CTIS411 Senior Project I

Software Design Description

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

6.1.2024

Change History

File Name	Document Type	Deliverable	Version	Submission Date
Team5_SDD_FR1	MS Word 2013	4	1	6.1.2024

Project Team

Team 5		
Name, Surname	**Student Id Number**	**e-mail**
Eray Altay	21803051	eray.altay@ug.bilkent.edu.tr
Talha Rehman Abid	22001505	rehman.abid@ug.bilkent.edu.tr
Mert Köroğlu	22003433	mert.koroglu@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	**Dr. Duygu Albayrak**
Github URL	**https://github.com/Smart-Indoor-Gardening-System**
WEB page	

Executive Summary

In this SDD document we provided High and Low Level Design and Architecture of our project. In the analysis model and planning part, we talked about which use cases will be implemented on the 1st, 2nd and 3rd increments of our software product. While identifying, we decided that for the 1st increment we will implement use cases “Connect to WiFi”, “Send Data to AWS” and “Read Sensors” for the hardware and we will also finish a part of the “View Plant Conditions” for the user. Important hardware functionalities like reading the sensor data, creating JSON from the data, connecting the WiFi and sending data to AWS IoT Core over MQTT topic will be implemented for the hardware and for the Web part we will implement real-time data displaying. In this part we also provided our latest Functional and Non-Functional requirements of our product. We also provided the APIs, libraries and services that will be used while creating the product. Most important services for this product include AWS services like Cognito, Lambda, IoT Core, DynamoDB Database, SNS (notification service) and Sagemaker. For the web part of the application we provided that we will use React and Chakra UI for building the front-end part of our software. For the High-Level Design of our system, we mentioned 3 architectures that we will use: Serverless (FaaS) Architecture, Event Driven Architecture and Modular Architecture. We provided a Block Diagram for showing the logical view, a Communication Diagram for process view and lastly a Deployment Diagram for the physical view of the software. Later we expressed how we will verify and validate the non-functional requirements to specify the design quality of our software. Lastly, we provided our pseudo codes and related diagrams to indicate the Low-Level Design of our system. These pseudo codes contain: “Arduino data reading and sending to ESP8266 WiFi module”, “WiFi Configuration and AWS IoT Core Data Sending”, “Sensor Data Processor Lambda Function Code”, and lastly” AWS CDK Stack implementation”.

Table of Contents

Page Number

Contents

List of Tables

Page Number

List of Figures

Page Number

Abbreviations

SDD	Software Design Description
AWS	Amazon Web Services
FaaS	Function as a Service

1. Scope

The Software Design Description (SDD) document provides an overview of the architecture and design used in the development of FloraVision. Objective of this document is to visualize and indicate the following:

- High Level Design
- Low Level Design
- Analysis Model and Planning

With High Level Design, we provided the architecture behind the FloraVision. To achieve this we used Block, Communication and Deployment Diagrams.

To understand the Low Level Design we provided pseudo codes and diagrams related to the codes to provide insight about the 1st increment functionalities.

2. Analysis Model & Planning

1. Functional Requirements

We provided our Use Case Diagram on the Figure 1

Figure 1 Use Case Diagram for FloraVision

In the 1st Increment, we will aim to finish the Use Cases regarding the Arduino User. These include, “Connect to WiFi”, “Send Data to AWS” and “Read Sensors”.

Also in the 1st increment, part of the use case “View Plant Conditions” also will be finished regarding the real-time data visualization.

In the 2nd increment, we aim to finish the “Login”, “Register” and “Manage Devices” use cases. The use case “Use AR” will also be started during this but the completion of it won’t be until the 3rd increment.

In the 3rd increment, we will finish the “Use AR” use case and finish the “View Plant Conditions” use case with adding health and growth analysis regarding the plant. Also, the Root User use cases “Manage User” and “Manage Device” will also be finished with the 3rd increment.

Functional Requirements:

Hardware

1. (FVF-1-1) The hardware kit shall use Arduino Nano as a microcontroller.
2. (FVF-1-2) The hardware kit shall use ESP8266 Wi-fi module to connect to Wi-Fi.
3. (FVF-1-3) The hardware kit shall use MQ-2 Gas/Smoke sensor to detect CO, CO2 to detect anomalies in the air quality.
4. (FVF-1-4) The hardware kit shall be connected to power using batteries.
5. (FVF-1-5) The hardware kit shall use the DHT11 Temperature/Humidity sensor.
6. (FVF-1-6) The hardware kit shall use a light sensitive resistor to detect the presence of light and its intensity.
7. (FVF-1-7) The hardware uses a soil moisture sensor to detect the plant’s soil moisture.
8. (FVF-1-8) The system shall let users power off the device.
9. (FVF-1-9) The system shall let users power on the device.

Sensor Data Reading

10. (FVF-2-1) The hardware part of the system shall read light sensor data through the A1 port on Arduino Nano.
11. (FVF-2-2) The hardware part of the system shall read humidity/temperature sensor data through the digital 9 port on Arduino Nano.
12. (FVF-2-3) The hardware part of the system shall read soil moisture sensor data through the A3 port on Arduino Nano.
13. (FVF-2-3) The hardware part of the system shall read air quality sensor data through the A2 port on Arduino Nano.

