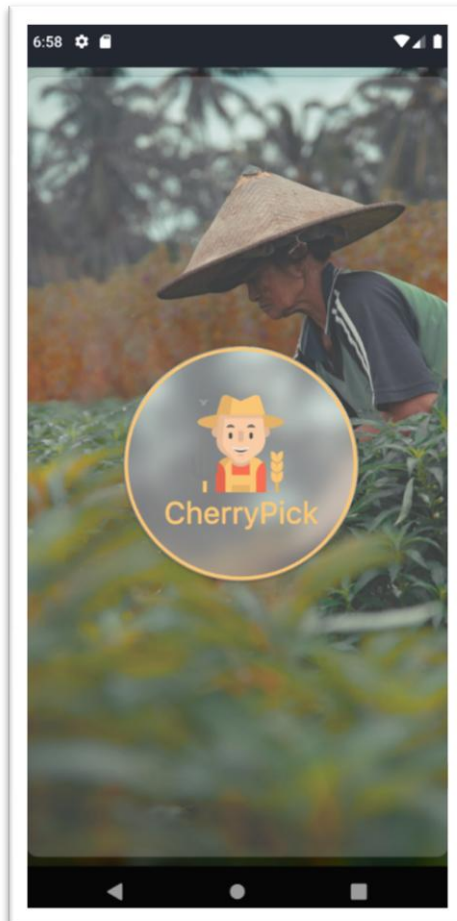


Figure 24: Splash Screen For App loading

The first page that meet with users is splash screen, opens during application loading state in the android application. Basically, we have some blur views, images etc. in this page. Also, there is no any request in this page. User might be directed to four possible pages, authentication page, orchard initialization page and main page.

If user has already logged in, then application direct to main page or orchard initialization page depending on number of orchards that user have. If user haven't initialize any orchard, then orchard initialization pages will be shown. Otherwise, application will be directed to onboarding pages or register page.

6.2 Onboarding

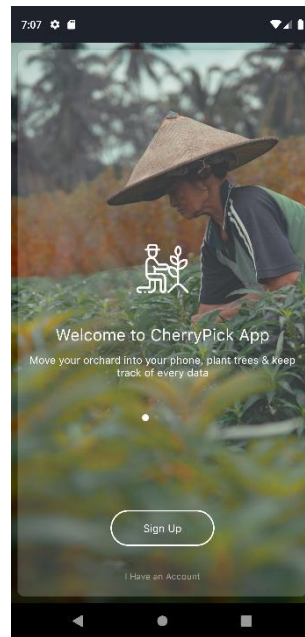


Figure 25:Onboarding Screen-1

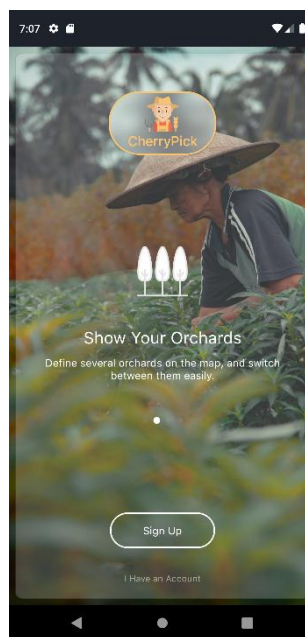


Figure 26:Onboarding Screen-2



Figure 27:Onboarding Screen-3

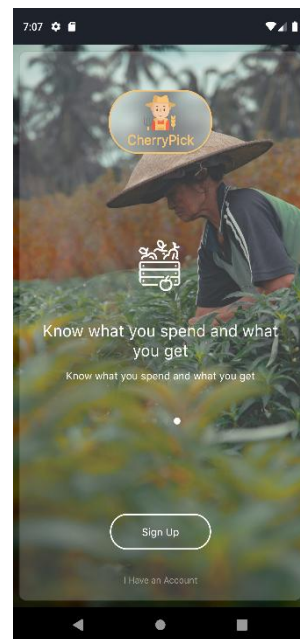


Figure 28:Onboarding Screen-4

In orchard management application, there are four onboarding pages that introduces that app for first installation case. User can move in each page with left to right scrolling easily. Also, user can review the current page via page control with white tint color. In this first page, we briefly show what orchard management application does. In the second page, there are some text about orchards and managing. In the third page, we state that our one of the best feature, managing orchard via harvest periods. Finally, we mentioned the statistics.

