CTIS411 Senior Project I

Software Project Management Plan

Web-AR Smart Indoor Gardening System

Team 5

Mert Köroğlu, 22003433

Eren Tarak, 21903229

Talha Rehman Abid, 22001505

Eray Altay, 21803051

Bilkent University

Department of Information Systems and Technologies

6.01.2024

Change History

File Name	Document Type	Deliverable	Version	Submission Date
Team5_SPMP_FR1	MS Word 2013	3	1	17.12.2023
Team5_SPMP_FR2	MS Word 2013	3	2	6.01.2024

Change List:
- Removed some empty whitespaces for formatting.

- Inserted new paragraphs to 4. Project Monitoring and Measuring

- Edited the GANTT Chart.

- Edited the discussion part.

Project Team

Team 5		
Name, Surname	**Student Id Number**	**e-mail**
Mert Köroğlu	22003433	mert.koroglu@ug.bilkent.edu.tr
Eray Altay	21803051	eray.altay@ug.bilkent.edu.tr
Talha Rehman Abid	22001505	rehman.abid@ug.bilkent.edu.tr
Eren Tarak	21903229	eren.tarak@ug.bilkent.edu.tr

Project Details

Project Name	**Web-AR Smart Indoor Gardening System**
Software Name	**FloraVision**
Academic Advisor	**Prof. Dr. Duygu Albayrak**
Github URL	**https://github.com/Smart-Indoor-Gardening-System**
WEB page	

Executive Summary

In this document we provided our SPMP document that contains information regarding the cost estimations, effort estimations, scheduling and more. In this document we first talked about the scope of this SPMP document. Then, we calculated the effort needed by using 3 techniques: Use Case points estimation, WBS effort estimation and User Story estimation. After creating each one of these estimations, we provided insight into why there are differences between each one of these estimations. According to our estimations, we find many differences between the Use Case Points, User Story Points and WBS effort estimation. While the WBS estimation was 348.1 person hours, the User Story estimation was 1,080 hours and Use Case points were 1,606 hours. Later, we talked about our project schedule. In this section, we find out the total cost will be spent each month, and we calculate the Net Present Value which is 5,582.06TL**.** In this section we also provided the GANTT chart regarding the scheduling of various jobs and milestones like project increments 1,2 and 3. In this chart we also showed the duration of Hardware, Front-End, Back-End, Data Science and AR jobs and in which increment we will finish which jobs. Most importantly, in this section we provided our Work Breakdown Structure (WBS) which indicated each work package, the assignees for this work, the estimated person hours and dependencies of each package. In the Project Monitoring section, we talked about which monitoring tools will be used for the project, why we will use them, when we will use them and which metrics, we will get from them. We talked about how we will use Zoom and Trello for monitoring the work done. At the verification part, we talked about which verification techniques will be used while developing the product, which tools will be used to verify what. And the scheduling regarding the use of these tools is also provided. In the 6th Section, Software Development Environment, we described each development tool's description, their version number.

Table of Contents

Page Number

List of Tables

<u>Page Number</u>

List of Figures

Abbreviations

SPMP	Software Project Management Plan

1. Scope

The Software Project Management Plan (SPMP) outlines the comprehensive strategy for managing the development of FloraVision, a cloud based smart IoT system that provides seamless urban gardening experience through an easy-to-set-up, low-cost hardware kit. Featuring an IOT device with battery and sensors, this kit enables the device camera to become a portal to comprehensive statistics and visually engaging representations during plant scanning, thanks to Augmented Reality. This document encompasses the project management activities, methodologies, and processes that will be employed throughout the software development lifecycle.

Goals

- FloraVision aims to empower plant owners and indoor farmers by offering an intuitive tool for in-depth plant insights.
- Users can effortlessly monitor real-time metrics such as soil moisture, light intensity, air quality, and temperature/humidity.
- FloraVision provides concise daily, weekly, and monthly charts, offering a comprehensive overview of plant status.
- The platform goes beyond by delivering future health and growth predictions, ensuring users stay informed about their plant's well-being.

Objectives

- Project Monitoring and Measuring
- Finding Costs
- Effort Estimation
- Agile Estimation
- Software Development Environment
- Product Verification and Validation

Project Management Activities

The project management activities outlined in this SPMP include, but are not limited to:

Deliverables

Key deliverables from the project include:

- First increment submission at 8.12.2023
- Second increment to be submitted at end of the February
- Third increment to be submit at the end of the May

Methodology Highlights:

Scrum: The project will be managed using the Scrum framework, which involves organizing work into time-boxed iterations called sprints.

Exclusions:

This project management plan does not cover:

- Post-Deployment Support: Long-term support and maintenance activities beyond the initial deployment phase are not part of this plan.
- The plan excludes marketing and sales activities for promoting the FloraVision product to potential customers.
- Shipping and Logistics: Activities associated with the transportation and logistics of the hardware kit to end-users are not within the scope of this plan.
- Detailed user training programs for utilizing the FloraVision system are excluded.

Stakeholders

- Dr. Duygu Albayrak
- Dr. Oumout Chouseinoglou
- CTIS Department

2. Project Effort Estimation

In this section of the file, we estimated the effort with 3 perspectives: use case points, user story points and WBS packages. Then, we compared our efforts regarding these 3 effort findings and discussed the differences between each effort calculation.

1) WBS Effort Estimation

We provided our WBS at Section 3.1 Table 6 on this document with estimated person hours for each package. According to these packages the estimated per person hours is **348,1 person hours.**

2) Use Case Based Estimation:

Unadjusted Actor Weighting Table:

Table 1 Unadjusted Actor Weighting Table

Actor Type	**Actor**	**Weighting Factor**	**Number**	**Result**
Simple	Arduino	1	1	1 * 1 = 1
Average	[None]	2	0	2 * 0 = 0
Complex	User, Root User	3	2	3 * 2 = 6
			Unadjusted Actor Weight Total (UAW)	**Total = 7**

Unadjusted Use Case Weighting Table:

Table 2 Unadjusted Use Case Weighting Table

Use Case Type	**Use Cases**	**Weighting Factor**	**Number**	**Result**
Simple	● Read Sensors ● Send Data to AWS ● Login ● Register	5	4	5 * 4 = 20

Average	● Manage User Credentials ● Manage Device Credentials	10	2	10 * 2 = 20
Complex	● View Plants Conditions ● Connect to WiFi ● Use AR	15	3	15 * 3 = 45
Unadjusted Use Case Weight Total (UUCW)				**Total = 85**

According to these calculations, unadjusted use-case points can be found using the following formula:

UUCP = UAW + UUCW

UUCP = 7 + 85

UUCP = 92

Technical Complexity Factors Table:

Table 3 Technical Complexity Factors Table

Factor Number	Description	Weight	Assigned Value (0-5)	Weighted Value
T1	Distributed system	2.0	5	2.0 * 5 = 10
T2	Response time or throughout performance objectives	1.0	5	1.0 * 5 = 5
T3	End-user online efficiency	1.0	1	1.0 * 1 = 1

T4	Complex internal processing	1.0	4	1.0 * 4 = 4
T5	Reusability of code	1.0	1	1.0 * 1 = 1
T6	Ease of installation	0.5	3	0.5 * 3 = 1.5
T7	Ease of use	0.5	4	0.5 * 4 = 2
T8	Portability	2.0	2	2.0 * 2 = 4
T9	Ease of change	1.0	1	1.0 * 1 = 1
T10	Concurrency	1.0	1	1.0 * 1 = 1
T11	Special security objectives included	1.0	3	1.0 * 3 = 3
T12	Direct access to third parties	1.0	3	1.0 * 3 = 3
T13	Special user training required	1.0	5	1.0 * 5 = 5
Technical Factor Value (TFactor)				**Total = 41.5**

We can find the technical complexity factor (TCF) with the following formula:

TCF = 0.6 + (0.01 * TFactor)

TCF = 0.6 + (0.01 * 41.5)

TCF = 1.015

Environmental Factors Table:

Table 4 Environmental Factors Table

Factor Number	Description	Weight	Assigned Value (0-5)	Weighted Value

E1	Familiarity with system development process being used	1.5	3	1.5 * 3 = 4.5
E2	Application experience	0.5	2	0.5 * 2 = 1
E3	Object-oriente d experience	1.0	3	1.0 * 3 = 3
E4	Lead analyst capability	0.5	1	0.5 * 1 = 0.5
E5	Motivation	1.0	4	1.0 * 4 = 4
E6	Requirements stability	2.0	4	2.0 * 4 = 8
E7	Part time staff	-1.0	0	-1.0 * 0 = -1
E8	Difficulty of programming language	-1.0	2	-1.0 * 2 = -2
			Environmental Factor Value (TFactor)	**Total = 18**

We can find the environmental factor (EF) with the following formula:

EF = 1.4 + (-0.03 * EFactor)

EF = 1.4 + (-0.03 * 18)

EF = 0.86

Now we can find the adjusted use case points (UCP) with the following formula:

UCP = UUCF * TCF * EF

UCP = 92 * 1.015 * 0.86

UCP = 80.30

Now that we find the UCP now we can calculate the effort:

Effort (in person hours) = UCP * PHM

Effort (in person hours) = 80.30 * 20

Effort = 1.606 person hours

3) Agile Estimation:

For agile estimation, we provided risk, complexity and repetition factors ranging from (1-5) and for the story points we used Fibonacci Sequence to assign the values.

Table 5 Agile Estimation Table

No	User Story	RSK	CPLX	RPT	SP
1	As a user, I want to configure the hardware kit easily, so I can start monitoring my plants	1	2	2	5
2	As a user, I want to be able to power on and off the hardware kit conveniently.	1	2		3
3	As a user, I want the system to accurately read and display data from various sensors such as light, humidity, temperature, and soil moisture, so I can monitor the environmental conditions of my plants.	3	5	5	8
4	As a user, I want the system to seamlessly transmit sensor data to AWS, ensuring that I receive real-time updates on my plant's conditions.	2	3	2	8
5	As a user, I want the web application to easily connect to and	4	4	2	8

	display information from the hardware kit, making it convenient for me to monitor my plants remotely.				
6	As a user, I want a secure authentication process for connecting to the hardware kit, ensuring that only authorized users can access and control the device.	2	3	3	5
7	As a user, I want a secure and user-friendly authentication process for accessing the web application, providing options such as email, Google, Facebook, and password authentication.	2	2	5	5
8	As a new user, I want a straightforward registration process with clear validation rules for email/password, ensuring a seamless onboarding experience.	2	2	5	5
9	As a user, I want a convenient login process, including options for password recovery, to ensure easy access to the web application.	3	2	5	5
10	As a user, I want to easily view and manage connected hardware devices through the web application, making it simple to monitor multiple plants.	2	4	3	8

11	As a user, I want a visually appealing and informative dashboard displaying real-time and historical sensor data, allowing me to analyze trends and make informed decisions about my plants.	3	3	3	8
12	As a user, I want to receive timely and relevant notifications about my plants, ensuring that I stay informed about any issues or updates.	2	3	3	8
13	As a user, I want the option to visualize my plant data in augmented reality, providing an innovative and immersive experience.	5	5	1	13
14	As a user, I want the system to utilize predictive analysis to forecast potential health issues for my plants, helping me take proactive measures.	2	5	2	13
15	As a root user, I want to have control over device names, passwords, and user management, ensuring efficient administration of the system.	2	2	4	3
16	As a user, I want to be able to trace the health and growth predictions of my plants with synced charts which help me to see the different predictions of my plants and gain comparison insights at a single time point	3	4	3	8

17	As a system admin user, I want to be able to see system's status on the web app from the admin dedicated endpoint	3	3	3	5
18	As a plant enthusiast (12+ years old), I want to easily view real-time insights about my indoor plants, so I can ensure they are healthy and thriving.	2	4	4	8
19	As an indoor farmer, I want to connect multiple IoT hardware devices to monitor my crops comprehensively, ensuring they receive the optimal environmental conditions.	2	4	4	8
20	As a normal user, I want to log in securely to access the web application and receive personalized insights for my connected device.	2	4	4	5
21	As a user with an indoor plant, I want to receive real-time notifications about my plant's health and growth, ensuring timely intervention if needed.	2	4	4	5
22	As a user, I want the web application to connect to hardware kits through the devices tab, making it easy to manage connected devices.	2	3	4	5
23	As a user, I want to register and log in to the web application securely	2	4	2	5

	using AWS Cognito, ensuring data privacy and user authentication.				
24	As a user, I want to customize my dashboard by selecting specific charts to display, allowing a personalized monitoring experience.	2	4	3	8
25	As a user, I want to view a notification modal with the latest notifications and mark them as read, ensuring I am aware of important updates.	2	4	4	5
26	As a user, I want to visualize real-time environmental data using AR features, enhancing my understanding of my plant's conditions.	5	5	1	13
27	As a user, I want the system to integrate a health prediction model, providing short-term forecasts for potential plant health issues.	3	5	2	13
28	As a user, I want the web application to be compatible with major browsers (Chrome, Firefox, Safari), ensuring a consistent experience across different platforms.	1	2	5	3
29	As a system admin user, I want the system to handle a minimum of 10 simultaneous users, ensuring optimal performance during peak usage times.	1	1	5	3
30	As a user, I want the system to hold the device's data history for at least	1	2	3	3