Data Transmission

14. (FVF-3-1) The hardware part of the system shall connect to Wi-Fi through digital ports on Arduino Nano. (ref. (FVF-1-6) [1])
15. (FVF-3-2) The hardware part of the system shall create a Json format sensor data every 5 seconds.
16. (FVF-3-3) The hardware part of the system shall send sensor data to AWS IOT core managed message broker service via MQTT. (ref. (FVF-1-5) [1])
17. (FVF-3-4) The hardware part of the system shall turn on/off if the button is pressed for 3 seconds.
18. (FVF-3-5) The hardware part of the system shall detect Wi-Fi if a button is pressed and released.
19. (FVF-3-6) The system shall let users connect to Wi-Fi through their phone dynamically using the WifiManager library.

Hardware-web integration

20. (FVF-4-1) The web application shall connect to hardware kits through the devices tab.
21. (FVF-4-2) The web application shall send the user to their plant's insights if the hardware kit is pressed in the devices tab.
22. (FVF-4-3) The web application shall send a GET request to AWS every 5 seconds to refresh plant data.

Hardware Kit Authentication

23. (FVF-5-1) The user shall connect to hardware using the password and id provided with the device.

24. (FVF-5-2) The system shall make the first connected user Root user.

25. (FVF-5-3) The system shall make every other user Normal user.

Web Authentication

26. (FVF-6-1) For the web application, the system shall implement user registration and login using AWS Cognito including Google, Facebook providers and email password authentication options.

Registration Page

27. (FVF-6-2) On the registration page, the system shall ask inputs regarding email, username, and password.
28. (FVF-6-3) On the registration page, the system shall give an error if the given email already exists in the AWS Cognito user pool.
29. (FVF-6-4) On the registration page if the user selected the email/password authentication, the system shall register the user if all the fields are entered regarding the rules of the registration.
30. (FVF-6-5) On the registration page if the user selected the email/password authentication, the system shall only accept passwords which contain at least one number, one symbol, one capital character and at least six characters.
31. (FVF-6-6) On the registration page if the user selected the email/password authentication, the system shall give an error if the password rules are not met after pressing the register button.

Login Page

32. (FVF-6-6) On the login page, if the user clicks on the login button, the system shall check the given email and password. (ref. (FVF-1-8) [1])

33. (FVF-6-6) On the login page, the system shall display a button called “sign up” at the bottom of the login page to navigate user to the registration page.
34. (FVF-6-7) In the login page the system shall give an option to recover existing user password if it's forgotten via a button called “forgot password”.
35. (FVF-6-8) In the login page the system shall send a confirmation code email to the user if the user clicked on the “forgot password” button. (ref. FVF-6-7).
36. (FVF-6-9) In the login page the system shall provide an input field asking for confirmation code when an email is sent to the user after they clicked on the “forgot password” button. (ref. FVF-6-7).
37. (FVF-6-10) In the login page the system shall navigate the user to a new password page if their code matches with the confirmation code in the email. (ref. FVF-6-7).
38. (FVF-6-11) In the login page the system shall navigate the user to the login page if their code does not match with the confirmation code in the email. (ref. FVF-6-7).
39. (FVF-6-12) In the login page the system shall check the new password in the new password page (ref. FVF-6-7).
40. (FVF-6-13) Web application shall keep the session of users to remember user credentials.

Failure Scenarios

41. (FVF-6-14) The system shall redirect the user to the login page if the user token expires.
42. (FVF-6-15) The system shall log out the user if the user clicks the log out button on the top navigation bar on the web application interface.
43. (FVF-6-16) If the login credentials do not match, the system will provide an error notification in the form of a text, indicating that either the password or email is incorrect.

Success Scenario

44. (FVF-6-17) Once the user is authenticated, the system shall enable the user to gain access to the system's other functionalities.

Hardware Setup on W*eb*

45. (FVF-7-1) The system shall display the connected hardware devices on the devices page in the web application.
46. (FVF-7-2) The system shall display general statistics such as temperature, soil moisture, light density and co2 density of the plant on the devices tab for each device.
47. (FVF-7-3) The system shall display a button to add new hardware devices to the application.
48. (FVF-7-4) The system shall enable the users to navigate to the selected device's dashboard if they click on the device.
49. (FVF-7-5) The system shall enable the user to remove the selected device if they press on the "-" button.
50. (FVF-7-6) The system shall provide battery status of each connected device on the devices page on the web.

Dashboard

51. (FVF-7-8) The system shall display the selected device's real time sensor metrics including temperature, light density, soil moisture, humidity, and air quality on the "today" tab on the dashboard page. (ref. (FVF-1-7), (FVF-1-13) [1])
52. (FVF-7-9) The system shall display tabs at the top of the dashboard including daily, weekly, monthly, and yearly displaying options. (ref. (FVF-1-11) [1])
53. (FVF-7-10) The system shall display today's charts about the sensor data from 00.00 to 23.59 right below the real-time data.
54. (FVF-7-11) The system shall display the plant species name as a link at the top of the dashboard to navigate the user to relevant plant species description and characteristics.

55. (FVF-7-12) The system shall enable users to customize their dashboard via an option menu consisting of checkboxes to select the charts out of weekly, monthly, and yearly options for displaying.
56. (FVF-7-13) The system shall navigate the user to a more detailed chart about the selected metric when the user clicks the chart.