There are two possible navigation from onboarding pages. User can be directed to sign up or sign in depending on what user want. These navigations are provided by two buttons in the bottom of the each onboarding pages, means that user can press buttons in any onboarding page. Also, there is no request in onboarding pages.

6.3 Sing Up

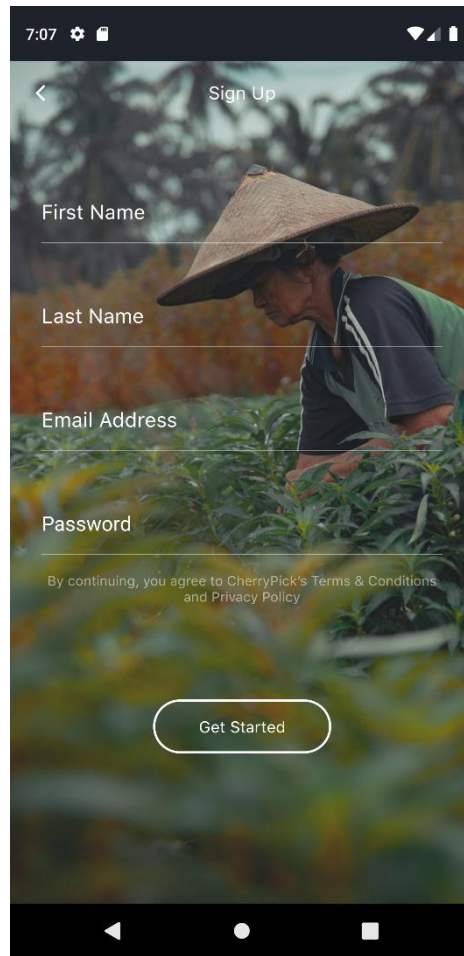


Figure 29: Register Page

User who has already account will be directed to login page. In this page, we have two text fields, email field and password field.

We send request to auth/login endpoint after user presses the login button in the page. In this stage, we also show a loader in the application while waiting state for response. If the application receives success block, then navigates through main application. Otherwise, the app present alert about error that received from API.

Also, there is a button for 'Forget Password' in the page that navigates user through forget password pages.

6.4 Forget Password

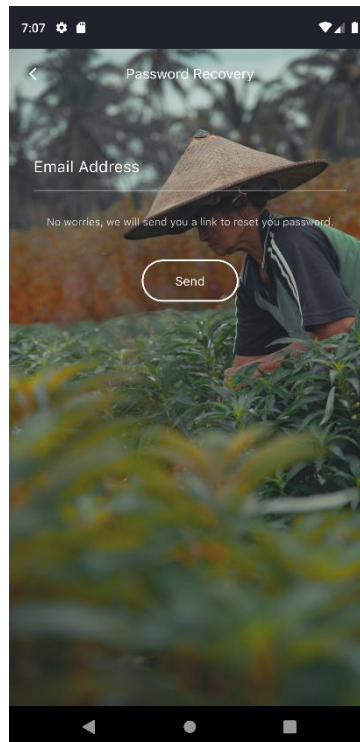


Figure 30: Forget Password First Stage

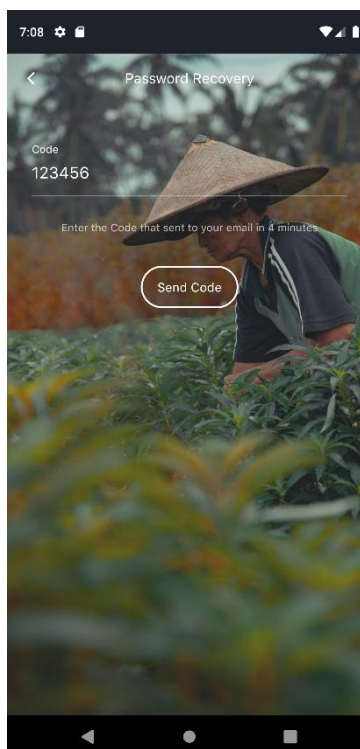


Figure 31: Forget Password Second Stage

There is a complex structure in forget password pages to provide more secure processes. In first stage, an email address is requested from user and sending request to 'auth/forgetPassword' endpoint with given email. During this process, API checks if there is an account with given email. If so, a dummy password is sent to user's email address. Then, user navigates through code state in the page. If user type his dummy code in 4 minutes without any mistyping, user will be directed to new password state. Also, in the case of user type invalid code, the API counts the number of invalid codes and disable the process if total number of invalid cases is equal to 3, then invalidates the forget password process. In this stage, the app send request to 'auth/checkForgetPassword'. In the last stage of forget password, the new password is requested from password and send request to 'auth/setForgetPassword' with new password. Finally, if all steps are succeeded, use will be navigated through login page again.