	1 month, allowing me to review historical trends and patterns.				
31	As a system admin user, I want the system to log every system failure, providing transparency and traceability in case of security incidents.	1	2	3	3
32	As a user, I want to log in from other devices that are compatible with the web product, ensuring flexibility in accessing the application.	1	2	4	3
33	As a user, I want the code for each hardware device to be specific to that device, ensuring security and individualized data tracking.	1	2	4	3
34	As a user, I want the product to adhere to privacy laws and rules about data sharing, keeping, and handling, ensuring legal compliance.	1	4	3	8
35	As a user, I want the system to provide error warnings for incorrect login credentials, ensuring a secure authentication process.	2	2	5	3
36	As a user, I want the system to warn me if I provide a non-existing or already in-use email while registering, preventing registration issues.	2	2	5	3
37	As a user, I want the system to warn me if I provide a password that	2	2	5	3

	does not meet the specified complexity rules during registration.				
38	As a user, I want the system to ignore incorrectly formatted messages from the hardware product and provide a log about the ignored message, ensuring data integrity.	2	2	5	5
39	As a user, I want the system to warn and redirect me to the login page if I try to access unauthorized locations in the web application, ensuring secure access.	2	2	5	3
40	As a user, I want the system to warn me if I enter and provide two different passwords while registering, preventing registration issues.	2	2	5	3
41	As a user, I want to receive alerts if a connected device goes offline, ensuring continuous monitoring.	2	3	3	3
42	As a user, I want to compare the current plant conditions with optimal values, facilitating better decision-making.	2	3	3	5
43	As a user, I want the ability to recover my account in case I forget my password, enhancing account accessibility	2	4	2	3
44	As a user, I want to generate and download reports summarizing the health and growth of my plants	2	3	2	8

45	As a user, I want anomalies to be categorized into severity levels (low, medium, high) based on the potential impact, prioritizing urgent issues.	2	3	2	8

Our total User Stories are **263** Story Points

Each sprint (1 week time) we will finish **15** Story Points each sprint.

So, we estimated that we would finish the project in **18 weeks.**

We planned that everyone would work **15 hours a week.** So based on this estimation we can find Effort (in person hours) using the following formula:

Effort (in person hours) = (TeamMembers * HoursPerWeek) * Weeks

Effort (in person hours) = (4 * 15) * 18

Effort (in person hours) = 60 * 18

Effort (in person hours) = 1,080

Discussion:

We found 3 different effort calculations.

- For WBS it is 348,1 person hours
- For Use Case Points it is 1,606 person hours
- For Story points it is 1,080 person hours

Because the WBS is relatively lower than both Use Case and Story Points estimation, we think that we made a mistake predicting how much each package would take. For Use Case and Story points, we still have a big difference between values, but they are still close, we can say that we might not have covered all User Stories for our system, and we could have generated more.

3. Project Schedule

In this section, we talked about the work packages, their dependencies, our cost estimations, and the overall schedule of the project.

1) Work Packages

Our WBS and corresponding work packages are shown below. On Table 6.

Table 6 FloraVision Work Breakdown Structure

Work Package ID	WBS Package Description	Dependencies	Assignee	Person Hours
1	**Hardware**			
1.1	**Connect Sensors to ports**		Mert Köroğlu	1
1.2	**Enable WiFi connection**		Mert Köroğlu	3
1.3	**Configure WiFiManager Library**	1.2	Mert Köroğlu	2
1.4	**Create connection with AWS IoT Core**	1.2	Mert Köroğlu	1
1.5	**Read Sensor inputs**	1.1	Mert Köroğlu	1
1.6	**Create JSON of inputs**	1.5	Mert Köroğlu	1
1.7	**Send JSON to AWS**		Mert Köroğlu	2
1.8	**Device creation on AWS**		Mert Köroğlu	2
1.9	**Test the hardware**		Mert Köroğlu	4
2	**Frontend**			
2.1	**Create a React.js V18.2 project for frontend**		Eren Tarak	0,1

2.2	**Configure Frontend Build Tool Vite**		Eren Tarak	0,2
2.3	**Dockerize Frontend**	2.1	Eren Tarak	0,1
2.4	**Configure Linting Utility Eslint**		Eren Tarak	0,5
2.5	**Set up Chakra UI Component Library**	2.1	Eren Tarak	0,2
2.6	**Configure a Customized Light and Dark Theme**		Eren Tarak	0,2
2.7	**Organize Frontend Project Folder Structure**		Eren Tarak	0,8
2.8	**Design "Create device/dashboard" UI**		Eren Tarak	0,6
2.10	**Design Login UI**		Eray Altay	4
2.11	**Design User Registration UI**		Eray Altay	4
2.12	**Design Dashboard UI**		Eray Altay	4
2.13	**Design App Navigation UI**		Eray Altay	4
2.14	**Setup Zustand Global State Management Library**		Eray Altay	2
2.15	**Setup React-Query Async State Management**		Eren Tarak	0,2
2.16	**Display Anomalies on Web Dashboard**	2.9	Eren Tarak	0,4
2.17	**Display Connected Devices**	2.9	Eray Altay	3
2.18	**Display Plant Statistics**	2.12	Eray Altay	3
2.19	**Provide a button and functionality to add new device**	2.9	Eray Altay	6
2.20	**Enable users to remove a selected device.**	2.9	Eray Altay	4
2.21	**Display battery status of each connected device**		Eray Altay	1

2.22	**Display tabs for daily, weekly, monthly, and yearly options**		Eray Altay	1
2.23	**Display charts for today's sensor data from 00:00 to 23:59.**		Eray Altay	3
2.24	**Enable users to customize the dashboard.**		Eray Altay	3
2.25	**Display deviations from optimum values with labels.**		Eray Altay	4
2.26	**Present average values on daily, monthly, and yearly basis.**		Eray Altay	2
2.27	**Design notification user interface modal.**		Mert Köroğlu	4
2.28	**Display notifications when the bell icon is clicked**		Mert Köroğlu	2
2.29	**Design a notifications page for all notifications.**		Mert Köroğlu	4
2.30	**Enable users to mark all notifications as read.**		Mert Köroğlu	1
2.31	**Enable real-time web push notifications.**		Mert Köroğlu	3
3	**AWS CDK Infrastructure Setup**			
3.1	**Set up a new AWS CDK project using Node.js TypeScript**		Eren Tarak	2
3.2	**Configure project dependencies.**		Eren Tarak	2
3.3	**Define stacks for Lambda functions, API Gateway, SNS, Cognito, DynamoDB, and IoT Core.**		Eren Tarak	2
4	**AWS Lambda Functions**			
4.1	**Implement Lambda function for user registration**		Eren Tarak	4
4.2	**Implement Lambda function for user login**		Eren Tarak	3