On the detailed page of the selected metric

57. (FVF-7-14) The system shall display a mixed chart combining line chart and bar plot.
58. (FVF-7-15) The system shall display the deviations from the plants corresponding optimum environmental value tagged with labels such as high, moderate, low gaps from the optimum value.
59. (FVF-7-16) The system shall present the average value of the relevant metric on a daily, monthly, and yearly basis, offering users comprehensive insights into the overall trends and patterns.

Notifications

60. (FVF-8-1) The system shall have a notification user interface modal to enable the user to track notifications.
61. (FVF-8-2) If the user clicks on the notification bell icon at the right of the dashboard navigation at the top of the page, The system shall open a notification modal including only several latest notifications with their time such as 2 minutes ago.
62. (FVF-8-3) If the user clicks on the “see more” link at the bottom of the notification modal, The system shall navigate the user notifications page where all notifications are listed.
63. (FVF-8-4) The system shall enable the user to mark all notifications as read when the user clicks on a button called “mark all as read”.
64. (FVF-8-5) The system shall ensure that notifications are informative, concise, and relevant, providing users with a clear understanding of the event or update. (ref. (FVF-1-12), (FVF-1-16) [1])
65. (FVF-8-6) The system shall have real-time web push notifications enabled as the default communication method for notifying users.

Additionally, users shall have the option to customize their notification preferences by adding email or SMS forms according to their choice.

66. (FVF-8-7) The system shall utilize AWS SNS (Simple Notification Service) to manage and deliver notifications efficiently. This service will be responsible for handling real-time web push notifications, emails, and SMS messages.

Augmented Reality

67. (FVF-9-1) The system shall utilize a marker based augmented reality for real time data visualization for the chrome users.
68. (FVF-9-2) The system shall display a fixed positioned button on the dashboard page to enable the user to start the AR visualization.
69. (FVF-9-3) The System shall recognize predefined markers associated with environmental data points.
70. (FVF-9-4) Upon detecting these markers through the device's camera, the system shall overlay real-time environmental metrics, including light intensity, soil moisture, and temperature, onto the markers in the AR visualization.
71. (FVF-9-5) The system shall display AR visualization content as 2d charts and metrics along with related icons.
72. (FVF-9-6) The system shall support the visualization of historical environmental data in AR, allowing users to toggle between real-time and historical view.
73. (FVF-9-7) If the user device or browser do not support AR, the system shall display an alert dialog indicating that the user device does not support AR and inform the user regarding supported browsers along with their versions and supported devices when the user clicks on the augmented reality button.
74. (FVF-9-8) If the user scans another marker, the system shall discard the previous 2d charts.

Predictions Data Science and Analysis

75. (FVF-10-1) The system shall integrate a health prediction model, leveraging scientific threshold values of relevant species and machine learning algorithms. This model should generate short-term and medium-term predictions, forecasting potential health issues based on environmental conditions before they impact the plants. (ref. (FVF-1-1), (FVF-1-3) [1])

Root User Managing Devices:

76. (FVF-11-1) The system shall enable root users to change device's name.
77. (FVF-11-2) The system shall enable root users to change device's password.

Root User Managing Users:

78. (FVF-11-3) The system shall enable root users to remove other users.
79. (FVF-11-4) The system shall enable root users to make another one root user.

2. Non-Functional Requirements

1. Usability requirements

- (FVN-1-1) The web application shall be compatible with major browsers (Chrome, Firefox, Safari) to provide a consistent experience across different platforms except the AR features.

2. Performance requirements

- (FVN-2-1) The system should handle a minimum of 10 simultaneous users. AWS Lambda, DynamoDB, and SNS are designed to scale horizontally to accommodate varying workloads.
- (FVN-2-2) The system shall hold the device's data history for at least 1 month.

3. Software system attributes

a) Reliability

- (FVN-3-1) The system shall be reliable by providing health and growth insights about the plant using reliable academic sources for creating reliable algorithms.
- (FVN-3-2) The system shall not include mid and high priority level failures/defects at launch.

b) Availability

- (FVN-4-1) The system shall have real time administration by sending notifications to system administrators when in the event of a system failure.

c) Security

- (FVN-5-1) The system shall log every system failure.

d) Maintainability

We did not identify any maintainability related non-functional requirements.

e) Portability

- (FVN-6-1) The web application shall enable users to log in from other devices that are compatible to use the web product.
- (FVN-6-2) The code for each hardware device shall be only for that device only. This is because every hardware device will have its own id for data.

4. Constraints

- (FVN-7-1) The product shall adhere to privacy laws and rules about data sharing, keeping, and handling.
- (FVN-7-2) The system shall only provide English as a language for all products.

5. Error Handling Requirements

- (FVN-8-1) If a user provides the wrong email or password while logging in, the system shall want the user.
- (FVN-8-2) If a user provides a non-existing or already in use email while registering, the system warns the user.
- (FVN-8-3) If a user provides a password with non-required type, the system shall warn the user.
- (FVN-8-4) If the AWS Core receives an incorrectly formatted message from the hardware product, the AWS Core shall ignore the message and provide a log about the ignored message.
- (FVN-8-5) If a user tries to access unauthorized locations in the web application, the system shall warn and redirect the user to the login page.
- (FVN-8-6) If a user enters and provides two different passwords while registering, the system shall warn the user.

3. Software Increments

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.