If there is any error in any step, an alert will be presented in the app and all process will be terminated.

6.6 Init Orchard



Figure 32:Orchard Initilize Page

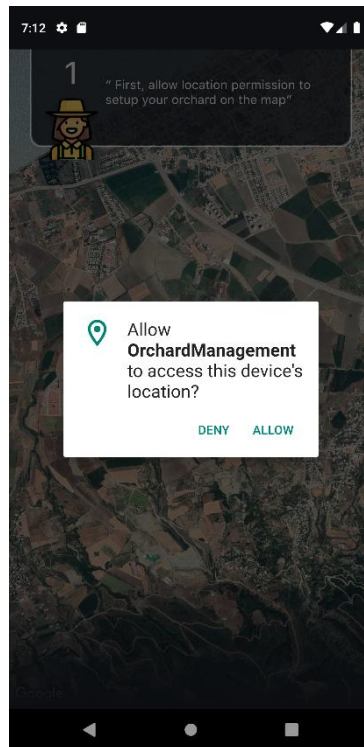


Figure 33:Location Permission View

If user has logged in and has no orchard, then user will be navigated through orchard initialization pages. In the first step, the app meet with user via Google Maps. We basically use Google Maps in anyplace that we need to use. Aim of this page is to request a permission for location. In the top of the page, there are some tips about each page. For instance, in this page, our app assistant mentions that location permission is mandatory to continue. Also, there is no required request so far, but at some point there will be some requests for init orchard. User will be navigated through orchard selection as soon as location is reachable.

6.6.1 Orchard Selection



Figure 34:Location Zoom in



Figure 35:Orchard Location Specifacition

In this page, map will focus on user's location and required to put markers in order to specify exact coordinates for his orchard. User can add unlimited number of markers. In order to accomplish orchard initialization, user must press 'complete' button. This button creates border and concatenates finish and end points immediately. So far, there is no required database process, all points are stored temporarily after some stage. We have also used Google Maps in this page. As soon as, orchard initialized, the user will be navigated through tree adding page.

6.6.2 Tree Add



Figure 36: Adding Tree on GoogleMaps



Figure 37: Adding Tree Completion

In order to accomplish orchard initialization, user must add a tree as well by either drag-drop or location. In location selection, tree will appear in user's current position in the map. This is very useful when user want to reach more accurate positioning in the orchard. In drag-drop option, user can move given tree in anywhere over orchard. As soon as user press complete button, orchard initialization process will be successful.

In this page there is a crucial request to *'orchard/initOrchard'* that takes position of orchard and tree and creates orchard with default tree type and accounting statements. During that request, a loading view will show up to the user. Then, user will be directed to main pages.

6.7 Login Screens

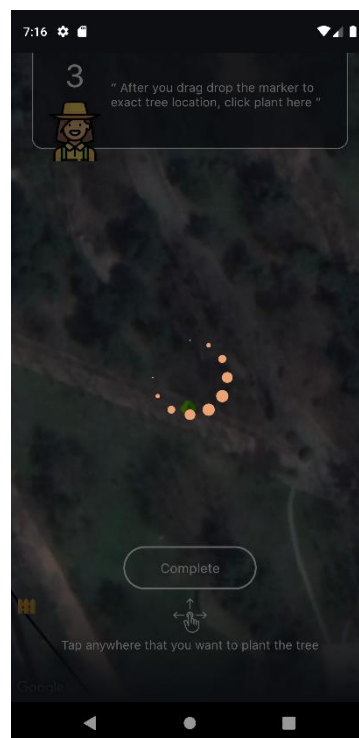


Figure 38:Loader View

In orchard management application, there are a lot of request that might take a while to get response. At this point, user see a loader page that we use most of the pages.