4.3	**Implement Lambda function for password recovery**		Eren Tarak	2
4.4	**Implement Lambda function for hardware kit authentication**		Eren Tarak	4
4.5	**Implement Lambda function for adding new devices.**		Eren Tarak	4
4.6	**Implement Lambda function for removing devices**		Eren Tarak	2
4.7	**Implement Lambda function for processing sensor data.**		Eren Tarak	4
4.8	**Implement Lambda function for generating health prediction**		Eren Tarak	7
4.9	**Implement Lambda function for sending real-time notifications.**		Eren Tarak	2
4.10	**Implement Lambda function for handling push notifications.**		Eren Tarak	3
5	**AWS API Gateway**			
5.1	**Define RESTful API endpoints for user authentication.**		Eren Tarak	2
5.2	**Define API endpoints for hardware kit authentication**		Eren Tarak	2
5.3	**Define API endpoints for sensor data retrieval.**		Eren Tarak	2
5.4	**Configure API Gateway to trigger Lambda functions.**		Eren Tarak	3
5.5	**Implement security measures like API key or Cognito**		Eren Tarak	4
6	**AWS Simple Notification Service (SNS)**			
6.1	**Create SNS topics for real-time notifications**		Eren Tarak	3
6.2	**Configure SNS topics for user subscriptions**	6.1	Eren Tarak	3

6.3	**Configure Lambda functions to publish notifications to SNS**	6.1	Eren Tarak	2
6.4	**Implement error handling for SNS notifications**		Eren Tarak	2
7	**AWS Cognito**			
7.1	**Integrate AWS Cognito for user registration and login**		Eren Tarak	3
7.2	**Configure Cognito user pool settings**		Eren Tarak	2
7.3	**Define user roles for Root user and Normal user**		Eren Tarak	1
7.4	**Implement Lambda function for assigning user roles**	7.3	Eren Tarak	4
8	**AWS DynamoDB**			
8.1	**Implement AWS CDK constructs for all tables.**		Eren Tarak	2
8.2	**Configure partition keys, sort keys, and attributes**		Eren Tarak	2
9	**AWS Iot Core**			
9.1	**Set up IoT Core to manage communication with hardware devices**		Mert Köroğlu	4
9.2	**Implement security measures such as device certificate**		Mert Köroğlu	4
9.3	**Define MQTT topics for communication with hardware devices**		Mert Köroğlu	1
9.4	**Configure IoT Core to trigger Lambda functions based on incoming messages**		Mert Köroğlu	5
9.5	**Implement Lambda functions for processing messages from IoT Core**		Mert Köroğlu	2

9.6	**Ensure Lambda functions update DynamoDB tables with sensor data**		Mert Köroğlu	2
10	**Error Handling**			
10.1	**Implement Error handling for incorrect login credentials**		Eray Altay	2
10.2	**Handle Errors for existing or already in use email during sign up**		Eray Altay	1
10.3	**Warn Users for non required password types during sign up**		Eray Altay	1
10.4	**Implement error handling for incorrectly formatted messages from the hardware product**		Eray Altay	2
10.5	**Warn and redirect users attempting to unauthorized locations**		Eray Altay	3
11	**Security Measures**			
11.1	**Implement a logging system to log every system failure**		Mert Köroğlu	5
11.2	**Store logs securely and enable log analysis**		Mert Köroğlu	3
11.3	**Implement privacy measures to adhere to privacy laws**		Mert Köroğlu	5
12	**Usability and Compatibility**			
12.1	**Implement Compatibility for Major browsers**		Eray Altay	4
12.2	**Ensure Consistent Experience across platforms**	12.1	Eray Altay	3
13	**Performance Optimization**			
13.1	**Optimize Lambda Functions and API Gateway for handling minimum 10 simultaneous user**		Eren Tarak	5

13.2	**Implement horizontal scaling strategies**		Eren Tarak	4
13.3	**Design Data Storage Architecture to hold device data history for at least 1 month**		Eren Tarak	2
13.4	**Optimize data retrival for historical data**		Eren Tarak	3
13.5	**Setup Elastic APM to monitor and trace the lamda functions performance at app code level**		Eren Tarak	3
13.6	**Integrate Elastic APM agents into lambda function for distributed tracing**		Eren Tarak	2
13.7	**Set up CloudWatch Alarms to monitor infrastructure-level metrics**		Eren Tarak	2
14	**Real time system administration**			
14.1	**Identify Critical Failure Conditions that should trigger notifications**		Mert Köroğlu	3
14.2	**Create an SNS Topic for system failures**		Mert Köroğlu	2
14.3	**Implement related lambda function to send notification to system admin**		Mert Köroğlu	3
14.4	**Design and API Gateway endpoint to check system status or notifications**		Mert Köroğlu	3
15	**Scientefic Data Collection**			
15.1	**Define Data Requirements**		Talha Abid	3
15.2	**Identify Data Sources**		Talha Abid	2
15.3	**Collect Primary Data**	15.2	Talha Abid	3
15.4	**Acquire Secondary Data**	15.2	Eray Altay	2
15.5	**Verify Data Accuracy**	15.3, 15.4	Talha Abid	3

16	**Data Preprocessing**			
16.1	**Cleanse Raw Data**		Talha Abid	5
16.2	**Handle Missing Values**		Talha Abid	3
16.3	**Standardize Data Formats**		Talha Abid	1
16.4	**Remove Duplicates**		Talha Abid	2
16.5	**Transform Data**		Talha Abid	2
17	**Data Integration**			
17.1	**Combine Different Data Sources**		Talha Abid	5
17.2	**Resolve Data Conflicts**		Talha Abid	3
17.3	**Document Data Collection Procedures**		Talha Abid	2
18	**Health and Growth Prediction Models Development (Local Part)**			
18.1	**Define Bayesian Neural Network Architecture**		Talha Abid	7
18.2	**Choose TensorFlow Probability or Pyro for Implementation**		Talha Abid	3
18.3	**Split Data into Training and Validation Sets**		Talha Abid	1
18.4	**Train Bayesian Neural Network**	18.3	Talha Abid	1
18.5	**Utilize Bayesian Optimization or Grid Search for Hyperparameter Tuning**		Talha Abid	1
18.6	**Evaluate Model Performance on Validation Set**		Talha Abid	0,1
18.7	**Adjust Model or Hyperparameters if Needed**		Talha Abid	0,1
18.8	**Save Trained Model in a Compatible Format**		Talha Abid	0,1