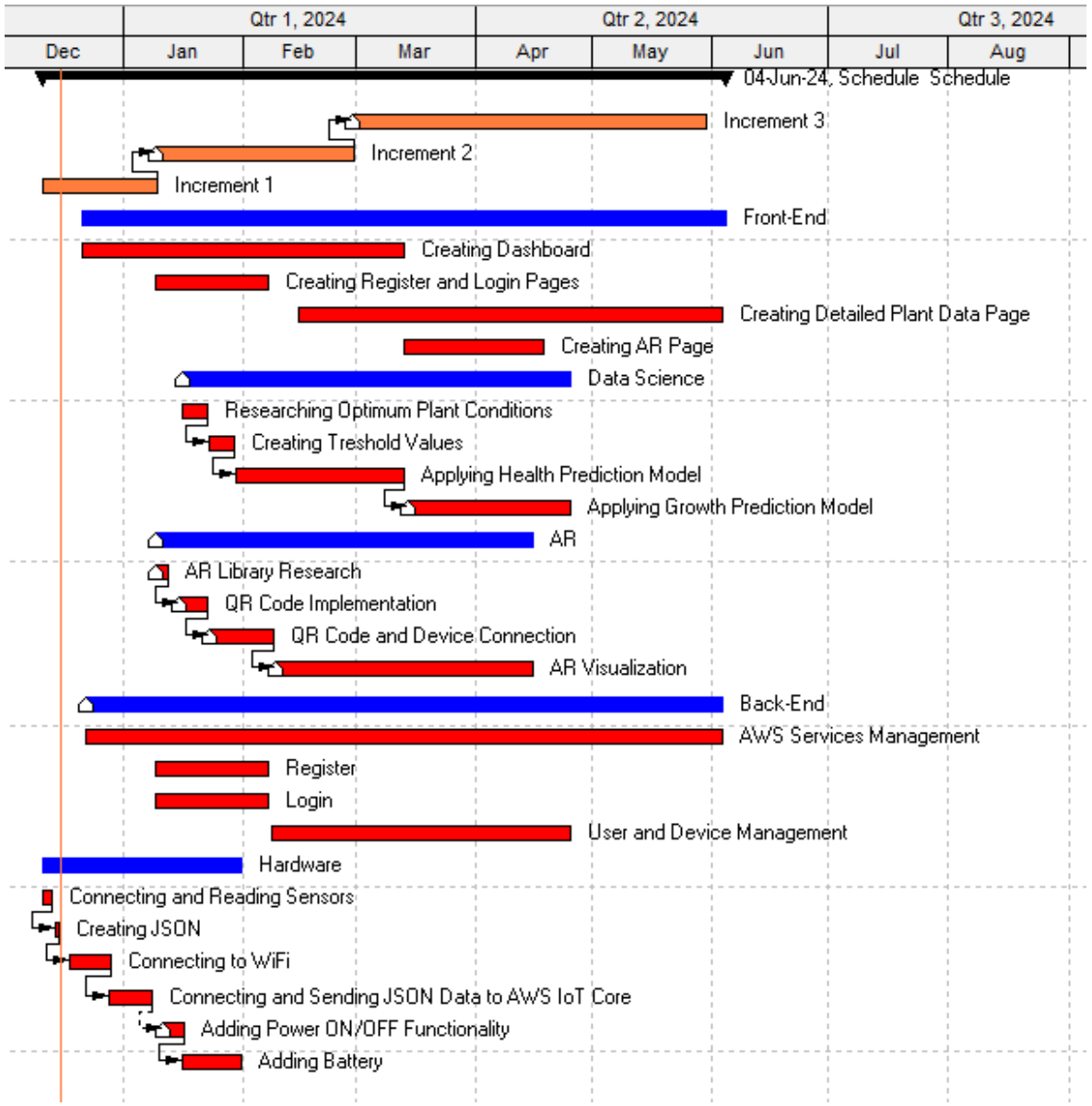


Figure 2 GANTT Chart for FloraVision

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 device's sensors, and we will get this data by connecting the backend to AWS IoT core.

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. 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. Frameworks, Libraries, Services, Databases and APIs

We will use many libraries, services, databases, and APIs during the development of the project.

AWS Services:

We will use many AWS services and their software development kits including AWS Cognito, AWS IoT Core,AWS KMS,AWS Amplify, AWS EC2, AWS S3,, AWS Lambda and AWS DynamoDB.

AWS Cognito:

Will be used for user authorization implementations. Will be used on the 2nd increment for “Login” and “Register” Use Cases.

AWS IoT Core:

Will be used for listening to hardware sensor data. Will be used in 1st increment with Arduino Use Cases.

AWS DynamoDB:

Our Database is AWS DynamoDB and will be used in all increments.

AWS SQS:

Utilizing AWS SQS for decoupled inter-Lambda communication enables asynchronous task handling, ensuring scalability and reliability. Tasks like device password verification (secured by AWS KMS) and device addition are seamlessly managed.

AWS KMS:

In our project, AWS Key Management Service (KMS) is used to ensure secure handling of sensitive data, such as device passwords. The verifyDevicePassword Lambda function decrypts stored passwords using KMS, ensuring secure verification. Additionally, AWS KMS keyring

configuration provides flexibility for decrypting data across multiple AWS Regions. This enhances security and scalability within our smart indoor gardening system.

AWS Amplify:

For deploying and hosting our React Vite frontend, we utilized AWS Amplify, enabling continuous delivery for seamless updates and scalability.