6.8 Specifying Tree Parameters

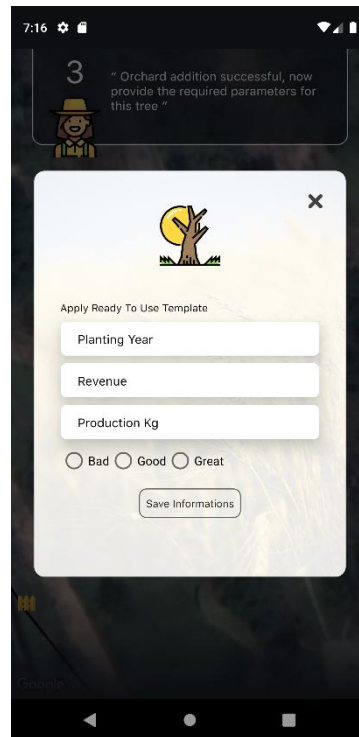


Figure 39:Updating Tree Parameter Pop-up

After a success initialization, there will be some requested parameters in the tree such as planting year, revenue, production in terms of kg, quality of the tree and so on. It's crucial to state them because there will be some statistics using those data. Also, user will be able to track tree considering its quality.

6.9 Home Page

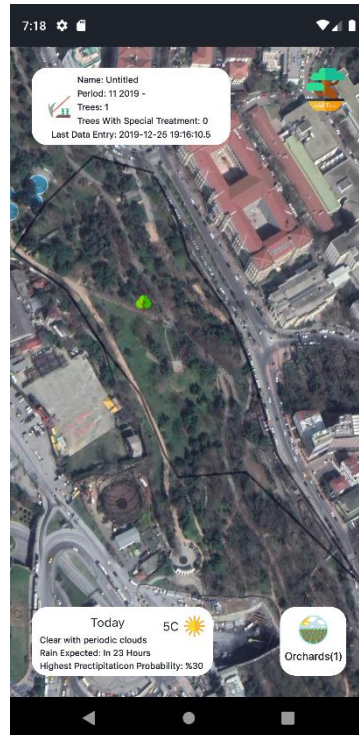


Figure 40:Home Page

In this page, user can see details of the orchard over Google Maps and some custom views. As you can see, in top left corner, there are some informations about orchard such as given name, period data, number of trees, last data entry etc. All this data provided by API via 'orchard/getAll' endpoint that returns every orchard, its sub harvest periods and sub trees of the periods as well. In the map, you can see the border of the orchard with black lines that has been specified in the orchard initialization process. Also, you can see the tree as well.

In the bottom-left corner, user can see detailed information about weather statistics on specified orchard that we get from weatherapi.com.

In the bottom-right corner, there is a button to switch orchards and also user can see the number of orchards he has.

In this page user can be navigated through tree update by click target tree from the map.

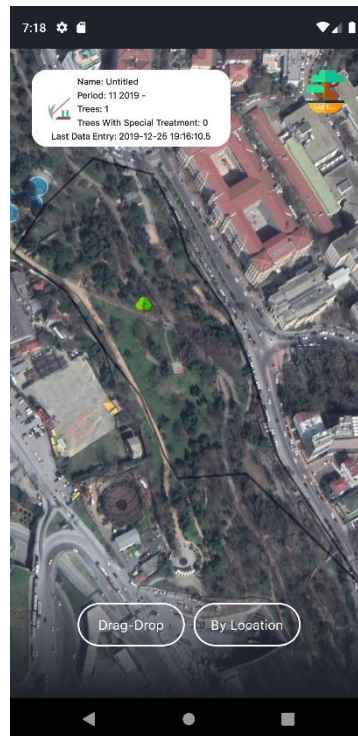


Figure 41: Adding New Tree

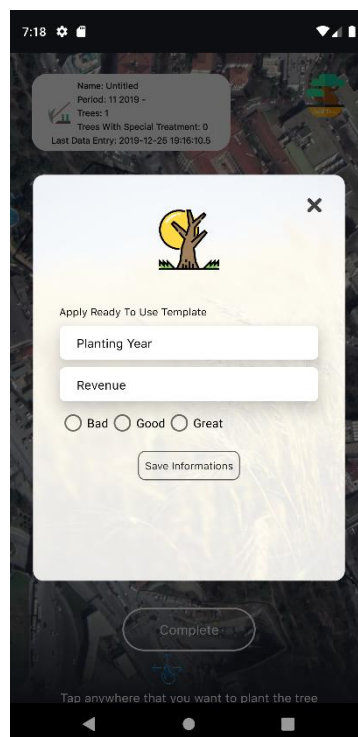


Figure 42: Updating Tree Parameters

In the top-right corner of the page, there is a button called add tree. In order to add a new tree for the orchard, user should click on it. As soon as user clicks the button, there will be two options drag-drop and by location. As we mentioned in orchard initialization process, user can add new tree by either drag-drop or location.