19	**Health and Growth Prediction Models Cloud Deployment (AWS SageMaker Part)**			
19.1	**Upload Model File to an S3 Bucket**		Eren Tarak	0,5
19.2	**Create Docker Container**		Eren Tarak	1
19.3	**Include Necessary Dependencies**		Eren Tarak	1
19.4	**Implement Inference Code**		Eren Tarak	1
19.5	**Use SageMaker SDK to Deploy Model as an Endpoint**		Eren Tarak	1
19.6	**Specify Instance Type and Number of Instances for Deployment**		Eren Tarak	1
19.7	**Set Up API Gateway**		Eren Tarak	1
19.8	**Grant SageMaker Permissions to Access Resources**		Eren Tarak	1
20	**Real Time Anomaly Detection**			
20.1	**Choose A Common algorithm for real-time anomaly detection in IoT systems**		Talha Abid	2
20.2	**Define Anomaly Types and Criteria**		Talha Abid	1
20.3	**Design Data Flow Architecture**		Talha Abid	5
20.4	**Implement Lambda Functions for Data Ingestion**		Eren Tarak	4
20.5	**Identify Relevant Features for Anomaly Detection**		Talha Abid	4
21	**AR**			
21.1	**Research AR libraries**		Mert Köroğlu	3
21.2	**Select AR library**	21.1	Mert Köroğlu	1

21.3	**Research QR Code scanning technologies**		Mert Köroğlu	3
21.4	**Create QR Code scanning code**	21.4	Mert Köroğlu	3
21.5	**Test QR Code scanner**	21.5	Mert Köroğlu	1
21.6	**Link QR Codes to Device ID**		Mert Köroğlu	2
21.7	**Get variables for plant from QR code**	21.6	Mert Köroğlu	3
21.8	**Display AR content as 2D charts and metrics with icons**		Mert Köroğlu	4
21.9	**Support visualization of historical environmental data in AR**		Mert Köroğlu	3
21.10	**Display alerts for unsupported devices or browsers.**		Eray Altay	3
22	**Frontend Testing**			
22.1	**Implement Frontend End to End testing using Cypress**		Eren Tarak	2
22.2	**Implement Frontend Integration Test using Cypress**		Eray Altay	2
23	**Backend Testing**			
23.1	**Implement lambda unit testing for each function**		Eren Tarak	2
23.2	**Implement lambda integration test**		Eren Tarak	2
23.3	**Implement end to end testing**		Eren Tarak	2

2) Dependencies

Dependencies are provided in our Work Breakdown Structure (WBS) on Section 3.1 at Table 6.

3) Resource Requirements

3.1 Personnel

Table 7 Personnel Table

Role	**Number of Personnel**	**Time Allocation (in hours)**
Scrum Master	1	29 hours
Software Developer	4	450 hours each (15 per week * 30 week)

3.2 Computer Resources

- Database Servers: 1 x Dedicated Server
- Workstations: 4 x Developer Workstations
- Cloud Resources: AWS subscription for scalable computing and storage needs

3.3 Support Software

- Integrated Development Environment (IDE): Visual Studio 2022, VS Code, Arduino IDE
- Version Control: Git and GitHub
- Project Management: Trello
- Design and Brainstorm: Miro and Figma
- Communication: WhatsApp, Zoom

3.4 Hardware

- 4 mid-high range computers for each member.
- At least 1 phone for testing the software.
- 1 x Arduino Nano board
- 1 x ESP8266 WiFi module.
- 1 x MQ3 Air Quality Sensor.
- 1 x Light resistant resistor.

- 1 x soil moisture sensor.
- 1 x DHT11 Temperature/Humidity sensor.
- 1 x Breadboard (for testing)
- at least 40 x Jumper Cables

3.5 Facilities

- C building of Bilkent University for software development, testing, face-to-face team meetings when needed.

3.6 Travel

- There are no travel resources planned for the project.

3.7 Maintenance

- Maintenance for the hardware product. [1]

4) Cost Estimation, Net Present Value, Budget, and Resource Allocation

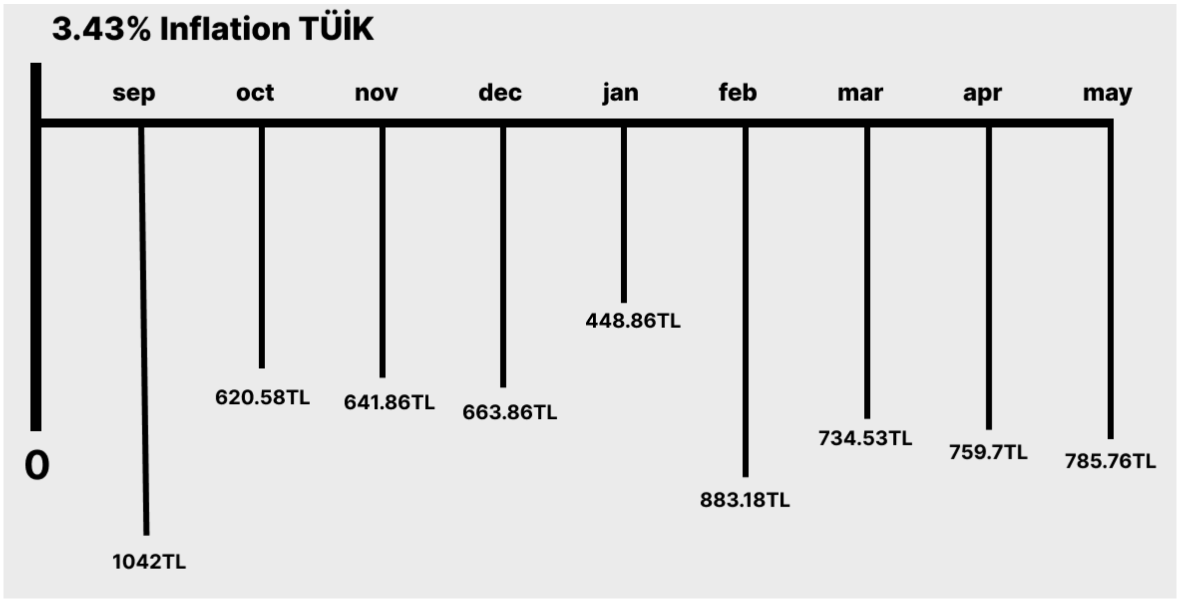

Figure 1 Cost Estimation for FloraVision

[1]GenAI tool: ChatGPT 3.5
Prompt: I'm developing an SPMP document for a software product. Can you develop an example for this part: "Resource Requirements
This subsection of the SPMP shall provide, as a function of time, estimates of the total resources required to complete the project. Numbers and types of personnel, computer time, support software, computer hardware, office and laboratory facilities, travel, and maintenance requirements for the project resources are typical resources that should be specified."
Rationale: To format the text. Have an idea