AWS S3:

AWS Lambda:

Will be used for creating functions between other AWS Services. It will be used in all increments. It will be used in the 1st increment while creating AWS IoT Core and AWS DynamoDB connection.

AWS API Gateway:

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.

Libraries:

AWS:

AWS CDK:

The AWS CDK lets you build reliable, scalable, cost-effective applications in the cloud with the considerable expressive power of a programming language. This approach yields many benefits, including:

- Build with high-level constructs that automatically provide sensible, secure defaults for your AWS resources, defining more infrastructure with less code.
- Put the infrastructure, application code, and configuration all in one place, ensuring that at every milestone you have a complete, cloud-deployable system.

@aws-sdk/client-dynamodb:

AWS SDK for JavaScript DynamoDB Client for Node.js

Hardware:

WiFi Manager:

WiFi Manager will be used during the dynamic WiFi configuration of the Arduino Device. It will be used in the 1st increment.

DHT:

Will be used for reading DHT humidity/temperature sensor. Will be implemented with the 1st increment.

SoftwareSerial:

Will be used for creating a Serial Communication between the Arduino and the ESP8266 WiFi module to transfer the JSON of sensor data. Will be implemented with the 1st increment.

ArduinoJson:

Used for creating the JSON of sensor data generated by Arduino. Will be implemented with the 1st increment.

MQUnifiedSensor:

Will be used for reading and configuring MQ2 sensor readings. Will be implemented with the 1st increment.

ArduinoMQTTClient:

Will be used for publishing the sensor JSON data to AWS IoT Core over the MQTT. Will be implemented with the 1st increment.

PubSubClient:

Will be used for handling server operations between the ESP8266 module and AWS IoT Core. Will be implemented with the 1st increment.

WiFiClientSecure:

Will be used with the PubSubClient. Will be implemented with the 1st increment.

Frontend:

React.js:

React is a JavaScript library for building more reusable user interfaces. Will be used in all increments.

ApexCharts.js:

Will be used for creating charts for the frontend part of the system. Will be used when implementing 2nd and 3rd increments.

Zustand:

Manages states for user authentication, device data, notifications and AR mode and data visualization. Will be used during the 2nd and 3rd increment.

Chakra UI:

Will be used for creating more user-friendly UI designs in the front-end. Will be used in 2nd and 3rd increments.

Database:

AWS DynamoDB

3. High-Level (Architecture) Design

1. Selected Architecture

Serverless (FaaS) Architecture:

- **Justification:**
  - **Usability (FVN-1-1):** Serverless architecture, with AWS Lambda as a key component, allows for easy scalability and compatibility across different platforms, ensuring a consistent user experience.
  - **Performance (FVN-2-1, FVN-2-2):** AWS Lambda is designed to scale horizontally to handle varying workloads, making it suitable for accommodating at least 10 simultaneous users and retaining device data history for at least 1 month.
  - **Availability (FVN-4-1):** Serverless architectures, when properly configured, offer high availability by distributing functions across multiple servers, reducing the risk of single points of failure.
  - **Portability (FVN-6-1):** Serverless architectures are inherently portable as they abstract away the underlying infrastructure, allowing users to access the application from different devices.
  - **Error Handling (FVN-8):** Serverless architectures support event-driven error handling, and AWS Lambda can log errors efficiently.
- **Advantages:**
  - Automatic scalability to handle varying workloads.
  - Reduced operational overhead as cloud providers manage server infrastructure.
  - Cost-effective, as you only pay for the actual compute time.
- **Disadvantages:**
  - Cold start latency might be a consideration for certain use cases.
  - Limited execution time for each function.

Event-driven Architecture:

- **Justification:**
  - **Performance (FVN-2-2):** Event-driven architectures, combined with AWS IoT Core, Lambda and SNS, support efficient handling of events and notifications. Sensor data is continuously generated, published via

MQTT to AWS IoT Core, where AWS Lambda processes and stores the data in DynamoDB with a time-based retention policy, enabling efficient querying of historical data for at least 1 month.

- **Availability (FVN-4-1):** Event-driven architectures can provide real-time administration through notifications in the event of system failure.