As soon as user specifies the location of the tree, user will be directed to tree details pop-up. In this pop-up, user should specify planting year and revenue. New tree get ready as soon as user fill all required fields and save. In this stage, app send request to “*tree/add*” endpoint with location parameters.

6.10 Update/ Delete Tree

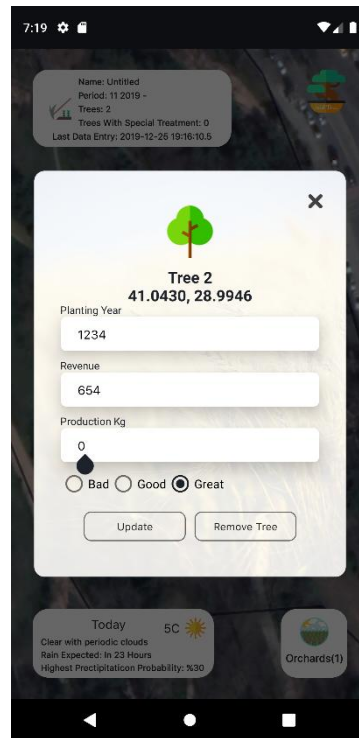


Figure 43:Update or Delete Tree Parameters

Users can update/delete tree by clicking tree on map. User can modify plating year, quality of the tree, revenue and production in terms of kg. The app send request to ‘*tree/update*’ endpoint to modify parameters.

6.11 Listing Orchards

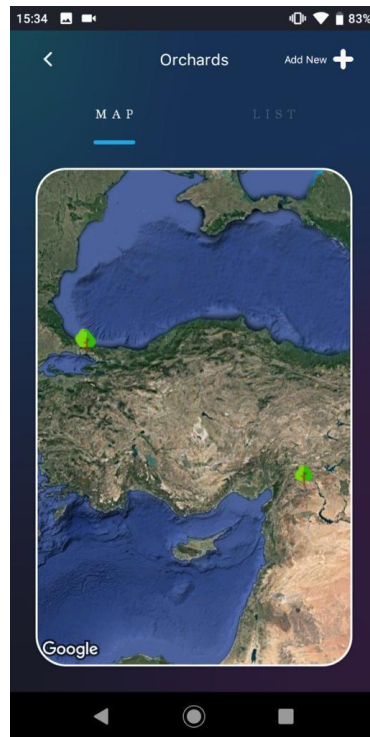


Figure 44: Listing Orchard Map

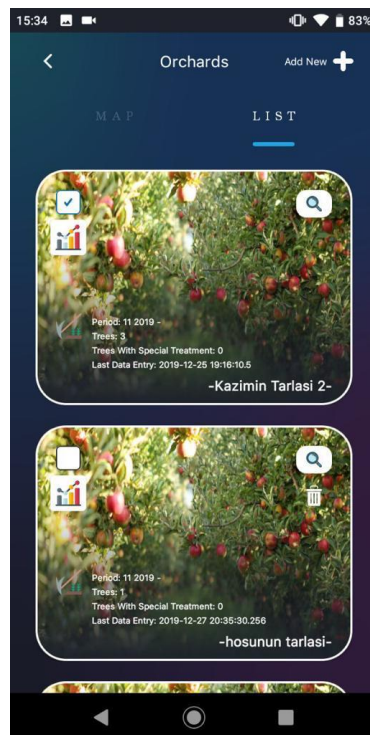


Figure 45: Listing Orchards List

In the app, users will be able to review their different orchards in two ways, either on map or on list. In this page, there is a side menu that user can switch through map or list. In map view, users can see their entire orchards on one map. Map automatically zooms to map considering location of orchards so that user can review all without scrolling.