The predicted costs for each month include:

September:

Arduino UNO: 176TL

DHT11: 36TL

Soil Moisture: 28TL

Light Resisting Sensor: 3TL

MQ-2 Gas Sensor: 55TL

ESP8266 WiFi Module: 82TL

40 M-F Jumper Cable: 31TL

40 M-M Jumper Cable: 31TL

Consumed food and drinks: 200TL

Travel costs: 400TL

Total: 1042TL

October:

Consumed food and drinks: 206.86TL

Travel costs: 413.72TL

Total: 620.58TL

November:

Consumed food and drinks: 213.95TL

Travel costs: 427.91TL

Total: 641.86TL

December:

Consumed food and drinks: 221.28TL

Travel costs: 442.58TL

Total: 663,86TL

January:

Consumed food and drinks: 228.86TL

Travel Costs: 0TL

2 Pots: 160TL

Plant Seeds: 60TL

Total: 448,86TL

February:

Domain Name “floravision.me”: 173TL

Consumed food and drinks: 236,72TL

Travel Costs: 473,46

Total: 883.18TL

March:

Consumed food and drinks: 244.83TL

Travel Costs: 489.69TL

Total: 734.53TL

April:

Consumed food and drinks: 253.22TL

Travel Costs: 506,48TL

Total: 759.7TL

May:

Consumed food and drinks: 261.91TL

Travel Costs: 523.85TL

Total: 785.76TL

Net Present Value:

To calculate the Net Present Value we will use the formula $PV = FV/(1+r)_n$

So net present value is **5,582.06TL**

5) Schedule

In this part we provided our GANTT chart regarding the schedule for the project. Also we provided information regarding which jobs will be done on each increment and the team members who are responsible for each task.

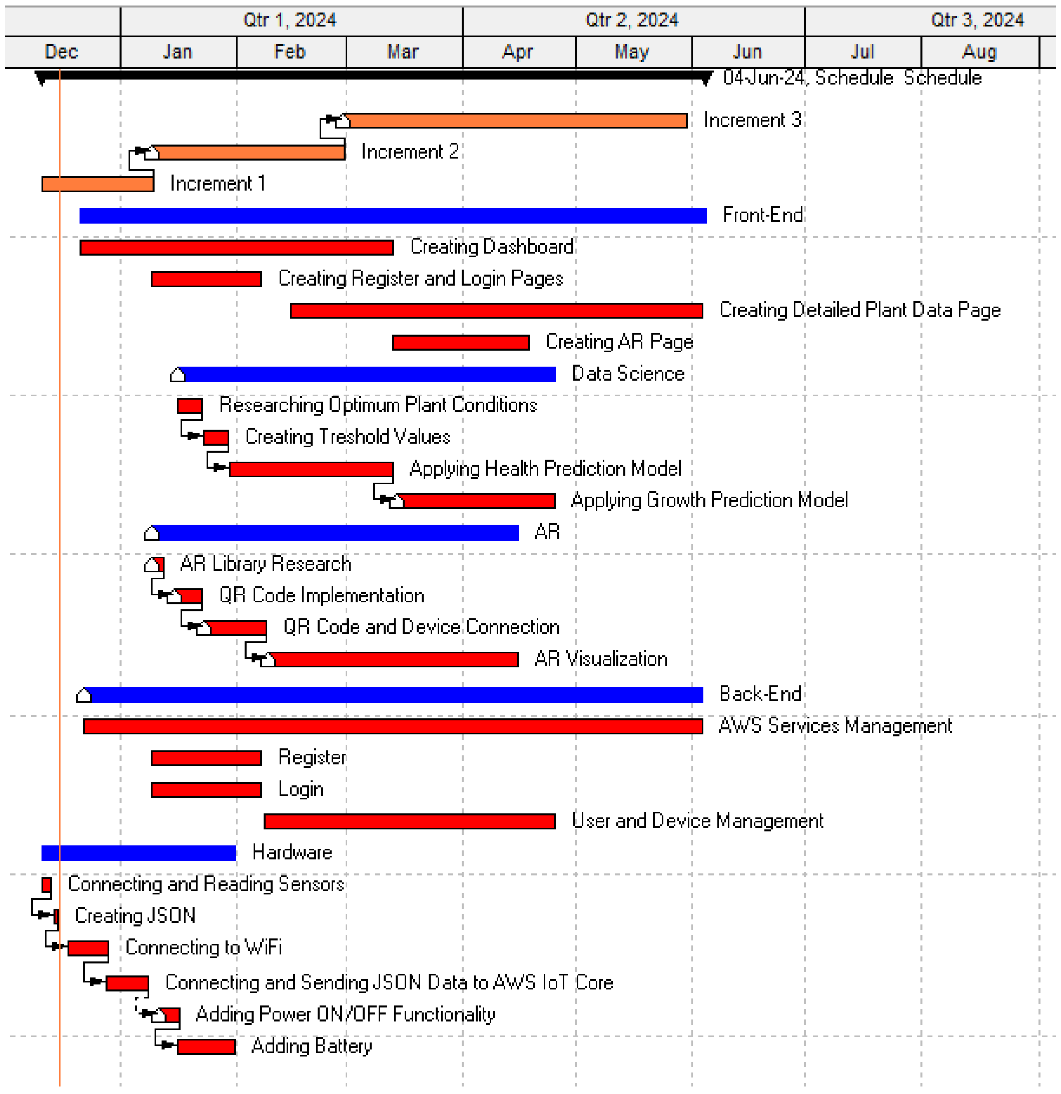

Figure 2 Schedule for FloraVision

In the first increment, we specified that we would finish some parts regarding the hardware, backend, and frontend parts. For hardware, we plan to finish the data creation, WiFi configuration, AWS data sending parts. For the frontend we want to display the real time values regarding the devices sensors, and we will get this data by connecting the backend to AWS IoT core. Additional parts for this increment can be the Login and Register functionalities for the Web Application.

For the first increment the job responsibilities are the following:

Hardware: Mert Köroğlu

Back-End: Eren Tarak, Eray Altay and Talha Rehman Abid

Front-End: Eren Tarak, Eray Altay and Talha Rehman Abid

In the second Increment, we will focus on finishing the hardware part by adding power on and power off features, battery supply and verification and testing requirements. For the front-end and backend, we want to finish on Login and Register functionalities if they are not finished on the 1st increment. Also, for the web, we want to add the Manage Device functionalities (Add Device, Remove Device) and show real-time charts regarding the device. We will add plant descriptions, device status (battery life), user credentials (root and normal user) functionalities also. For the Data Science part, we will agree on the plants that the system will accept, we will research the optimal values for these plants and we will start on creating a data science model for health and growth analysis. For the AR part, we will research the AR libraries and QR code reading functions. And we will try to create sending real-time values to AR from the web.