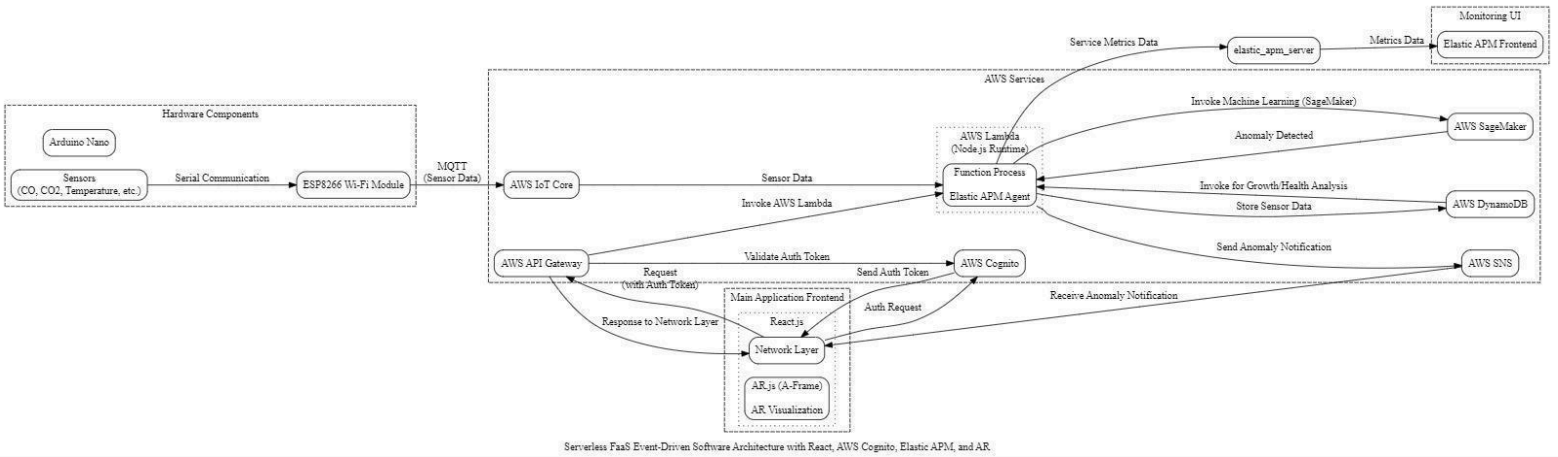
- **Advantages:**
  - Loose coupling between components, promoting independence and scalability.
  - Efficient handling of asynchronous events and notifications.
- **Disadvantages:**
  - Increased complexity in managing event flows.
  - Proper event sequencing and handling are crucial.

Modular Architecture:

When the system is designed with modularity, each module can be individually optimized for reliability. This includes the use of well-researched algorithms from academic sources to perform specific tasks within each module.

2. Logical View

We provided our block diagram for our software's architecture on Figure 3.



We provided our EER Diagram regarding the product's database in Figure 4.

Figure 4 EER Diagram for FloraVision

3. Process View

Our UML Communication Diagram is provided in Figure 5.

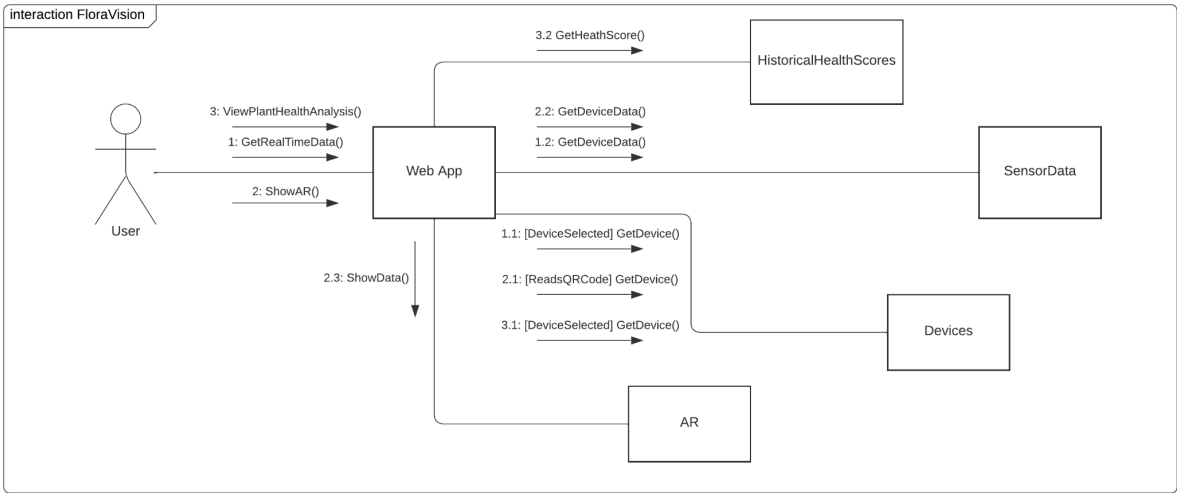
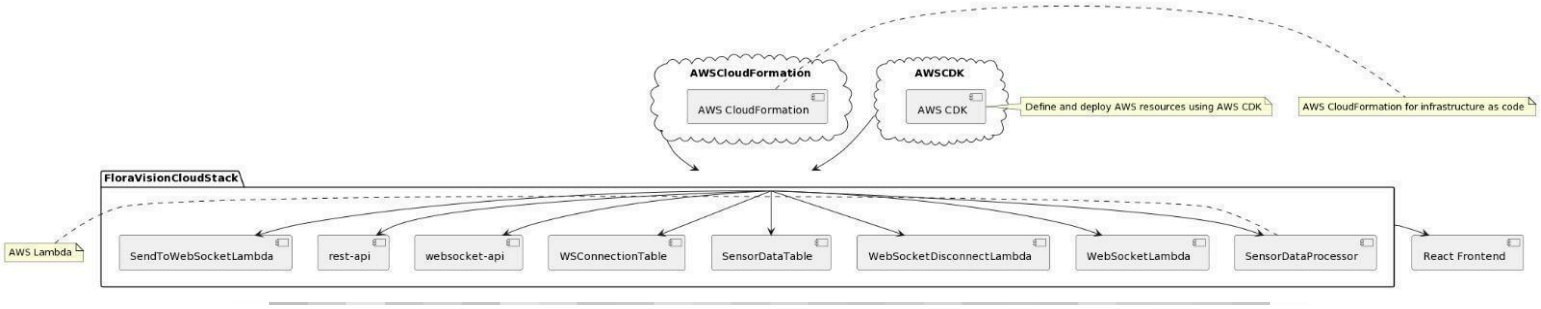


Figure 5 UML Communication Diagram for FloraVision

4. Physical View

Our UML Deployment Diagram is provided in Figure 6.