In the list section, user will be able to see orchards in a list with detailed information such as number of trees, period dates, special treatments etc.

In each orchard cell, user can set main orchard that is current orchard that showed on the map in main page, can delete orchard if it's not main orchard, can see details and statistics from buttons in each cell.

Also, users can add new orchard in the top right corner. In this process, user will be directed through orchard initialization page.

There are several requests in this page. The app send request to 'orchard/getAll' to list each orchards in both map and listing side. In order to delete an orchard app send request to 'orchard/delete' with id of selected orchard.

6.12 Creating and Listing Harvest Periods

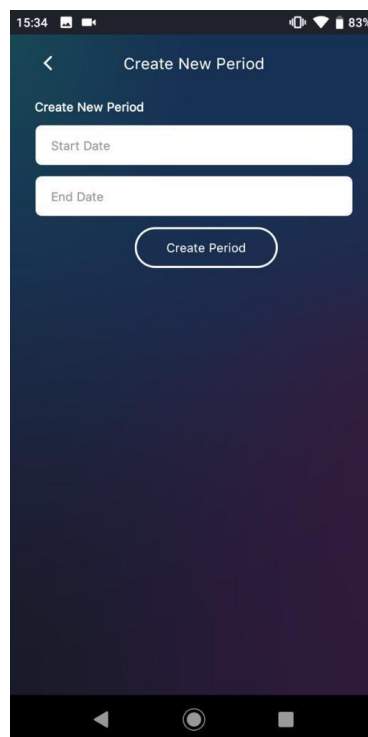


Figure 46:Create New Period

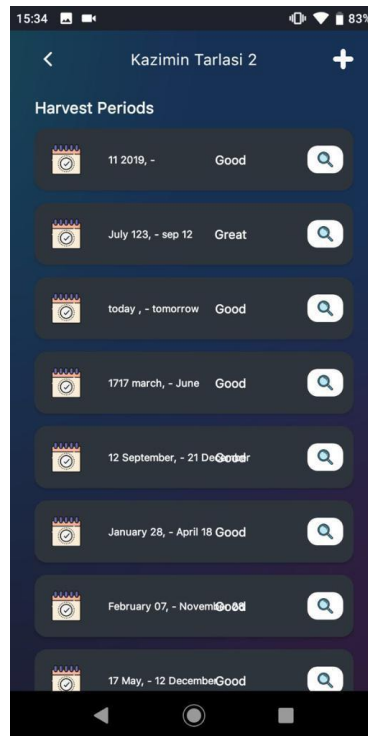


Figure 47:Listing Harvest Periods

This page will be reachable from Harvest's List page. In the right corner, Users will be able to add new harvest period with start and end dates. The app send request to 'harvestPeriod/add' endpoint. In this process, the API duplicates all trees that belongs to previous period and copy into new period. At this point, we classify the trees belongs to different periods with their timestamp. In the copy process, the API copy exactly the same timestamp value into the new tree. That's how new period added into the orchard.

Users will be able to list different harvest periods with their average quality. In the right of each period, there is a button that shows detail of each period. In this page, application sends request to 'orchard/getAll' to get entire orchard and its periods. In the right top corner, there is a button that navigates through create new period page.