For the second increment the job responsibilities are the following:

Hardware: Mert Köroğlu

Front-End: Eren Tarak, Eray Altay

Back-End: Eren Tarak, Eray Altay

AR: Mert Köroğlu

Data Science: Talha Rehman Abid, Eray Altay

In the third increment, we will finish on health and growth prediction models, their implementation on the web application. We will also finish and test the AR functionalities. The web part will be able to show daily, weekly, and monthly statistics and health and growth analysis. And parts of the product will be tested to solve all mid to high tier defects.

For the third increment the job responsibilities are the following:

Hardware: Mert Köroğlu

Front-End: Eren Tarak, Eray Altay

Back-End: Eren Tarak, Eray Altay

AR: Mert Köroğlu

Data Science: Talha Rehman Abid, Eray Altay

4. Project Monitoring and Measuring

We will monitor our process using various tools. We will use Zoom to conduct meetings regarding the Scrum review and retro and while assigning new tasks to the team. We will also use Trello to see the process of each requirement whether the team member started the assigned task, did he complete the task and waiting for peer reviews will be seen using the Trello board. We will also use GitHub to monitor the commits of the project and, we will see the process of each product on the project.

We will use the User Story completion rate to measure the metrics. All requirements will be turned into User Stories and will be collected on the Scrum backlog. After completing each User Stories each user story will be collected again on Trello's "Done" backlog. There we will be seeing our progress regarding the project. The reason of collecting this metric is to ensure that everything is completed in time like we scheduled. With this metric we will understand if we are running late, or we are on time. According to our analysis, we will act to eliminate the problems.

We will also keep a system defects list and every defect's severity and priority rankings. This will help us to check up on the current defects on our product. Because we aim to deliver the product with no mid to high tier defects, it is important for us to check up on these defects and their completion metrics. For this we will also use Trello with a custom dashboard for testing activities only. Metric for these defects will be the "completed" and "remaining" defects for each level (low, mid, and high).

1. User Story Completion Rate:

- **Metric Definition:** This metric measures the percentage of completed user stories in the Scrum backlog.
- **Measurement:** Calculate the ratio of completed user stories to the total number of user stories. This can be tracked over each sprint or iteration.
- **Purpose:** To monitor progress and ensure that the team is delivering the expected functionality on time.

2. Defect Tracking:

- **Metric Definition:** Monitor the severity and priority of defects to ensure that mid to high-tier defects are minimized.
- **Measurements:**
    - **Completed Defects:** Count the number of defects that have been fixed and verified.
    - **Remaining Defects:** Count the number of defects yet to be addressed.
- **Purpose:** To deliver a product with a high level of quality by addressing and resolving defects in a timely manner.

3. Meeting Efficiency:

- **Metric Definition:** Evaluate the efficiency of Scrum review, retrospectives, and task assignment meetings conducted through Zoom.
- **Measurements:**
    - **Meeting Duration:** Measure the time spent in each meeting.
    - **Action Item Completion Rate:** Track the completion of action items assigned during meetings.
- **Purpose:** To ensure that meetings are productive and that action items are implemented effectively.

4. Source Code Commits:

- **Metric Definition:** Monitor the number and frequency of commits on GitHub.
- **Measurements:**
    - **Commit Frequency:** Track how often code is being committed.
    - **Code Churn**: Measure the lines of code added, modified, and deleted.
- **Purpose:** To assess the progress of development activities and identify any code integration issues early.

5. Trello Board Metrics:

- **Metric Definition:** Evaluate the progress on Trello boards for task assignment and tracking.
- **Measurements:**
    - **Task Start and Completion Dates:** Record when tasks are initiated and completed.

- Peer Review Status: Monitor the status of tasks awaiting peer review.

- Purpose: To visually track the flow of work, identify bottlenecks, and ensure tasks are progressing as planned.

6. Project Timeline Adherence:

- Metric Definition: Assess whether the project is adhering to the scheduled timeline.

- Measurement: Compare planned milestones and deadlines against actual project **progress.**

- Purpose: To identify any deviations from the project timeline and take corrective actions as needed.

7. Customer Satisfaction Surveys:

- Metric Definition: Gather feedback from end-users or stakeholders through surveys.

- Measurement: Analyze survey responses for overall satisfaction and specific feedback.

- Purpose: To ensure that the project meets user expectations and to identify areas for improvement.

8. Risk Mitigation Effectiveness:

- Metric Definition: Evaluate the effectiveness of risk mitigation strategies.

- Measurement: Assess the frequency and impact of identified risks and compare them to mitigation efforts.

- Purpose: To proactively manage and minimize project risks.[2]

[2] GenAI tool: ChatGPT 3.5
Prompt: We provided our Project Monitoring and Measurement paragraphs and said, "give more details regarding metric ve measurements for the best practice of spmp document for smart indoor gardening system project"
Rationale: To find an answer.

5. Product Verification and Validation

1. Verification and Validation Techniques:

Code Reviews:

Technique: Conduct systematic code reviews using pull requests on the version control system (Git).

Purpose: Ensure adherence to coding standards, identify bugs, and promote knowledge sharing among team members.

Verification Criteria: All pull requests must undergo review by at least one team member other than the author before merging.

Unit Testing:

Technique: Develop and execute unit tests using Lambda-Tester, Chai, and Mocha for backend logic.

Purpose: Verify the functionality of individual code units and catch early-stage defects.

Verification Criteria: All unit tests should pass, covering critical paths and edge cases.

Integration Testing:

Technique: Conduct integration tests for AWS CDK constructs and Lambda functions interactions.

Purpose: Validate that different components work as expected when combined.

Verification Criteria: All integration tests should pass, ensuring seamless communication between components.

System Testing:

Technique: Evaluate the entire system's functionality through end-to-end tests covering frontend, backend, and AWS infrastructure.

Purpose: Ensure the entire system meets specified requirements and works seamlessly as a whole.

Verification Criteria: All system tests should pass, covering user scenarios and system functionalities.

User Acceptance Testing (UAT):

Technique: Involve end-users in testing to validate that the software meets their needs and expectations.

Purpose: Ensure the final product aligns with user expectations.

Verification Criteria: Positive feedback from end-users and successful completion of UAT scenarios.

2. Verification and Validation Tools:

GitHub Actions:

Tool: GitHub Actions for continuous integration and automated testing.

Purpose: Automate the execution of unit tests, integration tests, and other verification processes.

Verification Criteria: Ensure successful builds and passing tests on each pull request.

Cypress:

Tool: Cypress for frontend end-to-end and integration testing for frontend.