5. Design Quality

Usability Requirements:

How your design will meet: Ensure the web application is developed using cross-browser compatible technologies and conduct testing on major browsers.

How to verify & validate: Perform usability testing with real users on different browsers to ensure a consistent user experience.

Performance Requirements:

How your design will meet: Utilize AWS Lambda, DynamoDB, and SNS for horizontal scaling. Implement data retention policies to store device data history for at least 1 month.

How to verify & validate: Conduct performance testing with more than 10 simultaneous users. Monitor system performance and verify data retention capabilities.

Software System Attributes:

a) Reliability:

How your design will meet: Use reliable academic sources for plant insights algorithms. Prioritize testing to eliminate mid and high priority level failures.

How to verify & validate: Validate algorithms against academic sources. Perform thorough testing to identify and fix potential failures before launch.

b) Availability:

How your design will meet: Implement real-time administration for system failure notifications.

How to verify & validate: Test the system's response to simulated failures. Ensure administrators receive timely notifications during system downtime.

c) Security:

How your design will meet: Log every system failure for security monitoring.

How to verify & validate: Review system logs regularly. Simulate and monitor system failures to ensure proper logging.

d) Maintainability:

How your design will meet: No specific requirements identified.

How to verify & validate: Not applicable.

e) Portability:

How your design will meet: Enable users to log in from compatible devices. Ensure device-specific code for hardware devices.

How to verify & validate: Test login functionality on various devices. Validate device-specific code on corresponding hardware devices.

Constraints:

How your design will meet: Adhere to privacy laws, and offer only English language support.

How to verify & validate: Regularly update the system to comply with changing privacy laws. Ensure language settings are restricted to English.

Error Handling Requirements:

How your design will meet: Implement various error handling mechanisms as specified.

How to verify & validate: Test each error scenario thoroughly. Verify that the system responds appropriately and provides warnings or logs as required.[1]

[1] GenAI tool: ChatGPT 3.5
Prompt: Entered our Non-Functional requirements and said “Answer the following: "how your design will meet, and how you are going to verify & validate them.””
Rationale: To Find an Answer.

4. Low Level Design

For the Low-Level Design of the product, we included pseudo codes and related UML Activity Diagrams in this section.

Data Readings with Arduino and Sending it to ESP8266 WiFi Module:

Pseudo code:

// Initialize Libraries

#include "DHT.h"

#include <SoftwareSerial.h>

#include "ArduinoJson.h"

#include <MQUnifiedsensor.h>

Define variables

Create variables for HT, MQ, SoftwareSerial and JSONData.

//setup function will only run once during the initial run of the software.

void setup(){

    Start serial communication for debugging

    Initialize the ESP8266 module communication

    Define digital and analog input and output pins.

    Configure and initialize the gas sensor MQ2

    Calibrate the gas sensor MQ2// Initialize the DHT sensor

    Initial delay

}

// loops continuously while the Arduino is powered on.

void loop(){

    Clear the JSON document and string

    Update data from the gas sensor MQ2

Read sensor values and store them in the JSON document

Serialize the JSON document to a string

Send the JSON string to the ESP8266 module

Print the JSON string to the serial monitor

Delay before the next iteration

}

Activity Diagram:

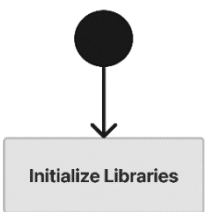


Figure 7 Activity Diagram for Code 1

WiFi Configuration and AWS IoT Core Data Sending:

Pseudo code:

```
// Include necessary libraries
#include <WiFiManager.h>
#include <ArduinoMqttClient.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>

// Define AWS IoT topics
AWS_IOT_PUBLISH_TOPIC = "FloraVision/pub"
AWS_IOT_SUBSCRIBE_TOPIC = "FloraVision/sub"

// Device-specific constants
id = 1
TIME_ZONE = 5

// WiFi and MQTT clients
Initialize Secure WiFiClient and MQTT Client

// Variables for sensor data
Initialize arduinoData, h, t, lastMillis, previousMillis, and interval

// Callback for received MQTT messages
```

Define messageReceived function:

Print received messages

void NTPConnect(){

Define NTPConnect function:

Print setting time using SNTP

Configure time using SNTP with a wait

Print current time after synchronization

}

void setup(){

Define setup function:

Set WiFi mode to STA

Initialize Serial communication

Initialize WiFiManager

Attempt WiFi connection

If connection successful:

Print connected WiFi SSID

Connect to AWS IoT

Set up secure connection

Subscribe to AWS IoT topic

Print AWS IoT connected message

}

void loop(){

Define loop function:

```
    Check WiFi connection status
    If connected:
        Read data from Serial
        Publish data to AWS IoT Core
}
```

Activity Diagram:

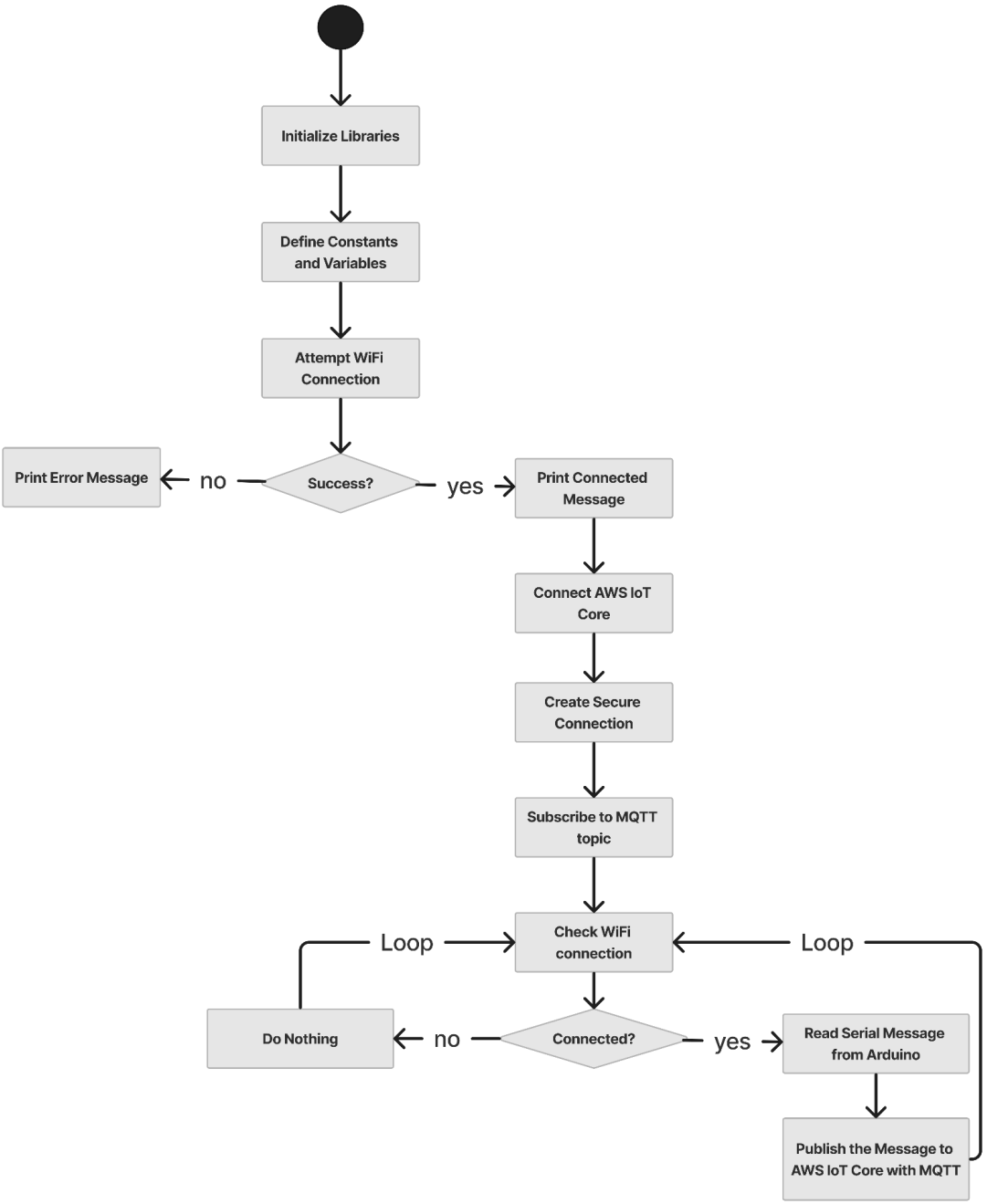


Figure 8 Activity Diagram for Code 2

AWS CDK Stack:

Pseudo Code:

AWS CDK Stack (Pseudo Code):// AWS CDK Stack - FloraVisionCloudStack

Create new AWS CDK Stack named FloraVisionCloudStack

Create DynamoDB table named SensorDataTable

Define partition key 'pk' of type STRING

Define sort key 'sk' of type STRING

Set removal policy to DESTROY (not for production)

Set billing mode to PAY_PER_REQUEST

Create Lambda function named SensorDataProcessor

Use Node.js runtime version 18.x

Entry point is 'lambda/index.ts'

Handler function is 'handler'

Set environment variable TABLE_NAME to SensorDataTable.tableName

Grant Lambda function permission to read and write data to SensorDataTable

Sensor Data Processor Lambda Function Code:

Pseudo Code:

SensorDataProcessor Lambda Function Code (Pseudo Code):// Lambda Function - SensorDataProcessor

Lambda function handler(event, context):

Try:

Extract sensor data from event

Generate a unique identifier (UUID)

Create DynamoDB client

Create SensorData object from extracted data

Put data into DynamoDB table:

Set partition key (pk) to "POST#"+UUID

Set sort key (sk) to current timestamp

Set other attributes using SensorData object

Catch any errors:

Log and throw the error

5. Discussions

1. Limitations and Constraints

- We did not face any limitations or constraints while developing this document.

2. Health and Safety Issues

- We did not face any health and safety issues while developing this document.

3. Legal Issues

- We did not face any legal issues while developing this document.

4. Economic Issues and Constraints

- We did not face any economic issues while developing this document.

5. Sustainability

- Only a digital copy of this document is used to avoid the usage of paper.

6. Ethical Issues

- We did not face any ethical issues while developing this document.

7. Multidisciplinary Collaboration

- We did not use any multidisciplinary collaboration.