6.13 Creating and Listing Expense/ Income For Periods

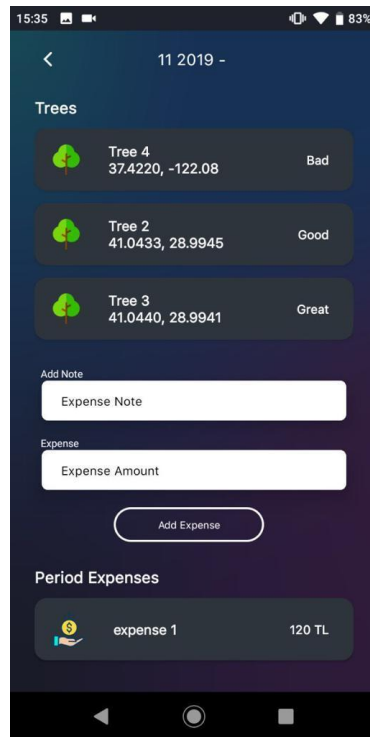


Figure 48:Creating Listing Expense

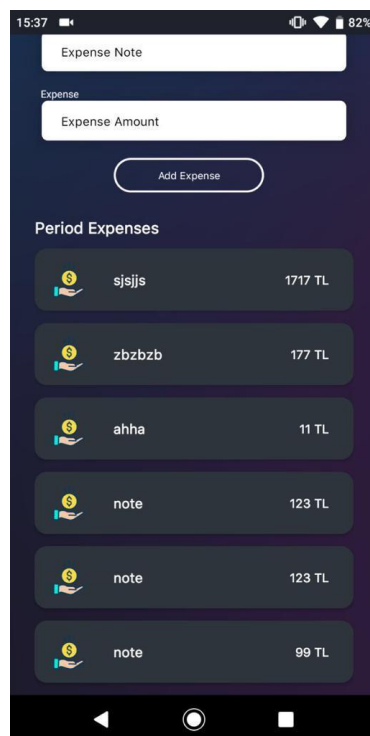
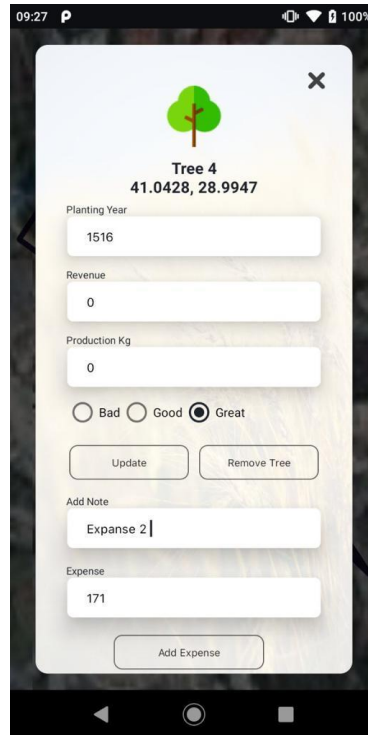


Figure 49:Listing Expense Period

User will be able to add and list income and expenses. In period detail page, user can see each tree with quality and add expense as well. In order to add new expense, a note and amount are required. As soon as user clicks ‘add expense’ button, the app send request to ‘accounting/addExpense’ endpoint and then, app send request ‘accounting/getPeriodExpenses’ endpoint with selected id in order to list entire expenses and incomes. As you can see in the period expenses section, user can see entire expenses.

6.14 Creating and Listing Expense / Income For Trees



The screenshot shows a mobile application interface for managing tree expenses. At the top, there's a header for 'Tree 4' with a green tree icon and a unique ID '41.0428, 28.9947'. Below this, the form contains several input fields: 'Planting Year' with the value '1516', 'Revenue' with '0', and 'Production Kg' with '0'. There are three radio buttons for quality: 'Bad', 'Good', and 'Great', with 'Great' being the selected option. Below the radio buttons are two buttons: 'Update' and 'Remove Tree'. Further down is an 'Add Note' section with a text input field containing 'Expense 2'. At the bottom, there's an 'Expense' field with the value '171' and an 'Add Expense' button. The background of the form is a blurred image of a tree.

Figure 50:Creating Expense For Trees

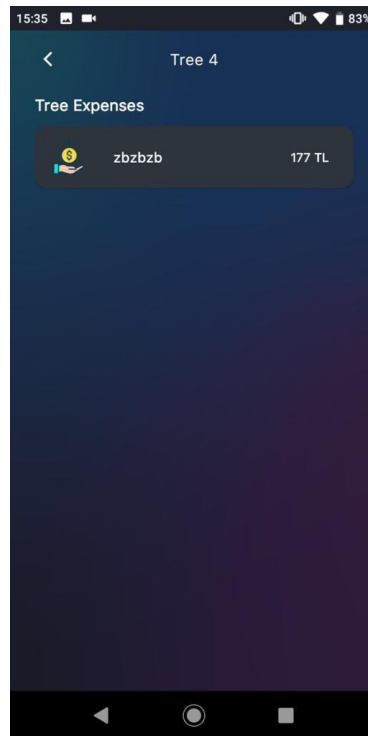


Figure 51:Listing Expense For Trees

User will be able to add and list income and expenses for their trees individually. In order to add expense to a tree, user should select the tree in maps page and eadd expense in the bottom of the page. As we have done in accounting of periods, app sends same request to 'accounting/addExpense' with only treeId. In this stage, expense of tree also added into current harvest period too.