Purpose: Implement automated tests to verify frontend functionalities.

Verification Criteria: All Cypress tests should pass, covering critical frontend scenarios.

Frontend Integration Testing:

Component Integration:

- Verify that individual components integrate correctly with each other.

- Ensure proper rendering and functionality when components are combined.

API Integration:

- Test the integration of frontend components with backend APIs.
- Validate that data is fetched, displayed, and updated accurately.

State Management:

- Test the interaction and synchronization of state management tools between components.
- Confirm that state changes trigger the expected updates in the UI.

Form Validation:

- Validate that form components perform proper validation.
- Ensure error messages are displayed for invalid inputs, and the form cannot be submitted with invalid data.

User Authentication:

- Test the integration of user authentication components.
- Confirm that users can log in, log out, and access authenticated features.

Navigation:

- Test navigation components, such as menus and links.
- Ensure users are directed to the correct pages when navigating through the application.

Error Handling:

- Verify that error messages are displayed appropriately when unexpected errors occur.
- Test error recovery mechanisms and ensure the application gracefully handles errors.

Frontend End to end Testing

User Scenarios:

- Test end-to-end user scenarios, covering the complete flow of common user journeys.
- Include scenarios such as user registration, login, and task completion.

Cross-Browser Compatibility:

- Validate that the application functions correctly across different browsers (e.g., Chrome, Firefox, Safari).
- Ensure consistent behavior and styling.

Responsive Design:

- Test the responsiveness of the application on various devices, including desktops, tablets, and mobile phones.
- Confirm that the layout adjusts appropriately to different screen sizes.

Performance:

- Test the application's performance by measuring page load times and responsiveness.
- Identify and address any bottlenecks affecting user experience.

Elastic APM and AWS CloudWatch:

Tool: elastic-apm-node for application performance monitoring and aws cloudwatch for infrastructure level monitoring

Purpose:

- Elastic APM for Application-Level Monitoring:
    - Use Elastic APM to monitor your Lambda functions at the application code level.
    - Utilize distributed tracing to understand interactions between various components of your application.
- AWS CloudWatch for Infrastructure Monitoring:
    - Set up CloudWatch Alarms to monitor infrastructure-level metrics, such as Lambda function errors, duration, and API Gateway errors.
    - Leverage CloudWatch for broader AWS resource monitoring, including DynamoDB tables, SNS topics, and more.
- Integration for Holistic Monitoring:
    - Establish a comprehensive monitoring strategy that integrates both Elastic APM and AWS CloudWatch.
    - Configure alerts and notifications in both systems to ensure prompt responses to different types of issues.

- Use CloudWatch Alarms to trigger notifications or actions based on infrastructure-level metrics.
- Use Elastic APM to identify and diagnose issues within the application code.

Verification Criteria: Ensure that performance metrics are within acceptable thresholds.

Verify that logs and metrics in AWS CloudWatch provide insights into application health and performance

Lambda-Tester, proxyquire, Chai, Mocha:

Tools: Lambda-Tester, proxyquire, Chai, and Mocha for unit testing Lambda functions.

Purpose: Implement and execute unit tests for backend logic.

Verification Criteria: All unit tests should pass, covering critical paths and edge cases.

3. Verification and Validation Schedule:

Iterative Development and Testing (Sprint-based):

Sprint 1 (Week 1-2):

- Code reviews for backend logic (Lambda functions and AWS CDK constructs).
- Unit testing for backend logic using Lambda-Tester, proxyquire, Chai, and Mocha.
- Initial integration tests for AWS CDK stacks and Lambda functions.

Sprint 2 (Week 3-4):

- Frontend unit testing using Cypress.
- System testing for critical user scenarios.
- Address and resolve issues identified during code reviews and testing.

Sprint 3 (Week 5-6):

- Full-system testing covering end-to-end scenarios.
- User acceptance testing (UAT) involving end-users.
- Address and resolve issues identified during UAT.

Sprint 4 (Week 7-8):

- Performance optimization and monitoring using Elastic APM and CloudWatch.
- Final verification and validation.
- Prepare for deployment.[3]

6. Software Development Environment

In this part we provided the software development environment for our project. We showed the Programming Languages that we will use, the Libraries, APIs, Databases, Cloud Services, and more tools that will help us while developing the project.

[3] GenAI tool: ChatGPT 3.5
Prompt: We provided our Development environment, our WBS, our Functional and Non-Functional Requirements and said "provided a detailed set of requirements and tasks related to the software development project, including hardware integration, frontend development, AWS infrastructure setup, Lambda functions, testing give us tools and schedule for testing and verification"
Rationale: To find an answer.

Table 8 Development Environment for FloraVision

Category	Tool/Technology	Version	Description/ Purpose
Programming Languages	JavaScript/TypeScript(Node.js)	18.x	Backend logic and AWS Lambda functions
Frontend Library	React.js	18.x	Building user interfaces for the web
Database	Amazon DynamoDB	2019.11.21	NoSQL database
Infrastructure Development and Deployment Kit	AWS Cloud Development Kit	2.115.0	provides the cdk command-line interface for easy integration and deployment of aws stacks, constructs and abstraction for infrastructure configuration
Cloud Service	AWS Cognito	SDK 6.3.6	Authentication and Management
Cloud Service	AWS IoT Core	-	Message Brokering
Cloud Service	AWS SNS	SDK 3.435.0	Publish/Subscribe Messaging for Iot Devices
Monitoring & Logging	elastic-apm-node	4.3.0	Application performance monitoring
Unit testing of the lambda functions	Lambda-Tester	4.0.1	A package that can help to run a lambda function locally.
Unit testing of the lambda functions	proxyquire	2.1.3	A package that can proxy the dependencies by overriding the functions inside it.
Unit testing of the lambda functions	Chai	4.3.10	Assertion library to verify if a given code is working correctly or not.
Unit testing of the	Mocha	10.2.0	For creating a test suite and running the test cases.

lambda functions			
Build Tool	Vite	4.5.0	Frontend Build Tool
Documentation	GitHub Markdown	-	markup language for documentation
Version Control	Git	-	Version control for source code.

7. Discussions

1. Limitations and Constraints

- Our only limitation was the scheduling. Because we are nearly at the end of the semester, there are many projects and homeworks we needed to deal with that affected our work on this document.

2. Health and Safety Issues

- We did not face any health and safety issues.

3. Legal Issues

- We did not face any legal issues.

4. Economic Issues and Constraints

- We did not face any economic issues or constraints

5. Sustainability

- We did not use any printed materials for this document, so we only used digital files to be more sustainable.

6. Ethical Issues

- We did not face any ethical issues.

7. Multidisciplinary Collaboration

- We did not use multidisciplinary collaboration.