In order to list tree expenses, user should select the period, then select individual tree from period detail and can see entire expenses from here.

6.15 Updating Harvest Period

 A mobile application screenshot showing a screen titled 'Update Period'. The screen contains a form with the following fields: 'Period Name' (containing 'July 123 - sep 12'), 'Start Date' (containing 'July 123'), 'End Date' (containing 'sep 12'), 'Revenue' (containing '1616'), and 'Production Kg' (containing '181'). Below these fields are three radio buttons labeled 'Bad', 'Good', and 'Great', with 'Great' being selected. At the bottom of the form is a button labeled 'Update Period'. The background is a dark blue gradient.

Figure 52:Updating Harvest Period

Users will be able to update each harvest period by selecting from harvest list page. In this page, user can change its name, dates, revenue and production in Kg as well. As soon as user presses the ‘update period’ button, app sends request to ‘harvestPeriod/update’ with selected period id and parameters in the page.

6.16 Statistics



Figure 53:Statistics



Figure 54: Statistics-2

Orchard owners can see detailed accounting statistics from statistics page which navigated from orchard lists. In this page, app sends request to 'orchard/getOrchardStats' endpoint with orchardId.

In the top of the page, you can see a line chart that shows total amount, incomes and expenses during the harvest periods. In the bottom of the page, you can see the datas that we have used in the charts as lists with harvest names as well.

7.RESULT AND CONCLUSION

As a conclusion, we had great experiences about how product development process is working for mobile applications with our orchard management application. Of course, there are some features that we have stated to implement, but couldn't implement because of limited time such as overall statistics, artificial intelligence based recommendations, ability to review orchards from all around the world as social media etc. But, we have built some great features such as orchard defining, creating harvest periods, creating trees for separate periods, income / expense tracking, statistics for accounting based on incomes and expenses etc.

We have got some experiences about team work besides the development processes. We believe that we have great task sharing considering member's abilities. For instance, one of us have focused on Android application more than application program interface etc.

Also, we have some great experiences about how to release an application considering long-term and application program interfaces. We have learned how to create proper Android application, how to prepare a good communication between client side and server side for better response performance.

Also, we have learned how to prepare documentation for software projects so that each of our team member will be able to create new large-scaled documentations for their own career.

8. GLOSSARY

Terms involved in this Project is showed in Table. Table hist of the Glossary.

Name	Description
Harvest Period	The time period for gathering products on the orchard or the farm
Management of Orchard	Handling, controlling the activities that are happening on the orchard lifecycle
Orchard Resources	The materials and the ingredients used on the field
Data	Information & resulted data that have been collected from the orchard (Amount of product, given water etc)
Transfer Data	Retrieving perspective of the data of the orchard
Critical Operations	Operations that are crucial for the owner to apply to his property in seasonly bases
User Interface	Point of human and computer interaction and communication in the system installed on Android Platform based devices
Operating Environment	The space/field that the project is planned to operating on, In our case Android Platform with Android operating system
Mockup	Type of modeling to show user interface with its boundaries
App Flow	Flowing on the app, detailing the directives of the app and system navigation
Database	Set of data and the way that we organize it the device in cases of local and remote
Authentication	Process to show and provide the functionality of the system to be developed
Login	Process to be logging the users into the application
Register	Process to save, create accounts for the user and give them access on the system
Tree Template	Ready to use templete that the user will create once and that makes it easy to apply for hund of trees
Drag Drop Tree Addition	Manuel tree addition with no exact location retrieval

9. REFERENCES

- [1], Orchardapp, 6/11/2019, <https://www.eorchardapp.com/>
- [2], Croptracker, 6/11/2019, <https://www.croptracker.com/product/orchard-management-software.html>
- [3], Hectre, 6/11/2019, <https://www.hectre.com/>
- [4], Cropdata, 6/11/2019, <https://www.cropdata.co.nz/>
- [5], MongoDB, 6/11/2019, <https://www.mongodb.com/>
- [6], Android, 6/11/2019, <https://developer.android.com/docs>
- [7], NodeJS, 6/11/2019, <https://nodejs.org/en/>
- [8], NPM, 6/11/2019, <https://www.npmjs.com/>
- [9], Visual Studio Code, 6/11/2019, <https://www.vscode.com/>